In presenting this dissertation in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission for extensive copying of my thesis for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this dissertation for financial gain shall not be allowed without my written permission.

Signature _____

Date _____

A Machine Learning Approach to Fuel Load Optimization in the Advanced Test Reactor

by

Brittany Jean Grayson

A dissertation

submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy in the Department of Nuclear Engineering

Idaho State University

Spring 2023

To the Graduate Faculty: The members of the committee appointed to examine the dissertation of Brittany Jean Grayson find it satisfactory and recommend that it be accepted.

Leslie Kerby, PhD,
Major Advisor

Chad Pope, PhD,
Committee Member

Mary Lou Dunzik-Gougar, PhD,
Committee Member

Joshua Peterson-Droogh, PhD,
Committee Member

David Beard, PhD,
Graduate Faculty Representative

Acknowledgements

I would like to acknowledge my advisor, Dr. Kerby, for her support and patience through this research.

This research made use of the resources of the High Performance Computing Center at Idaho National Laboratory, which is supported by the Office of Nuclear Energy of the U.S. Department of Energy and the Nuclear Science User Facilities under Contract No. DE-AC07-05ID14517.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ABC	Artificial Bee Colony
ACDPF	Adaptively Constrained Discontinuous Penalty Function
ANN	Artificial Neural Network
API	Application Programming Interface
ATR	Advanced Test Reactor
BANEC	Bat Algorithm Nodal Expansion Code
BBO	Biogeography-Based Optimization
BE	Binary Exchange
BOC	Beginning of Cycle
BOL	Beginning of Life
BWR	Boiling Water Reactor
C	Center
CA	Cellular Automata
CIC	Core Internals Changeout
CMCDT	Common Monte Carlo Design Tool
CPA	Core Physics Analysis
DGA	Distributed Genetic Algorithm
DNBR	Departure from Nucleate Boiling Ratio
DS	Direct Search
E	East
EDA	Exploratory Data Analysis
EFPD	Effective Full Power Days
EOC	End of Cycle
EOL	End of Life
FEID	Fuel Element ID
FORMOSA	Fuel Optimization for Reloads: Multiple Objectives for Simulated Annealing
g	Grams
GA	Genetic Algorithm
GPU	Graphic Processor Unit
GS	Greedy Search
GSA	Gravitational Search Algorithm

HNN	Hopfield Neural Networks
HPC	High Performance Computing
IAEA	International Atomic Energy Agency
INL	Idaho National Laboratory
KNN	K-Nearest Neighbors
LEU	Low Enriched Uranium
LP	Loading Pattern
LWR	Light Water Reactor
MAE	Mean Absolute Error
MAPE	Mean Absolute Percent Error
MOSA	Multi-Objective Simulated Annealing
MPNN	Multi-Layer Perceptron Neural Network
MPRR	Multipurpose Research Reactor
MSE	Mean Square Error
MTR	Material Test Reactor
MWd	Megawatt-Day
N	North
NE	Northeast
NW	Northwest
OSCC	Outer Shim Control Cylinder
PALM	Powered Axial Locator Mechanism
PSO	Particle Swarm Optimization
PUMA	Physics Unified Modeling and Analysis
PWR	Pressurized Water Reactor
QSA	Quasi-Simulated Annealing
ReLU	Rectified Linear Unit
RSME	Root Mean Square Error
RSS	Residual Sum of Squares
S	South
SA	Simulated Annealing
SE	Southeast
SMAPE	Symmetric Mean Absolute Percentage Error
SW	Southwest

TS	Tabu Search
TSS	Total Sum of Squares
VVER	Water-Water Energetic Reactor
W	West
YGN4	Yonggwang Nuclear Unit 4

A Machine Learning Approach to Fuel Load Optimization in the Advanced Test Reactor

Dissertation Abstract—Idaho State University

A fuel load optimization process for the ATR was developed using machine learning. Cycle data was collected from engineering documents for ATR operating cycles 46A through 169A. Cycles 165A through 169A were then held back to be used to test the machine learning process. The total of the training/testing dataset was 10,400 inputs over 260 ATR cycles. KNN-imputation was used in the instance that a missing value was present in the dataset. Once the data was fully collected, exploratory data analysis was completed to understand any trends in the dataset. Three regression algorithms were considered for the fuel load optimization: linear regression, random forest regression, and neural networks. Linear regression performed the worst overall and could not account for fuel element position in the dataset causing all models to be underfit. Random forest performed best in terms of R^2 value but contained severe spikes when incorrect. Neural networks were found to be the best fit due to predicting the closest to the as run burnup data. Both the feature selection and train-test split values were carefully considered with the best results coming from a 75%/25% train/test split with the important features being fuel element position, cycle MWd, total core power, and cycle length. Predicting fuel element burnup performed the best of all the machine learning algorithms and k-nearest neighbors was used to get a corresponding initial ^{235}U loading. The predicted values were then scaled based on errors within the dataset. Ultimately, six of the seven cycles tested were able to use fewer fresh fuel elements and the cycle that used more fresh fuel elements was the result of a mismatch between the desired cycle and reality. Ultimately, a total of 108 fresh fuel elements was used over seven cycles compared to 124 fresh fuel elements used in the corresponding as runs, with a 13% decrease in fresh fuel element usage.
Key words: Neural network, ATR, fuel load optimization, machine learning

1. Introduction

Fuel load optimization of nuclear reactors has largely been studied for pressurized water reactors (PWRs). Applying machine learning to any reactor is tricky but applying machine learning to a dynamic research reactor like the Advanced Test Reactor (ATR) is especially so. The ATR can be thought of as potentially 4.5 individual reactors that act tangentially to each other. Each outer lobe has a defined power split that is controlled by outer-shim control cylinders (OSCCs) and neck shims while the center power drifts based on the average power of the adjacent lobes. The goal of using fuel load optimization in the ATR is to be able to reduce the number of fresh fuel elements that are used and to simplify the fuel loading process for reactor engineers.

Because the fuel loading using the beginning-of-cycle (BOC) ^{235}U data has historically been highly dependent on human interface, the algorithms had a difficult time finding patterns within the dataset. The ^{235}U burnup was found to be the best estimator for optimization, followed by a nearest-neighbors application to pull out an anticipated BOC ^{235}U content.

Since the ATR is a research reactor, the use of engineering judgement should not be ignored in favor of the predictive algorithm. The cycles that were tested using the MC21 drum solver utilized some engineering judgement beyond what the model predicted and what would be reasonable to expect the model to predict. In some cases it may be necessary to use an YA or an NB fuel element instead of the default XA fuel element if a lobe is in need of more reactivity, less power peaking, or protection for other parts of the core. The algorithm will not be able to predict fuel element type as both the YA and NB elements are used sparingly within the dataset and therefore the dataset may not be effectively trained to determine which fuel element type.

1.1 Advanced Test Reactor

The ATR is a research reactor at the Idaho National Laboratory (INL). The ATR is light water cooled and beryllium reflected and consists of forty arcuate fuel elements arranged in a serpentine shape. There are nine flux traps in the ATR, labeled North (N), Northeast (NE), East (E), Southeast (SE), South (S), Southwest (SW), West (W), Northwest (NW), and Center (C). The NE, SE, SW, NW, and C flux traps are surrounded by the fuel elements completely. The A and H experiment positions are also located within the serpentine. The A positions can be separated into inner-A and outer-A positions. Inner-A positions sit between the H-housing and the neck shims while the outer-A positions sit between the neck shims and the corresponding NE, NW, SE, or SW lobe. The H-housing consists of 16 positions circling the center flux trap and contain 14 H experiment positions and two positions dedicated to the N16 system. The B positions are located within the OSCCs but outside the serpentine and consist of small B positions between the fuel and a cardinal-direction flux trap, and large B positions that sit between two OSCCs in a cardinal position. The I-positions are located outside both the serpentine and the OSCCs and consist of small, medium, and large positions. When the ATR is controlled, the neck shims are removed first while the goal is for the OSCCs to remain relatively stable and once all neck shims are pulled, the OSCCs will rotate out. A labeled diagram of the 94CIC ATR core model can be seen in Figure 1.

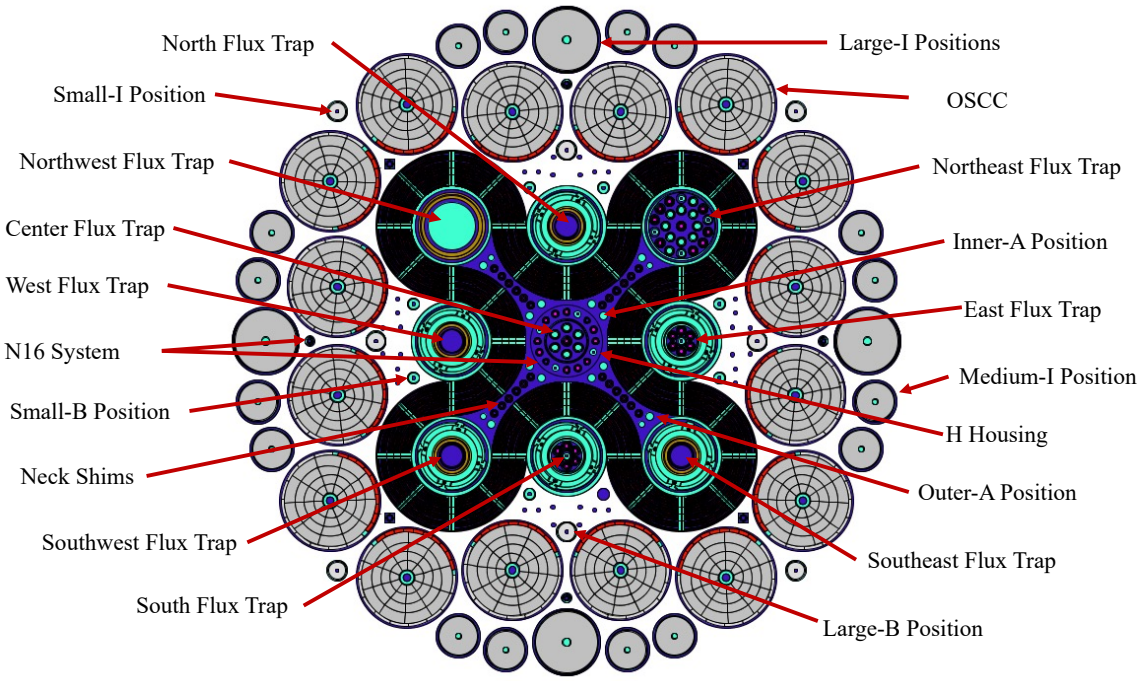

Figure 1 Labeled cross section of the ATR 94 CIC model in PUMA.

Approximately every ten years, the ATR goes through an extended shutdown where all reactor components such as the reflector, OSCCs, etc. are replaced during a process known as the core internals changeout (CIC).

The ATR is controlled by both the neck shim rods and the OSCCs. Neck shims are pulled first to control core reactivity and then the OSCCs are rotated out. The OSCCs are operated in pairs. Neck shims are typically pulled starting at the outer lobe and moving inwards to center. There are two regulating rods in the SE and SW positions, respectively. The SE and SW lobes also typically operate at the highest powers in a given cycle. The NW lobe usually has the largest reactivity sink and is the most minimally loaded lobe. Most often, the NW lobe begins removing neck shims first during a given cycle.

The ATR contains three different types of fuel elements denoted by XA, YA, and NB. XA fuel elements dominate the database and contain an initial loading of 1075g ^{235}U and 0.72 g of ^{10}B. The YA fuel elements contain 1022g ^{235}U and 0.54g ^{10}B initially. The NB fuel elements contain

the same 1075 g ^{235}U but contain no boron. NB fuel elements are utilized sparingly over cycles but are a consideration when a lobe may need a little more reactivity. When a lobe has a mixed loading, the fresh fuel elements are placed closer to the center of the core while the partially burnt fuel elements are placed on the outer portion of the lobes. Loading the ATR as such acts as protection for the reflector.

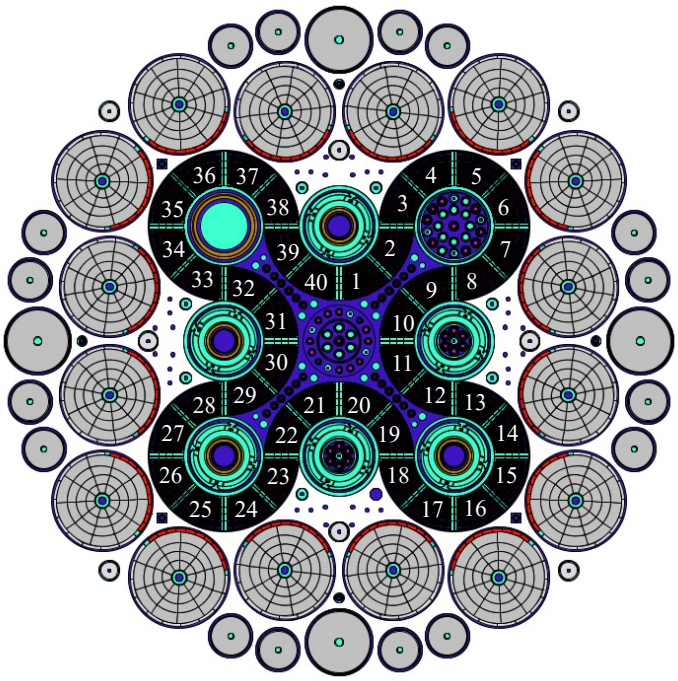

Figure 2 Cross section of ATR 94 CIC model with fuel elements labeled.

The current process for fuel loading in the ATR is established in GDE-185 and is largely based on the premise of finding a fuel loading that accounts for a total 1.3 g/MWd lobe power while also considering some experiment effects [1]. The fuel loading must account for sources/sinks that are introduced through experiment positions, different individual lobe powers, total core power, and different power tilts. The ATR will generally run 4-5 cycles in a year depending on the balance between regular and powered axial locator mechanism (PALM) cycles and the outage days between cycles. Outage length between ATR cycles can vary but are generally a minimum of two weeks with a maximum of approximately 100 days, with nominal outages being

approximately between 21-26 days in length [2]. In 2021, the standard PWR outage was approximately 32 days in length after a total cycle length of 18-24 months [3].

There are three main types of core loading patterns in a standard PWR. Out-in loading, scatter loading, and low-leakage core loading. Examples of out-in loading for a three-batch core, scatter loading for a three-batch core, and low-leakage core loading for a four-batch core can be seen in Figure 3. Out-in loading, which is no longer used, places the freshest fuel elements in the outermost part of the core and the most-burned fuel elements closer to the center, reducing the peak-to-average flux by moving the peak power to the center. However, the because the center fuel elements have the least uranium content, the core center has a power that is lower than the average core power. Also, the peak elements located on the outermost portion of the core causes fast fission neutrons to both leak out of the core and to hit and damage the pressure vessel. In scatter loading, the fresh fuel is again on the outermost ring, and within that ring is a symmetric mix of partially burned fuel elements. When loaded appropriately, a scatter loaded core can be a low-leakage core. Finally, the low-leakage core loading puts only burned fuel elements on the outermost ring of the reactor core and specifically places the most-burned or even stainless-steel dummies at areas where the peak flux occurs in an effort to minimize radiation damage to the pressure vessel. The fresher the fuel elements, the more likely those elements are to be moved to a less consequential position away from welds or any peaking. Low-leakage core loadings are typically only used for PWRs because boiling water reactors (BWRs) operate at a lower pressure and the damage to the vessel is less of a concern [4].

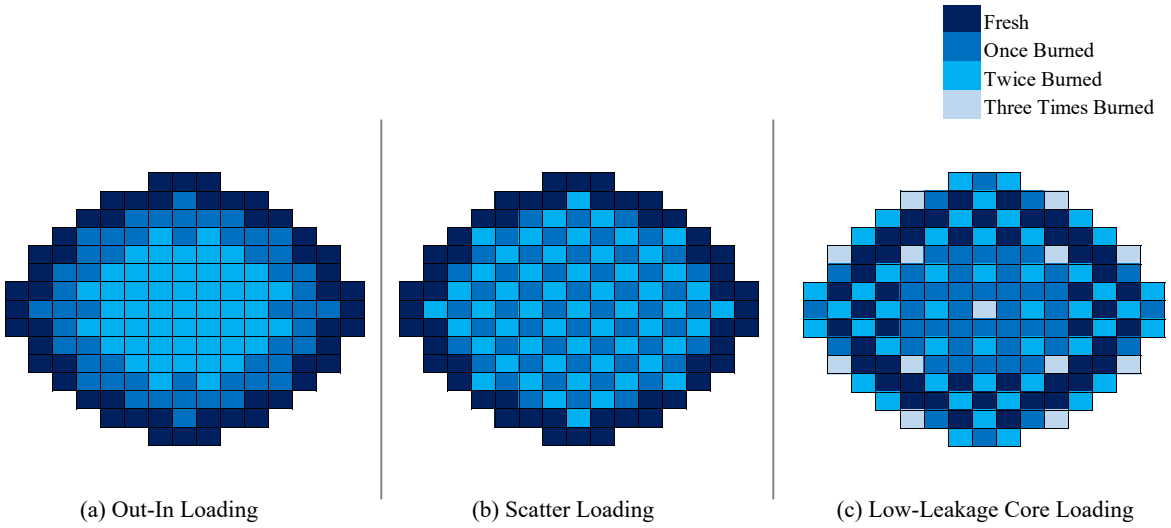

Figure 3 Common PWR reloading patterns.

All three of the standard fuel element loading patterns in reactors are set up to account for symmetry. Fuel loadings in standard reactors do not need to account for any experiment that acts as a source or a sink in the reactor, like the ATR does. The ATR also operates at five different power levels while the standard light water reactor (LWR) operates at one. A typical nuclear reactor can have 18-months to determine an appropriate core loading, which is roughly the same amount of time that the ATR runs four cycles.

1.2 Machine Learning

Machine learning is a branch of computer science and artificial intelligence which focuses on making algorithms that use data and statistics that computers may learn from. There are three types of machine learning, unsupervised machine learning, supervised machine learning, and reinforcement learning.

Supervised machine learning uses labeled data, or data where the output is known. The model is given an input and the algorithm adjusts weights based on the outcome until an appropriately fitted solution is found [5]. Classification models and regression models are the two

main types of supervised machine learning models. Classification algorithms work to separate data into certain groups while regression algorithms work to make predictions of projections based on the existing data. Some methods can be used for both regression and classification problems such as decision trees and random forest. A decision tree and by extension, random forest, can both be used to group items into a certain category, or they can be used as a regression algorithm. Since the output is known within the dataset, supervised machine learning algorithms predict the output based on the input, establishing answers based on trends in the data [6].

Unsupervised machine learning uses unlabeled data, meaning the answer is not stored within the dataset and the algorithm is able to make predictions based on the examining the structure of the dataset. One common type of unsupervised machine learning is clustering, where the algorithms group similar features together [7].

Reinforcement learning is similar to supervised machine learning in that the dataset contains both input and output features, supervised machine learning establishes a pattern between input and output while reinforcement learning operated by punishing bad outcomes and rewarding good outcomes [8].

Regardless of the type of machine learning algorithm used, there exists several steps before and after the model is implemented, including but not limited to, defining the problem, data collection and preparation, exploratory data analysis (EDA), feature and model selection, evaluation of model, and managing and presenting results. When defining the problem, it is important to consider the kinds of data that is accessible and consider what kinds of parameters can be used and at which point since it may not be prurient to use certain features as input if those features aren't always readily available. Data collection is the most time-consuming portion of machine learning. It is beneficial to collect as much data as possible at the beginning and then trim

the dataset later based on which features end up being important. Once the data is collected, EDA can be used to truly understand aspects and correlations within the dataset. Within the dataset, there will likely be missing or incomplete data where it is then appropriate to clean in some way. Cleaning the data could mean normalizing the data, removing unnecessary data, or using a methodology, such as KNN-imputation to replace missing data. The next step is choosing and tuning the models. It can be useful to test different algorithms within the same category to see which produces the most effective results for the problem and based on the results of said model, choosing which features to keep and which features to prune. Once the features and model have been selected, it is appropriate to run the models and process the results as necessary [9]. There are many types of machine learning algorithms. Sections 1.2.1 through 1.2.5 give some background on specific algorithms used for calculations.

1.2.1 KNN

KNN is an algorithm that pulls in the closest values for a datapoint based on the Euclidean distance of a normalized point to its neighbors. It is important to normalize datapoints prior to using KNN so that features of different orders of magnitude can be treated equitably within the algorithm. KNN uses min-max normalization which aligns all values in a dataset to exist between 0 and 1. The equation for the Euclidean distance between two points, p and q, is given in (1) and the equation for normalization of a point x in a dataset is given in (2) [10].

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{1}$$

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2}$$

The first usage of KNN occurs in the EDA section, where KNN imputation was used to clean up missing portions of the dataset. KNN is technically a classification algorithm but may be used to replace values such as with KNN imputation. The second usage of KNN takes the predicted output of the models and the estimated MWd and finds the five closest values within the dataset.

1.2.2 KNN Imputation

Machine learning algorithms generally require numerical input without any missing or NaN values. There are a handful of cycles that have some missing data but removing that data is undesirable. To utilize the maximum amount of data, KNN imputation was used. KNN imputation works by replacing missing NaN values in a dataset by taking the average of the nearest points in the dataset based on the data that is available. For example, if a dataset has 12 features and there is a row where 3 of those features show NaN values, KNN imputation will take the 9 available features in that row and find the k-closest matches within in the dataset and will replace the three NaN values with a statistically appropriate predicted value [11].

1.2.3 Linear Regression

Linear regression is probably the simplest regression algorithm. Linear regression takes the inputs or features of the dataset and separates the features linearly. In two dimensions the data is separated by a line, in three dimensions the data is separated by a plane, and in greater than three dimensions the data is separated by a hyperplane.

Equation 3 is the equation for a line that goes through the datapoints, where y is the output, x_i represents the datapoints, and β_i represents the line that goes through the datapoints:

$$y = \sum_{i=0}^{m} \beta_i x_i \tag{3}$$

The separation boundary is considered to be the best line that goes through the datapoint and is found through a process called least squares optimization, where the best line is determined to be the sum of the minimized squared difference between the actual value and the predicted value of each datapoint, or:

$$Separation\ Boundary = \sum_{j=0}^{N}\left(t_j - \sum_{i=0}^{m}\beta_i x_i\right)^2 \quad (4)$$

Figure 4 shows an example of a linear regression algorithm that intends to linearly split the blue circles and the green squares. A linear regression problem will usually not be able to find a line that entirely separates data but may be able to separation boundary that separates existing data while having the lowest distance between the boundary line and each point [12].

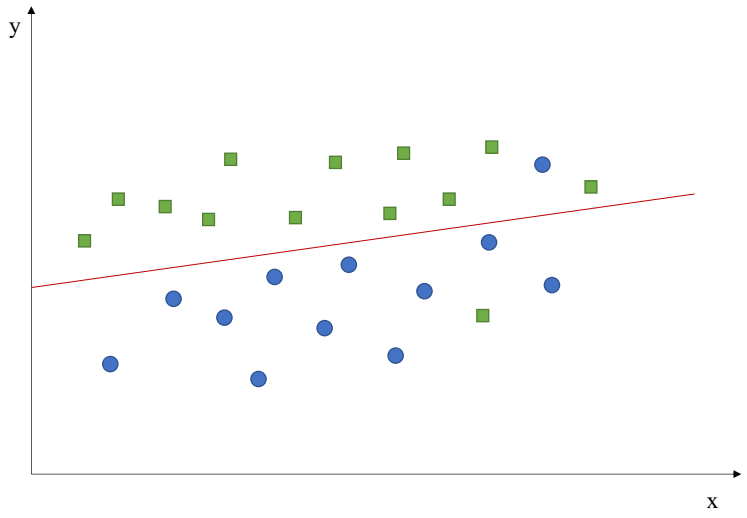

Figure 4 A linear regression example with a decision boundary.

1.2.4 Random Forest Regression

Random forest models in machine learning can be used for either regression tasks or classification tasks. Random forest regression is a form of ensemble learning. Ensemble learning works by combining results or predictions from multiple machine learning algorithms, and in the

case of random forest regress, combining the results of multiple decision tree regressions. In random forest, the algorithm selects a subset of features at random, with replacement, and calculates the individual outcome, and then combines all results to find a solution. The total solution can be represented by equation (5), where $f_m(x)$ is the m-th tree [13].

$$f(x) = \sum_{m=1}^{M} \frac{1}{M} f_m(x) \tag{5}$$

In a decision tree, the individual variance is very high, but when many different decision trees are combined and are allowed to randomly select features for prediction, the total variance of the model is low.

1.2.5 Neural Networks

McCulloch and Pitts established the first mathematical model of a neuron in 1943, which was minimally defined as the sum of inputs, x_i , multiplied by a corresponding weight, w_i, and by using an activation function, if the summation of all inputs times weights was greater than some threshold (θ), the neuron fired, if the summation was less than some threshold, the neuron does not fire. Equation (6) represents the function h, while equation (7) represents the activation function $g(h)$.

$$h = \sum_{i=1}^{m} w_i x_i \tag{6}$$

$$g(h) = \begin{cases} 1 \; if \; h > \theta \\ 0 \; if \; h \leq \theta \end{cases} \tag{7}$$

Figure 5 shows a diagram of a McCulloch and Pitts neuron.

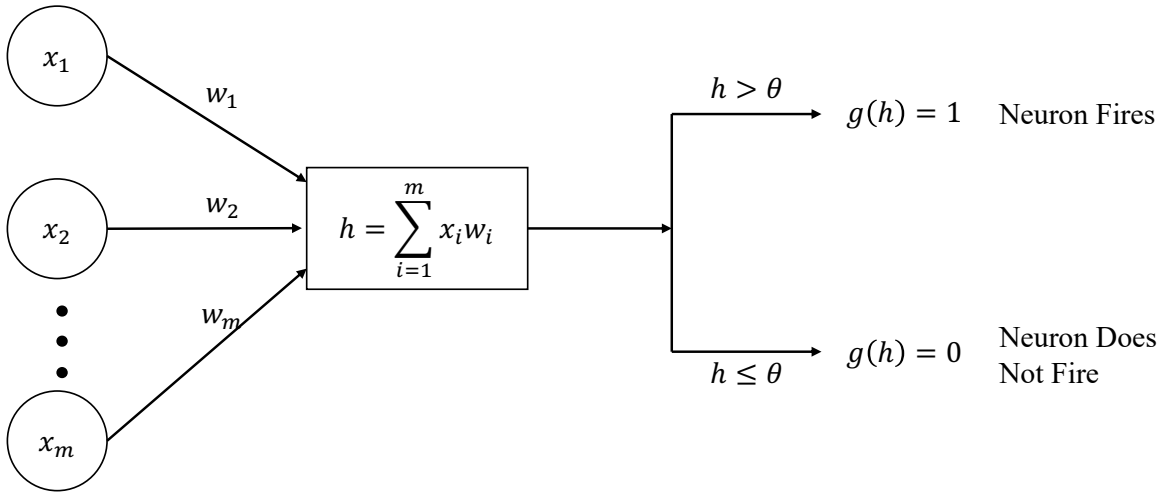

Figure 5 McCulloch and Pitts neuron.

McCulloch and Pitts neurons model only a single neuron, but a single neuron is not overly useful. In order to tie McCulloch and Pitts neurons into a neural network, input nodes with corresponding weights are added, and Perceptron is formed. The input layer, $\boldsymbol{x}$, is a vector comprised of i input nodes, ranging from 1 to m. Similarly, the output layer, $\boldsymbol{y}$, is a vector comprised of j outputs, ranging from 1 to n. Each input node connects to each neuron using a weighted connection, $w_{i,j}$, where i is the index of the input node ranging from 1 to m, and j is the index of the neuron, ranging from 1 to n. The Perceptron consists of an input layer and an output layer. Each input connects to each neuron by some weight, so there will be $w_{m,n}$ weights in the network. Figure 6 shows the perceptron network.

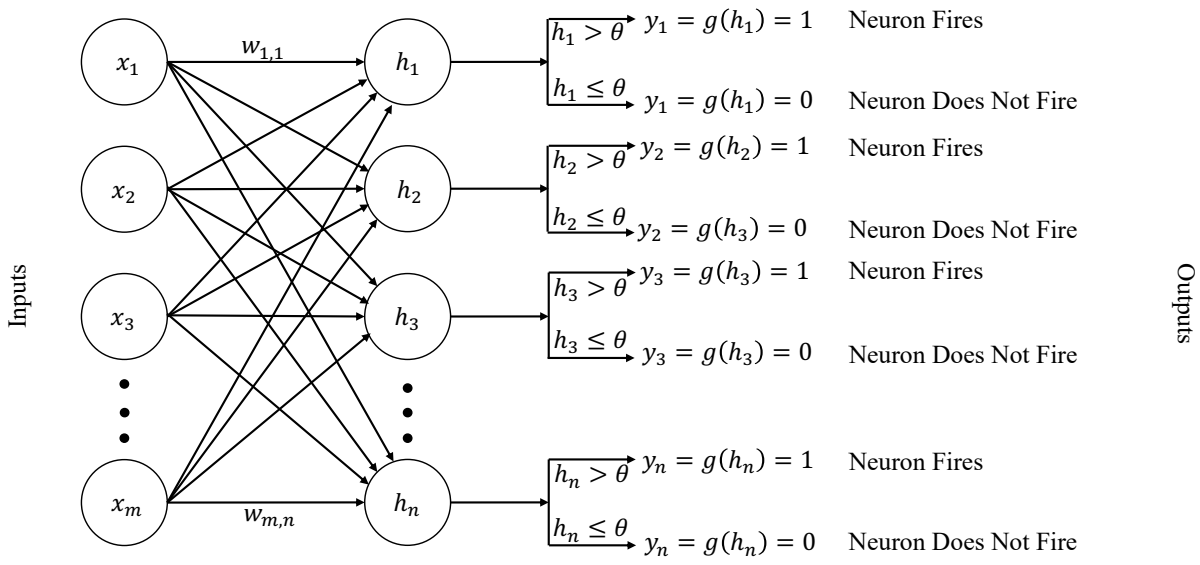

Figure 6 The Perceptron network.

The output layer is comprised of the predicted values of the neural network. Neural networks are supervised machine learning algorithms and, therefore, the predicted values of the perceptron algorithm can be compared to the correct values as given in the dataset. The vector of correct values is known as the target vector, $\boldsymbol{t}$, or just the targets. Neural networks consist of a forward phase and a backpropagation phase. Initially calculating the weights and the outputs comprised the forward phase. Once the forward phase is done, the predicted output is compared to the labeled data and the weights as they apply to the incorrect neurons are updated to hopefully land on the correct result. The new weight becomes the old weight plus some learning rate, η, multiplied by the initial input value and the difference between the target output, t_k, and the predicted output y_j. The learning rate defines how fast the neural network can learn and is typically between 0.1 and 0.4. Setting the learning rate low limits how fast the rates may change. Setting the learning rate to 1 is effectively the same as omitting n altogether and tends to create unstable networks due to how aggressively the weights update. Equation (8) shows the equation for updating the weights in the backpropagation step.

$$w_{ij} \leftarrow w_{ij} + \eta\left(t_j - y_j\right) * x_i \tag{8}$$

The process of calculating outputs and updating the weights is calculated over a defined number of iterations. The weights of a neural network are the most important part of the network. The perceptron algorithm can be expanded into a full neural network by adding one or more hidden layers of neurons to add more weights to the network. Hidden layers are layers of neurons between the input layer and the output layer. When using a single hidden layers the targets are unknown and when using more than one hidden layer, neither the inputs or the targets are known to the hidden layers. When using hidden layers, the output layer cannot see the inputs. In a neural network, the activation function is now a sigmoid function, represented as a, shown in equation (9), where beta is some positive number.

$$a = g(h) = \frac{1}{1 + \exp(-\beta h)} \tag{9}$$

Figure 7 shows a neural network with k hidden layers ranging from 1 to l.

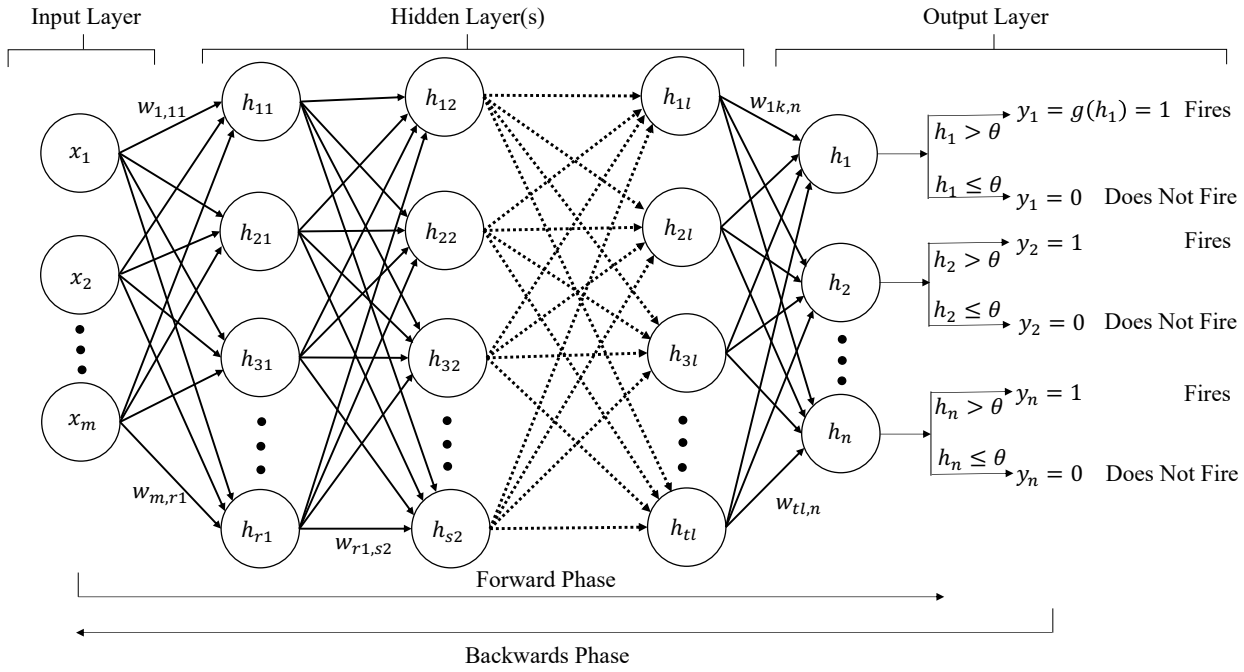

Figure 7 A neural network with hidden layers.

The activation function is calculated for each neuron in each hidden layer, shown in equation (11) with hidden layer weights represented using v.

$$h_j = \sum_i x_i v_{ij} \tag{10}$$

$$a_j = g(h_j) = \frac{1}{1 + \exp(-\beta h_j)} \tag{11}$$

Once the activation function is calculated for each hidden layer neuron, the output layer neurons can be calculated with similar equations.

$$h_j = \sum_i x_i v_{ij} \tag{12}$$

$$y_k = g(h_k) = \frac{1}{1 + \exp(-\beta h_k)} \tag{13}$$

Once y is calculated, the forward phase of the network is complete. Backpropogation begins with calculating the error between the calculated output and the actual target value. The error between the output layer and the last hidden layer is calculated first, in equation (14), and the error in the subsequent hidden layers, moving backwards through the network, is calculated with equation (15).

$$\delta_{ok} = (t_k - y_k) y_k (1 - y_k) \tag{14}$$

$$\delta_{hj} = a_j (1 - a_j) \sum_k w_{jk} \delta_{ok} \tag{15}$$

Once all the errors are calculated, the output layer weights can be updated using eq (16) and the subsequent hidden layer weights follow, again moving backwards through the neural network, with eq (17).

$$w_{jk} \leftarrow w_{jk} + \eta \delta_{ok} a_j^{hidden} \tag{16}$$

$$v_{ij} \leftarrow v_{ij} + \eta \delta_{hj} x_i \tag{17}$$

The forward/backward propagation process repeats until the learning ends. The error function defined in (18) for the perceptron algorithm no longer applies to the neural network. The multiple potential hidden layers each have their own associated errors, and in theory, the combination of weights could cancel each other out and trick the network into thinking an incorrect neural path has no error. To mitigate the possibility of cancelling errors, all errors are given the same sign using the sum-of-squares error instead.

$$E(\boldsymbol{t}, \boldsymbol{y}) = \frac{1}{2} \sum_{k=1}^{n} (t_k - y_k)^2 \tag{18}$$

The error of the network is calculated until a local minima is reached [12]. A more modern and more commonly used form of the activation function involves using piecewise linear activation functions known as the rectified linear unit or ReLU functions instead of the sigmoid seen in Equation (11). The ReLU function can output a true zero value unlike the sigmoid, which allows for the hidden layers in the neural network to have "true zero" values, which allows for models to train faster [14].

1.2.6 Train-Test Split

Train-test split is a methodology used in machine learning algorithms where a dataset can be separated into training data and testing data. The training data is used to establish the trends in, or fit, the model. The testing data is then used to assess how the model is performing. It is important to not train data on the exact values that the algorithm will be tested on or there will be a risk of overfitting because the model will have seen the correct answer already [15]. Chicco, et al. compared two of the best performing methods for assessing model performance, those being the coefficient of determination, or R^2 score, and the symmetric mean absolute percentage error

(SMAPE) and found that while both methodologies are some of the few that only report a high score when accurately predicting the data, the R^2 score performed better at reporting on how well a model predicted results. Other studied metrics included the mean square error (MSE), root mean square error (RMSE), mean absolute error (MAE), and mean absolute percent error (MAPE), all four of which are difficult to interpret due to all of them being able to range between 0 and infinity. The R^2 score is bound between negative infinity and one and can be thought of as a percentage, which makes the results simple to understand compared to MSE, MAE, RMSE, and MAPE. R^2 scores are interpreted by 1 being a perfectly fit model and the model becoming increasingly less fit the closer to negative infinity the R^2 score goes. Mathematically, the R^2 score is calculated as:

$$R^2 = 1 - \frac{Residual\ Sum\ of\ Squares\ (RSS)}{Total\ Sum\ of\ Squares\ (TSS)} \tag{19}$$

Where RSS and TSS are represented in equations (20) and (21):

$$RSS = \frac{1}{m}\sum_{i=1}^{m}(X_i - Y_i)^2 \tag{20}$$

$$TSS = \frac{1}{m}\sum_{i=1}^{m}(Y_i - \bar{Y})^2 \tag{21}$$

Where m is the number of samples, Y_i represents the actual i-th value, $\bar{Y}$ represents the mean of the Y values, and X_i is the predicted i-th value [16].

Therefore:

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(X_i - Y_i)^2}{\sum_{i=1}^{m}(Y_i - \bar{Y})^2} \tag{22}$$

1.2.7 Overfitting and Underfitting

Something to be cautious of in machine learning models is overfitting or underfitting a model. A good machine learning model is able to predict based on trends in the data. A model is

considered to be overfit when, instead of predicting based on data trends, the model is attempting to find an exact fit to every datapoint. A model is underfit when the model cannot create predictions based on the dataset. A balanced fit occurs when the model is able to predict outputs based on trends in the data but does not try to find an exact value for each point [17]. Figure 8 demonstrates balanced fit, underfit, and overfit where dots represent datapoints and the dotted red line represent the curve fit.

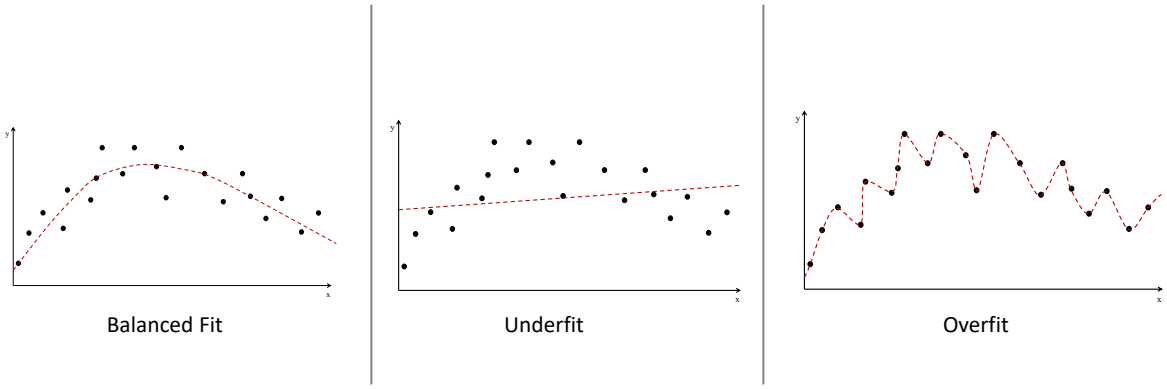

Figure 8 Examples of different machine learning fitting outcomes.

1.2.8 Sci-kit Learn/Python/Jupyter

The work presented utilized jupyter notebooks in python 3.7 that ran on a GPU that was able to use Tensorflow 2.1. Sci-kit learn version 0.23 was used for most of the regression models and keras was used to add extra functionality into the neural networks. Keras is an API that is built on TensorFlow 2 to help implement neural networks.

1.3 MC21 and the MC21 Drum Solver API

The Common Monte Carlo Design Tool (CMCDT) contains both MC21, the Monte Carlo code for neutron and photon transport, and PUMA, the API used for building MC21 models [18]. The ATR model was used for the work completed and is a java based, full core model of the ATR. The model contains cycle data such as fuel element loading and cycle as run data, models of many

of the experiments or the reactivity equivalents of the experiments, and individual cycle models that combine the necessary information for running in MC21. The MC21 model for the ATR includes two options for loading fuel elements. The first option is explicitly loading fuel elements by their corresponding fuel element ID and the second uses a generic setup where PUMA will pull in the nearest fuel element by ^{235}U gram loading and fuel element type. The work presented utilized the second, generic methodology due to limitations with listing existing fuel elements.

INL developed an API using MC21 to solve for critical drum positions. The API is a flexible tool that was designed specifically for the ATR and for the core physics analysis process. Therefore, there are many different ways the API may be used. The API first completes a critical drum search where the drums are rotated in various patterns so that the overall drum curve can be created. Once it completes that, the drum solver will iterate around user-input parameters including but not limited to desired nominal and maximum power splits, desired increase in drum rotation, desired neck shims (optional), desired critical eigenvalue, desired tolerance of critical eigenvalue, maximum desired drum rotation, etc. If the desired drum neck shim positions are not set, the API will pull a neck shim when the drum rotation exceeds the maximum desired rotation. Once it finds a suitable match, the API will run a spatial calculation to determine the critical eigenvalue and then run a depletion over a specified timestep length. Once the API moves on to the next timestep, it will start with the drum search again. Critical shim positions are not recalculated at every timestep, instead, the user indicates which timesteps the critical shim positions are recalculated.

The ATR has been working on transferring from the current code of record for the core physics analysis (CPA) calculations, HELIOS, to MC21. As run calculations have been completed for all cycles between 144B and 169A and a database of fuel element burnup has been completed for use in future cycles [19]. HELIOS remains the code of record for cycles 158A through 169A.

2. Literature Review

Kim et al. used artificial neural networks (ANNs) combined with a fuzzy logic rule-based system to create an optimal fuel shuffling system which sorted options via heuristic rules and improved searching speed from the fuzzy rule-based system. The optimal loading was found when the local power peaking factor was minimized while the k-effective was maximized. The authors used back-propagation networks for both the local power peaking factor and k-effective. The authors found that the use of the fuzzy membership function significantly reduced the searching space and time of the rule-based system. It was also found that the ANN predicted the core parameters faster than numerical codes. The advice of the authors was to use the system not as a real reloading tool, but as a supporting tool for the LP designer to obtain a more optimal solution [20].

Yamamoto studied the use of machine learning algorithms for fuel load optimization in a PWR. The simulated annealing (SA), direct search (DS), binary exchange (BE), and genetic algorithm (GA) were all considered along with hybrid methods where two methods were combined to account for the pitfalls of the individual algorithms. GAs can search a global area but cannot fine search a local area. BE and DS are unable to escape from local optima. SA can escape local optima but obtaining favorable results can be computationally expensive. The hybrid methods considered were DS paired with BE and GA paired with BE. Since SA and DS share the same methodology, the combination of DS and BE would produce the same results as SA and BE. SA produced the lowest standard deviation of the fitness value, which is an indication of the dependence on the random number seed. The GA-BE produced a similar standard deviation but calculated three times fewer loading patterns. The DS-BE hybrid method was unable to escape local optima and the fitness function was the same as the fitness function for DS alone [21].

Yamamoto et al. (2002) also studied the application of a distributed genetic algorithm (DGA) for in-core fuel load optimization. A DGA is a good tool for use in parallel computing. In a DGA, the population from the standard GA can be divided into islands capable of communicating with each other. One important consideration regarding the DGA is diversity. If the diversity is very small, the computation time is fast, but there is low probability of escaping local optima. Likewise, if the diversity is very large, the computation time will be very long, but local optima can be escaped. When the number of islands increases, the diversity of the population on each island increases, but when the number of islands increases, the population of each island decreases, and the diversity decreases. Therefore, there should be an optimal number of islands. The authors studied three different migration patterns – no migration, elite migration, and random migration. In elite migration, the superior island inhabitants were used as the migrants. In random migration, the migrants were selected randomly. The optimization found the best results when using elite migration for three islands with one migrant and a migration period of two cycles. DGA was found to perform better than the traditional GA [22].

Sadighi et al. utilized a hybridization of Hopfield neural networks (HNNs) and SA for the fuel loading. The HNN was selected for its parallel abilities and its capability to locate local minima. The HNN cannot escape local minima so SA was used to find a result closer to the global optimum. The HNN will always converge to a stable minimum, at which point SA can be used to find the global optimum [23]. Fadaei et al. also studied the use of HNNs alone and hybridized with SA for a water-water energetic reactor (VVER). The authors found that the using HNN alone found a solution at a local minimum and when combined with SA, the solution was considered the global optimum [24].

Tombakoglu et al. studied the use of GA's to complete and analyze the performance of a fuel load optimization. It was found that the population size and the random number seed strongly affected the convergence rate of the LP optimization. Discharge burnup was found to be more effective than cycle burnup and the effects of using discharge or cycle burnup are increased when there is an increase in the power peaking factor. Beginning-of-cycle (BOC) power peaking factors were also reduced by lowering the boron concentration in the core [25].

Faria et al. utilized an ANN to conduct a fuel load optimization. The criterion for the optimization was to minimize the maximum power peaking factor. The local power peaking factors were reduced, thus reducing the overall power peaking factor. The authors used a backpropagation ANN with input, output, and hidden layers. The cases with the lowest power peaking factors were selected for training. Weights were calculated via error backpropagation until the ANN returned an error less than the value of the prescribed error. Four loading patterns were chosen: one pre-defined optimum pattern, one randomly selected pattern, and two of the most successful patterns. The latter three patterns were used to minimize the power peaking factor. Composition, burnup, and enrichment defined the fuel type. Fuel composition and quantity of each fuel assembly are important factors in core optimization. The implemented process is dependent on the chosen initial configuration and the reference case strongly influenced in the generation of the new loading patterns, resulting in better results with fewer generated cases. There is no way of knowing whether the global optimum is obtained by the ANN methodology [26].

Ortiz et al. studied the use of a multistate recurrent neural network combined with a fuzzy logic rule for a fuel load optimization in a boiling water reactor (BWR). The multistate recurrent neural network was used to propose possible fuel lattices while the fuzzy logic rule determined if the individual lattices were acceptable. The optimization criteria were to minimize the local power

peaking factor while keeping k_∞ within a given range. The HELIOS code was used to create the fuel lattices. The fuzzy rule system was used to determine whether it would be worthwhile to allocate a fuel lattice with a local power peaking factor above a certain value to the bottom of a fuel assembly. The authors found that 30 iterations were sufficient to produce good fuel lattices [27].

Hill et al. used tabu search (TS) to optimize the fuel loading of a PWR. TS is a local-searching meta-heuristic method used to solve the combinatorial optimization problem. TS has limited capabilities for escaping local optima. TS uses an iterative process with a short-term memory to check for convergence on the optima. TS is forbidden from returning to previously visited solutions until the tabu tenure has been reached. Use of tabu tenure enables solutions to escape local optima. When the search slows, intensification may be used to focus on the search space surrounding the best solutions, and diversification can be used to improve results for lesser developed solutions. The TS algorithm was implemented in the Fuel Optimization for Reloads: Multiple Objectives for Simulated Annealing for PWRs (FORMOSA-P) code. The initial TS LP is created by mutating a user-specified LP 1,000 times. If that LP is not feasible, the process is repeated until a feasible LP is obtained. Feasibility is based on user-defined constraints, usually based on reactor safety and control limits. The established LP is mutated once to create a neighbor, checked for feasibility, and if the neighbor is feasible and not a part of the tabu list, is accepted and stored. If the LP is not feasible or is already a part of the tabu list, the LP is rejected and discarded. The tabu list is an array of the most recently evaluated loading patterns and the array length is equal to the tabu tenure. The process is repeated for the new LP until the desired number of LPs has been generated and stored. The best LP is selected and added to the tabu list. The LP is checked against the qualifications for diversification and intensification and if the qualifications

are not met the process is repeated using the new LP. The process continues until a maximum number of LPs are met. The authors found that diversification and intensification did not improve the results. The only control parameters were the tabu list length and neighborhood size. The authors also found that the TS implementation outperformed the SA and GA implementations available in the FORMOSA-P code [28].

Castillo et al. studied five different heuristic optimization techniques on a BWR. The techniques were the ant colony system, ANN, GA, greedy search (GS), and a path relinking and scatter search hybrid. The authors considered the maximum local power peaking factor, k_∞, average enrichment, average gadolinia concentration of the lattice, neutronic grade pre- and post-burnup, and the global cost in computation time. The authors used the position vector of minimum regret. The authors found that GS and ant colony systems found the best lattices neutronically before burnup. GS and GA performed the best neutronically after burnup. However, GS performed randomly and ranked the worst in terms of computation time. The path relinking and scatter search hybrid method performed the best for quickly finding optimal solutions. GAs and path relinking combined with scatter search performed the best in terms of global cost [29].

Safarzadeh et al. considered a combination of the artificial bee colony (ABC) and particle swarm optimization (PSO) algorithms to maximize the cycle length by increasing the initial cycle reactivity while maintaining safety limits. Parallelization was considered to improve optimization capabilities. It had been shown previously that ABC algorithm produced improved results over the PSO. The authors found that the hybridized PSO-ABC algorithm produced improved results from the ABC alone [30].

Hedayat studied the application of a modified simulated annealing algorithm to a 5-MW material test reactor (MTR). The optimization goals were to maximize the refueling cycle length

and thermal neutron flux while adhering to safety limits and operational constraints. It was found that the cycle length was extended significantly, and most of the neutron fluxes analyzed over the fixed boxes increased [31].

Kashi et al. utilized a bat algorithm approach for the load pattern optimization where the minimization of the power peaking factor and the maximization of k-effective were the optimization parameters and the overall goal was to maximize the economics while maintaining defined safety parameters. The bat algorithm optimization was modeled after the echolocation behavior of bats and, in specific circumstances, reduces to PSO and the harmony search algorithm method. The Bat Algorithm Nodal Expansion Code (BANEC) was developed to optimize the fuel LP. In BANEC, one node was assigned per fuel assembly against the established, validated, and benchmarked Average Current Nodal Expansion Code. The use of the nodal code decreases the computation time. Results for BANEC were promising when compared to the Continuous Firefly Algorithm Nodal Expansion Code [32].

Barati utilized a combination of cellular automata (CA) and quasi-simulated annealing (QSA) to minimize mass and deformation of a fuel plate for a multipurpose research reactor (MPRR). The goal was to increase the reliability and lifetime of the fuel plate. The CA-QSA methodology found comparable results to the genetic algorithm and neural network methods that were previously studied [33].

Khoshahval et al. studied the use of biogeography-based optimization (BBO) for fuel load optimization with the goal of minimizing the power peaking factor. The BBO algorithm is comparable to the PSO algorithm, but unlike the PSO algorithm, the BBO algorithm has a greater ability to escape from the local minima to achieve a global optimum. When results of the BBO

algorithm were compared to those of the PSO, BBO outperformed the PSO for the same initial random patterns [34].

Park et al. introduced a multi-objective simulated annealing (MOSA) algorithm which employed an adaptively constrained discontinuous penalty function (ACDPF) to solve the fuel load optimization problem for a Yonggwang Nuclear Unit 4 (YGN4) model PWR. The authors found that, for the cycle 4 YGN4 design, the updated ACDPF outperformed the original. It was also found that the ACDPF improved the efficiency of the MOSA optimization [35].

Saber et al. used a multi-layer perceptron neural network (MPNN), a priori association rule, and PSO to optimize the fuel loading for the 10-MW International Atomic Energy Agency (IAEA) low-enriched uranium (LEU) tank-in-pool MTR. The MTR is a benchmark reactor designed by IAEA for the conversion of high-enriched uranium to LEU. The authors used an a priori algorithm to help generate a set of training samples and combined that with the PSO. K-effective and the local power peaking factor were predicted using the MPNN. When compared to the Levenberg-Marquardt, quasi-Newton, and resilient propagation algorithms, the developed MPNN produced, in the lowest amount of time, results with the highest accuracy and fewest hidden layer nodes in while also producing the most effective features [36].

Mahmoudi et al. studied the application of the gravitational search algorithm (GSA) to the fuel load optimization problem. The goal was to minimize the power peaking factor while maximizing k-effective. The GSA is based on the law of gravity and consists of an isolated system of masses and, using the gravitational force, each mass in the system can communicate with all other masses in the system. Heavier masses will have a greater gravitational force but will be slower moving. The closer the masses exist in proximity to each other, the higher the gravitational force will be. Exploration was used at the beginning of the algorithm in order to escape local

minima. The best performance of the GSA was achieved by controlling exploration and exploitation so that only the best k-values would attract each other. When comparing results for Shekel's foxhole problem as applied to GSA, GA, SA, harmony search, and harmony search with differential mutation, the GSA was able to converge in the fewest iterations and had the lowest standard deviation [37].

Sobolev et al. studied the use of genetic algorithms for nuclear fuel load optimization by maximizing the fuel burnup depth for a high-power fast breeder reactor. Lucky answers for the first solution were shown to not effect subsequent solutions and the GA found improved results on the fuel burnup depth [38].

Ortiz-Servin used population-based metaheuristics and decision trees to optimize the fuel loading for a BWR. The goal was to maximize k-effective while minimizing the local power peaking factor – with an emphasis on reducing the computation time. Using decision trees, the computational time was reduced by a factor of 1,200 [39].

Ahmad et al. studied PSO for a material test reactor. The MTR is an asymmetric research reactor and therefore does not benefit from symmetry like PWR optimizations do. PSO is a population-based optimization technique inspired by the behavior of animals – such as birds – that lack a group leader. Each bird finds its own best food source then calls to the other birds. Since the PSO problem can get trapped in local optima, a catfish effect was also applied. In the catfish effect, 20% of the particles were replaced with catfish particles and positioned at extremes, enabling faster convergence on a global optimum. First, the single-objective problem of maximizing k-effective was addressed, followed by the multi-objective problem of k-effective maximization and local power peaking factor minimization. When k-effective was solved for as a single-objective function, the power peaking factor increased, so it was necessary to solve the

multi-objective function. The multi-objective problem was solved via a penalty function applied to k-effective whenever the local power peaking factor reached an upper limit. The convergence rate of PSO was improved by the catfish algorithm [40].

Israeli et al. studied a fuel load optimization based on core physics heuristics. First, the single-objective problem of maximizing k-effecting was solved, followed by the multi-objective function of maximizing k-effective while minimizing the local power peaking factor. An adaptive geometric crossover was developed that considered the geometry of the core. When compared to the non-geometric crossover that is typically used, the authors found that the geometric crossover produced better results for both the single- and multi-objective problems. To escape local minima, genetic diversity in the form of an adaptive mutation scheme was injected into the population as needed. Use of an adaptive mutation scheme improved the optimization process by adding in new control measures. The authors also found that an increased selection pressure increases the convergence rate and the genetic diversity of the population; however, too-high of a pressure results in premature convergence to a local optimum. On the other hand, if a pressure is too low, the convergence rate is slowed [41].

Oktavian et al. used a quantum-inspired evolutionary algorithm to complete a fuel load optimization on a Korean Standard Nuclear Power Plant -1000 (KSNP-1000) reactor core. Again, the goal was to maximize k-effective while minimizing the local power peaking factor, thus minimizing the operating cost of a nuclear power plant. Evolutionary algorithms are inspired by evolutionary processes. The quantum-inspired evolutionary algorithm utilizes quantum computing for calculating the evolutionary algorithm. The evolutionary-inspired GA produces results with fewer calculations than standard GAs. Using the quantum-inspired evolutionary algorithm, the authors were able to increase the length of the cycle [42].

Nasr et al. used a polar bear optimization to complete a fuel load optimization for a VVER-1000 reactor. The goals were to maximize cycle length, maximize the departure from nucleate boiling ratio (DNBR), and to flatten the power distribution while accounting for safety systems such as the power peaking factor, maximum fuel temperature, and maximum cladding temperatures. The polar bear optimization algorithm is meant to emulate how polar bears hunt for food. The global search parameter can be described thusly: the polar bear searches for food, if no food is found, the bear remains on the ice until a position for long-distance swimming is found. Once a hunting position is found, the polar bear searches for prey in the best location (local search). The authors found that the power peaking factor was flattened, the DNBR increased, and the centerline fuel temperature and axial cladding temperature were lower than the actual core configuration – all of which indicate a safer operational state [43].

Zameer et al. used a fractional order PSO method for calculating the optimal fuel load of a PWR. The goal was to increase the cycle length by maximizing k-effective and minimizing the power peaking factor. A fractional-order PSO algorithm improves on the standard PSO by utilizing fractional-order dynamics with a wavelet mutation mechanism to allow the optimization to escape local minima. The authors reported improved computational times and results compared to the PSO [44].

Jarrett and Heidet utilized an evolutionary algorithm to establish a “proof of concept” fuel optimization technique in the versatile test reactor, since the versatile test reactor is still in progress. Peaking factor, excess reactivity, and economic costs of different fuel assemblies were considered [45].

The research studied predominately covers the ways that machine learning has been used to complete fuel load optimizations for PWRs with some papers addressing fuel load optimization

for other kinds of research reactors. Different machine learning algorithms were studied over the literature review, but the genetic algorithm seemed to be the most promising, particularly when genetic algorithms were hybridized with another algorithm to accommodate the pitfalls of a single algorithm. There are far too many possible combinations of fuel elements in the standard PWR to be able to calculate the solution, even with modern high-performance computing resources, so machine learning and symmetry are used to reduce the number of required calculations. The standard PWR is able to utilize 1/8 symmetry reducing the computational load. The ATR does not need to utilize such symmetry because the ATR only has 40 fuel elements. Symmetry would also not be an appropriate application for a fuel load optimization problem in the ATR because of the different lobes having different requirements. A regular PWR will have one single job to produce power and will run at a uniform power across the reactor, the ATR has five lobes that all run at a different power. Another big difference in the ATR is that the ATR holds experiments that could introduce substantial sinks or sources to surrounding fuel elements while the standard PWR does not test experiments and will only hold fuel elements. A fuel load optimization has not been completed for the ATR. The ATR is a unique reactor with a unique fuel supply. Being able to reduce the number of fresh fuel elements would substantially reduce the economic cost of the ATR.

3. Methodology

The first step in fuel load optimization for the ATR was data collection and analysis. It is important to analyze data to be able to recognize expected or desired patterns in data that the machine learning algorithm will hopefully replicate. The maximum available amount of data was ultimately collected to be pared down later during feature selection. The most important consideration for finding the data was whether the data was available for every cycle. If the data was not consistently available in every cycle, it could not be effectively used. Occasionally, some features in the dataset would have a missing value, in the case of a missing value within the feature, KNN imputation was used to include an appropriately close value to what the value would most likely be given similar points within the dataset. Another consideration that became important later was considering which features would be available to analyst prior to the cycle run. The data was then analyzed based on different statistics and plots, a process often called exploratory data analysis.

Once the data was collected and analyzed, the next step was to select which models may be appropriate. Because the overall goal is prediction, regression models would be seen as the best option. The selected models include linear regression, random forest, and neural networks. All models were run on INL's HPC system using a GPU and did not experience any significant delays. The neural network models took the longest to train overall, however the run time was only a few minutes so the overall cost was considered to be negligible.

The next step was choosing the correct train test split and features. Choosing both values ended up being an iterative process between finding the best train test split and finding the best combination of input/output features based on the R^2 data. Once the best features were found, the models were trained and the burnup data was calculated. The existing burnup data was them

compared to cycles 165A, 166A, 166B, 167A, 168A, 168B, and 169A. The seven cycles analyzed were the most recent cycles before the ATR shut off for CIC.

After the burnup was found, it was important to find a way to get an appropriate BOC ^{235}U content out of the burnup. The chosen methodology was to take an average of the five nearest neighbors based on the input features, with the final output being the corresponding BOC ^{235}U. The models were then processed with fuel elements 1-40 and an initial estimated fuel loading was established. The performance of all the models was analyzed and the best model was chosen for runs using the MC21 drum solver.

The use of the MC21 drum solver was an iterative process due to existing biases in the dataset. The dataset was fully based off real world data that is the only data available and has successfully been used in running the ATR for over 55 years, however, biases in the data will also reflect in the model output. The initial ^{235}U gram loading was scaled appropriately based on reported biases in HELIOS data since HELIOS was the code used for the 60+ day cycles. Since burnup is calculated in the ATR per lobe, if a model scaled a value above the maximum possible initial ^{235}U gram loading, the excess could be pushed into any of the other 8 elements in the lobe until the overall lobe maximum was reached.

4. Results & Discussion

4.1 Data Analysis

Data used in the machine learning algorithms was collected from past engineering documents used for core physics analysis. The burnup of fuel elements was calculated using PDQ until cycle 155B, where CPAs were converted to the more modern code, HELIOS. Likewise, MC21 will soon replace HELIOS for CPA. Data was collected from cycles 46A through 164B, or 260 cycles. Cycles 165B through 169A were held back for testing the models. Of the cycles that were held back, 165A and 167A were PALM cycles and 166A, 166B, 168A, 168B, and 169A were regular cycles. Each cycle contains 40 entries, one for each fuel element, so the total dataset consists of 10,400 entries. Cycle 46A was chosen as the starting point because it was the first cycle in which the core loading files were available on the document management system. Cycles 1A through 24A were incomplete on atrfuel.inl.gov and potentially only partially loaded cores, so those cycles would not have been useful for the dataset and the unavailability of cycle 25A through 45B is not considered to have a significant effect on the outcome of the dataset. Cycles 46A through 164B ran between January 1980 through January 2019. Table 1 is the list of data collected for use at various points in the process.

Table 1 Definitions for Data Collected

Dataset ID	Definition
Cycle	Cycle ID
FEID	Fuel element ID that is assigned to the fuel element.
Position	Denotes the fuel element position in the ATR core denoted as a value from 1-40. While only being 1-40, the position marker contains implicit data that is important in the models.
MWd_Cycle	The MWd that ran over the given cycle. Usually given on the internal atrfuel.inl.gov, but when those values were not available was calculated by multiplying the cycle lobe power by the cycle length.
MWd_Prev	The burnup, in MWd, that a fuel element had experienced prior to the given cycle.
BOC_U235	Beginning of cycle ^{235}U in grams.
BOC_B10	Beginning of cycle ^{10}B in grams.
EOC_U235	End of cycle ^{235}U in grams. The most recent end of cycle ^{235}U is listed on atrfuel.inl.gov, otherwise the BOC ^{235}U from the following cycle was assumed to be the EOC ^{235}U from the previous cycle.
EOC_B10	End of cycle ^{10}B in grams. The most recent end of cycle ^{10}B is listed on atrfuel.inl.gov, otherwise the BOC ^{10}B from the following cycle was assumed to be the EOC ^{10}B from the previous cycle.
Lobe_Power	Lobe power, in MW, for the cycle. Fuel elements 2-9 are the NE lobe, fuel elements 12-19 are the SE lobe, fuel elements 22-29 are the SW lobe, fuel elements 32-39 are the NW lobe, and fuel elements 1, 10, 11, 20, 21, 30, 31, and 40 are the center lobe.
Core_Power	The sum of the lobe powers.
Cycle_Length	The number of days a cycle ran.
MWd_total	The sum of MWd_Prev and MWd_Cycle.
BU_U235	The grams of ^{235}U burnt over a cycle.
BU_B10	The grams of ^{10}B burnt over a cycle.
Status	Boolean. 0 represents a fuel element that is no longer available for use in further cycles, and 1 represents a fuel element still available for use in further cycles at the end of that given cycle.

A feature notably absent in the dataset is the experiment reactivity. Available experiment reactivity was not used due to an overall lack of data going back as far as the dataset went. Often, experiments that were available would have either been notional or restricted. Using the experiment data from a third party company could also be problematic as permission would need to be granted to report on any notable outcomes. However, it can be assumed that since the ATR

has run for so long, the effects of the experiment loading would be implicitly included in the dataset by the changes in fuel loadings, power splits, or more.

Most data was able to be retrieved from various engineering documents, but when some points were missing, KNN-imputation was used to replace missing values in the dataset with an appropriately close value. Table 2 describes how many instances of missing data were in the dataset for each analyzed feature.

Table 2 Number of Missing Elements per Feature

Feature	Number Missing Values	Percent of Dataset
Cycle	0	0.0
Position	0	0.0
FEID	0	0.0
MWd_Cycle	0	0.0
MWd_Prev	7	0.067
MWd_total	7	0.067
BOC_U235	4	0.038
BOC_B10	3	0.029
EOC_U235	6	0.058
EOC_B10	6	0.058
BU_U235	10	0.096
BU_B10	9	0.087
Lobe_Power	0	0.0
Core_Power	0	0.0
Cycle_Length	0	0.0
Status	0	0.0

Once the missing values were identified, the non-numerical features, FEID and Cycle, were removed from the dataframe and the numerical features were scaled by the min-max scaler in scikit-learn. Once the data was normalized, the KNN-imputer was able to find appropriate replacement values based on the nearest five values to the missing NaN datapoints, and the dataset is run through an inverse transform to return all features to their nominal, unscaled values. A check of the new dataset confirms that no missing values remain.

Figure 9 shows the relationship between the MWd run over a single cycle versus the corresponding ^{235}U burnup. The relationship between the burnup of ^{235}U and MWd is largely linear.

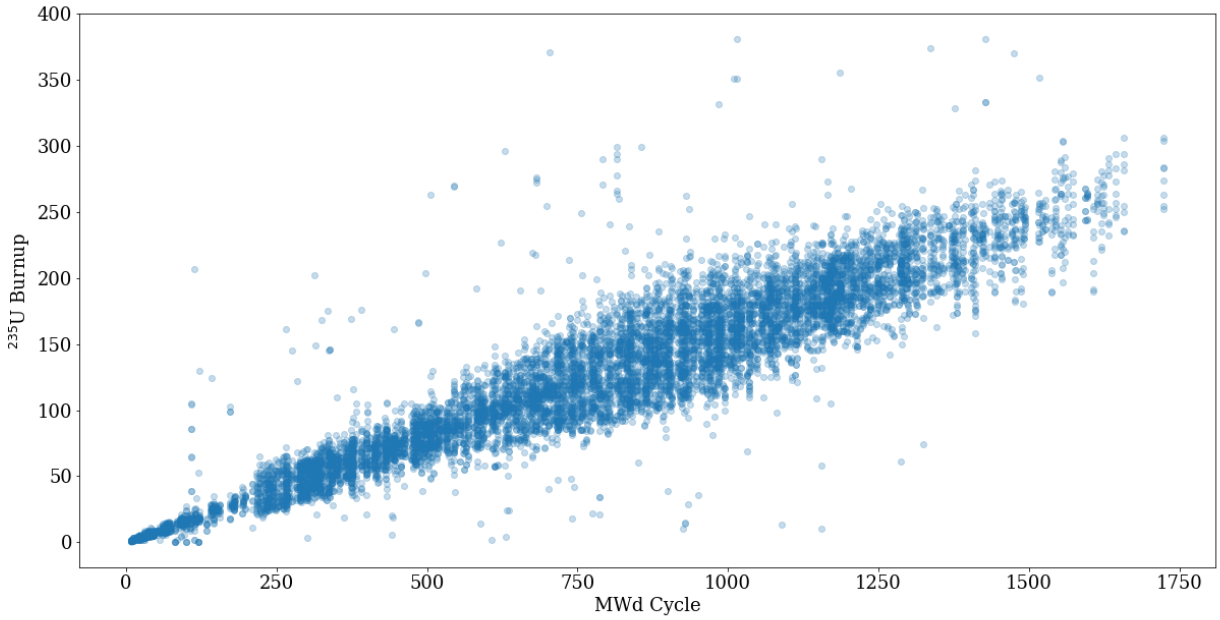

Figure 9 Cycle MWd vs. ^{235}U burnup for all elements in dataset.

Figure 10 shows the fuel element end of life in MWd. The average lifetime burnup of a fuel element is 2382 MWd with a standard deviation of 475 MWd. As can be expected from the linear relationship between burnup in MWd and burnup of ^{235}U, the histogram of fuel mass at the end of a fuel elements life is essentially the inverse of the total MWd run at the end of the fuel elements life. The average end of life ^{235}U content is 683 grams ^{235}U with a standard deviation of 68 grams. The disposal of very low burnup fuel elements is usually caused by damage to fuel element that prevents the element from being used again.

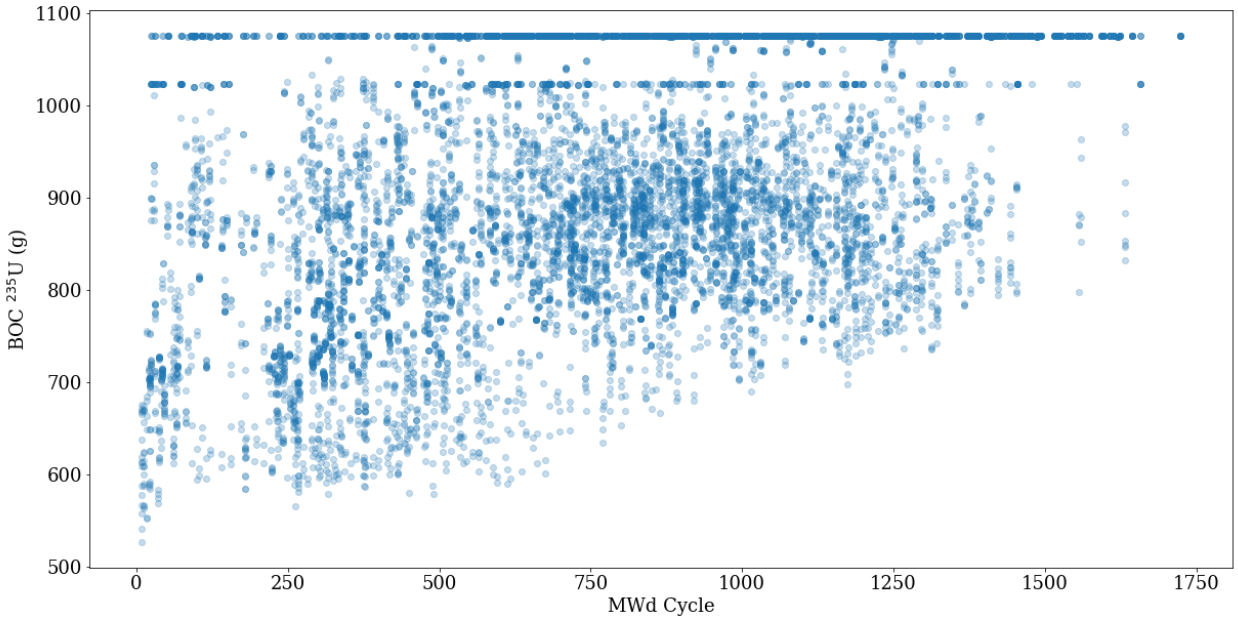

Figure 10 Comparison of the BOC ^{235}U content to the MWd of the cycle the fuel element ran in.

The maximum cycle MWd that did not utilize a fresh fuel element was 1559 MWd. Maximum overall cycle MWd was 1724 MWd. Non-fresh fuel elements begin appearing more frequently at MWd that are less than 1450 MWd. Occasionally a fresh fuel element is used for a PALM cycle to obtain the correct power split. Those values appear at between the 1075 g and 1022 g fresh fuel element lines. Cycles with a planned MWd lobe power of 750 or more will rarely be able to have fuel elements that begin at less than 700 g ^{235}U. The average cycle MWd is 792 MWd with a median value of 823.5 MWd.

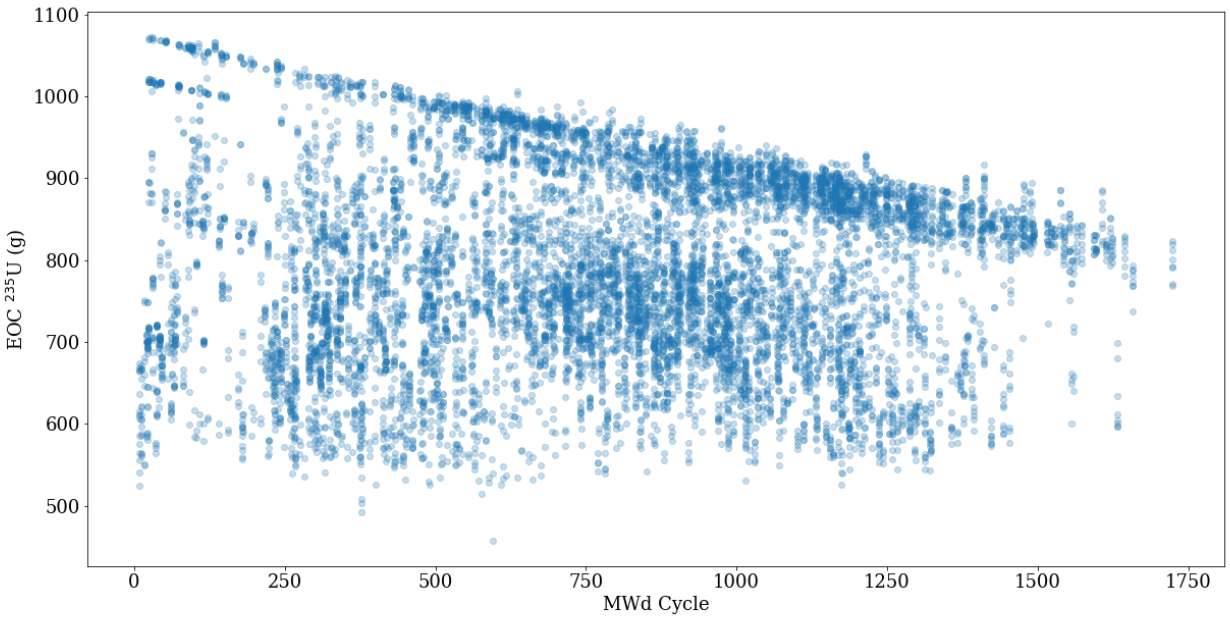

Figure 11 EOC ^{235}U content compared to the cycle MWd.

EOC ^{235}U was either listed as the EOC value as listed on atrfuel.inl.gov or, if the values were listed for a fuel element not on its most recent run, the EOC value was assumed to be the BOC value for the same fuel element in the next cycle that element was used in. There is a downward trend in the EOC ^{235}U content as the cycle lobe power in MWd increases. The reason for the downward line is twofold. First, higher MWd implies related to higher burnup. Second, very low MWd cycles correspond to PALM cycles which have aggressive power splits for a very low period and in order to reach said power split, fresh fuel elements are required despite low overall burnup. 560 g ^{235}U is approximately the minimum EOC ^{235}U content before recycling while the average is 683 g ^{235}U. The average value includes fuel elements that were damaged prior to reaching the maximum burnup potential, so it is slightly high overall.

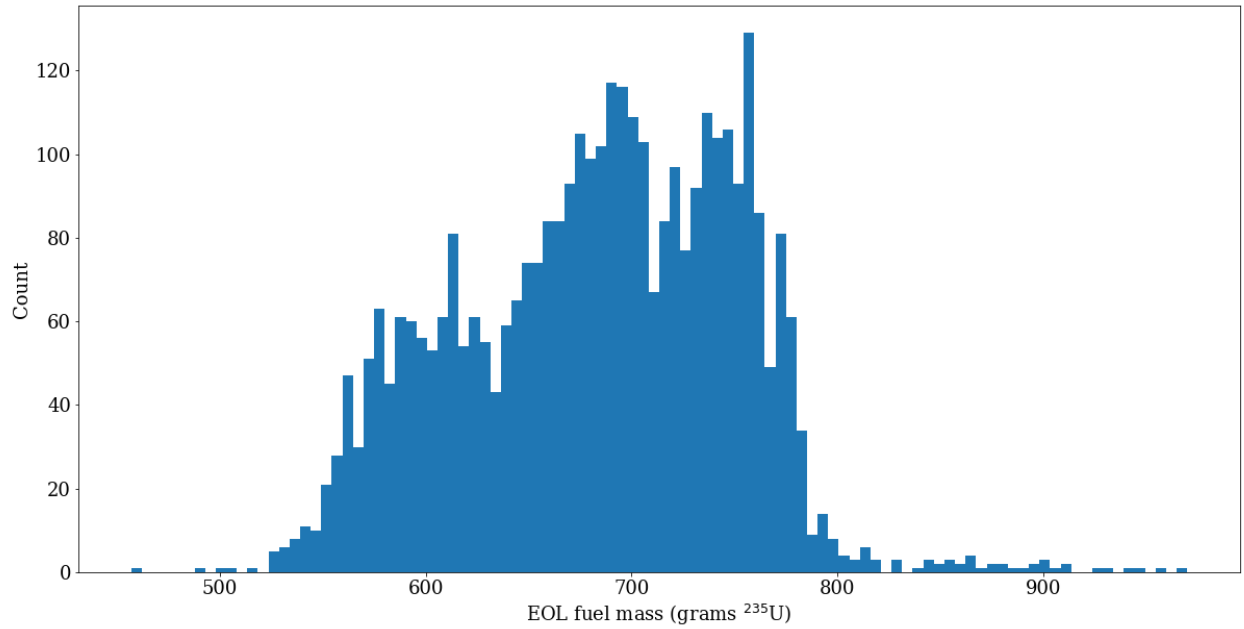

Figure 12 Histogram of ^{235}U content at fuel element end of life (EOL).

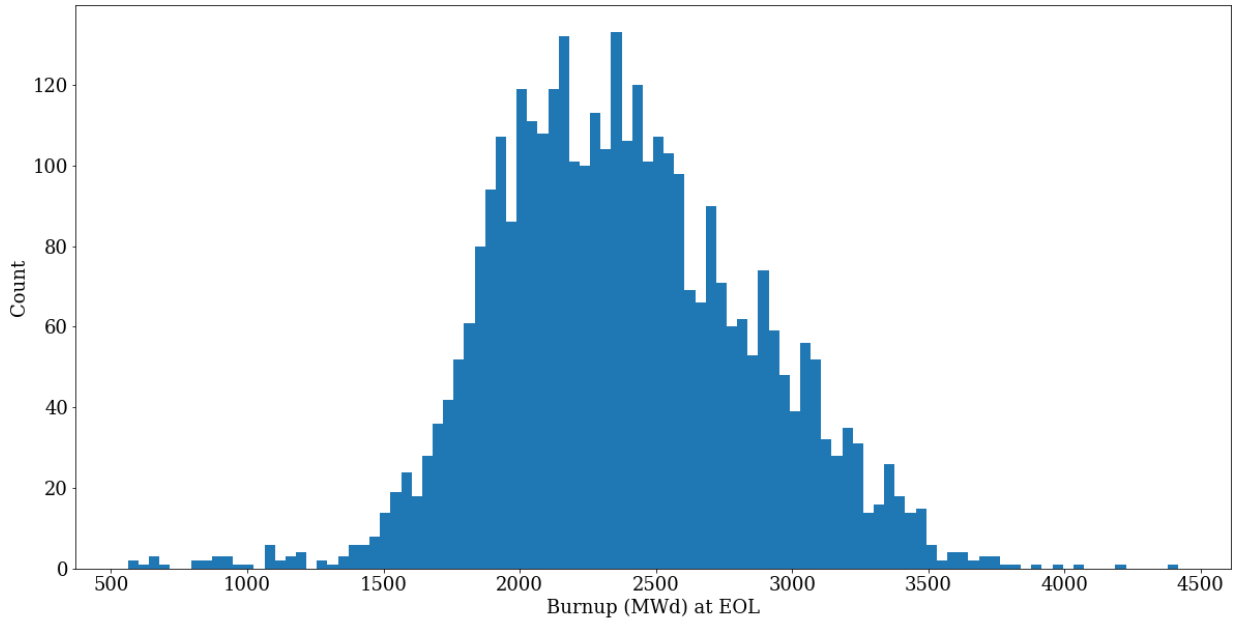

Figure 13 Histogram of MWd burnup at fuel element EOL.

While 10,400 burnup entries is the most that is reasonably available for the ATR, the total dataset is fairly small for a machine learning problem. PDQ and HELIOS are both two dimensional codes while MC21 is three dimensional.

4.2 Train-test Split and Feature Selection

The train test split is an important factor in machine learning modules. The train-test split is a percentage split of the dataset that represents the training data and the testing data. It is important to separate training and testing data to prevent overfitting by not testing the models on exactly the data that the models have already been trained on. The train/test split was set using feature inputs of position, MWd cycle, lobe power, core power, and cycle length. There was not a significant discrepancy between using those four features vs training the model on the highest 5 features, so it was assumed that those values would be similar.

A variety of train test splits were looked at for all three models to see which test-train split resulted in the best R^2 value without the model becoming overfit. A variety of possible combinations for the output parameters was also analyzed to be sure to pick the best performing features. Output features considered were BOC ^{235}U only, BOC ^{10}B only, BOC ^{235}U and BOC ^{10}B combined, burnup ^{235}U only, burnup ^{10}B only, and finally burnup ^{235}U and burnup ^{10}B combined. Overall, some form of ^{235}U was required to be a part of the output, but ^{10}B values were not. ^{10}B values were included in the prediction because, if the models performed well when predicting ^{10}B, then the result could be used in determining fuel element type.

Table 3 shows the R^2 values for the linear regression model. In order to get reproducible results over different runs, the random state was set to 1 for all the models. Overall, the R^2 value performed best for the burnup ^{235}U only with the best score being for 75% of the dataset being used for training the model and 25% of the dataset left for testing the model. Since the linear regression model is the simplest model, it cannot effectively separate based on the position, so the models peak at R^2 approximately equal to 86.6%. The next best performing model used an output of BU ^{235}U and BU ^{10}B, then BOC ^{235}U, BOC ^{235}U and BOC ^{10}B, BU ^{10}B, and finally BOC ^{10}B. It

is herd knowledge that HELIOS is a poor predictor of ^{10}B values and also the dataset includes fuel elements that cannot be ignored but also have 0 grams of ^{10}B at beginning of life, so the combination of those two factors results in skewed statistics from several 0 gram ^{10}B fuel elements that predictive models cannot statistically predict and therefor, poor predictive ^{10}B values.

Table 3 R^2 Values for the Linear Regression Model

Linear Regression							
		BOC ^{235}U, BOC ^{10}B	**BU ^{235}U, BU ^{10}B**	**BOC ^{235}U**	**BOC ^{10}B**	**BU ^{235}U**	**BU ^{10}B**
Train	**Test**	**Score**	**Score**	**Score**	**Score**	**Score**	**Score**
65	35	16.66%	43.45%	32.63%	0.32%	86.57%	0.69%
70	30	16.21%	43.27%	31.83%	0.27%	86.27%	0.59%
75	25	15.91%	43.44%	31.33%	0.20%	86.68%	0.48%
80	20	36.21%	65.68%	31.95%	44.76%	86.61%	40.47%
85	15	36.46%	66.01%	32.19%	45.76%	86.26%	40.72%
90	10	36.02%	66.00%	31.87%	45.71%	86.29%	40.17%
95	5	37.44%	66.42%	33.43%	46.19%	86.65%	41.44%

Table 4 shows the R^2 values for the random forest models. The random state was also set to 1 for the random forest runs to ensure reproducibility over different runs. Similar to the linear regression models, the random forest models also predicted burnup ^{235}U the best overall, followed by BOC ^{235}U, then BU ^{235}U and BU ^{10}B, then BOC ^{235}U and BOC ^{10}B, then BU ^{10}B, and finally BOC ^{10}B Again, the ^{10}B values were the worst predictive value.

The best R^2 value was again found to be a 75% training, 25% testing split on the data. For training splits above the 75%/25% mark, a slight drop in the values can be seen, followed by an increase again to the overall maximum value. When train test splits follow the behavior of a drop in the R^2 value followed by another increase later, the models are generally thought to be overfit. Therefore, the best value of a non-overfit model should be the highest before the initial drop in R^2 score.

Table 4 R^2 Value for Random Forest Model

Random Forest							
		BOC ^{235}U, BOC ^{10}B	**BU ^{235}U, BU ^{10}B**	**BOC ^{235}U**	**BOC ^{10}B**	**BU ^{235}U**	**BU ^{10}B**
Train	**Test**	**Score**	**Score**	**Score**	**Score**	**Score**	**Score**
65	35	32.29%	36.07%	75.99%	-5.95%	93.65%	-5.37%
70	30	35.77%	42.90%	76.69%	-2.62%	93.51%	-2.08%
75	25	36.30%	42.39%	77.25%	-0.87%	94.03%	-0.83%
80	20	-1689.07%	-4614.95%	77.75%	-11307.37%	93.69%	-1095.65%
85	15	-2629.17%	-3514.19%	78.51%	-7724.26%	93.55%	-1036.86%
90	10	-20.74%	80.58%	79.26%	-1890.73%	94.48%	-310.47%
95	5	-132.49%	69.35%	79.57%	-3334.19%	94.45%	-549.18%

Many of the R^2 scores across the random forest models are shown to be negative. As discussed earlier, negative R^2 scores represent a very poorly fit model. An R^2 value above 70% is considered to be desirable in terms of model fitting and for random forest, the BU ^{235}U, and the BOC ^{235}U both produced "acceptable" results via R^2 value.

The final model that was analyzed was the neural network. Due to the neural networks goal of replicating neurons in the brain, setting a random state value to ensure reproducibility is not an option, so all runs will likely be similar in results but not exactly the same. Table 5 shows the R^2 values for the neural network. Again, the best predictor was found to be a 75% train, 25% test split on the data. Again, the overfitting behavior can be seen for the training splits above 75%, especially on the non-BU ^{235}U categories where there is a significant jump between the 75%/25% and the following 80%/20% split. R^2 results for ^{10}B alone were again very poor, but close to 0%, which is still considered a very poor fit but is not as bad as a negative score.

Table 5 R^2 Value for the Neural Network Model

Neural Network							
		BOC ^{235}U, BOC ^{10}B	**BU ^{235}U, BU ^{10}B**	**BOC ^{235}U**	**BOC ^{10}B**	**BU ^{235}U**	**BU ^{10}B**
Train	**Test**	**Score**	**Score**	**Score**	**Score**	**Score**	**Score**
65	35	27.45%	46.82%	53.37%	0.39%	92.67%	0.80%
70	30	25.08%	46.24%	54.05%	0.29%	92.76%	0.61%
75	25	26.86%	46.15%	53.02%	0.17%	93.40%	0.32%
80	20	54.57%	75.14%	56.61%	63.08%	93.21%	55.62%
85	15	50.73%	79.25%	56.99%	64.76%	92.60%	52.37%
90	10	54.87%	76.43%	52.35%	61.60%	93.02%	52.68%
95	5	52.26%	78.77%	48.17%	61.90%	93.38%	55.50%

When considering results from all three models, it was determined that the best path forward was to run the machine learning models to predict burnup of ^{235}U and then to later find a way to tease out an appropriate BOC value out of the existing data. It was also determined that the 75%/25% train-test split was the best performing split that allowed for the highest R^2 value without the model becoming potentially overfit.

Once the best train-test split and output features were found based on all the features available, the process of feature selection was used to determine which combination of input features performed the best in the algorithms. Feature selection is an important part of machine learning. Some features may negatively affect the dataset while some may not affect the dataset at all, but their inclusion would slow town the computing time necessary. Features considered for the fuel load optimization input were based on information that would be available prior to a cycle and were position, cycle MWd for each lobe, individual lobe power (MW), total core power (MW), and cycle length. Table 6shows the R^2 value, as a percent, of the algorithms when analyzing the different features. If looking only at core position and lobe power, all three algorithms performed the worst. Random forest and neural networks were able to perform reasonably well even without power data by just having position and cycle length. Linear regression was largely stable and

performed the same with or without the inclusion of position, while the random forest regression and neural networks saw a nearly 10% reduction and a nearly 7% reduction in the R^2 value, respectively. Technically, the best performing linear regression of the cycle analyzed was position, MWd and cycle length, however none of those values are different enough from the 8 other linear regressions that also came in at 86.68%.

Table 6 Feature Selection Table

Position	MWd Cycle	Lobe Power	Core Power	Cycle Length	Random Forest %	Linear Regression %	Neural Network %
x	x	x	x	x	94.03%	86.68%	93.04%
	x	x	x	x	84.86%	86.68%	86.59%
x	x	x	x		93.92%	86.68%	93.29%
x		x	x	x	93.38%	78.69%	93.27%
x	x		x	x	94.10%	86.68%	93.30%
x	x	x		x	93.72%	86.68%	92.89%
x	x	x			93.59%	86.68%	92.93%
x	x		x		93.90%	86.68%	92.35%
x	x			x	93.77%	86.68%	92.94%
x		x	x		86.78%	26.78%	44.05%
x		x		x	93.15%	78.57%	92.73%
x			x	x	92.78%	66.51%	90.55%
x	x				92.50%	86.68%	92.18%
x		x			33.27%	0.44%	25.33%
x			x		65.77%	15.63%	35.03%
x				x	87.54%	62.15%	84.54%

For all three regression algorithms, it was found that including the individual lobe power had a negative effect on the R^2 value of the algorithm and was therefore not included in the algorithms. It was also found that for the random forest and neural networks, position was a necessary feature, while for the linear regression, the core position did not make a difference in the R^2 result. The position feature contains implicit data beyond the 1-40 that is listed in every cycle. The position can be used to denote which fuel elements belong in which lobe, thus separating

elements of similar powers but different lobes, and acknowledging the position also acknowledges the loading pattern of the dataset. Since linear regression only separates via hyperplanes, it does not have the same capability to acknowledge the implicit information in the fuel element position.

Once the best performing features were determined, the test-train split process was checked again with the selected features to ensure the train-test split and output features were still appropriate. Table 7 shows the R^2 values for the each analyzed train-test split and output feature combination for the linear regression algorithm. While some combinations of output features performed worse than the previously analyzed five features, ^{235}U burnup remained the best performer and resulted in the same R^2 value between removing the features.

Table 7 Train-Test Split R^2 Scores of Best Four Features for Linear Regression Models

Linear Regression - Four Features							
		BOC ^{235}U, BOC ^{10}B	**BU ^{235}U, BU ^{10}B**	**BOC ^{235}U**	**BOC ^{10}B**	**BU ^{235}U**	**BU ^{10}B**
Train	**Test**	**R^2 Score**	**R^2 Score**	**R^2 Score**	**R^2 Score**	**R^2 Score**	**R^2 Score**
65	35	11.48%	43.45%	23.96%	0.54%	86.58%	0.31%
70	30	11.85%	43.27%	23.25%	0.45%	86.28%	0.25%
75	25	11.51%	43.43%	22.66%	0.36%	86.68%	0.19%
80	20	27.93%	65.04%	23.48%	32.37%	86.61%	43.46%
85	15	28.54%	65.34%	24.08%	33.01%	86.27%	44.40%
90	10	27.90%	65.35%	23.43%	32.37%	86.30%	44.39%
95	5	28.02%	65.39%	24.01%	32.04%	86.66%	44.13%

Table 8 shows the R^2 value for each analyzed train-test split and output feature combination for the random forest model. Similar to the five-feature random forest model, most of the combinations of features perform incredibly poorly. R^2 can be as low as -infinity, however, anything negative is considered to be an awful fit in a model. Ideally, an appropriately fit model would have an R^2 value over 70%. Still, ^{235}U burnup performed well, and peaked without

becoming overfit at the 75%/25% split. The peak R^2 value of 94.1% is also incrementally better than the 94.03% seen with five features.

Table 8 Train-Test Split R^2 Values for Best Four Features for Random Forest Models

Random Forest - Four Features							
		BOC ^{235}U, BOC ^{10}B	**BU ^{235}U, BU ^{10}B**	**BOC ^{235}U**	**BOC ^{10}B**	**BU ^{235}U**	**BU ^{10}B**
Train	**Test**	**R^2 Score**	**R^2 Score**	**R^2 Score**	**R^2 Score**	**R^2 Score**	**R^2 Score**
65	35	35.53%	31.93%	74.94%	-6.91%	93.64%	-16.13%
70	30	36.75%	46.00%	75.64%	-5.01%	93.53%	-7.86%
75	25	37.66%	46.45%	76.22%	-2.89%	94.10%	-7.55%
80	20	-1225.45%	-3293.90%	77.23%	-426.70%	93.78%	-3672.33%
85	15	-1964.33%	-2220.71%	77.37%	-471.58%	93.55%	-1108.97%
90	10	-323.23%	63.17%	78.26%	-120.53%	94.45%	-137.27%
95	5	-627.76%	78.92%	79.32%	-1053.99%	94.33%	-1286.99%

Finally, Table 9 shows the R^2 value for all analyzed output features and train-test splits for the neural network models. In some of the cases, such as BOC ^{235}U, the ideal train-test split occurs at 70%/30% instead of 75%/25%, however, again, ^{235}U burnup far outperforms the other output feature options and the peak performance without becoming overfit can be seen using the 75%/25% split.

Table 9 Train-Test Split R^2 Results for Best Four Features for Neural Network Model

Neural Network - Four Features							
		BOC ^{235}U, BOC ^{10}B	**BU ^{235}U, BU ^{10}B**	**BOC ^{235}U**	**BOC ^{10}B**	**BU ^{235}U**	**BU ^{10}B**
Train	**Test**	**R^2 Score**	**R^2 Score**	**R^2 Score**	**R^2 Score**	**R^2 Score**	**R^2 Score**
65	35	29.66%	46.71%	54.85%	0.78%	92.99%	0.39%
70	30	26.97%	46.63%	53.88%	0.58%	92.93%	0.34%
75	25	26.35%	46.89%	52.10%	0.33%	93.06%	0.20%
80	20	55.78%	79.27%	55.89%	53.26%	92.78%	63.94%
85	15	52.23%	75.06%	57.11%	51.34%	92.79%	63.50%
90	10	52.96%	74.42%	53.42%	50.98%	93.46%	60.63%
95	5	53.27%	77.80%	52.88%	51.66%	93.74%	58.66%

A complete study was performed to choose the best combination of both input and output features and to choose the best train-test split. The best train-test split was found to be separating the dataset into 75% of the dataset is for training, while 25% of the data is for testing. The best output feature of the possible options was found to be BU ^{235}U alone for all models analyzed. The best combination of the available input features was found to be a combination of position, cycle MWd, total core power, and cycle length in days. Ultimately, the models were never able to fit to the ^{10}B data reasonably, so ^{10}B is not a viable option and should not be used in the models.

4.3 Burnup Prediction

The burnup of each fuel element was calculated per cycle and compared to the burnup calculated by HELIOS as part of the as run calculations. To ensure the closest possible measurements and to compare to the best of the ability of the models, the input parameters to the models used the as run information from AtrFuel.inl.gov. Cycles 165A and 167A were the two PALM cycles analyzed. PALM cycles are generally two weeks in length or less and have more aggressive power splits. The fuel loading in PALM cycles is not burnup dependent as they are short enough to have low burnup. Fresh fuel elements and lower burnup fuel elements are used in PALM cycles to achieve high power splits in certain lobes when necessary.

4.3.1 165A Burnup Prediction

Cycle 165A is the first of two PALM cycles analyzed. The input parameters are shown in Table 10. Often, PALM cycles will have a lower power split followed by a higher power split and the cycle will be split into two parts. Cycle 165A was split into two parts due to an extended outage mid-cycle, but did have a consistent power split, so the only change necessary to condense the PALM into one cycle is to sum the 165A-1 and 165A-2 cycle MWds to get the total.

Table 10 Power Splits and Cycle Length for 165A

Lobe	Element Positions	Cycle MWd	Lobe Power (MW)	Total Core Power (MW)	Cycle Length (days)
NE	2-9	51+206	19.2	144.9	13.4
SE	12-19	91+347	32.8	144.9	13.4
C	1, 10, 11, 20, 21, 30, 31, 40	79+327	30.4	144.9	13.4
SW	22-29	111+472	43.6	144.9	13.4
NW	32-39	51+201	18.9	144.9	13.4

Burnup prediction vs calculated burnup is shown in Figure 14. Linear regression performed the worst due to linear regression models being incapable of distinguishing shape behavior from position data. Random forest and neural networks performed similarly overall.

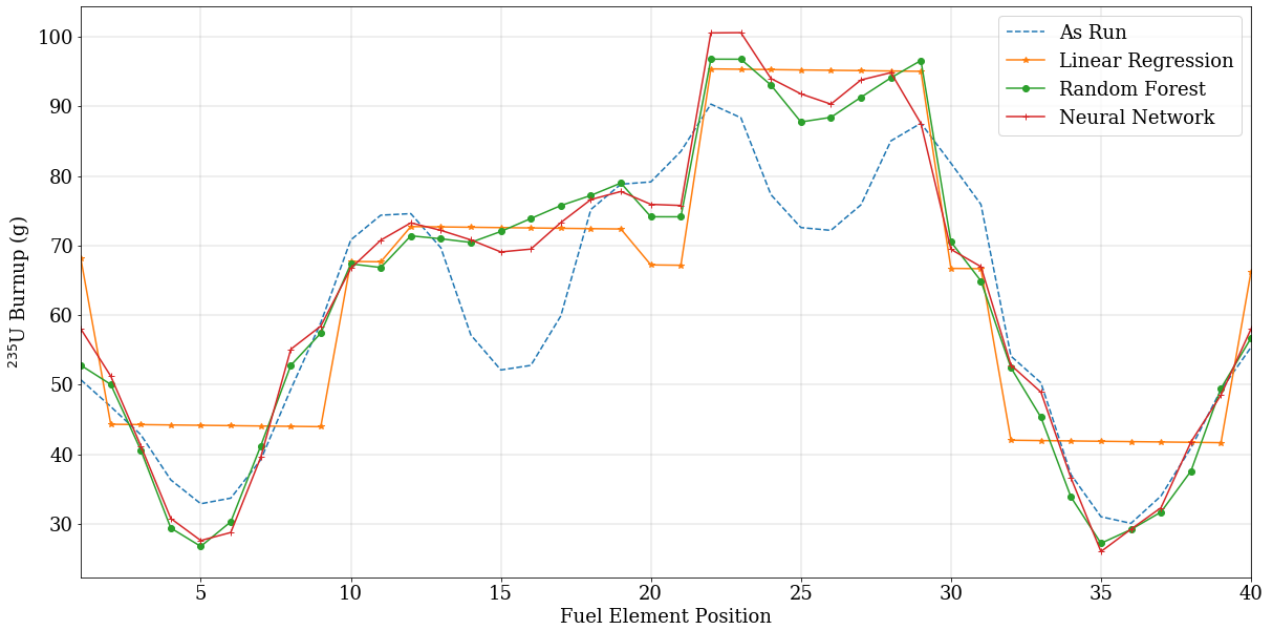

Figure 14 Actual vs. machine learning model predicted burnup over 40 fuel elements for cycle 165A.

Figure 15 shows the percent deviation from the as run data. All three models tested performed almost identically in deviation from the as run data in the SW and SE lobes. A percent deviation of zero represents a prediction that is identical the value predicted by the as run analysis. The

existing as run analysis predicts burnup between 30.1 g ^{235}U and 90.3 g ^{235}U, so the relatively high percent deviation from the as run data only amounts for being a few grams off of the as run data.

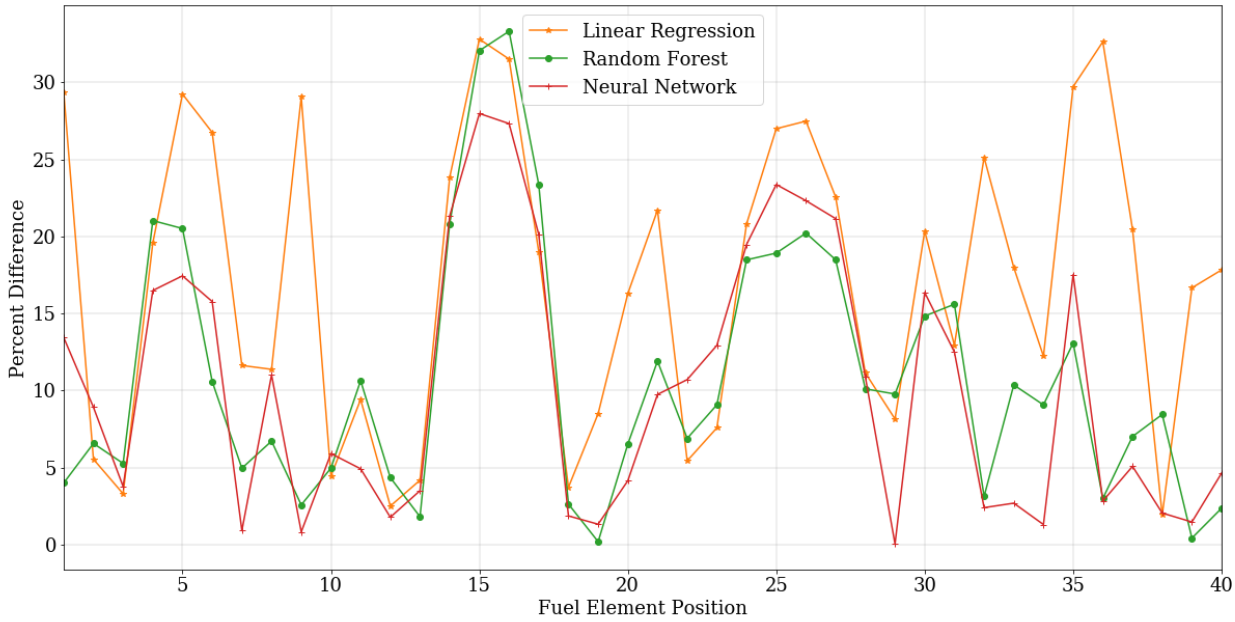

Figure 15 Absolute percent difference between the machine learning predicted burnup and the as run burnup for cycle 165A.

The linear regression model had a minimum absolute deviation from zero of 1.94% and a peak absolute deviation from zero of 32.79% with the average deviation from zero being 17.05%. The random forest model had a minimum absolute deviation from zero of 0.21%, a maximum absolute deviation from zero of 33.30% and an average absolute deviation from zero of 10.86%. The neural network model had a minimum absolute deviation from zero of 0.06%, a maximum absolute deviation from zero of 27.97%, and an average absolute deviation from zero of 10.21%. Neural networks performed marginally better overall for cycle 165A, but random forest did not produce an appreciably different result. Linear regression performed the worst, but was often close to the same deviation as the random forest and neural network models.

4.3.2 166A Burnup prediction

Cycle 166A was the first full length, non-PALM cycle analyzed. Table 11 shows the input parameters for cycle 166A. The output parameter was burnup ^{235}U (g). The difference in cycle length can be attributed to existing rounding assumptions made since the cycle length here was established as the Cycle MWd divided the lobe power (MW). The lobe power feature was only used to calculate the cycle MWd and not used in the model predictions.

Table 11 Power Splits and Cycle Length for 166A

Lobe	Element Positions	Cycle MWd	Lobe Power (MW)	Total Core Power (MW)	Cycle Length (days)
NE	2-9	1055	16.9	110.1	62.4
SE	12-19	1603	25.6	110.1	62.6
C	1, 10, 11, 20, 21, 30, 31, 40	1370	21.9	110.1	62.5
SW	22-29	1610	25.8	110.1	62.4
NW	32-39	1244	19.9	110.1	62.5

Figure 16 and Figure 17 compare the predicted values from each of the models to the burnup calculated from the HELIOS as runs. Figure 16 shows the total burnup or total predicted burnup in grams of ^{235}U. The lack of spatial awareness of the linear regressions model shows that the linear regression model either dramatically over predicts burnup in most fuel elements.

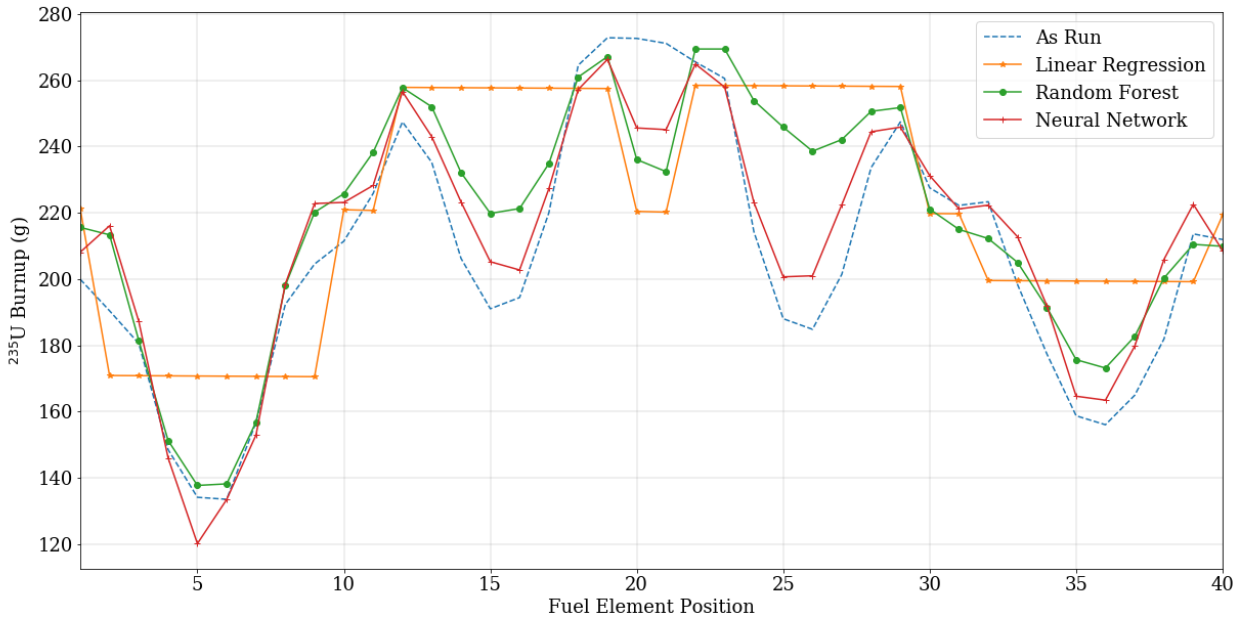

Figure 16 Burnup over fuel element position for predictive models and as run data for cycle 166A.

The dramatic over/under prediction of the linear regression model is also shown in Figure *17* with a maximum percent difference of 33.15% and a minimum percent deviation from 0 of 0.59% with the average absolute percent deviation from zero of 13.32%.

Random forest was the second-best predictor of burnup and had a maximum absolute percent deviation of 26.65%, a minimum absolute percent deviation of 0.08% and an average absolute percent deviation of 7.62%. Neural networks performed the best with an average absolute percent deviation of 4.91% with a maximum absolute percent deviation of 12.62% and a minimum absolute percent deviation of 0.04%.

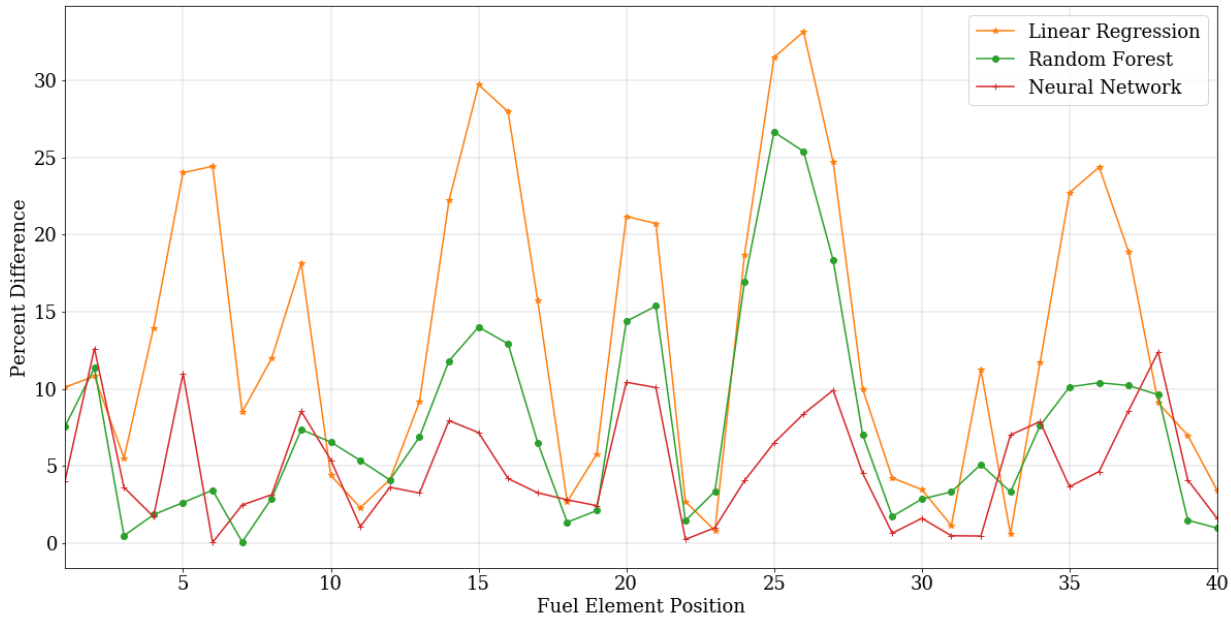

Figure 17 Absolute percent difference between the predicted burnup and the as run burnup for cycle 166A.

Overall, the linear regression performed the worst for cycle 166A with the highest average, maximum, and minimum percent deviation from 0, meaning that overall, at linear regressions best and worst performance, the linear regression model was the farthest away from the existing as run data. Neural networks performed the best overall with the lowest average, maximum, and minimum percent deviation from the existing as run data. The neural network performed nearly 2.7 times better on average than linear regression and nearly 1.5 times better than the random forest model. Neural networks maximum deviation from the as run model of 12.62% was also less than the average deviation from the as runs as predicted by the linear regression model. Also, the maximum deviation of 12.62% for the neural network was greater than half the maximum deviation for the random forest model and the linear regression model.

4.3.3 166B Burnup Prediction

Cycle 166B was also a full length, non-PALM cycle. Table 12shows the input parameters used in the prediction of the burnup.

Table 12 Power Splits and Cycle Length for 166B

Lobe	Element Positions	Cycle MWd	Lobe Power (MW)	Total Core Power (MW)	Cycle Length (days)
NE	2-9	1034.39	16.9	109.8	61.2
SE	12-19	1528.59	25	109.8	61.1
C	1, 10, 11, 20, 21, 30, 31, 40	1355.4	22.1	109.8	61.3
SW	22-29	1586.13	25.9	109.8	61.2
NW	32-39	1221.19	19.9	109.8	61.4

Figure 18 and Figure 19 show the results of the burnup prediction for cycle 166B. Similarly to cycle 166A, the linear regression models overpredicts burnup in 25 out of 40 fuel elements and underpredicts burnup in the remaining 15 fuel elements. Unlike the previous cycle, the neural network model and the random forest model produced closer results but neural networks were still the superior model. Random forest and neural networks both overpredicted burnup in the SW and SE lobes, but overall, the random forest model performed notably bad when predicted the SW lobe, with results close to those of the linear regression model and significantly overpredicting burnup.

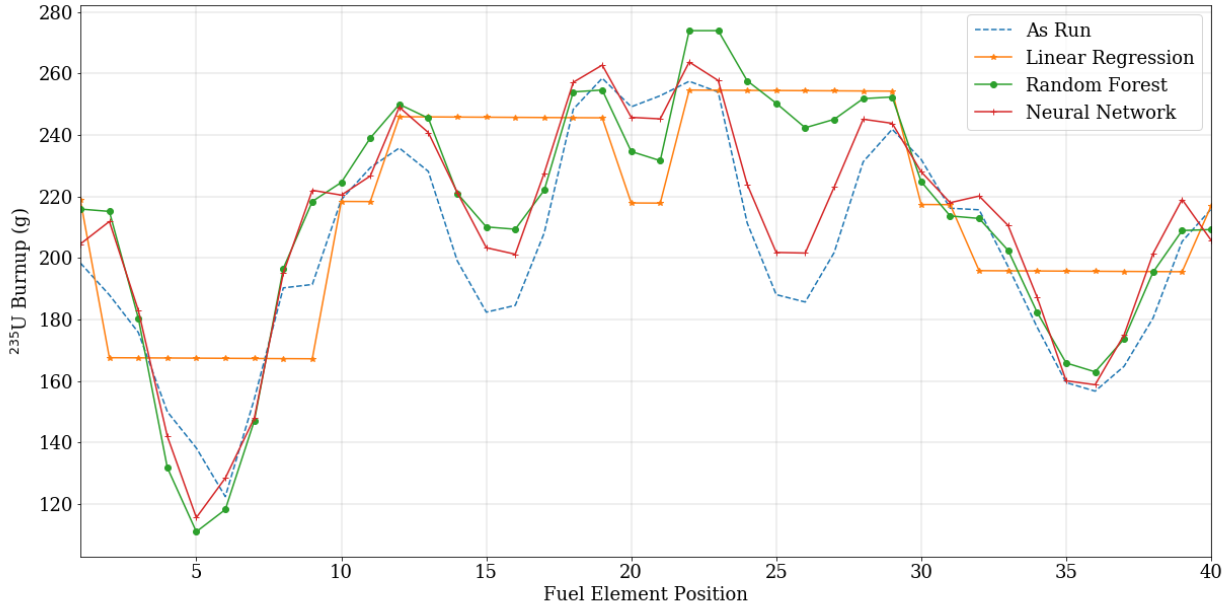

Figure 18 Burnup over fuel element position for predictive models and as run data for cycle 166B.

Prediction of the NE lobe performed the worst overall for all three models with the neural network performing the best at a percent deviation of 19.58% low at the maximum. Linear regression peaked in percent deviation of 25% or more in three of the five lobes. Random forest was able to perform slightly better than the neural network in the NW lobe only, however, random forest performed worse than neural networks in the other four lobes overall and performed substantially bad in the SW lobe.

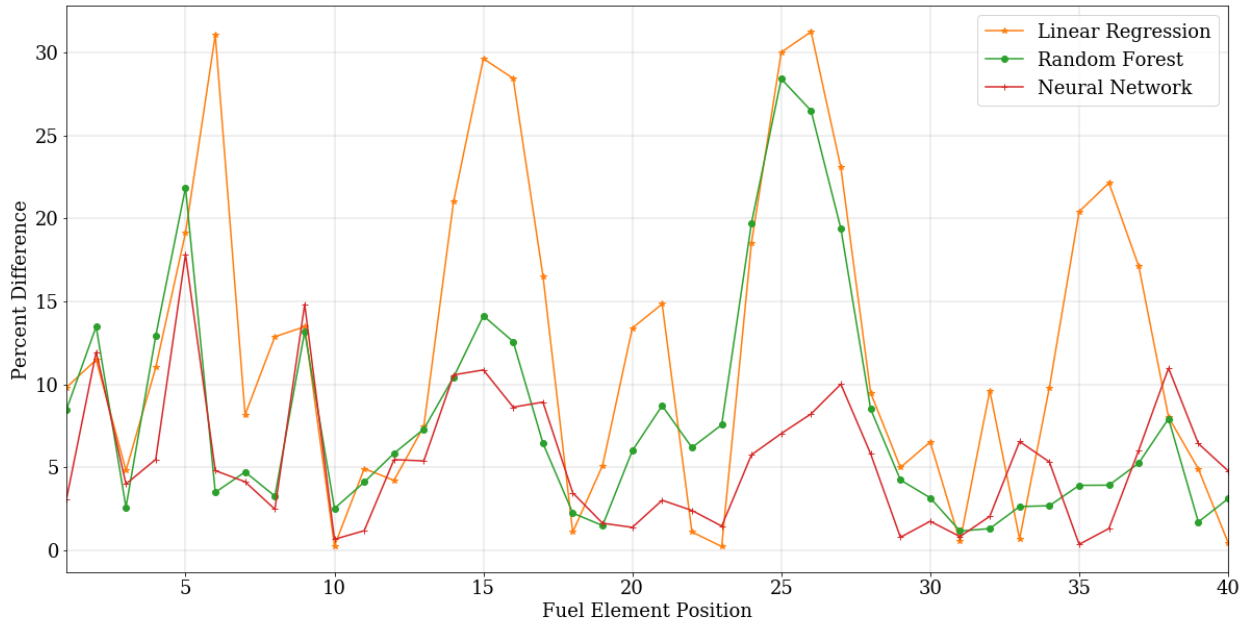

Figure 19 Absolute percent difference between the predicted burnup and the as run burnup for cycle 166B.

Neural networks performed the best overall, again, with an average absolute percent deviation from the as run burnup of 5.45%. Random forest and linear regressions had average absolute percent deviations of 8.08% and 12.19%, respectively. Linear regression did produce the closest result to the as run data at 0.24% difference from actual, followed by the neural network at 0.36% difference from actual, and finally, random forest at a closest comparison of 1.17% different from actual. Linear regression also produced fuel elements with the largest deviation at 31.24% from actual, followed by random forest at a peak deviation of 28.40%, and the neural network with a peak deviation of 17.84% difference. The peak neural network difference of 17.84% is also significantly higher than the deviations of the remaining 39 fuel elements with the second worst deviation being 14.83%. Comparatively, random forest had five fuel elements with a deviation from the as run data above 15%, while linear regression had 13 fuel elements with a deviation from the as run data above 15%.

4.3.4 167A Burnup Prediction

Cycle 167A is the second PALM cycle that was analyzed. Cycle 167A ran officially for five days at a low power split followed by two days at a higher power split. The average power splits were then condensed into a power split and cycle length that would be equivalent to the total MWd seen in the cycle. Table 13 shows the condensed split.

Table 13 Power Splits and Cycle Length for 167A

Lobe	Element Positions	Cycle MWd	Lobe Power (MW)	Total Core Power (MW)	Cycle Length (days)
NE	2-9	54.23	23.3	174.1	2.33
SE	12-19	118.78	51	174.1	2.33
C	1, 10, 11, 20, 21, 30, 31, 40	78.01	33.5	174.1	2.33
SW	22-29	97.25	41.8	174.1	2.33
NW	32-39	56.98	24.5	174.1	2.33

Figure 20 shows the predicted vs as run burnup for cycle 167A. The random forest model overpredicted burnup compared to the as run data significantly, with only two fuel elements in the NW having a lower predicted burnup than the as run data. The random forest model predicts a significant overestimate in the peak positions of the SW and SE lobes while also underpredicting burnup for most of the cycle. The neural network model also produces a visible outlier in fuel element 35 but is otherwise the closest to the as run data in value and shape.

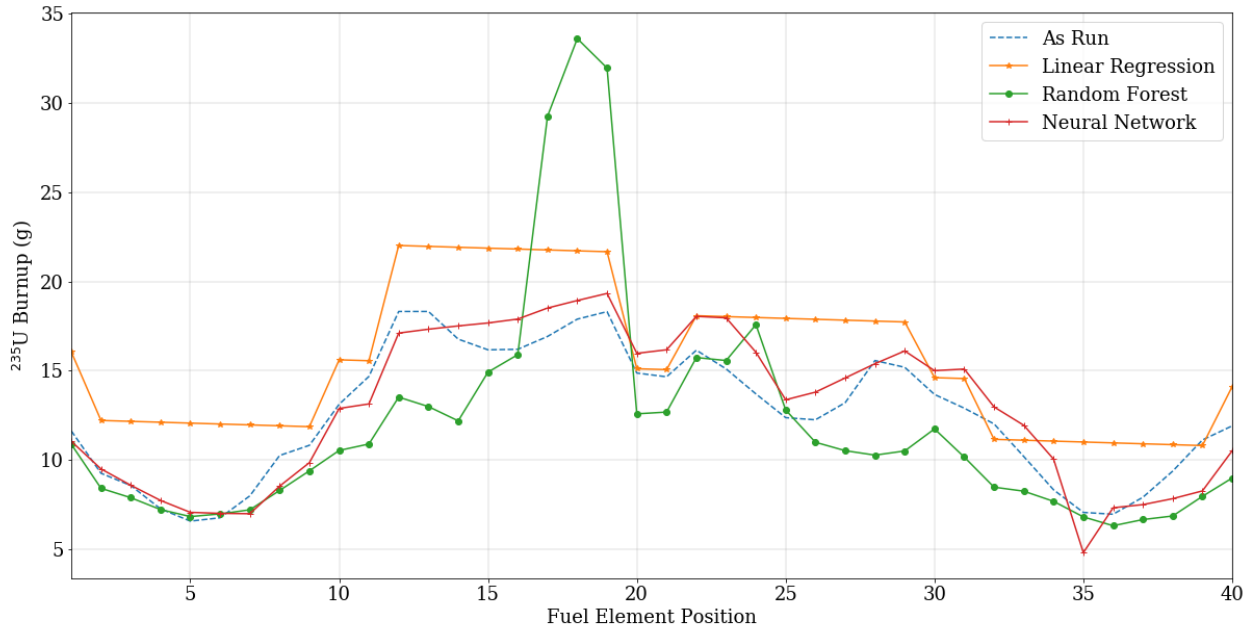

Figure 20 Burnup over fuel element position for predictive models and as run data for cycle 167A.

Figure 21 shows the percent difference from the as run data, with zero being an exact prediction of the as run burnup. Similar to cycle 165A, a higher percent difference in PALM cycles would be somewhat expected due to a lower range of deviation. Peak burnup for cycle 167A was less than 20 grams, so a 10% deviation would account for less than 2 grams ^{235}U predicted in burnup.

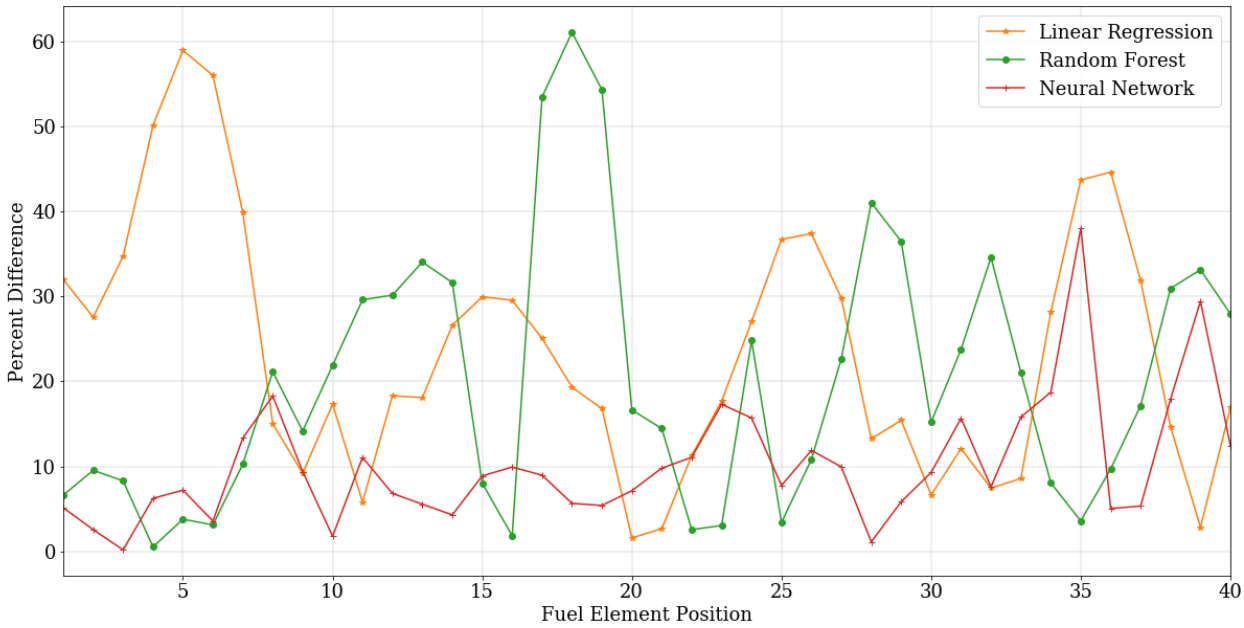

Figure 21 Absolute percent difference between the predicted burnup and the as run burnup for cycle 167A.

The linear regression model had a minimum absolute deviation from zero of 1.61%, a maximum absolute deviation from zero of 58.94% and an average absolute deviation from zero of 23.52%. The random forest model had a minimum absolute deviation from zero of 0.55%, a maximum absolute deviation from zero of 61.05% and an average absolute deviation from zero of 20.12%. The neural network model had a minimum absolute deviation from zero of 0.2%, a maximum absolute deviation from zero of 38.02%, and an average absolute deviation from zero of 10.19%. The neural network model had an average absolute deviation from zero that was approximately half of the other two and had the closest value overall to the as run burnup, therefor, the neural network has performed the best overall. Linear regression was the second-best performer. Random forest was the worst performer when predicting cycle 167A.

4.3.5 168A Burnup Prediction

Cycle 168A was a regular cycle. Table 14 shows the reactor data used to calculate the results.

Table 14 Power Splits and Cycle Length for 168A

Lobe	Element Positions	Cycle MWd	Lobe Power (MW)	Total Core Power (MW)	Cycle Length (days)
NE	2-9	1208.81	19.8	111.4	61.05
SE	12-19	1403.16	23	111.4	60.0
C	1, 10, 11, 20, 21, 30, 31, 40	1208.81	21.5	111.4	60.9
SW	22-29	1658.89	27.2	111.4	60.9
NW	32-39	1216.23	19.9	111.4	61.1

Figure 22 shows the predicted burnup for each of the models and the calculated as run burnup. The linear regression model can be seen overpredicting in the NE, NW, SE, and SW lobes, particularly on elements on the outer fuel elements in the lobe, and underpredicting burnup in the C fuel elements. Random forest and neural network produce comparable results in the SE and NW lobes and random forest performed the worst out of all three models in the SE lobe. The neural network model produced a better shape for the burnup vs element in the NE lobe but underpredicted in the outermost three elements of the NE lobe. Random forest, however, underpredicted burnup on the inner elements of the lobe.

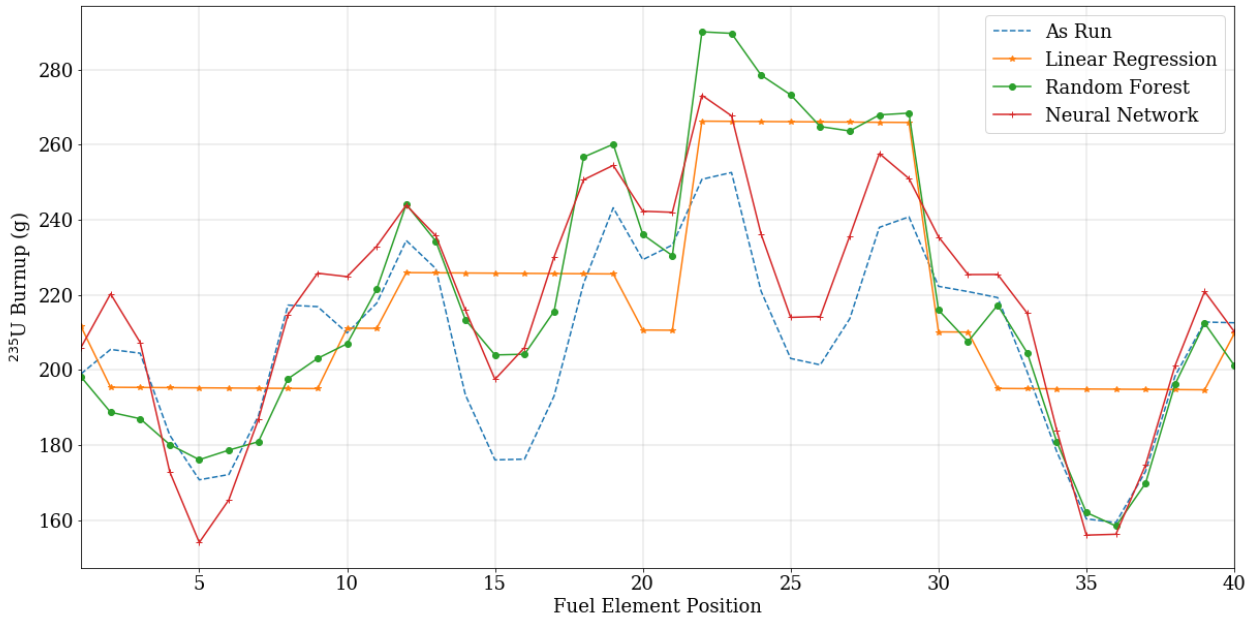

Figure 22 Burnup over fuel element position for predictive models and as run data for cycle 168A.

Figure 23 shows the percent deviation of each of the models from the as run data with 0% being a perfect prediction. Linear regression had the largest deviation overall, but the random forest model had a higher peak in the SW lobe. The neural network model produced the closest overall deviation from the as run data. The random forest model produced the best results in the NW lobe.

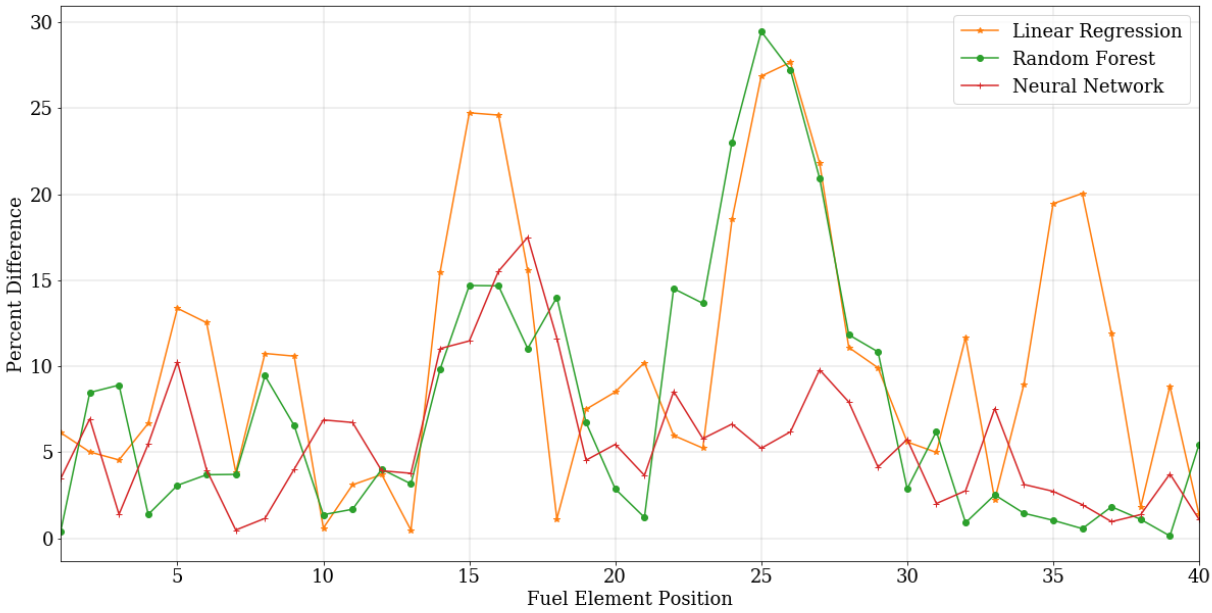

Figure 23 Absolute percent difference between the predicted burnup and the as run burnup for cycle 168A.

The linear regression model had an average deviation from the as run data of 10.33% with a minimum deviation of 0.48% and a maximum deviation of 27.67%. The random forest model had an average deviation from the as run data of 7.67%, with a minimum deviation of 0.15% and a maximum deviation of 29.46%. The neural network model had an average deviation from the as run data of 5.67% with a minimum deviation of 0.49% and a maximum deviation of 17.50%. Neural networks performed the best overall while technically having the worst nearest value to the actual data, however, the neural networks had the lowest overall average and maximum deviations from the as run data. The random forest model had the closest point to the as run data but ranks as the second-best model in cycle 168A given that the random forest model had a higher overall maximum. The random forest model had the second best average overall. The linear regression model ranks third because it has the highest overall average deviation and the second highest maximum value that is 1.58 times the maximum deviation seen in the neural network model.

4.3.6 168B Burnup Prediction

Predicted data for cycle 168B is given in Table 15. Due to the shorter cycle length that 168B was calculated over, the training data is likely more equipped to handle the inputs as 60+ day cycles are relatively new to the ATR and 45 days was the nominal cycle length for much of the ATR lifetime.

Table 15 Power Splits and Cycle Length for 168B

Lobe	Element Positions	Cycle MWd	Lobe Power (MW)	Total Core Power (MW)	Cycle Length (days)
NE	2-9	1136.63	19.8	106.3	57.41
SE	12-19	1312.14	22.8	106.3	57.55
C	1, 10, 11, 20, 21, 30, 31, 40	1266.06	22	106.3	57.55
SW	22-29	1355.81	23.6	106.3	57.45
NW	32-39	1042.38	18.1	106.3	57.59

Figure 24 shows the predicted versus calculated burnup. Both the random forest models and the neural network models floor the shape of the burnup curve closely while slightly overpredicting burnup. The random forest model has a peak in the western part of the SW lobe. The linear regression model again performed the worst overall due to the linear regression model being agnostic to fuel element position.

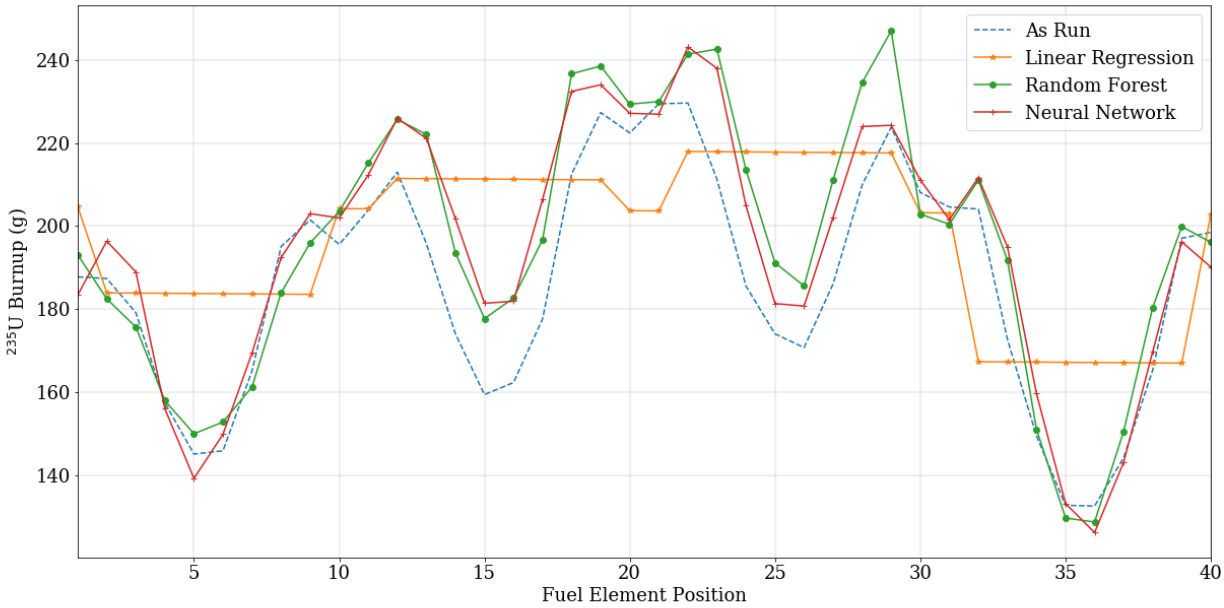

Figure 24 Burnup over fuel element position for predictive models and as run data for cycle 168B.

Figure 25 shows the percent deviation of each of the three predictive models from the predicted as run burnup. The linear regression model had large spikes in percent difference, again to be attributed to the model being agnostic to fuel element position. The random forest model and the neural network model show similar results, but overall it appears that the neural network performs slightly better.

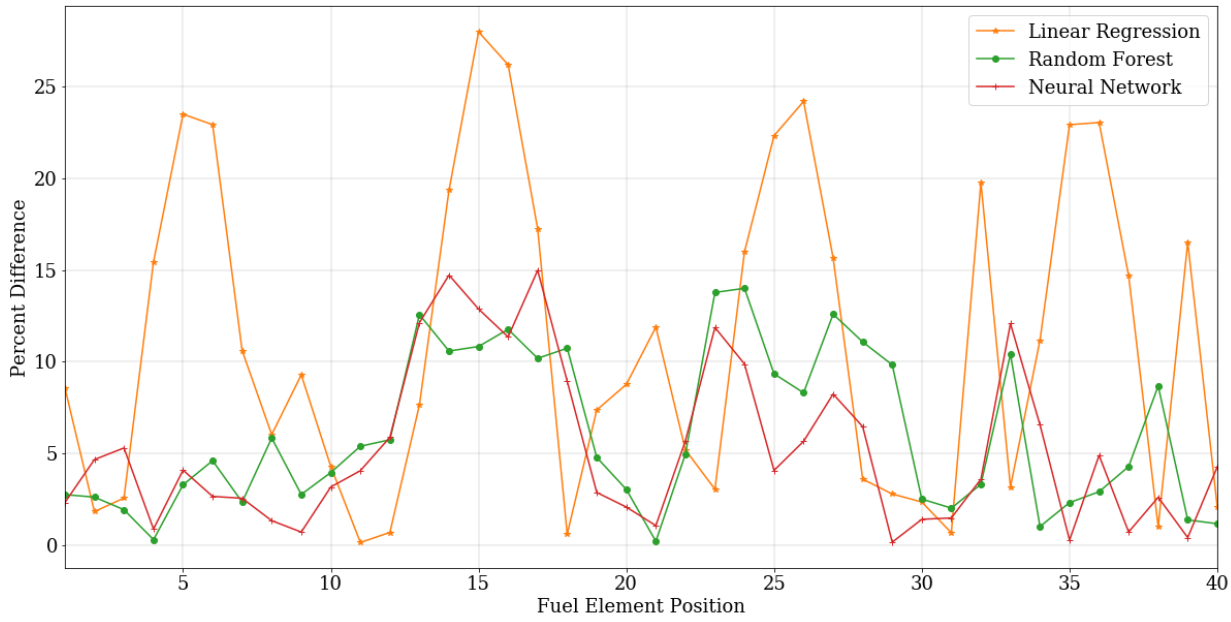

Figure 25 Absolute percent difference between the predicted burnup and the as run burnup for cycle 168B.

The linear regression model had an average deviation from the as run value of 11.08% with a minimum deviation from the as run value of 0.18% and a maximum deviation from the as run value of 27.95%. The random forest model had an average deviation from the as run value of 6.02% with a minimum deviation from the as run value of 0.24% and a maximum deviation from the as run value of 14.00%. The neural network model had an average deviation from the as run value of 5.24% with a minimum deviation from the as run values of 0.19% and a maximum deviation from the as run value of 14.97%. Random forest and neural networks result in comparable results with the neural network having slightly better performance on average and a slightly improved minimum deviation. The random forest model resulted in the lowest maximum value.

4.3.7 169A Burnup Prediction

Cycle 169A was a full-length cycle and the longest cycle analyzed. The data for cycle 169A is given in Table 16.

Table 16 Power Splits and Cycle Length for 169A

Lobe	Element Positions	Cycle MWd	Lobe Power (MW)	Total Core Power (MW)	Cycle Length (days)
NE	2-9	1267	19.99	106.83	63.38
SE	12-19	1458	23.01	106.83	63.36
C	1, 10, 11, 20, 21, 30, 31, 40	1347	21.26	106.83	63.38
SW	22-29	1433	22.61	106.83	63.36
NW	32-39	1265	19.96	106.83	63.38

Figure 26 shows the predicted burnup for each of the three models compared to the as run burnup. The linear regression model overpredicts in the NW, NE, SW, and SE lobes overall while underpredicting burnup in the C lobe. Occasionally, some of the outer lobe fuel elements that are near center will be underpredicted also. The random forest model mostly overpredicted burnup but did succeed in following the overall shape of the burnup. The random forest model also significantly overpredicted burnup in the peak burnup positions of the SE and SW lobes. The neural network model closely followed the actual as run burnup for most of the cycle.

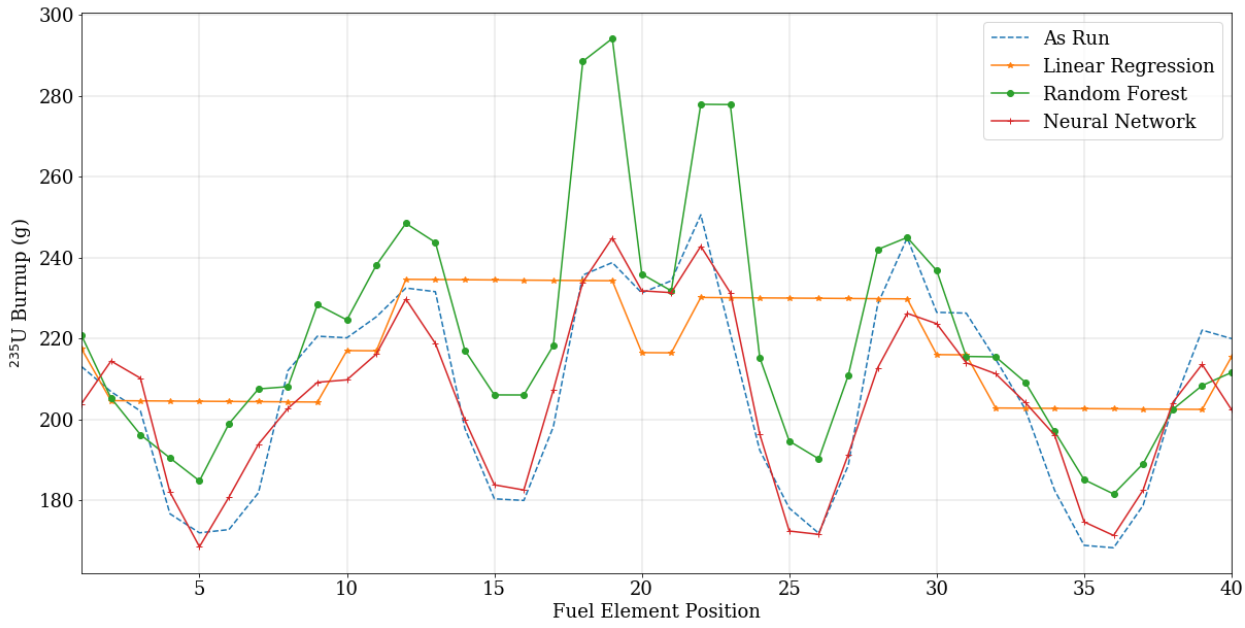

Figure 26 Burnup over fuel element position for predictive models and as run data for cycle 169A.

Figure 27 shows the percent deviation from the as run data for each of the predictive models. Random forest and linear regression performed comparably to each other, and neural networks performed the best by a large margin.

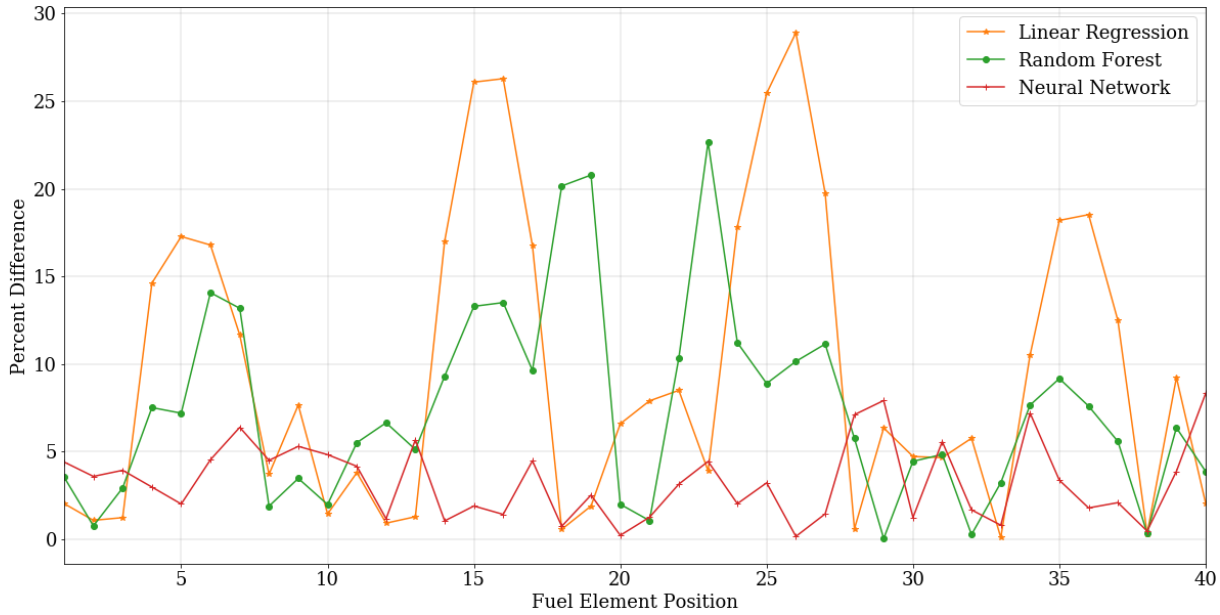

Figure 27 Absolute percent difference between the predicted burnup and the as run burnup for cycle 169A.

The linear regression model had an average percent deviation from the as run data of 9.61% with a minimum deviation from the as run burnup of 0.10% and a maximum deviation from the as run burnup of 28.90%. The random forest model had an average percent deviation of 7.43% with a minimum deviation from the as run burnup of 0.04% and a maximum deviation from the as run burnup of 22.66%. The neural network model had an average percent deviation from the as run burnup of 3.33%, with a minimum deviation from the as run burnup of 0.17% and a maximum deviation from the as run burnup of 8.33%. The neural network model had an average deviation of the as run burnup that was 2.89 times lower than the average deviation of the linear regression model and 2.23 times lower than the average deviation of the random forest model. The neural network model for cycle 169A performed the best overall for all cycles analyzed.

At this point, linear regression can be removed as a potential option due to the lack of spatial awareness and inability to see patterns across individual elements in the lobe rather than solving for one value for each lobe, leading to underfitting. Linear regression performed the worst in all the cycles analyzed in terms of both R^2 and burnup and without the ability to tell which elements peak in burnup and which elements have lower burnup for very similar input features, the model is not a viable option.

Random forest may also be removed as a potential option despite having the best R^2 value. In terms of functionally predicting burnup, the random forest model performed on the level of the linear regression models in at least one lobe in five of the seven cycles analyzed.

The neural network model will be used as a starting point for a fuel loading in MC21. The neural network performed the best overall in all seven cycles and was able to remain the closest to the actual predicted burnup.

4.4 Nominal Predicted Fuel Loading

Effectively creating a fuel load requires using cycle information available before the cycle runs. Comparative values were found in the individual cycles CPA on the internal EDMS. Table *17* shows the BOC total ^{235}U and the total number of fresh fuel elements for each cycle based on nominal data. In the calculations, any fuel element with ^{235}U loading of 1020 or greater was considered to be fresh. Occasionally, there will be fresh fuel elements with very low burnup from running once in a PALM cycle, but these fuel elements make up a statistically insignificant part of the dataset and may be considered fresh.

Table 17 Number of Fresh Fuel Elements in Each Analyzed Cycle

Cycle	Fresh Fuel Elements Used in As Run
165A	11
166A	26
166B	24
167A	10
168A	24
168B	6
169A	23

It was determined in the previous section that the neural network predicted the closest to the burnup calculated from PDQ or HELIOS as run data. The existing burnup data is the data of record and the use of the burnup data has aided in successfully running the ATR for more than 50 years, so the data in itself is not unusable, however, biases exist in the model that require the initial neural network output to be scaled. Since most of the cycles analyzed follow the 60-day or more cycle length that the ATR has only gone with in recent years, i.e. since the implementation of HELIOS over PDQ, the bias in the HELIOS data as it pertains to lobe power will be used to scale the models. Cycle 158A was the only 60-day cycle to use PDQ. Other potential uncertainties that may be considered when scaling would include an +/- 8.5% lobe power uncertainty from the N-16 system. Any overprediction or underprediction in existing lobe power would result in a corresponding overprediction or underprediction in burnup. An average value of the HELIOS lobe power errors from cycle 162A through cycle 164B [46]. Table 18 shows the average lobe power error +/- the 8.5% lobe power uncertainty to give a starting range. The minimum and maximum errors in lobe powers are also given in Table 18. Typically, keeping scaling within a range of the average error +/- the total lobe power uncertainty will envelope the possible error in the model, however, in some cases like the SE lobe, the maximum deviation from the actual lobe power was found to be 18.81% for a PALM cycle, which is significantly higher than the average error + 8.5

lobe power scaling factor of 4.27. It would not be typical to expect such a high deviation, but adjusting to that point under certain circumstances may be viable.

Table 18 Potential Error Range of Dataset

	NW	NE	C	SW	SE
Average - 8.5% Lobe Power Uncertainty	-15.97	-3.99	2.35	-13.98	-12.73
Minimum	-12.31	-10.54	9.11	-9.60	-13.71
Average	-7.474	4.51	10.85	-5.48	-4.23
Maximum	0.79	11.11	11.94	2.71	18.81
Average + 8.5% Lobe Power Uncertainty	1.03	13.01	19.35	3.02	4.27

The NW lobe was shown to be the most affected overall by HELIOS and underpredicts burnup on average of 7.5% without accounting for the N-16 system uncertainty and peaked at approximately 12% underprediction of lobe power. Therefore, the NW lobe could potentially be predicting values that are up to 20% low based on existing errors and uncertainties. Another potential issue with the NW lobe is not related to burnup as much as it is related to the lobe being a large reactivity sink, so more reactivity is required initially over a similar power in the NE lobe. The NE lobe is the only controlled lobe where HELIOS overpredicts lobe power on average of 4.5%. The C lobe is not explicitly controlled, and it drifts, therefore, in order to conservatively account for potential errors, the error was calculated instead for the N, E, S, and W lobes by taking the arithmetic mean of the error in the center and the two nearest controlled lobes. Averaging the three known lobes is generally how power is calculated for experiments in the N, E, W, and S lobes since there is not explicit monitoring there. Table 19 shows the results for the averaged errors.

Table 19 Error Range for Center Lobe

Lobe	Fuel Element ID	+8.5%	Average	-8.5%
N	1, 40	11.13	2.63	-5.87
E	10, 11	12.21	3.71	-4.79
S	20, 21	8.88	0.38	-8.12
W	30, 31	7.80	-0.70	-9.20

Note, in the cases of Table 18 and Table 19 the values are listed least conservatively to most conservatively and a positive value results in an overprediction of lobe power and a corresponding overprediction of burnup and a negative value results in an underprediction of lobe power and a corresponding underprediction of burnup. It would not be appropriate to attempt to account for the error during the prediction of burnup because the error is built into the dataset.

Another potential introduction of error into the set, even if burnup is overpredicted by the neural network is the use of KNN to extrapolate BOC ^{235}U content based on the burnup data. If the nearest neighbor is found to be within the HELIOS data, which is likely since the HELIOS data contains a majority of the instances of that burnup with that corresponding MWd and cycle length, the overall BOC data that the model is pulling could be skewed. If HELIOS is artificially underpredicting or overpredicting burnup, then the actual ^{235}U content of the fuel element would also be skewed, particularly in cases where fuel elements were initially used in the ATR at cycle 158A or later. There could be a few potential ways to handle the scaling issue in the dataset. Adding more MWd to the cycle is one of the potential solutions, however, if the cycle and power are increased too much then the model attempts to predict values that are outside the scope of the dataset and could default to entirely fresh fuel elements. Since most of the non-PALM cycles between 165A through 169A were modeled to produce at least 60 days of cycle length, adding on MWd to the model is inappropriate and will likely land outside of the range of the dataset. Another option is to scale the model at the burnup stage, which would prevent from having to rearrange

fuel mass at the end due to a limiting total amount of fuel that may go into any element or lobe. However, the pitfall of that methodology is scaling the data in the middle of two ML algorithms, which could unnecessarily complicate the process flow and also may result in an abundance of fresh fuel elements due to the new data potentially being out of the range of the dataset. Finally, and the methodology that was ultimately chosen involves taking the final predicted fuel loading and scaling that up or down based on the biases in the dataset. Doing so allows to easily add or reduce fuel element loading based on how well a cycle works without having to go through another tedious ML process. The NN/KNN methodology produces a good starting point based on the dataset, but likely need scaled based on errors within the data of record and the fact that every ATR cycle is unique and will require unique, engineering judgement, decisions that a ML model cannot handle. For example, when to use the non-borated NB fuel elements. Each cycle only contains one or two NB elements, at most, so predicting fuel element type to be used in the models is not viable. Similarly, only a few YA fuel elements are used per cycle, and are typically used to control power peaking. The cycles analyzed used predominately XA fuel elements, with some YA fuel elements used in certain position as defined as mandatory in the corresponding CPA for that cycle.

When creating a fuel loading, PALM cycles will be treated slightly different from non-PALM cycles. Due to the PALM cycles being at the bottom of the burnup curve, a good fuel loading starting point would be created just by adding on some extra days to a nominal power split. Increasing the MWd will subsequently increase predicted burnup and will create a good starting point. Increasing the cycle length for the 60 days runs very quickly forces the model to attempt to predict burnup measures on values that are higher than what exists in the dataset and will not produce good results, so the starting point for the full length, 60-day runs will be the predicted

value scaled using the appropriate average lobe power error. In the case of a fuel element exceeding the maximum allowable fuel content for that element, the excess will be added to other elements within the lobe because ATR burnup estimations are based on a total for the lobe and not the individual fuel elements [47]. The excess may be added until the maximum possible fuel loading is reached in that lobe, after which any excess will be ignored.

A common rule of thumb when using the drum solver is to take the average startup eigenvalue of the previous five cycles as the target eigenvalue stated in the API script, however, due to a drop in the eigenvalue between initial critical and the subsequent timestep, it was found that a more appropriate methodology may be to take the average of the last five startup eigenvalues for the current cycle startup and then to take the average, non-SCRAM, eigenvalue of the past five cycles. The reason for not using the startup eigenvalue is that in some cases, the reactivity difference between the average, non-SCRAM, eigenvalue and the startup eigenvalue exceeds the defined convergence parameters. The convergence parameter is set to be +/-0.0063 or +/-0.875β. Instead of expanding the convergence criteria, it was deemed more realistic to adjust the target eigenvalue to what would be expected out of the whole cycle. Table 20 shows the eigenvalues and the corresponding reactivity difference, in $, using the ATR β for a mixed core of 0.0072. Cycles 165A, 168A, 168B, and 169A all exceed the convergence parameters and would make convergence on the drum solver difficult.

Table 20 Comparison of BOC vs Cycle Average Eigenvalue

Cycle	Average Startup Eigenvalue of Past Five Cycles	Average of Past Five Cycle Average Eigenvalues	Cycle Average Eigenvalue	Reactivity Difference ($) Between Average Startup Eigenvalue and Cycle Average Eigenvalue
165A	1.0007	0.9984	0.9987	1.32
166A	0.9995	0.9977	0.9972	0.61
166B	1.0001	0.9977	0.9964	0.69
167A	1.0013	0.9977	0.9972	0.69
168A	1.0003	0.9972	0.9963	1.11
168B	1.0003	0.9971	0.9951	1.25
169A	0.9996	0.9964	0.9937	1.01

4.5 Fuel Load Optimization Predictions

4.5.1 165A Fuel Load Optimization

Cycle 165A is the first of two PALM cycles and the nominal cycle information used in the neural network model can be seen in Table 21. Once an appropriate fuel loading was calculated, the next step was to set up the appropriate xml file that the API takes search parameters from. Cycle 165A ran for a consistent power for a total of cycle length of 14 days. The maximum desired power split is also listed. The minimum power split is not listed due to there being no way to define a minimum allowable power in the API, however, at the very least, each lobe is typically allowed to drop below the nominal desired value by at least 1 MW.

Table 21 Desired Power Split for Cycle 165A

	NE	SE	C	SW	NW
Desired Power Split	20	45	30	43	20
Maximum Power Split	21	55	41	53	21

Table 22 shows the results for the first iteration where only the neural network calculated fuel loading was run. The final timestep of 1E-5 days was included because MC21 calculates the eigenvalue as part of the spatial at the beginning of a timestep, prior to depletion. In the first

iteration the total ^{235}U gram loading within the core was 35,039 grams ^{235}U and there were 8 fresh fuel element. Comparatively, the as run contained 34,678 grams ^{235}U with 11 total fresh fuel elements. While a successful cycle in terms of completion, it was noted that the NE lobe still had all neck shims inserted, indicating slightly too much fuel in the NE, so a second iteration on the fuel loading was completed. Shorter timesteps at the beginning of a cycle is to account for the Xe burn in of the reactor at start up. For the second iteration, the NE lobe was scaled down by 13%. The results for the first iteration are seen in Table 22. The second iteration fuel loading contained 34,393 g ^{235}U and contained 8 total fresh fuel elements.

Table 22 Timestep and Eigenvalue for the MC21 API for 165A Iteration 1

Timestep	Timestep Length (days)	EFPD of Eigenvalue Calculation	Target Eigenvalue	Calculated Eigenvalue Iteration 1	Calculated Eigenvalue Iteration 2
1	1.0	0.0	1.0007	0.9999	1.0009
2	2.0	1.0	0.9984	0.9951	0.9972
3	2.0	3.0	0.9984	1.0001	0.9988
4	5.0	5.0	0.9984	1.0000	1.0014
5	4.0	10.0	0.9984	0.9964	0.9993
6	1E-05	14.0	0.9984	0.9948	0.9977

The API calculated power splits for the first iteration is given in Table 23. The desired power split, as given in Table 23 was within the range established in Table 21 along with the permissible lobe power lower threshold of -5% β. The actual allowable lower threshold for 165A in both the SE and SW lobes was +/- 10 MW.

Table 23 Calculated Power Splits at Timestep for 165A Iteration 1

Timestep	NE	SE	C	SW	NW
1	20.16	46.05	38.13	42.24	19.55
2	20.79	45.42	35.60	41.53	20.26
3	19.89	44.74	37.04	43.42	19.95
4	20.03	44.63	36.18	44.09	19.25
5	20.64	43.89	34.71	43.09	20.39
6	20.60	42.90	33.58	43.94	20.58
Cycle Average Power	20.35	44.60	35.87	43.05	20.00

The NE lobe also had all neck shims out at the end of cycle. Therefore, the second iteration is considered to be the best for cycle 165A. The API calculated power splits for iteration 2 are given in Table 24 and are within the established core parameters of the MC21 API.

Table 24 Calculated Power Splits at Timestep for 165A Iteration 2

Timestep	NE	SE	C	SW	NW
1	20.28	45.52	37.31	42.26	19.94
2	19.97	43.62	34.78	44.48	19.93
3	19.72	43.15	36.10	44.73	20.40
4	20.45	43.68	36.03	44.09	20.24
5	20.38	43.53	35.30	43.63	19.95
6	20.36	43.54	35.13	44.13	20.31
Cycle Average Power	20.19	43.84	35.78	43.84	20.13

Figure 28 shows the comparison of the as run eigenvalue to the two calculated API iterations. In the case of the as run plot, only the days at power were considered and total days included outages were not included. Cycle 165A, likely as a part of a PALM cycle, ran with a substantial outage between the first few days and the last two weeks, for the drum solver, that outage was not modeled. Modeling the cycle as such should not affect burnup data because there was a long enough time between the first and second portion of the cycle that any reactivity effects from short-lived fission products would be gone. Figure 28 shows only the days at power for the cycle as run and does not show the substantial outage due to the way plotting the cycle as such would skew the plot.

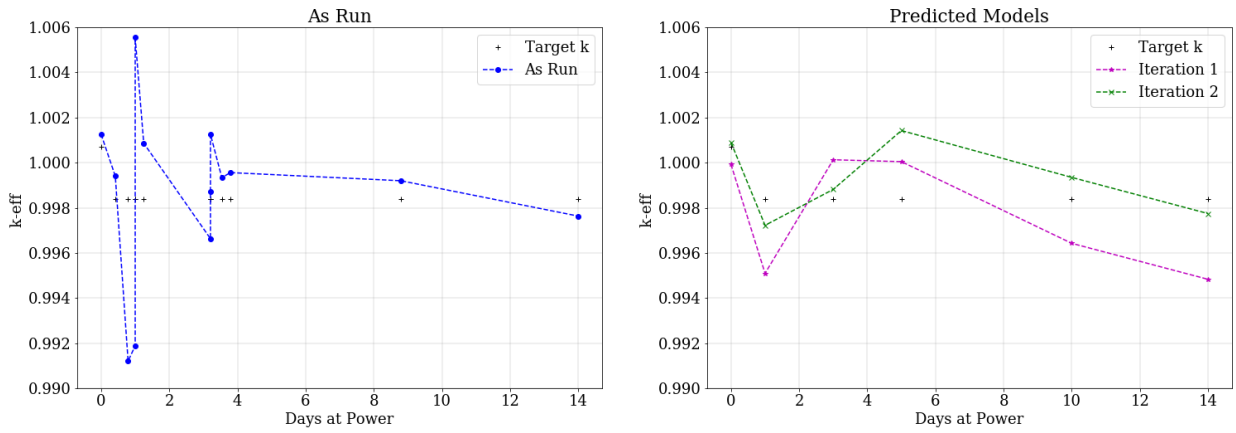

Figure 28 Cycle eigenvalue for as run and API predicted models.

A comparison of the fuel loading for the as run, iteration one, and iteration two is shown in Figure *29*.

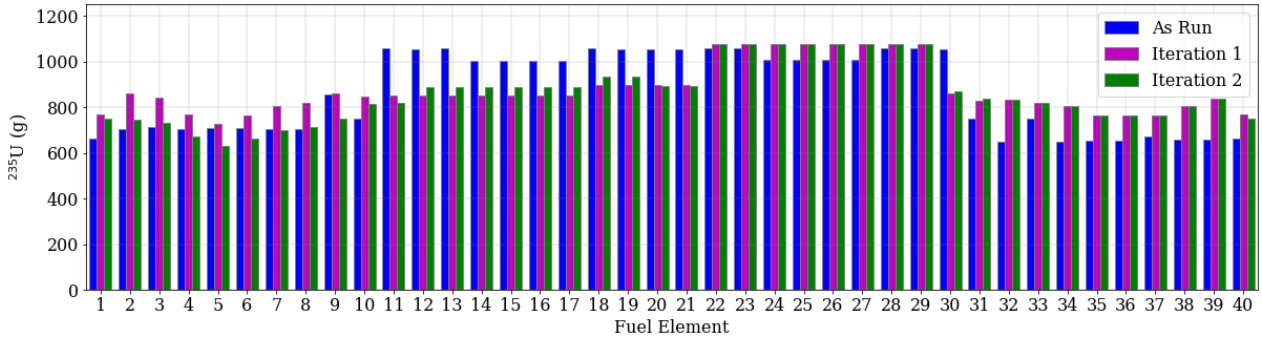

Figure 29 Initial ^{235}U composition for each fuel element for the as runs and neural network models.

The main difference between iteration one and two was reducing the overall ^{235}U gram loading in the NE lobe or elements 2-9 since at the end of the first iteration, a majority of the neck shims were still inserted, which suggests less fuel can be used. The second iteration was able to use less fresh fuel elements and less total fuel overall while staying closer to the target eigenvalue.

5.5.2 166A Fuel Load Optimization

Cycle 166A was a standard ATR cycle that ran for a total of 62.5 EFPD with the target power split given in Table 25. Cycle 166A also used 26 total fresh fuel elements with a total of 40,366 g ^{235}U in the core.

Table 25 Power Splits for API input for 166A

	NE	SE	C	SW	NW
Desired Power Split	17.0	25.0	22.0	25.0	20.0
Maximum Power Split	18.0	28.0	30.0	26.0	21.0

Two successful fuel element loadings were created by the MC21 drum solver API using a 62-day cycle. The first successful fuel loading consisted of 25 fresh fuel elements and a total 40,548 g ^{235}U. The second successful fuel loading contained 19 fresh fuel elements and a total ^{235}U content of 39,767 g ^{235}U and was created based on adjustments to the first successful loading.

To scale to the average error during the first iteration, the NE lobe was scaled down by 4.51%, the SE and SW lobes were not scaled as they were already predicted to be fresh fuel elements and therefore full based on nominal scaling, and the center lobe was scaled based on the nominal scaling given in Table 18 with the exception of fuel elements 30 and 31 which were scaled up to accommodate fuel requirements in the adjacent NWFT. An example of the scaling is given in Table 26. Ideally, the goal would be to match the lobe gram loading completely, however, there is an absolute maximum gram loading that any lobe may contain and in the case of the NW lobe, and restrictions based on element 33 requiring a YA element, the most ^{235}U the lobe can contain is 8531 grams. Adjusting the predicted values based on the kinds of fuel elements available in the inventory would also be important given the rare occurrence that a fuel element will exist that has 1032 g ^{235}U. Such fuel elements would have been run a single time in a very specific PALM cycle and only exist approximately 1% of the time given the dataset. The reason behind a fuel element being predicted at 1034 grams of ^{235}U is an effect from taking the average of the 5 nearest neighbors. In the case of a fuel element having a BOC value of 1034 ^{235}U is really saying that of the 5 nearest neighbors found, 4 were fresh XA or NB elements with BOC ^{235}U of 1075 and one nearest neighbor was a fresh YA element of 1022 g ^{235}U.

Table 26 Rearrangement of Fuel Elements in the NW Lobe for Iteration 1

Fuel Element	Predicted	Scaled to 12%	Rearrangement to conserve mass
32	914	1024	1073
33	947	948	1020
34	1034	1158	1073
35	1011	1132	1073
36	1011	1132	1073
37	1032	1156	1073
38	937	1049	1073
39	914	1024	1073
Lobe ^{235}U gram loading	7700	8624	8531

Table 27 gives the established timesteps and eigenvalues for both iterations for cycle 166A. For the second iteration, the last timestep was not reduced based on a similar dip seen in the cycle as run to model shut down. The second iteration held slightly higher eigenvalues overall and was a smoother cycle in its ability to converge to the target eigenvalue despite having less fuel loaded, implying some effect from more control in tangential lobes reducing the eigenvalue of the NW lobe.

Table 27 Timesteps and Eigenvalue for 166A

Timestep	Timestep Length (days)	EFPD of Eigenvalue	Target Eigenvalue Iteration 1	Calculated Eigenvalue Iteration 1	Target Eigenvalue Iteration 2	Calculated Eigenvalue Iteration 2
1	1.0	0.0	0.9995	1.0002	0.9995	1.0004
2	2.0	1.0	0.9977	0.9980	0.9977	0.9991
3	7.0	3.0	0.9977	0.9965	0.9977	0.9982
4	7.0	10.0	0.9977	0.9962	0.9977	0.9974
5	7.0	17.0	0.9977	0.9979	0.9977	0.9984
6	7.0	24.0	0.9977	0.9982	0.9977	1.0005
7	7.0	31.0	0.9977	0.9988	0.9977	0.9989
8	7.0	38.0	0.9977	0.9996	0.9977	0.9999
9	7.0	45.0	0.9977	0.9986	0.9977	0.9973
10	7.0	52.0	0.9977	0.9996	0.9977	0.9985
11	3.0	59.0	0.9977	0.9984	0.9977	0.9991
12	1E-05	62.0	0.981	0.9813	0.9977	0.9965

Table 28 gives the power splits calculated by the MC21 API for the first iteration. Overall, the cycle was able to maintain appropriate power splits over the cycle, however at the end of the cycle, all but one neck shim was still inserted in the SE and SW lobes, implying that fuel could be reduced there.

Table 28 Calculated Power Splits for Cycle 166A Iteration 1

Timestep	NE	SE	C	SW	NW
1	17.79	24.27	30.73	24.51	20.42
2	17.03	24.36	26.78	25.06	20.55
3	16.73	24.79	26.64	24.85	20.63
4	16.55	26.11	25.76	24.29	20.05
5	16.93	24.33	24.82	25.61	20.12
6	16.88	24.26	24.00	25.92	19.94
7	17.15	24.62	24.44	25.07	20.16
8	17.05	23.97	23.32	25.90	20.08
9	16.88	24.95	22.47	25.78	19.39
10	17.71	24.69	22.28	25.05	19.55
11	17.01	25.23	21.82	25.65	19.12
12	17.24	24.75	23.10	24.50	20.51
Cycle Average Power	17.08	24.69	24.68	25.18	20.04

The nominal predicted fuel loading contained almost all fresh fuel elements in both the SE and SW lobes. With the exception of fuel elements 12 and 22, the SE and SW lobes were unable to be scaled up from the nominal predicted values based on Table 18 and Table 19. Within the error established within the dataset, the SE lobe could be scaled down by 4.3% and the SW lobe could be scaled down by 3%. The total lobe ^{235}U gram loading was applied for the SE and SW lobe since scaling down the total fuel reduces individual elements to 1028 and 1042 grams of ^{235}U per fuel element for the SE and SW lobe, respectively. The fuel elements are rearranged to maintain the same total lobe gram loading and rearranged to a combination of fresh fuel elements and recycled fuel elements. Table *29* gives an example of how the SW lobe was scaled to account for the same ^{235}U gram loading with a fuel loading that was more readily available in the inventory.

Table 29 Rearrangement of Fuel Elements in the SE Lobe

Fuel Element	Predicted	Scaled Down	Rearrangement to conserve mass
22	985	955	955
23	1073	1042	1073
24	1073	1042	1073
25	1073	1042	959
26	1073	1042	960
27	1073	1042	1073
28	1073	1042	1073
29	1073	1042	1073
Lobe ^{235}U gram loading	8585	8249	8249

The second iteration was able to converge completely without a reduction in the target eigenvalue at EOC. The second iteration also had sufficient neck shims removed at end of cycle to not damper neighboring lobes. Table 30 shows the API calculated power splits over 166A, all of which are contained within the input bounds.

Table 30 Power Splits over Cycle 166A Iteration 2

Timestep	NE	SE	C	SW	NW
1	17.19	25.42	30.91	25.16	19.23
2	16.96	24.52	26.66	24.94	20.58
3	17.33	25.53	26.46	24.54	19.61
4	16.42	24.52	26.11	25.48	20.57
5	16.99	25.68	25.21	24.72	19.61
6	16.80	25.82	24.13	25.07	19.31
7	16.77	25.69	23.51	25.05	19.49
8	16.77	24.92	23.26	25.18	20.13
9	16.83	24.98	22.61	25.57	19.62
10	17.32	25.05	22.22	25.10	19.53
11	17.43	24.85	22.97	25.02	19.71
12	16.99	24.81	22.78	25.50	19.70
Cycle Average Power	16.98	25.15	24.73	25.11	19.76

Figure 30 shows a comparison between the fuel loading of the as run to the two iterations.

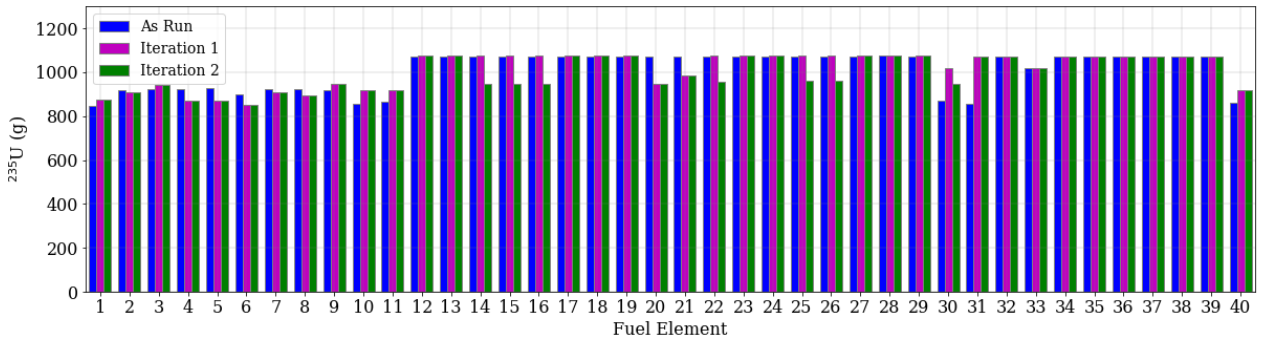

Figure 30 BOC ^{235}U for as run and both neural network iterations.

Figure 31 shows a comparison between the as run eigenvalue and the two API solutions. The first iteration saved one fuel element compared to the as run but did load more total ^{235}U into the core when scaled. The second iteration successfully saved 7 fuel elements and loaded less fuel than what was used in the as run. The as run eigenvalue over the cycle includes ascent to power and any outages over the cycle.

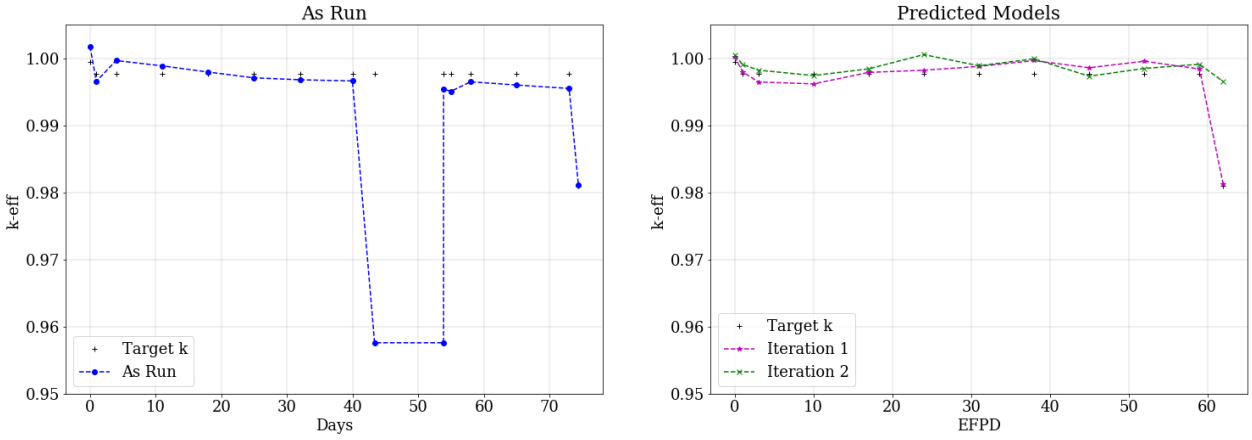

Figure 31 Eigenvalue comparison between as run and API NN iterations.

The second iteration was a more stable cycle over all 62 EFPD and utilized 19 fresh fuel elements and less total ^{235}U than the as run and is therefore considered the optimized cycle.

4.5.3 166B Fuel Load Optimization

Cycle 166B was a standard ATR cycle that ran for a total of 61.2 EFPD with a desired power split as shown in Table 31. Cycle 166B contained 40,240 g ^{235}U and 24 fresh fuel elements.

In the case of Cycle 166B, the last timestep was allowed to be slightly smaller so mimic the final timestep that occurred in the as run.

Table 31 Desired Power Splits for 166B

	NE	SE	C	SW	NW
Desired Power Split	17.0	25.0	22.0	25.0	20.0
Maximum Power Split	18.0	28.0	30.0	26.0	21.0

Two successful fuel loadings were found using the desired parameters and cycle lengths. The projected cycle length for cycle 166A was also 62.5 days, which was slightly longer than the as run data. Adding some time to the end of an API run can add some conservatism into ensuring a fuel loading is appropriate. The first successful iteration for cycle 166B contained a total of 22 fresh fuel elements and 39,605 g ^{235}U. Iteration 1 required both the NW lobe to be scaled to entirely fresh fuel elements, but also center lobe elements 31 and 40 also needed to be fresh. Scaling the NW and C fuel elements was within the established potential error in the models. Past unsuccessful iterations of cycle 166B showed that an overloading of the NE and SW lobes adjacent to the NW lobe have an effect on the drum solver APIs ability to find convergence towards the end of cycle. For example, in the case of earlier 166A iterations, an overloading in the rest of the lobes caused more reactor control elements to be inserted into the core and were contributing to a damping of reactivity in not just the desired lobe, but the nearby NW lobe. The second successful iteration of the fuel loading reduced the scaling in C lobe fuel elements 31 and 40 to their originally calculated values instead of being upscaled to fresh fuel elements. The second iteration fuel loading had 20 total fuel elements and 39,285 g ^{235}U total. Table 32 shows the timesteps and eigenvalues for the two successful iterations of 166B.

Table 32 Timesteps and Eigenvalues for 166B

Timestep	Timestep Length (days)	EFPD of Eigenvalue	Target Eigenvalue	Calculated Eigenvalue Iteration 1	Calculated Eigenvalue Iteration 2
1	1.0	0.0	1.0001	0.9988	0.9995
2	2.0	1.0	0.9977	0.9983	1.0001
3	7.0	3.0	0.9977	0.9994	0.9995
4	7.0	10.0	0.9977	0.9976	0.9964
5	7.0	17.0	0.9977	0.9997	0.9991
6	7.0	24.0	0.9977	0.9983	0.9975
7	7.0	31.0	0.9977	0.9981	0.9983
8	7.0	38.0	0.9977	0.9999	0.9979
9	7.0	45.0	0.9977	0.9992	0.9988
10	7.0	52.0	0.9977	0.9996	0.9993
11	3.5	59.0	0.9977	0.9998	0.9980
12	1E-05	62.5	0.995	0.9954	0.9948

Table 33 shows the calculated power for iteration 1 of cycle 166B over time. All timesteps were able to operate within the established range given in Table 31. Neck shims remaining in the core at EOC contribute to requiring excess reactivity in the NW lobe to overcome the control, therefore a second iteration was run where fuel was reduced in the SW and SE lobes.

Table 33 Calculated API Powers for Cycle 166B Iteration 1

Timestep	NE	SE	C	SW	NW
1	17.53	24.50	30.27	25.07	19.90
2	16.69	25.12	25.35	25.16	20.02
3	17.18	24.94	25.18	24.57	20.31
4	17.01	25.03	24.59	24.99	19.98
5	16.99	24.78	24.34	24.87	20.36
6	16.98	25.31	23.76	24.56	20.14
7	16.74	24.56	23.01	25.65	20.05
8	16.91	25.00	22.95	25.24	19.86
9	17.09	24.48	22.94	25.68	19.75
10	16.48	25.48	22.96	25.43	19.61
11	16.79	25.06	23.35	25.43	19.72
12	16.86	24.89	23.25	25.27	19.99
Cycle Average Power	16.94	24.93	24.33	25.16	19.98

Table 34 shows the calculated power over the cycle. The API calculated power is within the desired parameters stated in Table 31.

Table 34 Calculated API Powers for Cycle 166B Iteration 2

Timestep	NE	SE	C	SW	NW
1	16.84	24.32	29.15	24.32	20.84
2	16.43	25.17	24.96	25.17	20.03
3	16.79	24.96	25.00	24.96	20.38
4	17.07	25.09	24.40	25.09	20.36
5	16.42	24.64	24.16	24.64	20.71
6	16.32	24.84	23.49	24.84	20.62
7	16.41	24.89	22.85	24.89	20.41
8	16.46	24.35	22.33	24.35	20.33
9	16.95	24.62	21.88	24.62	20.02
10	17.39	25.28	22.59	25.28	19.82
11	16.92	25.07	22.74	25.07	19.63
12	16.87	25.00	22.50	25.00	19.56
Cycle Average Power	16.74	24.85	23.84	24.85	20.23

Figure 32 shows the ^{235}U gram loading for the as run, first, and second iteration of the optimization. Iteration 2 used the least overall ^{235}U and the least number of fresh fuel elements.

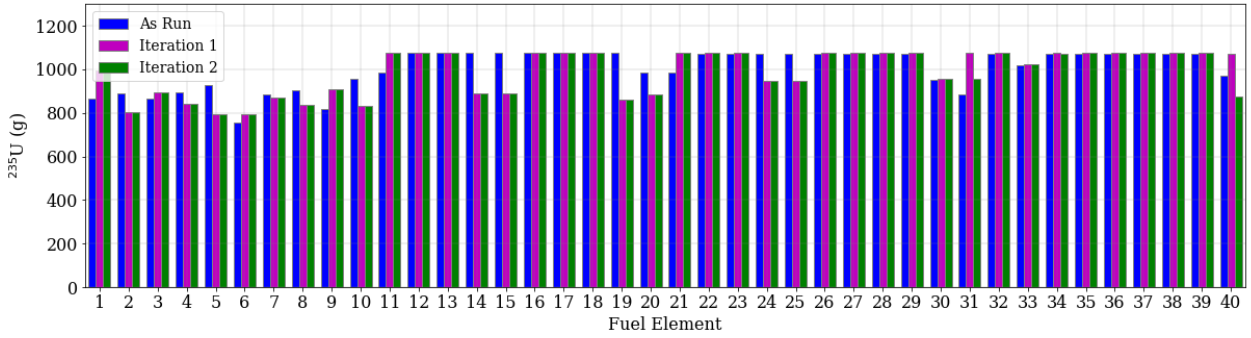

Figure 32 BOC ^{235}U for both As Run and Predictive Iterations for Cycle 166B

Figure 33 shows both iterations in comparison to the as run. The as run showed a downward eigenvalue drift over the course of the cycle but did not have any outages.

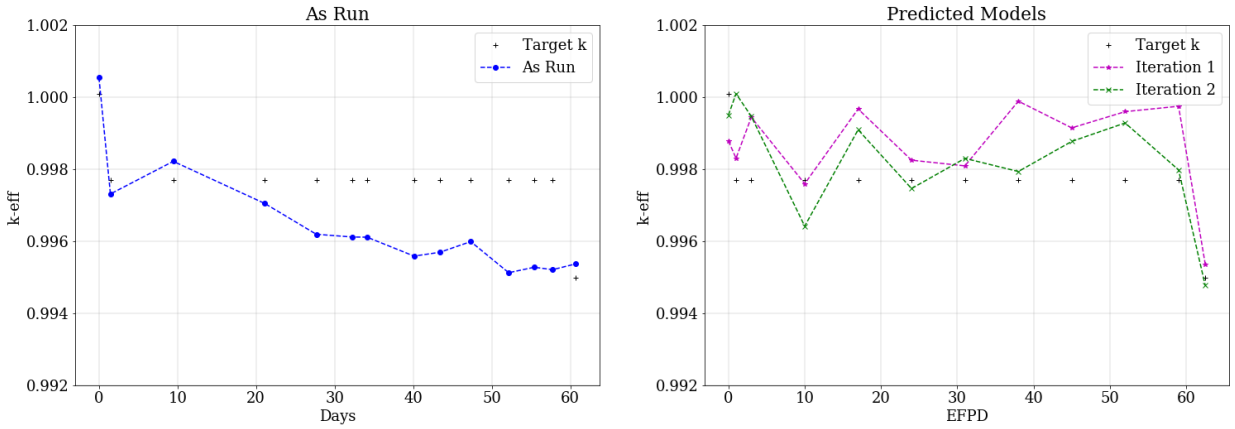

Figure 33 Eigenvalue Comparison between as run and MC21 API for Cycle 166B.

Overall, Iteration 2 shows a lower overall difference from the target eigenvalue, contained fewer fresh fuel elements and less total ^{235}U.

4.5.4 167A Fuel Load Optimization

Cycle 167A was the second PALM cycle and began with a 5-day low powered section followed by two days at high power in the SE and SW lobes for a total cycle length of seven days. Table 35 gives the desired power over the timesteps for cycle 167A.

Table 35 Desired Power Splits for Cycle 167A

	NE	SE	C	SW	NW
Desired Power Split – Timestep 1	5.0	10.0	6.0	10.0	5.0
Desired Power Split – Timesteps 2 and 3	18.0	44.0	33.0	48.0	20.0
Maximum Power Split	21.0	54.0	46.0	58.0	23.0

The desired eigenvalue and established timesteps are given in Table 36 along with the target and calculated eigenvalue. The NW lobe required scaling up by the maximum value of nearly 16% while the NE lobe was scaled up to the maximum value of 3.99%. The SE lobe also required significant scaling down of 18.8%. Overall, the cycle ended up with 31,504 g ^{235}U with 8 total fresh fuel elements. Scaling the power here required utilizing the maximum lobe power error that was seen in Table 18. Scaling down by 18.8% was justified because cycle 167A is a PALM cycle that has differing behaviors from standard cycles.

Table 36 Timesteps and Eigenvalue for 167A

Timestep	Timestep Length (days)	EFPD of Eigenvalue	Target Eigenvalue	Calculated Eigenvalue
1	5.0	0.0	0.9977	0.9956
2	2.0	5.0	0.9977	0.9948
3	1.0E-05	7.0	0.9977	0.9979

Table 37 shows the calculated power for the cycle. SE, SW, and C lobe hit the nominal power with C lobe being 1.5 MW high. The C lobe will generally have the most swing in the power splits due to the power in the lobe drifting since it is not explicitly controlled. The NE and NW lobes run ~0.2 MW shy in the first timestep, however, there is allowable downward drift of ~1 MW at a minimum for ATR control. The NE lobe also maintains a lobe power slightly less than nominal for the high powered timesteps, while the NW lobe is only slightly lower on the second timestep. The discrepancy between nominal and actual is still within an acceptable range. SE and SW lobes were allowed to drift down 10 MW from nominal over the cycle.

Table 37 API Calculated Power Splits for Cycle 167A

Timestep	NE	SE	C	SW	NW
1	4.80	10.01	7.57	10.37	4.83
2	17.81	42.00	34.41	50.81	19.39
3	17.93	42.31	35.30	49.61	20.16

Figure 34 shows the fuel loading for the as run and the scaled fuel loading. The 167A as run contains 32,831 g ^{235}U and 10 fresh fuel elements. The 167A optimized fuel loading contains 8 fresh fuel elements and 31,631 g ^{235}U.

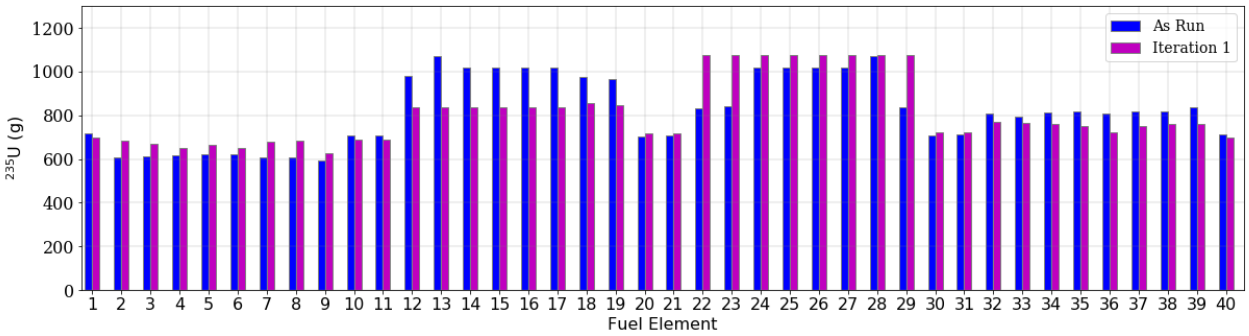

Figure 34 BOC ^{235}U for as run and predictive model.

Figure 35 shows the comparison of the as run to the optimized fuel loading. The EFPDs match between cycles, the MC21 drum solver API does not account for time spent in ascent to power as it can directly scale.

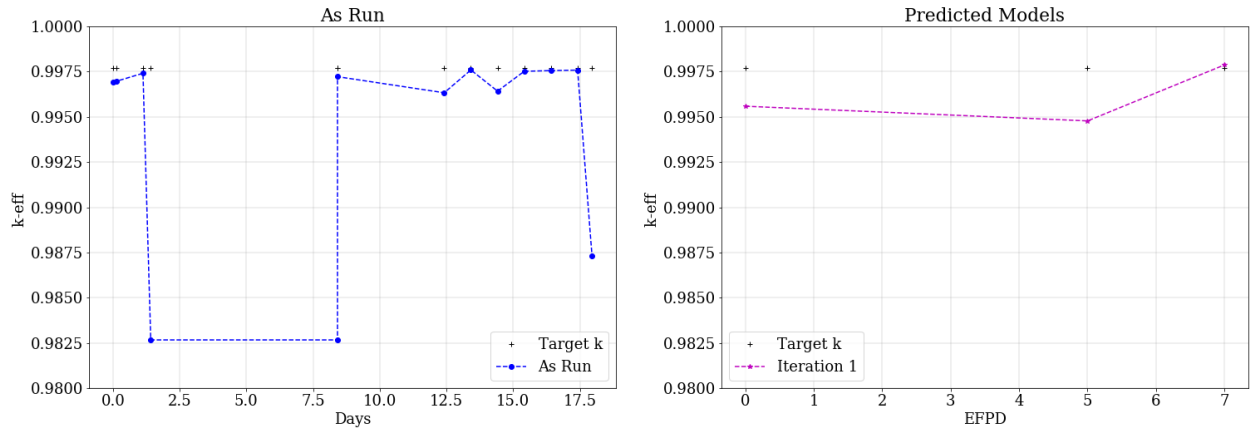

Figure 35 As run and API predicted eigenvalues for cycle 167A.

4.5.5 168A Fuel Load Optimization

Cycle 168A is a full-length cycle that lasted 61 EFPD. Table 38 gives the desired power split and upper bound for power that is set in the .xml file. The initial cycle contained 40,488 g ^{235}U with 24 total fresh fuel elements.

Table 38 Desired Power Splits for 168A

	NE	SE	C	SW	NW
Desired Power Split	19.0	25.0	22.0	25.0	20.0
Maximum Power Split	20.0	28.0	30.0	26.0	23.0

One cycle was created that had an acceptable fuel loading containing 19 fresh fuel elements and 38,911 g ^{235}U. The NE, SE, SW, and C lobes were scaled based on the nominal average error within Table 18 while the NW lobe was scaled using an 12% increase. The 12% increase is within the range of the established error and is the maximum error defined in Table 18 for HELIOS power errors. The cycle was able to maintain stability over all 61 days.

Table 39 Timesteps and Eigenvalue for Cycle 168A

Timestep	Timestep Length (days)	EFPD of Eigenvalue	Target Eigenvalue	Calculated Eigenvalue
1	1.0	0.0	1.00027	0.9994
2	2.0	1.0	0.9972	0.9996
3	7.0	3.0	0.9972	0.9986
4	10.0	10.0	0.9972	0.9991
5	10.0	20.0	0.9972	0.9968
6	10.0	30.0	0.9972	0.9983
7	10.0	40.0	0.9972	0.9991
8	10.0	50.0	0.9972	0.9965
9	1.0	60.0	0.9972	0.9980
10	1.0E-05	61.0	0.9972	0.9965

Table 40 shows the calculated lobe power over the timesteps. NE and NW lobes were approximately 0.2 MW low on average, however, the deduction is within the -2 MW and -3 MW range established by the ISOP for the NE and NW lobes, respectively. Per the as run data available on atrfuel.inl.gov, the cycle ran for NE and NW powers of 19.8 and 19.9 MW, respectively.

Table 40 API Calculated Power Splits for 168A

Timestep	NE	SE	C	SW	NW
1	19.03	25.37	27.39	24.60	20.01
2	18.97	25.08	24.22	25.22	19.73
3	19.83	26.07	24.16	24.03	19.08
4	18.67	25.25	23.21	25.77	19.31
5	18.63	25.87	22.23	25.00	19.51
6	18.48	25.17	21.61	25.34	20.01
7	19.41	25.06	21.92	24.82	19.71
8	18.60	24.98	21.65	25.41	20.01
9	18.17	24.56	21.72	26.06	20.21
10	18.43	24.71	21.81	25.64	20.22
Cycle Average Power	18.82	25.21	22.99	25.19	19.78

Figure 36 shows the comparison of the as run fuel loading compared to the fuel load optimization predicted fuel loading. Overall, the fuel load optimization process predicts 5 fewer fresh fuel elements and 1577 fewer grams ^{235}U initially.

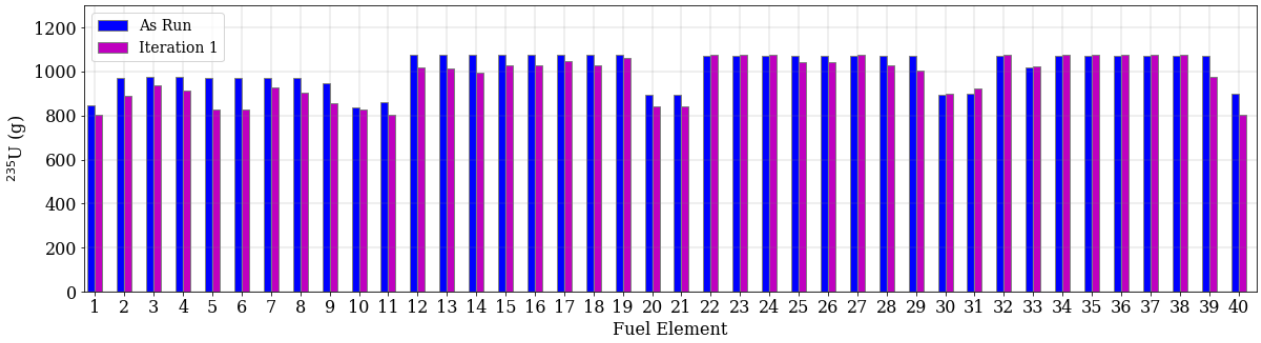

Figure 36 BOC ^{235}U comparison between as run and predicted cycle loading.

Figure 37 shows a comparison of the as run and the predictive model. The as run contained two outages but maintains the same number of EFPD. Overall, the drum solver was able to maintain a stable cycle that tracked very closely with the target eigenvalue prediction.

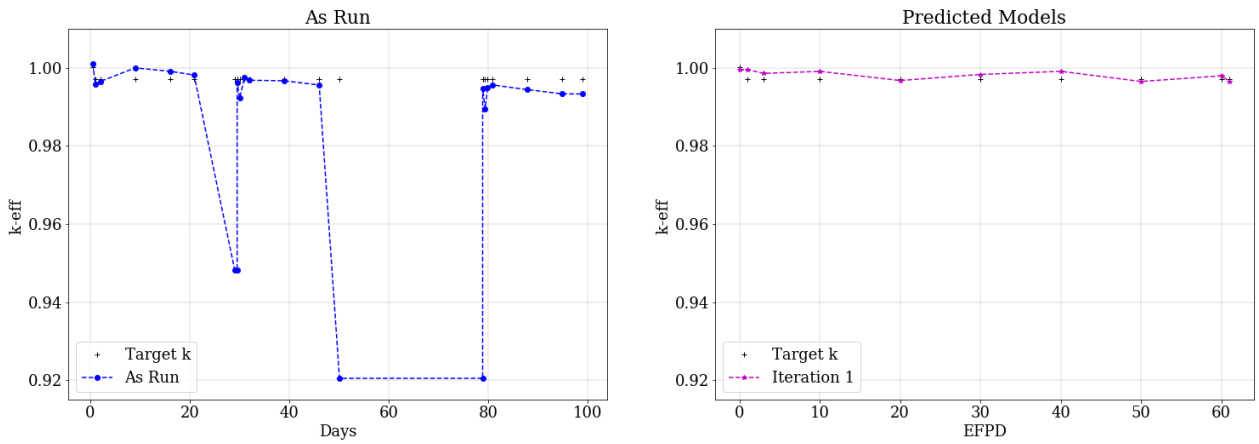

Figure 37 Eigenvalue comparison between as run and predicted model for 168A.

4.5.6 168B Fuel Load Optimization

Cycle 168B was supposed to be a 60-day cycle with the power split defined in Table 41. However, due to a core underloading, the cycle actually ran for 57.5 EFPD and while the as run ran at 19.8 MW in the NE and 23.6MW in the SW, the SE ran 1.2 MW low over the cycle, and the NW ran 1.9 MW low over the cycle. While the lower acceptable range is established, it is undesirable to run a cycle 1-2 MW below target. The as run only contained 6 fresh fuel elements and a total 37,422 g ^{235}U initially. Also, the reason why the cycle ended was due to the inability of the reactor to maintain criticality.

Table 41 Desired Power Splits for Cycle 168B

	NE	SE	C	SW	NW
Desired Power Split	19.0	24.0	21.0	23.0	20.0
Maximum Power Split	20.0	26.0	25.0	24.0	23.0

Cycle 168B is the only predictive model where the estimated number of fresh fuel elements exceeds the number that was established for the actual cycle. The model was created based on predicting the power split from Table 42 and scaling appropriately to run for an EFPD of 60 days. The fuel load optimization process predicted needing 19 fresh fuel elements with a total ^{235}U gram loading of 39,190.

Table 42 Timesteps and Eigenvalue for 168B

Timestep	Timestep Length (days)	EFPD of Eigenvalue	Target Eigenvalue	Calculated Eigenvalue
1	1.0	0.0	1.0003	1.0000
2	2.0	1.0	0.9971	0.9968
3	7.0	3.0	0.9971	0.9984
4	7.0	10.0	0.9971	0.9991
5	7.0	17.0	0.9971	0.9976
6	7.0	24.0	0.9971	0.9990
7	7.0	31.0	0.9971	1.0003
8	7.0	38.0	0.9971	0.9970
9	7.0	45.0	0.9971	0.9977
10	7.0	52.0	0.9971	0.9992
11	1.0	59.0	0.9971	0.9978
12	1E-05	60.0	0.9971	0.9966

Table 43 gives the lobe power calculated by the MC21 API over each timestep along with a cycle average power. Overall, the reactor was able to run for a total of 60 days while maintaining the desired lobe power in all lobes over the entire cycle. Therefore, the fuel loading can be seen as successful.

Table 43 API Calculated Power Splits for 168B

Timestep	NE	SE	C	SW	NW
1	19.32	23.06	26.91	22.84	20.78
2	19.90	23.83	23.74	22.22	20.05
3	18.54	24.02	23.68	23.19	20.25
4	18.96	23.63	23.44	23.19	20.22
5	18.98	24.19	22.67	22.81	20.02
6	18.81	24.18	21.68	23.29	19.72
7	18.75	24.51	21.07	22.97	19.77
8	18.87	24.30	20.51	23.07	19.75
9	19.41	24.21	20.56	22.44	19.95
10	19.58	23.74	21.10	22.79	19.89
11	19.19	24.32	21.64	23.06	19.42
12	19.18	24.04	21.65	23.03	19.76
Cycle Average Power	19.12	24.00	22.39	22.91	19.97

Figure 38 shows the comparison of the as run fuel loading to the fuel load optimization prediction for cycle 168B. Overall the cycle used 13 more fresh fuel elements than was loaded, but was able to reach its desired cycle length at its desires power, unlike the as run.

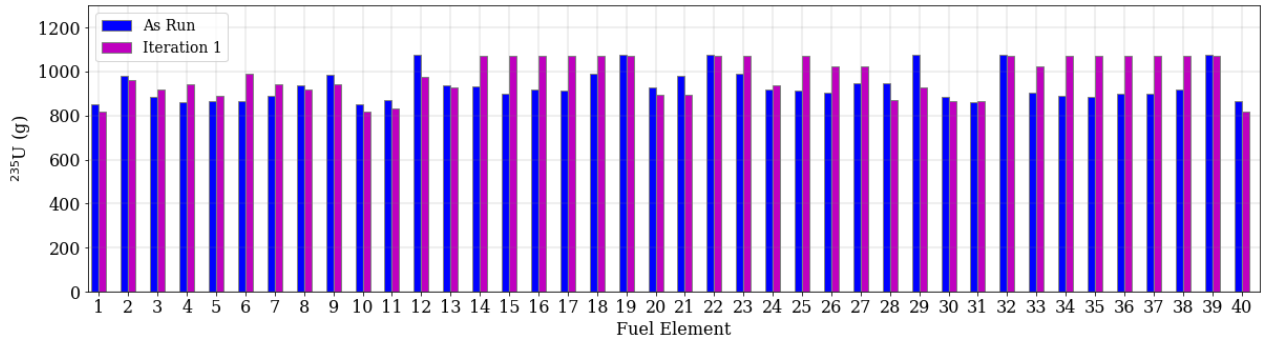

Figure 38 BOC ^{235}U for As Run and Predictive Models

Figure 39 shows a comparison of the cycle eigenvalue over the course of the cycle. The eigenvalue in the as run shows a significant decrease over the cycle due to slipping lobe power while the eigenvalue for the predictive model sits mostly above the cycle target eigenvalue. However, all eigenvalues were within the defined allowable range set by the drum solver or the drum solver would not have converged on a solution. Cycle 168B also has a single outage.

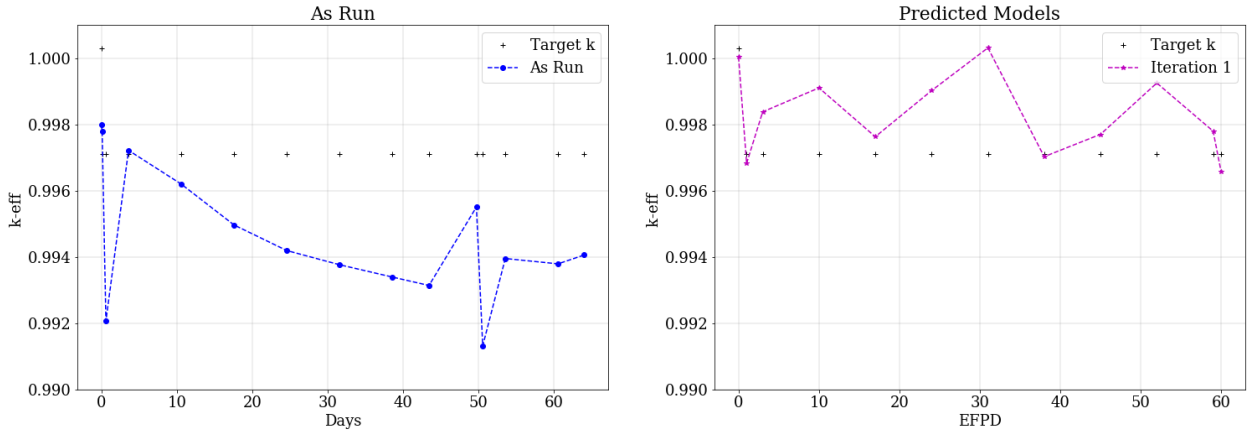

Figure 39 Eigenvalue comparison between as run and predicted model for cycle 168B.

4.5.7 169A Fuel Load Optimization

Cycle 169A was the longest overall cycle and lasted for 63 EFPD. Cycle 169A contained 23 total fresh fuel elements and 40,058 g ^{235}U. Table 44 gives the desired and maximum lobe powers for cycle 169A.

Table 44 Desired Power Splits for 169A

	NE	SE	C	SW	NW
Desired Power Split	20.0	23.0	22.0	25.0	20.0
Maximum Power Split	21.0	26.0	25.0	26.0	23.0

Overall, three different iterations were completed to fully optimize the core. The first iteration contained 19 fresh fuel elements and 39,890 g ^{235}U. Table 45 shows the timestep and the calculated versus target eigenvalue for the iteration.

Table 45 Timesteps and Eigenvalue for 169A

Timestep	Timestep Length (days)	EFPD of Eigenvalue	Target Eigenvalue	Calculated Eigenvalue Iteration 1	Calculated Eigenvalue Iteration 2	Calculated Eigenvalue Iteration 3
1	1.0	0.0	0.9996	0.9991	0.9991	0.9990
2	2.0	1.0	0.9964	0.9997	0.9992	0.9978
3	7.0	3.0	0.9964	0.9979	0.9977	0.9982
4	7.0	10.0	0.9964	0.9980	1.0012	0.9963
5	7.0	17.0	0.9964	0.9963	0.9956	0.9970
6	7.0	24.0	0.9964	0.9969	0.9967	0.9977
7	7.0	31.0	0.9964	0.9991	0.9968	0.9987
8	7.0	38.0	0.9964	0.9965	0.9982	0.9983
9	7.0	45.0	0.9964	0.9980	0.9982	0.9991
10	7.0	52.0	0.9964	0.9981	0.9987	0.9983
11	4.0	59.0	0.9964	0.9969	0.9985	0.9984
12	1E-05	63.0	0.9964	0.9981	0.9963	0.9964

Table 46 gives the calculated lobe power for each timestep in the first iteration along with the cycle average power. The SW lobe averages 0.24 MW low and the NW lobe averages 0.04 MW low, both values are within the 5% allowable below-desired lobe power. At the end of the cycle, the SE lobe was determined to have significant control mechanisms inserted, so a second iteration with a reduction in the fuel in the SE lobe was appropriate.

Table 46 API Calculated Power Splits for 169A Iteration 1

Timestep	NE	SE	C	SW	NW
1	19.88	23.01	29.62	25.28	19.82
2	20.22	22.87	25.28	24.92	20.00
3	20.35	23.72	25.47	24.14	19.79
4	20.30	23.31	24.79	24.57	19.82
5	20.01	22.83	24.31	25.22	19.94
6	20.19	22.99	23.46	25.02	19.80
7	19.67	23.35	23.12	24.23	20.75
8	19.84	23.56	22.49	24.55	20.04
9	20.56	23.27	22.47	24.37	19.81
10	19.46	23.20	22.86	25.38	19.95
11	20.07	23.79	23.89	24.43	19.71
12	19.66	23.32	23.52	24.99	20.04
Cycle Average Power	20.02	23.27	24.27	24.76	19.96

The second iteration contained 16 fresh fuel elements and a total core loading of 39,607 g ^{235}U. The scaling for the second iteration consisted of taking the nominally predicted fuel loading for the SE lobe. Using the nominally predicted value takes data that is between the nominal average error that would increase the fuel element loading by 4.2% and the least conservative error which would reduce the nominal fuel loading by 4.3%.

Table 47 gives the calculated lobe power for each timestep in the second iteration along with the cycle average power. Overall, at most, the NW and NE lobes converged to slightly below the desired but were still within the -5% of desired lobe power that is allowed per the drum solver. Overall, there was still some room in the SE lobe to further reduce fuel loading, so the least conservative option was considered.

Table 47 API Calculated Power Splits for 169A Iteration 2

Timestep	NE	SE	C	SW	NW
1	19.61	22.94	29.59	26.26	19.21
2	19.78	22.82	25.46	24.99	20.41
3	19.87	23.00	25.53	25.02	20.11
4	20.10	23.11	24.31	25.10	19.69
5	19.83	23.28	24.39	24.90	20.00
6	19.89	23.12	23.39	25.19	19.80
7	19.81	22.82	22.61	25.46	19.92
8	20.18	23.09	22.64	24.44	20.29
9	20.25	22.76	22.80	25.15	19.85
10	20.02	23.65	23.10	24.63	19.70
11	19.95	23.40	24.04	25.04	19.61
12	19.95	23.04	23.58	25.12	19.89
Cycle Average Power	19.94	23.09	24.29	25.11	19.87

The third and final iteration reduced the nominal predicted fuel loading by the least conservative error in the SE lobe of 4.3%. The third iteration used 15 fresh fuel elements and 39,267 g ^{235}U. Table 48 gives the calculated vs target eigenvalue for the third iteration. Table 48 gives the calculated power over the cycle and the average cycle power for the third iteration fuel loading. All lobes remain within tolerances for the drum solver convergence criteria.

Table 48 API Calculated Power Splits for 169A Iteration 3

Timestep	NE	SE	C	SW	NW
1	19.87	22.67	29.56	24.85	20.61
2	20.99	23.11	25.09	24.51	19.40
3	20.74	23.38	24.93	24.82	19.07
4	20.25	22.23	24.70	25.32	20.21
5	19.90	23.01	23.86	25.20	19.89
6	20.10	23.07	23.07	25.15	19.68
7	19.84	22.62	22.68	25.11	20.42
8	19.89	23.89	22.78	24.55	19.67
9	19.28	22.90	22.93	26.15	19.67
10	19.98	22.84	23.41	24.94	20.24
11	19.61	22.90	24.34	25.52	19.97
12	19.68	22.63	24.66	25.33	20.36
Cycle Average Power	20.01	22.94	24.34	25.12	19.93

Figure 40 shows the comparison of the as run to each iteration. Overall, the differences between the first, second, and third iteration are attributed to scaling down the SE lobe appropriately. Since the first and second iterations still had the neck shims inside the core at EOC, the lobes were reduced in ^{235}U. The first iteration used 19 fresh fuel elements and saved 5 fuel elements from the as run.

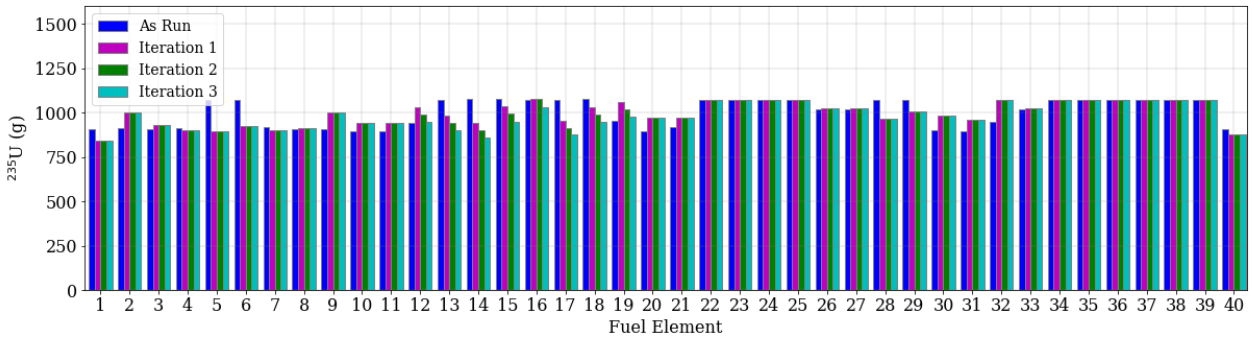

Figure 40 BOC ^{235}U comparison between as run and predictive models for 169A.

A comparison of the as run and predicted model eigenvalue is given in Figure 41. Cycle 169A experienced no mid-cycle outages but did display the eigenvalue drift over time. Of the three

example iterations, iteration 3 was the smoothest in relation to peaking of the eigenvalues. Iteration 2 showed the largest drift in the cycle.

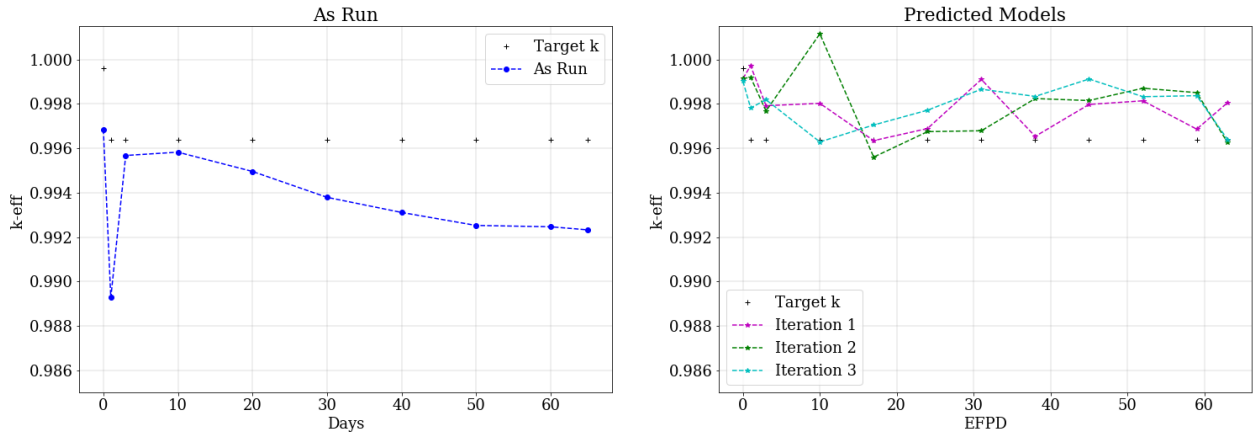

Figure 41 Eigenvalue comparison between as run and API calculated eigenvalues for 169A.

Overall, the third iteration used 8 fewer fresh fuel elements and was able to maintain the most stable critical eigenvalue over the cycle. However, the second iteration would use only one more fresh fuel elements and give more conservatism in the SE lobe.

5. Conclusions

A fuel load optimization process was completed for the ATR. Multiple machine learning algorithms were evaluated for their predictive abilities and based on the results obtained by analyzing linear regression, random forest regression, and neural networks. Neural networks were found to perform the best in terms of approaching a comparable burnup and was ultimately used in creating fuel loadings run in the MC21 drum solver API. The best performing neural network model predicted burnup instead of BOC ^{235}U so a methodology using the KNN algorithm was utilized to get BOC ^{235}U from expected core parameters and expected burnup. Once KNN was used, the fuel loading was scaled and potentially rearranged based on conserving mass within a lobe. Ultimately, the best performing optimization saved 16 fresh fuel elements over the as run data including the single cycle that was not able to predict a loading with fewer fresh fuel elements than was loaded into the core. Table 49 shows the number of fuel elements that were considered fresh, which included all elements with a fuel loading >1020 g ^{235}U.

Table 49 Fresh Fuel Element Comparison Between As Run and Optimized Models

Cycle	As Run	Iteration 1	Iteration 2	Iteration 3	Minimum Difference	Maximum Difference
165A	11	8	8	----	3	3
166A	26	25	19	----	1	7
166B	24	22	20	----	2	4
167A	10	8	----	----	2	2
168A	24	19	----	----	5	5
168B	6	19	----	----	13	13
169A	23	19	16	15	4	8
Overall	124	120	109	108	4	16

There were approximately two calendar years between cycle 165A and cycle 169A, in the worst-case prediction, the model was able to save a total of four fuel elements and in the best case the model was able to save 16 fresh fuel elements, both of which including cycle 168B that was predicted at core parameters that would exist prior to a cycle run, and did not completely match in

cycle length or power splits. If cycle 168B is excluded due to the insufficient loading of fuel causing the as run to run at lower powers for less time than desired, the total number of fresh fuel elements of the other 6 cycles is 118 fuel elements while the best-case optimized fuel loading resulted in 89 fuel elements, or a total reduction of fresh fuel elements of 24.6%. If cycle 168B is included despite the discrepancies between desired and actual runs, the total reduction from 124 to 108 fresh fuel elements is equivalently a reduction of 13%.

Machine learning algorithms benefit from larger datasets and as the dataset is extended and improved upon, the prediction algorithms will get better. The error scaling should be reevaluated over time to account for biases in the dataset being used and not necessarily the current dataset. Even though the data of record was used for the current work, transitioning burnup data away from HELIOS and to results garnered from the MC21 as run cycles could also improve errors within the dataset. Future effort could be placed in finding an improved transition from the ^{235}U burnup to an appropriate BOC ^{235}U gram loading.

Works Cited

[1] Idaho National Laboratory , "GDE-185 Selection of ATR Fuel Loading," 2021.

[2] Advanced Test Reactor, "FY23 Integrated Strategic Operational Plan," 2022.

[3] E. a. J. S. Fasching, "U.S. Energy Information Administration," eia, 1 October 2021. [Online]. Available: https://www.eia.gov/todayinenergy/detail.php?id=49796. [Accessed 26 February 2023].

[4] N. Tsoulfanidis, The Nuclear Fuel Cycle, La Grange Park, IL: American Nuclear Society, 2013.

[5] IBM, "What is supervised learning?," [Online]. Available: https://www.ibm.com/topics/supervised-learning#:~:text=Supervised%20learning%2C%20also%20known%20as,data%20or%20predict%20outcomes%20accurately.. [Accessed 5 March 2023].

[6] J. Brownlee, "14 Different Types of Learning in Machine Learning," Machine Learning Mastery, 11 November 2019. [Online]. Available: https://machinelearningmastery.com/types-of-learning-in-machine-learning/.

[7] codecademy, "Supervised vs. Unsupervised," codecademy, [Online]. Available: https://www.codecademy.com/article/machine-learning-supervised-vs-unsupervised. [Accessed 6 March 2023].

[8] S. Bhatt, "Reinforcement Learning 101," Towards Data Science, 19 March 2018. [Online]. Available: https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292. [Accessed 6 March 2023].

[9] codecademy, "The Machine Learning Process," [Online]. Available: https://www.codecademy.com/paths/machine-learning/tracks/introduction-to-machine-learning-skill-path/modules/introduction-to-machine-learning-skill-path/articles/the-ml-process. [Accessed 6 March 2023].

[10] codecademy, "Classification: K-Nearest Neighbors," codecademy, [Online]. Available: https://www.codecademy.com/learn/machine-learning-k-nearest-neighbors/modules/knn-classification-course/cheatsheet. [Accessed 6 March 2023].

[11] K. S. Htoon, "A Guide to KNN Imputation," Medium, 2 July 2020. [Online]. Available: https://medium.com/@kyawsawhtoon/a-guide-to-knn-imputation-95e2dc496e. [Accessed 6 March 2023].

[12] S. Marsland, Machine Learning: An Algorithmic Perspective, Boca Raton: Chapman & Hall/CRC , 2009.

[13] K. P. Murphy, Machine Learning: A Probabilistic Perspective, Cambridge: Massachusetts Institute of Technology, 2012.

[14] J. Brownlee, "A Gentle Introduction to the Rectified Linear Unit (ReLU)," Machine Learning Mastery, 20 August 2020. [Online]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/. [Accessed 6 March 2023].

[15] J. Brownlee, "Train-Test Split for Evaluating Machine Learning Algorithms," Machine Learning Mastery, 26 August 2020. [Online]. Available: https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/. [Accessed 6 March 2023].

[16] D. Chicco, M. J. Warrens and G. Jurman, "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE, and RMSE in regression analysis evaluation," *PeerJ Comut Sci.,* vol. 7, no. 623, 2021.

[17] W. Koehrsen, "Overfitting vs. Underfitting: A Complete Example," Towards Data Science, 28 Jan 2018. [Online]. Available: https://towardsdatascience.com/overfitting-vs-underfitting-a-complete-example-d05dd7e19765. [Accessed 2023 6 March].

[18] T. Sutton, T. Donovan, T. Trumbull, P. Dobreff, E. Caro, D. Griesheimer, L. Tyburski, D. Caprenter and H. Joo, "THe MC21 Monte Carlo Transport Code," in *Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications (M&C + SNA)*, Monterey, 2007.

[19] D. S. Blight, B. J. Grayson and J. W. Nielsen, Cycle As Run Calculations of ATR Operating Cycles 144B through 169A Using the Common Monte Carlo Design Tool, INL, 2019.

[20] H. G. Kim, S. H. Chang and B. H. Lee, "A Study on the Optimal Fuel Loading Pattern Design in Pressurized Water Reactor Using the Artificial Neural Network and the Fuzzy Rule Based System," Taipei, Taiwan, 1994.

[21] A. Yamamoto, "A Quantitative Comparison of Loading Pattern Optimization Methods for In-Core Fuel Management of PWR," *Journal of Nuclear Science and Technology,* vol. 34, no. 4, pp. 339-347, 1997.

[22] A. Yamamoto and H. Hashimoto, "Application of the Distributed Genetic Algorithm for In-Core Fuel Optimization Problems under Parallel Computational Environment," *Journal of Nuclear Science and Technology,* vol. 39, no. 12, pp. 1281-1288, 2002.

[23] M. Sadighi, S. Setayeshi and A. A. Salehi, "PWR fuel management optimization using neural networks," *Annals of Nuclear Energy ,* vol. 29, 2002.

[24] A. Fadaei and S. Setayeshi, "LONSA as a toold for loading pattern optimization for VVER-1000 using synergy of a neural network and simulated annealing," vol. 35, 2008.

[25] M. Tombakoglu, K. B. Bekar and A. O. Erdemli, "Performance Evaluation of Genetic Algorithms on Loading Pattern Optimization of PWRs," Portoroz, 2001.

[26] E. F. Faria and C. Pereira, "Nuclear fuel loading pattern optimisation using a neural network," *Annals of Nuclear Energy ,* vol. 30, pp. 603-613, 2002.

[27] J. J. Ortiz, A. Castillo, J. L. Montes, R. Perusquia and J. L. Hernandez, "Nuclear Fuel Lattice Optimization Using Neural Networks and a Fuzzy Logic System," *Nuclear Science and Engineering,* no. 162, pp. 148-157, 2009.

[28] N. J. Hill and G. T. Parks, "Pressurized water reactor in-core nuclear fuel management by tabu search," *Annals of Nuclear Energy,* vol. 75, pp. 64-71, 2015.

[29] A. Castillo, C. Martin-del-Campo, J.-L. Montes-Tadeo, J.-L. Francois, J.-J. Ortiz-Servin and R. Perusquia-del-Cueto, "Comparison of heuristic optimization techniques for the enrichment and gadolinia distribution in BWR fuel lattices and decidion analysis," *Annals of Nuclear Energy,* vol. 63, pp. 556-564, 2013.

[30] O. Safarzadeh, A. Zolfaghari, M. Zangian and O. Noori-karkhoran, "Pattern optimization of PWR reactor using hybrid parallel Artificial Bee Colony," *Annals of Nuclear Energy,* vol. 63, pp. 295-301, 2014.

[31] A. Hedayat, "Developing a practical optimization of the refueling program for ordinary research reactors using a modified simulated annealing method," *Progress in Nuclear Energy,* vol. 76, pp. 191-205, 2014.

[32] S. Kashi, A. Minuchehr, N. Poursalehi and A. Zolfaghari, "Bat algorithm for the fuel arrangement optimization of a reactor core," *Annals of Nuclear Energy,* vol. 64, pp. 144-151, 2014.

[33] R. Barati, "A novel approach in optimization problem for research reactors fuel plate using a synergy between cellular automata and quasi-simulated annealing methods," *Annals of Nuclear Energy ,* vol. 70, pp. 56-63, 2014.

[34] F. Khoshahval, A. Zolfaghari and H. Minuchehr, "A new method for multi-objective in core fuel management optimization using biogeography based algorithm," *Annals of Nuclear Energy,* vol. 73, 2014.

[35] T. K. Park, H. G. Joo and C. H. Kim, "Multicycle Fuel Loading Pattern Optimization by Multiobjective Simulated Annealing Employing Adaptively Constrained Discontinuous Penalty Function," *Nuclear Science and Engineering ,* vol. 176, pp. 226-239, 2014.

[36] A. S. Saber, M. S. El-Koliel, M. A. El-Rashidy and E. T. Taha, "Nuclear Reactors Safety Core Parameters," pp. 163-168, 2015.

[37] S. M. Mahmoudi, M. Aghaie, M. Bahonar and N. Poursalehi, "A novel optimization method, Gravitational Search Algorithm (GSA), for PWR core optimization," *Annals of Nuclear Energy,* vol. 95, pp. 23-34, 2016.

[38] A. V. Sobolev, A. S. Gazetdinov and D. S. Samokhin, "Genetic algorithms for nuclear reactor fuel loaf and reload optimization problems," *Nuclear Energy and Technology ,* vol. 3, pp. 231-235, 16 August 2017.

[39] J. J. Ortiz-Servin, D. A. Pelta, J. M. Cadenas, A. Castillo and J. L. Montes-Tadeo, "Methodology for integrated fuel lattice and fuel load optimization using population-based metaheuristics and decision trees," *Progress in Nuclear Energy,* vol. 104, pp. 264-270, 2018.

[40] A. Ahmad and S.-u.-I. Ahmad, "Optimization of fuel loading pattern for a material test reactor using swarm intelligence," *Progress in Nuclear Energy,* vol. 103, pp. 45-50, 2018.

[41] E. Israeli and E. Gilad, "Novel genetic algorithm for loading pattern optimization based on core physics heuristics," *Annals of Nuclear Energy,* vol. 118, pp. 35-48, 2018.

[42] M. R. Oktavian, A. Agung and A. W. Harto, "Fuel Loading Pattern Optimization with Constraint on Fuel Assembly Inventory Using Quantum-Inspired Evolutionary Algorithm," in *E3S Web of Conferences*, Yogyakarta, 2018.

[43] M. A. Nasr, M. Zangian, M. Abbasi and A. Zolfaghari, "Neutronic and thermal-hydraulic aspects of loading pattern optimization during the first cycle of VVER-1000 reactor using Polar Bear Optimization method," *Annals of Nuclear Energy,* vol. 133, pp. 538-548, 2019.

[44] A. Zameer, M. Muneeb, S. M. Mirza and M. A. Z. Raja, "Fractional-order particle swarm based multi-objective PWR core loading pattern optimization," *Annals of Nuclear Energy,* vol. 135, 2020.

[45] M. a. H. F. Jarrett, "Automated Fuel Management Optimization for Fast Reactors," in *PHYSOR2020*, 2020.

[46] N. H. Manwaring, "ECAR-4767 ATR As-run Analysis for Cycles 166A-1 and 166A-2," INL , 2019.

[47] J. Brasier, "Curves for Establishing ATR Fuel Loadings in the NW, NE, SW, and SE Lobes," Advanced Test Reactor, 1984.