Loan Application Status Prediction

Introduction :

Load Application Status Prediction is a task that can be done based on historical information of the customer and bank. By checking the dataset already existed regarding the status of the Load Application and creating a model will help us to Predict the further Loan Application Status.Dataset includes details of applicants who have applied for loan. The dataset includes details like credit history, loan amount, their income, dependents etc.Will build a model that can predict whether the loan of the applicant will be approved or not on the basis of the details provided in the dataset.

Defining the problem statement :

About Company
Dream Housing Finance company deals in all home loans. They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan.

Problem
Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have given a problem to identify the customers segments, those are eligible for loan amount so that they can specifically target these customers. Here they have provided a partial data set.

Data Pre-processing Done:

The purpose of preprocessing is to convert raw data into a form that fits machine learning. Structured and clean data allows a data scientist to get more precise results from an applied machine learning model. The technique includes data formatting, cleaning, and sampling.

- **Hardware and Software Requirements and Tools Used**

- *Tools: Python 3.8.5, Jupyter Notebook, Numpy, Pandas, Matplotlib, Seaborn, Scikit-learn, Scipy*
- *Techniques: LogisticRegression, GaussianNB, SVC, RandomForestClassifier, AdaBoostClassifier, DecisionTreeClassifier, GradientBoostingClassifier.*
- *Hardware: I3 processor, 4GB RAM*

- **Model/s Development and Evaluation :**

I have used the below models for classification:

- LogisticRegression
- SVC
- RandomForestClassifier
- AdaBoostClassifier
- DecisionTreeClassifier
- *GaussianNB*
- GradientBoostingClassifier

- Identification of possible problem-solving approaches (methods)

- Read the data (from csv)
- Identify the dependent and independent variables.
- Check if the data has missing values or the data is categorical or not.
- If yes, apply basic data preprocessing operations to bring the data in a go to go format.
- Now split the data into the groups of training and testing for the respective purpose.
- After splitting data, fit it to a most suitable model. (How to find a suitable model is answered below)
- Validate the model. If satisfactory, then go with it, else tune the parameters and keep testing. In a few cases, you can also try different algorithms for the same problem to understand the difference between the accuracies.
- From step 7 one can also learn about accuracy paradox.
- Visualize the data.

- **Testing of Identified Approaches (Algorithms)**

  - Naive Bayes
  - Cross validation
  - Confusion matrix
  - Accuracy score
  - Classification report

- **Run and Evaluate selected models**

I have used the below models for classification:

Logistic regression

```
lg=LogisticRegression()
lg.fit(x_train,y_train)
pred=lg.predict(x_test)
lg_accu=accuracy_score(y_test,pred)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.7586206896551724
[[ 25  46]
 [  3 129]]
              precision    recall  f1-score   support

           0       0.89      0.35      0.51        71
           1       0.74      0.98      0.84       132

    accuracy                           0.76       203
   macro avg       0.81      0.66      0.67       203
weighted avg       0.79      0.76      0.72       203
```

RandomForestClassifier

```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
pred=rf.predict(x_test)
rf_accu=accuracy_score(y_test,pred)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.7684729064039408
[[ 29  42]
 [  5 127]]
              precision    recall  f1-score   support

           0       0.85      0.41      0.55        71
           1       0.75      0.96      0.84       132

    accuracy                           0.77       203
   macro avg       0.80      0.69      0.70       203
weighted avg       0.79      0.77      0.74       203
```

SVC

```
svc=SVC()
svc.fit(x_train,y_train)
pred=svc.predict(x_test)
svc_accu=accuracy_score(y_test,pred)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.6502463054187192
[[  0  71]
 [  0 132]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        71
           1       0.65      1.00      0.79       132

    accuracy                           0.65       203
   macro avg       0.33      0.50      0.39       203
weighted avg       0.42      0.65      0.51       203
```

AdaBoostClassifier ¶

```
ad=AdaBoostClassifier()
ad.fit(x_train,y_train)
pred=ad.predict(x_test)
ad_accu=accuracy_score(y_test,pred)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.7586206896551724
[[ 28  43]
 [  6 126]]
              precision    recall  f1-score   support

           0       0.82      0.39      0.53        71
           1       0.75      0.95      0.84       132

    accuracy                           0.76       203
   macro avg       0.78      0.67      0.69       203
weighted avg       0.77      0.76      0.73       203
```

DecisionTreeClassifier

```
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
pred=dt.predict(x_test)
dt_accu=accuracy_score(y_test,pred)

print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.6896551724137931
[[ 38  33]
 [ 30 102]]
              precision    recall  f1-score   support

           0       0.56      0.54      0.55        71
           1       0.76      0.77      0.76       132

    accuracy                           0.69       203
   macro avg       0.66      0.65      0.66       203
weighted avg       0.69      0.69      0.69       203
```

GradientBoostingClassifier

```
gbc=GradientBoostingClassifier()
gbc.fit(x_train,y_train)
pred=gbc.predict(x_test)
gbc_accu=accuracy_score(y_test,pred)

print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.7487684729064039
[[ 29  42]
 [  9 123]]
              precision    recall  f1-score   support

           0       0.76      0.41      0.53        71
           1       0.75      0.93      0.83       132

    accuracy                           0.75       203
   macro avg       0.75      0.67      0.68       203
weighted avg       0.75      0.75      0.72       203
```

- Conclusion:

```
conclusion
```

	0	1	2	3	4	5	6	7	8	9	...	193	194	195	196	197	198	199	200	201	202
predicted	1	0	0	1	0	0	0	0	1	0	...	1	1	0	1	1	1	1	1	0	1
original	1	1	1	0	1	0	1	0	1	0	...	1	1	0	1	1	1	1	0	1	1

2 rows × 203 columns

Random Forest Classifier is accuracy score ,precision ,recall and f1-score is good than other models,Hence Random Forest Classifier is performing is good.Our model shows 88% accuracy, which predicts the status of loanThis is how Model for Load Application Status Prediction is created. Random Forest Classifier is best model for this dataset.