ASTRO 410 FINAL RESEARCH PAPER
IDENTIFYING EXO-PLANETS USING MACHINE LEARNING

Swapnil Dubey

ABSTRACT

The goal of this project was to replicate the results that were achieved in identifying possible exoplanets. As the machine learning techniques are becoming easier to implement and use, it is becoming easier to test and implement prediction models to achieve tasks that would take hours or days for a team of humans to achieve. There are three models used for this project, Support Vector Machine (SVM), a Random Forest Classifier and an Artificial Neural Network with 3 layers using Keras in python. It is possible to achieve a very high accuracy for prediction using any of these models with a relatively small dataset of only 5087 observations. While this paper shows a successful proof of concept, future endeavors hope to work with a much larger dataset and hope to add to the catalogue of exoplanets that have been discovered by Kepler Space Telescope.

INTRODUCTION

The first exo-planet was discovered on 9 January 1992. It has only been roughly 30 years since it was first confirmed that stars can and do host planets in their orbits. This opened a new frontier of exploration and research, to hunt and study planetary bodies outside our own solar system. While the search began slow, with only Earth based telescopes, it picked up pace as new equipment and techniques were developed to detect these planets. One of the biggest contributors to the exoplanet catalogue, at the time of writing, has been the Kepler space telescope. It was launched on March 7, 2009 and since then has photometrically observed more than 200,000 stars. This has led to the discovery of thousands of transiting exoplanets. The sudden rise in the abundance of observation data has led to researchers trying new and innovative ways to go through the data in search of exoplanets.

In this paper, we hope to replicate the results of building a Machine Learning model to classify transiting exoplanet data as gathered by the Kepler Space Telescope. Due to the sheer volume of data publicly available from Kepler and as the onpour of data continues with TESS (Transiting Exoplanet Survey Satellite), it has become necessary to use computer models to do relatively menial and repetitive tasks such as identifying exoplanets from raw data. The output of the task is a binary result, either a star has an exoplanet or not, which makes it easier for our machine learning model. By using computer models to classify datasets that span over years, researchers can devote more time in learning about those exoplanets instead of simply finding them.

A possible problem for such a methodology would be getting false positives, or rather, false negatives. This is because it would be much better to have falsely identified a star having an exoplanet, than missing a star with an exoplanet. Unfortunately this project has not solved that problem yet but there is hope that by training on a much bigger dataset, it would become possible to overcome this effect.

METHODS & MODELS

While there may be many ways of implementing machine learning for the Kepler data, for this paper we are strictly focusing on transiting exoplanets. We use datasets from Kaggle, where the shape of our training dataset is 5087 rows by 3198 columns.

```
[4]  M↓
test_data = pd.read_csv('data\exoTest.csv').fillna(0)
train_data = pd.read_csv('data\exoTrain.csv').fillna(0)
print(train_data.shape)
train_data.head()
```

```
(5087, 3198)
```

	LABEL	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6
0	2	93.85	83.81	20.10	-26.98	-39.56	-124.71
1	2	-38.88	-33.83	-58.54	-40.09	-79.31	-72.81
2	2	532.64	535.92	513.73	496.92	456.45	466.00
3	2	326.52	347.39	302.35	298.13	317.74	312.70
4	2	-1107.21	-1112.59	-1118.95	-1095.10	-1057.55	-1034.48

5 rows × 3198 columns

Figure 1. Previewing training dataset

The first column “LABEL” indicates whether the star has any exoplanets or none at all. The rest of the 3197 columns are the flux values at regular time intervals. The objective is to teach the computer to find patterns for stars with exoplanets and use those patterns to classify new or unknown stars for exoplanets. The columns are therefore referred to as “features” for our Machine Learning model.

Unfortunately the most difficult aspect of working with Machine Learning models is not the model itself, but the many steps that are required to preprocess the data. Below is a summary of the steps that were used to prepare the datasets to be used for a training model using Keras with Tensorflow.

Data Pre-Processing

To enable efficient use of computer resources, we implement a memory optimizations function. This is achieved by changing the data types of our integer and decimal values. For this data set we know that we will not be requiring flux values precision of more than two decimal places, so we can save on a lot of memory by changing the data type of the cells in our DataFrame.

```
[6]  M↓
re_test_data = reduce_memory(test_data)
Memory usage of dataframe is 13.91 MB
Memory usage after optimization is: 6.25 MB
Decreased by 55.1%
```

Figure 2. Data set memory optimisation results

By making such a change early in our project we can save on processing time throughout our calculations with different models and training. Also, the original data set indicates a star with an exoplanet using 2 and no exoplanet using 1. Although those are two states that can also work, it is my personal opinion that it would be better to change them to 0 and 1 where 1 indicates an exoplanet and 0 does not.

Further exploring our data set, it is found that there is a massive imbalance in the amount of stars with and without exoplanets.

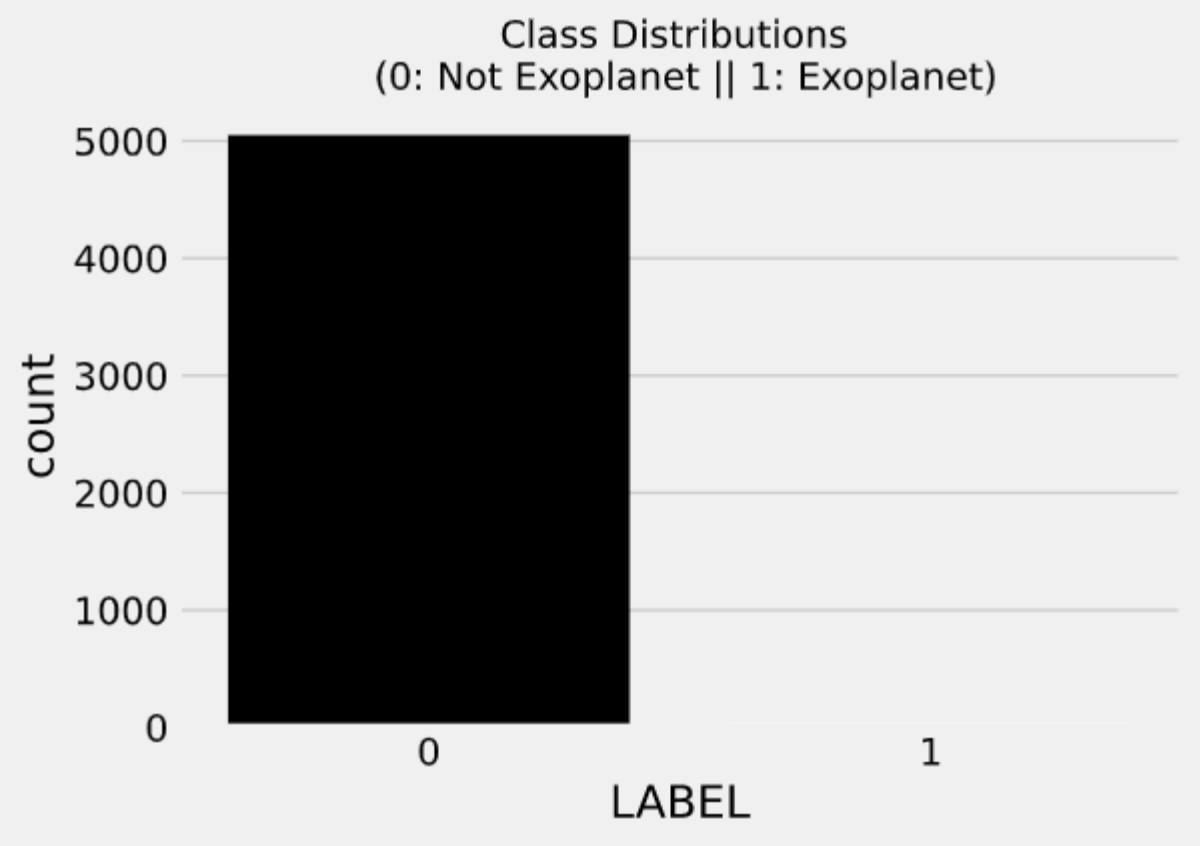

Figure 3. Data distribution, 0: No Exoplanet || 1: with Exoplanet

The actual ratio between stars labeled “0” to stars labeled “1” is:

5050 : 37

Something like this is very bad for a model to learn from. Machine Learning models are based on probabilistic factors. The idea is that the model gets fed huge amounts of data, and

for each input the model will use certain weights for the features of that input and generate an output. The final test of our dataset will be using a neural network and the accuracy it is able to achieve with a relatively small dataset as this. Below is an example of a simple neural network showing weights in an algorithm as neurons, that are connected and affected by all neurons from the layers before and after them. This forms a regression model through which the weights of each neuron are systematically changed, until the algorithm learns to make the correct predictions.

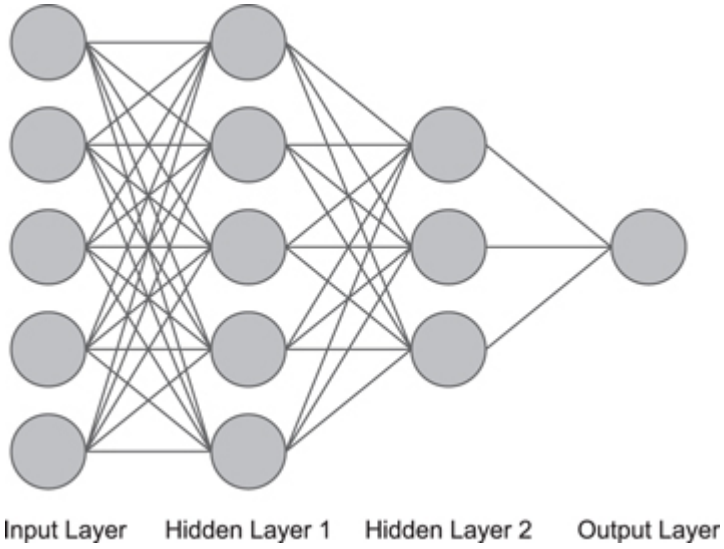

Figure 4. A simple Neural Network illustration

Smote

While there can be more complicated models with a spectrum as it's output, it is convenient for this project to get only a binary output as mentioned above. Getting a correct prediction reinforces that the weights and biases it has developed are either correct or close to correct. By having imbalanced data fed to such a model would reinforce the biases of the data onto its classification model. For our particular dataset, where more than 99% data has the label "0" (no exoplanet), the model will be reinforced to predict a 0 no matter what the input is. We can solve this by using the SMOTE (Synthetic Minority Over-sampling TEchnique) resampling method in python to get a ratio of 1:1 of 0 with 1 for our training data set.

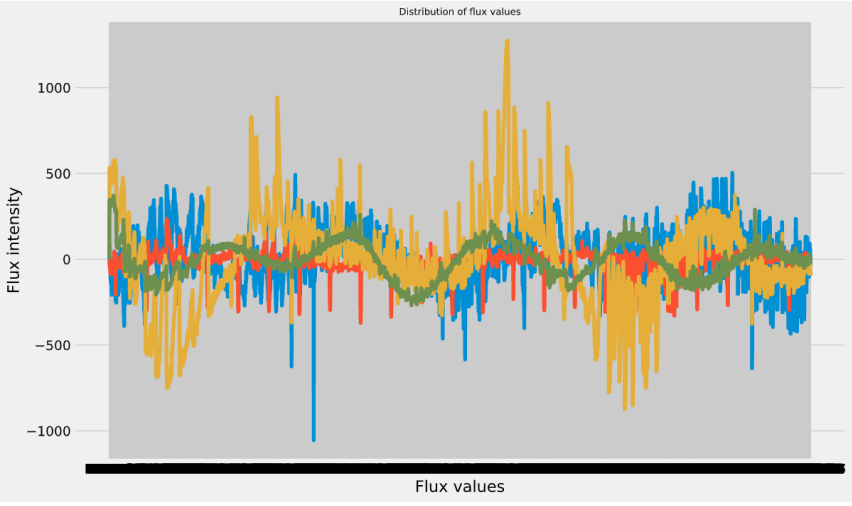

Figure 5. Flux intensity plots for the first 4 rows

Normalisation

The above plot shows that the flux intensity of the stars are not normalized. Normalizing our dataset allows for a more accurate comparison of data from different stars. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values.

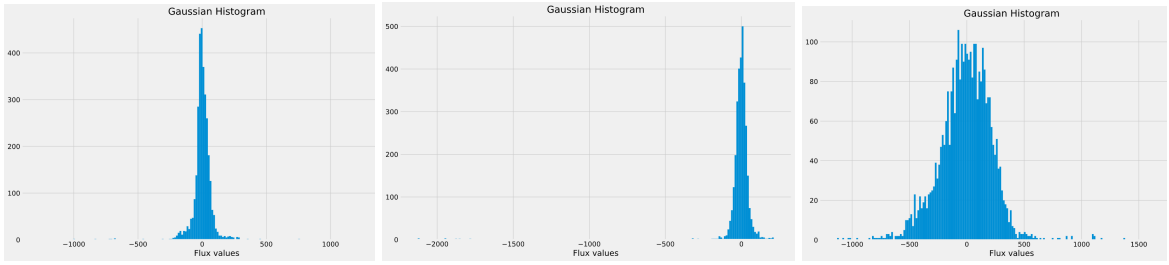

Figure 6. Flux distribution of 3 stars without exoplanets

The spread of the flux values in the plot above is small. Comparing this with stars known to have exoplanets, one could see a very apparent difference in their distribution.

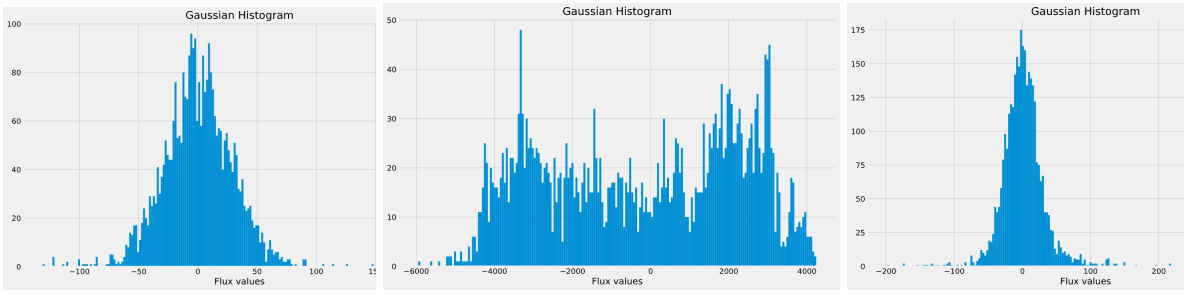

Figure 7. Flux distribution of 3 stars with exoplanets

PCA

Having too many features can also be a negative. Each feature is an input variable that the model has to compute and analyse to make accurate predictions. Unfortunately, to accurately calculate the weights for each dimension, the required data would need to be increased exponentially. For most cases, having too many features for the model to track would get diminishing returns without the data to back it up. For this data set, 3197 features are too many, therefore we perform a Principal Component Analysis using the sklearn package in Python. This reduces the dataset to contain the most relevant features in our dataset. Their relevance is calculated by checking the specific feature's value variance throughout our data. If the values remain the same throughout the dataset with none or negligible difference between stars with exoplanets and without, those features will be discarded. Our dataset is able to retain up to 98.8% variance of our dataset using only 37 features.

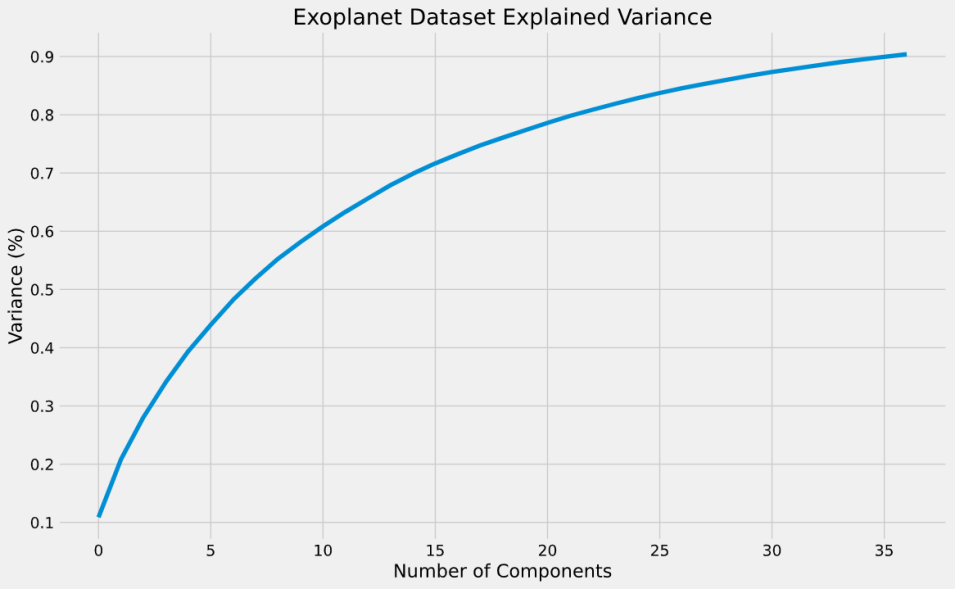

Figure 8. Variance retention after using PCA on dataset

It is now possible to use the processed datasets as inputs for the Machine Learning models that we will experiment with to get the best results for predictions.

RESULTS

There are many different machine learning models that could be used for the exoplanet dataset. For this project we tested three models, the Support Vector Machine (SVM) algorithm, the Random Forest Classification algorithm and implementing a neural network using Keras with Tensorflow.

SVM

Support Vector Machine (SVM) is an algorithm that one could easily find in the sklearn module in python. Below is an image showing SVM learning to differentiate between 2 classes. This algorithm is known as a “Support” Vector because as can be seen in the visualisation, the boundary “vectors” (data points) guide the point of segregation between the two classes. Together with the gap, it could be visualised as the vectors supporting or upholding that column and hence the name Support Vector Machine.

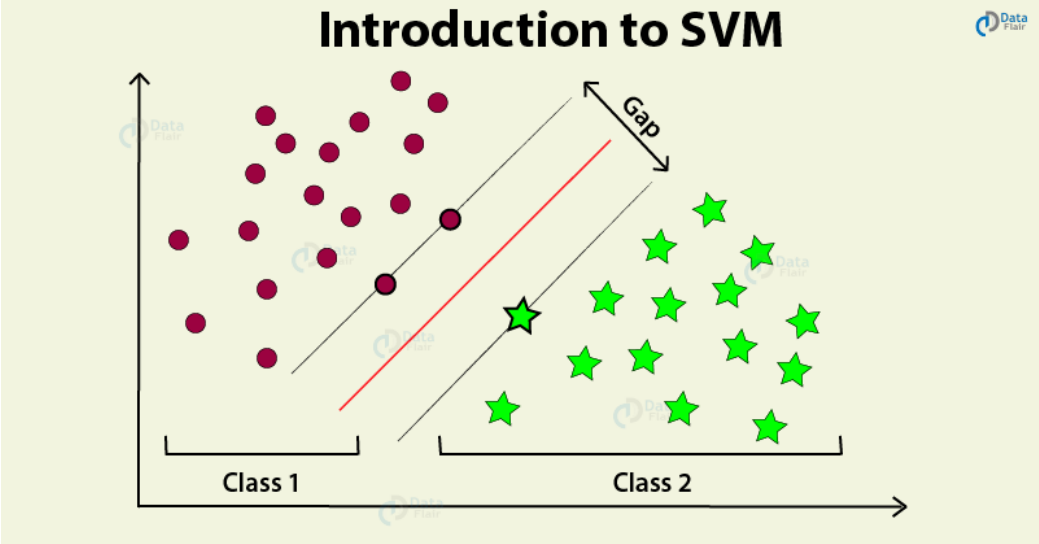

Figure 9. Example 2D illustration of a Support Vector Machine algorithm

Unfortunately, the above example only shows classification over 2 dimensions and from the data pre-processing above, we know this dataset has 37 dimensions over which our model will be learning on. Importing and running the SVM model, we get the following results:

```
Accuracy mean: 0.9998019801980199
Accuracy variance: 0.00039603960396039603964

accuracy_score : 0.9912280701754386

classification report :
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       565
           1       0.00      0.00      0.00         5

    accuracy                           0.99       570
   macro avg       0.50      0.50      0.50       570
weighted avg       0.98      0.99      0.99       570
```

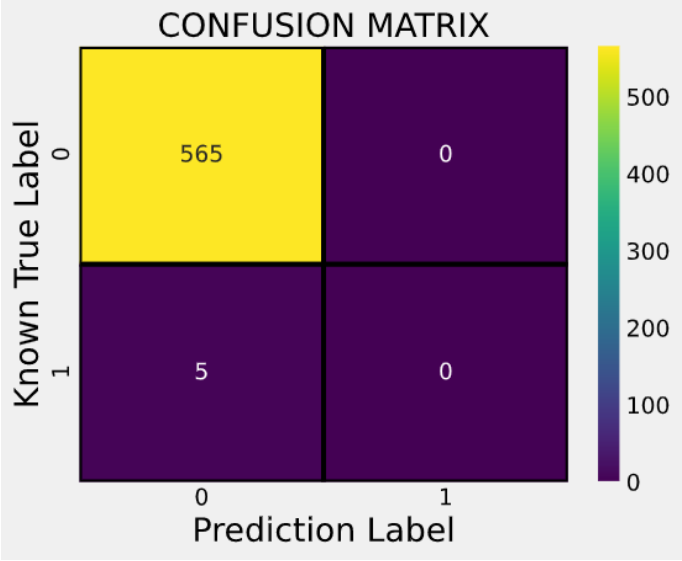

Figure 10. Prediction results with a test set of 570 stars using SVM

As can be seen from the results above, SVM is able to get an accuracy of more than 99.1%. It was successfully able to predict the 565 out of 570 stars. This is an amazing result considering that once setup, it only took the model 2-5 minutes to get these predictions which is significantly faster than humans.
Considering the bulk of the work is needed to preprocess the data which we have already done, it is also easy to implement a Random Forest Classifier. This is also a module from scikit learn in python. Importing it and running a Random Forest Classifier on the data we achieve the following results:

```
from sklearn.ensemble import RandomForestClassifier
rf_classifier = RandomForestClassifier()
model(rf_classifier,x_train_res,y_train_res,x_test,y_test)
```

```
Accuracy mean: 0.9995049504950495
Accuracy variance: 0.0005422995618863035

accuracy_score : 0.9912280701754386

classification report :
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       565
           1       0.00      0.00      0.00         5

    accuracy                           0.99       570
   macro avg       0.50      0.50      0.50       570
weighted avg       0.98      0.99      0.99       570
```

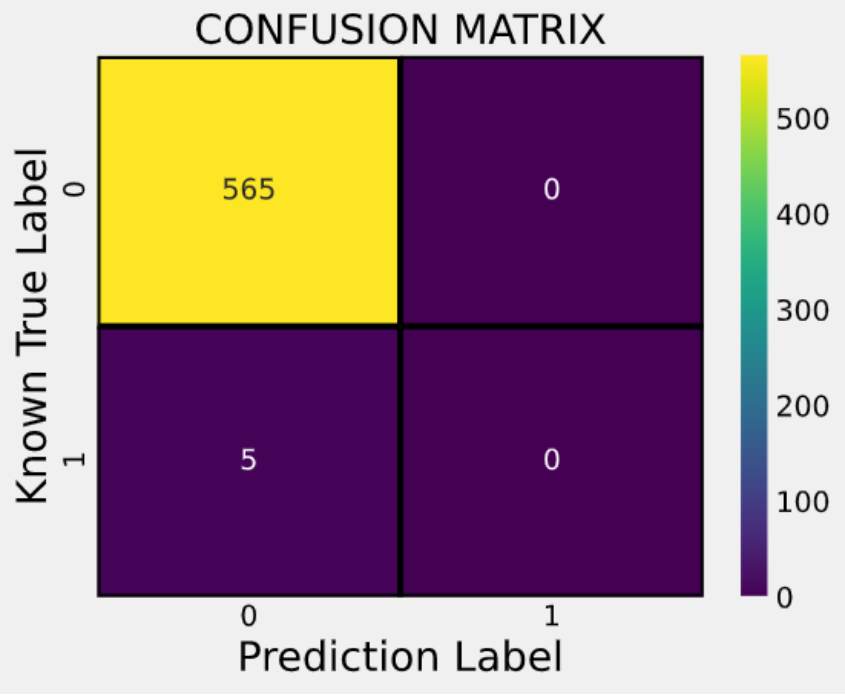

Figure 11. Predictions for 570 stars using Random Forest Classification

These results are very similar to our SVM algorithm. Considering that the accuracy is of more than 99% would mean saving researchers time from sifting through thousands of observation charts considering that Kepler recorded data for more than 200,000 stars. To better understand how the machine learning model has built the weights for the 37 features, we can plot their usefulness as a barplot.

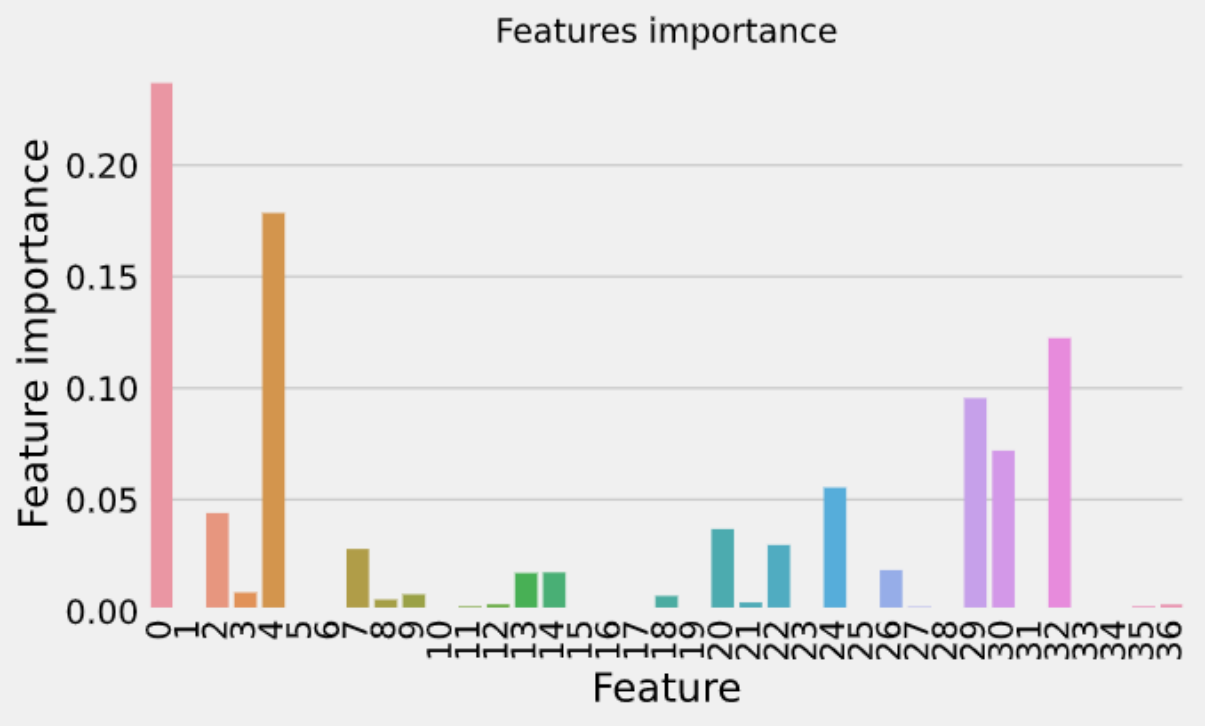

Figure 12. Feature importance for the Random Forest Classification algorithm

The last model we use to fit data is using an Artificial Neural Network (ANN). The neural network is implemented using Keras with a “ReLu” function for the first and second layer and a sigmoid function as the activation function for the third layer. The following is the result that the neural network was able to achieve with only 80 epochs:

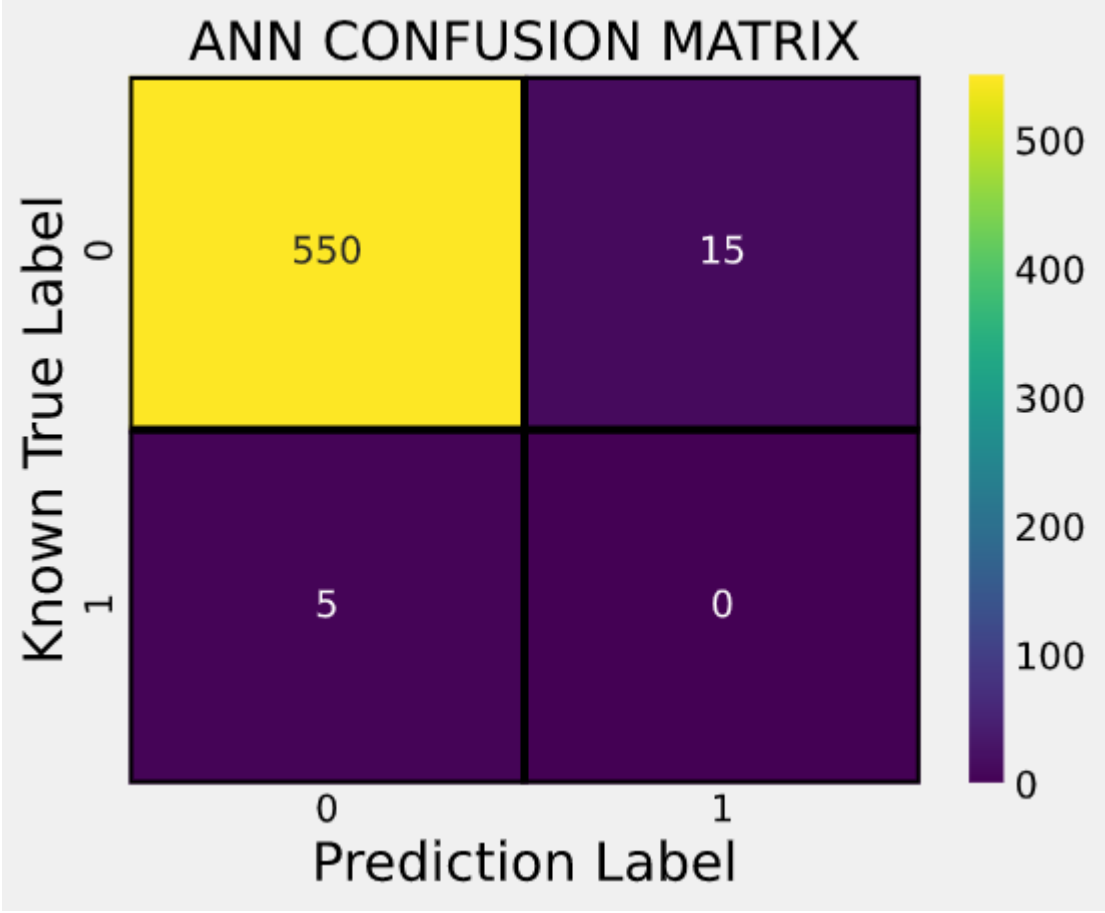

Figure 13. Confusion matrix showing the results of ANN predictions

For the neural network we achieved a mean accuracy of 85% and accuracy variance of 19%. This is still a good result considering that it can still save a lot of time for basic classification.

CONCLUSIONS

While the results of this experiment seem promising with accuracy of over 99% with some algorithms, there still seems to be a bias in our machine learning models. One could try working with more data by downloading it from the Mikulski Archive for Space Telescopes. I have already downloaded upto 30GB of data to further improve the models by adding more layers to the neural network. As new data is still being generated from missions like TESS there is always more data that can be used to benefit our models. One such study called the AstroNet project has already showcased how important something like this can be by discovering two exoplanets using neural networks.

Once a Machine Learning or Neural Network model does a first pass over the data and shortlists the best candidates, researchers can then work through that small subset of data which would increase the rate of discovery of exoplanets. This would be thousands of hours saved, and the increased rate of exoplanet discovery would propel the field of exoplanet study for astronomers.

REFERENCES

- Chauhan, Nagesh Singh. “Exoplanet Hunting Using Machine Learning.” *KDnuggets*, www.kdnuggets.com/2020/01/exoplanet-hunting-machine-learning.html.
- Christopher J. Shallue and Andrew Vanderburg 2018 AJ 155 94
- Shallue, C. J., & Vanderburg, A. (2018). Identifying Exoplanets with Deep Learning: A Five-planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90. The Astronomical Journal, 155(2), 94.