VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component report

Programme : B.Tech

Course Title : Essentials of Data Analytics

Course Code : CSE3506

Slot : G1/G2

Title :Exoplanet Exploration using Data Analytics

Team Members:

S.no	Name	Register Number
1.	Disha Gupta	20BPS1078
2.	Preyash	20BPS1022
3.	Pranav Vishwanathan	20BPS1106
4.	Priyanshu Singh	20BCE1397

Faculty: Dr. Vergin M

Sign: [illegible]

Date: 6/4/2023

1. Abstract:

Exoplanet Exploration is the study of planets outside of our solar system. Advanced statistical and machine learning techniques are used to examine the data gathered during exoplanet exploration in order to find patterns and relationships. populations of exoplanets. The process of precisely identifying and describing exoplanets remains difficult despite the advancements made in this science. This occurs as a result of exoplanets' weak emissions and their near proximity to blazing stars.

In this project we are using Kepler Mission Data for analysis. It consists of orbital periods, impact paramter, transit duration, radius, temperature, insolation flux and gravity like parameters. The data will be processed and cleaned before implementing the classification models. The methodology includes the models like : Decision Tree, Naive Bayes, Logistic Regression, Extra Tree classifier, KNN, Random Forest, XGB classifier. The accuracy and classificaiton report for each model will be generated to conclude the best algorithm to determine the potential candidate for like earth like planet.

Keywords: *Exoplanets, Machine Learning, Classification Algorithm, Kepler Mission Data*

2. Introduction:

Over 4,000 exoplanets have been confirmed as of 2021, with a sizable number of candidates currently awaiting approval. The most popular approaches for finding exoplanets are the transit and radial velocity methods. However, additional methods including astrometry, microlensing, and direct imaging are also used to collect data. The process of precisely identifying and describing exoplanets remains difficult despite the advancements made in this science. This occurs as a result of exoplanets' weak emissions and their near proximity to blazing stars. Since they have significant consequences for our understanding of the cosmos and the possibility of life beyond our solar system, accurate detection and characterization are essential components of this project. There is always an ongoing quest for the exoplanet which is capable of having life. Understanding the data captured from satellites and telescope is the most crucial part. How the parameters affect the conclusion should be thoroughly studied. Therefore, our problem statement is based on the analysis of the explanets dataset and studying the features involved that affects the chances of life present on the planet. We know that presence of basic elements is necessary for the life to sustain. These elements include oxygen, hydrogen, nitrogen, etc. Only the presence of this elements are not enough for life, but the proper atmospheric conditions, temperature and other physical factors are also essential for life to exists.

There are many classification algorithms using which we can do the analysis but our aim should be to come up the algorithm which is accurate and not very complex. There are some specific algorithms made only for the purpose of the exoplanet exploration. These include deep learning model ASTRONET. This algorithm predicts the habitability of the exoplanet. In this project we are implementing different classification algorithms and comparing their results with the numerical data obtained as well as by plotting their graphs such as heatmaps and ROC curves. The Kepler Mission data is used and the target variable is classified into two values making it a binary classification approach. The features which are higle correlated and those have no significance in the result are removed. The dataset will be examined based on the output results in order to derive useful insights. We will be able to assess the possibility of life existing on suitable exoplanets, evaluate the efficacy of various models for the provided dataset, and forecast the discovery of new exoplanets in the future. The methodology includes the models like : Decision Tree, Naive Bayes, Logistic Regression, Extra Tree classifier, KNN, Random Forest, XGB classifier.The data is split into the training and testing values. Before training the dataset the data is preprocessed using R and the models are trained and tested in python. The validation accuracy for Extra Tree Classifier is found to be maximum i.e. 95.30% while the average accuracy of Random Forest is maximum 95.19%.

3. Literature Survey:

[1] Habitability of Exoplanets using Deep Learning

This paper proposes a deep learning model ASTRONET to predict the habitability of Exoplanets. In this era, the space exploration has become a vast and important topic and every aerospace company is in the race of searching the bodies which are preferable for life. The parameters used in this paper are gravity, mass, radius, eccentricity, orbital stability, temperature and many other attributes. This papers states that ASTRONET model would examine the bid data of interplanetary objects and we will be able to recognise the abnormalities.

[2] Deep learning exoplanets detection by combining real and synthetic data

This paper works on the data collected by Kepler space telescope to study the light curves. The paper approaches a convolutional neural network(CNN) to study the light curves. With the aid of the feature extraction technique 2D phase folding, real and artificial light-curves with transit can be described by images that can be distinguished from those without transit.The primary contribution of this study is the improvement of a detection model that produces synthetic light-curves with transit using parameters that are calculated.

[3] Detecting Exoplanet Transits through Machine Learning Techniques with Convolutional Neural Networks

In this study, five different types of deep learning models with and without foldings were constructed and studied. With the help of a synthetic light curve dataset, we assessed how well the CNN models performed. This dataset offered us a thorough analysis of the CNN models' performance under various circumstances and included light curves with various noise levels and transit depths. The results demonstrated that when compared to conventional machine learning methods, CNN models outperformed them in terms of accuracy and computing efficiency.

As a result of our research, it is possible that CNNs may considerably increase the precision of exoplanetary transit detection by 20% and aid in the search for new exoplanets.

[4] Scientific Domain Knowledge Improves Exoplanet Transit Classification with Deep Learning

This study examines the effects of adding scientific domain knowledge into deep learning systems for classifying exoplanet transit. To examine both simulated and actual planetary transit data, the study uses a deep neural network model that was trained on synthetic light curves.The study has implemented data augmentation techniques to alleviate model over-fitting.This methodology reduces the size of the model, while still maintaining improved performance. The outcomes demonstrate that including scientific domain knowledge considerably improves the accuracy, precision, and recall of the deep learning system.

[5] Exoplanet detection capability of the COROT space mission

This paper, we are concerned with the instrumentation,data analysis and expected performances of COROT in terms of exoplanet detection. the first stepin the data processing will be to high-pass filter the lightcurves with a cut-off frequency of say 1/(4 τmax). The light curves are then averaged on a trial transit duration τ to increase the S/N, and cross-correlated with a transit-like signalat a trial period P. Crosscorrelation products C(τ, P, φ) have to be computed for enoughtrial triplets to correctly explore the parameter space. We find that our requirement of less than one false detection for the entire mission is met for $\beta = 7$, if a Gaussian statistics is assumed for C.

[6] Machine-learning approaches to exoplanet transit detection and candidate validation in wide-fifield ground-based surveys

The paper completely focuses on WASP data. The project was designed using existing commercial components to reduce costs, so each location is made up of eight commercial cameras mounted together and using science-grade CCDs for imaging.While the WASP data alone is not of sufficient quality to definitively identify planets from the data, it has proven to be very effective in producing new candidates for future follow-up and eventual planet status. The large size of the WASP archive makes it undesirable for human observers to manually look at each one to determine whether it is a good candidate for further study.

[7] Identifying Exoplanets With Deep Learning: A Five Planet Resonant Chain Around Kepler-80 And An Eighth Planet Around Kepler-90

NASA's Kepler Space Telescope was designed to tally the number of Earth-sized planets orbiting Sun-like stars, but the mission's discovery sensitivity is extremely low for these planets. Classifying the possible exoplanet transmissions discovered by NASA's Kepler Space Telescope required a deep learning approach using a deep convolutional neural network (CNN). The technique was developed to sort through potential transiting exoplanet signals from those that could have been caused by astrophysical or instrumental artefacts. To this end, the deep learning model was taught, and it performed admirably, with 98.8% of plausible planet signals being ranked higher than false positive signals in the test set.

[8] Unsupervised Machine Learning for Exploratory Data Analysis of Exoplanet Transmission Spectra

This paper focuses on unsupervised techniques for analyzing spectral data from transiting exoplanets to decode their chemical compositions. The study uses infrared spectroscopy of planetary atmospheres during transit events to determine their chemical composition.Anomaly detection is an unsupervised machine learning task where the goal is to identify unusual instances in data by learning what is considered "normal". The results showed that there is a high degree of correlation in the spectral data and that a three-dimensional representation captures the correlations better.

4. Proposed Work:

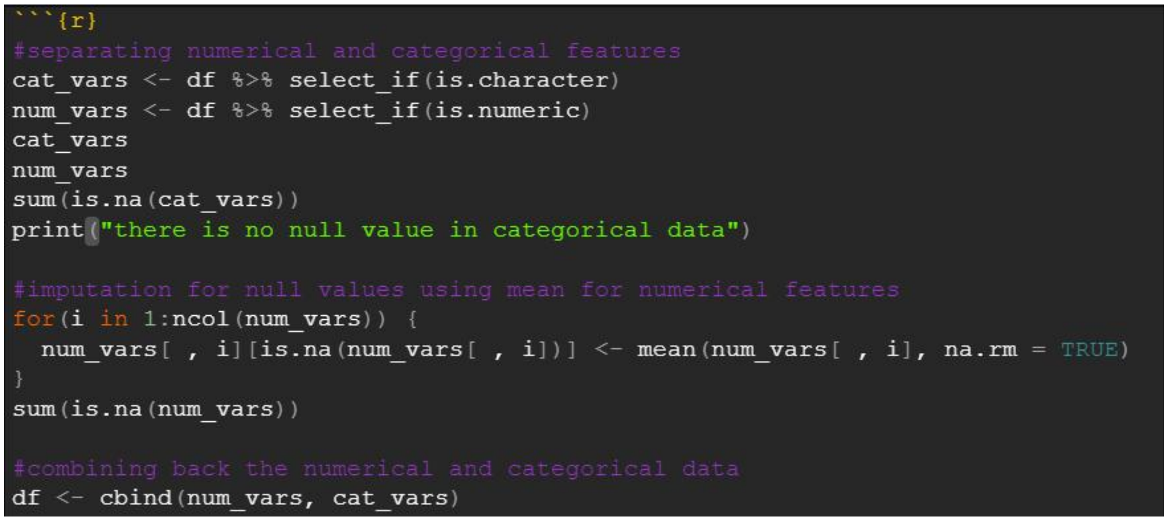
4.1 Data Source

The data source is Kepler Mission Data. The consists of the parameter like

- orbital periods
- impact paramter
- transit duration
- radius
- temperature,
- insolation flux
- gravity

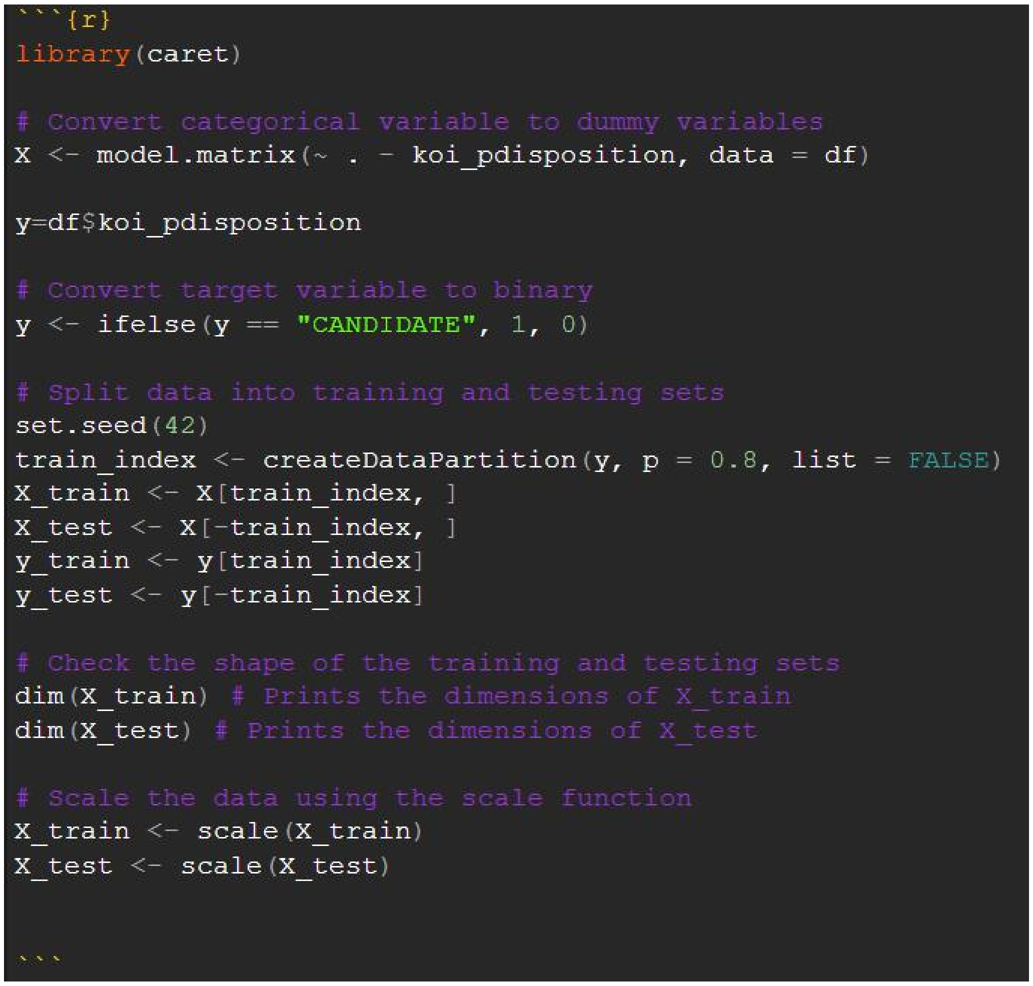
The data also consists of other parameters which do not contribute for prediction. The raw data is unfit for training models as it comprises null values. The data after cleaning consists of 10533 rows and 38 features.Datapreprocessing includes the following code:

R Code for removing Null values:

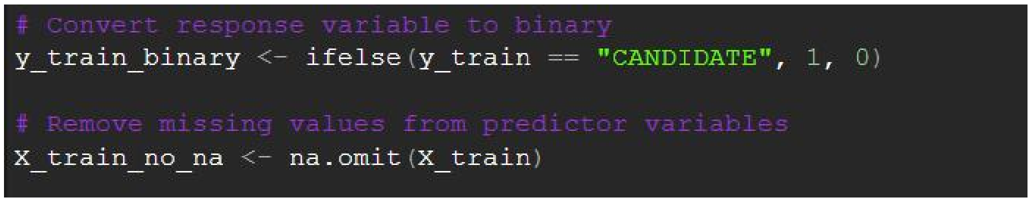


~~~
```{r}
#separating numerical and categorical features
cat_vars <- df %>% select_if(is.character)
num_vars <- df %>% select_if(is.numeric)
cat_vars
num_vars
sum(is.na(cat_vars))
print("there is no null value in categorical data")

#imputation for null values using mean for numerical features
for(i in 1:ncol(num_vars)) {
  num_vars[ , i][is.na(num_vars[ , i])] <- mean(num_vars[ , i], na.rm = TRUE)
}
sum(is.na(num_vars))

#combining back the numerical and categorical data
df <- cbind(num_vars, cat_vars)
~~~

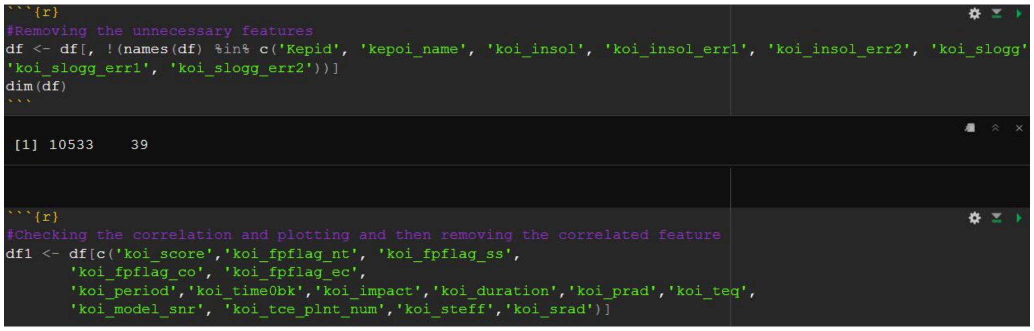
R Code for splitting the dataset and making binary classification dataset and scaling the dataset after spliting



~~~r
```{r}
library(caret)

# Convert categorical variable to dummy variables
X <- model.matrix(~ . - koi_pdisposition, data = df)

y=df$koi_pdisposition

# Convert target variable to binary
y <- ifelse(y == "CANDIDATE", 1, 0)

# Split data into training and testing sets
set.seed(42)
train_index <- createDataPartition(y, p = 0.8, list = FALSE)
X_train <- X[train_index, ]
X_test <- X[-train_index, ]
y_train <- y[train_index]
y_test <- y[-train_index]

# Check the shape of the training and testing sets
dim(X_train) # Prints the dimensions of X_train
dim(X_test) # Prints the dimensions of X_test

# Scale the data using the scale function
X_train <- scale(X_train)
X_test <- scale(X_test)


```
~~~



```r
# Convert response variable to binary
y_train_binary <- ifelse(y_train == "CANDIDATE", 1, 0)

# Remove missing values from predictor variables
X_train_no_na <- na.omit(X_train)
```



~~~r
```{r}
#Removing the unnecessary features
df <- df[, !(names(df) %in% c('Kepid', 'kepoi_name', 'koi_insol', 'koi_insol_err1', 'koi_insol_err2', 'koi_slogg'
'koi_slogg_err1', 'koi_slogg_err2'))]
dim(df)
```
~~~

```
[1] 10533    39
```

~~~r
```{r}
#Checking the correlation and plotting and then removing the correlated feature
df1 <- df[c('koi_score','koi_fpflag_nt', 'koi_fpflag_ss',
       'koi_fpflag_co', 'koi_fpflag_ec',
       'koi_period','koi_time0bk','koi_impact','koi_duration','koi_prad','koi_teq',
       'koi_model_snr', 'koi_tce_plnt_num','koi_steff','koi_srad')]
~~~

4.2 Data Analytics Models

Flowchart

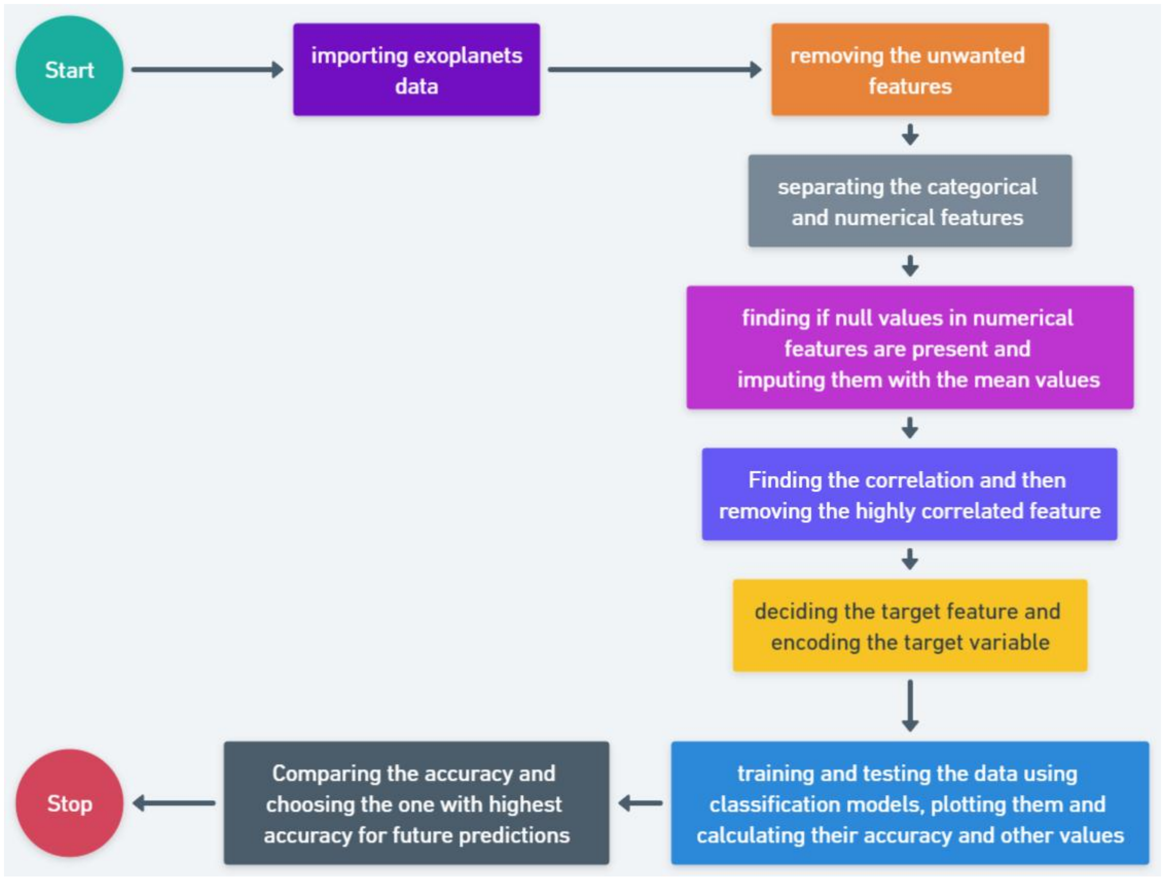

Start
importing exoplanets data
removing the unwanted features
separating the categorical and numerical features
finding if null values in numerical features are present and imputing them with the mean values
Finding the correlation and then removing the highly correlated feature
deciding the target feature and encoding the target variable
training and testing the data using classification models, plotting them and calculating their accuracy and other values
Comparing the accuracy and choosing the one with highest accuracy for future predictions
Stop

Figure-1

4.2.1. Logistic Regression

It comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

The equation for this model is :

$$y=mx+c$$

where y is the dependent variable, m is the slope, x is independent variable and c is the constant.

The curve from the logistic function indicates the likelihood of something Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

MERITS

• It gives direction of association between variables i.e. positive or negative.

• It is easy to implement, interpret and train.

DEMERITS

• It doesn’t work for continuous dataset.

• A condition called overfit arises when the observations are lesser than features.

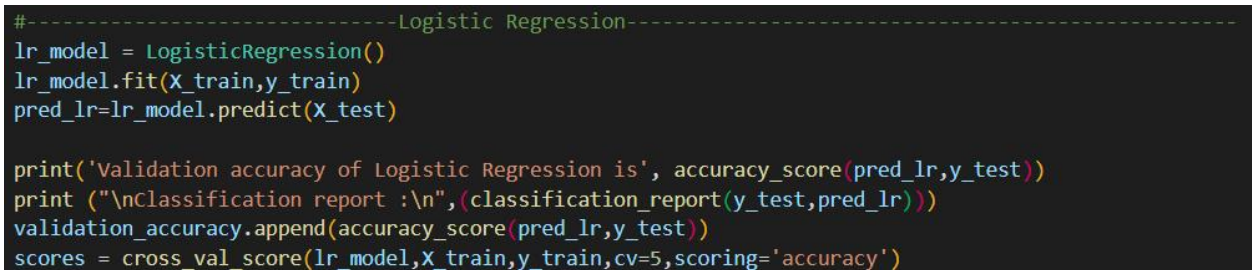
PSEUDOCODE

• Given a dataset of n examples with m features and binary labels, we want to find the weights w such that the predicted probability p of the positive class given the features is as close as possible to the true label.

• We assume the probability p follows a logistic function and use maximum likelihood estimation to find the optimal weights.

• We define a loss function based on the difference between the predicted probabilities and the true labels and use gradient descent to update the weights.

• After training, we can make predictions on new data by computing the predicted probability using the learned weights and passing it through the logistic function.



```
#------------------------------Logistic Regression------------------------------------------------
lr_model = LogisticRegression()
lr_model.fit(X_train,y_train)
pred_lr=lr_model.predict(X_test)

print('Validation accuracy of Logistic Regression is', accuracy_score(pred_lr,y_test))
print ("\nClassification report :\n",(classification_report(y_test,pred_lr)))
validation_accuracy.append(accuracy_score(pred_lr,y_test))
scores = cross_val_score(lr_model,X_train,y_train,cv=5,scoring='accuracy')
```

4.2.2. KNN Classifier

Among all machine learning algorithms, the k-NN method is one of the most straightforward. The K-NN algorithm places the new case in the category that is most similar to the available categories based on the assumption that the new case and the data are comparable to the available cases. A new data point is classified using the K-NN method based on similarity after all the existing data has been stored. This implies that using the K-NN algorithm, new data can be quickly and accurately classified into a suitable category.

MERITS

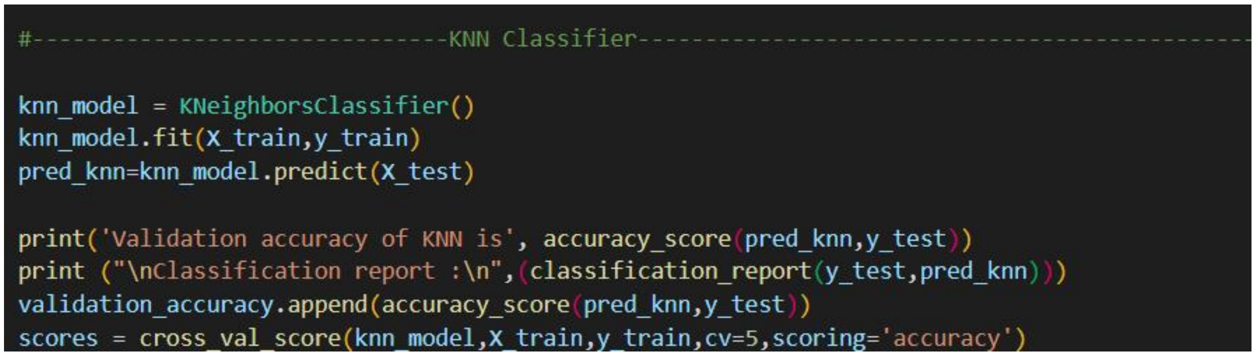
• It saves time because it doesn't require a training period and is simple to apply because we only need to figure out how far apart the points are.

DEMERITS

• KNN algorithm is not cost-effective for calculating distances between each data instance for big datasets.
• Calculating distance is difficult for bigger dimensions as well.
• It struggles when there are missing numbers.

PSEUDOCODE

- Define the number of neighbors K.
- For each point in the dataset: a. Calculate the distance between the query point and the current point. b. Add the distance and the index of the point to a collection.
- Sort the collection of distances and indices from smallest to largest based on the distances.
- Select the first K entries from the sorted collection.
- Retrieve the labels of the selected K entries.
- If it's a regression problem, calculate the mean of the K labels and return it.
- If it's a classification problem, calculate the mode of the K labels and return it.



```
#------------------------------KNN Classifier-------------------------------------------

knn_model = KNeighborsClassifier()
knn_model.fit(X_train,y_train)
pred_knn=knn_model.predict(X_test)

print('Validation accuracy of KNN is', accuracy_score(pred_knn,y_test))
print ("\nClassification report :\n",(classification_report(y_test,pred_knn)))
validation_accuracy.append(accuracy_score(pred_knn,y_test))
scores = cross_val_score(knn_model,X_train,y_train,cv=5,scoring='accuracy')
```

4.2.3. Random Forerst Classifier

Random Forest is a popular ensemble learning algorithm that uses multiple decision trees to improve the accuracy and stability of the final prediction.

MERITS

• It is one of the most accurate algorithms available for classification and regression tasks.

• Robust to noise and outliers: Random Forest is robust to noise and outliers, making it a good choice for noisy datasets.

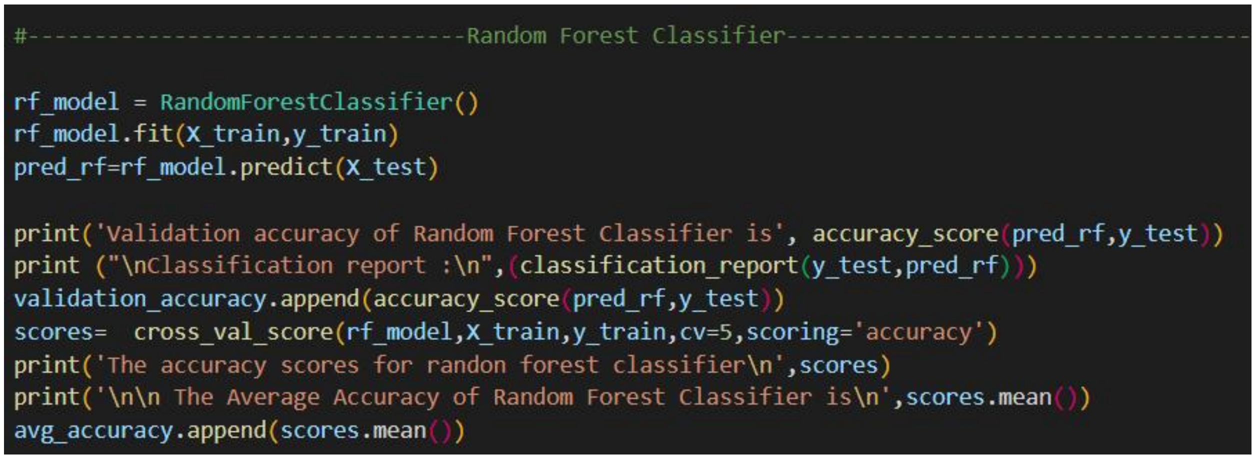
DEMERITS

• It is a complex algorithm, and the training process can be slow for large datasets.

• Requires tuning: Random Forest has several hyperparameters that require tuning to achieve optimal performance.

PSEUDOCODE

- Define the number of trees to be used in the forest (n_trees) and the number of features to be considered for each split (max_features).
- For each tree in the forest: a. Randomly select a subset of the original dataset with replacement (bootstrap sample) to create a new dataset. b. Randomly select a subset of features (max_features) to consider at each split of the tree. c. Build a decision tree using the selected dataset and features.
- Predict the output of a new data point by aggregating the predictions of all trees in the forest:
    - a. For classification, take the majority vote of the predicted class labels.
    - b. For regression, take the average of the predicted values.



```
#--------------------------------Random Forest Classifier----------------------------------

rf_model = RandomForestClassifier()
rf_model.fit(X_train,y_train)
pred_rf=rf_model.predict(X_test)

print('Validation accuracy of Random Forest Classifier is', accuracy_score(pred_rf,y_test))
print ("\nClassification report :\n",(classification_report(y_test,pred_rf)))
validation_accuracy.append(accuracy_score(pred_rf,y_test))
scores=  cross_val_score(rf_model,X_train,y_train,cv=5,scoring='accuracy')
print('The accuracy scores for randon forest classifier\n',scores)
print('\n\n The Average Accuracy of Random Forest Classifier is\n',scores.mean())
avg_accuracy.append(scores.mean())
```

4.2.4. Decision Tree

It works by recursively splitting the data based on the most important features until a stopping criterion is met, such as reaching a certain depth or minimum number of samples in each leaf node.

MERITS

• Easy to interpret: Decision trees are easy to interpret and explain, making them a popular choice for decision-making tasks.

• Can handle both numerical and categorical data: Decision trees can handle both numerical and categorical data without requiring any preprocessing.

• Efficient for small datasets: Decision trees are efficient for small datasets and can be trained quickly.

DEMERITS

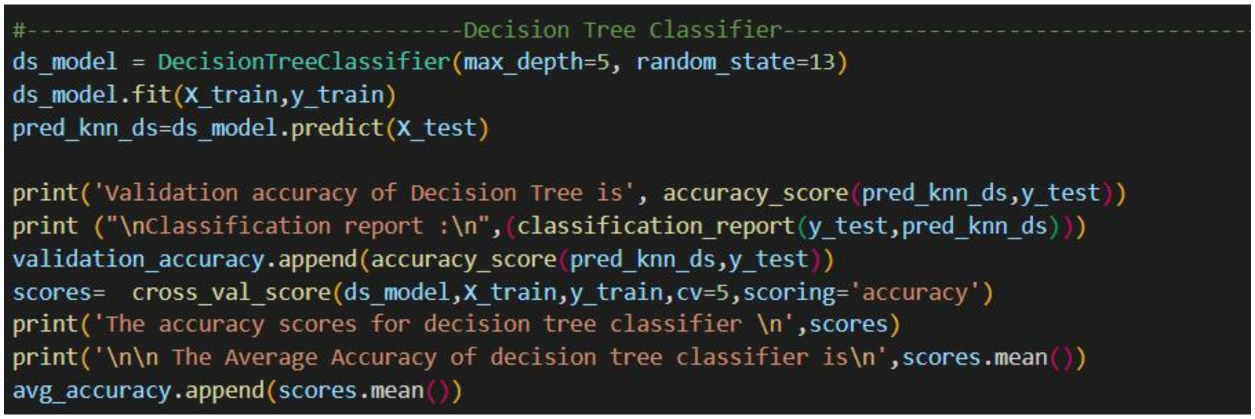
• Prone to overfitting: Decision trees are prone to overfitting the training data, especially if the tree is deep.

• Limited expressiveness: Decision trees are limited in their expressiveness and may not be able to capture complex relationships between features.

PSEUDOCODE

- Define the dataset and the target variable to be predicted.
- Select a feature to split the dataset based on a criterion such as information gain or Gini index.

- Split the dataset into two subsets based on the selected feature value.
- Recursively repeat steps 2-3 for each subset until a stopping criterion is met (e.g., a maximum depth or a minimum number of samples per leaf).
- Assign the target variable value to each leaf node based on the majority class or the mean value of the samples in that node.
- Predict the target variable of a new data point by traversing the tree from the root node to a leaf node based on the feature values of the data point.
- Assign the target variable value of the corresponding leaf node as the predicted value.



```
#-------------------------------Decision Tree Classifier----------------------------------
ds_model = DecisionTreeClassifier(max_depth=5, random_state=13)
ds_model.fit(X_train,y_train)
pred_knn_ds=ds_model.predict(X_test)

print('Validation accuracy of Decision Tree is', accuracy_score(pred_knn_ds,y_test))
print ("\nClassification report :\n",(classification_report(y_test,pred_knn_ds)))
validation_accuracy.append(accuracy_score(pred_knn_ds,y_test))
scores=  cross_val_score(ds_model,X_train,y_train,cv=5,scoring='accuracy')
print('The accuracy scores for decision tree classifier \n',scores)
print('\n\n The Average Accuracy of decision tree classifier is\n',scores.mean())
avg_accuracy.append(scores.mean())
```

4.2.5. Naive Bayes

Naïve Bayes classifier is used in supervised learning method and it is based on “probability” concept to classify new entities. It assigns a new observation to the most probable class. The classification process comprises two stages as follows:

• Training stage: Using the training samples, the method computes the probability distribution of that sample.

• Prediction stage: For test sample, the method computes the posterior probability of that unknown instance. The posterior is predicting that the sample belonging to each class according to the largest posterior probability, which is called Maximum A Posterior (MAP)

MERITS

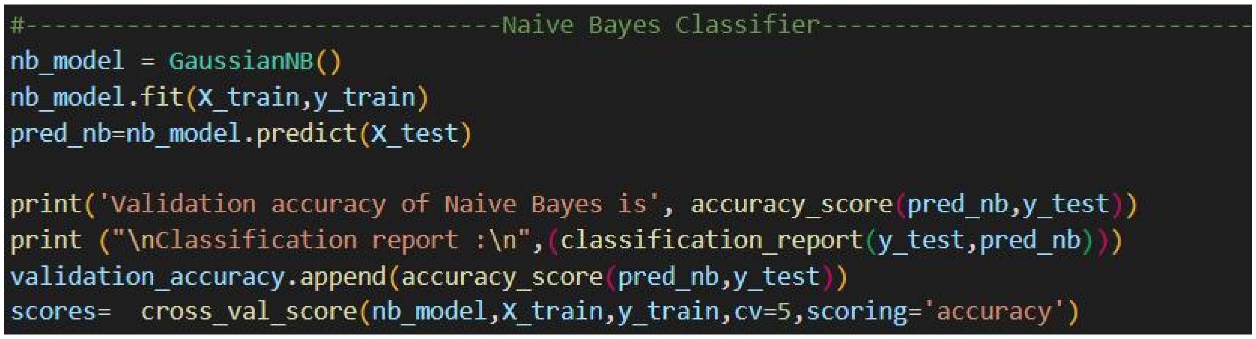
• It is easier to implement.
• It works well with continuous and discrete both.
• It can be implemented in real time problems

DEMERITS

• It has zero frequency problem; it assigns zero probability to categorical data.
• Its predictions may not be correct at times.

PSEUDOCODE

- Define the dataset and the target variable to be predicted.
- For each class c, calculate the prior probability P(c) as the proportion of the samples in the dataset that belong to class c.
- For each feature xi, calculate the conditional probability P(xi|c) as the proportion of the samples in class c that have feature xi.
- For a new data point with feature values x1, x2, ..., xn: a. For each class c, calculate the posterior probability P(c|x1, x2, ..., xn) using Bayes' rule: P(c|x1, x2, ..., xn) = P(c) * P(x1|c) * P(x2|c) * ... * P(xn|c) / P(x1, x2, ..., xn) where P(x1, x2, ..., xn) is a normalization constant that ensures the probabilities sum up to 1. b. Assign the predicted class as the one with the highest posterior probability.



```
#---------------------------------Naive Bayes Classifier-------------------------------
nb_model = GaussianNB()
nb_model.fit(X_train,y_train)
pred_nb=nb_model.predict(X_test)

print('Validation accuracy of Naive Bayes is', accuracy_score(pred_nb,y_test))
print ("\nClassification report :\n",(classification_report(y_test,pred_nb)))
validation_accuracy.append(accuracy_score(pred_nb,y_test))
scores=  cross_val_score(nb_model,X_train,y_train,cv=5,scoring='accuracy')
```

4.2.6. XGB Classifier

XGBoost (Extreme Gradient Boosting) is a gradient boosting algorithm that is widely used for classification and regression problems. It builds a series of decision trees, and each tree is trained to correct the errors made by the previous tree.

MERITS

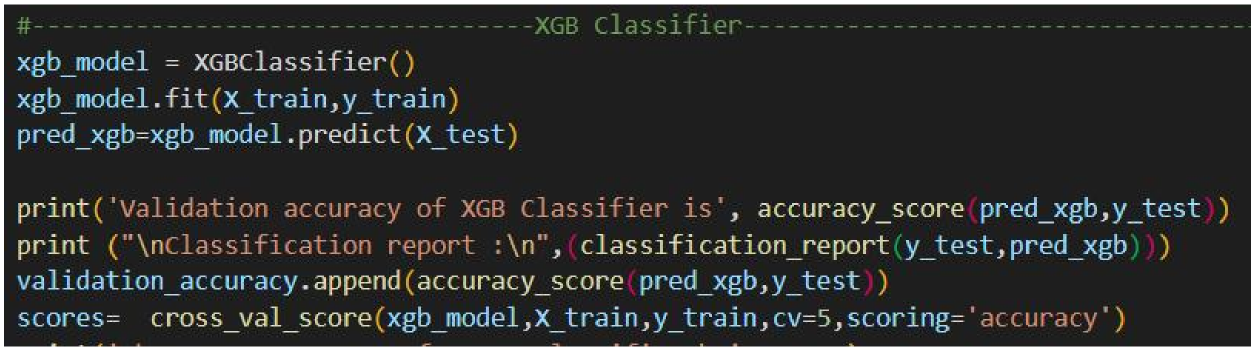
- It has high accuracy and is widely used in machine learning competitions.
- It can handle missing data.
- It has a parallel processing capability that makes it fast.

DEMERITS

- It can be sensitive to overfitting.
- It may require tuning of parameters to achieve optimal performance.

PSEUDOCODE

- Define the dataset and the target variable to be predicted.
- Define the parameters for the XGBoost model, such as the learning rate, the number of trees to build (n_estimators), and the maximum depth of each tree (max_depth).
- For each tree in the model: a. Use gradient boosting to fit the residual errors of the previous tree, rather than the actual target values, to the current tree. b. Split the dataset into subsets based on a selected feature using a criterion such as information gain or Gini index. c. Recursively repeat steps 3a-3b for each subset until a stopping criterion is met (e.g., a maximum depth or a minimum number of samples per leaf).
- Assign the predicted class as the one with the highest probability based on the output of all trees in the model: a. Use the softmax function to convert the output of each tree into class probabilities. b. Sum the probabilities of each class across all trees. c. Assign the predicted class as the one with the highest probability.



```
#----------------------------------XGB Classifier----------------------------------
xgb_model = XGBClassifier()
xgb_model.fit(X_train,y_train)
pred_xgb=xgb_model.predict(X_test)

print('Validation accuracy of XGB Classifier is', accuracy_score(pred_xgb,y_test))
print ("\nClassification report :\n",(classification_report(y_test,pred_xgb)))
validation_accuracy.append(accuracy_score(pred_xgb,y_test))
scores=  cross_val_score(xgb_model,X_train,y_train,cv=5,scoring='accuracy')
```

4.2.6.Extra Tree Classifier

Extra Trees Classifier is a machine learning algorithm that is based on the Random Forest algorithm, which uses multiple decision trees to improve the accuracy and stability of the final prediction. However, Extra Trees Classifier differs from Random Forest in that it uses a randomized split point selection to further reduce variance and increase performance.

MERITS

•High accuracy: Extra Trees Classifier can achieve high accuracy with relatively few trees, making it a computationally efficient algorithm.

• Low variance: Extra Trees Classifier has low variance compared to Random Forest, making it less prone to overfitting.

DEMERITS

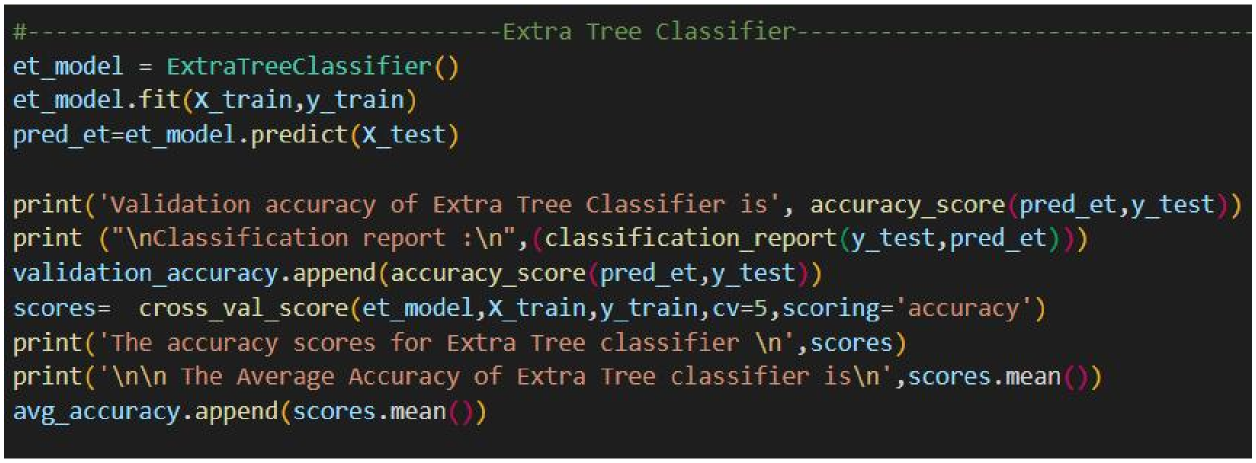
• Limited interpretability: Extra Trees Classifier is less interpretable than other decision tree-based algorithms, making it difficult to explain the final prediction.

•Bias towards majority class: In imbalanced datasets, Extra Trees Classifier tends to be biased towards the majority class.

PSEUDOCODE

- Define the dataset and the target variable to be predicted.
- Define the parameters for the Extra Trees model, such as the number of trees to build (n_estimators), the maximum depth of each tree (max_depth), and the number of features to consider for each split (max_features).

- For each tree in the model: a. Randomly select a subset of the original dataset with replacement (bootstrap sample) to create a new dataset. b. Randomly select a subset of features (max_features) to consider at each split of the tree. c. Build a decision tree using the selected dataset and features. d. For each split, randomly select a threshold value for each selected feature to further randomize the split.
- Assign the predicted class as the one with the highest frequency across all trees in the model: a. For each data point, predict the class label of the point based on the majority vote of the predicted class labels from all trees in the model.

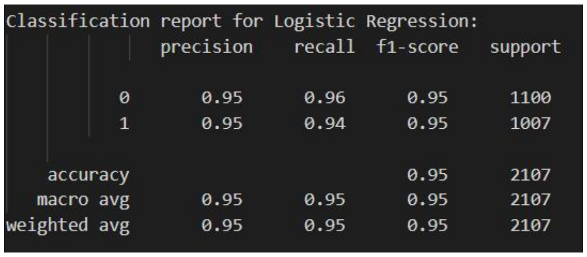


```
#---------------------------------Extra Tree Classifier--------------------------------
et_model = ExtraTreeClassifier()
et_model.fit(X_train,y_train)
pred_et=et_model.predict(X_test)

print('Validation accuracy of Extra Tree Classifier is', accuracy_score(pred_et,y_test))
print ("\nClassification report :\n",(classification_report(y_test,pred_et)))
validation_accuracy.append(accuracy_score(pred_et,y_test))
scores=  cross_val_score(et_model,X_train,y_train,cv=5,scoring='accuracy')
print('The accuracy scores for Extra Tree classifier \n',scores)
print('\n\n The Average Accuracy of Extra Tree classifier is\n',scores.mean())
avg_accuracy.append(scores.mean())
```

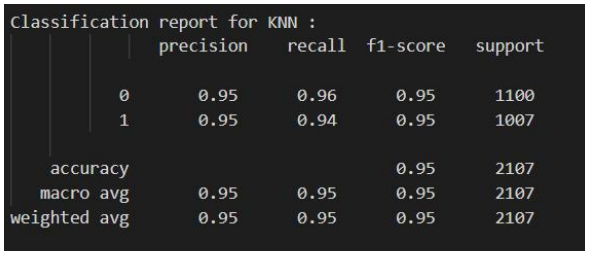
5. Results and Discussions:

5.1. Statistical

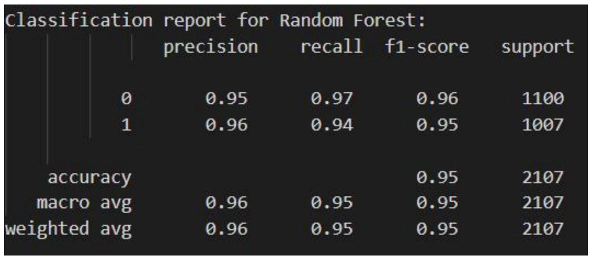
For each classification model the classification report is printed along with their accuracies. The validation accuracy for Extra Tree Classifier is found to be maximum i.e. 95.30% while the average accuracy of Random Forest is maximum 95.19%.



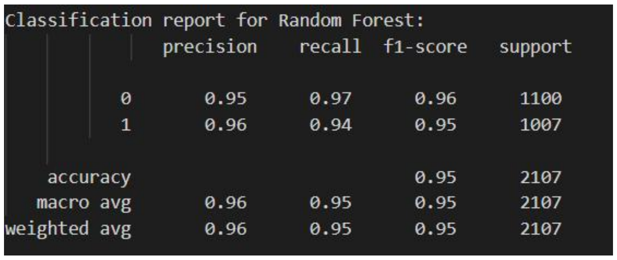
```
Classification report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.95      0.96      0.95      1100
           1       0.95      0.94      0.95      1007

    accuracy                           0.95      2107
   macro avg       0.95      0.95      0.95      2107
weighted avg       0.95      0.95      0.95      2107
```



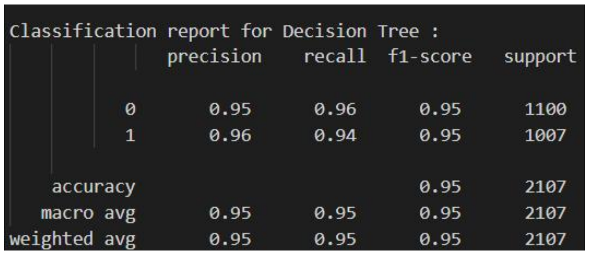
```
Classification report for KNN :
              precision    recall  f1-score   support

           0       0.95      0.96      0.95      1100
           1       0.95      0.94      0.95      1007

    accuracy                           0.95      2107
   macro avg       0.95      0.95      0.95      2107
weighted avg       0.95      0.95      0.95      2107
```



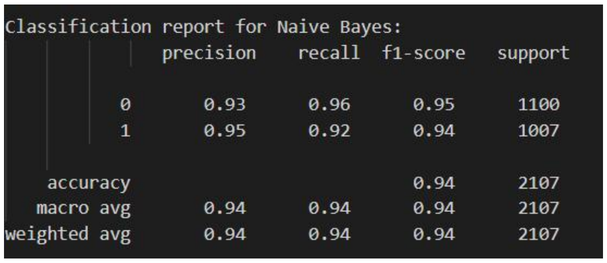
```
Classification report for Random Forest:
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      1100
           1       0.96      0.94      0.95      1007

    accuracy                           0.95      2107
   macro avg       0.96      0.95      0.95      2107
weighted avg       0.96      0.95      0.95      2107
```



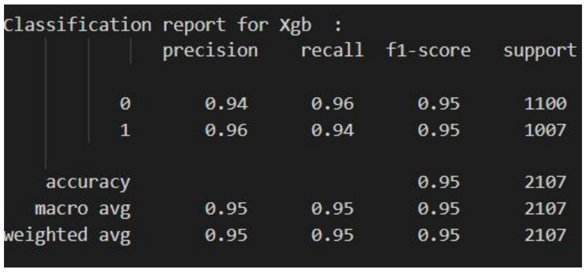
```
Classification report for Random Forest:
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      1100
           1       0.96      0.94      0.95      1007

    accuracy                           0.95      2107
   macro avg       0.96      0.95      0.95      2107
weighted avg       0.96      0.95      0.95      2107
```



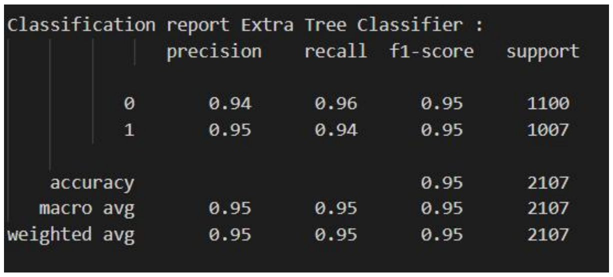
```
Classification report for Decision Tree :
              precision    recall  f1-score   support

           0       0.95      0.96      0.95      1100
           1       0.96      0.94      0.95      1007

    accuracy                           0.95      2107
   macro avg       0.95      0.95      0.95      2107
weighted avg       0.95      0.95      0.95      2107
```



```
Classification report for Naive Bayes:
              precision    recall  f1-score   support

           0       0.93      0.96      0.95      1100
           1       0.95      0.92      0.94      1007

    accuracy                           0.94      2107
   macro avg       0.94      0.94      0.94      2107
weighted avg       0.94      0.94      0.94      2107
```

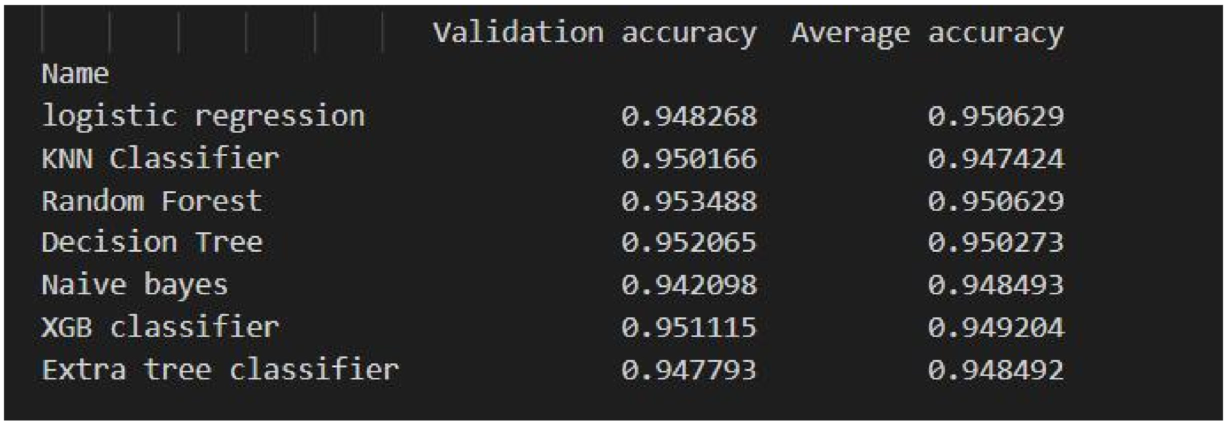


```
Classification report for Xgb  :
              precision    recall  f1-score   support

           0       0.94      0.96      0.95      1100
           1       0.96      0.94      0.95      1007

    accuracy                           0.95      2107
   macro avg       0.95      0.95      0.95      2107
weighted avg       0.95      0.95      0.95      2107
```



```
Classification report Extra Tree Classifier :
              precision    recall  f1-score   support

           0       0.94      0.96      0.95      1100
           1       0.95      0.94      0.95      1007

    accuracy                           0.95      2107
   macro avg       0.95      0.95      0.95      2107
weighted avg       0.95      0.95      0.95      2107
```

The matrix representing the accuracies of various models. Average accuracy measures overall performance of a model on the entire data while validation accuracy measure performance specifically on a subset of data that was not used for the training.



```
                          Validation accuracy  Average accuracy
Name
logistic regression                  0.948268          0.950629
KNN Classifier                       0.950166          0.947424
Random Forest                        0.953488          0.950629
Decision Tree                        0.952065          0.950273
Naive bayes                          0.942098          0.948493
XGB classifier                       0.951115          0.949204
Extra tree classifier                0.947793          0.948492
```

Figure-2

5.2. Plots

Plotting the target variable

~~~
```{r}
#plotting koi_pdisposition
library(ggplot2)
count <- table(df$koi_pdisposition)
barplot(count,col=c("brown","purple"))

```
~~~

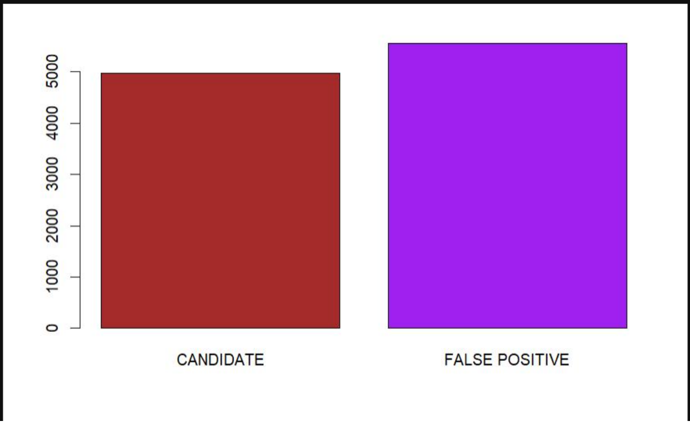


Figure-3

Plotting transit duration hours and transit depth (transit is basically the movement of the planet)

~~~
```{r}
plot(df$koi_depth, df$koi_duration, col="blue", main="Disposition Using Kepler Data")
```
~~~

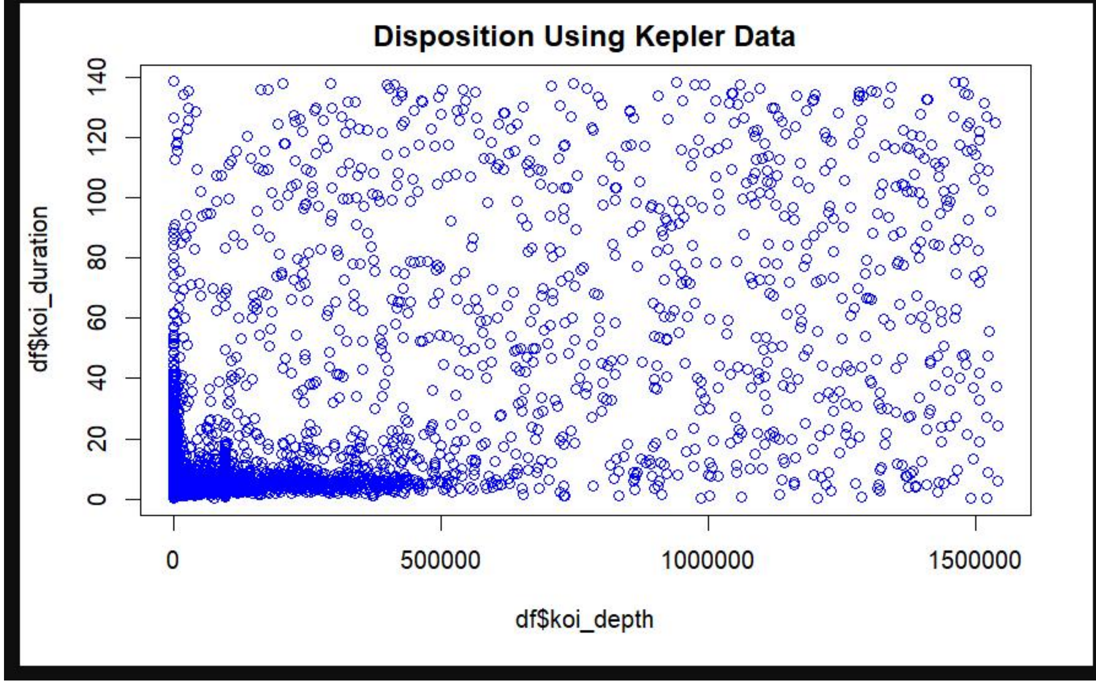
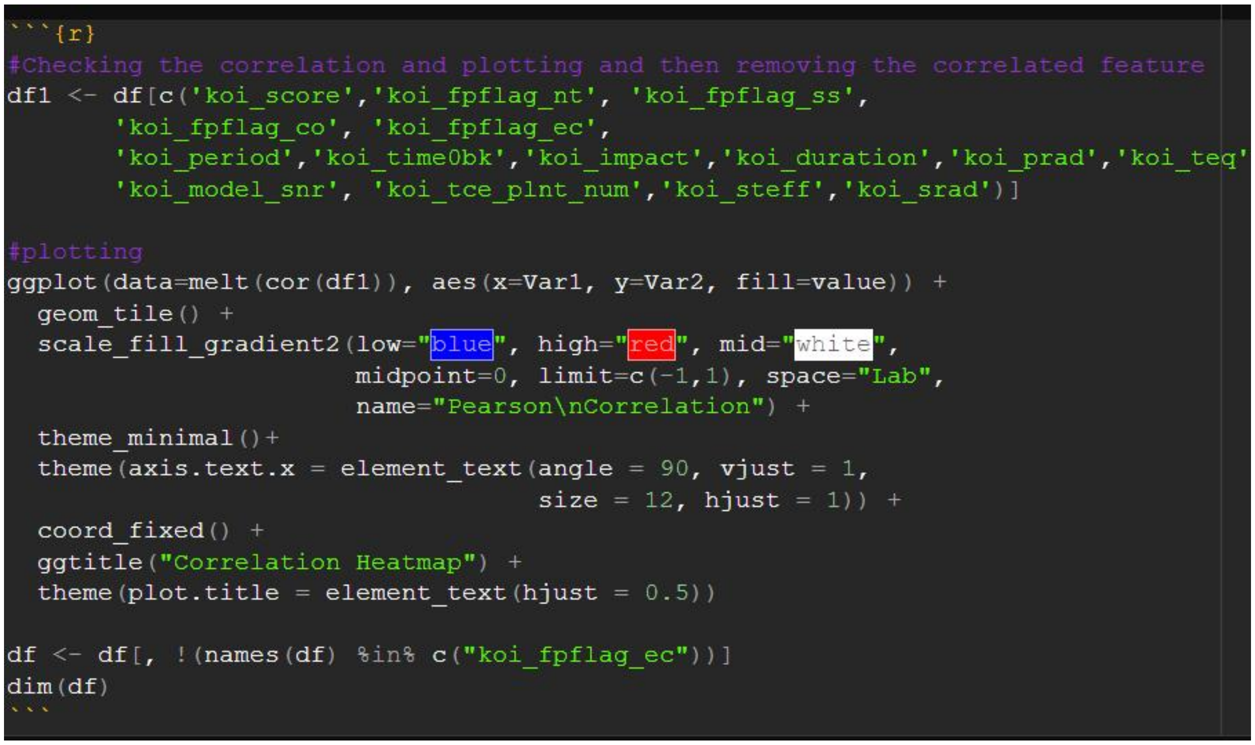


Figure-4

Plotting the correlation heatmap for the dataset to find out the highly correlated parameters



~~~
```{r}
#Checking the correlation and plotting and then removing the correlated feature
df1 <- df[c('koi_score','koi_fpflag_nt', 'koi_fpflag_ss',
       'koi_fpflag_co', 'koi_fpflag_ec',
       'koi_period','koi_time0bk','koi_impact','koi_duration','koi_prad','koi_teq'
       'koi_model_snr', 'koi_tce_plnt_num','koi_steff','koi_srad')]

#plotting
ggplot(data=melt(cor(df1)), aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  scale_fill_gradient2(low="blue", high="red", mid="white",
                       midpoint=0, limit=c(-1,1), space="Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 90, vjust = 1,
                                   size = 12, hjust = 1)) +
  coord_fixed() +
  ggtitle("Correlation Heatmap") +
  theme(plot.title = element_text(hjust = 0.5))

df <- df[, !(names(df) %in% c("koi_fpflag_ec"))]
dim(df)
```
~~~

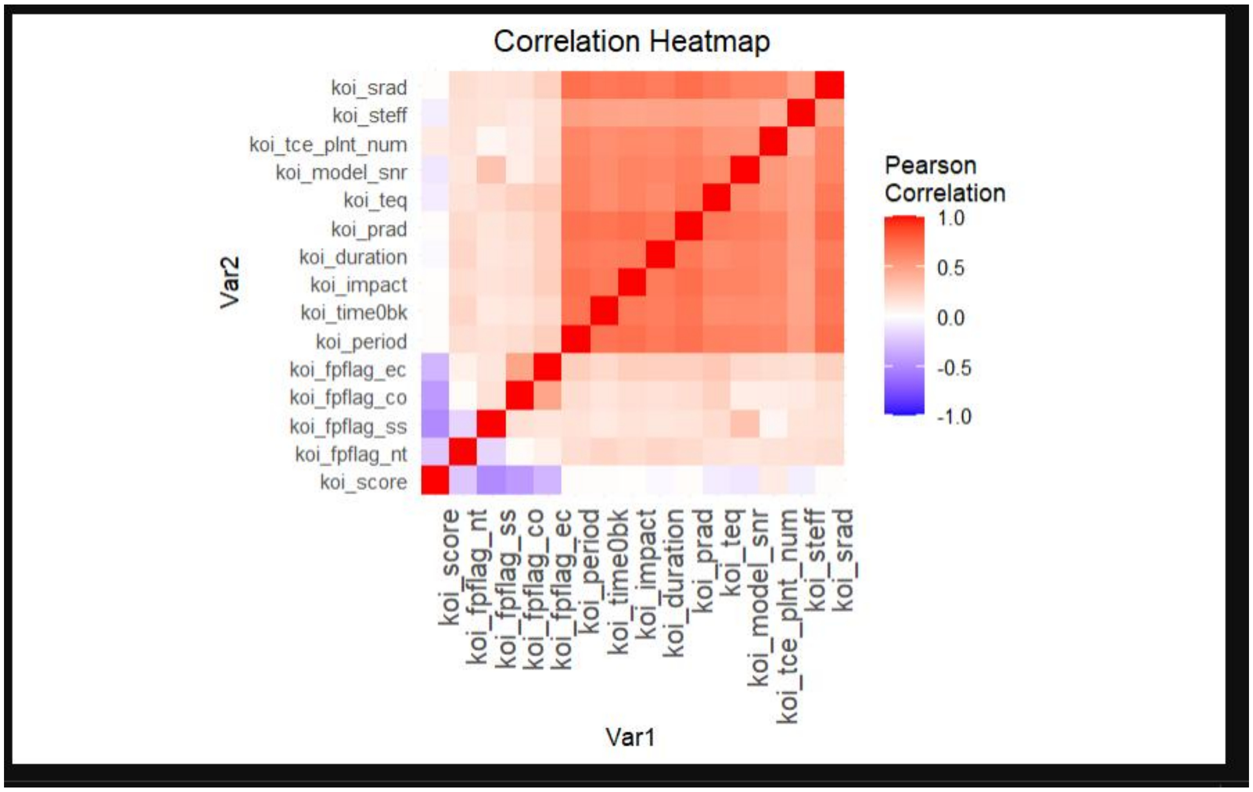


Figure-5

These are confusion matrix plots for each models The diagonal cells of the matrix represent correct predictions, while the off-diagonal cells represent incorrect predictions.

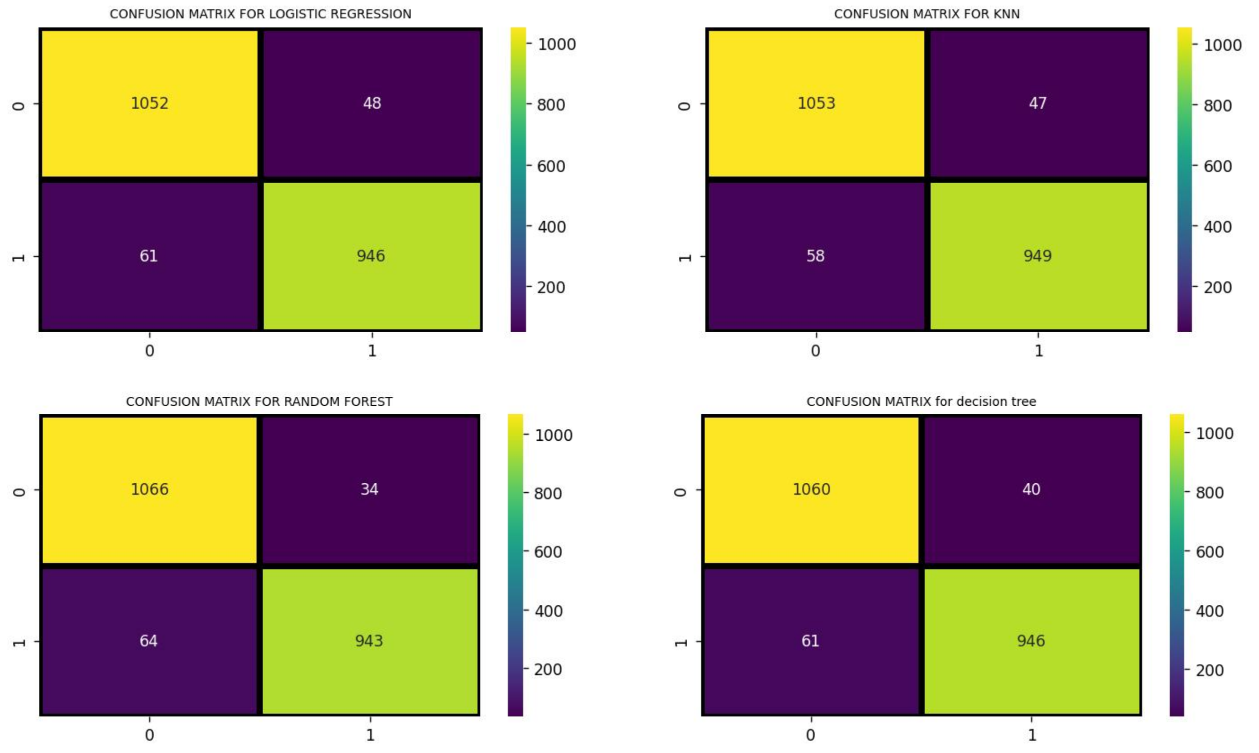


Figure-6

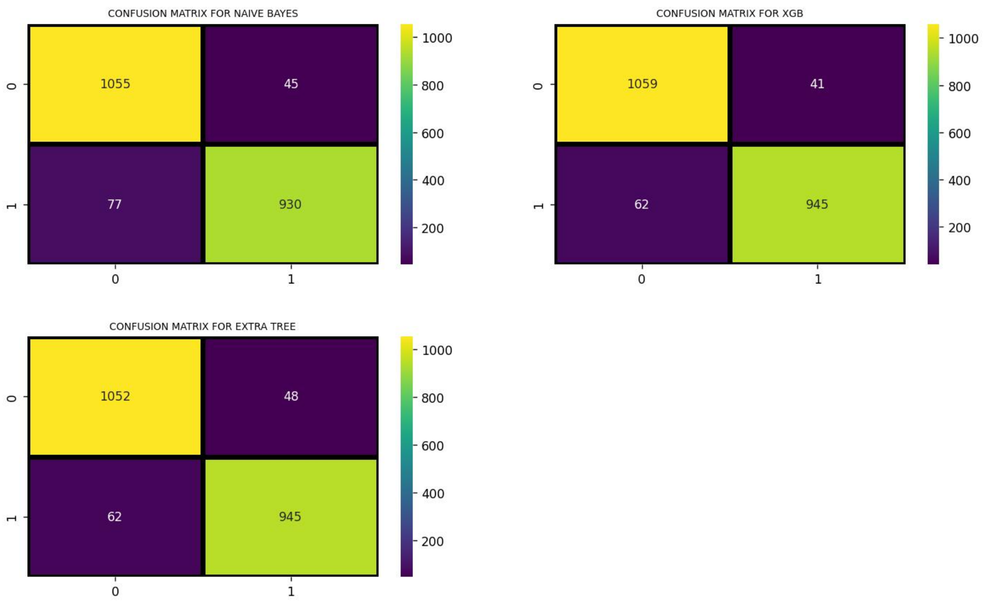


Figure-7

The ROC curve is generated by plotting the true positive rate (TPR) versus the false positive rate (FPR) at different threshold values of the model's predicted probability of a positive outcome.

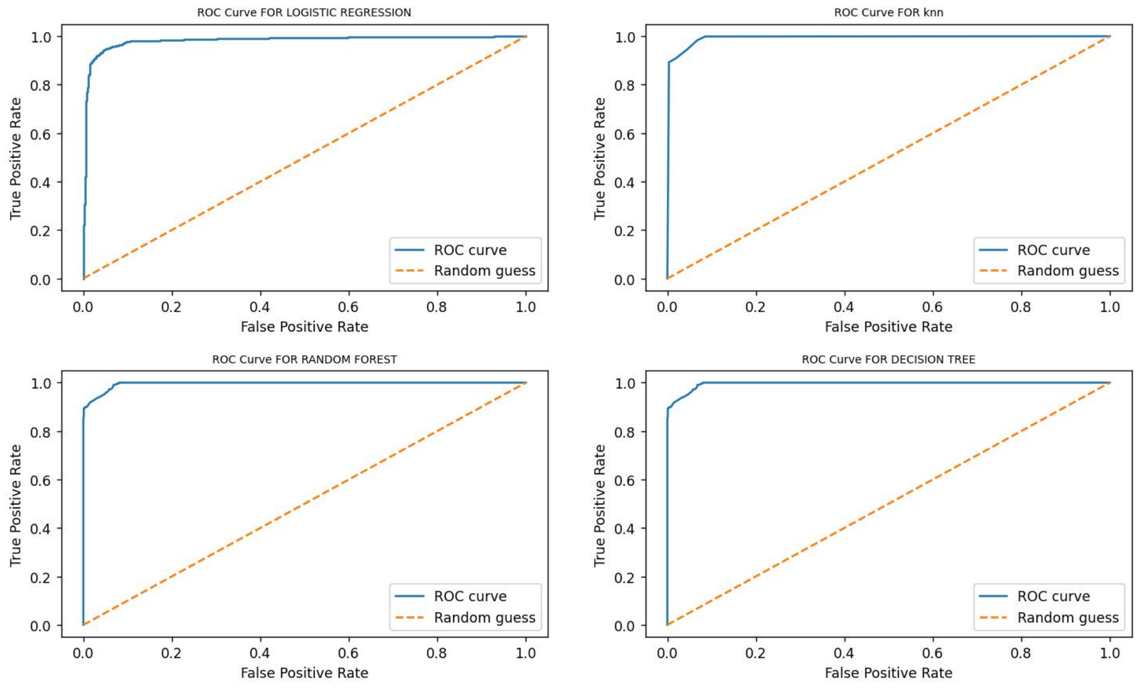


Figure-7

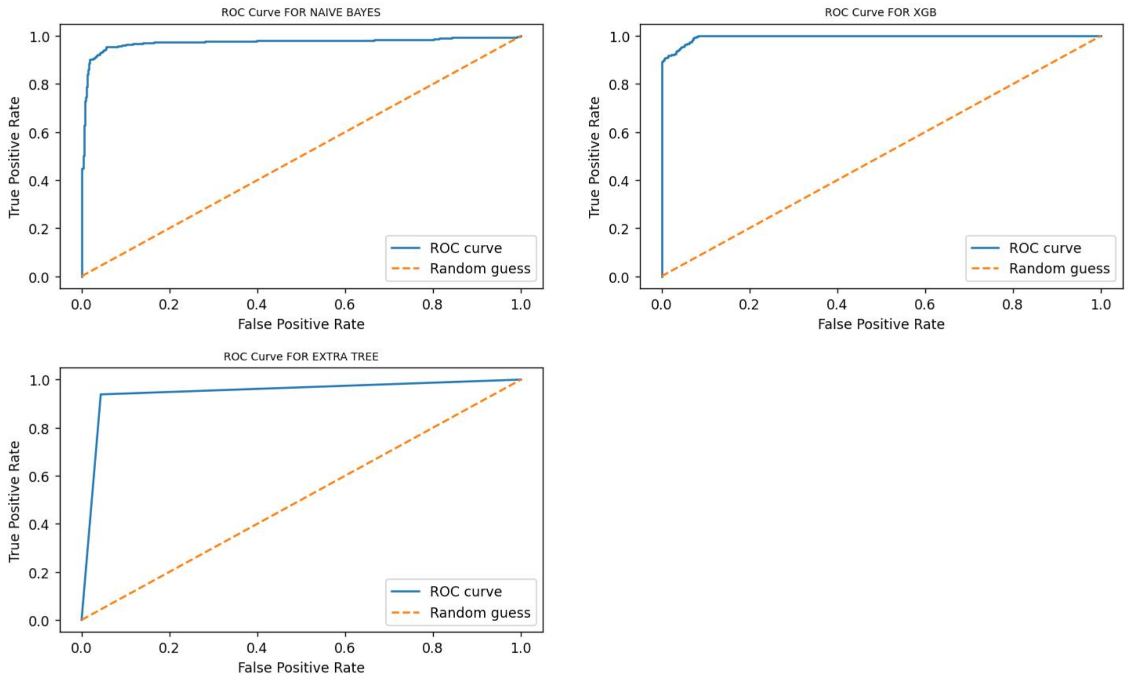


Figure-8

6. Conclusion:

The results show that the validation accuracy for Extra Tree Classifier is found to be maximum i.e. 95.30% while the average accuracy of Random Forest is maximum 95.19%. The exoplanet exploration has become an important aspect because the humans are continously exhausting the precious elements of the earth. The reason for exploration may vary from searching life to searching for the presence of precious minerals. From this project we predicted the planets which are a candidate for the habitable planet based on the trained models. The future scopes may include the development of models which are specifically dedicated to this field since it is a very vast domain and also has high levels of accuracy from the present ones.

7. References

[1] Ansdell, M. et al. (2018) “Scientific domain knowledge improves exoplanet transit classification with deep learning,” The astrophysical journal. Letters, 869(1), p. L7. doi: 10.3847/2041-8213/aaf23b.

[2] Borde, P., Rouan, D. and Leger, A. (2003) “Exoplanet detection capability of the COROT space mission,” arXiv [astro-ph]. Available at: http://arxiv.org/abs/astro-ph/0305159.

[3] Chintarungruangchai, P. and Jiang, I.-G. (2019) “Detecting exoplanet transits through machine learning techniques with convolutional neural networks,” arXiv [astro-ph.EP]. Available at: http://arxiv.org/abs/1904.12419.

[4] Cuéllar, S. et al. (2022) “Deep learning exoplanets detection by combining real and synthetic data,” PloS one, 17(5), p. e0268199. doi: 10.1371/journal.pone.0268199.

[5] Jagtap, R. et al. (2021) “Habitability of Exoplanets using Deep Learning,” in 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS). IEEE.

[6] Matchev, K. T., Matcheva, K. and Roman, A. (2022) “Unsupervised machine learning for exploratory data analysis of exoplanet transmission spectra,” The planetary science journal, 3(9), p. 205. doi: 10.3847/psj/ac880b.

[7] Schanche, N. et al. (2019) “Machine-learning approaches to exoplanet transit detection and candidate validation in wide-field ground-based surveys,” Monthly notices of the Royal Astronomical Society, 483(4), pp. 5534–5547. doi: 10.1093/mnras/sty3146.

[8] Shallue, C. J. and Vanderburg, A. (2018) “Identifying exoplanets with deep learning: A five-planet resonant chain around Kepler-80 and an eighth planet around Kepler-90,” The Astronomical journal, 155(2), p. 94. doi: 10.3847/1538-3881/aa9e09.

[9] Herur, A.N., Tajmohamed, R. and Ponsam, J.G., 2022, July. Exploring Exoplanets using kNN, Logistic Regression and Decision Trees. In 2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES) (pp. 1-7). IEEE.

[10] Devirian, M., 2008, March. Exploring ExoPlanets: NASA's Future Space Missions. In 2008 IEEE Aerospace Conference (pp. 1-8). IEEE.

[11] Bhamare, A.R., Baral, A. and Agarwal, S., 2021, June. Analysis of Kepler objects of interest using machine learning for Exoplanet Identification. In 2021 International Conference on Intelligent Technologies (CONIT) (pp. 1-8). IEEE.

[12] Feliz, D. L. et al. (2021) “NEMESIS: Exoplanet transit survey of nearby M-dwarfs in TESS FFIs. I,” The Astronomical journal, 161(5), p. 247. doi: 10.3847/1538-3881/abedb3.

[13] Hara, N. C. et al. (2022) “Detecting exoplanets with the false inclusion probability: Comparison with other detection criteria in the context of radial velocities,” Astronomy and astrophysics, 663, p. A14. doi: 10.1051/0004-6361/202140543.

[14] Mirrashid, M. and Naderpour, H., 2022. Transit search: An optimization algorithm based on exoplanet exploration. Results in Control and Optimization, 7, p.100127.

[15] Edwards, B., Rice, M., Zingales, T., Tessenyi, M., Waldmann, I., Tinetti, G., Pascale, E., Savini, G. and Sarkar, S., 2019. Exoplanet spectroscopy and photometry with the Twinkle space telescope. Experimental Astronomy, 47, pp.29-63.

[16] Barnes, R., 2017. Tidal locking of habitable exoplanets. Celestial mechanics and dynamical astronomy, 129, pp.509-536.

[17] Loredo, T.J., Berger, J.O., Chernoff, D.F., Clyde, M.A. and Liu, B., 2012. Bayesian methods for analysis and adaptive scheduling of exoplanet observations. Statistical Methodology, 9(1-2), pp.101-114.

[18] Van Hoolst, T., Noack, L. and Rivoldini, A., 2019. Exoplanet interiors and habitability. Advances in Physics: X, 4(1), p.1630316.

[19] Mirrashid, M. and Naderpour, H., 2022. Transit search: An optimization algorithm based on exoplanet exploration. Results in Control and Optimization, 7, p.100127.

[20] McNutt Jr, R.L., Wimmer-Schweingruber, R.F., Gruntman, M., Krimigis, S.M., Roelof, E.C., Brandt, P.C., Vernon, S.R., Paul, M.V., Lathrop, B.W., Mehoke, D.S. and Napolillo, D.H., 2019. Near-term interstellar probe: First step. Acta Astronautica, 162, pp.284-299