Team No. **61**, Team Name: **Beta Squad**

Regression algorithm finds the mapping function to map the input to a continuous output.

About algorithms used in model building.:

Classification Algorithms:-

-Classification algorithms are used to predict static or fixed value yes/no, true/false, rice/not rice.
-Classification algorithm is trained on a training data set and based on that training it categorises new test data.
-It finds a mapping function to map an input or to a discrete singular,output.

Regression Algorithms:-

-The task of the Regression algorithm is to find the mapping function to map the input variable(x) to the continuous output variable(y).
-It is used to find correlation between dependent and independent variables .
-And especially to predict continuous values such as (share price, SENSEX,price of a commodity).

- **To List Classification algorithms, they are :-**
- **Logistic regression.**
- **XGBoost.**
- **SVM.**
- **Decision Tree.**

We have discussed in detail the algorithms pertaining to this project:-

- **Random forest Algorithm (Ensemble of decision tree)**

The Random forest or Random Decision Forest is a supervised Machine learning algorithm used for classification, regression, and other tasks using decision trees. The Random forest classifier creates a set of decision trees from a randomly selected subset of the training set. It is basically a set of decision trees (DT) from a randomly selected subset of the training set and then It collects the votes from different decision trees to decide the final prediction.

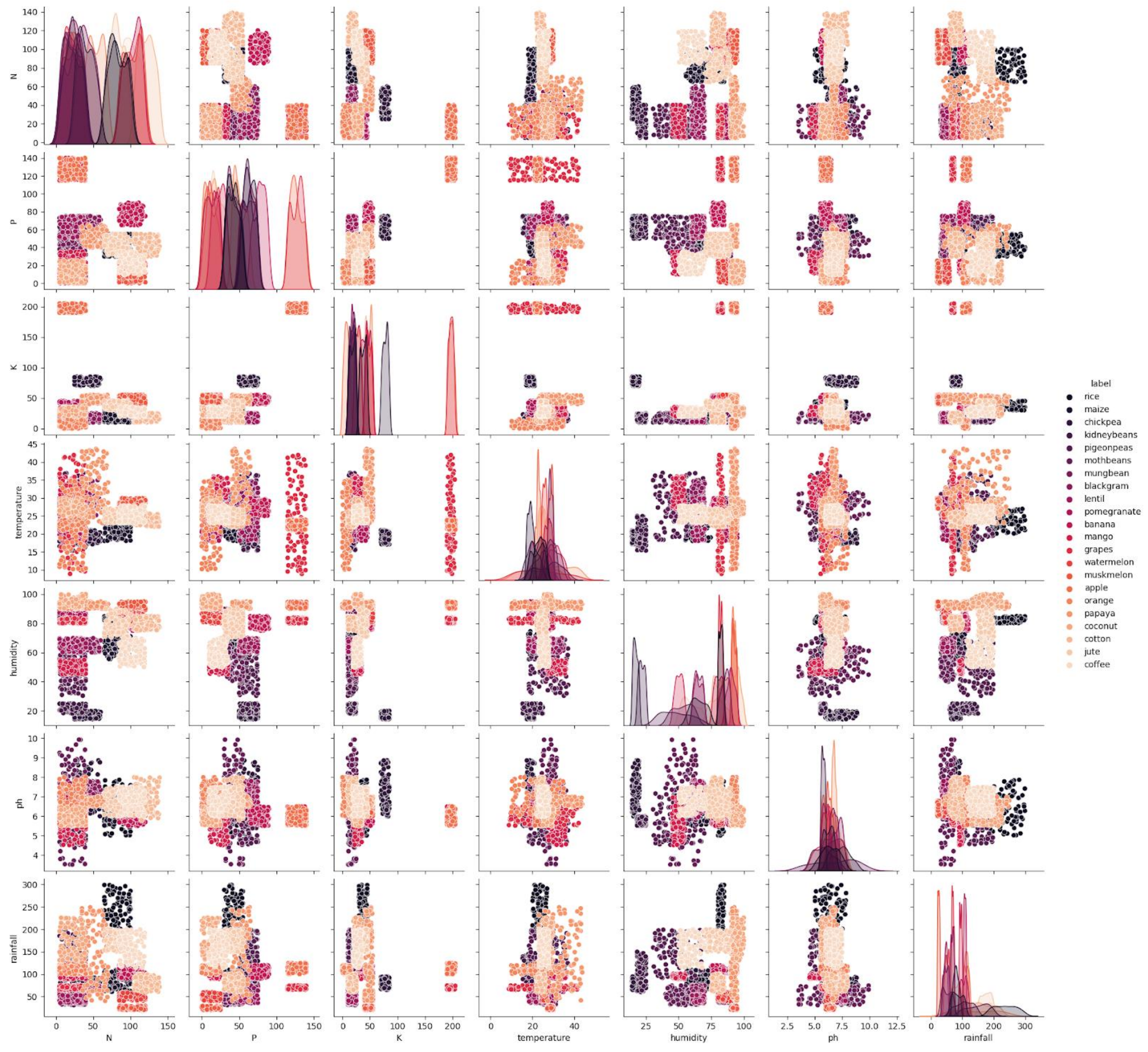
- **K-Nearest Neighbour Algorithm**:

This algorithm is used to solve the classification model problems. The K-nearest neighbour or K-NN algorithm basically creates an imaginary boundary to classify the data. When new data points come in, the algorithm will try to predict that to the nearest of the boundary line.

Therefore, larger k value means smoother curves of separation resulting in less complex models. Whereas, smaller k values tend to overfit the data and result in complex models.

- **<u>Why have we used the specified algorithm?</u>**

On Visualising the Data:



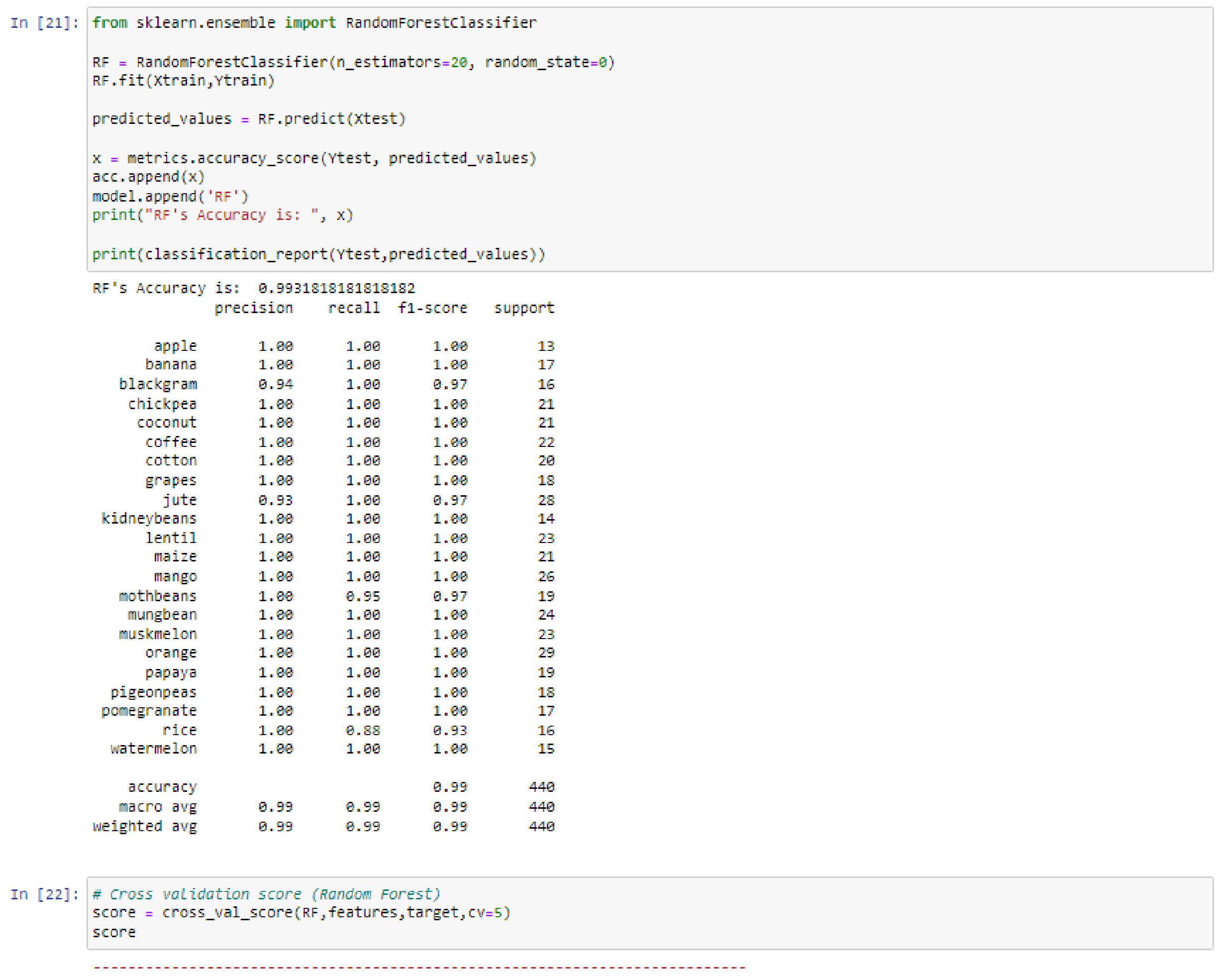
On seeing the data distribution and the scattering of data, It was clear that Linear regression couldn't be used due to excessive overlapping of data points.

Observing the overlapping of plotted points the most suitable model to choose was a decision tree based model such as random forest classifier.

As Per the above inference, The K Nearest Neighbours and the Random Forest Classifier models were tested.

- **Accuracy report :-**

    - RFC accuracy report.



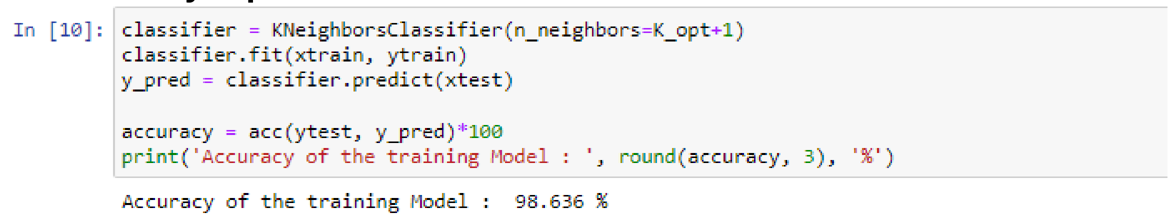
```
In [21]: from sklearn.ensemble import RandomForestClassifier

         RF = RandomForestClassifier(n_estimators=20, random_state=0)
         RF.fit(Xtrain,Ytrain)

         predicted_values = RF.predict(Xtest)

         x = metrics.accuracy_score(Ytest, predicted_values)
         acc.append(x)
         model.append('RF')
         print("RF's Accuracy is: ", x)

         print(classification_report(Ytest,predicted_values))
```

```
RF's Accuracy is:  0.9931818181818182
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       0.94      1.00      0.97        16
    chickpea       1.00      1.00      1.00        21
     coconut       1.00      1.00      1.00        21
      coffee       1.00      1.00      1.00        22
      cotton       1.00      1.00      1.00        20
      grapes       1.00      1.00      1.00        18
        jute       0.93      1.00      0.97        28
 kidneybeans       1.00      1.00      1.00        14
      lentil       1.00      1.00      1.00        23
       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        26
   mothbeans       1.00      0.95      0.97        19
    mungbean       1.00      1.00      1.00        24
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        29
      papaya       1.00      1.00      1.00        19
  pigeonpeas       1.00      1.00      1.00        18
 pomegranate       1.00      1.00      1.00        17
        rice       1.00      0.88      0.93        16
  watermelon       1.00      1.00      1.00        15

    accuracy                           0.99       440
   macro avg       0.99      0.99      0.99       440
weighted avg       0.99      0.99      0.99       440
```

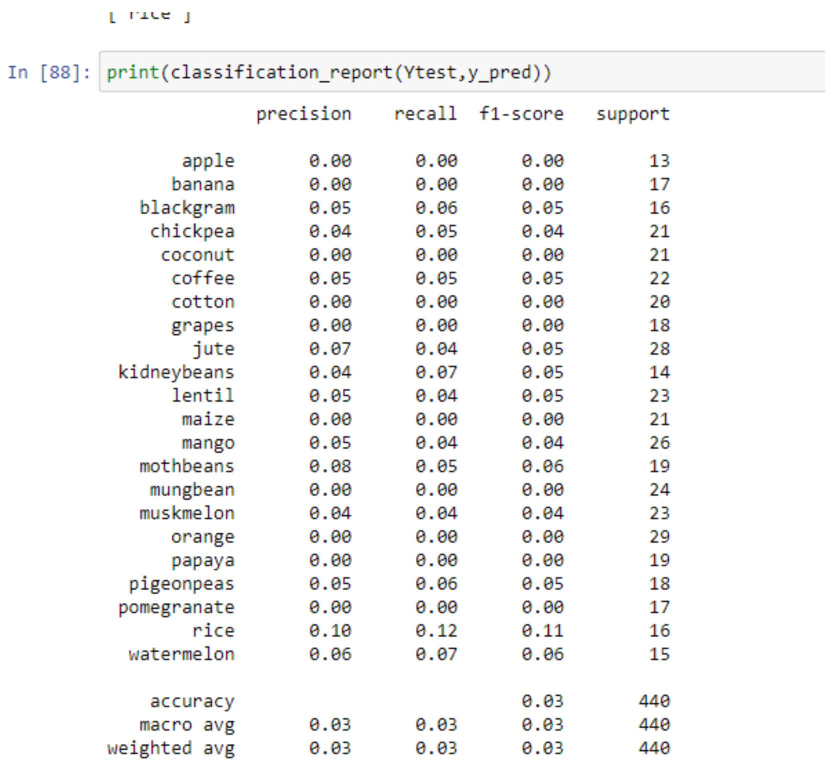
```
In [22]: # Cross validation score (Random Forest)
         score = cross_val_score(RF,features,target,cv=5)
         score
```

```
---------------------------------------------------------------------------
```

- **KNN Accuracy report**



```
In [10]: classifier = KNeighborsClassifier(n_neighbors=K_opt+1)
         classifier.fit(xtrain, ytrain)
         y_pred = classifier.predict(xtest)

         accuracy = acc(ytest, y_pred)*100
         print('Accuracy of the training Model : ', round(accuracy, 3), '%')

         Accuracy of the training Model :  98.636 %
```



```
        [ rice ]

In [88]: print(classification_report(Ytest,y_pred))

              precision    recall  f1-score   support

       apple       0.00      0.00      0.00        13
      banana       0.00      0.00      0.00        17
   blackgram       0.05      0.06      0.05        16
    chickpea       0.04      0.05      0.04        21
     coconut       0.00      0.00      0.00        21
      coffee       0.05      0.05      0.05        22
      cotton       0.00      0.00      0.00        20
      grapes       0.00      0.00      0.00        18
        jute       0.07      0.04      0.05        28
 kidneybeans       0.04      0.07      0.05        14
      lentil       0.05      0.04      0.05        23
       maize       0.00      0.00      0.00        21
       mango       0.05      0.04      0.04        26
   mothbeans       0.08      0.05      0.06        19
    mungbean       0.00      0.00      0.00        24
   muskmelon       0.04      0.04      0.04        23
      orange       0.00      0.00      0.00        29
      papaya       0.00      0.00      0.00        19
  pigeonpeas       0.05      0.06      0.05        18
 pomegranate       0.00      0.00      0.00        17
        rice       0.10      0.12      0.11        16
  watermelon       0.06      0.07      0.06        15

    accuracy                           0.03       440
   macro avg       0.03      0.03      0.03       440
weighted avg       0.03      0.03      0.03       440
```

Based on the above accuracy conclusions we chose the Random Forest Classifier algorithm present in the SKL library.

- **Hyperparameters used in our model testing :**

Hyperparameters used in Random Forest Classifier

1. n_estimator:

This is the number of trees in the forest. With an increase in the number of trees, the precision of the outcome increases - therefore better accuracy, and overfitting is reduced. However, this will make your model slower - therefore choosing an n_estimator value which your processor can handle allows your model to be more stable and perform well. We have declared the value for n_estimator as 20 considering the above factors.

2. Random_state:

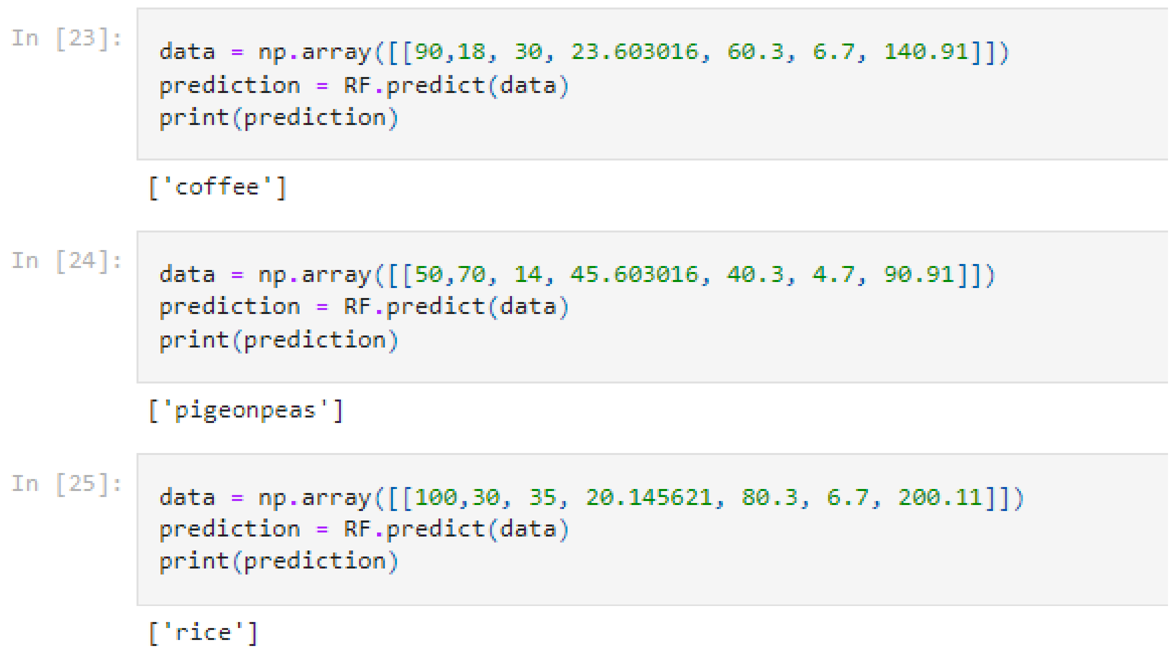
This parameter controls both the randomness of the bootstrapping of the samples used when building trees and the sampling of the features to consider when looking for the best split at each node. random_state is declared as zero for our model.

<u>Process followed for model building:</u>

- Collection of data.

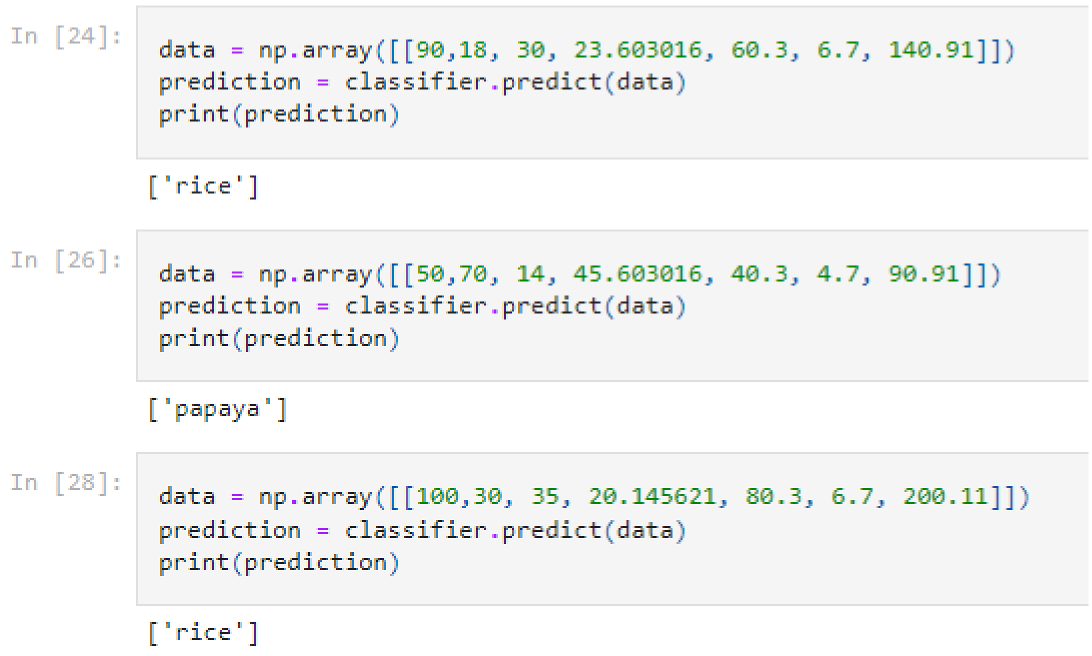
- Training the chosen model (RFC) by passing the prepared data split for testing.
- Evaluating the model by passing the data split for testing.
- Making predictions based on unseen parameters.

Predictions used by the Trained-Model (RFC):-



```
In [23]:

data = np.array([[90,18, 30, 23.603016, 60.3, 6.7, 140.91]])
prediction = RF.predict(data)
print(prediction)

['coffee']

In [24]:

data = np.array([[50,70, 14, 45.603016, 40.3, 4.7, 90.91]])
prediction = RF.predict(data)
print(prediction)

['pigeonpeas']

In [25]:

data = np.array([[100,30, 35, 20.145621, 80.3, 6.7, 200.11]])
prediction = RF.predict(data)
print(prediction)

['rice']
```

Prediction used by the Trained KNN model:



```
In [24]:
data = np.array([[90,18, 30, 23.603016, 60.3, 6.7, 140.91]])
prediction = classifier.predict(data)
print(prediction)

['rice']

In [26]:
data = np.array([[50,70, 14, 45.603016, 40.3, 4.7, 90.91]])
prediction = classifier.predict(data)
print(prediction)

['papaya']

In [28]:
data = np.array([[100,30, 35, 20.145621, 80.3, 6.7, 200.11]])
prediction = classifier.predict(data)
print(prediction)

['rice']
```

Tech used for model building:

Python Programming Language, Jupyter Notebook, SKL library, Pandas Library.

Github Repository link:

https://github.com/isosceles45/61_BetaSquad_2