MACHINE LEARNING ASSIGMENT

TASK 1: BECOME A MACHINE LEARNING SUPERSTAR

Lecturer Name: Dr. David Leonard

Student Name: Deepshikha Wadikar

Student Number: D17128916

Course Code: DT228A

Year: 2018-19

1. INTRODUCTION

This report is intended to provide summary of Machine Learning model to build the loan status prediction model. The code is written in Python Note book. Kaggle is a place for all data science and machine learning projects. Bank Loan Status Dataset is downloaded from Kaggle website as a part of this assignment. The link for Dataset is: https://www.kaggle.com/zaurbegiev/my-dataset

The dataset includes training and test dataset with 18 predictor variables listed in below Table 1.1.

VARIABLE	DESCRIPTION	DATA TYPE
Loan ID	The unique ID of Loan	Integer
Customer ID	The unique ID of Customer	Integer
Current Loan Amount	Loan Amount	Integer
Term	Loan Term	String
Credit Score	Credit Score of Customer	Integer
Annual Income	Annual Income of Customer	Integer
Years in current job	Years in current job	Integer
Home Ownership	Home Ownership	String
Purpose	Purpose of the Loan	String
Monthly Debt	Monthly Debt	Integer
Years of Credit History	Number of Years	Integer
Months since last delinquent	Fail to pay	Integer
Number of Open Accounts	Total number of active accounts	Integer
Number of Credit Problems	Number of Credit problems	Integer
Current Credit Balance	Current Credit balance	Integer
Maximum Open Credit	Maximum Open Credit	String
Bankruptcies	Bankruptcies details	Double
Tax Liens	Tax Liens	String
Loan Status	Status of Loan whether Paid Fully or charged off	String

Table 1.1 Bank Loan Dataset Overview

2. EXISTING CONTRIBUTIONS

There are 3 kernels and 1 discussion exists for the given Bank Loan Status Dataset. In the existing kernels the data cleaning is performed and then classification models are built to predict the Loan Status for future customers. Logistic Regression, K Neighbor Classifier, XGB Classifier and SGDC Classifier models were built on the dataset to find the prediction. Logistic Regression, K Neighbor Classifier and XGB Classifier predicted the Loan Status with an accuracy of 77.15% (approx.) whereas SGDC Classifier model predicted with an accuracy little lower of 74% (approx.).

In one of the kernel Exploratory Data Analysis has been performed on the Bank Loan Status Dataset. The author has also listed few learning after performing EDA on the dataset. The data should be correct before performing any imputations. Imputations should be performed in such a way that it should not affect the original distribution. Always remove data which contains high percentage of missing data and are negligible interest for predicting the target variable. Visualizations are performed using matplotlib and seaborn libraries.

In the third kernel five different machine learning Classification Models – Logistic Regression, k-nearest neighbors classifier, Naïve Bayes and Random Forest Classification were compared to find the prediction results. It has been observed that Logistic Regression and Gradient Boosting models have highest model accuracy followed by Random forest, KNN and Naïve Bayes respectively. Also, in this kernel the Feature Importance for Random Forest is determined. In the existing kernels the Logistic Regression model accuracy was noted as 81.97%, Gradient Boosting as 81.98%, Random Forest as 80.17%, K-NN as 79.03% and Naïve Bayes as 41.93% being the lowest.

3. EXISTING RESEARCH GAP AND FINDINGS

In the Existing kernels it has been observed that the missing values were not handled correctly and instead of imputing the missing values those were simply removed from the dataset.

```
#Determining the missing values
loanStatusData.isnull().sum()
```

```
Loan Status                     514
Current Loan Amount             514
Term                            514
Credit Score                  19668
Annual Income                 19668
Years in current job           4736
Home Ownership                  514
Purpose                         514
Monthly Debt                    514
Years of Credit History         514
Months since last delinquent  53655
Number of Open Accounts         514
Number of Credit Problems       514
Current Credit Balance          514
Maximum Open Credit             516
Bankruptcies                    718
Tax Liens                       524
dtype: int64
```

Figure 3.1: Missing Values

In every column we can see there are missing values present. In the column ‘Months since last delinquent’ more than 50% are missing so it is difficult to impute those missing values as it will make the data bias. In the existing kernels all these instances were dropped from data set.

4. DATA PRE-PROCESSING

4.1 LOADING DATA SET

The csv file is loaded in python using pandas library.

```
loanStatusData = pd.read_csv('C:/Users/Dell/Downloads/MSc 2nd_Sem/Machine Learning/Assignment/credit_train.csv')
```

```
loanStatusData.shape
```

```
(100514, 19)
```

Figure 4.1.1: Loading Dataset and rows and columns details

4.2 DATA PREPARATION

Data Preparation is the process of cleaning data before performing statistical analysis. It helps in finding the awkward data i.e., missing data.

describe() method is used to display the count, mean, standard deviation etc of numeric data.

```
loanStatusData.describe()
```

	Current Loan Amount	Credit Score	Annual Income	Monthly Debt	Years of Credit History	Months since last delinquent	Number of Open Accounts	Number of Credit Problems	Current Credit Balance	Maximum Open Credit	Ban
count	1.000000e+05	80846.000000	8.084600e+04	100000.000000	100000.000000	46859.000000	100000.00000	100000.000000	1.000000e+05	9.999800e+04	997
mean	1.176045e+07	1076.456089	1.378277e+06	18472.412336	18.199141	34.901321	11.12853	0.168310	2.946374e+05	7.607984e+05	
std	3.178394e+07	1475.403791	1.081360e+06	12174.992609	7.015324	21.997829	5.00987	0.482705	3.761709e+05	8.384503e+06	
min	1.080200e+04	585.000000	7.662700e+04	0.000000	3.600000	0.000000	0.00000	0.000000	0.000000e+00	0.000000e+00	
25%	1.796520e+05	705.000000	8.488440e+05	10214.162500	13.500000	16.000000	8.00000	0.000000	1.126700e+05	2.734380e+05	
50%	3.122460e+05	724.000000	1.174162e+06	16220.300000	16.900000	32.000000	10.00000	0.000000	2.098170e+05	4.678740e+05	
75%	5.249420e+05	741.000000	1.650663e+06	24012.057500	21.700000	51.000000	14.00000	0.000000	3.679588e+05	7.829580e+05	
max	1.000000e+08	7510.000000	1.655574e+08	435843.280000	70.500000	176.000000	76.00000	15.000000	3.287897e+07	1.539738e+09	

Figure 4.2.1: Output of describe() method

4.3MISSING VALUES

As discussed in the existing kernels the attributes and instances with missing values were all dropped from the data set and then model is built. In this I have handled all those missing values. As there are more than 50% missing values so 'Months since last delinquent' column has been dropped from the data set. Also it has been noticed that 514 rows in every column are missing so those rows were dropped and the final missing value count are displayed below –

```
#Now dropping the last 514 rows from the dataframe
loanStatusData.drop(loanStatusData.tail(514).index,inplace = True)
loanStatusData.isnull().sum()
```

```
Loan Status                     0
Current Loan Amount             0
Term                            0
Credit Score                19154
Annual Income               19154
Years in current job         4222
Home Ownership                  0
Purpose                         0
Monthly Debt                    0
Years of Credit History         0
Number of Open Accounts         0
Number of Credit Problems       0
Current Credit Balance          0
Maximum Open Credit             2
Bankruptcies                  204
Tax Liens                      10
dtype: int64
```

Figure 4.3.1: Details of Missing Values

Now the attributes 'Credit Score' and 'Annual Income' were numeric attributes so those missing values are imputed by the mean of the respective column values. The attributes 'Bankruptcies' and 'Tax Liens' were imputed by the fillna method. Lastly the column 'Maximum Open Credit' has only 2 missing values so those rows/instances were removed from the data set.

5. FEATURE ENGINEERING

Before going ahead with modeling we have to select the appropriate features, create new features based on existing features and also to convert string attributes into numeric.

Here in this data set we have 'Loan Status' as our target variable which is present as text as 'Fully Paid' and 'Charged Off'. This is encoded as 1 and 0 in binary format using Label Encoder and set as Fully Paid = 1; Charged Off as 0. Also the column 'Term' is replaced as Short Term = 0; Loan Term = 1. All the remaining categorical variables were converted into dummy variables.

```
In [49]: #Encoding Categorical Data
         categorical_data = loanStatusData[['Years in current job','Home Ownership','Purpose']]
         categorical_subset = pd.get_dummies(categorical_data)

In [50]: loanStatusData.drop(labels=['Years in current job', 'Home Ownership', 'Purpose'], axis=1, inplace=True)
         loanStatusData = pd.concat([loanStatusData, categorical_subset], axis = 1)

In [51]: loanStatusData.head()

Out[51]:
```

	Loan Status	Current Loan Amount	Term	Credit Score	Annual Income	Monthly Debt	Years of Credit History	Number of Open Accounts	Number of Credit Problems	Current Credit Balance	...	Purpose_Medical Bills	Purpose_Other	Purpose_Take a Trip
0	1	445412.0	0	709.000000	1.167493e+06	5214.74	17.2	6.0	1.0	228190.0	...	0	0	0
1	1	262328.0	0	1076.382448	1.378271e+06	33295.98	21.1	35.0	0.0	229976.0	...	0	0	0
2	1	99999999.0	0	741.000000	2.231892e+06	29200.53	14.9	18.0	1.0	297996.0	...	0	0	0
3	1	347666.0	1	721.000000	8.069490e+05	8741.90	12.0	9.0	0.0	256329.0	...	0	0	0
4	1	176220.0	0	1076.382448	1.378271e+06	20639.70	6.1	15.0	0.0	253460.0	...	0	0	0

5 rows × 44 columns

Figure 5.1: Dummy Variable creation

Correlation between attributes are also determined using corr() function.

```
loanStatusData.corr()
```

	Loan Status	Current Loan Amount	Term	Credit Score	Annual Income	Monthly Debt	Years of Credit History	Number of Open Accounts	Number of Credit Problems	Current Credit Balance	...	Purpose_Medical Bills	P
Loan Status	1.000000	0.194626	-0.110665	-0.410974	0.046741	-0.007903	0.023709	-0.011987	-0.002371	0.009634	...	-0.005178	
Current Loan Amount	0.194626	1.000000	-0.059011	-0.095142	0.012911	-0.006640	0.019283	0.001471	-0.002797	0.003875	...	0.000129	
Term	-0.110665	-0.059011	1.000000	0.031430	0.078143	0.158648	0.041507	0.082594	-0.026164	0.104711	...	-0.022886	
Credit Score	-0.410974	-0.095142	0.031430	1.000000	-0.017100	-0.001578	-0.008756	0.005828	-0.002731	-0.000060	...	0.001570	
Annual Income	0.046741	0.012911	0.078143	-0.017100	1.000000	0.438402	0.145173	0.131940	-0.015451	0.284952	...	0.002783	
Monthly Debt	-0.007903	-0.006640	0.158648	-0.001578	0.438402	1.000000	0.199280	0.411359	-0.055381	0.481363	...	-0.006750	
Years of Credit History	0.023709	0.019283	0.041507	-0.008756	0.145173	0.199280	1.000000	0.132347	0.061589	0.208473	...	0.011972	
Number of Open Accounts	-0.011987	0.001471	0.082594	0.005828	0.131940	0.411359	0.132347	1.000000	-0.014002	0.228124	...	-0.014352	
Number of Credit Problems	-0.002371	-0.002797	-0.026164	-0.002731	-0.015451	-0.055381	0.061589	-0.014002	1.000000	-0.112523	...	0.006734	
Current Credit Balance	0.009634	0.003875	0.104711	-0.000060	0.284952	0.481363	0.208473	0.228124	-0.112523	1.000000	...	-0.015128	
Maximum Open Credit	0.008404	-0.001271	0.008348	-0.002089	0.039205	0.039268	0.031124	0.031341	-0.012072	0.139204	...	-0.001917	
Bankruptcies	0.006330	-0.000793	-0.028900	-0.006263	-0.042795	-0.078686	0.065914	-0.024509	0.751909	-0.122186	...	0.001163	
Tax Liens	-0.010228	-0.002048	-0.003424	0.004752	0.037032	0.020129	0.017239	0.006541	0.581268	-0.015651	...	0.003657	
Years in current job_1 year	-0.002977	-0.002328	-0.020378	0.007413	-0.017548	-0.034972	-0.062939	-0.008193	-0.015364	-0.021787	...	0.000846	
Years in current job_10+ years	0.000276	-0.002917	0.043782	-0.002071	0.041555	0.086138	0.258715	0.028244	0.046608	0.084928	...	0.001523	

Figure 5.2: Result of corr() method

6. MODELLING

Prediction model Decision Tree is built on Bank Loan Status dataset. I have chosen Decision Tree model as this model was not built for prediction in the existing research kernels.

DECISION TREE - A decision tree is a tree like structure in which each internal node applies a test on an attribute each branch represents the outcome of the test results, and each leaf node represents a class label. Decision Tree built on the Loan Status dataset has an accuracy of **81.62%** which is quite good. **DecisionTreeClassifier** library is imported for building Decision Tree model.

7. MODEL EVALUATION

This phase focuses on evaluating the model. Classification model is evaluated on the basis of Confusion Matrix, Accuracy, Precision, Recall and F-measure.

Confusion Matrix and Accuracy

Confusion Matrix is built using sklearn.metrics by importing confusion_matrix.

Accuracy of the model is calculated by importing accuracy_score from sklearn.metrics. The accuracy is calculated as the ratio of total correct predictions and total number of predictions.

Accuracy = (True Positive + True Negative)/(True Positive + True Negative + False Positive + False Negative)

```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test,predictions) * 100

print("Accuracy using Decision Tree: ",round(accuracy,2),"%")

Accuracy using Decision Tree:  81.62 %

from sklearn.metrics import confusion_matrix
confusionMatrixDT = confusion_matrix(y_test,predictions)
print('Confusion Matrix')
print(confusionMatrixDT)

Confusion Matrix
[[ 1036  3555]
 [  120 15289]]
```

Figure 7.1: Decision Tree Model Accuracy and Confusion Matrix

Using the Confusion Matrix the Precision, Recall and F1-score is calculated using Confusion Matrix.

Precision – is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved.

Precision = True Positives/(True positives + False positives)

Recall - is the ratio of the number of relevant records retrieved to the total number of relevant records in the database.

Recall = True Positive/(True positive + False Negative)

F-measure - is a measure of test's accuracy. It contains both precision p and recall r of the test to compute the F-score or F measure of the model.

F-Measure = 2* (precison*recall)/(precision+recall)

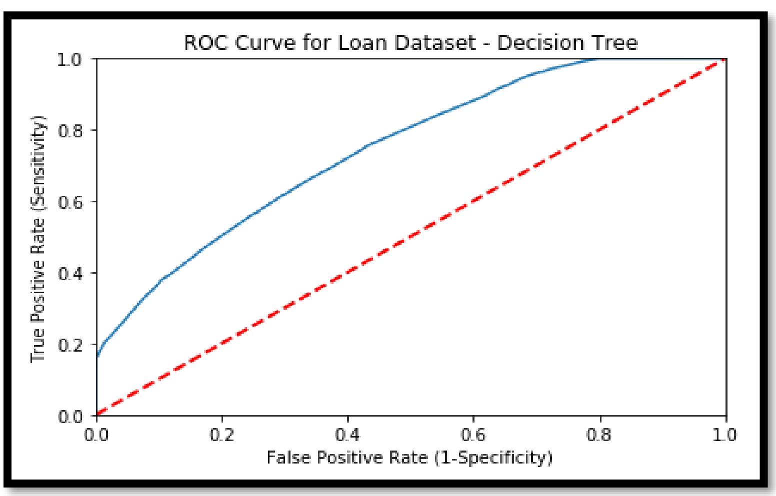

Figure 7.2: ROC Curve - Decision Tree Model

Hyperparameter Tuning with Randomized Search is performed on the Decision Tree and slightly better results were obtained and accuracy is 81.97%. The Cross Validation is set as 5. The gini criteria is the best to get the results. The details are in the below snap –

```
#Performing Hyperparameter tuning with Randomized Search
#Importing libraries
from scipy.stats import randint
from sklearn.model_selection import RandomizedSearchCV

#Setup Parameters and distributions to sample from: param_dist
param_dist = {"max_depth" :[3,None],
              "min_samples_leaf":(1,8),
              "criterion":["gini","entropy"]}

#Instantiate a Decision Tree Classifier: tree
tree = DecisionTreeClassifier()

#Instantiate the RandomizedSearchCV object: tree_cv
tree_cv = RandomizedSearchCV(tree,param_dist,cv=5,n_iter = 7)

#Fit it to the data
tree_cv.fit(X_train,y_train)

RandomizedSearchCV(cv=5, error_score='raise',
          estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'),
```

Figure 7.3: Randomized Search CV Decision Tree Model

```
#Print the tuned parameters and score
print("Tuned Decision Tree Parameters: {}".format(tree_cv.best_params_))
print("Best score is: {}".format(tree_cv.best_score_))

Tuned Decision Tree Parameters: {'min_samples_leaf': 8, 'max_depth': 3, 'criterion': 'gini'}
Best score is: 0.8197579939498487

#Predicting the Test Results
y_pred = tree_cv.predict(X_test)

y_pred

array([1, 1, 1, ..., 1, 1, 1], dtype=int64)

#Making Confusion Matrix
confusionMatrixRandomized = confusion_matrix(y_test,y_pred)
print('Confusion Matrix')
print(confusionMatrixRandomized)

Confusion Matrix
[[  923  3668]
 [    0 15409]]
```

Figure 7.4: Randomized Search CV Model Accuracy and Confusion matrix

8. DISCUSSION

The Decision Tree model accuracy (81.62%) has been observed as better than the models in the existing research. This is the first time the Decision Tree model is built on the Bank Loan Status Dataset. When the Randomized Search CV is performed on Decision Tree the accuracy is even improved and better than the results of existing research. After Randomized Search CV the accuracy is reached to 81.97%. This outperforms the existing research work also.