Discover Exoplanet Transit Events by Semi-Supervised Learning Approach

EE 660 Course Project

Chenying Wang, chenying.wang@usc.edu

December 3, 2020

Project Type: (1) Design a system based on real-world data

Abstract

In the project, several supervised learning models and semi-supervised learning models are presented and implemented for classification of Threshold-Crossing Events (TCEs) to help identify the exoplanets. A semi-supervised model based on cluster-then-label is proposed in this project. Together with some feature engineering methods, the final results shows that random forest classifier achieves the highest test accuracy of 96.51% as well as highest test precision of 94.03%, while our cluster-then-label classifier reaches the 96.55% test accuracy and 93.36% precision. Meanwhile, the semi-supervised cluster-then-label classifier shows its strength and robustness in the lack-of-labeled-sample scenario since the cost of labeling TCEs cannot be neglected.

1 Introduction

Kepler [1] is a NASA exoplanets (extrasolar plants) search mission by analysis of the contiguous, photometric time series of stars collected by *Kepler* space telescope. From March 2009 to October 2018, *Kepler* mission has been able to discover 2,394 exoplanets from almost 200,000 target stars.

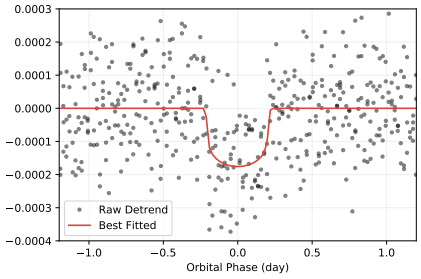


Figure 1: Relative Flux of *Kepler*-452b during a Transit Period

1.1 Problem Assessment and Goals

There are Threshold-Crossing Events (TCEs) in the host stars of exoplanets, which are periodic transit-like features in the flux time series, caused by the blocking of the steller light when the exoplanet is between the *Kepler* space telescope and the host star. In Fig. 1, it shows the flux of a confirmed exoplanets during the period of transit. These TCEs are detected and collected from by *Kepler* processing pipeline [2], and deliver several features for each TCE at mean time. However,

most TCEs are not a potential planet candidate (PC) but may result from astrophysical false positive (AFP), *e.g.* eclipsing binary stars or non-transiting phenomena, *e.g.* instrumental noise. As a result, further review process is required for each TCE to determine if it is corresponding to existence of a exoplanet.

In this project, two classes, AFP and NTP, will be combined to a single negative classes in this problem, so a binary classification problem is raised as whether a given TCE is related to a planet candidate. *Kepler* Q1-Q17 Data Release 24 [3] will be utilized for this project. Additionally, ground truth of 4,630 TCEs in the dataset are still unknown (UNK) and we regard those TCEs as unlabeled data samples. Also, considering there are 3,600 positive labeled samples and 12,137 negative labeled samples, the dataset suffers unbalanced issue. Feature engineering is required for this project as there are 141 features of TCEs, including numerical, categorical and descriptive. The classification models will be measured by precision, accuracy, receiver operating characteristic (ROC) curve and area under curve (AUC).

1.2 Literature Review

McCauliff *et al.* [4] presented *Autovetter* project to classify TCEs with a random forest based model. Armstrong *et al.* [5] proposed a unsupervised machine learning model based on self-organizing maps (SOMs) to identify planets in *Kepler* and *K2*. Shallue and Vanderburg [6] proposed a deep convolutional neural network (CNN), that directly use time series data of global and local view of TCEs, *Astronet* to estimate if a TCE is related to exoplanet transits. Ansdell *et al.* [7] enhanced the network architecture *Astronet* model with additional scientific domain knowledge and improve the input representation in order to increase the accuracy of the CNN model. Osborn *et al.* [8] presented a neural network model to identify planet candidates by using *TESS* pixel-level simulations data.

1.3 Our Prior and Related Work - None

1.4 Overview of Our Approach

In this project, machine learning approaches will be utilized for TCE classification problem. In Section 2, the dataset will be briefly described and the dataset usage and feature engineering methods will also be elaborated. Moreover, several supervised models, including Naïve Bayes and random forest, and two semi-supervised learning models will be presented, trained and validated on the dataset containing both labeled and unlabeled data samples. Training hyperparameter of each classification models will be searched by group *5*-fold cross validation upon disjoint training and validation datasets. Then in Section 3, the selected classification models will be tested and compared on test dataset. Supervised learning models and semi-supervised learning model will also be trained and tested on dataset with different number of labeled training data samples.

2 Implementation

In the project, machine learning models and numerical operations are implemented based on *scikit-learn* [9] and *NumPy* [10]. The visualization is delivered with *Matplotlib* [11].

2.1 Dataset

Kepler Threshold-Crossing Events Q1-Q17 Data Release 24 [1] will be used in this project. In the TCE table, there are 20,367 data samples collected from Quarter 1 to Quarter 17 of which each has about 140 features and the label to be estimated, `av_training_set`. Each data sample represents a TCE, and consists of five major groups of features and label.

1. Transit Fit Parameters: Transit-related best-fit parameters to multi-quarter light curves

[1] https://exoplanetarchive.ipac.caltech.edu

2. Scaled Planetary Parameters: Physical planetary fit parameters

3. Stellar Parameters: Physical parameters of the host star of the TCE

4. Light Curve-Based Statistics: Wavelet-based statistics on light curve derived by Transiting Planet Search (TPS) module in *Kepler* pipeline

5. Pixel-Based Statistics: Pixel-based parameters derived from pixel-level data by the flux-weighted centroid analysis and the PRF-fit difference image method

6. Label `av_training_set`: Disposition of TCE (PC, AFP, NTP or UNK)

Features in the dataset comes into three form: 1) numerical, 2) categorical and 3) descriptive text/comment. Most numerical features also come with numerical positive uncertainty and negative uncertainty. More specific details of features and label are given at *Kepler TCE Data Column Definitions*.

2.2 Dataset Methodology

Firstly, the whole dataset $\mathcal{D}$ will be operated essential feature and label engineering, including to convert categorical features to numerical features, and merge label AFP and label NTP as single negative label as mentioned above. Then, the dataset $\mathcal{D}$ will be split into two parts unlabeled dataset $\mathcal{D}^{(u)}$ and labeled dataset $\mathcal{D}^{(l)}$ based on whether the sample is labeled as UNK.

Next, the labeled dataset will be split into three disjoint parts, *i.e.* pre-train dataset $\mathcal{D}^{(l)}_{PT}$, training dataset $\mathcal{D}^{(l)}_{Tr}$ and test dataset $\mathcal{D}^{(l)}_{Test}$. The pre-train dataset $\mathcal{D}^{(l)}_{PT}$ is used in model consideration and feature selection. The training dataset $\mathcal{D}^{(l)}_{Tr}$ is utilized in major training process and model selection. And the test dataset $\mathcal{D}^{(l)}_{Test}$ is to estimate the out-of-sample performance of the optimal model. Moreover, the unlabeled dataset $\mathcal{D}^{(u)}$ will also be utilized to assist the semi-supervised learning models in the training process.

Algorithm 1: k-Fold Cross Validation for Semi-Supervised Learning Models

Input: h, $\mathcal{D}^{(l)}_{Tr}$, $\mathcal{D}^{(u)}$
Output: avg_score
$\mathcal{D}_{Tr} = \mathcal{D}^{(l)}_{Tr} \bigcup \mathcal{D}^{(u)}$;
$\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_k\} = \text{Split}(\mathcal{D}^{(l)}_{Tr}, k)$;
$score := 0$;
for $i := 1$ **to** k **do**
  Train(h, $\mathcal{D}_{Tr} \setminus \mathcal{D}_i$);
  $score$ += Test(h, $\mathcal{D}_i$);
end
$avg\_score := score/k$;

In this project, group k-fold cross validation is used for model selection, shown in Algorithm 1, in which the number of splits $k = 5$. For supervised learning models, the only difference is simply to discard unlabeled dataset $\mathcal{D}^{(u)}$ in the training process. The data-based feature preprocessing, including feature standardization, are inside the training process of models. Once the model selection is finished, the optimal model h_g will be selected based on best cross-validation accuracy. The optimal model h_g will be retrained on whole training dataset, both $\mathcal{D}^{(l)}_{Tr}$ and $\mathcal{D}^{(u)}$, then will be tested on the test dataset only once, and no hypothesis, analysis and decision is made on test dataset in this project.

Additionally, considering the characteristics of this problem and this project, some TCEs may share the identical stellar parameters as they have same host star, and thus same KepID (*Kepler* star identification). This fact is attributed to more than one TCEs and planets locating in the same

star system, and one wellknown example is that there are eight known planets within our solar system. As a consequence, in order to avoid any data snooping issue may impair the generalization of classification models, the TCEs have no intersection in KepID and stellar parameters between training and test datasets and also among cross-validation datasets.

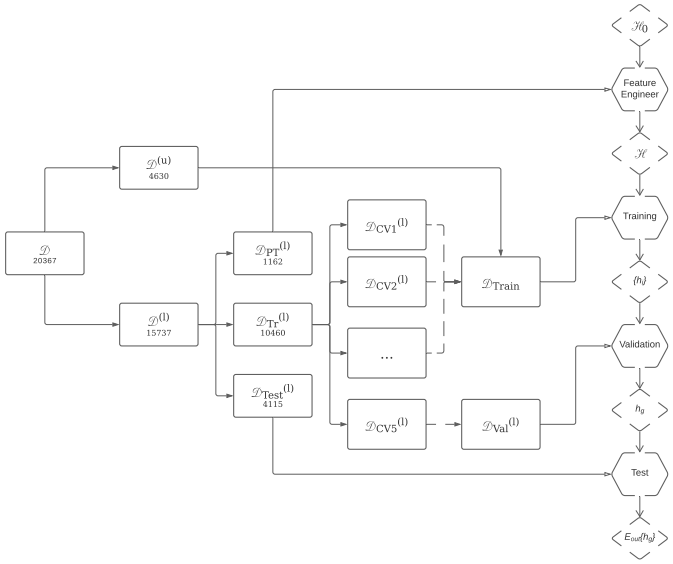


Figure 2: Project Diagram

In a nutshell, the diagram of whole dataset and model procedures of the project is given in Fig. 2 with size of each dataset.

2.3 Preprocessing, Feature Extraction, Dimensionality Adjustment

Firstly, handcrafted feature selection is operated on dataset. We checked the missing ratio of each feature. and features with 50% missing ratio or above is dropped from the dataset as well as some significantly irrelevant features. From total 141 features, 95 features are selected to use for subsequence procedures. Then the missing values in selected features is filled with mean value of each feature.

Secondly, a machine-learning-based feature selection is also introduced to this project in order to reduce the dimension of feature space. Feature selection relies on support vector machine (SVM) with linear kernel and L1 regularizer. The objective function of SVM with L1 regularization is

$$J(\underline{w}, w_0, \lambda) \ = \sum_{i=1}^{N_{PT}} \mathbb{I}(\tilde{y}_i(\underline{w}^T \underline{x}_i + w_0) \geq 0) + \lambda ||\underline{w}||_1$$

As L1 regularization term $\mathrm{L}_1(\underline{w}, \lambda) = \lambda ||\underline{w}||_1$ can provide more sparsity for weight vector, it may help us select key features. Pre-train dataset $\mathcal{D}_{PT}^{(l)}$ is used in this L1-based feature selection and the procedure is as follows. Standardize the pre-train dataset based on its own statistics, then train aforementioned SVM model on standardized pre-train dataset to select N_{FS} features, then perform a *3*-fold cross validation of a perceptron model trained on selected features of pre-train dataset. The results of average cross validation accuracy of the perceptron model on pre-train dataset against different number of features N_{FS} is given in Fig. 3. As observed from Fig. 3, the model has the best cross validation accuracy with 60 features. Therefore, $N_{FS} = 60$ features are selected by L1-regularized linear SVM based on pre-train dataset.

Then all features must be scaled to same range, zero mean and unit variance, and the standardization scaler is trained based on training dataset group or whole training dataset in cross validation

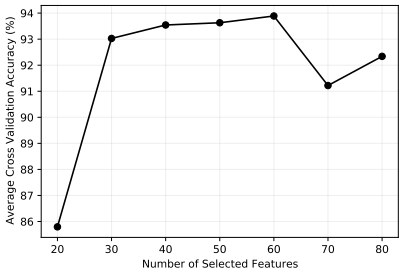


Figure 3: Cross Validation Accuracy of Different N_{FS} on Pre-Train Dataset

and test respectively. That is to say the scaler is trained just before the classification model is trained.

2.4 Training Process

Since there are unlabeled data in the dataset, both supervised learning (SL) model and semi-supervised learning (SSL) models will be utilized and implemented in the project. All models, except Naïve Bayes, will be operated grid search with group 5-fold cross validation on the labeled training dataset $\mathcal{D}_{Tr}^{(l)}$ and the unlabeled $\mathcal{D}^{(u)}$ (for SSL models) to find the best hyper-parameters. To avoid data snooping, note that the feature standardization are inside the cross validation loop.

2.4.1 Supervised Learning

We use Naïve Bayes as our SL baseline model with assumption data samples within each classes is Gaussian distributed as follows.

$$p(\underline{x}|y_i) = N(\underline{x}|\mu_i, \sigma_i^2)$$

The average 5-fold cross validation accuracy is 78.26%.

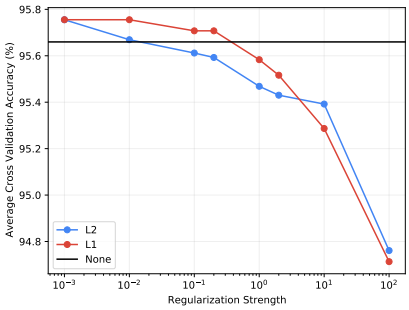


Figure 4: Cross Validation Accuracy of Logistic Regression with Regularization Configurations

Another SL model used in the project is logistic regression since this model is simple and linear and the complexity of logistic regression is very low compared following models. Additionally, to avoid overfitting, L2 and L1 regularization with different strength will be grid-searched and 5-fold cross validated on training dataset. The cross validation accuracy of different regularization configuration is shown in Fig. 4 and the highest cross validation accuracy is 95.76% where the strength of L2 regularization is 0.001.

Non-linear classification models are also used, and the random forest, as a bootstrap aggregating (bagging) method, is considered for the project. For random forest, the number of trees N_{Tree} and then maximum depth d_{max} of trees are searched through cross validation. The cross validation of

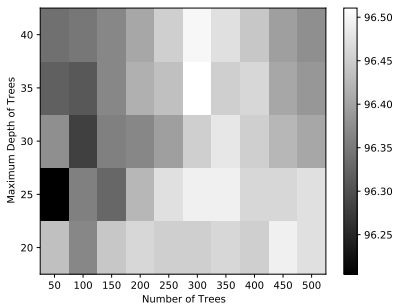


Figure 5: Cross Validation Accuracy of Random Forest with Parameters

grid search of random forest parameters is shown in Fig. 5. The best parameter of random forest is $N_{Tree} = 300$ trees with maximum depth of $d_{max} = 35$, and the cross validation accuracy is 96.51%.

Besides bagging model, another ensemble model, Adaboost with decision stumps is utilized for this classification problem to compare the performance of two different ensemble models. As observed from Fig. 6, it turns out that Adaboost with 400 decision stumps reaches the highest cross validation accuracy of 96.22%.

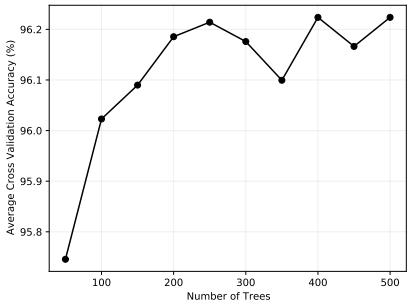


Figure 6: Cross Validation Accuracy of Adaboost with Number of Decision Stumps

Lastly, to compare the following SSL propagating k-NN algorithm, here we also tested SL k-NN with different number of neighbors From Fig. 7, *6*-NN has the highest cross validation accuracy of 94.75%.

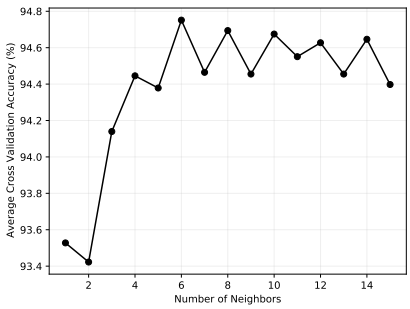


Figure 7: Cross Validation Accuracy of k-NN

2.4.2 Semi-Supervised Learning

For this particular project and problem, there are some unlabeled data samples and it is very difficult to label a data sample as well. Thus, a semi-supervised learning model will make valuable contribution to the problem.

Firstly, the propagating k-NN model is a self-training SSL model based on k-NN. A self-training model can soft-label the unlabeled data samples, and add them to the labeled dataset during the training process. For propagating k-NN model, at each iteration in the training process, it will soft-label a nearest unlabeled data sample, then add it into the labeled training dataset. And the iteration will continue until the unlabeled dataset is empty. However, in light of the running time of training process, at each iteration, $M = 100$ nearest data samples will be soft-labeled and then added into the labeled training dataset.

Algorithm 2: Propagating k-NN Algorithm

Input: $\mathcal{D}^{(l)}$, $\mathcal{D}^{(u)}$, k
Output: h_k
kNN_Train(h_k, $\mathcal{D}^{(l)}$, k);
repeat
  for $i := 1$ **to** M **do**
    $\underline{x} = \text{argmin}_{\underline{x} \in \mathcal{D}^{(u)}} d(\underline{x}|h_k)$;
    $y = \text{kNN\_Predict}(h_k, \underline{x})$;
    $\mathcal{D}^{(u)}$ -= $\underline{x}$;
    $\mathcal{D}^{(l)}$ += $(\underline{x}, y)$;
  end
  kNN_Train(h_k, $\mathcal{D}^{(l)}$, k);
until $\mathcal{D}^{(u)}$ *is empty*;

Similar with k-NN, we also search the number of neighbors k by *5*-fold cross validation and the results are shown in Fig. 8 where the propagating *10*-NN reaches the highest cross validation accuracy of 94.76%.

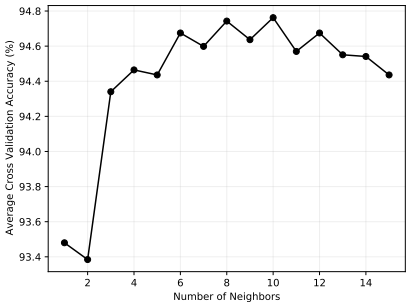


Figure 8: Cross Validation Accuracy of Propagating k-NN

Another SSL model used in the project is cluster-then-label described in [12] as follows and in Algorithm 3. The first step of this algorithm is to cluster the whole training dataset, including labeled and unlabeled, to M clusters. And then M classifiers are trained for each cluster with in-cluster labeled samples, or with all labeled samples if there is no labeled sample in the cluster.

In Zhu and Goldberg [12] for prediction, the unseen data is identified the cluster and then

Algorithm 3: Cluster-Then-Label Algorithm

Input: $\mathcal{D}^{(l)}$, $\mathcal{D}^{(u)}$, cluster algorithm $\mathcal{A}$, classification algorithm $\mathcal{L}$, number of clusters M
Output: f, $\{g_i\}_{i=1}^{M}$
f = Train($\mathcal{A}$, $\mathcal{D}^{(l)} \bigcup \mathcal{D}^{(u)}$, M);
for $i := 1$ **to** M **do**
  $\mathcal{D}_i^{(l)} = \{\underline{x}_j, y_j | f(\underline{x}_j) = i, \underline{x}_j \in \mathcal{D}^{(l)}\}$;
  if $\mathcal{D}_i^{(l)}$ *is not empty* **then**
    g_i = Train($\mathcal{L}$, $\mathcal{D}_i^{(l)}$);
  else
    g_i = Train($\mathcal{L}$, $\mathcal{D}^{(l)}$);
  end
end

predicted based on the corresponding classifier as

$$\begin{aligned} f_i(\underline{x}) &= p(z_i|\underline{x}, \underline{\theta}_{\mathcal{A}}) \\ g_i(\underline{x}) &= p(y|\underline{x}, z = z_i, \underline{\theta}_{\mathcal{L}}) \\ p(y|\underline{x}, \underline{\theta}_{\mathcal{A}}, \underline{\theta}_{\mathcal{L}}) &= g_{\text{argmax}_i f_i(\underline{x})}(\underline{x}) \end{aligned}$$

where f_i is the probability estimation of cluster z_i of unsupervised model $\mathcal{A}$ and g_i is the classification estimation of the supervised model $\mathcal{L}$ of cluster z_i. However, we use the probability mixture prediction of supervised models as

$$\begin{aligned} p(y|\underline{x}, \underline{\theta}_{\mathcal{A}}, \underline{\theta}_{\mathcal{L}}) &= \sum_{i=1}^{M} p(y|\underline{x}, z = z_i, \underline{\theta}_{\mathcal{L}}) p(z_i|\underline{x}, \underline{\theta}_{\mathcal{A}}) \\ &= \sum_{i=1}^{M} f_i(\underline{x}) g_i(\underline{x}) \end{aligned}$$

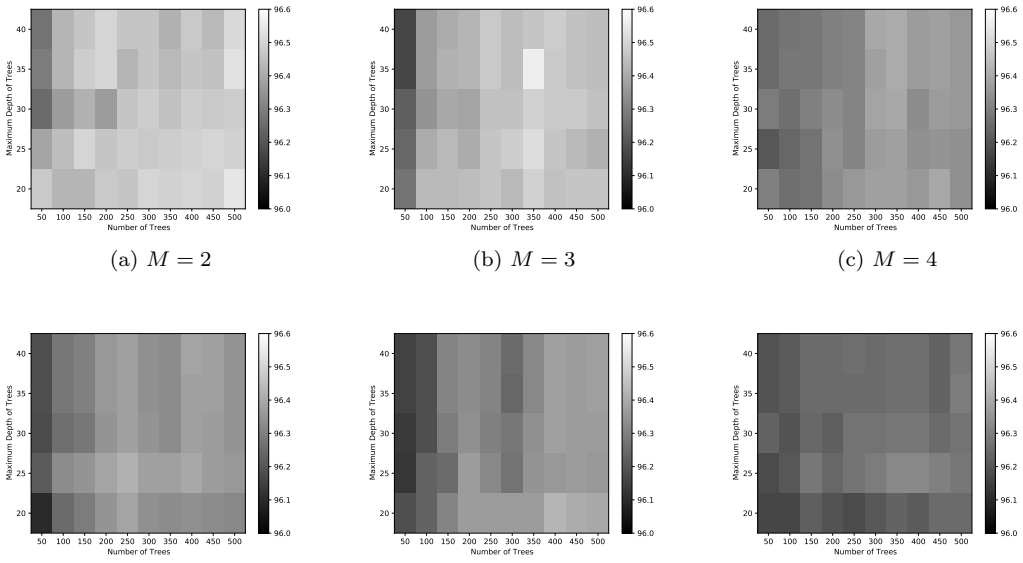
And the classification estimation is weighted sum of each supervised models in which the weight is the cluster probability $f_i(\underline{x})$. This cluster-then label algorithm is similar with a bootstrap aggregating (bagging) model since each supervised model is trained on in-cluster data samples, and the final estimation is based on the probability mixture of all supervised models instead of by voting. Another reason to do that is to prevent some data samples are misclassified, especially for those on or near the cluster boundaries.

The cluster algorithm used in the cluster-then-label is the expectation-maximization (EM) with Gaussian mixture model, and the classification algorithm in the cluster-then-label is random forest. Number of clusters M of Gaussian mixture model, number of trees N_{Tree} and maximum depth d_{max} of trees in random forest are searched by cross validation and the cross validation results are shown in Fig. 9. Parameters $M = 3$, $N_{Tree} = 300$ and $d_{max} = 35$ are selected for next step.

2.5 Model Selection and Comparison of Results

In Table 1, the optimal group 5-fold cross validation results of each SL and SSL classification model are given, where the accuracy is the average across cross-validation runs. For supervised learning, it turns out that logistic regression, random forest, Adaboost and k-NN have ideal cross validation accuracy of around 95% in comparison with the 78.26% accuracy of baseline model Naïve Bayes. and the best supervised learning model is random forest.

For semi-supervised learning models, propagating k-NN has higher cross validation accuracy than SL k-NN with only 0.01%, which is not as expected. On the other hand, cluster-then-label model has the highest average cross validation accuracy in both SL and SSL models. This indicates



(a) $M = 2$ (b) $M = 3$ (c) $M = 4$

(d) $M = 5$ (e) $M = 6$ (f) $M = 7$

Figure 9: Cross Validation Accuracy of Cluster-Then-Label

that SSL models training on both labeled and unlabeled data samples may have better classification performance.

Therefore, we select and move random forest and cluster-then-label classifier to the subsequent test step, and also Naïve Bayes for baseline comparison.

Table 1: Cross Validation Results of Classification Models

Model Type	**Model**	**CV Accuracy (%)**
Supervised	Naïve Bayes	78.26
	Logistic Regression	95.76
	Random Forest	**96.51**
	Adaboost	96.22
	k-NN	94.75
	Autovetter [4]	97.56
	Astronet [6]	96.00
Semi-Supervised	Propagating k-NN	94.76
	Cluster-Then-Label	**96.56**

3 Final Results, Interpretation

Now, two models, random forest and cluster-then-label classifier, are selected based on cross validation and along with baseline Naïve Bayes they will be retrained on whole training dataset then tested on the test dataset $\mathcal{D}_{Test}^{(l)}$. The final test results are shown in Table 2 where test metrics is precision, accuracy and area under the ROC curve (AUC). Also the ROC curves of all three models is shown in Fig. 10. These test-dataset-based results should be the unbiased estimation of out-of-sample performance as there are limited number of models (hypotheses) are selected from cross validation procedure.

As shown in Table 2, both random forest and cluster-then-label have much higher performance than the baseline Naïve Bayes. And the random forest shows the best performance on the test

Table 2: Test Results of Classification Models

Model Type	Model	Precision (%)	Accuracy (%)	AUC
Supervised	Naïve Bayes	51.01	77.47	0.8886
	Random Forest	**94.03**	**96.62**	**0.9925**
Semi-Supervised	Cluster-Then-Label	93.36	96.55	0.9918

dataset, which is not consistent with the cross validation results though it is reasonable. This may result from that there are about 69.3% labeled data samples in training dataset, and our SSL model may not show its strength on such high labeled-rate dataset. As a result, we will do more experiments to test how theses classifier perform when number of labeled data samples is decreased.

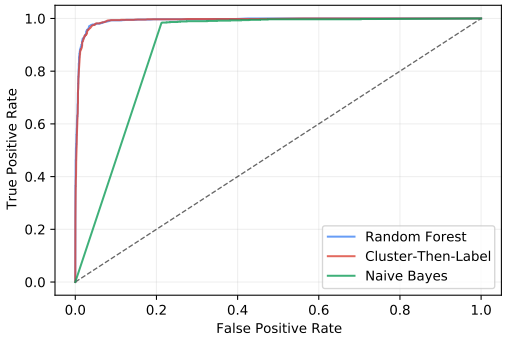


Figure 10: Test ROC Curves

For following part, labels of some labeled samples in the training dataset will be discarded and never used any more. After discarding label of different number of data samples, the models are trained and tested to get the results shown in Fig. 11. From that, it turns out that when the number of available labels reduced, the performance of models will also decrease. However, the size of labeled dataset has larger impact on SL models than SSL models, since the performance gap between cluster-then-label classifier and random forest become larger if less labels is available. That illustrates the robustness and key value of our SSL cluster-then-label classifier, particular for this problem where the cost to label a data sample is huge and the labeled dataset thus will not be large enough.

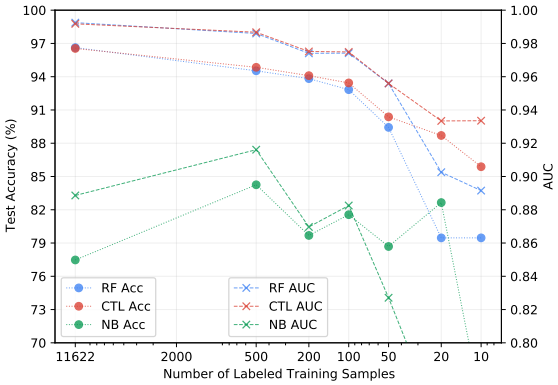


Figure 11: Test Performance over Different Size of Labeled Training Datasets

4 Summary and Conclusions

In the project, we solved the classification problem on Threshold-Crossing Events which can help us discover the exoplanets. Then several supervised and semi-supervised learning classifiers were introduced and implemented for the project, and then were tested upon *Kepler* TCE Table. The experimental results revealed that the random forest shows the highest accuracy of 94.03% as well as highest AUC of 0.9925 on the dataset, while our SSL cluster-then-label classifier has better performance than the SL random forest when less labeled training samples is available.

References

[1] W. J. Borucki, "Kepler mission: development and overview," *Reports on Progress in Physics*, vol. 79, no. 3, p. 036901, 2016.

[2] J. M. Jenkins, D. A. Caldwell, H. Chandrasekaran, J. D. Twicken, S. T. Bryson, E. V. Quintana, B. D. Clarke, J. Li, C. Allen, P. Tenenbaum, *et al.*, "Overview of the kepler science processing pipeline," *The Astrophysical Journal Letters*, vol. 713, no. 2, p. L87, 2010.

[3] S. Seader, J. M. Jenkins, P. Tenenbaum, J. D. Twicken, J. C. Smith, R. Morris, J. Catanzarite, B. D. Clarke, J. Li, M. T. Cote, *et al.*, "Detection of potential transit signals in 17 quarters of kepler mission data," *The Astrophysical Journal Supplement Series*, vol. 217, no. 1, p. 18, 2015.

[4] S. D. McCauliff, J. M. Jenkins, J. Catanzarite, C. J. Burke, J. L. Coughlin, J. D. Twicken, P. Tenenbaum, S. Seader, J. Li, and M. Cote, "Automatic classification of kepler planetary transit candidates," *The Astrophysical Journal*, vol. 806, no. 1, p. 6, 2015.

[5] D. J. Armstrong, D. Pollacco, and A. Santerne, "Transit shapes and self organising maps as a tool for ranking planetary candidates: Application to kepler and k2," *Monthly Notices of the Royal Astronomical Society*, p. stw2881, 2016.

[6] C. J. Shallue and A. 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, p. 94, 2018.

[7] M. Ansdell, Y. Ioannou, H. P. Osborn, M. Sasdelli, J. C. Smith, D. Caldwell, J. M. Jenkins, C. Räissi, D. Angerhausen, *et al.*, "Scientific domain knowledge improves exoplanet transit classification with deep learning," *The Astrophysical Journal Letters*, vol. 869, no. 1, p. L7, 2018.

[8] H. P. Osborn, M. Ansdell, Y. Ioannou, M. Sasdelli, D. Angerhausen, D. Caldwell, J. M. Jenkins, C. Räissi, and J. C. Smith, "Rapid classification of tess planet candidates with convolutional neural networks," *Astronomy & Astrophysics*, vol. 633, p. A53, 2020.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[10] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, *et al.*, "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[11] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[12] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.