Exoplanet Orbital Semi-Major Axis Prediction: A Regression Approach

Name:	**Arihant Tiwari**
Registration No./Roll No.:	19050
Institute/University Name:	IISER Bhopal
Program/Stream:	BS-MS (Physics)
Problem Release date:	February 02, 2022
Date of Submission:	April 24, 2022

Introduction

Since the detection of the first exoplanet in 1995, the quest for habitable exoplanets has become a pinnacle of research in Astronomy. An Exoplanet is a planet outside our solar system. Knowledge of orbital properties helps significantly in classifying the exoplanets as habitable planets. The objective of the project was to use machine learning to prepare a regression model that can predict the semi-major axis of exoplanet orbit using the data provided by the NASA Exoplanet Archive. Although several methods are deployed to predict the candidate as an exoplanet, there is little to no work done towards dealing with Semi-Major Axis prediction. This paper contains various methods and techniques to extract the relevant parameteres from the Kepler Dataset and use them to classify a candidate as exoplanet. It also discusses methods for data cleaning and dataset pre-processing, which was an essential step in this project as the dataset provided was very messy in terms of missing values in the feature vectors, data instances containing non-relevant entries etc. Hence this project can be divided into two major sections, data cleaning and model training.

The dataset for the project can be found here. The description of the dataset is given below:

- **exoplanet_trn_data.csv:** This file contains training feature vectors. It has 17969 rows (data points) and 288 columns (features).
- **exoplanet_trn_data_targets.csv:** This file contains training target values (semi major axis) . It has 17969 rows (data points) and 2 columns (index and semi major axis).
- **exoplanet_tst_data.csv:** This file contains test feature vectors. It has 1997 rows (data points) and 288 columns (features).
- **exoplanet_data_desc.txt:** This is a text file that contains the description of all 288 features available in our dataset.

1 Methods

The project's proceeding can be divided into two sections, Data Processing and Model Execution. The various methods used in each of these sections are extensively discussed below:

1.1 Data Processing

There were four data files as discussed in the section above. Three of them were imported to form three databases (using the Pandas module for Python), called *train_data*, which contained the training data,

train_labels, which contained the labels for each of the instances in the training dataset, and *test_data* which contained no labels and was a relatively shorted data as compared to the training data. The models were supposed to predict the labels for the *test_data*.
The training data contains, 17969 instances, whereas the test data has 1997 instances with a total of 288 features each.
A thorough look at the training dataset revealed that the dataset contained a lot of features that had no values for most of the instances. According to this article, any feature which contained more than 90% of the total instance values as missing should be removed as it contributes nothing to the model training. Hence we dropped such features, to improve the working of our models.

1.1.1 Missing Data Feature Removal

1. Calculate the total number of null values for each feature in the dataset.
2. Arrange the features in descending order of number of null values present in them.
3. Select the top features that have more null values than the decided threshold.
4. Remove these features from the dataset and create a new dataset named *tr05*.

Since some of the instances in few 'numerical' features contained non-numeric values, it lead to their classification as a Categorical Column, which is actually incorrect. This is resolved in the following section.

1.1.2 Wrongly Classified Column Correction

1. Create a list of 'Object Type' features in the dataset.
2. Find the total number of such columns. (Found to be 37)
3. Use domain expertise to manually identify such column and make a list of wrong classified columns.
4. Convert the datatype of the above identified columns as floats. (The datatype to which they actually should be belonging)

1.1.3 Conversion of Categorical Data to Numerical for Model Run

Since there were categorical features that contained the class name as strings, which could not be passed down a regression model, we converted such unique 'string' values into numerical value classifiers.

1. Create a list of categorical features.
2. Assign a numerical value to each of the unique class in the categorical feature.

1.1.4 Filling the Missing Data

As we can observe that there are a lot of 'NaN' values in the columns, we will replace these values with the median of the data as it tends to be a more appropriate choice here, because the range of data is huge, and the extremities are separated by a large gap.

1. Calculate the Median of each feature vector.
2. Find the null values in each of the feature vector and replace it with the respective median.
3. Check if all the feature columns are free of Nan values.
4. Remove such instances when features contain non-numerical values for numerical features.

1.1.5 Outlier Removal

Several Columns in the dataset tend to contain contrasting data values, ranging from Mico range to Mega. Such difference in the dataset in way more than the acceptable range of the standard variance. Hence we removed the outliers in a hope to increase the correlation.
Remove all the instances that lie beyond the following threshold.

$$Outlier_{value} >= \mu \pm (10 \times \sigma)$$

Where μ is the mean of the data, and σ is the variance of the dataset.
Some feature vectors become null after this operation hence we remove them as well.

1.1.6 Correlation Calculation

The Correlation was calculated for each of the feature in the training set and a C-map was produced to visualise the features having maximum correlation with the target variable.
The *Planet_Orbital Period, and its errors* were found to have maximum correlation with the target variable.

1.1.7 Feature Engineering

Applying Kepler's third law, we applied certain mathematical transformations on the orbital period to form two new feature vectors which were included in the training dataset in order to increase training quality.
The correlation with these new features was also found to be substantially high.

1.2 Modelling

The following models were performed on the training set and the the corresponding **RMS values** were calculated for each of the training models. The models run on the dataset were,

1. Linear Regression
2. Decision Tree
3. Support Vector Regression
4. Random Forest Regression

The random forest regression was found to provide the minimum error hence it was utilised to perform the final prediction on the test dataset.
The final values of the target variable as predicted by the **Random Forest** were saved into a text file named, **predicted_semi_major_axis.txt**.

2 Experimental Analysis

Mean square error (MSE) is the average of the square of the errors. The lesser the error, the better the model.

The *mean_squared_error* for different models after hyperparameter tuning are as follows:

- Linear Regression: 0.0617095
- Decision Tree Regression: 0.0160685
- Support Vector Regression: 0.0105230
- Random Forest Regression: 0.0078295

R^2 score represents the proportion of variance (of y) that has been explained by the independent variables in the model. It ranges between 0 and 1 and higher the score, better is the regression model.

The *r2_score* for different models after hyperparameter tuning are as follows:

- Linear Regression: 0.9096385
- Decision Tree Regression: 0.9764708
- Support Vector Regression: 0.9845910
- Random Forest Regression: 0.9885352

The lowest value of mean-squared error is found to be for Random Forest Regression. Hence, this is our best model. Also, we can see that Random Forest Regression is giving the highest *r2_score* as well, which supports our decision of considering it the best model.

Discussions

The proposed method is capable of handling all types of missing data, outliers and wrongly classified categorical data. The research data is not always clean for use. This method is self-sufficient to clean any new research data and can make predictions of the semi-major axis for the exoplanets. The method is not dependent on any prior domain knowledge, it learns by itself the important features that can be used for regression problems. There is no bias to any single feature in our model. The feature selection is done on the basis of the correlation matrix. This way, even a person who is not from the Astrophysics domain can use this method and get insights into the Exoplanet data.

The main limitation of our method is that the time complexity is high. The parameter tuning of different models on the large Exoplanet dataset takes a lot of computational power. Also, the parameters chosen for the hyperparameter tuning are hand-picked. This approach can be automized. We have done feature selection on the basis of only correlation matrix. We planned to try on with some other metrics like p-value for feature selection. We also planned on using ensemble techniques where we can use various regression models and combine them using boosting techniques to come up with a better regression model than the individual regression models. Finally, if we get our hands on the eccentricity of the exoplanets and habitable zone of star, we can use these semi-major axis values to detect whether an exoplanet is habitable or not. This will open a completely new horizon for this project, where it can be used for finding our new home!

On the idea of taking our project further, there will be new exoplanet data released by the survey of Transiting Exoplanets Survey Satellite (TESS) in some time. We are planning on making the changes and modifications stated above and will then run the model on the latest TESS dataset.