# Major Project Synopsis

***On***

# Loan Approval Prediction using Machine Learning

**In partial fulfilment of requirements for the degree**

***Of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

***Submitted by :***

**GARIMA YADAV [20100BTCSDSI07271]**

**MOHIT KUSHWAH [20100BTCSDSI07279]**

**SAMARTH AMRUTHE [20100BTCSDSI07292]**

**SANSKAR GUPTA [20100BTCSDSI07294]**

*Under the guidance of*

**Prof. Om Kant Sharma**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY**

**SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALA, INDORE**

**JUL-DEC – 2022**

**SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY**

# Word file on Loan Approval Prediction using Machine Learning..

**Introduction:** A loan is the core business part of banks. The main portion the bank's profit is directly come from the profit earned from the loans. Though bank pproves loan after a regress process of verification and testimonial but still there's no surety whether the chosen hopeful is the right hopeful or not. This process takes fresh time while doing it manually. We can prophesy whether that particular hopeful is safe or not and the whole process of testimonial is automated by machine literacy style. Loan Prognostic is really helpful for retainer of banks as well as for the hopeful also.

Loans are the core business of banks. The main profit comes directly from the loan's interest. The loan companies grant a loan after an intensive process of verification and validation. However, they still don't have assurance if the applicant is able to repay the loan with no difficulties.

In this tutorial, we'll build a predictive model to predict if an applicant is able to repay the lending company or not. We will prepare the data using Jupyter Notebook and use various models to predict the target variable.

**Problem: Existing system-**

Bank employees check the details of applicant manually and give the loan to eligible applicant. Checking the details of all applicants takes lot of time. The artificial neural network model for predict the credit risk of a bank. The Feed- forward back propagation neural network is used to forecast the credit default. The method in which two or more classifiers are combined together to produce a ensemble model for the better prediction. They used the bagging and boosting techniques and then used random forest technique. The process of classifiers is to improve the performance of the data and it gives better efficiency. In this work, the authors describe various ensemble techniques for binary classification and also for multi class classification. The new technique that is described by the authors for

ensemble is COB which gives effective performance of classification but it also compromised with noise and outlier
data of classification. Finally they concluded that the ensemble based algorithm improves the results for training data set.

**Drawback of Existing System** Checking details of all applicants consumes lot of time and efforts. There is chances of human error may occur due checking all details manually. There is possibility of assigning loan to ineligible applicant.

**Proposed System**

To deal with the problem, we developed automatic loan prediction using machine learning techniques. We will train the machine with previous dataset. so machine can analyse and understand the process . Then machine will check for eligible applicant and give us result.

Advantages

- Time period for loan sanctioning will be reduced.
- Whole process will be automated , so human error will be avoided
- Eligible applicant will be sanctioned loan without any delay.

## Methodology:

Banks are making major part of profits through loans. Though lot of people are applying for loans. It’s hard to select the genuine applicant, who will repay the loan. While doing the process manually, lot of misconception may happen to select the genuine applicant. Therefore we are developing loan prediction system using machine learning, so the system automatically selects the eligible candidates. This is helpful to both bank staff and applicant. The time period for the sanction of loan will be drastically reduced. In this paper we are predicting the loan data by using some machine learning algorithms that are:

- **KNeighborsClassifiers**
- **RandomForestClassifiers**
- **Support Vector Classifiers (SVC)**
- **Logistics Regression**

**Result: y**. The final results have shown that the model produce different results. Model is marginally better because it includes variables (personal attributes of customer like age, purpose, credit history, credit amount, credit duration, etc.) other than checking account information (which shows wealth of a customer) that should be taken into account to calculate the probability of default on loan correctly. Therefore, by using a Random Forest Classifier approach, the right customers to be targeted for granting loan can be easily detected by evaluating their likelihood of default on loan..

Random Forest Classifier is giving the best accuracy with an accuracy score of 82% for the testing dataset. because

- Random forests is considered as a highly accurate and robust method because of the number of decision trees participating in the process.
- It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.
- The algorithm can be used in both classification and regression problems.
- Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values.
- You can get the relative feature importance, which helps in selecting the most contributing features for the classifier.

**Conclusion:** The interaction of expectation begins from cleaning and handling of information, attribution of missing qualities, exploratory investigation of informational collection and afterward model structure to assessment of model and testing on test information. On Data set, the best-case precision acquired on the first informational collection is 0.811. The accompanying ends are reached after examination that those candidates whose credit score rating was most noticeably awful will neglect to get advance endorsement, because of a higher likelihood of not repaying the credit sum. More often than not, those candidates who have top level salary and requests for lower measure of advance are bound to get affirmed which bodes well, bound to take care of their credits. Some other trademark like gender and conjugal status appears to be not to be mulled over by the organization..

**Codes for Loan Approval Prediction using Machine Learning..**

```
# Importing Libraries and Dataset
Firstly we have to import libraries :
```

In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv("C:\\Users\\Samarth-PC\\Downloads\\LoanApprovalPrediction.csv")
```

In [2]:

```python
data
```

Out[2]:

| s | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Te |
|---|---|---|---|---|---|---|
| Ɔ | Graduate | No | 5849 | 0.0 | NaN | 36 |
| Ɔ | Graduate | No | 4583 | 1508.0 | 128.0 | 36 |
| Ɔ | Graduate | Yes | 3000 | 0.0 | 66.0 | 36 |
| Ɔ | Not Graduate | No | 2583 | 2358.0 | 120.0 | 36 |
| Ɔ | Graduate | No | 6000 | 0.0 | 141.0 | 36 |
| .. | ... | ... | ... | ... | ... | |
| Ɔ | Graduate | No | 2900 | 0.0 | 71.0 | 36 |
| Ɔ | Graduate | No | 4106 | 0.0 | 40.0 | 18 |
| Ɔ | Graduate | No | 8072 | 240.0 | 253.0 | 36 |
| Ɔ | Graduate | No | 7583 | 0.0 | 187.0 | 36 |
| Ɔ | Graduate | Yes | 4583 | 0.0 | 133.0 | 36 |

In [3]:

```
data.head(5)
```

Out[3]:

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | Coapp |
|---|---|---|---|---|---|---|---|---|
| **0** | LP001002 | Male | No | 0.0 | Graduate | No | 5849 | |
| **1** | LP001003 | Male | Yes | 1.0 | Graduate | No | 4583 | |
| **2** | LP001005 | Male | Yes | 0.0 | Graduate | Yes | 3000 | |
| **3** | LP001006 | Male | Yes | 0.0 | Not Graduate | No | 2583 | |
| **4** | LP001008 | Male | No | 0.0 | Graduate | No | 6000 | |

In [4]:

```
data.tail(5)
```

Out[4]:

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | Coa |
|---|---|---|---|---|---|---|---|---|
| **593** | LP002978 | Female | No | 0.0 | Graduate | No | 2900 | |
| **594** | LP002979 | Male | Yes | 3.0 | Graduate | No | 4106 | |
| **595** | LP002983 | Male | Yes | 1.0 | Graduate | No | 8072 | |
| **596** | LP002984 | Male | Yes | 2.0 | Graduate | No | 7583 | |
| **597** | LP002990 | Female | No | 0.0 | Graduate | Yes | 4583 | |

```
Data Preprocessing and Visualization
Get the number of columns of object datatype.
```

In [5]:

```
obj = (data.dtypes == 'object')
print("Categorical variables:",len(list(obj[obj].index)))
```

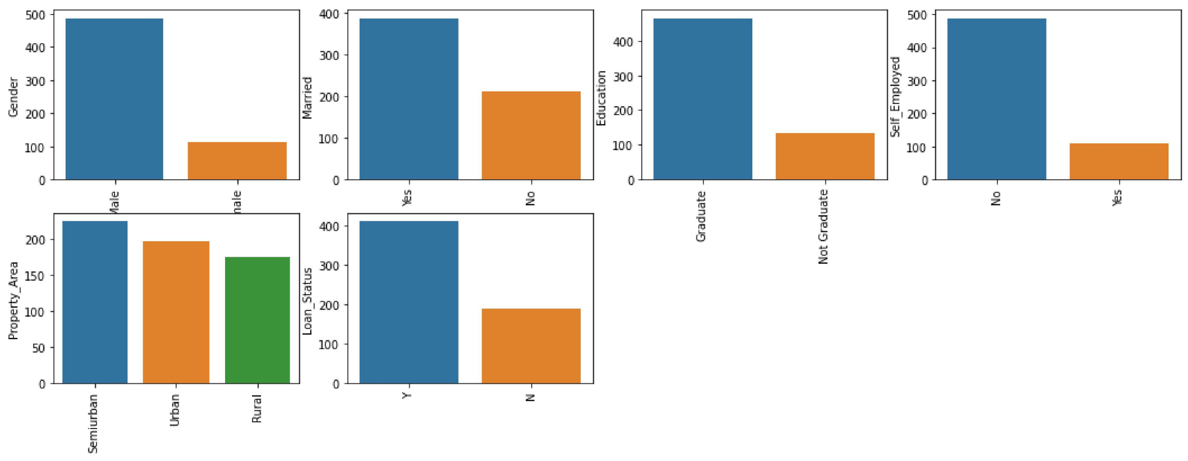
```
Categorical variables: 7
```

In [6]:

```
# Dropping Loan_ID column
data.drop(['Loan_ID'],axis=1,inplace=True)
```

In [7]:

```
obj = (data.dtypes == 'object')
object_cols = list(obj[obj].index)
plt.figure(figsize=(18,36))
index = 1

for col in object_cols:
    y=data[col].value_counts()
    plt.subplot(11,4,index)
    plt.xticks(rotation=90)
    sns.barplot(x=list(y.index), y=y)
    index +=1
```



In [8]:

```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how
# to understand word labels.
label_encoder = preprocessing.LabelEncoder()
obj = (data.dtypes == 'object')
for col in list(obj[obj].index):
    data[col] = label_encoder.fit_transform(data[col])
```

In [9]:

```
# To find the number of columns with
# datatype==object
obj = (data.dtypes == 'object')
print("Categorical variables:",len(list(obj[obj].index)))
```

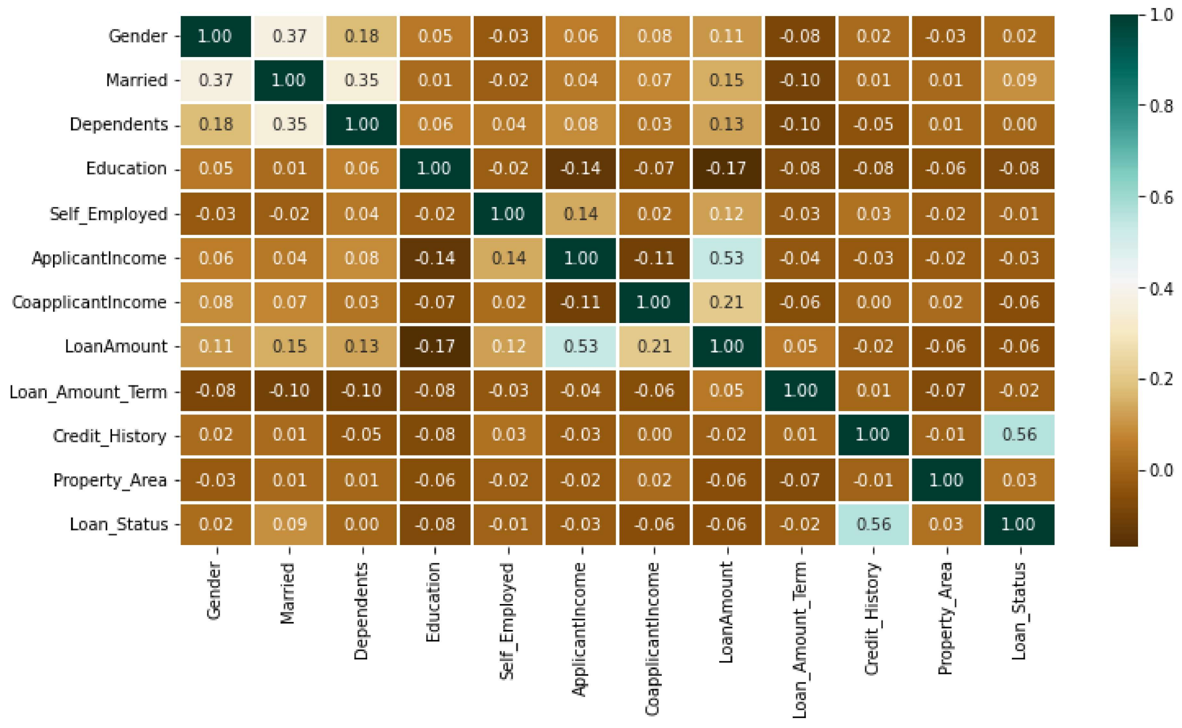
```
Categorical variables: 0
```

In [10]:

```
plt.figure(figsize=(12,6))

sns.heatmap(data.corr(),cmap='BrBG',fmt='.2f',
            linewidths=2,annot=True)
```

Out[10]:

```
<AxesSubplot:>
```

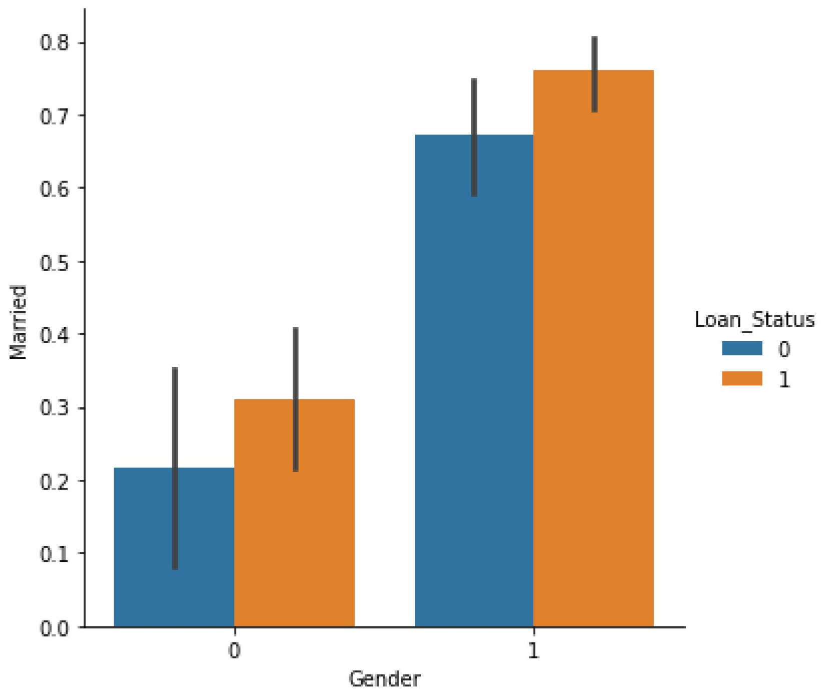


In [11]:

```
sns.catplot(x="Gender", y="Married",
            hue="Loan_Status",
            kind="bar",
            data=data)
```

Out[11]:

```
<seaborn.axisgrid.FacetGrid at 0x1d9afe0a7f0>
```



In [12]:

```
for col in data.columns:
    data[col] = data[col].fillna(data[col].mean())
data.isna().sum()
```

Out[12]:

```
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
```

```
Splitting Dataset
```

In [13]:

```
from sklearn.model_selection import train_test_split

X = data.drop(['Loan_Status'],axis=1)
Y = data['Loan_Status']
X.shape,Y.shape

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    test_size=0.4,
                                                    random_state=1)
X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
```

Out[13]:

```
((358, 11), (240, 11), (358,), (240,))
```

```
Model Training and Evaluation
As this is a classification problem so we will be using these models :
    KNeighborsClassifiers
    RandomForestClassifiers
    Support Vector Classifiers (SVC)
    Logistics Regression
```

In [14]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression

from sklearn import metrics

knn = KNeighborsClassifier(n_neighbors=3)
rfc = RandomForestClassifier(n_estimators = 7,
                            criterion = 'entropy',
                            random_state =7)
svc = SVC()
lc = LogisticRegression()

# making predictions on the training set
for clf in (rfc, knn, svc,lc):
    clf.fit(X_train, Y_train)
    Y_pred = clf.predict(X_train)
    print("Accuracy score of ",
        clf.__class__.__name__,
        "=",100*metrics.accuracy_score(Y_train,
                                        Y_pred))
```

```
Accuracy score of  RandomForestClassifier = 98.04469273743017
Accuracy score of  KNeighborsClassifier = 78.49162011173185
Accuracy score of  SVC = 68.71508379888269
Accuracy score of  LogisticRegression = 79.60893854748603

C:\Users\Samarth-PC\anaconda3\lib\site-packages\sklearn\linear_model\_logist
ic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scik
it-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  n_iter_i = _check_optimize_result(
```

```
Prediction on the test set:
```

In [15]:

```
# making predictions on the testing set
for clf in (rfc, knn, svc,lc):
    clf.fit(X_train, Y_train)
    Y_pred = clf.predict(X_test)
    print("Accuracy score of ",
        clf.__class__.__name__,"=",
        100*metrics.accuracy_score(Y_test,
                                    Y_pred))
```

```
Accuracy score of  RandomForestClassifier = 82.5
Accuracy score of  KNeighborsClassifier = 63.74999999999999
Accuracy score of  SVC = 69.16666666666667
Accuracy score of  LogisticRegression = 81.25

C:\Users\Samarth-PC\anaconda3\lib\site-packages\sklearn\linear_model\_logist
ic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scik
it-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  n_iter_i = _check_optimize_result(
```

In [18]:

```
x=data.iloc[:,1:5]      # independent variable
y=data['Loan_Status']       # Dependent variable
x.shape,y.shape
```

Out[18]:

```
((598, 4), (598,))
```

In [19]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.25)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[19]:

```
((448, 4), (150, 4), (448,), (150,))
```

In [20]:

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
```

In [21]:

```
lr.fit(x_train,y_train)
```

Out[21]:

```
LogisticRegression()
```

In [22]:

```
y_pred_lr=lr.predict(x_test)
y_pred_lr[:5],y_test.values[:5]
```

Out[22]:

```
(array([1, 1, 1, 1, 1]), array([1, 1, 1, 0, 1]))
```

In [23]:

```
print(lr.score(x_train,y_train))
print(lr.score(x_test,y_test))
```

```
0.7098214285714286
0.62
```

In [24]:

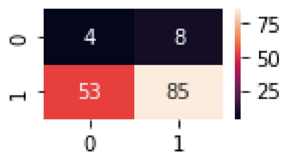
```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
print(confusion_matrix(y_pred_lr,y_test))
print(classification_report(y_pred_lr,y_test))
print(f'model_score- {lr.score(x_test,y_test)} ')
print(f'accuracy_score- {accuracy_score(y_pred_lr,y_test)} ')
```

```
[[ 0  0]
 [57 93]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      0.62      0.77       150

    accuracy                           0.62       150
   macro avg       0.50      0.31      0.38       150
weighted avg       1.00      0.62      0.77       150

model_score- 0.62
accuracy_score- 0.62

C:\Users\Samarth-PC\anaconda3\lib\site-packages\sklearn\metrics\_classificat
ion.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` paramet
er to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Samarth-PC\anaconda3\lib\site-packages\sklearn\metrics\_classificat
ion.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` paramet
er to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Samarth-PC\anaconda3\lib\site-packages\sklearn\metrics\_classificat
ion.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` paramet
er to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [28]:

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
y_pred_dtc=dtc.predict(x_test)
print(f'Predicted_y{y_pred_dtc[:5]} Actual_y{y_test.values[:5]}')
print(confusion_matrix(y_pred_dtc,y_test))
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
cm=confusion_matrix(y_pred_dtc,y_test)
plt.figure(figsize=(2,1))
print(sns.heatmap(cm,annot=True))
plt.show()
print(classification_report(y_pred_dtc,y_test))
print(f'model_score- {dtc.score(x_test,y_test)} ')
print(f'accuracy_score- {accuracy_score(y_pred_dtc,y_test)} ')
ax = plt.axes(projection ='3d')
ax.scatter3D(x_test['ApplicantIncome'],x_test['ApplicantIncome'],y_test)
ax.scatter3D(x_test['ApplicantIncome'],x_test['ApplicantIncome'],y_pred_dtc,'black')
plt.show()
```

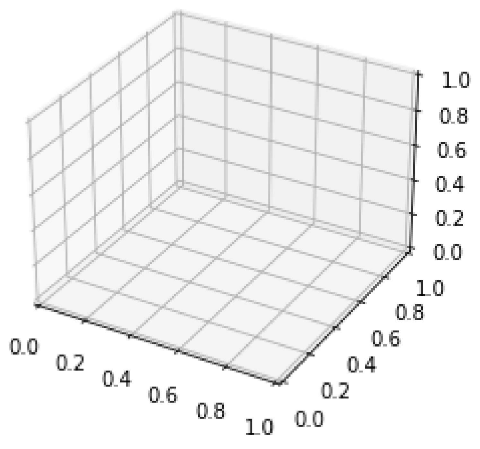
```
Predicted_y[1 1 1 1 1] Actual_y[1 1 1 0 1]
[[ 4  8]
 [53 85]]
AxesSubplot(0.125,0.125;0.62x0.755)
```



```
              precision    recall  f1-score   support

           0       0.07      0.33      0.12        12
           1       0.91      0.62      0.74       138

    accuracy                           0.59       150
   macro avg       0.49      0.47      0.43       150
weighted avg       0.85      0.59      0.69       150

model_score- 0.5933333333333334
accuracy_score- 0.5933333333333334
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3621, in Inde
x.get_loc(self, key, method, tolerance)
   3620 try:
-> 3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:136, in pandas._li
bs.index.IndexEngine.get_loc()

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:163, in pandas._li
bs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5198, in pandas._libs.hashtabl
```

```
e.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5206, in pandas._libs.hashtabl
e.PyObjectHashTable.get_item()

KeyError: 'ApplicantIncome'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Input In [28], in <cell line: 16>()
     14 print(f'accuracy_score- {accuracy_score(y_pred_dtc,y_test)} ')
     15 ax = plt.axes(projection ='3d')
---> 16 ax.scatter3D(x_test['ApplicantIncome'],x_test['ApplicantIncome'],y_t
est)
     17 ax.scatter3D(x_test['ApplicantIncome'],x_test['ApplicantIncome'],y_p
red_dtc,'black')
     18 plt.show()

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:3505, in DataFrame._
_getitem__(self, key)
   3503 if self.columns.nlevels > 1:
   3504     return self._getitem_multilevel(key)
-> 3505 indexer = self.columns.get_loc(key)
   3506 if is_integer(indexer):
   3507     indexer = [indexer]

File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3623, in Inde
x.get_loc(self, key, method, tolerance)
   3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:
-> 3623     raise KeyError(key) from err
   3624 except TypeError:
   3625     # If we have a listlike key, _check_indexing_error will raise
   3626     #  InvalidIndexError. Otherwise we fall through and re-raise
   3627     #  the TypeError.
   3628     self._check_indexing_error(key)

KeyError: 'ApplicantIncome'
```



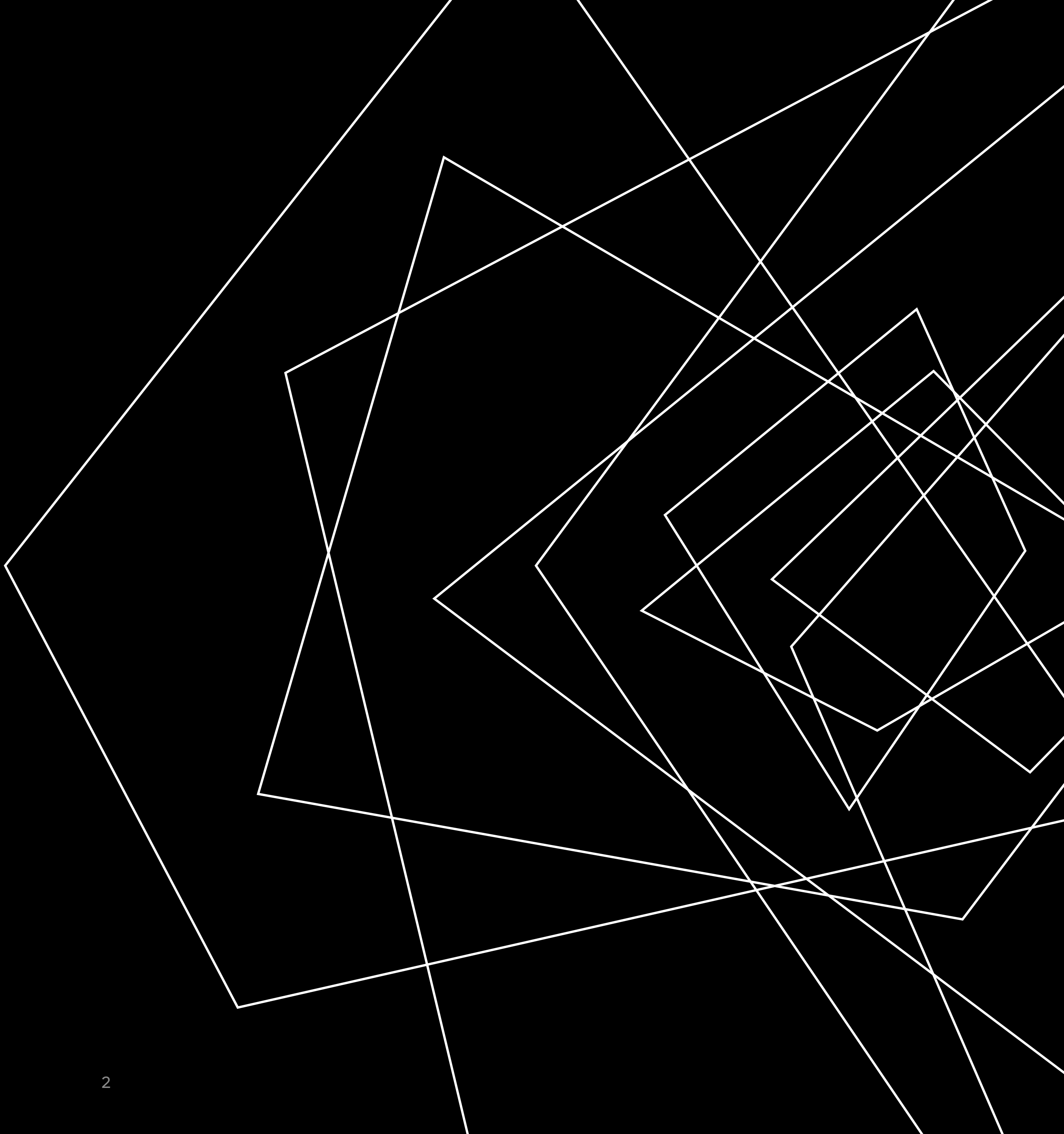
In [ ]:

# LOAN APPROVAL PREDICTION BASED ON MACHINE LEARNING APPROACH

By Samarth Amrute
Garima Yadav
Mohit Kushwah
Sanskar Gupta

## AGENDA

- Motivation
- Problem statement
- Objectives
- Background
- Dataset Specification
- Machine Learing Prediction Mode
- Experimental Result



## MOTIVATION

Distribution of the loans is the core business part of almost every banks. The main portion the bank's assets is directly came from the profit earned from the loans distributed by the banks.The prime objective in banking environment is to invest their assets in safe hands Today many banks/financial companies approves loan after a regress process of verification and validation but still there is no surety whether the chosen applicant is the deserving right applicant out of all applicants Through this system we can predict whether that particular applicant is safe or not and the whole process of validation of features is automated by machine learning technique.

## PROBLEM STATEMENT

- Loan Prediction is very helpful for employee of banks as well as for the applicant also. The aim of this Search is to provide quick, immediate and easy way to choose the deserving applicants. It can provide special advantages to the bank. The Loan Prediction System can automatically calculate the weight of each features taking part in loan processing and on new test data same features are processed with respect to their associated weight.
- A time limit can be set for the applicant to check whether his/her loan can be sanctioned or not. Loan Prediction System allows jumping to specific application so that it can be check on priority basis.
- This Search is exclusively for the managing authority of Bank/finance company, whole process of prediction is done privately no stakeholders would be able to alter the processing. Result against particular Loan Id can be send to various department of banks so that they can take appropriate action on application.
- This helps all others department to carried out other formalities

## OBJECTIVES

- The primary goal of this search is to extract patterns from a common loan approved dataset, and then build a model based on these extracted patterns, in order to predict the likely loan defaulters by using classification data mining algorithms.
- The historical data of the customers will be used in order to do the analysis. Later on, some analysis will also be done to find the most relevant attributes, i.e., the factors that affect the prediction result the most.

## DATASET SPECIFICATION

| | | |
|---|---|---|
| 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 | ApplicantIncome | Applicant income |
| 8 | CoapplicantIncome | Co-applicant income |
| 9 | LoanAmount | Loan amount (in thousands) |
| 10 | Loan_Amount_Term | Terms 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 |

## MACHINE LEARNING PREDICTION MODEL

### **Decision Tree Classifier**

- The basic algorithm of decision tree requires all attributes or features should be discretized. Feature selection is based on greatest information gain of features.
- The knowledge depicted in decision tree can represented in the form of IF-THEN rules.
- This model is an extension of C4.5 classification algorithms described by Quinlan.

## MACHINE LEARNING PREDICTION MODEL

**K-neighbors Classifier.**

The k-neighbors is commonly used and easy to apply classification method which implements the k neighbors queries to classify data. It is an instant-based and non-parametric learning method. In this method, the classifier learns from the instances in the training dataset and classifies new input by using the previously measured scores.

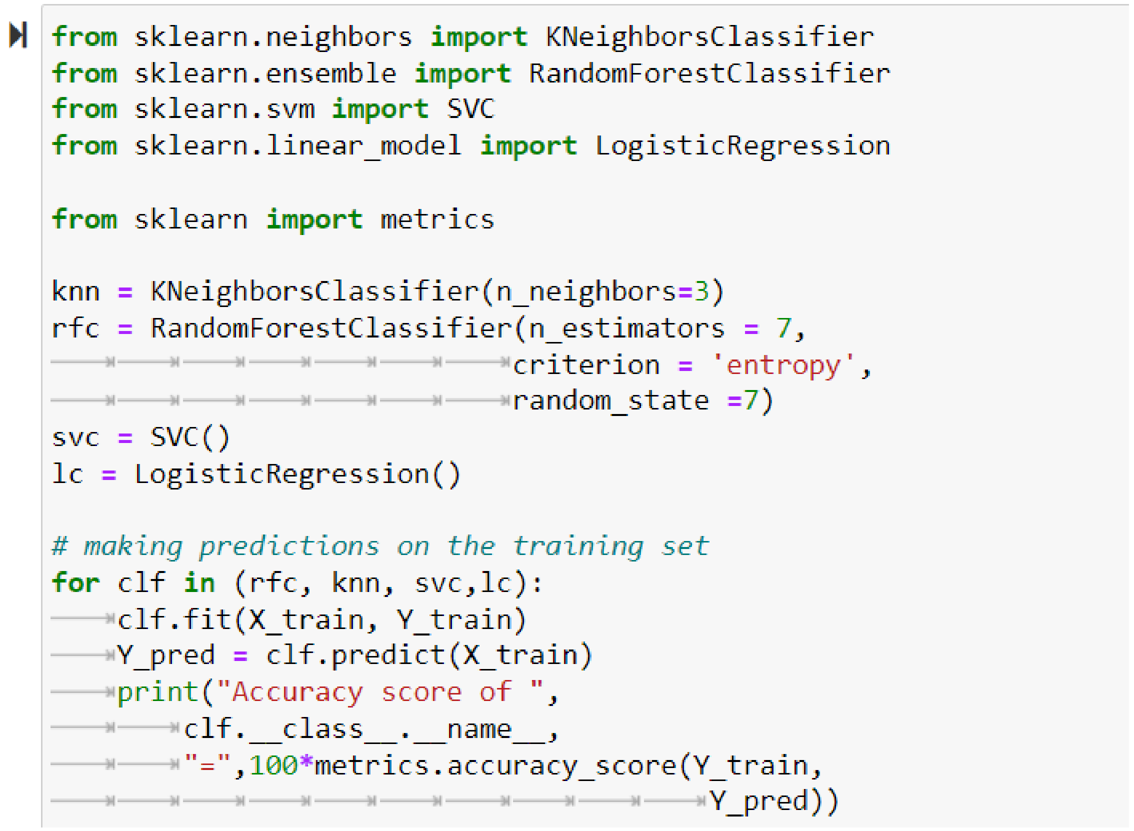
Scikit-learn API provides the KNeighborsClassifier class to implement k-neighbors method for classification problems. In this tutorial, we'll briefly learn how to classify data by using the KNeighborsClassifier class in Python.

## MACHINE LEARNING PREDICTION MODEL

**Logistic Regression**

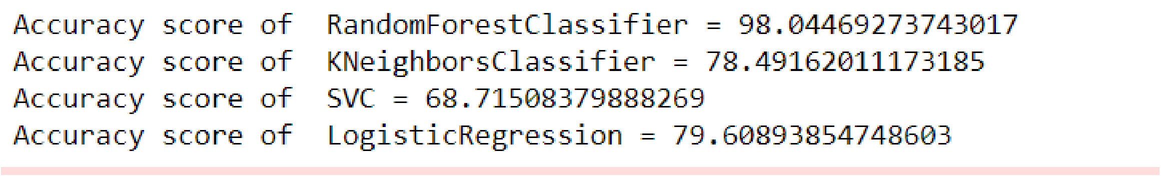
- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.

# MODEL IMPLEMENTATION

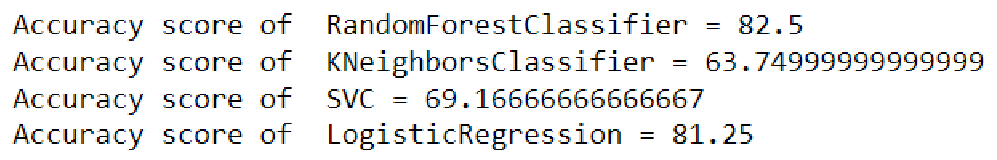


```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression

from sklearn import metrics

knn = KNeighborsClassifier(n_neighbors=3)
rfc = RandomForestClassifier(n_estimators = 7,
                            criterion = 'entropy',
                            random_state =7)
svc = SVC()
lc = LogisticRegression()

# making predictions on the training set
for clf in (rfc, knn, svc,lc):
    clf.fit(X_train, Y_train)
    Y_pred = clf.predict(X_train)
    print("Accuracy score of ",
        clf.__class__.__name__,
        "=",100*metrics.accuracy_score(Y_train,
                                        Y_pred))
```

## EXPERIMENTAL RESULT



```
Accuracy score of  RandomForestClassifier = 98.04469273743017
Accuracy score of  KNeighborsClassifier = 78.49162011173185
Accuracy score of  SVC = 68.71508379888269
Accuracy score of  LogisticRegression = 79.60893854748603
```



```
Accuracy score of  RandomForestClassifier = 82.5
Accuracy score of  KNeighborsClassifier = 63.74999999999999
Accuracy score of  SVC = 69.16666666666667
Accuracy score of  LogisticRegression = 81.25
```

## CONCLUSION

Random Forest Classifier is giving the best accuracy with an accuracy score of 82% for the testing dataset..

## THANKYOU!!!!

SAMARTH AMRUTE

GARIMA YADAV

MOHIT KUSHWAH

SANSKAR GUPTA