Lab -3

PRML
AY 2020-21 Trimester - III

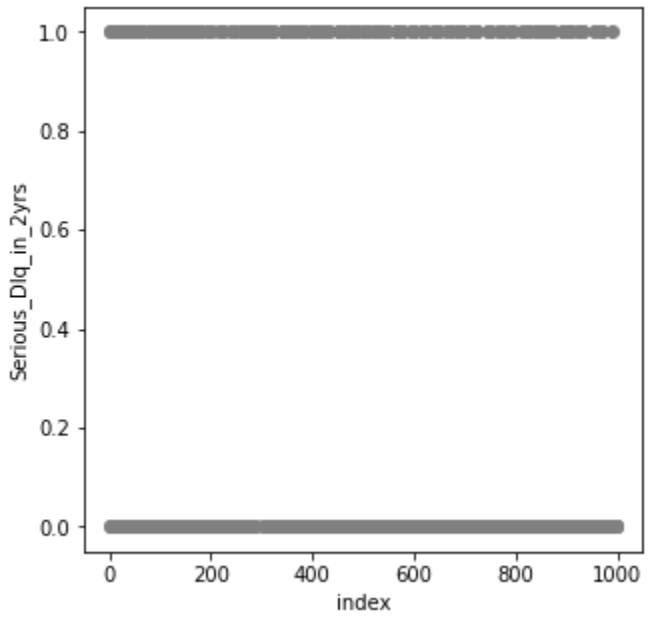
March 21, 2021

Random Forest and Bagging **J Jahnavi - B19CSE109**

```
RandomForestClassifier
```

1.Preprocessing the data:

a. Plot the distribution of the target variable:




```
plt.scatter(data.index , data['Serious_Dlq_in_2yrs'],color='grey')
```

A scatter plot is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data, the above plot depicts the distribution of the target variable for each row in the dataset.

b. Handle the NaN values:
   - Data contained 186 NaN records in ‘Monthly income’ and 23 NaN values in ‘Dependents’ columns which would be difficult to handle for our learning model.
   - Replaced the NaN values with the means of their respective columns as follows:

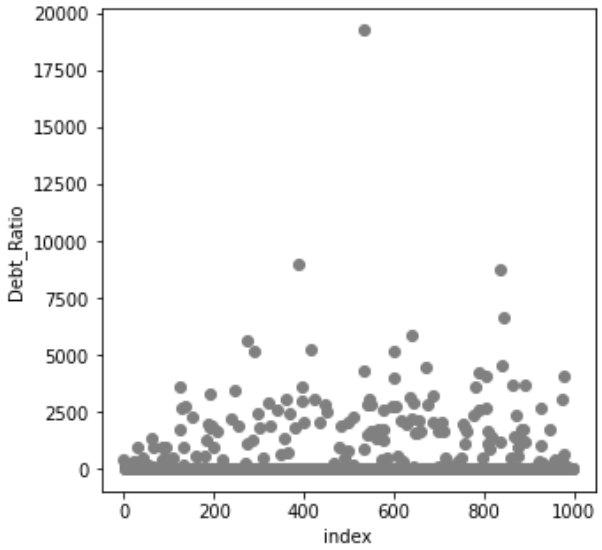
```
data.isna().sum()
```

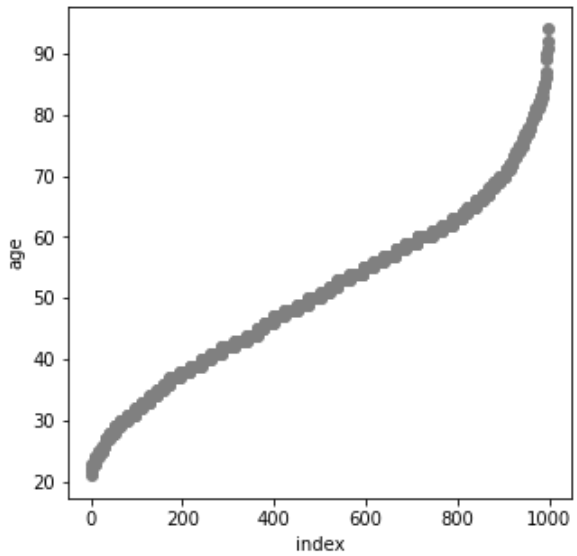
```
Serious_Dlq_in_2yrs                   0
Debt_Ratio                            0
age                                   0
Number_Of_Time_30-59_Days_Past_Due    0
Number_Of_Time_60-89_Days_Past_Due    0
Number_Of_Times_90_Days_Late          0
Monthly_Income                      186
Dependents                           23
dtype: int64
```

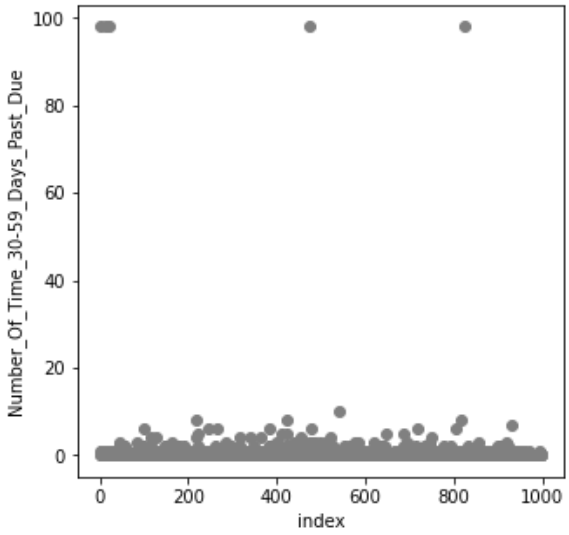
```
[261] # Handle the NaN values
data['Monthly_Income'].fillna((data['Monthly_Income'].mean()), inplace=True)
data['Dependents'].fillna((data['Dependents'].mean()), inplace=True)
data.isna().sum() #verifying
```

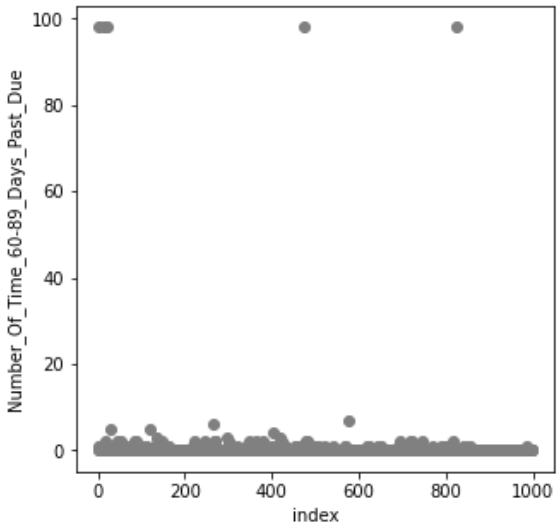
```
Serious_Dlq_in_2yrs                   0
Debt_Ratio                            0
age                                   0
Number_Of_Time_30-59_Days_Past_Due    0
Number_Of_Time_60-89_Days_Past_Due    0
Number_Of_Times_90_Days_Late          0
Monthly_Income                        0
Dependents                            0
dtype: int64
```

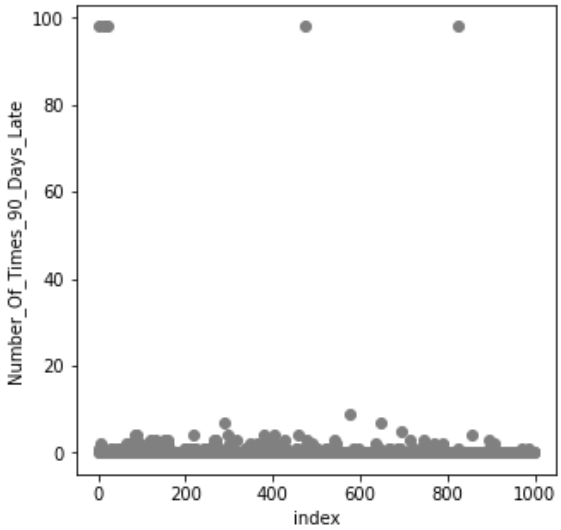
c. Visualize the distribution of data for every feature:

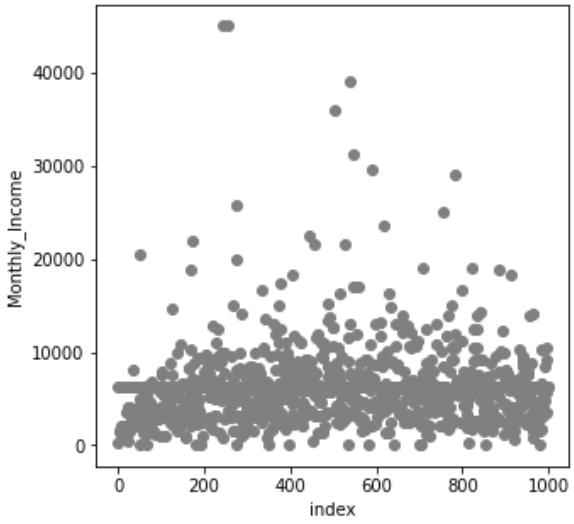


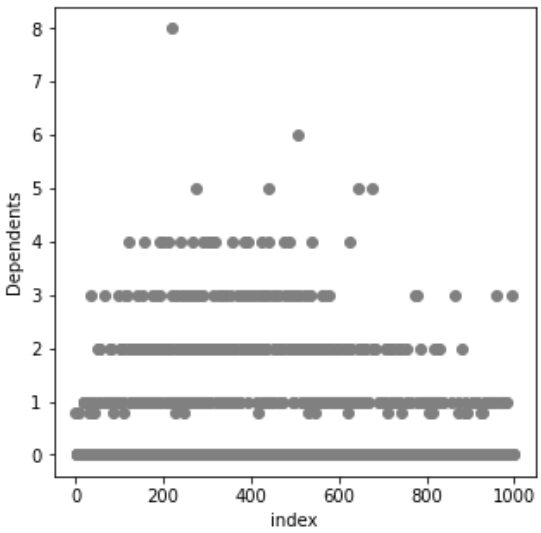



Number_Of_Time_30-59_Days_Past_Due
index
0
20
40
60
80
100
0
200
400
600
800
1000


Number_Of_Time_60-89_Days_Past_Due
index
0
20
40
60
80
100
0
200
400
600
800
1000


Number_Of_Times_90_Days_Late
index
0
20
40
60
80
100
0
200
400
600
800
1000





- The above plots depict the distribution of data for each feature
- Y-axis: feature and X-axis: index
- These plots were plotted using the steps below:

```
from matplotlib import pyplot as plt
for x in data.columns:
  if x != "Serious_Dlq_in_2yrs":
    plt.figure(figsize=(5,5))
    plt.xlabel("index")
    plt.ylabel(x)
    plt.scatter(data.index,data[x],color='grey')
    plt.show()
```

2.Train the Random Forest Classifier with the different parameters

- Trained the random forest classifier manually with different parameters
- Performed final model selection after manual check of parameters
- The model has a prediction accuracy of `0.8222222222222222`
- Upon Stratified KFold Cross-validation, average testing accuracies were found to be :`[0.82, 0.825, 0.755, 0.805, 0.7989949748743719]`

3.Perform 5 fold cross-validation and look at the ROC AUC against different values of the parameters and Perform the grid-search for the parameters to find the optimal value of the parameters.

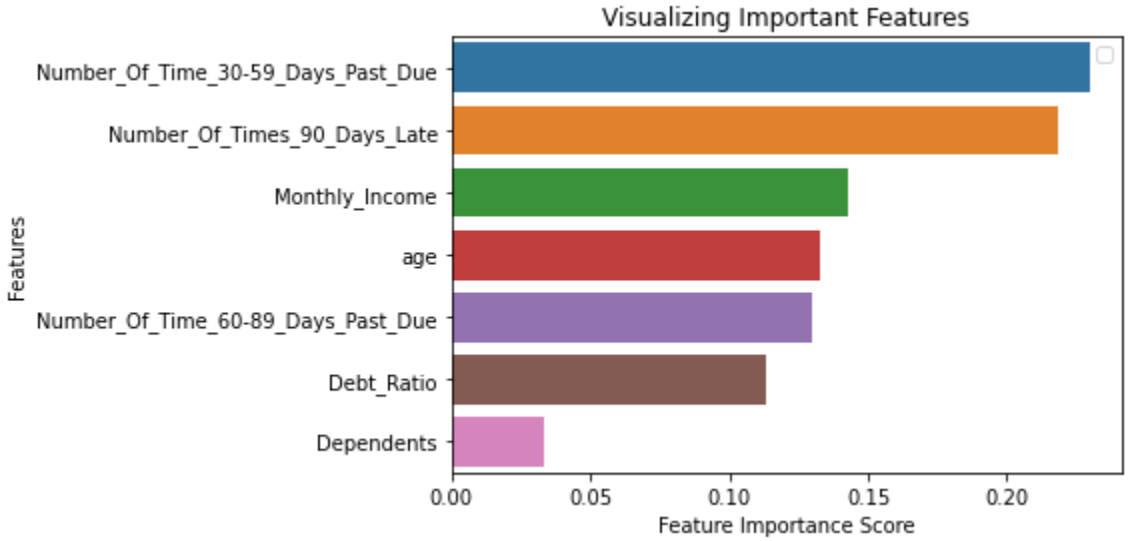
- Using GridSearchCV, performed grid search with **'roc_auc'** scoring and stratified K fold cross validation
- Upon cross validation, the predicted accuracies are : `array([0.82446429, 0.76470238, 0.87071429, 0.80029762, 0.78770983])`
    - Which implies, on an average the model gives an accuracy above 78%
- The best parameters from the given are : `{'max_depth': 4, 'max_features': 4, 'min_samples_leaf': 7, 'n_estimators': 50}`

4.The best score from grid search : `0.8132068452380954`

```
[490] #Final Trained Model
      model.fit(x_train,y_train)

      RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                             criterion='gini', max_depth=3, max_features=4,
                             max_leaf_nodes=None, max_samples=None,
                             min_impurity_decrease=0.0, min_impurity_split=None,
                             min_samples_leaf=1, min_samples_split=2,
                             min_weight_fraction_leaf=0.0, n_estimators=50,
                             n_jobs=None, oob_score=False, random_state=2, verbose=0,
                             warm_start=False)
```

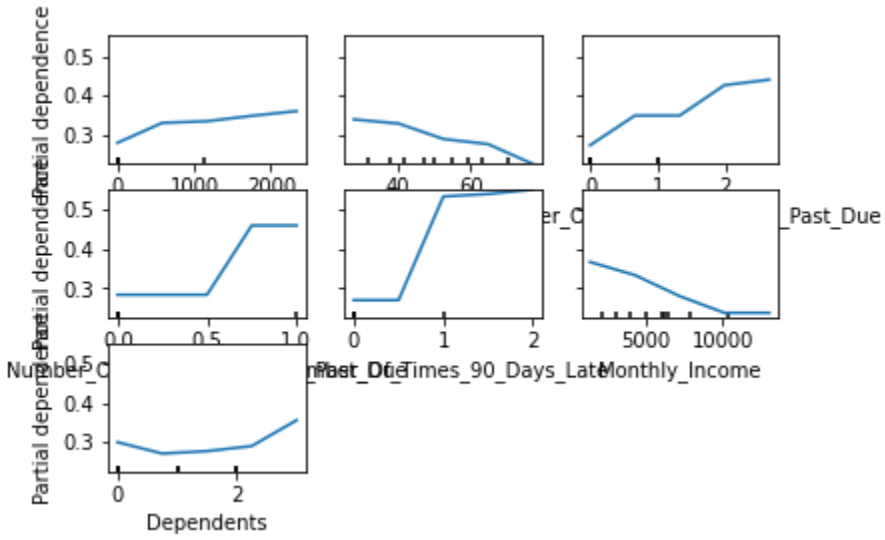
5.Feature which has the weakest impact in Random Forest Model: 'Dependents'

- Based on important feature index (plot shown below), the feature 'Dependents' has the weakest impact on the random forest model



- Based on the partial dependence of the output on the input features(plots shown below), the output shows to have the least dependence on the feature

'Dependents'



Plot of partial dependencies

```
BaggingClassifier
```

6.Perform bagging-based classification using Decision Tree as the base classifier:

a. The number of trees to be considered is {2,3,4}.

```
[584] # get a list of models to evaluate
      def get_models():
        models = dict()
        # define number of trees to consider
        n_trees = [2,3,4]
        for n in n_trees:
          models[str(n)] = BaggingClassifier(n_estimators=n)
        return models
```

b. Perform 5 fold cross-validation using ROC AUC metric to evaluate the models and collect the cross-validation scores(implemented as follows):

```
# evaluate a given model using cross-validation
def evaluate_model(model, x, y):
  # define the evaluation procedure
  cv = StratifiedKFold(n_splits=5)
  # evaluate the model and collect the results
  scores = cross_val_score(model, x, y, scoring='accuracy', cv=cv, n_jobs=-1)
  return scores
```

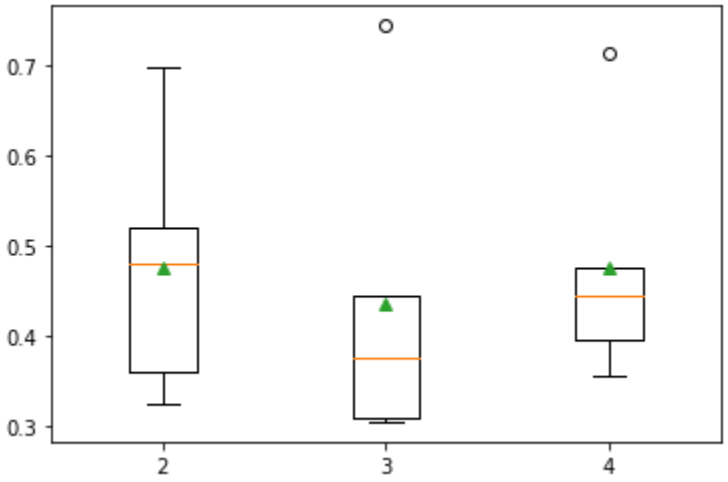
Number of trees	cross_val_score
2	[0.325 0.52 0.36 0.48 0.69849246]
3	[0.305 0.375 0.445 0.31 0.74371859]
4	[0.445 0.475 0.355 0.395 0.71356784]

c. Summarize the performance by getting mean and standard deviation of scores

```
# Summarize the performance by getting mean and standard deviation of scores
from matplotlib import pyplot
models = get_models()

results, names, means = list(), list(), list()
for name, model in models.items():

 scores = evaluate_model(model, x, y)
 print(scores)

 results.append(scores)
 names.append(name)
 means.append(scores.mean())

 print('At  %s, mean is %.3f and standard deviation is (%.3f)' % (name, scores.mean(), scores.std()))
 print("\n")
```

Number of trees	Mean	Standard deviation
2	0.477	0.132
3	0.436	0.162
4	0.477	0.125

d. Plot the model performance for comparison using boxplot.



Box plot is used to show the shape of the distribution, its central value, and its variability

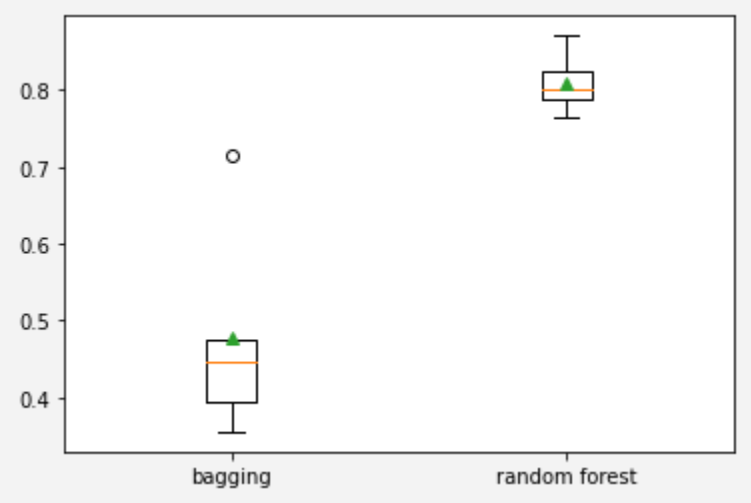
7.Compare the best performance of bagging with random forest by plotting using boxplot:

```
# Compare the best performance of bagging with random forest
i = means.index(max(means))
rf_score = clf.score(x_test,y_test)
print('The best performance of bagging is achieved at',(i+1)/10)
print('Accuracy of bagging model: ',max(means))
print('Accuracy of random forest model: ', rf_score)
print('\n')

if rf_score>max(means):
  print('Random forest model performed better')
elif max(means)>rf_score:
  print('Bagging model performed bettter')
else:
  print('Both models were equally good')
```

```
The best performance of bagging is achieved at 0.3
Accuracy of bagging model:  0.47671356783919594
Accuracy of random forest model:  0.7587939698492462


Random forest model performed better
```

This shows that Random forest model performed better than Bagging model

Using boxplot:



8.Upon tuning the bagging model, score has improved to `0.7688442211055276` from `0.47671356783919594`