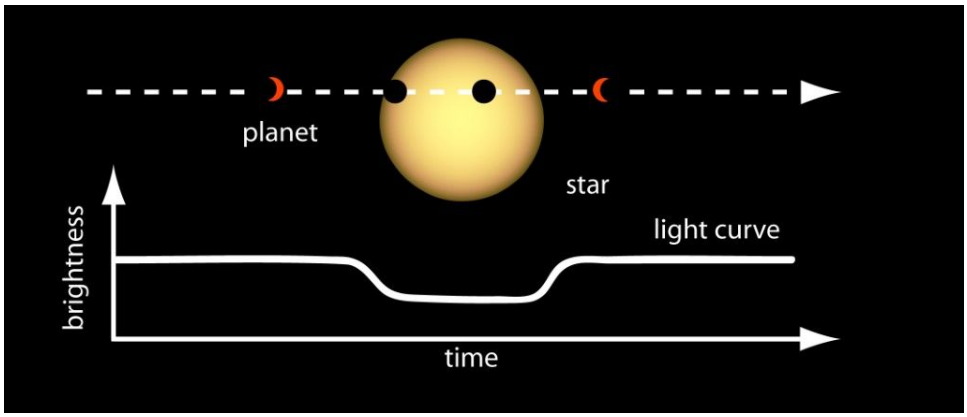
Exoplanet detection

Data at a glance:

The given data describes the change in flux (light intensity) of several thousand stars. A binary label of 2 or 1 is assigned to each star. Here 2 indicates that the star is confirmed to have at least one exoplanet in orbit.
As you can imagine, planets themselves do not emit light, but the stars that they orbit do. If the said star is watched over several months or years, there may be a regular 'dimming' of the flux (the light intensity). This is evidence that there may be an orbiting body around the star; such a star could be considered to be a 'candidate' system. Further study of our candidate system, could solidify the belief that the candidate can, in fact, be 'confirmed'.



Description:

- 3961 rows or observations.
- 3198 columns or features.
- Column 1 is the label vector. Columns 2 to 3198 are the flux values over time.
- 33 confirmed exoplanet-stars and 3928 non-exoplanet-stars.

Data Preprocessing:

Standardization – Each row was scaled to zero mean and unit variance for further analysis. Row wise scaling was done because each row is a time series of a planet/ exo-planet.

Exploratory Data Analysis:

Firstly, graphs were plotted for some row between the flux values and time (as columns denote passage of time). This was done to get first hand idea of the data distribution.

Why Filters?

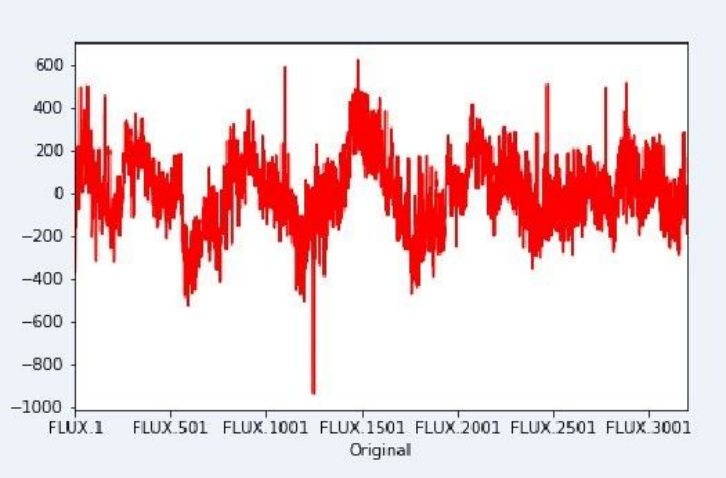
As mentioned in some of the research papers and it was observed that the data had high frequency noise embedded into it. A common problem in reconstructing and cleaning data is elimination of noise. Noise can corrupt a signal through many means: quantization, measurement noise, errors in sampling time, sensor bias, sensor nonlinearities, signal cross coupling, etc.
There are many methods which can be used to eliminate the noise on a signal. Some common approaches include use of a **Median filter, Gaussian filter**, **Kalman filtering**, **Wiener filtering**, construction of a custom optimization problem, and any number of ad-hoc approaches. So, for further analysis the following filters were applied on the data to remove noises and smoothen it out.

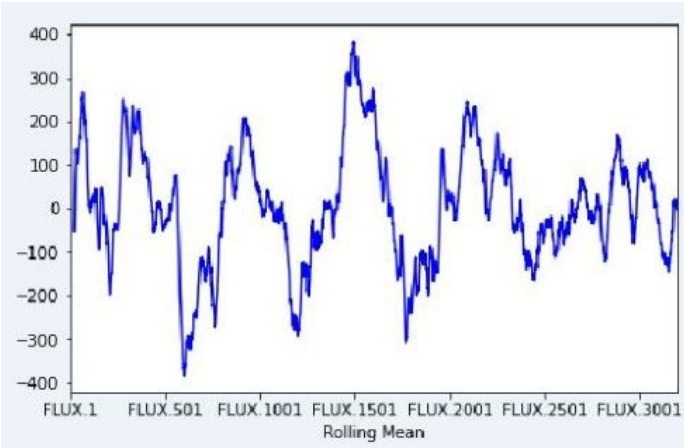
1. Rolling mean/ Uniform filter: window size 100
2. Median Filter: windows size 100
3. Gaussian Filter: sigma 1
4. Fourier Transformation

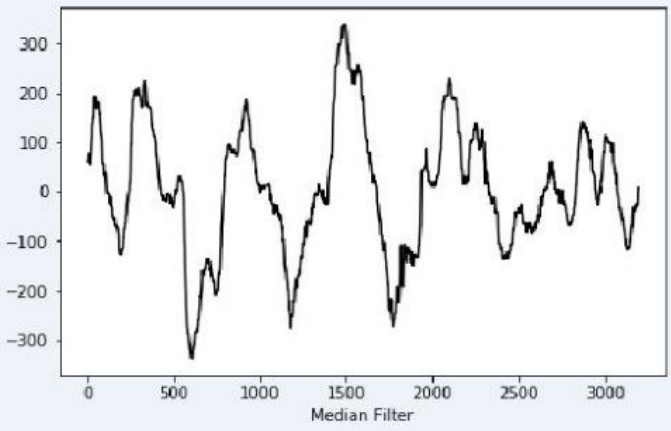
Fourier Filter:

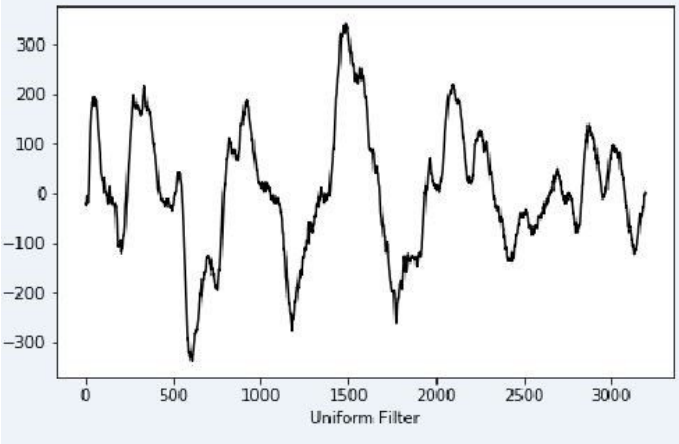
One way to quickly filter a dataset without much effort is to use a **Fourier transform**. A Fourier transform is a way to decompose a signal into a sum of sine waves. The amplitude and phase associated with each sine wave is known as the spectrum of a signal. If the spectrum of the noise if away from the spectrum of the original signal, then original signal can be filtered by taking a Fourier transform, filtering the Fourier transform, then using the inverse Fourier transform to reconstruct the signal.

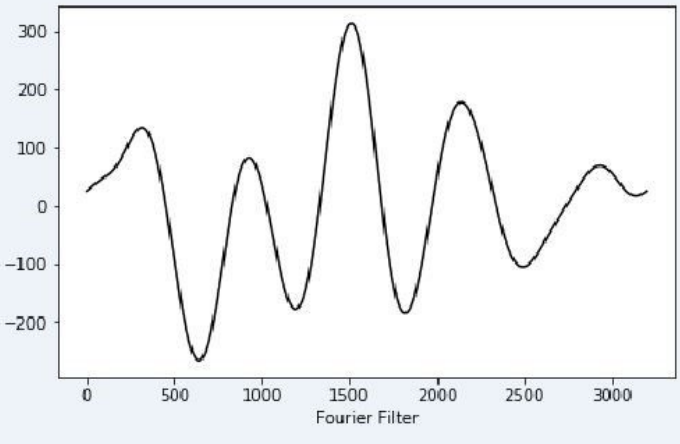
So we Manually created a Fourier Filter by transforming the data to frequency domain and then make up a narrow bandpass with a Gaussian to filter the spectrum to remove noise from data and it is shown in 5th figure.



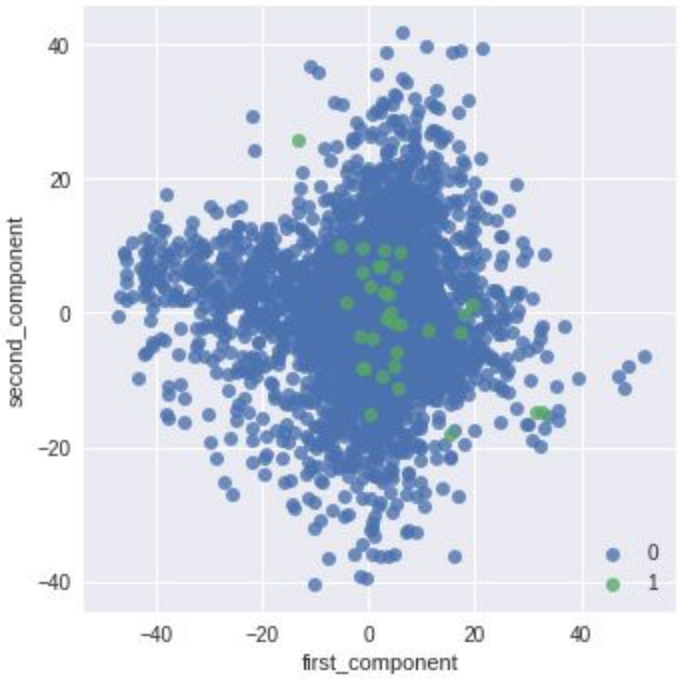








Then, as there are a lot of columns (3198) we used **PCA** on the data for dimensionality reduction, and plotted the first 2 and 3 components to visualize the data. The variance retained by the first 2 components was very low around ~0.08 percent.



We also used the Manifold learning methods and PCA on the transpose of data (because each row is a time series) but results were approximately same.

Machine Learning Models:

Data Split:

Data for training: 90%

Data for validation/testing: 10%

As the data is skewed, we used oversampling technique to balance the data (we tried replicating the smaller class data 20-30 times) and then applied the model on it.

XGB MODEL: we used xgboost algorithm to classify the imbalanced class data by different hyperparameter tuning but the results were very poor. We also tried to get important features from it but it gave us only 86 features with > 0 importance and all the other features were with feature importance = 0.

We used K-mean clustering to cluster data of similar characteristics. But that didn't work.
We tried some different algorithms also and the results are in below table.

Method	F1 score (macro)	F1 score (micro)/Accuracy
Xgboost + original data	0.709	0.979
Xgboost + gaussian filter data	0.649	0.964
Xgboost + uniform filter data	0.483	0.934
Xgboost + Fourier filter data	0.491	0.961
Logistic Regression (all type data)	0.498	0.992
SVM (RBF Kernel)	0.498	0.992

The reason for the failure of the model was that it took the columns to be features of the problem rather than a sequential pattern and hence unable to separate the pattern of exoplanet from the non-exoplanets.

Deep Learning Models:

Since Normal Machine learning methods and oversampling did not work, to increase the data size we tried **data augmentation technique** i.e we took some rows of the data and transformed them in time. For example, in a particular row, we took the first 100 fluxes and moved them to the last 100 places and adjusted the rest of data accordingly(We rolled the time series to generate a new one). This is an excellent data generation technique for the problem. Since the data is periodic so the newly generated data would be a valid data point i.e a robust classifier would identify this new data to be in the same class as the one it is generated This is different from conventional oversampling techniques as it generates new data on the fly.

In deep learning these are some popular methods for these type of time series problems:

1. LSTM (Long Short Term Memory)
2. 1-D CNN
3. 1-D CNN (1-dimensional convolution neural network) + LSTM

LSTM:

LSTM are an attractive choice for sequence labeling because they are flexible in their use of context information, due to the fact that they can learn what to store and what to forget. They can recognize sequential complex patterns in the presence of sequential distortions.
We trained a simple LSTM on the training data, and tried predicting the results on the test data but the model is classifying every data point as a non-exoplanet. This might be due to small amount of data available as LSTM needs large data to correctly identify patterns.

1-D CNN:

Convolution is an important operation in signal and image processing. The following can be considered *strengths* of a convolutional neural network (CNN), compared to fully-connected models, when dealing with certain types of data:

- The use of shared weights for each location that the convolution processes significantly reduces the number of parameters that need to be learned, compared to the same data processed through fully-connected network.
- Shared weights are a form of regularization.
- The structure of a convolutional model makes strong assumptions about local relationships in the data, which when true make it a good fit to the problem.
- Local patterns provide good predictive data (and/or can be usefully combined into more complex predictive patterns in higher layers)
- The types of pattern found in the data can be found in multiple places. Finding the same pattern in a different set of data points is meaningful.
- 1-D CNNs are at least 40-50 percent faster than normal LSTMs because of simpler computational structure.

We used this method for our Final model.

1-D CNN + LSTM:

We also tried to combine CNN and LSTM together by first making some CNN layers and the connecting it to LSTM layers but the results was not as good as Fully CNN.

Our Model:

Because LSTM was not giving us good results. So we searched a lot in some research papers and found that convolution neural network can work pretty good on these type of problems. So we tried different architectures on 1-D convolution. We used Batch Normalisation layer and Dropout layers to make our model more robust and accurate.

The model is inspired by Image processing methods as RGB images have 3 layers. We made two layers by stacking the standardised original data and the Gaussian filtered data together. We didn't remove outliers as our method can automatically learn by comparing original and filtered data to tackle outliers.

1D CNN (filters: 16,size: 11,relu), Batch Normalization, Max Pooling (size: 4)
1D CNN (filters: 32,size: 11,relu), Batch Normalization, Max Pooling (size: 4)
1D CNN (filters: 64,size: 11,relu), Batch Normalization, Max Pooling (size: 4)
1D CNN (filters: 64,size: 11,relu), Batch Normalization, Max Pooling (size: 4)
Flatten layer, Dropout (0.25), Fully connected layer (size: 32, relu)
Dropout (0.25), Fully connected layer (size: 64, relu)
Fully connected layer (size: 1, activation: sigmoid)

Model architecture

We used ADAM optimiser to learn the weights and also decreased the learning rate in subsequent training of the model over time to make the model more robust. We used 10% of the data as validation/testing to compare the F1 score between train and test data.

The model is written in keras and data is fed in batches using a batch generator function which also augments the data on fly.

We tried many combinations of the data-filter layers and the results are in below table.

Method	F1 score (macro)	F1 score (micro)
CNN (Original data)	0.9279	0.9974
CNN (Best filter: Gaussian)	0.7474	0.9898
CNN (Original+Fourier+Gaussian)	0.7461	0.9848
CNN (Original + Gaussian)(**Final**)	0.9279	0.9974

We selected the 4th one for our final model as it compares the gaussian filtered data with original data by stacking them, as well as do computation to get good results and work well on the final test data. We can combine it with 1st one for more improvement.

Since we are dealing with highly imbalanced data, validation has become difficult because eliminating a few data points for validation leaves the model with inconsistent results. However we are trying different random states to select the data points which could expose the model to the more important data points.

Goal:

Our main goal is to classify all exoplanet with minimum number of non-planet classified as planet.
Or we can say our goal is to increase the F1 (Macro) with increase in Recall of exoplanet class.

Evaluation:

The evaluation metric is **F1 score (Macro)** because it is an imbalanced class problem and **other F1 score (Micro, Weighted, Binary) will give us the approximately same results as Accuracy** and it is not a good idea to use these in an imbalanced class problem.

F1 score (Macro) :
The method is straightforward. Just take the average of the precision and recall of the system on different sets. For example, the macro-average precision and recall of the system for the given example is

Macro-average precision = (P1+P2)/2 = (57.14+68.49)/2 = 62.82
Macro-average recall = (R1+R2)/2 = (80+84.75)/2 = 82.25

The Macro-average F-Score will be simply the harmonic mean of these two figures. Macro-average method can be used when you want to know how the system performs overall across the sets of data.In, imbalanced class problem it is very useful to know F1 score on both classes.

Future work:

We are now testing other filters that we haven't used yet like **Kalman, wiener, detrend** to improve the model by getting optimal parameters and the optimal data filter layers to improve model performance.
We can use some CNN model as an encoder to **learn embeddings of the data** and use other ML/DL methods on these embeddings to get some good results. It will work as an ensembling technique to combine the models with little bit of diversity amongst them to improve the overall model performance.

References:

These are some research papers that helped us in coming up with the idea for the model.

- Artificial Intelligence on the Final Frontier: Using Machine Learning to Find New Earths - Abraham Botros
- Searching for Exoplanets using Artificial Intelligence - Kyle A. Pearson, Leon Palafox, and Caitlin A. Griffith
- Data Augmentation for Time Series Classification using Convolutional Neural Networks - Arthur Le Guennec, Simon Malinowski, and Romain Tavenard
- Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J. Leon Zhao
- Searching for exoplanets in the Kepler public data - Xiaofan Jin, David Glass