Data Mining as a Means of Confirming Candidate Exoplanets

Owen Gordon
B.S. Computer Science
Indiana University
Bloomington, IN, USA
owgordon@iu.edu

Jonah Slack
B.S. Computer Science
Indiana University
Bloomington, IN, USA
joslack@iu.edu

ABSTRACT

Using that data from the NASA Exoplanet Archive, machine learning models can be trained with existing exoplanet classifications to classify candidate exoplanets. Six different algorithms were used: Naive Bayes, KNN, Decision Tree, Random Forest, Principal Component Analysis, and Neural Networks. The random forest model had the highest five-fold cross validation accuracy after training of 91.8%. To apply the models to candidate data, a regression method was used to score the candidate planets from 0 to 1 on a likelihood the object was an exoplanet. The neural network models used for this regression reduced the search space by three orders of magnitude.

I. INTRODUCTION

Since the dawn of time, humanity has wondered if there are others like us floating among the stars on planets similar to our own. In pursuit of this question, thinkers and dreamers have developed methods to gaze into the sky to look for an answer. A substantial leap forward in this quest was the Kepler Space telescope, named for astronomer Johannes Kepler, and launched into heliocentric orbit by NASA in March, 2009. The goal of this mission was to look for transit events across distant stars, periodic reductions in light that indicate the presence of an orbiting planet, in the hopes of discovering Earth-sized planets. The mission lasted over nine and a half years, eventually ending in October 2018. During the duration of the mission, Kepler observed 530,506 stars and identified hundreds of thousands of transit events for thousands of planets. The entirety of the recorded data for the mission was released to the scientific community, resulting in the confirmation of many planets. Even more than 3 years after the end of the mission, there are still many data records labeled as "candidate" planets. These objects are neither confirmed exoplanets, or false positives, like binary stars. The goal of this project was to build different classifier models to use the existing confirmed and false positive dispositions of planets to predict if any of candidate planets are indeed true exoplanets.

Before diving into the data and methods, it is important to recognize that the existing confirmed planets were identified using vastly different methods than machine learning classification. The traditional way of identifying exoplanets involves modeling the period data and comparing this to other measurements taken of the specific observation. If a candidate planet fails any of these tests, it is automatically downgraded to a false positive, but if a candidate planet passes all of the complete tests, it can be upgraded to a confirmed planet. Therefore, the existing candidate planets in this dataset are observations that may have already passed some confidence tests, but have yet to fail a single test. This also has another important consequence, the only reason machine learning classification is possible is by relying on the results of the traditional classification method. If any of the tests used are actually faulty tests misclassifying records as confirmed or false positives, then the classification models will continue to perpetuate this bias. Perfecting the exoplanet classification process is a much larger task than the scope of this project, but it is important to acknowledge that the presented results could be entirely corrupted by this intermediary step.

II. METHODS

Data records for this project were built using one to six parameters measured or derived from the transit data, along with the existing classification. The classification was binary, with a false positive record being 0 and a confirmed planet being 1. Then each of the six potential attributes were continuous, and all normalized to fall within -1 to 1. The six attributes used were:

1) Orbital Period: The interval between recorded transit events. (Observed attribute)
2) Impact Parameter: The distance between the center of stellar disc and center of the planet disc at conjunction. (Observed attribute)
3) Transit Duration: The duration between first contact between the planet and star until last contact. (Derived attribute)
4) Transit Depth: The fraction of stellar flux lost at the minimum of the plantar transit. (Derived attribute)
5) Planetary Radius: The radius of the planet. Calculated as the product of the planet-star radius ratio and stellar radius. (Derived attribute, but good enough to be an observed attribute)
6) Transit Signal-to-Noise: Transit depth normalized by the mean uncertainty in the flux during the transits. (Derived attribute)

After consulting with a student at IU who is working towards their Master's Degree in Astrophysics, these six parameters were chosen to be used in the models. Attributes chosen are those that were directly observed or as good as directly observed, with the goal of making each attribute independent, which tends to yield better results for machine learning models. After setting up the datasets to be used for the project, the construction of classifiers could begin.

The methods used in this project ranged from simple predictors like Naive Bayes, to modern and advanced neural networks built using TensorFlow. The total list of different classification methods used is:

a. Naive Bayes model - implemented by hand
b. K nearest neighbor - implemented by hand
c. Decision Tree - implemented using the sklearn library
d. Random Forest - implemented using sklearn library
e. 2D and 3D Principal Component Analysis - implemented using sklearn library
f. Neural Network - implemented using sklearn and TensorFlow/Keras libraries

Before training and testing any models, the neural network is predicted to be the best performing classifier. This is a modern machine learning technique which thrives on large datasets. In order to determine which model is the best, five-fold cross validation will be performed to test each model. Additionally, the confusion matrix metrics will be computed for each trained model. The primary metrics used will be:

a. Accuracy: $\frac{TP + TN}{TP + TN + FP + FN}$
b. Precision: $\frac{TP}{TP + FP}$
c. Recall: $\frac{TP}{TP + FN}$

Lastly, the recall, also called "TP Rate" will be recorded along with the "FP Rate" so that the model can be plotted as a point along an ROC (Receiver Operating Characteristic) curve. An ROC plot is a graphical representation of a model's performance and quickly shows the trade-off between detection rate and false-alarm rate. Since the ratio between confirmed planets and false positives is about 1:2, using these metrics in addition to accuracy can provide a clearer picture of which model performs the best.

III. RESULTS

The different classification models used can fall into a few different categories. The first category are the simple and time consuming models, Naive Bayes and K Nearest Neighbor. These models were used as a baseline and starting point for developing and comparing the more advanced models. The next category of models are the tree based models. A decision tree was used so that performance in a single tree could be tested. In addition, random forest was used to build upon the performance of a single tree and hopefully give improved results. The third category of classifiers isn't a traditional classifier, but is a dimensionality technique. Principal component analysis was used to attempt to see if the training data was linearly separable in two or three dimensions, and to see which attributes provided data that was linearly separable. Looking at linear separability was important for the development of the final category of models: the neural network models. The implementation of a neural network using sklearn and TensorFlow was performed since the two packages have different strengths. Many different types of classifiers were used to show the difference between simple and far more advanced algorithms, as well as to see how different types of classifiers could perform with this exoplanet data.

The naive bayes and KNN classifiers were chosen because of the simplistic nature of their algorithms. Both of these models, but naive bayes in particular, have been outperformed by more advanced models in current machine learning applications. Since the dataset is relatively small, the classification is binary, and there are not many features, these two models could provide an introductory view into machine learning classification for the exoplanet data.

The chosen hyperparameters for the Naive Bayes model were 10 quantiles, and an additive smoothing value of 3. Additive smoothing is used to smooth categorical data so that the posterior probability in the probability calculation is not zero. With 10 quantiles, each continuous feature was mapped into 10 discrete features. To test the different combinations of data parameters, each combination containing one feature to six features was used. The model which performed the best scored a 75.8% accuracy, and used orbital period, planetary radius, and transit signal-to-noise as the attributes.

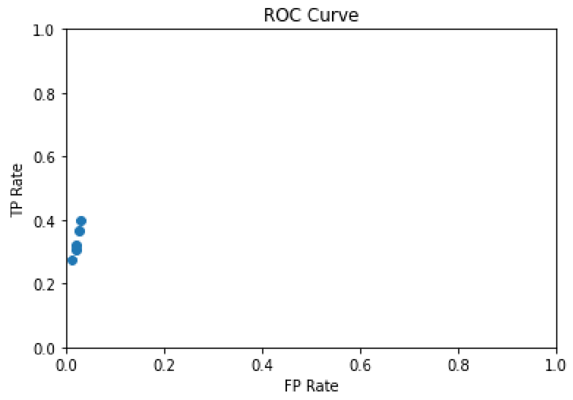

Figure 1: ROC Curve for Naive Bayes

The ROC curve for the Naive Bayes model (figure 1) shows that the models had consistently low FP rates, but also low TP rates. This can be attributed to the dataset as a whole having twice as many false positives as confirmed planets. The Bayesian probability predictions leaned towards classifying every instance as a false positive since the posterior probability was much higher.

While none of these models had exceptional performance, and nearly all other models performed at 67% accuracy, the baseline due to the class imbalance, this shows that higher accuracy is potentially possible with more advanced models.

The KNN classifier was tested on values of k from three to nine. To find the best performing KNN classifier, all combinations of the six different features were tested. The best performing model was a classifier with k=9 using the attributes orbital period, impact parameter, transit depth, planetary radius and transit signal-to-noise. This model had an accuracy of 86.2%.

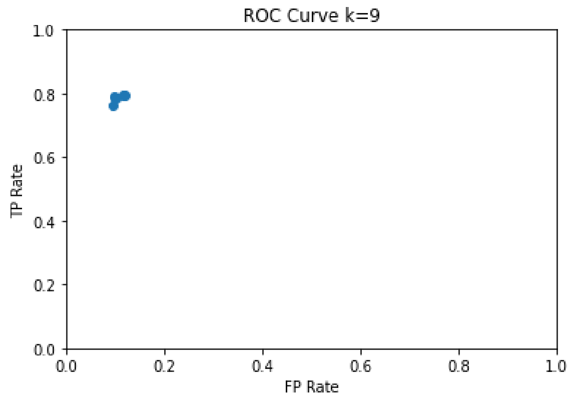

Figure 2: ROC Curve for KNN

The ROC curve (Figure 2) shows that the TP Rate and FP rate of the KNN classifier are far more balanced than the Naive Bayes ROC curve. This leads to greater accuracy and performance overall, but also means that the classifier performed better with the class imbalance too. While the accuracy of some of the KNN classifiers reached 85%, most of the accuracies were much lower, some being no better than a random guess with accuracies of 66% to 73%. This shows how important it is to choose the best features, even if that means training and testing many classifiers.

With the KNN classifier, the performance dramatically improved over the Naive Bayes classifier, but the time complexity of this model caused training to take much longer as well. Both of these models were significantly affected by the features used, but in testing combinations of features, the most important features presented themselves. It is also interesting that neither classifier performed the best when all six features were used, this suggests that a subset of the six features may be better at distinguishing between false positives and confirmed exoplanets.

The next category of models are the tree classifiers. These classifiers train much quicker than the previous two types, and they also produce significantly better performance results after 5-fold cross validation.

For the decision tree classifier, there are no hyperparameters to specify since no tree pruning is being performed. It was decided to not use tree pruning since this adds an extra layer of testing complexity, and the main purpose of the decision tree classifier is to have a model that trains and tests very quickly. After doing the same style of testing by using every combination of attributes, the best decision tree model scored an accuracy of 87.3% using all six possible attributes.

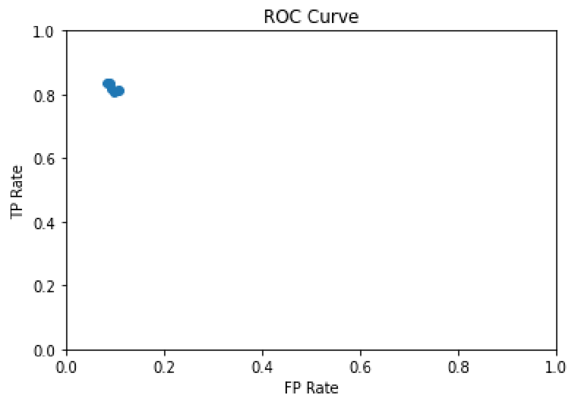

Figure 3: ROC Curve for Decision Tree

The ROC curve for the decision tree (Figure 3) shows TP Rate and FP rate are balanced, and the distance from the upper left corner to the five models is getting smaller, showing that these models are improving upon the previously tested classifier types.

The random forest classifier is meant to improve upon the performance of a single decision tree. By using many classifiers, the goal is to choose the classification with majority vote, and since the objective of the classification is binary, a majority vote has the potential to significantly improve accuracy. The two choices of hyperparameters for the random forest are the number of estimators used, and the max depth of each tree. Increasing the number of estimators increases the number of votes used for the majority vote classification, but it also can slow down the overall performance because many more trees need to be created. To balance this, the max depth of the tree can be specified which could prevent massive trees from forming. Since at a maximum, only six features are being used, none of the trees will be very deep, so increasing the number of estimators within the forest could have a significant effect without causing time complexity to increase very much. The random forest which performed the best was a model with 100 estimators, and using all six attributes. This model had an accuracy of 91.8%.

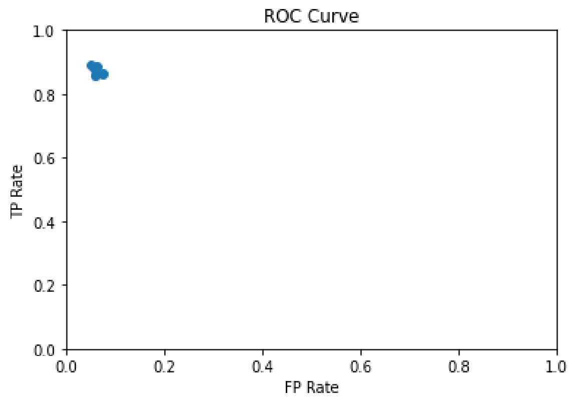

Figure 4: ROC Curve for Random Forest

The ROC curve for the Random forest model (Figure 4) shows that these models are performing well, but this is not a hugely significant improvement over the decision tree. Since at most, there can be six different splits in the tree for the deepest branch, this does not allow the random forest to greatly improve over the decision tree. The number of features used in the model appears to actually be the most important hyperparameter for both of these classifier types because as less features were used the performance of the models decreased dramatically. Unlike KNN and the Naive Bayes classifiers, the selection of attributes is less important than the number of attributes total.

The next category of classifiers does not give an accuracy for each classifier, but instead shows the separability between the two classes. Principal component analysis transforms the dataset into a vector space where the basis are the vectors that are most influential on the original data. By plotting the data in both 2 dimensions and 3 dimensions and distinguishing between the two classes in the plot, one can visually tell the difference between the classes. If the two classes have minimal overlap, this means they are separable and the features which gave this separation might perform well when training and testing neural networks.

So far, the attributes which have been part of the best performing models are Orbital Period, Planetary Radius, Transit Signal-to-Noise, and somewhat Transit Depth. Looking at the PCA plots for these attributes might help explain why they have performed better than Impact Parameter and Transit Duration.

The 2-dimensional PCA plot with the greatest visual separation in classes used the attributes orbital period, impact parameter, orbital depth, planetary radius, and transit signal-to-noise. The 2 dimensional PCA plot is shown in figure 5:

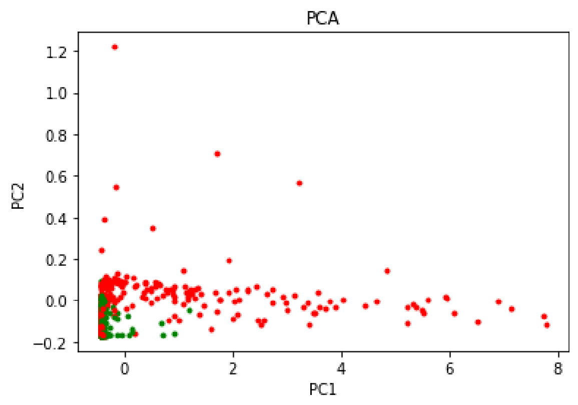

Figure 5: 2D PCA Plot

The 3-dimensional PCA plot for these same attributes is shown in Figure 6.

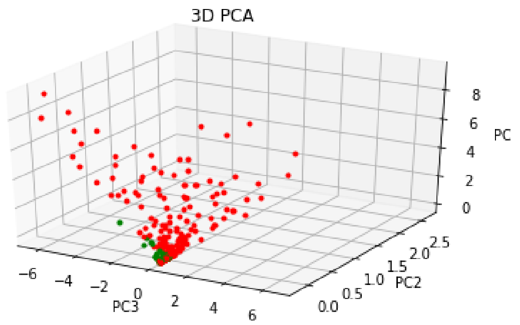

Figure 6: 3D PCA Plot

Looking at the plot in 3D shows that most of the points fall on the same plane, but the plane is not necessarily parallel with any of the axes. Since the third principal component vector adds such a small addition to the overall basis after the first and second components, it makes sense that none of the plots appear to have greater separation than they originally did in two dimensions. The three attributes that were predicted to be the most significant appear in the attributes used for these plots, this confirms that they are important. On their own they don't have high separability so it seems that they can provide context for the other attributes instead of being important attributes themselves.

The final category of classifiers are the neural networks. Neural networks are currently the most popular machine learning models since the results seem to be magic. The two different neural network implementations used are from the TensorFlow library, and the sklearn library. TensorFlow works well on extremely large datasets since it uses batch processing during training. While 6000 planetary samples seems large, in terms of data processing this isn't enough to fully reap all the benefits of the library. Due to this, the training takes longer than the sklearn library due to all of the initialization steps. The sklearn library takes all the samples at once and then performs the backtracking algorithm. This would be slower for working with millions of data samples, but for the exoplanet data it was much quicker than TensorFlow. The other advantage TensorFlow provides is different layer types which come in useful for different domains. The focus of the neural network step will be to use a densely connected sequential neural network, but using dropout within the TensorFlow model might be able to reduce overfitting. Both implementations seem to give roughly the same accuracies when using the same network architecture, so all the experimentation will take place using sklearn, and TensorFlow will be used to try to improve a network that is overfitting. Lastly, since there are too many combinations of attributes to test every single one in the neural network step, the attributes used will be the five attributes from the principal component analysis, and all six attributes.

The first neural network type tested was the sklearn version. Sklearn provides fewer hyperparameter options for the user which allows for fewer total combinations. There are still too many different network architectures to test exhaustivity, but general ideas can be tried. A few of the different networks tested were: networks with a single hidden layer, networks with many hidden layers of the same size, and networks with many hidden layers of different sizes. As more layers are added to a network's structure, the complexity goes up and overfitting becomes a problem. THerefore, the network architecture that performed the best was a single hidden layer containing 7500 neurons. These hyperparameter choices balanced complexity and ability to perform equally well on training data and validation data. The accuracy for the neural network using sklearn after training for 300 epochs was 90.6%.

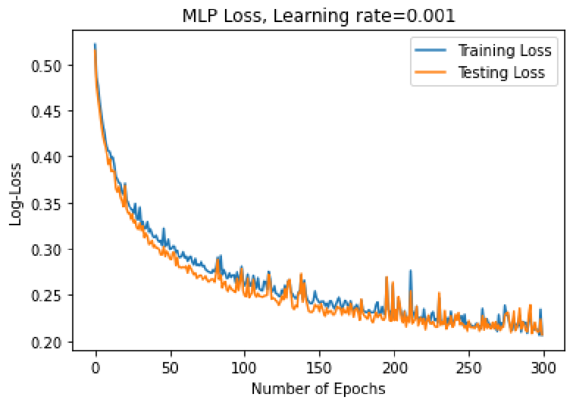

Figure 7: Sklearn Neural Network Learning Curve

The learning curve (Figure 7) shows that there was no significant overfitting since testing loss match training loss through the entire training.

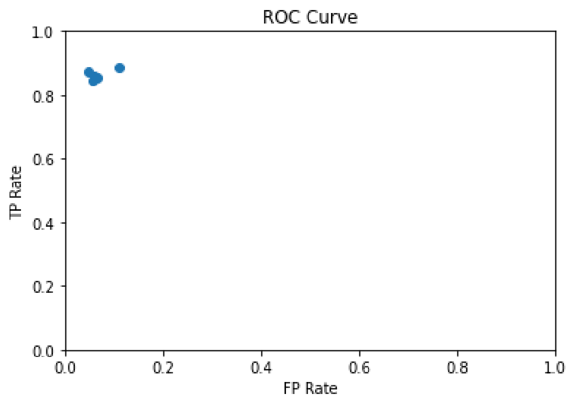

Figure 8: Sklearn Neural Network ROC Curve

The ROC curve (Figure 8) also shows that the model performance is significantly higher than random guess, and the TP rate is balanced with the FP rate. Due to the class imbalance, it is important that the precision and recall are also investigated, and the ROC curve shows that the model still performs well.

The TensorFlow neural network provides more customization for the user, but this also creates more options and more complexity when designing a neural network. Fortunately, most of the customization options are not intended to be used in this type of domain, so the options are narrowed down. Again, there are too many total options available, but the one that provided the most enhancement to the architecture was the "dropout" layer. Unlike a traditional hidden layer, the dropout layer randomly removes connections between hidden layers which, in theory, should reduce overall complexity and likewise reduce overfitting. The dropout layer was applied to the same model types tested for sklearn, and the model which performed the best was a network with three hidden layers of 50 neurons, and then between the first and second layers was 25% dropout, and between the second and third layers was 10% dropout. The accuracy achieved by this network after training for 500 epochs was 89.2%.

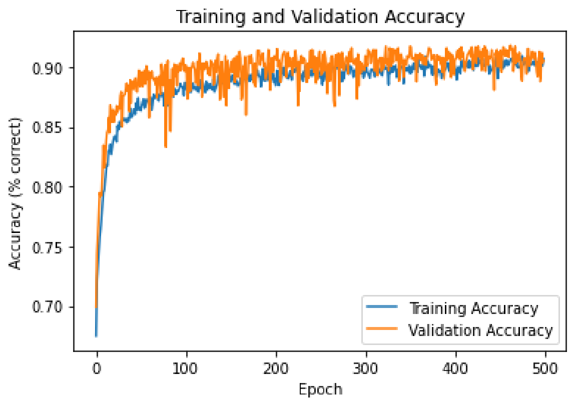

Figure 9: TensorFlow Neural Network Accuracy Rate

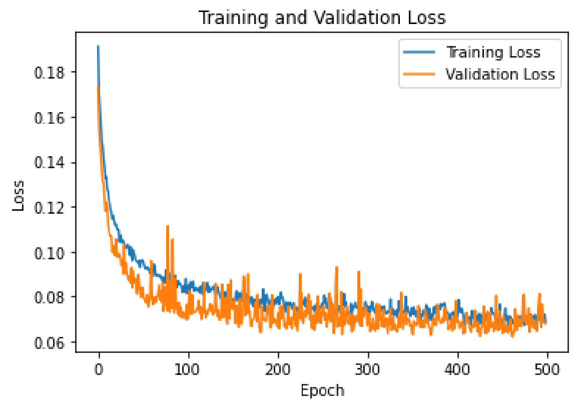

Figure 10: TensorFlow Neural Network Learning Curve

Like with the sklearn training, the primary goal was to avoid overfitting when training the TensorFlow models. Both of the training curves (Figure 10) show that no overfitting occurred. The training and validation accuracies (Figure 9) remained close together just as the training and validation losses.

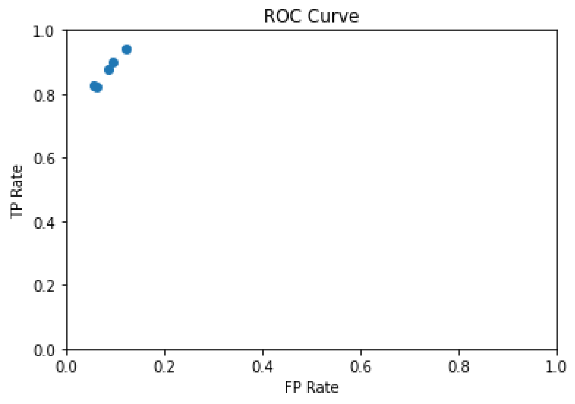

Figure 11: TensorFlow Neural Network ROC Curve

The ROC curve (Figure 11) also shows that the models perform much better than a random guess model.

The performance of all the models is shown in the table 1:

Model	Accuracy	Precision	Recall
Naive Bayes	75.8%	0.888	0.333
KNN	86.2%	0.802	0.795
Decision Tree	87.3%	0.819	0.811
Random Forest	91.8%	0.878	0.880
Sklearn Neural Network	90.6%	0.865	0.859
TensorFlow Neural Network	89.2%	0.862	0.822

Table 1: Summary of model performance

IV. DISCUSSION

The performance of the models used was largely what was expected. The naive bayes model performed worse than everything else, and the more complex models performed much better. The most surprising performance result was the success of the random forest model. This model performed the best, even better than the neural network models. One possible explanation for this result is that there was just simply not enough data for the neural networks to train on. Neural networks and especially deep neural networks need exponentially increasing sizes of data to train on ever expanding model architecture. In the future, after more transit surveys have been completed and more data is gathered, it is entirely possible that the neural network approach beats the random forest model.

The other surprising performance result was the precision metric of the naive bayes classifier. Precision is a measure of the true positives divided by the sum of the true positives and false positives. Since naive bayes had the lowest accuracy score, it might be reasonable to assume that the other metrics were also the lowest but naive bayes had the highest precision score. When dealing with class imbalance, looking at precision and recall is necessary, and the recall score for naive bayes was by far the lowest, which accounts for the low accuracy, but it is still surprising that the precision value was so high.

The final step in this project was to actually apply the classification models to the candidate planets which have not been classified yet. This is only possible using the neural networks models since these models can give a regression value as well as a binary classification. Since labeling all nearly 3,000 candidate objects wouldn't be very helpful, the approach that was taken was to instead predict a likelihood that any planet was a true exoplanet. This way, the machine learning models could be used to narrow down the possibilities for scientists to look into more closely, and reduce the total number of planets considered.

Of the nearly 3,000 candidate planets, the sklearn model gave a certainty value above 95% for 85 planets, a value above 99% for 20 planets, and a value above 99.9% for 4 planets. Similarly, the TensorFlow model gave a certainty value above 95% for 97 planets, above 99% for 14 planets and above 99.9% for 8 planets. This is an overall reduction of nearly three orders of magnitude for possible planets to consider. As more data is generated, it is going to be more and more critical to narrow down the search space so scientists can focus their energy on planets that are more promising.

Looking towards the future, as more transit surveys are completed and more accurate tests are developed to confirm exoplanets, a classification method might become more useful. The world is also heading in a direction that shows that quantity of data is extremely important. NASA missions will continue to produce this planetary data, and it will be important to continue to apply the most advanced methods of statistics and machine learning to analyze more data faster. This could be important to identifying target planets for probe missions, or any missions which would bring humans, or human created technology, to alien worlds.

Both partners, Owen Gordon and Jonah Slack contributed equally to this project. The coding, paper, and presentation work was split between the two.

Code for this project can be found on GitHub at: https://github.com/OwenGordon/B365Project/blob/main/B365ExoplanetAnaysis.ipynb

Works Cited

Goyal, Armaan. Interview. By Owen Gordon, Jonah Slack. 26 October 2021

“Data Columns in Kepler Objects of Interest Table.” *NASA Exoplanet Archive*, https://exoplanetarchive.ipac.caltech.edu/docs/API_kepcandidate_columns.html