LA-UR-18-20175
Approved for public release; distribution is unlimited.

Title:	Using Machine Learning to Predict MCNP Bias
Author(s):	Grechanuk, Pavel Aleksandrovi
Intended for:	Report
Issued:	2018-01-09

Using Machine Learning to Predict MCNP Bias

Pavel Grechanuk

September 1, 2017

1 Introduction

For many real-world applications in radiation transport where simulations are compared to experimental measurements, like in nuclear criticality safety, the bias (simulated - experimental k_{eff}) in the calculation is an extremely important quantity used for code validation. The objective of this project is to accurately predict the bias of MCNP6 [1] criticality calculations using machine learning (ML) algorithms, with the intention of creating a tool that can complement the current nuclear criticality safety methods. In the latest release of MCNP6, the Whisper tool [2] is available for criticality safety analysts and includes a large catalogue of experimental benchmarks, sensitivity profiles, and nuclear data covariance matrices. This data, coming from 1100+ benchmark cases, is used in this study of ML algorithms for criticality safety bias predictions.

Machine learning methods use algorithms which learn from data, and are subsequently able to make predictions based on new never before seen instances. A type of machine learning called supervised learning feeds the solutions, called labels, to the algorithm with the training data. A common supervised learning task is regression, which tries to predict a numeric value (example: home value) based on a given set of features (example: number of bedrooms, year of construction, square footage, etc.). Typically a machine learning algorithm performs best when there is a large quantity of data available. If the training sample is too small, the data is non-representative of the whole picture, and the algorithm under-performs. This can also occur with a large dataset, if the data is not sampled properly (sampling bias).

Typically the data fed to a machine learning algorithm is split into a test and training set. The standard practice is to train on approximately 80% of the data, and evaluate the model on the remaining 20%. The error rate on the testing set is called the generalization error, which tells you the error rate to expect from your model on instances it has never encountered before. The best way to evaluate the performance of a model is using a technique called cross validation. This method works by splitting up the training set into a certain number of subsets (called folds), and the model is evaluated on one fold and trained on the remaining. This is done for every possible combination until every fold has been evaluated, with the other folds acting as training data. The measures of fit from each round are averaged to calculate a more representative value of model generalization error.

2 Methods

The sensitivity vectors generated by MCNP6, which describe how k_{eff} is impacted by the nuclear data of a given application, are chosen as the features, due to the assumption that they inherently carry enough information to characterize a system, and consequently can be used to find patterns that influence bias (simulated - experimental k_{eff}). The main challenge of preparing the data was that the sensitivity vector length varied between test cases, since they have different materials and reactions, but ML algorithms require that the features all be the same shape. The solution was to find all of the unique isotope reaction pairs (2040 of them), and to create a template onto which all of the sensitivity vectors for each test case are written to. There are 44 energy bins in each sensitivity vector and 2040 individual isotope reaction pairs, so each test case has 89,760 features associated with it, and since there are 1,100 cases this means the algorithm must process almost 100 million data points during training. The models were also evaluated on a modified set of sensitivity vectors, where the sensitivity values for each isotope reaction are summed over energy. This removes the energy dependence, making it so that the models are trained on the benchmark's sensitivity to each isotope reaction. Consequently, this reduces the number of features to 2,040 per benchmark, which significantly reduces training time and speeds up optimization.

Ten fold cross validation was used to evaluate the mean average error (MAE) and the root mean squared error (RMSE) of each algorithm to find the best one. Every model has a set of hyper-parameters that act as tuning nobs, which affect the performance of the model drastically. To find the ideal set of hyper-parameters, a grid search was performed in which the cross validation scores were computed for different combinations of parameter values specified by a grid.

2.1 Decision Trees

Decision Trees are very powerful machine learning algorithms that are used for regression and classification. Scikit-Learn, an open source machine learning module in Python [3], uses the Classification and Regression Tree (CART) algorithm to "grow" decision trees [4]. The algorithm first takes the training data, and tries to split it into two subsets using a feature k and a threshold t. This is a numerical procedure where many threshold points are tested based on a feature in order to minimize a cost function, which for regression is either the mean squared error or the mean absolute error.

Once the dataset is split into two, the subsets undergo the same operation, until the algorithm can not find a split that would reduce the cost function. This eventually leads to a decision tree that has many branches (node splits) and leaves (node ends). The model works by starting at the top of the tree, and undergoing logical operators at each branch node until arriving at a leaf node which has an associated value with it. Decision trees are used in this study due to their superior performance and simplicity when compared to other regression models. There are a variety of subclasses of decision trees, and many of them were tested to find the one with the lowest error rate.

A very useful feature of decision trees is that they can effectively calculate the importance of each feature on the estimated value. The most important features are more likely to appear near the root of the tree, while the less important features will appear further down the tree. Scikit Learn automatically estimates the feature importance by computing the average depth of each feature across all of the tress. The importance can be thought of as the influence of each isotope reaction pair on whether bias is present. Since the features are broken into 44 energy groups, the importance of each isotope reaction can be analyzed at the energy level, and this provides insight into what cross sections or physics models can be improved in MCNP to reduce bias. A script was written to show the most important isotope reaction pair sensitivities on influencing bias, and the most important isotope reaction pairs at the thermal, intermediate, and fast energies.

2.2 Artificial Neural Networks

Artificial neural networks (ANNs) were also investigated for this task, as they are very powerful algorithms used for everything from computer vision, to speech recognition, to playing complicated games like Go. Given sufficient training samples and a large enough network, ANNs are able to model any continuous function. No matter how complicated the function is (many inputs, many outputs), it is guaranteed that there is a neural network that can approximate the output for every input. An ANN is composed of connected units called artificial neurons, which transmit signals to other neurons. The receiving neurons process the signal using an activation function, and then send it downstream. The neurons and the connections between them have a weight that is modified during training using a back propagation algorithm. Typically neural networks are built in layers: the first layer having the same number of neurons as features, then one or more hidden layers of neurons, and the final output layer which has as many neurons as the output of the function that is being approximated. Typically the performance of an ANN is increased by adding more hidden neurons, as they allow the model to learn more complicated relationships.

The process of training an ANN will be explained as simply as possible in the next section; the intricate details and mathematics are outside the scope of this paper and can be found in K. Hornik's, et al. paper [5]. For each training instance the network feeds the data through the network and calculates the output of every neuron in each consecutive layer. The output error is measured at the output (difference between the actual and predicted values), and the influence of each neuron in the preceding layer on the error of the output neurons is calculated. The network then calculates how much of these error contributions came from each neuron in the previous layer, and so forth until the network reaches the input layer. This process measures the error gradient across all network connections by effectively propagating the error gradient backwards through the network [6]. Finally all of the connection weights are slightly adjusted using the propagated error gradient to produce a better model. This process is iterated until the output error does not decrease significantly.

3 Results - Decision Trees

3.1 Adaboost Regression Trees

The AdaBoost regressor is unique in that it goes through multiple iterations where the next iteration tries to correct the predecessor by paying more attention to the instances that the predecessor made mistakes on. The way the method works is by first training a decision tree, and making predictions on a subset of the training set. From the first iteration the weighted predictor error is calculated, from which a predictor weight is extracted (see [7] or [8] for the mathematics). Using the predictor weight the instance weights are adjusted and the misclassified instances are boosted. The next iteration is trained on the data with the new weights, the new predictor weight is calculated, the feature weights are updated, another predictor is trained, and so on.

The process is repeated until the desired number of iterations is reached. To make the predictions on new instances the algorithm calculates the predictions of every predictor, weighs them using their predictor weights, and calculates the mean. This method of grouping many predictors to make a single estimate is called ensemble learning. The performance of the ensemble model is often superior than the performance of any individual tree, which can be attributed to the 'wisdom of the crowd' effect. This method produces results more accurate than the standard decision tree model, due to being an ensemble and increasingly focusing on the cases that it gets wrong. The performance statistics are summarized in Table 1.

Features	MAE	RMSE
89,760	0.00396	0.00576
2,040	0.00385	0.00549

Table 1: Statistics for Adaboost regressor from 10 fold cross validation.

From Table 1 it can be seen that including all of the data points produces a marginally better model. The mean squared error for the Adaboost regressor using 2,040 features, and a histogram of the difference between predicted and actual bias can be seen in Figure 1.

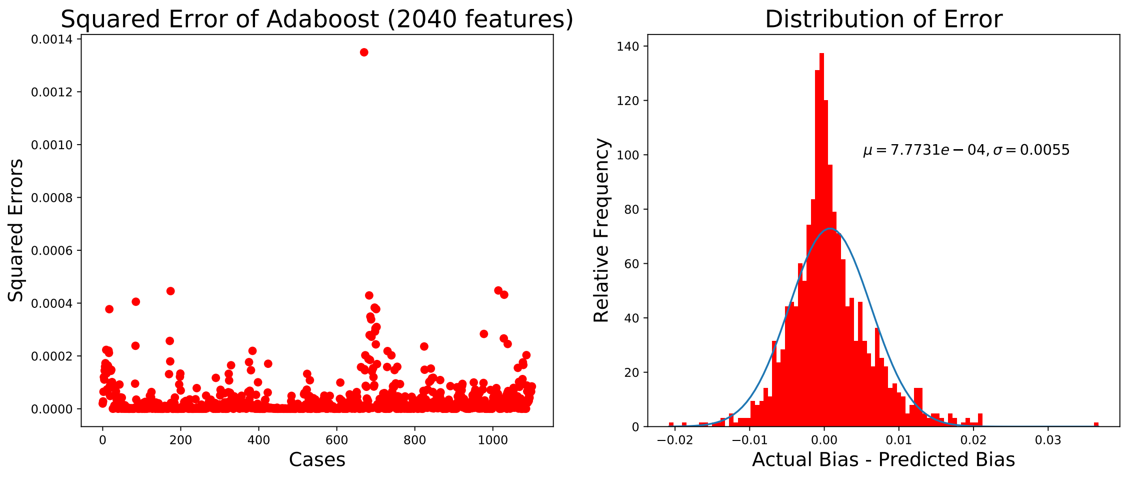

Figure 1: Mean squared error (left), and histogram of prediction errors (right) for the Adaboost regressor, obtained from 10 fold cross validation.

Looking at the histogram it can be seen that the error on the predictions forms a normal distribution. There are a multitude of factors which affect the error rate, and due to the central limit theorem we can assume that the sum of the individual effects will tend to form a normal distribution, which they appear to do. An identical plot is generated for the larger data set and can be seen in Figure 2. Both models have a similar performance, and they seem to make mistakes on the same cases.

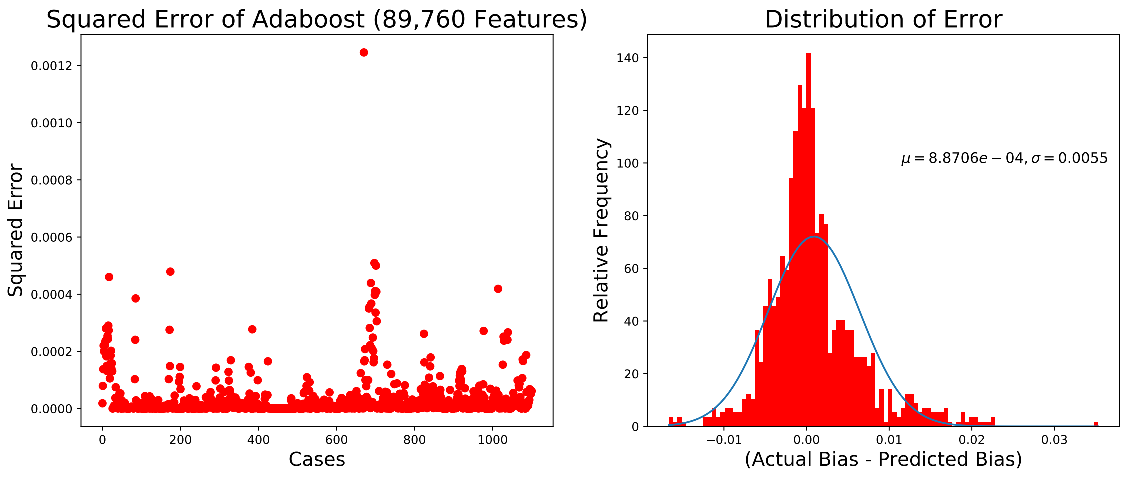

Figure 2: Mean squared error (left), and histogram of prediction errors (right) for the Adaboost regressor, obtained from 10 fold cross validation.

Using the built in features in Scikit Learn the importance of each isotope reaction was calculated for both data sets, and can the top 10 can be found in Table 2. The importance across all reactions adds up to unity, so the numbers are the relative importance of the reaction's influence on bias.

89,760 Features	2,040 Features
92232.80c total nu, 0.101553	92232.80c total nu, 0.094887
6000.80c elastic, 0.063773	92232.80c fission, 0.075492
94239.80c ngamma, 0.063368	**92233.80c ngamma, 0.062631**
6000.80c ngamma, 0.061759	8016.80c elastic, 0.033377
92232.80c elastic, 0.059249	6000.80c ngamma, 0.029548
poly.20t inelastic, 0.037004	92234.80c ngamma, 0.026564
92232.80c fission, 0.028408	**92232.80c n2n, 0.022677**
8016.80c elastic, 0.028022	**92232.80c ngamma, 0.020410**
92234.80c ngamma, 0.018935	poly.20t inelastic, 0.017924
94240.80c elastic, 0.018753	**6000.80c nalpha, 0.015111**

Table 2: Top ten relative importances of each isotope reaction to the Adaboost regressor on calculating bias.

Many of the reactions with the highest importance are shared among the two models which is expected, as the main difference is the form of the sensitivity vectors. The features that are not shared among the top ten between the two datasets are marked in bold. The model trained on 2,040 features only sees the integral importance of each reaction, and the energy dependence of the sensitivity vector is lost.

3.2 Random Forest Decision Trees

The random forest regressor is an ensemble method that trains a number of decision trees each on a random subsets of the data, with the intention of reducing over-fitting. The individual trees also differ because instead of looking for the best feature to split the tree, it searches through a random subset of features to find the best split. This introduces extra randomness into the decision trees, and results in an increased tree diversity. This ultimately trades a higher bias for a reduced variance, which reduces over-fitting and most often results in a better overall model. The statistics for this model can be seen in Table 3.

Features	MAE	RMSE
89,760	0.00397	0.00572
2,040	0.00374	0.00537

Table 3: Statistics for Random Forest regressor from 10 fold cross validation.

The random forest model slightly under-performs compared against the Adaboost regressor, and by looking at the mean squared error plot in Figure 3 (2,040 features), it can be seen that both algorithms make mistakes on the same cases (the order is the same).

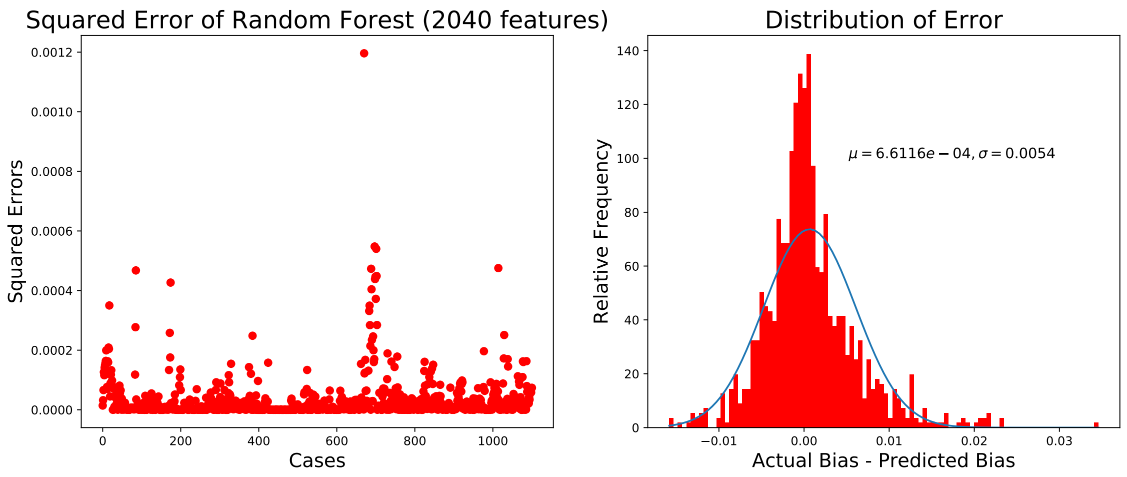

Figure 3: Mean squared error (left), and histogram of prediction errors (right), obtained from 10 fold cross validation.

It can be seen that the difference between the real and predicted bias forms a normal distribution. This same plot is recreated for the dataset with 89,760 features, and can be seen in figure 4.

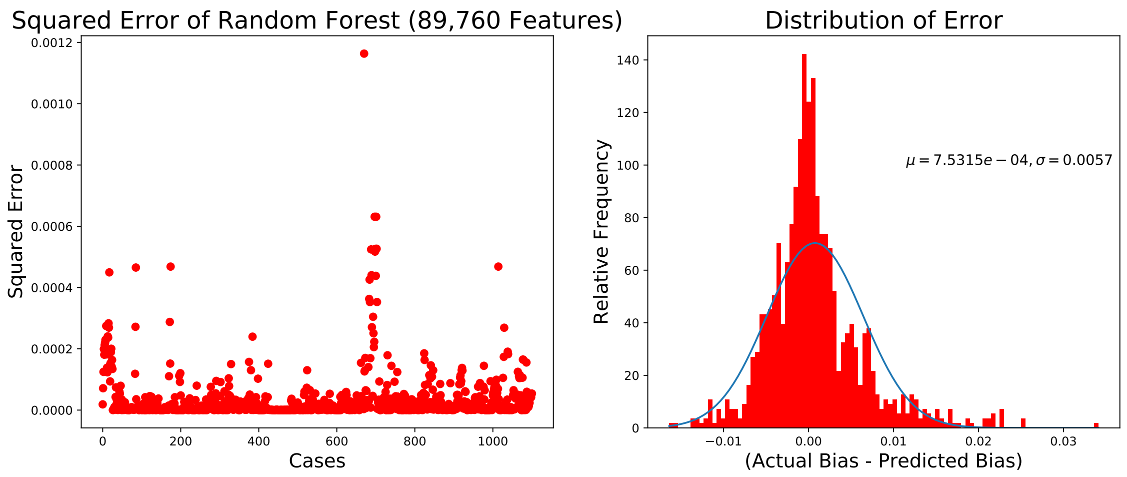

Figure 4: Mean squared error (left), and histogram of prediction errors (right), obtained from 10 fold cross validation.

The order of cases between the two data sets is the same, and it can be seen that the average magnitude of errors from the both models is relatively similar. It can also be seen that both models make errors on the same cases, which will be discussed in the Discussion section. The importance of each feature was calculated for both models and the top ten most important features can be found in the Table 4.

89,760 Features	2,040 Features
92233.80c n,gamma, 0.043283	92234.80c total nu, 0.025210
92233.80c total nu, 0.036926	92234.80c fission, 0.023612
92233.80c fission, 0.036024	92234.80c n,gamma, 0.021593
6000.80c n,gamma, 0.027969	92232.80c total nu, 0.018872
6000.80c elastic, 0.024428	92232.80c n,gamma, 0.017970
92234.80c n,gamma, 0.024359	6000.80c n,gamma, 0.017487
94239.80c fission, 0.021407	9019.80c n,p, 0.017361
94239.80c n,gamma, 0.021331	92233.80c n,gamma, 0.017193
94239.80c total nu, 0.020735	92233.80c inelastic, 0.017156
9019.80c elastic, 0.018545	9019.80c n,alpha, 0.016872

Table 4: Relative importances of each isotope reaction to the Random Forest regressor on calculating bias.

The importances of these features differ from the Adaboost case, because of the way the Random Forest randomly samples features from the whole feature set. This spreads out the relative importance, since instead of looking for the best feature for a split from all of the features, the model selects one from a random subset of features. This generates a variety of trees who each focus on a subset of the total features to make their predictions. Due to the larger dataset incorporating energy dependence, the relative importance of each reaction can be found by energy group. The top ten most important reactions separated by energy group can be found in Table 5.

Thermal (0 - 0.625 ev)	Intermediate (1.0 ev - 0.1 Mev)	Fast (0.4 Mev - 20 Mev)
6000.80c n,gamma, 0.014562	**92233.80c n,gamma, 0.018457**	**92233.80c fission, 0.015264**
92233.80c total nu, 0.011437	**92233.80c fission, 0.015724**	**92233.80c inelastic, 0.013543**
92233.80c n,gamma, 0.010641	**92233.80c total nu, 0.012844**	**92233.80c n,gamma, 0.012739**
92234.80c n,gamma, 0.009479	**92234.80c n,gamma, 0.011945**	**92233.80c total nu, 0.012644**
1001.80c n,gamma, 0.009069	**94239.80c n,gamma, 0.011687**	9019.80c inelastic, 0.010355
poly.20t inelastic, 0.008879	6000.80c n,gamma, 0.008924	6000.80c elastic, 0.009997
be.20t elastic, 0.008204	**94239.80c total nu, 0.008325**	**92233.80c fission chi, 0.008758**
94239.80c n,gamma, 0.007522	**94239.80c fission, 0.008208**	**92234.80c total nu, 0.008494**
94239.80c fission, 0.007427	6000.80c elastic, 0.007817	**92234.80c fission, 0.008008**
9019.80c n,gamma, 0.007201	**92232.80c total nu, 0.006668**	1001.80c elastic, 0.007938

Table 5: Top ten relative importances of each isotope reaction energy group to the Random Forest regressor on calculating bias. These importances make physical sense, capture at low energies, capture and fission for actinides at intermediate energies, inelastic scattering and fission at high energies.

It can be seen that all three energy groups have important cases that are used to predict bias.

The uranium and plutonium isotopes are marked in bold, and it can be seen that they are prevalent across all three energy groups, which makes sense since this benchmark suite primarily deals with those metals. An interesting note is that the neutron gamma reactions have a large importance across the thermal and intermediate energy groups.

3.3 Extremely Randomized Trees

The extra trees method is identical to the random forest ensemble method in the fact that it uses a random subset of the features at every split. The main difference is that extra randomness is inserted by also searching for the best slit using random thresholds for the features. A model using these more randomized trees is called an extremely randomized tree ensemble. This model again trades increased bias for a reduced variance, and it often performs very similarly to the random forest model. The performance of the model is summarized in Table 6.

Features	MAE	RMSE
89,760	0.00413	0.00596
2,040	0.00393	0.00572

Table 6: Statistics for Extremely Randomized Trees regressor from 10 fold cross validation.

Again this model performs very similarly to the other decision tree regressors, primarily since they are all based of the same algorithm (CART). The plot of the squared errors and error distribution for the 2,040 feature dataset can be seen in Figure 5.

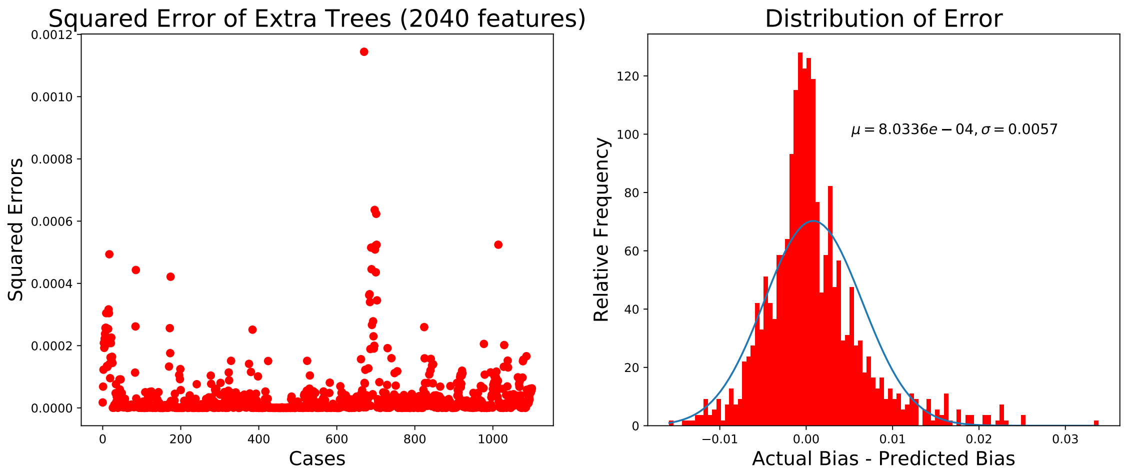

Figure 5: Mean squared error (left), and histogram of prediction errors (right), obtained from 10 fold cross validation.

An identical plot for the larger data set can be seen in Figure 6, and again the models are very similar.

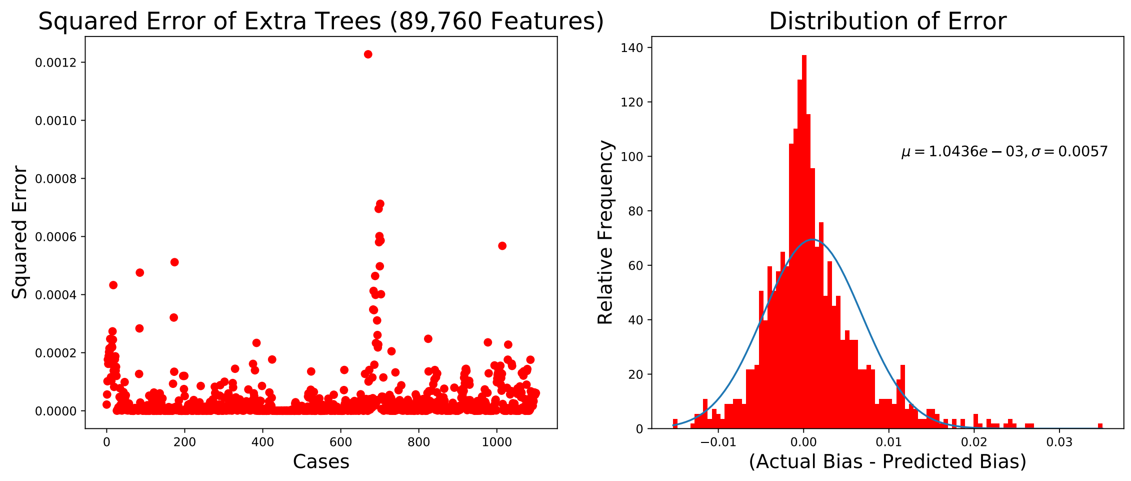

Figure 6: Mean squared error (left), and histogram of prediction errors (right), obtained from 10 fold cross validation.

The relative importances of the features can be seen in Table 7

89,760 Features	2,040 Features
92232.80c fission, 0.057565	92232.80c total nu, 0.043314
92232.80c total nu, 0.057309	92232.80c fission, 0.034945
92232.80c n,gamma, 0.049576	6000.80c inelastic, 0.028446
6000.80c n,gamma, 0.026784	92232.80c n2n, 0.024835
92233.80c fission, 0.023372	92232.80c n,gamma, 0.023902
94239.80c total nu, 0.021561	92233.80c n,gamma, 0.015658
94239.80c n,gamma, 0.018211	92235.80c total nu, 0.014238
92233.80c total nu, 0.015394	6000.80c n,gamma, 0.013787
92233.80c n,gamma, 0.015269	92234.80c n,gamma, 0.013068
9019.80c n,gamma, 0.015244	94239.80c n,gamma, 0.012768

Table 7: Relative importances of each isotope reaction to the Extra Trees regressor on calculating bias.

All three models share some of the same isotope reactions in the top ten important features, which indicates that those reactions have the most importance. Some of the most common reactions among the top ten important features from the models involve n, gamma reactions for U-233, U-234, U-232, and carbon.

3.4 Including Simulated k in Training

The whole goal of this project is to predict the bias using sensitivity vectors, and in order to generate the vectors a MCNP simulation must be run, which also generates $k_{simulated}$. The simulated k_{eff} value is another piece of information that can be used as a feature to improve the accuracy of the model. There is a strong linear relationship between computational bias and k_{sim}, which can be seen in Figure 7.

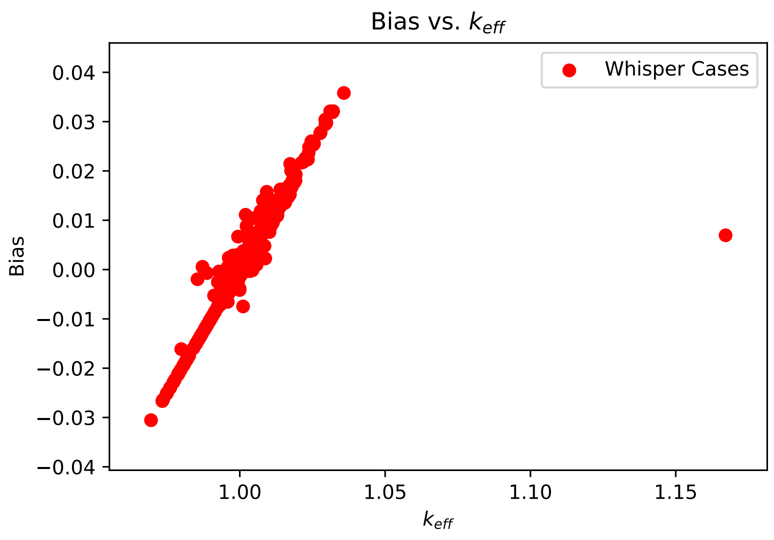

Figure 7: Bias vs k_{sim} for all the cases in the Whisper library.

A new dataset was created with k_{sim} included, so the two datasets now have 2,041 and 89,761 features respectively (k_{sim} followed by the sensitivity vector). The decision tree algorithms were trained on the new dataset, and the performance improved as can be seen in Table 8.

Model	MAE	RMSE
Adaboost (L)	0.00102	0.00217
Random Forest (L)	0.00253	0.00415
Extra Trees (L)	0.00326	0.00484
Adaboost (S)	0.00103	0.00216
Random Forest (S)	0.00251	0.00397
Extra Trees (S)	0.00340	0.00497

Table 8: Statistics for decision tree models with k_{sim} in training, from 10 fold cross validation (S = 2,041, and L = 89,761 features).

The Adaboost model performs the best for both datasets, while the performance of the other models improves marginally. It is interesting to note that the performance of the other models does not improve significantly with the addition of k_{sim}, this is particularly interesting since there is such a strong linear relationship between bias and k_{eff}. To find the cause of this the feature importances for k_{sim} were investigated and can be seen in Table 9.

Model	Feature Importance of k_{sim}
Adaboost	0.868030
Random Forest	0.156393
Extra Trees	0.144139

Table 9: Relative importances of k_{sim} in the different models.

From the table is can be seen that the Adaboost model seems to be over fitting to the k_{sim} bias relationship. Even though it's performance statistics are better than the other models, it is unlikely to generalize well to cases that lie outside of the k_{sim} bias line. It seems that the Adaboost model iteratively strengthened the weight of k_{sim} to the point that the model is over-fitting on k_{sim} and disregarding the other features. The Random Forest and Extra Trees models avoid this because they randomly sample features from the feature set for every tree that they grow, and as such they are much more likely to generalize well to cases that lie outside of the linear bias k_{sim} relationship.

4 Results - Neural Networks

A large variety of neural networks were trained and evaluated using Scikit - Learn to find the best one. A variety of activation functions for the neurons, back propagation learning algorithms, and other hyper parameters were tuned to find the most accurate model. The best assortment of hyperparameters was found using a grid search, and they are the neuron exponential linear activation function, the adadelta learning algorithm, and a learning rate of 0.01. Another factor that was considered was the shape of the hidden layers. The smaller dataset was used because the networks take a lengthy amount of time to train. Even when using thirty nodes on the computing servers, it still takes over 5 hours to train a single network, and much longer to perform cross validation. The input and output layers of every model is the same, with the primary variation being the shape and number of neurons in the hidden layers. The performance of the two best models is summarized in Table 10. (I'm working on adding another one Mike.)

Hidden Layers Shape	MAE	RMSE
1020, 510, 255, 128, 64, 32, 16	$3.63348 * 10^{-5}$	$5.36915 * 10^{-5}$

Table 10: Statistics for neural networks, from 10 fold cross validation. Data flows from left to right.

The performance of the the neural networks is superb, however they are essentially black boxes, when compared to decision trees. It is virtually impossible to understand why neural networks make the predictions that they do, while for decision trees you can actually visualize the tree and see how each node is split on the features. However the utility of these models should not be overlooked just because it is not clear how they make their predictions, since they are very accurate at predicting the bias of a criticallity calculation. An interesting note is that the relative errors that the neural networks make are uncorrelated with the errors of the decision trees, as can be seen in Figures 8 & 9 .

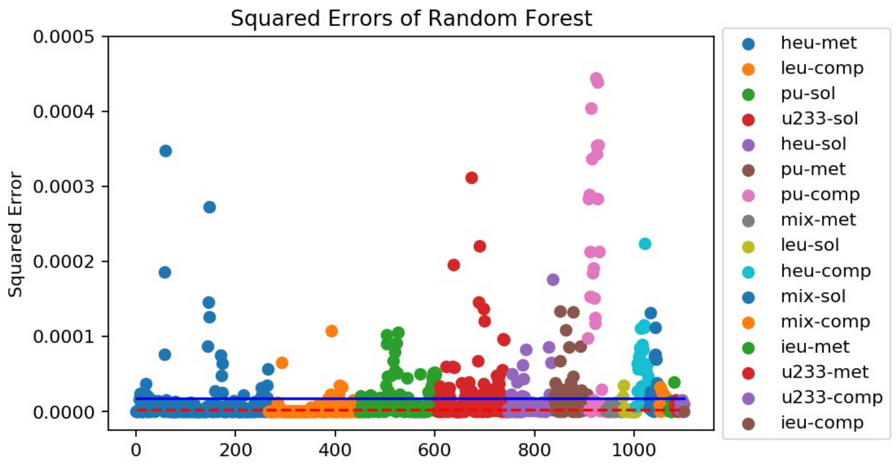

Figure 8: The errors among the different cases for the most accurate random forest model.

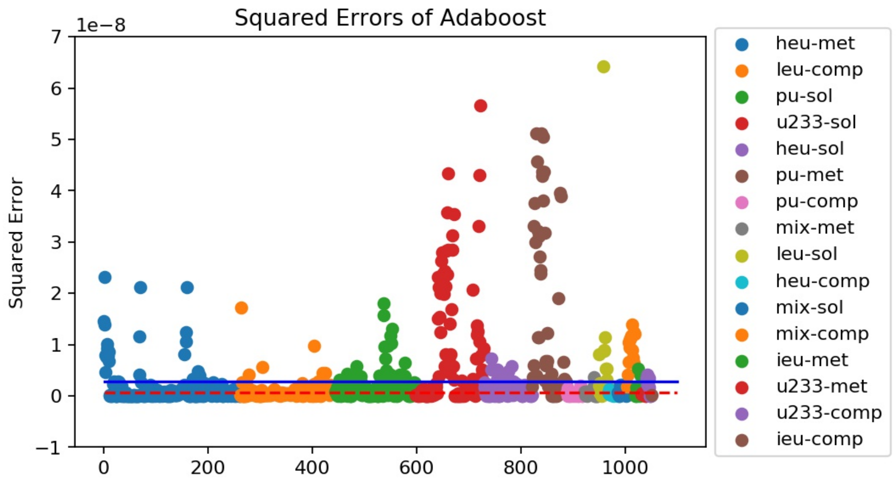

Figure 9: The errors among the different cases for the most accurate neural network model.

The highest errors for the decision trees are the pink pu-comp and the light blue heu-comp cases, while the neural networks excel at those cases and have higher relative errors for the red u233-sol and the brown pu-met cases. The overall error for the neural network is much lower, but it's relative error between the cases is much different. The cause of this is not clear, but it provides an opportunity for a machine learning method called stacking. The idea is to train a ML model on the outputs of other ML models, with the intention of the top model learning to see what errors the bottom layer makes, and learns to correct for them. The fact that the errors between the decision trees and neural networks are uncorrelated provides an opportunity to generate an even more powerful model. The accurate and precise performance of these models reinforces that the sensitivity vectors are excellent features in that they carry enough information to fully characterize a system, which is then used by the models to find patterns which are used to predict the bias accurately.

5 Discussion

5.1 Errors

An interesting observation is that many of the decision tree models make mistakes on the same cases. For example, among the 2,040 dataset there are three cases which have a MSE of greater than 0.0003 throughout every model. The case with the highest consistent error is the mix-sol-therm-001-011 which fluctuates around a MSE of 0.001. To begin examining why all of the models make errors on this case, the bias and k_{eff} off all the mix-sol-therm cases was plotted, which can be seen in Figure 10.

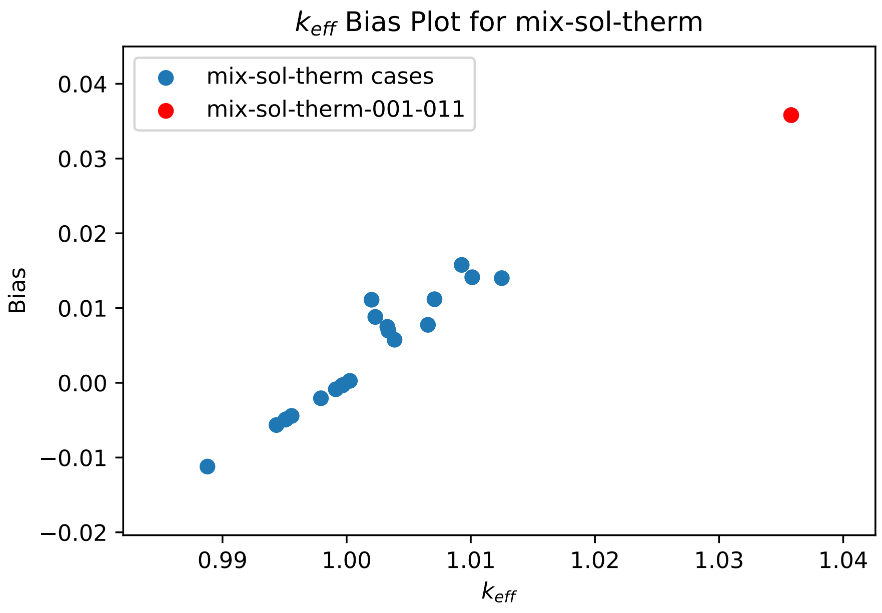

Figure 10: The mix-sol-therm cases from Whisper along with the case with the most error.

As can be seen by the plot the 001-011 case is quite different than the rest of the mix-sol-therm benchmarks (much higher k_{eff}), which explains why the models make such large errors on this case. The problem has a similar sensitivity profile (average cosine similarity of .995 with other cases), but it is a much larger bias, so the model is confused after being trained on the other cases with a much lower bias. This reinforces that machine learning algorithms do not perform well on instances that are very dissimilar to the training set. For this case, the sensitivity vectors are not adequate to reliably predict bias, since the case is much different than the cases which have similar sensitivity vectors. Additionally the error is compounded by the small sample size, as there are only 21 mix-sol-therm cases, which is a tiny sample size by machine learning standards. By removing the 001-011 case from the dataset the performance of the random forest regressor on 2,040 features reduces the MSE by 4.5%.

Another instance on which the models make large errors on is the heu-met-fast-057-003 case. In this instance the bias lies relatively outside of the main cases, which can be seen by figure 11.

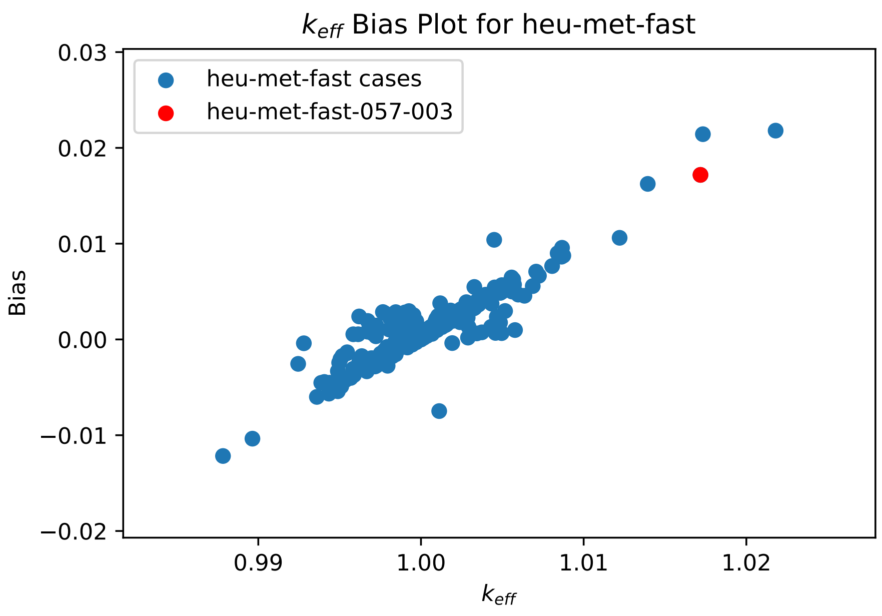

Figure 11: The heu-met-fast cases from Whisper along with the case with the most error.

There are only five heu-met-fast cases with a k_{eff} greater than 1.01, which is fairly small sample, and their sensitivity vectors are different than the other heu-met-fast cases. A good training sample is extremely important to the resulting accuracy of a ML model. The MSE and the RMSE for every type of problem with more than 10 instances can be found summarized in Figure 12.

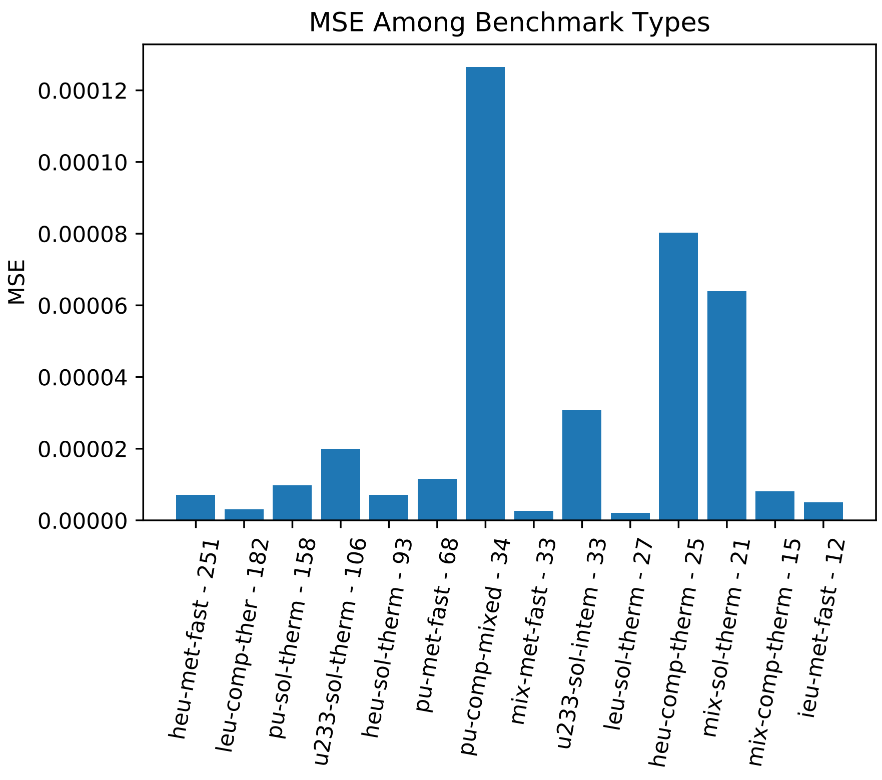

Figure 12: The MSE from the Random Forest model for every case with more than 10 instances. The number after the name is the number of cases in that category.

From the plot it can be seen that the performance on most of the cases is good, while errors are above the mean for the Pu and heu composites and the mixed and U-233 solution cases. For cases with less than 68 cases (pu-met-fast) there is increased variance in the errors, so for any real world applications of these models it should be a rule of thumb to have at least 50 similar cases. The performance and generalization ability of the decision tree models would improve, if there were more training samples for the rare cases. The performance of these models could also improve if there were more features to fit the models to, such as the average neutron energy causing fission, the prompt removal lifetime, the percentages of fissions caused by neutrons in the thermal, intermediate, and fast neutron ranges, and other data provided by MCNP6 outputs. It is difficult to predict what features will lead to good results, and the only way to find out is to experiment and evaluate the models using different features.

5.2 Removing Outliers

The cases on which the models make errors are effectively outliers, since there are not enough similar cases to make accurate predictions. The increased errors on these cases severely impact the statistics on the model, especially the rmse as it is very sensitive to outliers. The standard procedure for dealing with outliers in machine learning is to remove them, so that when the models are trained they are not skewed. A summary of model performance after removing the top 50 cases (4.5% of total cases) which produce the highest error can be found in Table 11.

Model	Mean Absolute Error	Root Mean Squared Error
Adaboost (L)	0.00083	0.00161
Random Forest (L)	0.00210	0.00355
Extra Trees (L)	0.00279	0.00340
Adaboost (S)	0.00077	0.00146
Random Forest (S)	0.00212	0.00296
Extra Trees (S)	0.00256	0.00346

Table 11: Statistics for decision tree models from 10 fold cross validation after removing outliers (S = 2,041, and L = 89,761 features).

The statistics of the models improves significantly as is expected, since the outliers were the cases with the greatest error. To better predict on those difficult cases either more training data or a more complicated model is needed. An interesting note is that of the 50 cases that were excluded here, 34 are automatically excluded from Whisper for failing the chi squared test and treated as outliers.

5.3 Feature Importances

Among the expected isotope-reactions that are among the most important to the decision, trees there are some rare elements like U-234, specifically the neutron gamma reaction. Looking at the leu-comp-therm-079-010 case, which is modeling a reactor, the atom density of U-234 is 5.1985e-6 $\frac{10^{24} atoms}{cm^3}$, while the atom density of the fuel rod is 7.01323699e-2 $\frac{10^{24} atoms}{cm^3}$, so it makes up about 0.007412 % of the rod. The average non zero sensitivity summed over all energies for that case is 0.00726, and the sensitivity for the U-234 neutron gamma reaction is .000913, or 12.58% of the average. The isotope makes up a minuscule proportion of the mass in the rod, and yet it contributes more to the sensitivity than would be expected. The isotope of U-236 makes up a similar atom density to U-234 in the fuel rod, and yet the U-234 neutron gamma reaction has three times the sensitivity of the U-236 neutron gamma reaction. This pattern of low concentration and high sensitivity would be hard to spot just by looking at the sensitivity profiles, but machine learning algorithms excel and finding patterns in big data.

Feature importances were also explored using feature ranking with recursive feature elimination and cross-validated selection of the best number of features (RFECV)[3]. This method works by first fitting to all of the features, and the two least important features are dropped. This continues as an iterative process using cross validation at each step to evaluate model performance until the model stops improving with successive iterations. This removes both redundancy and noise from the data set, as in the end the model is only trained on the features that are important to predicting the bias. This was performed using the summed sensitivity vectors with the random forest regressor excluding k_{siml} as a feature, and a list of only the crucial features was extracted. Next the remaining features (1,482 of them) were fed back into a random forest model to find the feature importances and they can be found in Table 12.

Isotope Reaction	Relative Importance
92233.80c n,gamma	0.046818
92232.80c total nu	0.045100
92232.80c fission	0.039334
92234.80c n,gamma	0.035280
6000.80c n,gamma	0.032351
92234.80c fission	0.031656
92234.80c total nu	0.030931
92232.80c n,gamma	0.027735
6000.80c n,alpha	0.025528
6000.80c inelastic	0.024418

Table 12: Top ten relative importances of each isotope reaction to the random forest regressor after performing RFECV.

Most of these isotope reactions are already in the top ten throughout the importances of the other models. However, this list is most likely the most accurate when it comes to predicting feature importances. An interesting note is that there are only four isotopes present in this list natural carbon, U-232, U-233, U-234. These are the isotopes that should be looked at first, when trying to understand the source of bias in MCNP calculations. The feature importances provide a unique insight into the sensitivity of bias to the various isotope-reaction pairs, and should be explored further.

The performance of the random forest model with this reduced feature set improved dramatically when compared to the random forest model in section 3.2, which uses all of the features. By removing the features that were noisy or irrelevant the RMSE dropped from .00545 to 0.00373 a 31% reduction in error. This shows that it is very important to get rid of the irrelevant features as they have an adverse affect on the overall model performance, and it shows that the models can produce accurate results even in the absence of k_{sim} in the data. This further reinforces that the sensitivity vectors are excellent features to use for ML tasks in criticallity calculations.

5.4 Comparing Against Whisper Bias & GLLSM

Whisper calculates bias using extreme value theory (EVT) in order to provide a conservative estimate for criticallity calculations, in order to provide an adequate margin of safety. The error statistics for Whisper are relatively large compared to the ML rates, with a RMSE of 0.01329 and a MAE of 0.00906. Following is a plot comparing the Bias calculated by Whisper against the bias predicted by the random forest model.

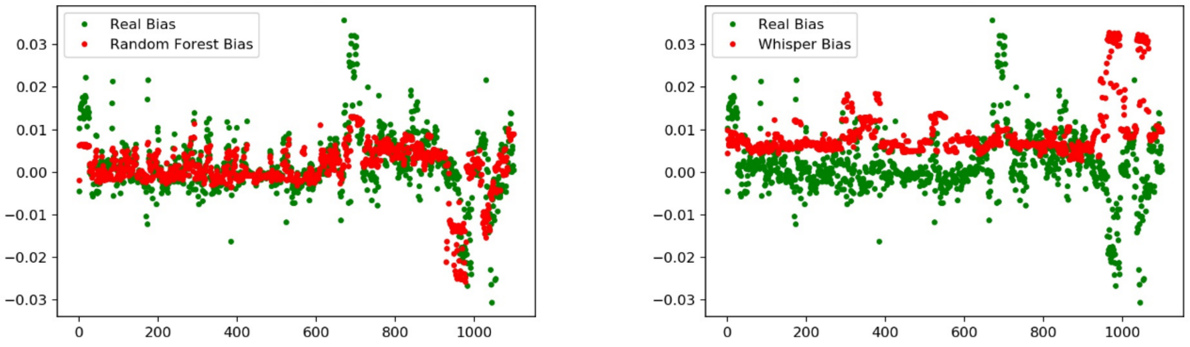

Figure 13: The random forest predictions (left), and the whisper predictions (right).

From the plots it can be seen that Whisper's predictions are more or less along a straight line above the actual bias, while the random forest fits the actual bias more closely. This is expected as Whisper uses EVT to provide a conservative estimate, which tries to be larger than the actual bias. An interesting note is that whisper and the random forest model make errors on the same cases. Both of the models under predict the bias for some U-233 solution cases (the peak between 600 and 800), and some of the heu metal cases near the first benchmarks. Overall the random forest model is using patterns in the sensitivity vectors to make decent predictions, but errs on many of the more difficult cases, which signals that a more powerful model is needed.

Whisper also uses the generalized linear least squares method (GLLSM) to calculate a minimum margin of sub-criticality due to nuclear data uncertainties. Specifically the GLLSM method is used to obtain an adjusted nuclear data covariance library, which is then used to modify the cross-sections in order to increase the agreement between calculated and experimental k_{eff} measurements. This produces a modified value for k_{sim} that is closer to the experimental one, and has a reduced uncertainty. The resulting bias from the GLLSM method was calculated for the Whisper dataset and can be seen plotted in Figure 15.

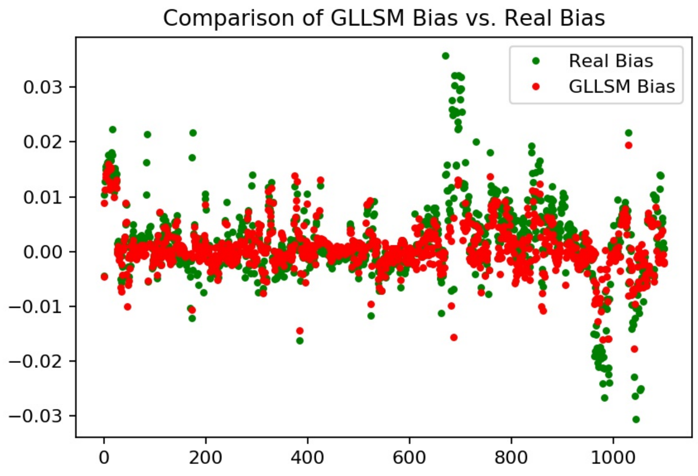

Figure 14: The bias predicted using the GLLSM method compared against the real bias.

These predictions are more accurate than the bias calculated by Whisper's EVT, but much less conservative. Just by looking at the graph it is tough to tell if it is better than the random forest model, but by looking at the summary statistics for all the models in Table 13, it can be seen that the ML models outperform the GLLSM method.

Model	Mean Absolute Error	Root Mean Squared Error
Whisper	0.00906	0.01329
GLLSM	0.00645	0.00959
Extra Trees (S)	0.00256	0.00346
Random Forest (S)	0.00212	0.00296
Adaboost (S)	0.00077	0.00146
Neural Network (S)	$3.63348 * 10^{-5}$	$5.36915 * 10^{-5}$

Table 13: Statistics for the machine learning models from 10 fold cross validation, GLLSM, and Whisper.

5.5 Predicting k_{meas} Instead of Bias

The models were also trained and evaluated with k_{meas} as the target instead of the bias; the two targets have the same units, so it was assumed that the results would be very similar. This was initially done so that a C/E plot could be created in order to visualize the percent difference of the predictions from experiment. A C/E plot can be found in the following figure for the Adaboost regressor using all of the features and all of the benchmarks.

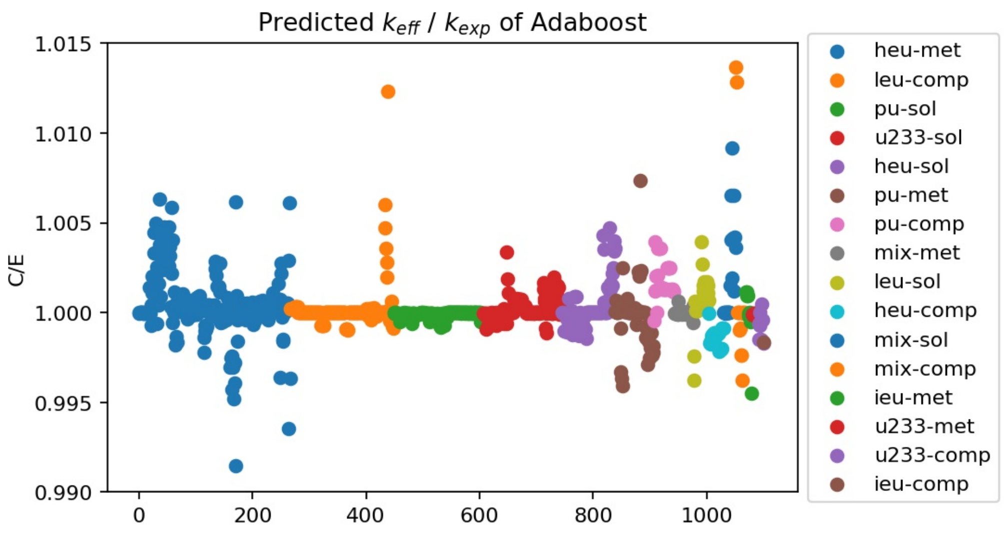

Figure 15: The predicted k_{eff} divided by the measured.

It can be seen that for many of the cases the model is very accurate, although the heu-met cases have a lot of variance. Also the cases of the right side with less similar instances have an increased variance, which is expected. The 50 cases with the greatest error were removed, the models were retrained, and the resulting statistics can be found in Table 14.

Model	Mean Absolute Error	Root Mean Squared Error
Extra Trees (L)	0.00074	0.00133
Random Forest (L)	0.00079	0.00136
Adaboost (L)	0.00057	0.00125

Table 14: Statistics for the decision tree models trained on the large dataset, from 10 fold cross validation when predicting k_{meas}.

It came quite as a surprise that the performance of the models predicting k_{meas} is superior than those trying to predict bias. The units of both bias and k_{meas} are the same, so the statistics can be directly compared. The cause of the increased accuracy is not clear, but I assume it is because the models learn that the value of k_{meas} is fairly close to the value of k_{sim}. The models also make errors on different cases when predicting k_{meas} as can be seen by Figure 16.

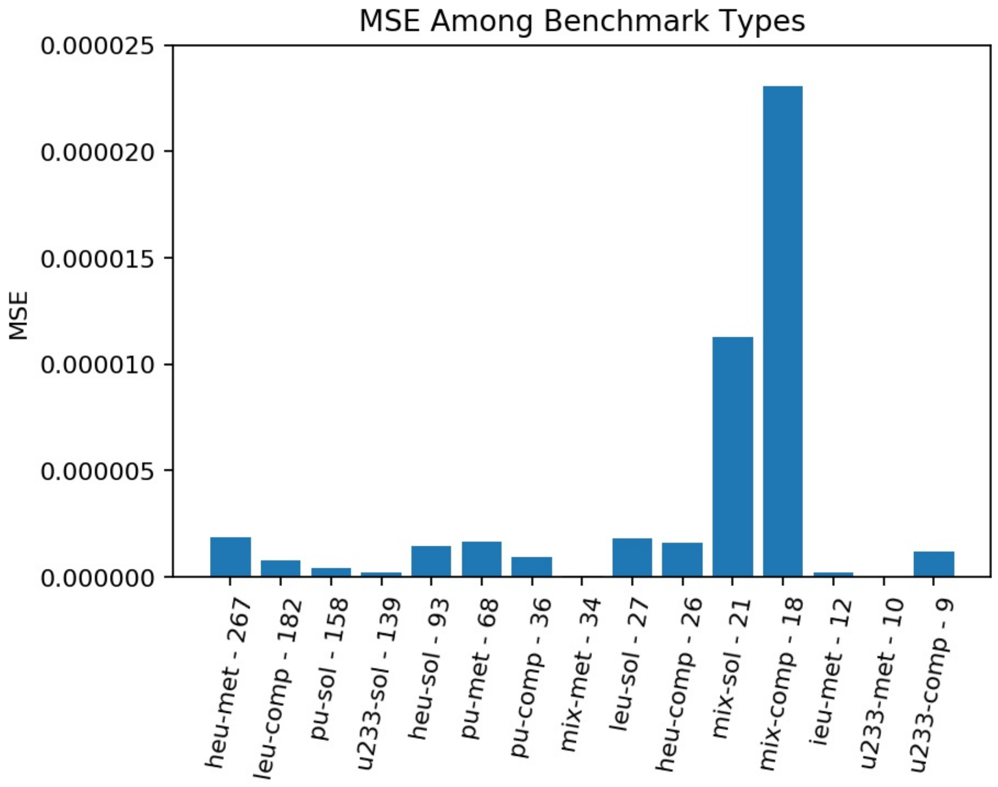

Figure 16: Errors by benchmark type, when predicting k_{meas} by the Adaboost model.

For instance the models predicting bias have very high errors on the Pu composite cases while these models are much more accurate on them. Overall the performance of the models predicting k_{meas} is superior than those predicting bias, and they should be used when looking for an accurate estimation of bias.

6 Conclusion

The mean standard deviation of experimental k_{eff} measurements for the benchmarks used in this study is 0.00328, and the mean absolute error is below that for all of the models. Which means that on average the error on the predicted bias is less than the uncertainty of the experimental k_{eff} measurements. The accuracy of these models verifies that the sensitivity vectors are able to characterize a problem effectively, and provide an applicable feature for machine learning algorithms to predict bias. Machine learning algorithms have shown to be very accurate in predicting bias by using the sensitivity vectors and $k_{simulated}$ as features. Additionally the feature importances from decision trees are able to inform what isotopes and reactions are leading to a divergence between MCNP6 and experimental k_{eff} values. Additional research needs to be done to verify the predictions of the feature importances, and to investigate the performance of machine learning algorithms on a more diverse set of benchmarks.

7 Future Research

- The predictions for the feature importances for the reactions should be explored to see if they are accurate. They could be compared against the changes that the GLLS method that whisper uses to reduce uncertainty from cross section data. A perturbation study of the isotopes cross sections that have high feature importances could be done to see how much k_{sim} changes.
- A study similar to this should be repeated with more features like the average neutron energy causing fission, the prompt removal lifetime, the percentages of fissions caused by neutrons in the thermal, intermediate, and fast neutron ranges, and other data provided by MCNP6 outputs. More benchmarks could also be added for the cases that that number less than 50 so the model's generalization can be improved.
- Since machine learning methods have shown to be accurate in cases with very large datasets and challenging problems, it would be interesting to see how well they would perform in classification of special nuclear material based on detector responses. The outputs by the Detector Response Function Toolkit could be generated for thousands of cases while varying the shielding and source to train an algorithm to identify the specific isotopes present and the enrichment.
- Neural networks and decision trees should be further explored using stacking to see if a better model can be created.

Acknowledgements

Thank you Michael Rising for your support and flexibility in my research this summer. This work was supported by the DOE Nuclear Criticality Safety Program, funded and managed by the National Nuclear Security Administration for the Department of Energy.

References

[1] J. T. Goorley, M. James, T. Booth, F. Brown, J. Bull, J. Cox, and J. Durkee, "Initial mcnp6 release overview," *Nuclear Technology*, vol. 180, Jan 2012.

[2] B. C. Kiedrowski, F. B. Brown, J. L. Conlin, J. A. Favorite, A. C. Kahler, A. R. Kersting, D. K. Parsons, and J. L. Walker, "Whisper: Sensitivity/uncertainty-based computational methods and software for determining baseline upper subcritical limits," *Nuclear Science and Engineering*, vol. 181, no. 1, pp. 17–47, 2015.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[4] D. G. Denison, B. K. Mallick, and A. F. Smith, "A bayesian cart algorithm," *Biometrika*, vol. 85, no. 2, pp. 363–377, 1998.

[5] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[7] D. P. Solomatine and D. L. Shrestha, "Adaboost. rt: a boosting algorithm for regression problems," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2, pp. 1163–1168, IEEE, 2004.

[8] H. Drucker, "Improving regressors using boosting techniques," in *ICML*, vol. 97, pp. 107–115, 1997.