Exoplanet Detection Using Transit Method

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

(Deemed to be University), Patiala – 147004

Machine Learning Project

Submitted By:
BE THIRD YEAR CO12

Chhavi Gupta

Roll No: 102103322

Angel

Roll No: 102103328

Submitted To:

Dr. Jyoti Maggu

Index

1. Introduction

Exoplanets are planets beyond our own solar system. Thousands have been discovered in the past two decades, mostly with NASA’s Kepler Space Telescope. These exoplanets come in a huge variety of sizes and orbits. Some are gigantic planets hugging close to their parent stars; others are icy, some rocky. NASA and other agencies are looking for a special kind of planet: one that’s the same size as Earth, orbiting a sun-like star in the habitable zone

The Kepler space telescope is a retired space telescope launched by NASA to discover Earth-size planets orbiting other stars. Designed to survey a portion of Earth's region of the Milky Way to discover Earth-size exoplanets in or near habitable zones and estimate how many of the billions of stars in the Milky Way have such planets, Kepler's sole scientific instrument is a photometer that continually monitored the brightness of approximately 150,000 main sequence stars in a fixed field of view. These data are transmitted to Earth, then analyzed to detect periodic dimming caused by exoplanets that cross in front of their host star. Only planets whose orbits are seen edge-on from Earth can be detected. During its over nine and a half years of service, Kepler observed 530,506 stars and detected 2,662 planets. Scientists discovered a very efficient way to study these occurrences; planets themselves do not emit light, but the stars around which they orbit do. Considering this fact into account scientists at NASA developed a method which they called Transit method in which a digital-camera-like technology is used to detect and measure tiny dips in a star’s brightness as a planet crosses in front of the star. With observations of transiting planets, astronomers can calculate the ratio of a planet’s radius to that of its star essentially the size of the planet’s shadow and with that ratio, they can calculate the planet’s size.

Kepler Space Telescope’s primary method of searching for planets was the “Transit” method which utlises the fact that the starlight intensity drops because it is partially obscured by the planet, given our position. The starlight rises back to its original value once the planets crosses in front of the star, leading the flux values.

The goal is to create a model that can predict the existence of an Exoplanet, utilizing the flux (light intensity) readings from 3198 different stars over time.

1.1. Kepler Dataset

Dataset Link

https://www.kaggle.com/datasets/keplersmachines/kepler-labelled-time-series-data

1.2 Description

- 5087 rows or observations.
- 3198 columns or features.
- Column 1 is the label vector. Columns 2 are the flux values over time.
- 37 confirmed exoplanet-stars and 5050 non-exoplanet-stars.

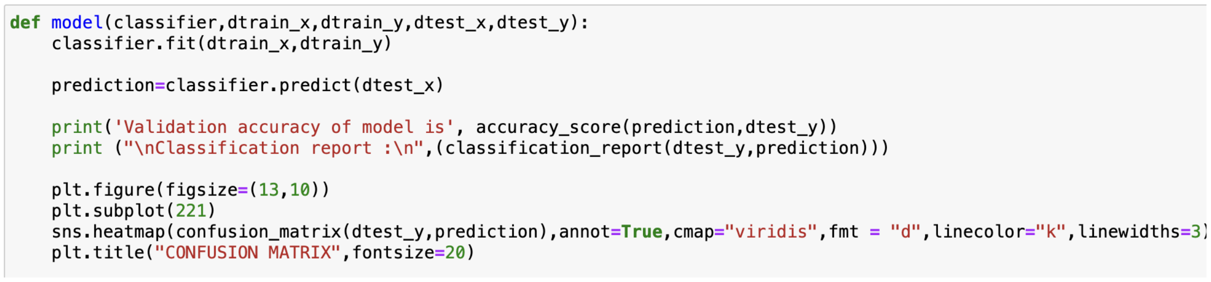
The data describe the change in flux (light intensity) of several thousand stars. Each star has a binary label of 2 or 1. 2 indicated that the star is confirmed to have at least one exoplanet in orbit; some observations are in fact multi-planet systems.

Planets themselves do not emit light, but the stars that they orbit do. If said star is observed over several months or years, there may be a regular 'dimming' of the flux (the light intensity), this is called 'transiting' and the absolute percentage of light being dimmed is astronomically small - on the order of a measuring the decrease in flux from a lighthouse when a fly transits, observed from many miles away. This flux dimming 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, for example by a satellite that captures light at a different wavelength, could solidify the belief that the candidate can in fact be 'confirmed' for having exoplanets.

2. Libraries Used

- math: Standard mathematical operations.
- warnings: Handling warning messages.
- numpy (np): Numerical operations and array manipulation.
- pandas (pd): Data manipulation and analysis for efficient handling of tabular data.
- matplotlib.pyplot as plt: Visualization of data and results.
- scipy.ndimage: Supporting image processing tasks.
- seaborn: Enhanced statistical graphics for data visualization.
- imblearn.over_sampling.SMOTE: Mitigating class imbalance through SMOTE.
- StandardScaler, normalize: Preprocessing for feature scaling and normalization.
- train_test_split: Partitioning datasets for model training and evaluation.
- cross_val_score: Cross-validated performance evaluation.
- recall_score, precision_score, classification_report, accuracy_score, confusion_matrix: Metrics for comprehensive model evaluation.

3. Algorithm(s) Used



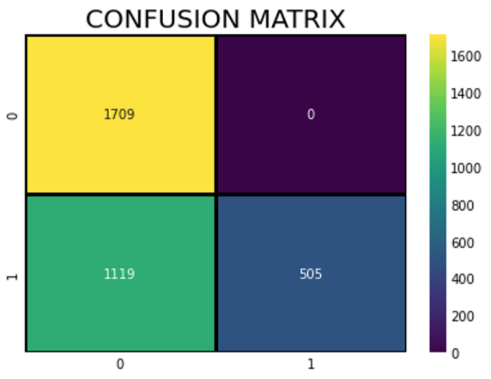
```
def model(classifier,dtrain_x,dtrain_y,dtest_x,dtest_y):
    classifier.fit(dtrain_x,dtrain_y)

    prediction=classifier.predict(dtest_x)

    print('Validation accuracy of model is', accuracy_score(prediction,dtest_y))
    print ("\nClassification report :\n",(classification_report(dtest_y,prediction)))

    plt.figure(figsize=(13,10))
    plt.subplot(221)
    sns.heatmap(confusion_matrix(dtest_y,prediction),annot=True,cmap="viridis",fmt = "d",linecolor="k",linewidths=3)
    plt.title("CONFUSION MATRIX",fontsize=20)
```

3.1. Support Vector Machine (SVM)

SVM is a supervised machine learning algorithm commonly used for classification and regression tasks. It is particularly well-suited for high-dimensional data, making it a valuable tool in various domains.

The fundamental principle behind SVM lies in finding a hyperplane that optimally separates two classes of data points. In the context of distinguishing stars with exoplanets from those without, SVM aims to establish a clear boundary between the two categories based on a set of stellar features, making it suitable for distinguishing stars with exoplanets from those without.

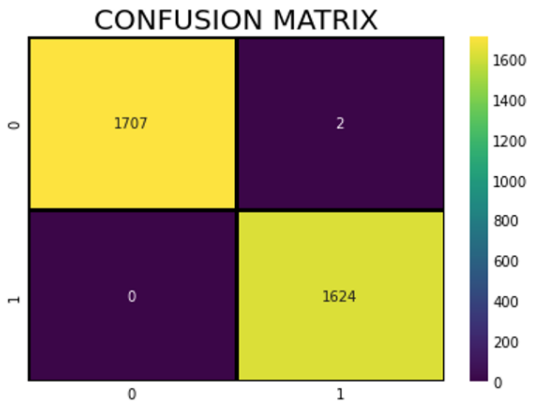
```
Validation accuracy of model is 0.6642664266426642

Classification report :
              precision    recall  f1-score   support

           0       0.60      1.00      0.75      1709
           1       1.00      0.31      0.47      1624

    accuracy                           0.66      3333
   macro avg       0.80      0.66      0.61      3333
weighted avg       0.80      0.66      0.62      3333
```



3.2. Random Forest Classifier

The Random Forest model is a versatile machine learning algorithm that has gained widespread popularity in various fields. Its ability to handle large datasets, capture complex relationships, and produce robust predictions makes it an ideal choice for identifying potential exoplanet candidates.Exoplanet research often involves analyzing vast amounts of data collected from telescopes and space missions. The Random Forest model is well-equipped to handle such large datasets efficiently. Its ensemble structure, composed of multiple decision trees, allows it to parallelize computations, significantly reducing training and prediction times.

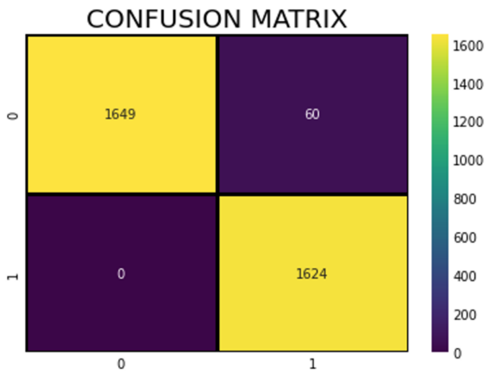
```
Validation accuracy of model is 0.9993999399939995

Classification report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1709
           1       1.00      1.00      1.00      1624

    accuracy                           1.00      3333
   macro avg       1.00      1.00      1.00      3333
weighted avg       1.00      1.00      1.00      3333
```



3.3. K Nearest Neighbours (KNN)

K-Nearest Neighbors (KNN) is a non-parametric machine learning algorithm that excels in classification tasks. Its simplicity, efficiency, and robustness make it a popular choice for various applications.Its classification process involves minimal calculations, making it suitable for real-time applications where speed is crucial.KNN's nonparametric nature means it does not make assumptions about the underlying data distribution. This flexibility allows it to adapt to various data shapes and patterns, making it well-suited for complex and unpredictable datasets like those encountered in exoplanet research.

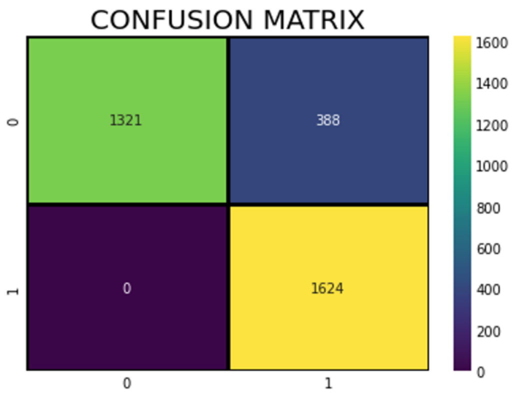
```
Validation accuracy of model is 0.981998199819982

Classification report :
              precision    recall  f1-score   support

           0       1.00      0.96      0.98      1709
           1       0.96      1.00      0.98      1624

    accuracy                           0.98      3333
   macro avg       0.98      0.98      0.98      3333
weighted avg       0.98      0.98      0.98      3333
```



3.4. Logistic Regression

Logistic Regression is a fundamental statistical learning algorithm used for binary classification tasks. In exoplanet exploration, it serves as a baseline model, providing a benchmark against which to compare the performance of more complex machine learning algorithms.Logistic Regression assumes a linear relationship between the input features and the target variable. This assumption allows it to not only predict the classification of new data points but also provide insights into the relative importance of each feature.

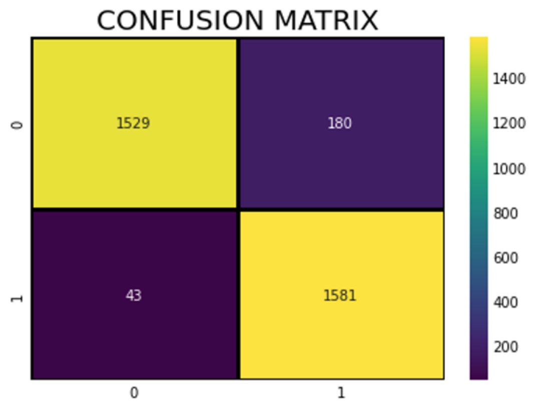
```
Validation accuracy of model is 0.8835883588358836

Classification report :
              precision    recall  f1-score   support

           0       1.00      0.77      0.87      1709
           1       0.81      1.00      0.89      1624

    accuracy                           0.88      3333
   macro avg       0.90      0.89      0.88      3333
weighted avg       0.91      0.88      0.88      3333
```



3.5. Decision Tree Algorithm

Decision trees are non-parametric supervised machine learning algorithms that excel in classification and regression tasks. They represent the decision-making process in the form of a tree-like structure, where each internal node represents a feature-based decision, and each leaf node represents a class label or a continuous prediction value.Decision trees are particularly well-suited for modeling complex decision boundaries, as they can effectively capture intricate relationships between features without making assumptions about the underlying data distribution.

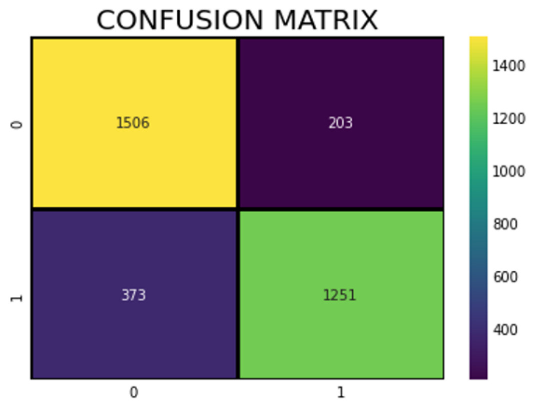
```
Validation accuracy of model is 0.933093309330933

Classification report :
              precision    recall  f1-score   support

           0       0.97      0.89      0.93      1709
           1       0.90      0.97      0.93      1624

    accuracy                           0.93      3333
   macro avg       0.94      0.93      0.93      3333
weighted avg       0.94      0.93      0.93      3333
```



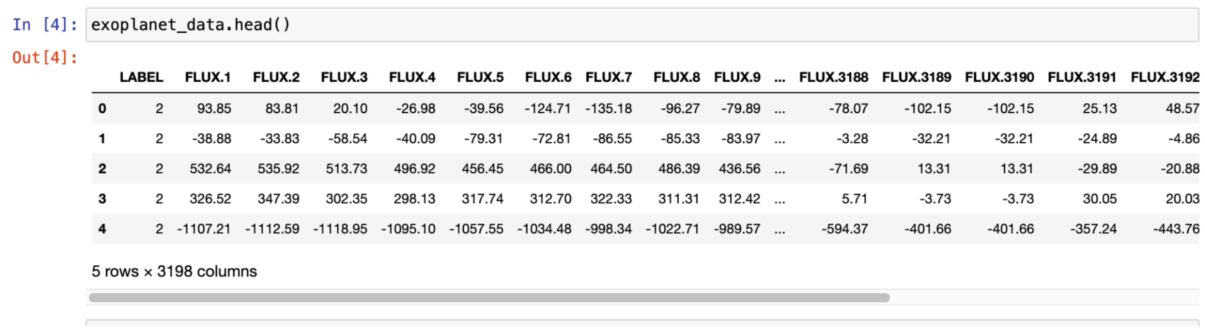
3.6. Bernoulli Naive Bayes

The Bernoulli Naive Bayes model is a probabilistic machine learning algorithm specifically designed for binary features. It is based on Bayes' theorem and assumes that the features are independent of each other, making it computationally efficient and well-suited for large datasets.The Bernoulli Naive Bayes model's efficiency stems from its utilization of binary probabilities. For each feature, the model calculates the probability of it being present or absent given the presence or absence of exoplanets. This simplification significantly reduces the computational overhead compared to other models that handle continuous or multi-value features.

```
Validation accuracy of model is 0.8271827182718272

Classification report :
              precision    recall  f1-score   support

           0       0.80      0.88      0.84      1709
           1       0.86      0.77      0.81      1624

    accuracy                           0.83      3333
   macro avg       0.83      0.83      0.83      3333
weighted avg       0.83      0.83      0.83      3333
```



4. Data preprocessing



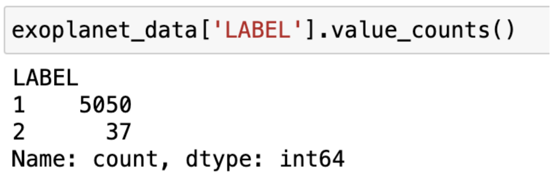
```
In [4]: exoplanet_data.head()
Out[4]:
```

	LABEL	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLUX.9	...	FLUX.3188	FLUX.3189	FLUX.3190	FLUX.3191	FLUX.3192
0	2	93.85	83.81	20.10	-26.98	-39.56	-124.71	-135.18	-96.27	-79.89	...	-78.07	-102.15	-102.15	25.13	48.57
1	2	-38.88	-33.83	-58.54	-40.09	-79.31	-72.81	-86.55	-85.33	-83.97	...	-3.28	-32.21	-32.21	-24.89	-4.86
2	2	532.64	535.92	513.73	496.92	456.45	466.00	464.50	486.39	436.56	...	-71.69	13.31	13.31	-29.89	-20.88
3	2	326.52	347.39	302.35	298.13	317.74	312.70	322.33	311.31	312.42	...	5.71	-3.73	-3.73	30.05	20.03
4	2	-1107.21	-1112.59	-1118.95	-1095.10	-1057.55	-1034.48	-998.34	-1022.71	-989.57	...	-594.37	-401.66	-401.66	-357.24	-443.76

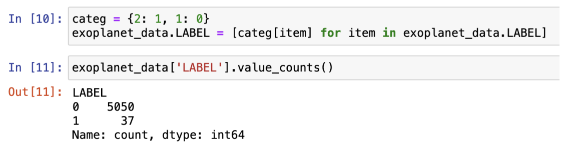
5 rows × 3198 columns

4.1 Label Encoding

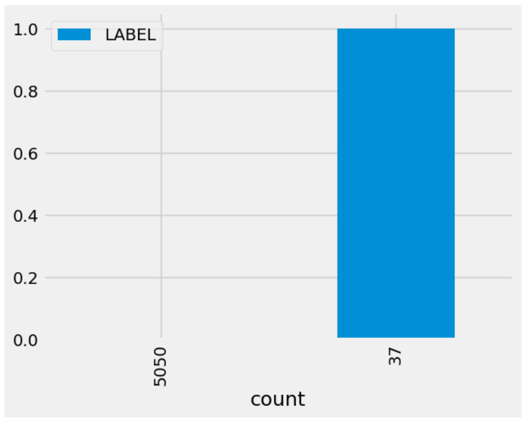
The target column LABEL consists of two categories 1(does not represent exoplanet) and 2 (represents the presence of exoplanet). So we convert them to binary values for easier processing of data.



```
exoplanet_data['LABEL'].value_counts()

LABEL
1    5050
2      37
Name: count, dtype: int64
```



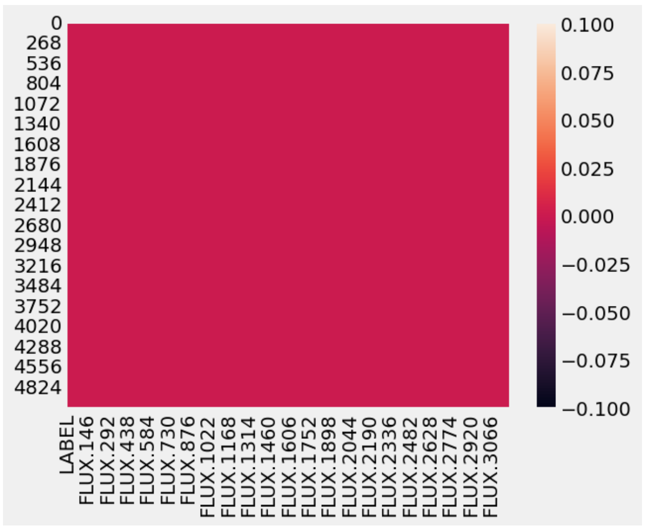
```
In [10]: categ = {2: 1, 1: 0}
         exoplanet_data.LABEL = [categ[item] for item in exoplanet_data.LABEL]

In [11]: exoplanet_data['LABEL'].value_counts()
Out[11]: LABEL
         0    5050
         1      37
         Name: count, dtype: int64
```



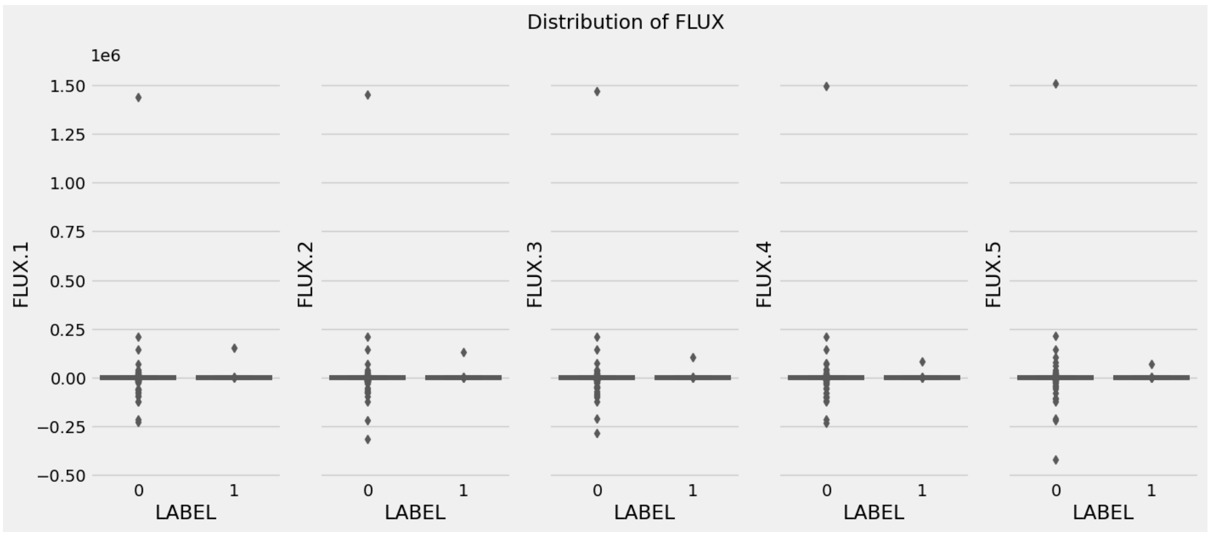
It turns out that the data is highly imbalanced. So first let us start with data preprocessing techniques.

4.2 Handling Missing Values

We can see from the heat map that we don't have any missing values in our dataset



4.3 Outlier Detection and Removal

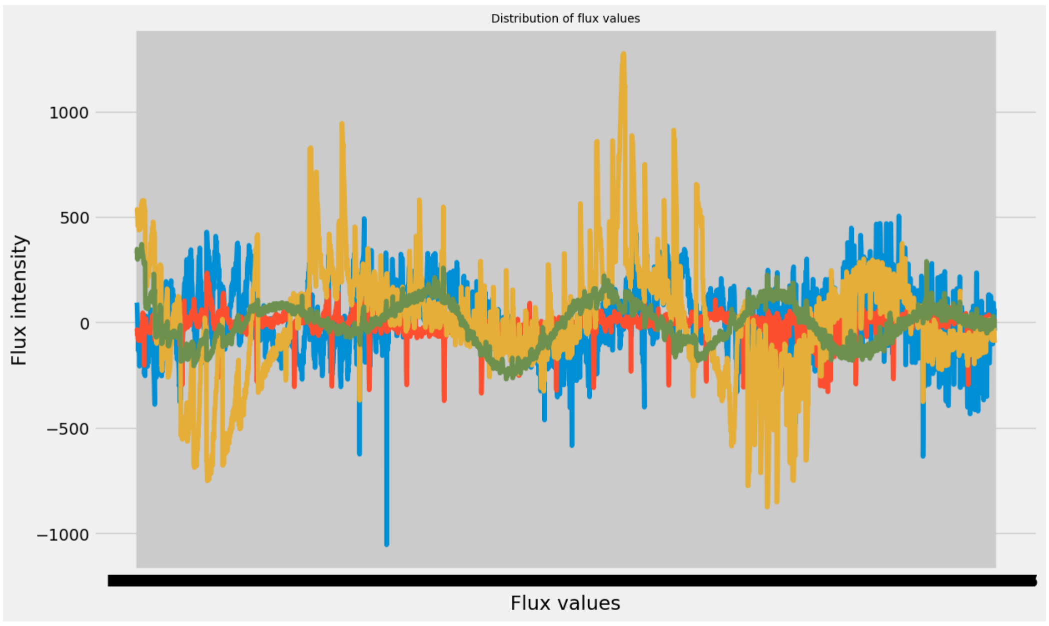


```
In [15]: exoplanet_data.drop(exoplanet_data[exoplanet_data['FLUX.1']>250000].index, axis=0, inplace=True)
```

We drop columns from the DataFrame where the value in the flux column is greater than 250,000. This is a common practice in data preprocessing to handle outliers or extreme values that might negatively impact the analysis or modeling process.

4.4 Data Normalization

Data Normalization is a technique often applied as part of data preparation for machine learning. 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. This step ensures that each feature contributes proportionally to the model training process.



Since the data is not normalized, we normalize the flux values.

Splitting the dataset into Input Features and Output Feature

```
x_train = exoplanet_data.drop(["LABEL"],axis=1)
y_train = exoplanet_data["LABEL"]
```

Data Normalization

```
x_train = normalized = normalize(x_train)
```

Guassian Filters

```
x_train = filtered = ndimage.filters.gaussian_filter(x_train, sigma=10)
```

Feature Scaling

```
std_scaler = StandardScaler()
x_train = scaled = std_scaler.fit_transform(x_train)
```

4.5 Laplace-Gaussian Filter

The Laplace-Gaussian filter is applied to the data to enhance the robustness of the observations. This filter aids in smoothing the dataset, making it conducive for the subsequent machine learning models.

In probability theory, the normal (or Gaussian or Gauss or Laplace–Gauss) distribution is a very common continuous probability distribution. Normal distributions are important in statistics and are often used in the natural and social sciences to represent real-valued random variables whose distributions are not known.

Feature Scaling is done so that all the values remain in the comparable range.

4.6 Principal Component Analysis (PCA)

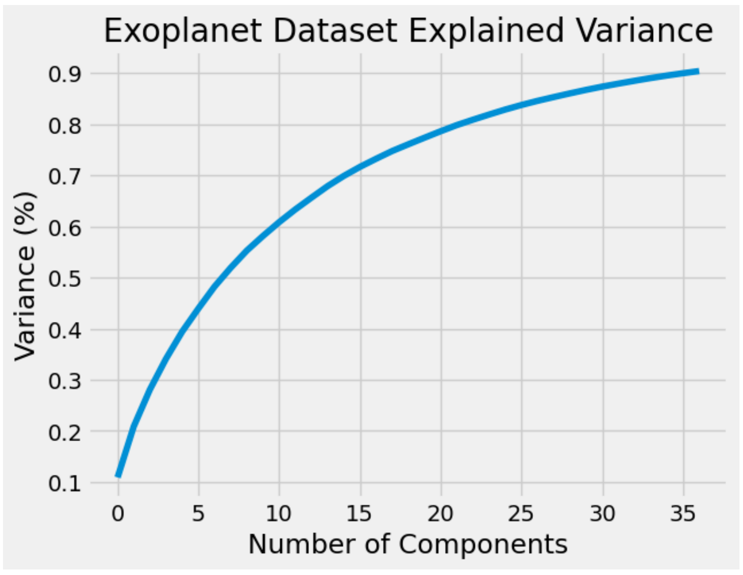
To address the 'curse of dimensionality' in a high-dimensional dataset (3198 columns), PCA is employed. This reduction technique preserves essential information while reducing the number of features, enhancing computational efficiency and mitigating potential overfitting.

To perform PCA we have to choose the number of features/dimensions that we want in our data. With the value k=37

```
from sklearn.decomposition import PCA
pca = PCA()
x_train = pca.fit_transform(x_train)
x_train = pca.transform(x_train)
total=sum(pca.explained_variance_)
k=0
current_variance=0
while current_variance/total < 0.90:
    current_variance += pca.explained_variance_[k]
    k=k+1
print(k)
```

```
37
```

```
pca = PCA(n_components=37)
x_train = pca.fit_transform(x_train)
plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of Components')
plt.ylabel('Variance (%)')
plt.title('Exoplanet Dataset Explained Variance')
plt.show()
```



The above plot tells us that selecting 37 components we can preserve something around 98.8% or 99% of the total variance of the data. It makes sense, we’ll not use 100% of our variance, because it denotes all components, and we want only the principal ones.

```
In [22]: x_train.shape
Out[22]: (5086, 37)
```

4.7 SMOTE Resampling Method

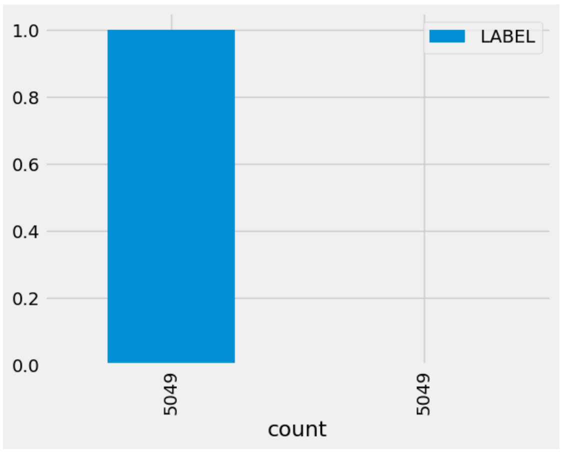
The target class is not equally distributed and one class dominates the other. So we need to resample our data so that the target class is equally distributed.

To tackle class this imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) is utilized. This method generates synthetic samples of the minority class, ensuring a more balanced representation and preventing bias in model training.

It creates synthetic (not duplicate) samples of the minority class. Hence making the minority class equal to the majority class. SMOTE does this by selecting similar records and altering that record one column at a time by a random amount within the difference to the neighboring records.

```
from imblearn.over_sampling import SMOTE
model = SMOTE()
ov_train_x,ov_train_y = model.fit_resample(exoplanet_data.drop('LABEL',axis=1), exoplanet_data['LABEL'])
ov_train_y = ov_train_y.astype('int')

ov_train_y.value_counts().reset_index().plot(kind='bar', x='count', y='LABEL')
```

```
<Axes: xlabel='count'>
```



5. Conclusion and Future Scope

The integration of advanced data preprocessing techniques and diverse machine learning models demonstrates a comprehensive approach to exoplanet detection. The combination of dimensionality reduction, normalization, and resampling methods contributes to the robustness of the models. The results from the ensemble of models provide a multifaceted understanding of the dataset, advancing our capabilities in the automated identification of exoplanets using the transit method and machine learning.

Here are some potential future improvements and directions for enhancement:

- Implement various other resampling techniques to address class imbalance problems.
- Advanced Dimensionality Reduction: Investigate more sophisticated dimensionality reduction techniques beyond PCA. Non-linear methods like autoencoders may capture complex relationships in the data more effectively.
- Hyperparameter Tuning: Conduct an extensive search for optimal hyperparameters for each machine learning model. Utilize techniques such as grid search or random search to find the best configuration, which can significantly impact model performance.
- Deep Learning Architectures: Investigate the use of more sophisticated deep learning architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), to capture intricate patterns in time-series data more effectively.
- Dynamic Learning: Implement models that can adapt to changes in the dataset over time. This could involve using online learning techniques or recurrent neural networks that can learn evolving patterns in the flux data.
- Uncertainty Estimation: Develop models that provide uncertainty estimates for their predictions. This is crucial in astrophysical applications where uncertainty in the data and observations is inherent.
- Real-time Detection: Aim for real-time or near-real-time detection capabilities, especially for applications where continuous monitoring of stars is possible. This could involve optimizing model architectures for speed and efficiency.

5. References

https://ijrpr.com/uploads/V3ISSUE2/ijrpr2746-detection-of-exoplanets-using-machine-learning.pdf

https://academic.oup.com/mnras/article-abstract/513/4/5505/6472249?redirectedFrom=fulltext

https://newsroom.usra.edu/discovery-of-69-new-exoplanets-using-machine-learning/

https://www.researchgate.net/publication/346510487_Exoplanet_Detection_using_Machine_Learning

https://www.sciencedirect.com/science/article/pii/S1877050922008535