Machine Learning Approaches to Exoplanet Identification via Transit Photometry

Aditya Radhakrishnan[1], Srinidhi Sunkara[2], Veeralakshmi P.[3]

[1]adityaradk@gmail.com

[2]srinidhis17112@it.ssn.edu.in

[3]veeralakshmi17181@ece.ssn.edu.in

I. Rationale

The identification of extrasolar planets is an important problem in astronomy, having major implications for planetary science, astrophysics, astrobiology, SETI, and more. Since 1995, when the first exoplanet was discovered, over 4,100 exoplanets have been confirmed. Of these, over 75% were discovered using the Transit Photometry method. This involves the analysis of tiny decreases in a star's observable brightness as a result of an exoplanet passing in front of the stellar disc.

The processing of these 'light curves' is complex, with many nuances and intricacies, traditionally necessitating a trained astronomer to sift through piles of data. With the advent of newer observatories, such as TESS, JWST, and WFIRST, the number of observations that have to be processed will only increase.

Computer models for exoplanet identification may help partially automate the process of examining the vast volumes of photometry data which will be obtained from these new observatories. They may even have the potential to find elusive exoplanets that have been overlooked.

We thus propose, excluding ensembles, 7 machine learning-based models to perform exoplanet identification using stellar brightness data captured by NASA's Kepler space telescope.

II. Problem Description

The task (T) is assessing if a light curve is indicative of the presence of an exoplanet. It is also important for the models to avoid misclassification of non-exoplanet stars.

The models gain experience (E) by repeatedly sifting through labelled light curves and optimizing their parameters in the process.

We aim to improve the performance (P) of the models by minimizing their classification error.

III. Data Collection and Details

The dataset under consideration is the "Exoplanet Hunting in Deep Space" dataset available on Kaggle, created by user WinterDelta (WΔ) [1]. This dataset is a selection of "cleaned" observations that have been classified into confirmed negatives and positives by expert astronomers. Each denoised observation consists of 3,198 light intensity readings. The dataset has also been pre-partitioned into a train and test set, each having 5087 and 570 samples respectively.

Each sample consists of 3,197 flux (light intensity) readings over time along with a label specifying if the observation is indicative of the presence of an exoplanet. Observations with specific, regular patterns of dips in flux measurements sometimes imply the presence of an exoplanet. This is a result of the planet(s) occluding a part of its/their parent star.

However, many transit light curves are not indicative of a planetary transit. Many phenomena, such as binary star systems and starspots, result in light curves that are deceptively similar to exoplanets. This is an important challenge for a detection model to overcome. This complex characteristic makes it an apt problem for machine learning-based models.

The data was obtained by the dataset creator from the Milunski Archive [2], which contains all the observations recorded by the Kepler observatory, open-sourced by NASA. In particular, the majority of it comes from Campaign 3.

IV. Data Preprocessing

A. Normalization

Prior to normalization, the flux readings ranged from approximately negative 2.4 million to positive 4.3 million. Using L2 normalization, this range was brought down to (-0.77, 1). The mean was also brought down from ~130 to a value close to 0. Similarly, the targets, which were either 1 or 2, were shifted to be either 0 or 1, with 1 indicating the presence of an exoplanet and 0 indicating otherwise.

B. Partitioning

From the test set, 10% of the positive and negative training samples were randomly removed to form a validation set. This set was used for all model comparisons, detecting overfitting, etc. The test set (as partitioned by the dataset creator) was only used at the final stage, after the best model had been selected.

C. Oversampling

A key challenge in the problem of Transit Photometry-based detection is the significant class imbalance. By a factor of ~136, there are far fewer positive samples (exoplanet stars) than there are negatives (non-exoplanet stars) within the dataset.

While most stars have planets, the orbital plane of an exoplanet must be perfectly aligned with the observer (Kepler), thus resulting in the detection of very few planets relative to the number of stars observed.

To address this massive class imbalance, oversampling techniques were used.

1) SMOTE: The Synthetic Minority Oversampling Technique, or SMOTE, generates synthetic samples for the minority class. It was first introduced by N. V. Chawla et al. [3] in 2002. Using SMOTE, the number of positive samples jumped from 34 to 4,545 - equivalent to the number of negative samples.

2) ADASYN: The Adaptive Synthetic Sampling Method for Imbalanced Data, or ADASYN, is an approach built on SMOTE. It was first introduced by Haibo He et al. [4] in 2008. Using ADASYN, the number of positive samples increased to 4,538.

D. Augmentation

Another approach to addressing the class imbalance problem is augmentation (which itself can be seen as a form of oversampling). This helps create additional positive samples.

The augmentation strategy used here involved shifting each measurement forward in the time series by a certain randomly generated integer in the range (0, 3,197). Values that were shifted beyond the length of the series were added back to the start of the series. This is also referred to as *rotation*.

For some models, this approach proved far more effective than the oversampling techniques. This is further described in the next section.

V. Models

A. Multilayer Perceptron (MLP)

This approach uses a feedforward neural network with layers of fully-connected perceptrons having nonlinear activations. Connections are present only between adjacent layers.

The MLP had 3,197 neurons in its input layer (for the 3,197 measurements in the time series) and 1 output neuron (for predicting the probability of the presence of an exoplanet) with a sigmoid activation function.

For intermediate (hidden) layers, variations in the number of layers and in the number of neurons per layer were examined and their performance evaluated. The ReLU activation was always used as the activation function for these layers. This is an established practice as ReLU's gradient does not saturate, leading to rapid convergence, amongst other advantages.

Prior to varying the hyperparameters for the hidden layers, we first examined the performance of the different oversampling techniques and augmentation.

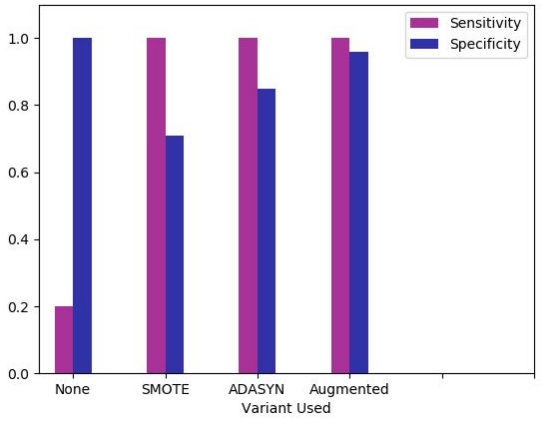

Fig. 1 Comparison of oversampling and augmentation for MLP

From these tests, it is evident that the MLP trained on the augmented data performs best.

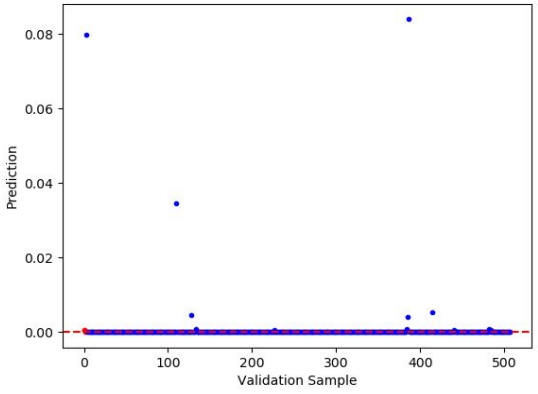

Fig. 2 Predictions of MLP with ADASYN

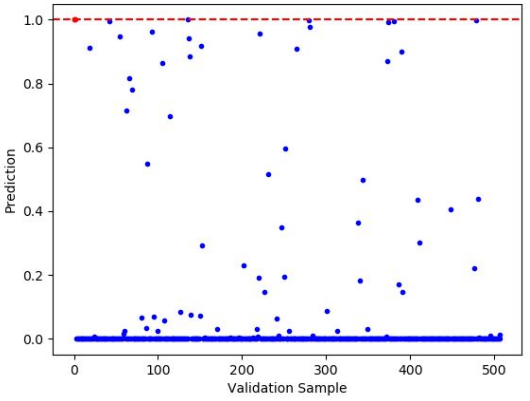

Fig. 3 Predictions of MLP with Augmentation

In the above scatterplots, the red dotted line is the threshold, the red points (validation samples 1-3) are samples indicative of an exoplanet and the blue points (validation samples 4-505) are samples of non-exoplanet stars.

While, in Fig. 1, SMOTE and ADASYN seem to be trailing not too far behind, when examining how distinctly the model predicts positive and negative samples, the model trained on the augmented set does significantly better. It clearly divides positive and negative samples and has a far lower loss. We can see this stark difference in Figures 2, 3.

We thus proceeded to choose hidden layer hyperparameters using only the augmented dataset, as it clearly outperforms SMOTE and ADASYN.

When training the models, we saved models at every epoch and tracked how validation performance changed as training progressed. When, for 20 consecutive epochs, the validation error fails to decrease, training is halted and the best performing model (on the validation set) is saved. We use this buffer of 20 epochs because the validation set, like the train and test sets, also has a massive class imbalance. This can cause the validation accuracies/losses to fluctuate significantly, so a temporary increase in validation loss may not indicate overfitting.

We used the Binary Cross-Entropy loss function to train the MLP.

Four key variants of the MLP with different hidden layers were experimented with. Alongside these, several iterative tweaks to the hyperparameters were made to narrow down on the best performing model.

These four key variants were the following:

- 1024 neuron hidden layer
- 0.3 Dropout + 1024 neuron hidden layer
- 0.3 Dropout + 1024 neuron hidden layer + 0.3 Dropout + 64 neuron hidden layer
- 0.3 Dropout + 2048 neuron hidden layer + 0.3 Dropout + 128 neuron hidden layer

The specificities and sensitivities of each of these models are shown in the bar graph below:

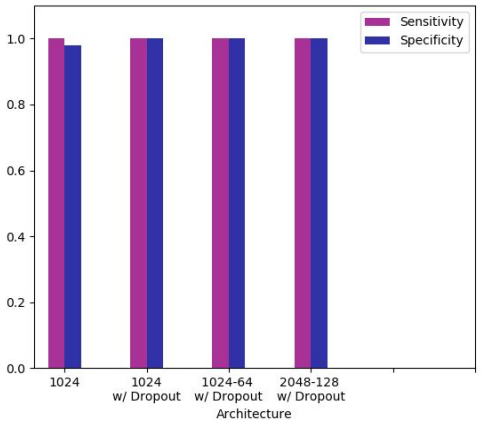

Fig. 4 Comparisons of different MLP architectures

The addition of dropout enables the classifier to correctly classify many negative samples correctly that would've become false positives if not used.

The addition of another hidden layer with 64 neurons does not change the already perfect specificity and sensitivity, but when analyzing predictions in a similar manner to what was done in Fig. 2, 3, we can determine that the addition of a 64 neuron hidden layer improves performance. It results in a lower average BCE loss on the validation set.

The use of the 2048-128 architecture provided no performance gain and resulted in a slightly higher BCE loss. Training was also halted far sooner than the other three architectures - it was the fastest to overfit.

From these tests performed on the validation set, we selected the 1024-64 architecture using the augmented dataset.

B. Convolutional Neural Network (CNN)

This approach involves the use of an artificial neural network variant geared towards finding spatial patterns in data. It consists of layers of convolutional filters, activation layers, fully-connected layers of perceptrons, etc.

We use this model with the expectation that the periodic reductions in stellar brightness can be interpreted as a spatial pattern that the CNN can identify.

In a similar manner to what was done with the MLP, we first analyzed the performance of the model using different oversampling methods and unbalanced data:

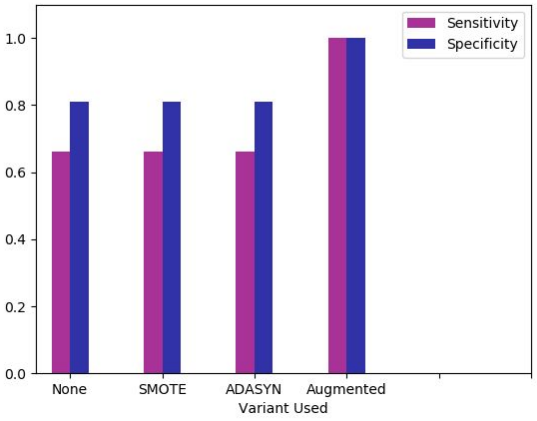

Fig. 5 Comparison of oversampling and augmentation for CNN

Once again, the augmented data results in far better performance than SMOTE, ADASYN, or the unbalanced dataset. Here, this discrepancy can possibly be explained - the augmented dataset diversifies and enriches the spatial variations in the dataset, while SMOTE and ADASYN do little in this regard.

Similar to the MLP, we tested a few key architectures using a BCE loss and the same early stopping procedure.

The model that was finally chosen for its performance on the validation set had the following layers:

1. A 1D convolutional layer with 8 filters with a size of 11 (ReLU activation).
2. A Max Pooling layer
3. A Batch Normalization layer
4. A 1D convolutional layer with 16 filters with a size of 11 (ReLU activation).
5. A Max Pooling layer
6. A Flatten layer
7. A dropout of 0.3
8. A 64 neuron ReLU activation fully-connected layer
9. A dropout of 0.2
10. A single neuron output layer with a sigmoid activation function

Using fewer layers/neurons/filters caused the model to have a lower accuracy, likely because of mild underfitting. Adding more parameters was redundant and only made the models more prone to overfitting.

C. Random Forest Classifier

A Random Forest Classifier, when applied to the normalized data, was not able to detect the presence of an exoplanet. On oversampled data obtained after applying the ADASYN method with a one tree model, a few exoplanets were detected, but a few incorrect predictions were also made. With 2 trees the classifier was able to perfectly classify the data. The classifier already performed fairly well on the unbalanced data. With the addition of class balancing, the random forest classifier is able to classify data better. With the application of class weighting, no improvement was observed.

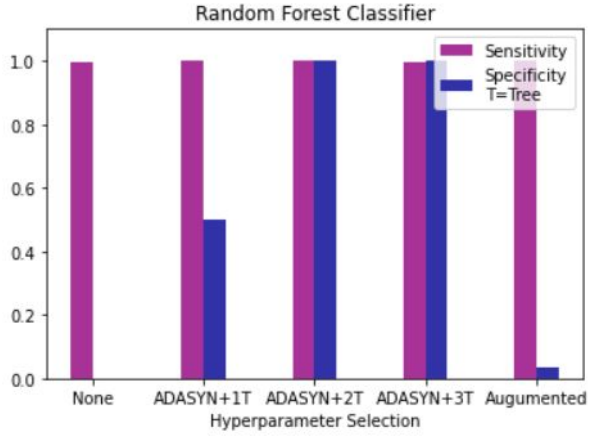

Fig. 6 Comparison of oversampling and augmentation and different hyperparameters for Random Forest

D. K-Nearest Neighbors Classifier

A K-Nearest Neighbors classifier, when applied to the unbalanced dataset, classified every validation datapoint as a negative. With the use of ADASYN, the classifier was able to detect the presence of an exoplanet, but it came with more cost as it detected around 20% of samples with no planet presence as the ones with planets. This classifier worked better on ADASYN data than on the SMOTE data. Adasyn focuses on the samples which are difficult to classify with a nearest-neighbors rule while regular Smote will not make any distinction. Since in KNN the distance metric is used for the classification, ADASYN data gave good results whereas with SMOTE it did not. Class weighting yielded no performance improvement.

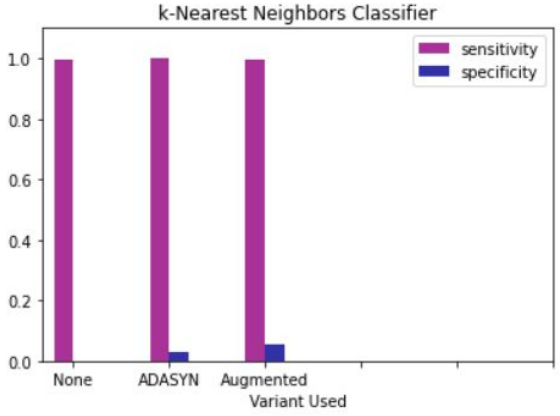

Fig. 7 Comparison of oversampling and augmentation for KNN

E. Naive Bayes Classifier

Naive Bayes Classifier was able to detect one positive sample when applied on normalized data. This classifier is sensitive to the class imbalance, so when applied on SMOTE data it was able to predict two positive samples and 496 negative samples correctly with only 10 false predictions. Once again, class weighting had no positive impact on performance.

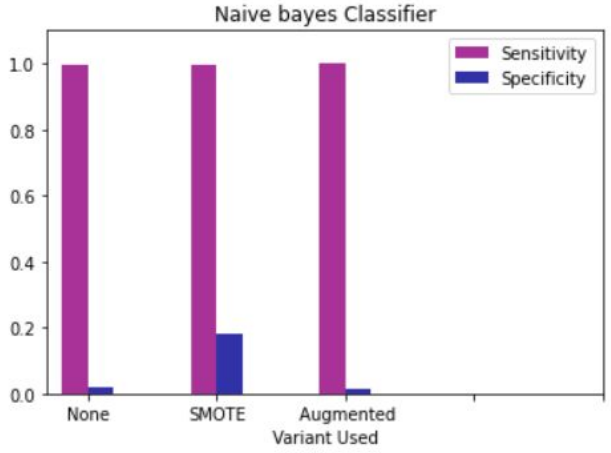

Fig.8 .Comparison of oversampling and augmentation for Naive Bayes

F. Support Vector Machine (SVM) Classifier

Support Vector Machine Classifier, when applied on normal data, classified everything as negative samples which is absolutely incorrect. For the SMOTE data, SVM managed to predict all positive samples correctly with 93

false predictions. All the non-linear kernels applied on SMOTE data gave the same results because for the high dimensional data linear solvers are fast to train and extremely fast during validation than non linear kernels. SMOTE generates the same number of synthetic samples for each original minority sample.. Since the decision made by svm is based on support vectors it is important to generate those vectors properly and SMOTE was able to do that better than ADASYN. Class weighting failed to provide any performance improvements.

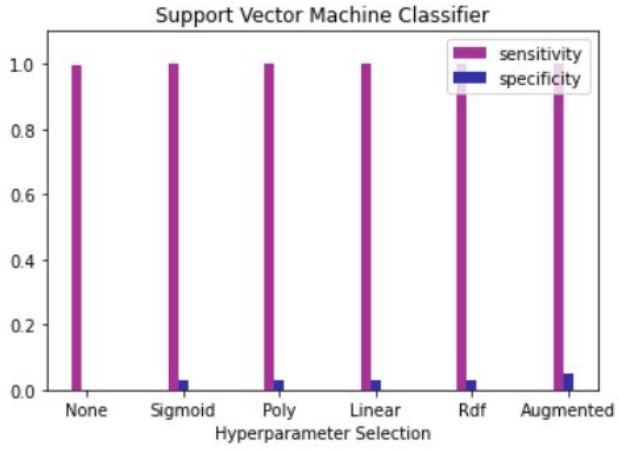

Fig. 9 Comparison of oversampling and augmentation and different hyperparameters for SVM

G. AdaBoost Classifier

The AdaBoost classifier is used to boost the performance of any machine learning algorithm. It is best used with weak learners since the dataset is highly imbalanced. On the validation set, AdaBoost managed to predict all positives correctly. No improvement was observed with the inclusion of

H. Bagging Ensemble

A bagging ensemble learns from multiple weak learners. By doing this, it can outperform a single strong learner in many tasks. This ensemble method reduces variance and avoids overfitting. These features of bagging helped the model to predict accurately on the validation set.

I. Voting Ensemble

This approach involves combining select models from subsections A-G, where each model casts a vote depending on whether it thinks a sample is positive or negative. The mode of the votes is taken as the consensus - the prediction of the model. If there are two modes, the vote is interpreted as a positive prediction, as the importance of a true positive is far greater than a true negative.

To select models for the voting ensemble, a “brute force” approach was used, where all the 121 model combinations were evaluated on the test set. It was intended that the ensemble that outperforms the rest would be chosen.

However, many ensembles ended up having perfect accuracy. To narrow the list, we analyzed the frequency of the use of different models across the ensembles that had a perfect score. We found that the AdaBoost, MLP, CNN, and Random Forest classifiers were the most used. Additionally, this helped remove redundant models that did not contribute much to the final outcome.

With this rationale, a voting ensemble of these four models was created.

VI. Results

In this section, we wish to discuss the performance of the models selected in section V on the test set. Note that each of the tables contain rounded values - please refer to the confusion matrices for the raw results.

A. Multilayer Perceptron (MLP)

The performance of the MLP was decent, with 4 exoplanets being correctly classified, but with a fair share of false positives.

TABLE I
MLP Test Set Results

Measurement	Value
Specificity (TP Rate)	0.80
Sensitivity (TN Rate)	0.96
Accuracy	0.96
Mean of TP & TN Rates	0.88
Precision	0.80
F1 Score	0.27

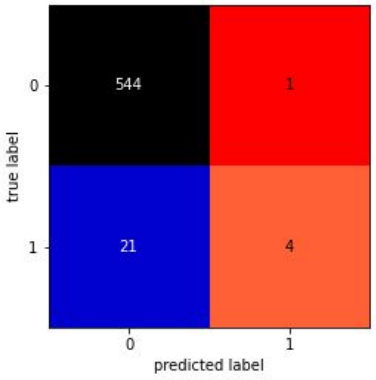

Fig. 10 Confusion matrix for MLP

B. Convolutional Neural Network (CNN)

The CNN is the highest performing model by a fair margin. It has a perfect TP rate and only misclassified 3 negative samples as positives.

TABLE II
CNN Test Set Results

Measurement	Value
Specificity (TP Rate)	1.00

Sensitivity (TN Rate)	0.99
Accuracy	0.99
Mean of TP & TN Rates	1.00
Precision	1.00
F1 Score	0.77

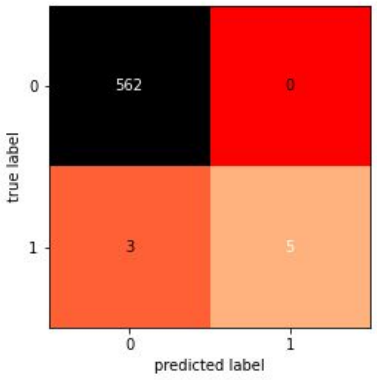

Fig. 11 Confusion matrix for CNN

C. *Random Forest Classifier*

This model fails to detect true positives.

TABLE III
RANDOM FOREST TEST SET RESULTS

Measurement	**Value**
Specificity (TP Rate)	0.00
Sensitivity (TN Rate)	1.00
Accuracy	0.99
Mean of TP & TN Rates	0.50
Precision	0.00
F1 Score	0.00

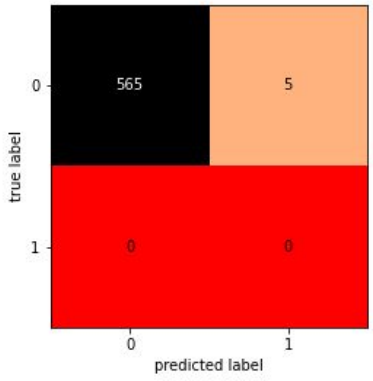

Fig. 12 Confusion matrix for Random Forest

D. *K-Nearest Neighbors Classifier*

This model fails to detect true positives.

TABLE IV
KNN TEST SET RESULTS

Measurement	**Value**
Specificity (TP Rate)	0.00
Sensitivity (TN Rate)	1.00
Accuracy	0.99
Mean of TP & TN Rates	0.50
Precision	0.00
F1 Score	0.00

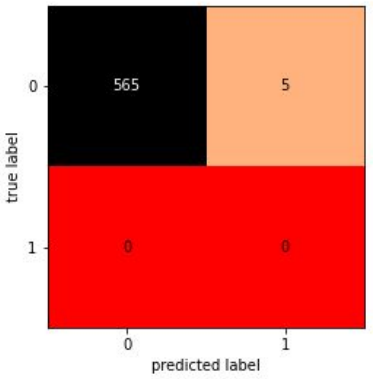

Fig. 13 Confusion matrix for KNN

E. *Naive Bayes Classifier*

The Naive Bayes classifier is the most performant non-NN model amongst the ones taken into consideration.

TABLE V
NAIVE BAYES TEST SET RESULTS

Measurement	**Value**
Specificity (TP Rate)	0.60
Sensitivity (TN Rate)	0.93
Accuracy	0.93
Mean of TP & TN Rates	0.77
Precision	0.60
F1 Score	0.13

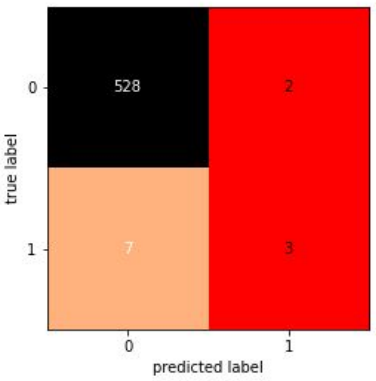

Fig. 14 Confusion matrix for Naive Bayes

F. *Support Vector Machine (SVM) Classifier*

This model fails to detect true positives.

TABLE VI
SVM TEST SET RESULTS

Measurement	**Value**
Specificity (TP Rate)	0.00
Sensitivity (TN Rate)	1.00

Accuracy	0.99
Mean of TP & TN Rates	0.50
Precision	0.00
F1 Score	0.00

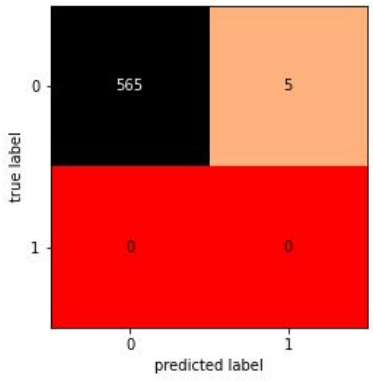

Fig. 15 Confusion matrix for SVM

G. *AdaBoost*

This model fails to detect true positives.

TABLE VII
ADABOOST TEST SET RESULTS

Measurement	**Value**
Specificity (TP Rate)	0.00
Sensitivity (TN Rate)	1.00
Accuracy	0.99
Mean of TP & TN Rates	0.50
Precision	0.00
F1 Score	0.00

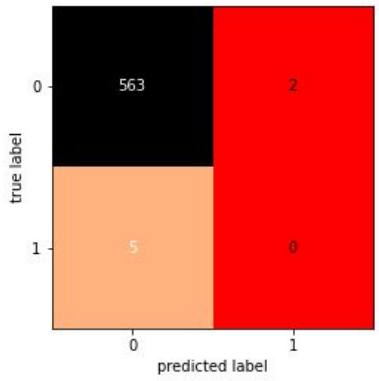

Fig. 16 Confusion matrix for AdaBoost

H. *Bagging Ensemble*

This model fails to detect true positives. It also misclassifies a considerable number of negatives as positives.

TABLE VIII
BAGGING ENSEMBLE TEST SET RESULTS

Measurement	**Value**
Specificity (TP Rate)	0.00
Sensitivity (TN Rate)	0.97
Accuracy	0.96
Mean of TP & TN Rates	0.48
Precision	0.00
F1 Score	0.00

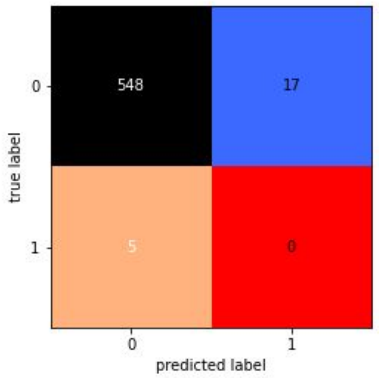

Fig. 17 Confusion matrix for Bagging Ensemble

I. *Voting Ensemble*

The voting ensemble, while having a decent performance, performs far worse than some of its constituents. It is likely that the three non-NN models are essentially 'dragging' the neural networks down. While the TN rate has improved, it is merely a consequence of the non-NN models' overall negative bias. It is clear that it does not make sense to use the voting ensemble over the independent neural networks.

TABLE IX
RANDOM FOREST TEST SET RESULTS

Measurement	**Value**
Specificity (TP Rate)	0.60
Sensitivity (TN Rate)	1.00
Accuracy	0.99
Mean of TP & TN Rates	0.80
Precision	0.60
Recall	0.75
F1 Score	0.67

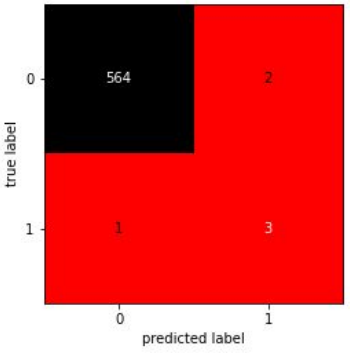

Fig. 18 Confusion matrix for Voting Ensemble

VII. Conclusion

From the test results, we have determined that the neural network-based approaches, in particular the CNN, outrank the other classifiers in terms of performance. The best non-NN classifier is the Naive Bayes model, which is also able to perform classification to some degree. Most of the non-NN models struggle to detect positive samples in the test set and correctly classify them as such. Ensembles also do not help improve performance for this task.

The best models that have been trained are very performant, with the best perfectly detecting every exoplanet in the test set and only misclassifying 3 negative light curves as positives.

The application of machine learning techniques is thus likely a useful tool for the detection of exoplanets. This difficult task is certainly addressable using some of these models.

VIII. Future Work

The effectiveness of the model in correctly detecting "deceptive" phenomena (binary stars, starspots, etc.) was not evaluated in this work. The dataset used did not include metadata for each sample, which is needed for this kind of analysis to be performed. While the dataset did likely contain many samples with these phenomena, a quantitative analysis needs to be performed.

In general, the use of more data will likely have a significant positive impact. By processing observations with inconsistent formats from different missions, a larger dataset can be created, resulting in better models and stronger analyses.

References

[1] WinterDelta, *Exoplanet Hunting in Deep Space*, Kaggle, 2016. Accessed on: June 8, 2020. [Online]. Available: https://www.kaggle.com/keplersmachines/kepler-labelled-time-series-data

[2] Milunski Archive, *Kepler*. Accessed on: June 8, 2020. [Online]. Available: https://archive.stsci.edu/kepler/

[3] N. V. Chawla et al., "SMOTE: Synthetic Minority Over-sampling Technique," arXiv:1106.1813 [cs.AI], Jun. 2011.

[4] Haibo He et al., *ADASYN: Adaptive Synthetic Sampling Approach for ImbalancedLearning*, April 2010. Accessed on: June 8, 2020. [Online]
Available: https://sci2s.ugr.es/keel/pdf/algorithm/congreso/2008-He-ieee.pdf