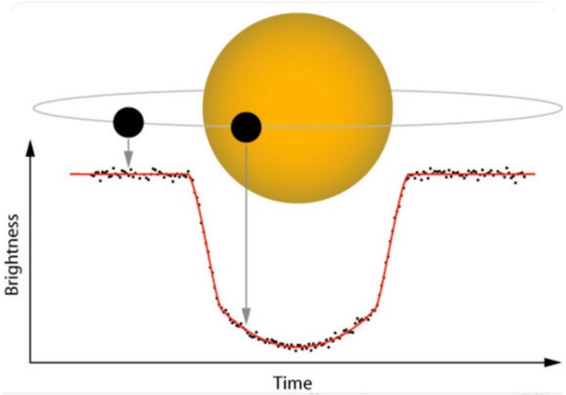
Gened 1070 Final Project
Zoe Todd
Michael Kolor

Exoplanet Hunting in Deep Space through Deep Learning Techniques

Background

For my final project, I chose to explore exoplanet hunting in deep space. A major area n the search for life outside of Earth is the detection and location of exoplanet, or planets located outside the solar system. Such detection is quite difficult, as planets do not emit light like stars. Planets are also vastly smaller than stars. In light of these limitations, scientists have devised several different methods in order to locate and identify exoplanets. One such method is the transit method. (“Down in Front”).

The transit method involves recording the measurements of flux, or the light intensity, of stars in space. The flux for stars over long periods of time are observed ,and they analyzed for variation over time. Variation, specifically drops in the measured brightness in the star, could be indicators for an existing exoplanet. This is depicted in the image (Chin) below:



When a planet passes between a star and the Earth, it blocks some of the light from being visible from the Earth (or whichever perspective).   This is marked by a drop in the observed brightness for a period of time. As the planet orbits the star, seeing repeated, identical drops on a regular interval would lead scientists to presume a planet exists around that star. The transit method is especially effective in finding exoplanets in close orbits with stars, and it also can be used to measure exoplanet diameters; how much a star dims directly relates with the relative size of the star and the planet. However, it cannot be used to find exoplanets that do not cross stars between the Earth, and it is poor at determining the blockage crossing the star’s path is a planet or simply a smaller star.   This can lead to many false positives, and it requires further examination of exoplanet “candidates” identified by the data. Still, the transit method is considered the most effective and sensitive method for detecting exoplanets, especially if equipment that can record measurements for weeks or months is available. Surveys conducted by this equipment can simultaneously study as many as 100,000 stars at a time (“Down in Front”).

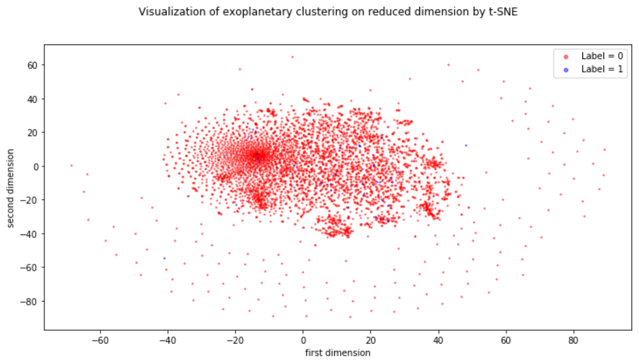
Deep Learning with the Transit Method: Building a Classification Model

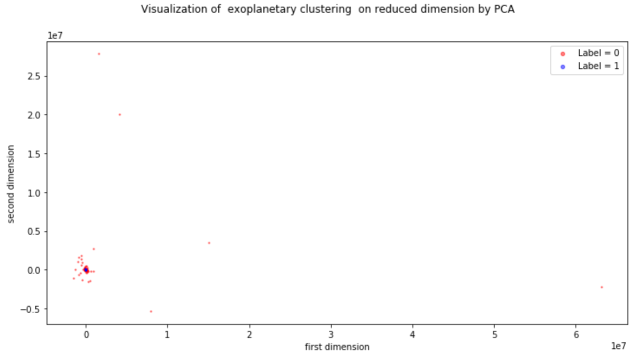
With the large magnitude of data being processed, scientists have looked for models to help classify the many stars as being candidates for having exoplanets or not. The exploration of modeling has led exoplanet detection to the field of deep learning (a subset of machine learning including types of neural networks) . Deep learning models have been used with some success in correctly classifying and identifying exoplanets in outer space (Shallue and Vanderburg). As an aspiring data scientist, I hoped to apply some recently learned concepts from Harvard's *Introduction to Data Science* two-semester course to star flux intensity data in hopes to identify exoplanets by the application of the transit method.

Data Exploration

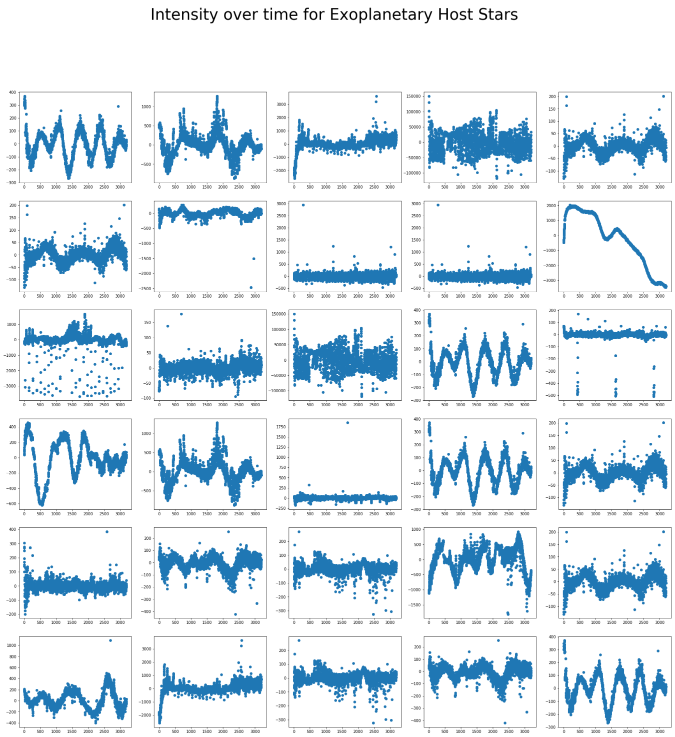
Searching through the internet, I was able to locate a dataset recorded by the Kepler satellite-telescope during its K2 phase and made open-source in the Mikulski Archive ("K2"). This data was then saved as comma-separated-files (CSV's) and published on the website *Kaggle*, a site for data science challenges and community. I worked with the data and modeling in Python, a high-level programming language with a lot of built-in functionality.

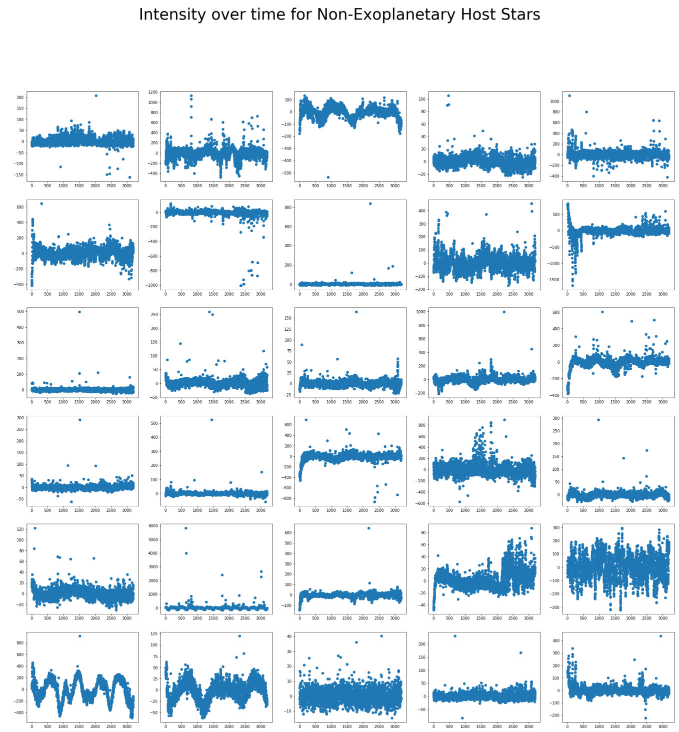
Upon initial observation, it could already begin to be appreciated the difficulty for scientists in identifying exoplanets. From the two datasets supplied, one for training models and the other for observing performance, there were only forty-two confirmed stars, with only five in the testing dataset. Compared to the almost 5600 other stars in the dataset, this makes training a model very hard, as the data is very imbalanced. The task for the model is the performance equivalent of "finding the needle in the haystack". The data was labeled, meaning that stars that were confirmed to have planets were labeled as "1", and stars not confirmed were labeled as "0". With this data, the hope would be to train a model that performs so well on the "labeled" data that, when trained on data where the candidacy for planets were unknown, it would provide useful information on the likelihood a planet existed at the star. For each star, 3200 different flux intensity readings were supplied. To begin to understand the data I was working with, I graphed scatterplots of several stars, both with and without exoplanets, as well as their clustering with the data reduced to two-dimensions. These plots can be seen below:



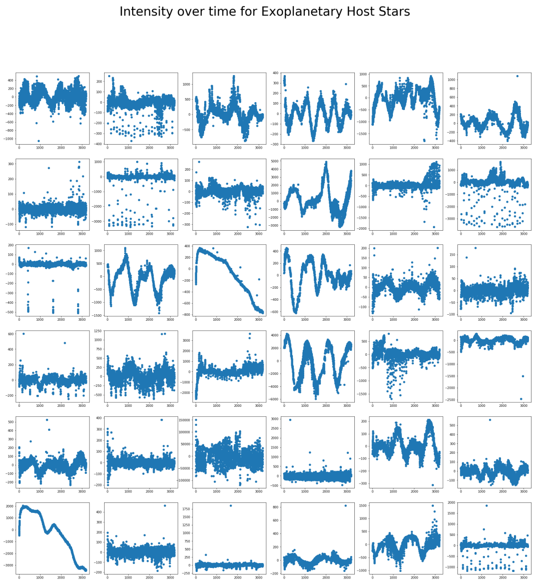


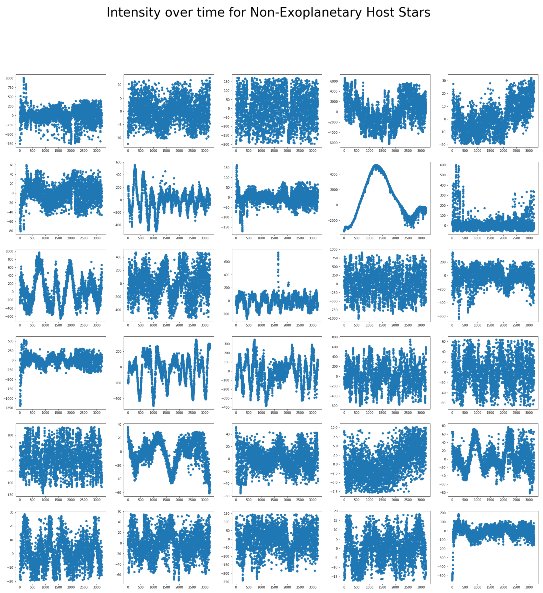
The above plots represents the clustering of the stars over the two dimensions with the most variation. To get these dimensions, the datasets were tranformed using two different methods, t-SNE, involving non-linear methods (on the left), and PCA, involving linear methods (on the right). As these plots show, clustering between exoplanet and non-exoplanet stars is not a low dimensional problem. It is likely quite sophisticated.


Intensity over time for Exoplanetary Host Stars


Intensity over time for Non-Exoplanetary Host Stars

The above plots show the scatterplots of light intensity for exoplanetary host stars (left) and non-exoplanetary host stars (right). As can be seen, there is a lot of variation in plots and intensities. Some stars without confirmed planets appear to show very consistent periodic drops in flux intensity, while some stars with confirmed exoplanets do not appear to show strong dips. There also appear to be several outliers in flux readings for all stars, as certain points in the scatterplot are very isolated. Upon initial poor performance in the modeling process, I decided to try and remove the outliers to better "smooth" the performance. I located the data with the approximate top 1% in magnitude difference compared to the point's neighbors as well as the most extreme-valued points in the dataset, and I set them equal to the average of their five closest neighbors on either side using a manually-built function. I then re-plotted scatter plots of the fine-tuned data below:


Intensity over time for Exoplanetary Host Stars


Intensity over time for Non-Exoplanetary Host Stars

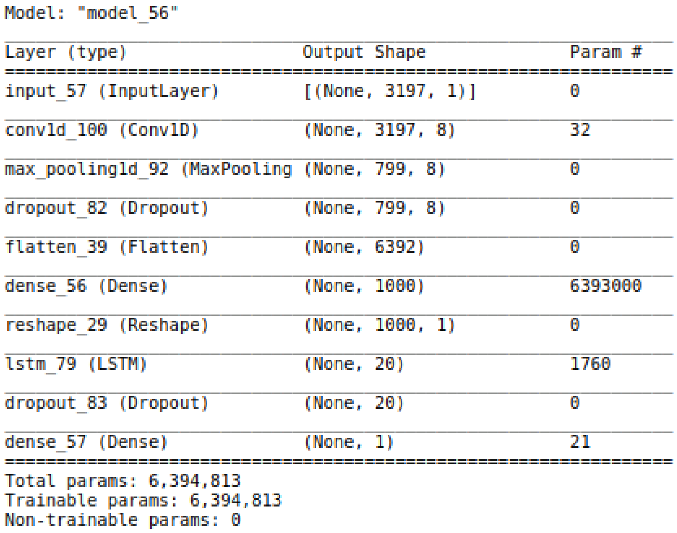
Some improvement could be seen! The scatterplots look “thicker”, which is an indication that the scales of plots have decreased dramatically due to the removal of the outliers. I then standardized the data, meaning that I made the variance of flux for each star equal 1 and its mean equal 0, as this is useful for improving performance in deep learning models. After this, I proceeded to modeling.

Models

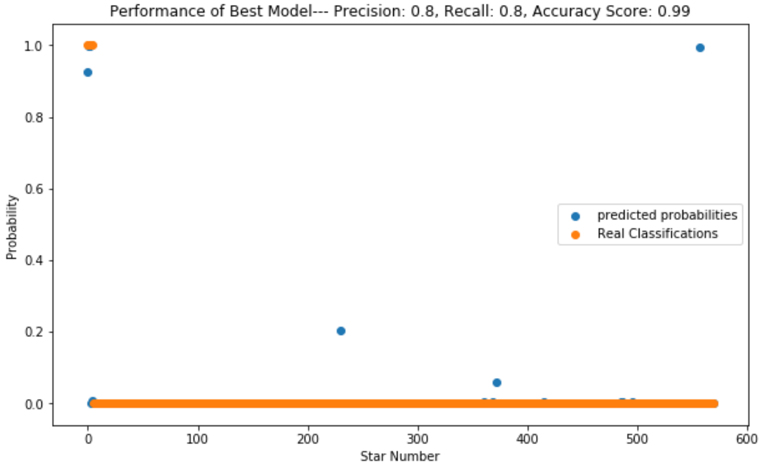
The main model I chose to focus on was the LSTM, or Long-Short-Term-Memory model. Network. This is a special case of a neural network. Neural networks are statistical, “black-box” models with extremely high performance with lower inferential understanding compared to a simpler model. They are able to learn complicated, hierarchical features in a dataset. A special subset of neural networks is recurrent neural networks, which are better at handling time-series, or sequential data. These networks store memory of previous values fed into the model and consider their relationship as information. Among the most advanced of these networks is the LSTM. At a high level, the LSTM operates similar to a human brain, with functions called “gates” that control what information is forgotten, what information is remembered and passed on for future reference, and what information is especially relevant in the present moment (Brownlee). This model has been known to have great power in working with sequential data, and given that the interest in the transit method is the sequence of light intensity, I chose to focus my work with this model.

As previously mentioned, the data is highly imbalanced. Because of this, training a simple model that optimizes correctly classifying a star as a candidate or not can achieve greater than 99% accuracy by simply predicting “no”. Thus, certain changes needed to be applied. I chose to optimize for the model’s precision, which is especially useful when a model is mostly negative values. The precision of a model is the fraction of total stars predicted to have planets by the model (true and false positives) that are actually true planets (true positives). Initially, with the uncleaned data, the model struggled significantly to differentiate the exoplanet candidates at all from the non-exoplanet candidate, so this led to the model predicting all planets as star candidates, which is a trivial model that provides no additional information to scientists. To try and account for this, I started oversampling the candidate stars. This means that I started copying the stars that have exoplanets and putting their data into the dataset multiple times. I experimented with three and six times, as doing this too much can be bad. I also put a much harsher penalty in my model for incorrectly classifying star candidates as opposed to non-exoplanetary stars, as in real life models may often overpredict in this field, and that is a less bad outcome than missing a candidate, as models that overpredict still provide useful information to rule out certain stars, while models that miss candidates are not useful (Shallue and Vanderburg).

For a while, I struggled to meet my objective goal in prediction accuracy. I implemented various model architectures, including augmenting another model before my LSTM to further analyze the data trends, but I struggled to find useful insights. However, I eventually created a model that performed very well on data I did not feed into the model! I created a model that was first a Convolutional Neural Network, or CNN, which processed the data to search for important features. It then fed its filtered data into the LSTM, which after 10 epochs of training, had both a Precision and Recall of 0.8 on the test dataset for for planets, meaning that it correctly classified 4 out of 5 exoplanet candidates in the test sets as correctly while falsely classifying only one other star as a candidates. Also, considering the performance of this model and the fact that non-candidates still could have planets orbiting around them, it makes me curious to review the “false positive’s” data. Compared to other models I created, oversampling certain data and providing harsher penalties for certain classes appears to have been a very effective tactic! My model architecture and a plot of the classifications vs the predicted probabilities are shown below:



```
Model: "model_56"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_57 (InputLayer)        [(None, 3197, 1)]         0
_________________________________________________________________
conv1d_100 (Conv1D)          (None, 3197, 8)           32
_________________________________________________________________
max_pooling1d_92 (MaxPooling (None, 799, 8)            0
_________________________________________________________________
dropout_82 (Dropout)         (None, 799, 8)            0
_________________________________________________________________
flatten_39 (Flatten)         (None, 6392)              0
_________________________________________________________________
dense_56 (Dense)             (None, 1000)              6393000
_________________________________________________________________
reshape_29 (Reshape)         (None, 1000, 1)           0
_________________________________________________________________
lstm_79 (LSTM)               (None, 20)                1760
_________________________________________________________________
dropout_83 (Dropout)         (None, 20)                0
_________________________________________________________________
dense_57 (Dense)             (None, 1)                 21
=================================================================
Total params: 6,394,813
Trainable params: 6,394,813
Non-trainable params: 0
```



In the future, I could try other complex methods to clean the data, such as Fast-Fourier Transform and Gaussian Filtering.

Conclusions

This project was a really exciting way to combine my passion for outer space with my passion for data science. The results of my modeling process truly exceeded all expectations I had! There was variation in the data that I do not have the astrophysical background to understand how to fully manage, and it was a lesson that effective data scientists need to have a lot of knowledge in the field of their data as well as their model. However, my results show that powerful models can capture so much insight on their own! I learned from my failures in the modeling process and adapted and improved, and I will continue working on this model even after I finish this course to see if I can further improve it.

Works Cited

Brownlee, Jason. “Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras.” *Machine Learning Mastery*, 7 Aug. 2019, machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/.

Chin, Andrew. “The Transit Method.” *Quora*, 4 May 2018, www.quora.com/How-does-the-transit-method-work-in-practice-for-exoplanet-discovery.

“Down in Front!: The Transit Photometry Method.” *The Planetary Society Blog*, 2020, www.planetary.org/explore/space-topics/exoplanets/transit-photometry.html.

“K2.” *Mikulski Archive Space Telescope*, archive.stsci.edu/k2/.

Shallue, Christopher J., and Andrew Vanderburg. “Identifying Exoplanets with Deep Learning: A Five-*Planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90.” The Astronomical Journal*, vol. 155, no. 2, 2018, p. 21.