**LOAN APPLICATION STATUS PREDICTION**

LOANS are the major requirement of the modern world. By this only, Banks get a major part of the total profit. It is beneficial for students to manage their education and living expenses, and for people to buy any kind of luxury like houses, cars, etc.

But when it comes to deciding whether the applicant’s profile is relevant to be granted with loan or not. Banks have to look after many aspects.

So, here we will be using Machine Learning with Python to ease their work and predict whether the candidate’s profile is relevant or not using key features like Marital Status, Education, Applicant Income, Credit History, etc.

**Loan Approval Prediction using Machine Learning**

The dataset cantains 13 features:

1.Loan : A unique id

2.Gender : Gender of the applicant Male/female

3.Married : Marital Status of the applicant, values will be Yes/No.

4.Dependents : It tells whether the applicant has any dependents or not.

5.Education : It will tell us whether the applicant is Graduated or not.

6.Self\_Employed : This defines that the applicant is self-employed i.e. Yes/No

7.ApplicationIncome : Applicant income

8. CoapplicantIncome : Co-applicant income

9.LoanAmount : Loan amount (in thousands)

10.Loan\_Amount\_Term : Term of loan(in months)

11.Credit\_History : Credit history of individual’s repayment of their debts.

12.Property\_Area : Area of property i.e. Rural/Urban/Semi-urban

13.Loan\_Status : Status of Loan Approved or not i.e. Y-Yes, N-No

**Importin Libraries and Dataset**

Firstly we have to import libraries :

**Numpy** : Import Numpy

**Pandas :** To load the Dataframe

**Matplotlib** : To visualize the data features

**Seaborn :**To see the correlation between features using heatmap

**Train-Test-split** :Divide the dataset into training and testing set for prediction.

**Accuracy\_score, confusion\_matrix,** : For model evaluation.

**Classification\_report** : For model report.

**LogisticRegression** : Machine learning algorithm

**DecisionTreeClassifier** : Machine learning algorithm

**RandomForestClassifier** : Machine learning algorithm

**AdaBoostClassifier** : Machine learning algorithm

**Import the dataset and put it in a dataframe.**

df=pd.read\_csv(r"C:\Users\OM RAJ PANDEY\Desktop\DSData-master\DSData-master\loan\_prediction.csv")

Once we imported check the top 5 columns of dataset.

df.head()

**Data Preprocessing and Visualization**

**# number of rows and columns**

df.shape

* Check the shape of dataset which is 614 rows and 13 columns.
* Using describe method to check the statistical measures.
* #statistical measures
* df.describe()
* This function shows the description of numerical columns not the categorical.
* Check the missing values of each column using isnull function.
* #number of missing values in each column
* df.isnull().sum()
* Loan\_ID 0
* Gender 13
* Married 3
* Dependents 15
* Education 0
* Self\_Employed 32
* ApplicantIncome 0
* CoapplicantIncome 0
* LoanAmount 22
* Loan\_Amount\_Term 14
* Credit\_History 50
* Property\_Area 0
* Loan\_Status 0
* dtype: int64
* There are some null columns present in the dataset.
* We drop the null columns using dropna method.
* #dropping the missing values
* df=df.dropna()
* Again check the null columns, now no null value present in the dataset.
* df.isnull().sum()
* Loan\_ID 0
* Gender 0
* Married 0
* Dependents 0
* Education 0
* Self\_Employed 0
* ApplicantIncome 0
* CoapplicantIncome 0
* LoanAmount 0
* Loan\_Amount\_Term 0
* Credit\_History 0
* Property\_Area 0
* Loan\_Status 0
* dtype: int64
* There is no null column in the dataset
* Using Label encoding in Loan Status column replace ‘No’ with ‘0’ and ‘Yes’ with ‘1’
* #label encoding
* df.replace({'Loan\_Status':{'N':0, 'Y':1}},inplace=True)
* Again check the top 5 columns to see whether label encoding is applied or not.It is replaced.
* df.head()
* Check the values of dependent column using value\_count function.
* #Dependent column values
* df['Dependents'].value\_counts()
* 0 274
* 2 85
* 1 80
* 3+ 41
* Name: Dependents, dtype: int64
* Replace the value ‘3+’ of dependent column with ‘4’ using replace function.
* #replacing the value of 3+ to 4
* df=df.replace(to\_replace='3+', value=4)
* Now, again check the values of dependent columns whether it is replaced or not. It is replaced.
* #dependent values
* df['Dependents'].value\_counts()
* 0 274
* 2 85
* 1 80
* 4 41
* Name: Dependents, dtype: int64
* Make the count plot in between ‘Education’ and ‘Loan Status’ column.We see Graduate people have more chance to get the loan.
* # education and Loan Status
* sns.countplot(x='Education', hue='Loan\_Status', data=df)
* Make the count plot between ‘Married’ and ‘Loan\_Status’ column. We see married people have more chance to get the loan.
* #marital status & Loan Status
* sns.countplot(x='Married', hue='Loan\_Status', data=df)
* Convert categorical columns ‘Married’,’Gender’,’Self\_Employed’,’Property\_Area’,’Education’ into numerical columns using ‘replace’ function.
* # convert categorical columns to numerical values
* df.replace({'Married':{'No':0,'Yes':1},'Gender':{'Male':1,'Female':0},'Self\_Employed':{'No':0,'Yes':1},
* 'Property\_Area':{'Rural':0,'Semiurban':1,'Urban':2},'Education':{'Graduate':1,'Not Graduate':0}},inplace=True)
* Check the top 5 columns of dataset whether it is replaced or not. It is replaced.
* df.head()

**Model\_Building:**

* Saparate the data in ‘x’ column and label in ‘y’ column.
* In ‘x’ column we drop the ‘Loan\_id’ and ‘Loan\_status’ column.
* In ‘y’ column we put ‘Loan\_status’ column.
* #separating the data and label
* x=df.drop(columns=['Loan\_ID','Loan\_Status'],axis=1)
* y=df['Loan\_Status']
* Print ‘x’ and ‘y’
* Gender Married Dependents Education Self\_Employed ApplicantIncome \
* 1 1 1 1 1 0 4583
* 2 1 1 0 1 1 3000
* 3 1 1 0 0 0 2583
* 4 1 0 0 1 0 6000
* 5 1 1 2 1 1 5417
* .. ... ... ... ... ... ...
* 609 0 0 0 1 0 2900
* 610 1 1 4 1 0 4106
* 611 1 1 1 1 0 8072
* 612 1 1 2 1 0 7583
* 613 0 0 0 1 1 4583
* CoapplicantIncome LoanAmount Loan\_Amount\_Term Credit\_History \
* 1 1508.0 128.0 360.0 1.0
* 2 0.0 66.0 360.0 1.0
* 3 2358.0 120.0 360.0 1.0
* 4 0.0 141.0 360.0 1.0
* 5 4196.0 267.0 360.0 1.0
* .. ... ... ... ...
* 609 0.0 71.0 360.0 1.0
* 610 0.0 40.0 180.0 1.0
* 611 240.0 253.0 360.0 1.0
* 612 0.0 187.0 360.0 1.0
* 613 0.0 133.0 360.0 0.0
* Property\_Area
* 1 0
* 2 2
* 3 2
* 4 2
* 5 2
* .. ...
* 609 0
* 610 0
* 611 2
* 612 2
* 613 1
* [480 rows x 11 columns]
* 1 0
* 2 1
* 3 1
* 4 1
* 5 1
* ..
* 609 1
* 610 1
* 611 1
* 612 1
* 613 0
* Name: Loan\_Status, Length: 480, dtype: int64

**Train-Test Split:**

* Divide the dataset as training and testing data.We put 20% data for testing.
* x\_train, x\_test, y\_train, y\_test=train\_test\_split(x, y, test\_size=0.2, stratify=y, random\_state=2)

**Model\_fitting**

lr=LogisticRegression()

dt=DecisionTreeClassifier()

rf=RandomForestClassifier()

adb=AdaBoostClassifier()

lr.fit(x\_train, y\_train)

dt.fit(x\_train, y\_train)

rf.fit(x\_train, y\_train)

adb.fit(x\_train, y\_train)

* Fit the x\_train and y\_train data in LogisticRegression, DecisionTree, RandomForest and AdaBoost classification algorithm.
* Print Classification score for LogisticRegression,DecisionTree,RandomForest and AdaBoostClassifier.
* # classification score
* print("lr classification score", lr.score(x\_train, y\_train))
* print("dt classification score", dt.score(x\_train, y\_train))
* print("rf classification score", rf.score(x\_train, y\_train))
* print("adb classification score", adb.score(x\_train, y\_train))
* LogisticRegression 79%,DecisionTree 100%,RandomForest 100% and AdaBoostClassifier has 86%.

**Model Evaluation:**

* Predict x\_test by all four algorithms, using predict function.
* lr\_ypred=lr.predict(x\_test)
* dt\_ypred=dt.predict(x\_test)
* rf\_ypred=rf.predict(x\_test)
* adb\_ypred=adb.predict(x\_test)
* Derive the confusion matrix for all four algorithms.
* LogisticRegression confusion\_matrix.
* # Using confusion matrix in order to evaluate model accuracy
* lr\_conf\_mat=confusion\_matrix(y\_test, lr\_ypred)
* print(lr\_conf\_mat)
* [[16 14]
* [ 2 64]]
* DecisionTree confusion\_matrix.
* dt\_conf\_mat=confusion\_matrix(y\_test, dt\_ypred)
* print(dt\_conf\_mat)
* [[17 13]
* [19 47]]
* RandomForest confusion\_matrix.
* rf\_conf\_mat=confusion\_matrix(y\_test, rf\_ypred)
* print(rf\_conf\_mat)
* [[17 13]
* [ 4 62]]
* AdaBoost confusion\_matrix.
* adb\_conf\_mat=confusion\_matrix(y\_test, adb\_ypred)
* print(adb\_conf\_mat)
* [[18 12]
* [ 8 58]]

**Check classification report for each model**

* Logistic Regression classification report.
* lr\_report=classification\_report(y\_test, lr\_ypred)
* print(lr\_report)
* precision recall f1-score support
* 0 0.89 0.53 0.67 30
* 1 0.82 0.97 0.89 66
* accuracy 0.83 96
* macro avg 0.85 0.75 0.78 96
* weighted avg 0.84 0.83 0.82 96
* DecisionTree classification report.
* dt\_report=classification\_report(y\_test, dt\_ypred)
* print(dt\_report)
* precision recall f1-score support
* 0 0.47 0.57 0.52 30
* 1 0.78 0.71 0.75 66
* accuracy 0.67 96
* macro avg 0.63 0.64 0.63 96
* weighted avg 0.69 0.67 0.67 96
* RandomForest classification report.
* rf\_report=classification\_report(y\_test, rf\_ypred)
* print(rf\_report)
* precision recall f1-score support
* 0 0.81 0.57 0.67 30
* 1 0.83 0.94 0.88 66
* accuracy 0.82 96
* macro avg 0.82 0.75 0.77 96
* weighted avg 0.82 0.82 0.81 96
* AdaBoostClassifier classification report.
* adb\_report=classification\_report(y\_test, adb\_ypred)
* print(adb\_report)
* precision recall f1-score support
* 0 0.69 0.60 0.64 30
* 1 0.83 0.88 0.85 66
* accuracy 0.79 96
* macro avg 0.76 0.74 0.75 96
* weighted avg 0.79 0.79 0.79 96

**ROC AUC SCORE FOR ALL FOUR MODELS USING SKLEARN LIBRARY**

from sklearn.metrics import roc\_curve, auc, roc\_auc\_score

from sklearn.metrics import plot\_roc\_curve

# importing the roc and auc from sklearn and predict the x\_test and checking the roc\_auc\_score

print(roc\_auc\_score(y\_test, lr.predict(x\_test)))

print(roc\_auc\_score(y\_test, dt.predict(x\_test)))

print(roc\_auc\_score(y\_test, rf.predict(x\_test)))

print(roc\_auc\_score(y\_test, adb.predict(x\_test)))

LogisticRegression: 0.7515151515151516

DecisionTreeClassifier : 0.6393939393939394

RandomForestClassifier : 0.753030303030303

AdaBoostClassifier : 0.7393939393939394

Now we plot the roc curve to check the best fitted model.

# lets find the roc curve to check the best fitted model

disp=plot\_roc\_curve(dt, x\_test, y\_test)

plot\_roc\_curve(lr, x\_test, y\_test, ax=disp.ax\_)

plot\_roc\_curve(rf, x\_test, y\_test, ax=disp.ax\_)

plot\_roc\_curve(adb, x\_test, y\_test, ax=disp.ax\_)

plt.legend(prop={'size':11}, loc='lower right')

LogisticRegression aucis:0.74

DecisionTree auc is : 0.64

RandomForest auc is :0.80

AdaBoost auc is :0.77

From the above observation we can see RandomForest is our best fitted model.

Now take **cross validation score** for Random Forest model. i.e.

Mean of Cross validation score for Random Forest model => 0.8020833333333333

Now do the **Hyperparameter tuning for using Grid SearchCV.**

# Number of trees in random forest

n\_estimators=[int(x) for x in np.linspace(start=10, stop=80, num=10)]

# Number of features to consider at every split

max\_features=['auto', 'sqrt']

# Maximum number of levels in tree

max\_depth=[2,4]

# Minimum number of samples required to split a node

min\_samples\_split=[2, 5]

# Minimum number of samples required at each leaf node

min\_samples\_leaf=[1, 2]

# Method of selecting samples for training each tree

bootstrap=[True, False]

# Create the param grid

param\_grid={'n\_estimators':n\_estimators,

'max\_features':max\_features,

'max\_depth':max\_depth,

'min\_samples\_split':min\_samples\_split,

'min\_samples\_leaf':min\_samples\_leaf,

'bootstrap':bootstrap}

print(param\_grid)

{'n\_estimators': [10, 17, 25, 33, 41, 48, 56, 64, 72, 80], 'max\_features': ['auto', 'sqrt'], 'max\_depth': [2, 4], 'min\_samples\_split': [2, 5], 'min\_samples\_leaf': [1, 2], 'bootstrap': [True, False]}

rf\_model=RandomForestClassifier()

from sklearn.model\_selection import GridSearchCV

rf\_grid=GridSearchCV(estimator=rf\_model, param\_grid=param\_grid, cv=3, verbose=2, n\_jobs=4 )

rf\_grid.fit(x\_train, y\_train)

Fitting 3 folds for each of 320 candidates, totalling 960 fits

Out[46]:

GridSearchCV(cv=3, estimator=RandomForestClassifier(), n\_jobs=4,

param\_grid={'bootstrap': [True, False], 'max\_depth': [2, 4],

'max\_features': ['auto', 'sqrt'],

'min\_samples\_leaf': [1, 2],

'min\_samples\_split': [2, 5],

'n\_estimators': [10, 17, 25, 33, 41, 48, 56, 64, 72,

80]},

verbose=2)

rf\_grid.best\_params\_

{'bootstrap': True,

'max\_depth': 4,

'max\_features': 'auto',

'min\_samples\_leaf': 1,

'min\_samples\_split': 2,

'n\_estimators': 64}

**Now check the train and test accuracy:**

print(f'Train Accuracy-:{rf\_grid.score(x\_train, y\_train):.3f}')

print(f'Test Accuracy-:{rf\_grid.score(x\_test, y\_test):.3f}')

Train Accuracy-:0.818

Test Accuracy-:0.823

**Now saving the model using joblib:**

rf=RandomForestClassifier()

rf.fit(x, y)

RandomForestClassifier()

import joblib

joblib.dump(rf, 'model\_joblib\_rf')

['model\_joblib\_rf']

model=joblib.load('model\_joblib\_rf')

**Conclusion :**

Random Forest Classifier is giving the best accuracy with an accuracy score of train accuracy:81% and test accuracy : 82%.