**Assignment 2(Hackathon)**

**Priya Bhatia (AI20MTECH14015)**

**Akshita Parekh (CS20RESCH14001)**

**Libraries Used:**

* pandas
* seaborn
* numpy
* sklearn
* matplotlib

**Running Code:**

The sections are commented appropriately with suitable headings and in order to be executed in the notebook, please run them in specified order below.

Given:

The dataset was provided in the Kaggle Competition site.

Dataset mainly contains three files namely:

* ../input/iith-foml-2020/iith\_foml\_2020\_train.csv
* ../input/iith-foml-2020/iith\_foml\_2020\_test.csv
* ../input/iith-foml-2020/iith\_foml\_2020\_sample.csv

**Training Dataset Preparation before model training:**

Step 1: Loading Training Dataset

The training dataset can be loaded with the help of pandas library by specifying appropriate path as follows:

train\_dataframe = pd.read\_csv('../input/iith-foml-2020/iith\_foml\_2020\_train.csv')

# To get the above file as train\_input

train\_dataframe.to\_csv('train\_input.csv',index=False)

train\_dataframe = pd.read\_csv('./train\_input.csv')

Initial shape of train\_dataframe was analysed using the below code: (994, 25)

train\_dataframe.shape

**Feature Engineering Tasks:**

Step 2: Analysis of null values in Training Dataset

To do feature engineering, find the number of null values in each feature using the below code:

train\_dataframe.isnull().sum()

After that drop those features which contain approximately more than 50% of the null values in any feature and we analyse that those features in training dataset are Feature 16, Feature 17 and Feature 18. So, drop these feature using below code:

train\_dataframe.drop(['Feature 16'],axis=1,inplace=True)

train\_dataframe.drop(['Feature 17'],axis=1,inplace=True)

train\_dataframe.drop(['Feature 18'],axis=1,inplace=True)

After that replace the null values in features with mean and median for those features which contains less than 50% of the null values in any feature and we analyse that those features in training dataset are Feature 9, Feature 10, Feature 11, Feature 12, Feature 13, Feature 14, Feature 15, Feature 24.

So, replace the null values in feature using below code:

train\_dataframe['Feature 10']=train\_dataframe['Feature 10'].fillna(train\_dataframe['Feature 10'].median())

train\_dataframe['Feature 11']=train\_dataframe['Feature 11'].fillna(train\_dataframe['Feature 11'].median())

train\_dataframe['Feature 12']=train\_dataframe['Feature 12'].fillna(train\_dataframe['Feature 12'].median())

train\_dataframe['Feature 13']=train\_dataframe['Feature 13'].fillna(train\_dataframe['Feature 13'].median())

train\_dataframe['Feature 24']=train\_dataframe['Feature 24'].fillna(train\_dataframe['Feature 24'].median())

train\_dataframe['Feature 9']=train\_dataframe['Feature 9'].fillna(train\_dataframe['Feature 9'].mean())

train\_dataframe['Feature 15']=train\_dataframe['Feature 15'].fillna(train\_dataframe['Feature 15'].mean())

Step 3: Analysis of highly correlated features in Training Dataset

Firstly, correlation matrix was created using below code:

corr\_matrix = train\_dataframe.corr().abs()

print(corr\_matrix)

Secondly, print those features which are highly correlated (More than 95%) using below code:

upper = corr\_matrix.where(np.triu(np.ones(corr\_matrix.shape),k=1).astype(np.bool))

to\_drop = [column for column in upper.columns if any(upper[column] > 0.95)]

print(to\_drop)

Lastly, drop all highly correlated features using below code:

train\_dataframe.drop(['Feature 14'],axis=1,inplace=True)

After doing all above tasks out train\_dataframe shape is having: (994,21)

**Testing Dataset Preparation before model testing:**

Step 4: Loading Testing Dataset

The testing Dataset can be loaded with the help of pandas library by specifying appropriate path as follows:

test\_dataframe = pd.read\_csv('../input/iith-foml-2020/iith\_foml\_2020\_test.csv')

# To rename the above file to train\_input

test\_dataframe.to\_csv('test\_input.csv',index=False)

test\_dataframe = pd.read\_csv('./test\_input.csv')

Initial shape of test\_dataframe was analysed using the below code: (426,24)

test\_dataframe.shape

**Feature Engineering Tasks:**

Step 5: Analysis of null values in Testing Dataset

To do feature engineering, find the number of null values in each feature using the below code:

test\_dataframe.isnull().sum()

After that drop those features which contain approx more than 50% of the null values in any feature and we analyse that those features in testing dataset are Feature 16, Feature 17, Feature 18. So, drop these feature using below code:

test\_dataframe.drop(['Feature 16'],axis=1,inplace=True)

test\_dataframe.drop(['Feature 17'],axis=1,inplace=True)

test\_dataframe.drop(['Feature 18'],axis=1,inplace=True)

After that replace the null values in features with mean and median for those features which contains less than 50% of the null values in any feature and we analyse that those features in testing dataset are Feature 9, Feature 15.

So, replace the null values in feature using below code:

test\_dataframe['Feature 9']=test\_dataframe['Feature 9'].fillna(test\_dataframe['Feature 9'].mean())

test\_dataframe['Feature 15']=test\_dataframe['Feature 15'].fillna(test\_dataframe['Feature 15'].mean())

Step 6 : Analysis of highly correlated features in Testing Dataset

Firstly, correlation matrix was created using below code:

corr\_matrix\_test = test\_dataframe.corr().abs()

print(corr\_matrix\_test)

Secondly, print those features which are highly correlated (More than 95%) using below code:

upper = corr\_matrix\_test.where(np.triu(np.ones(corr\_matrix\_test.shape),k=1).astype(np.bool))

to\_drop = [column for column in upper.columns if any(upper[column] > 0.95)]

print(to\_drop)

Lastly, drop all highly correlated features using below code:

test\_dataframe.drop(['Feature 14'],axis=1,inplace=True)

After doing all above tasks out test\_dataframe shape is having: (426,20)

**Model Training Task:**

Step 7: Specifying X\_train and y\_train after data preparation

As Target Variable (Discrete) is the predicted values which we want to predict. Thus, in training part it was stored in y\_train and remaining features stored in X\_train which was the input in testing dataset. So, this task is achieved using below code:

X\_train = train\_dataframe.drop(['Target Variable (Discrete)'],axis=1)

y\_train = train\_dataframe['Target Variable (Discrete)']

Step 8: Specifying X\_test in which test\_dataframe was stored

X\_test = test\_dataframe

Step 9: Hyperparameter tuning using GridSearchCV (This step took some time as training was done on this step)

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import GridSearchCV

param\_grid = [

{ 'n\_estimators' : [int(x) for x in np.linspace(start = 20, stop = 1000, num = 5)], 'max\_features': [5, 10],

'max\_depth': [int(x) for x in np.linspace(1, 100, num = 3)], 'bootstrap': [True, False]}

]

forest = RandomForestClassifier(n\_jobs=-1)

grid\_search\_forest = GridSearchCV(forest, param\_grid, cv=10, scoring='neg\_mean\_squared\_error')

grid\_search\_forest.fit(X\_train, y\_train)

Step 10: Analysis of best estimator using GridSearchCV for RandomForest Classifier using below code

grid\_search\_forest.best\_estimator\_

Step 11: Prediction on X\_test data using above model

rf\_grid = grid\_search\_forest.best\_estimator\_.predict(X\_test)

Step 12: Final Step to generate test\_output.csv file which contains Id and Category (Predicted values)

pred=pd.DataFrame(rf\_grid)

pred.index = np.arange(1, len(pred) + 1)

pred.columns=['Id']

pred.reset\_index(inplace=True)

pred.columns=['Id','Category']

pred.to\_csv('test\_output.csv',index=False)

**Final Best Score Generated by Kaggle on submission of test\_output file: 0.93896(Public Leaderboard)**

**Final Best Score Generated by Kaggle on submission of test\_output file: 0.94366 (Private Leaderboard)**