Mini-Project Report

On

LOAN APPROVAL PREDICTION

By

Sanjana Gupta

(171500290)

Kratika Upadhyay

(171500163)

Under the Supervision of

Mr Mandeep Singh

(Technical Trainer)

Department of Computer Engineering and Applications

GLA UNIVERSITY

Mathura

Department of computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

Declaration

We hereby declare that the work which is being presented in the project**"**Loan approval prediction**"**, in partial fulfillment of the requirements for the project, is an authentic record of our own work carried under the supervision of "Mr.Mandeep Singh".

Sign ______________________________ Sign ______________________________

Name of Candidate:Sanjana Gupta Name of Candidate: Kratika Upadhyay

University Roll No. : 171500290 University Roll No.: 171500163

Certificate

Thisis to certify that the above statements made by the candidates are correct to the best of our knowledge and belief.

____

Supervisor

Mr.Mandeep Singh

(Technical Trainer)

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our mentor **Mr.Mandeep Singh,Technical Trainer**, **Dept. of CEA** for providing the guidance on this project. We deeply respect our instructor for his vast knowledge, numerous suggestions, and strong passion to complete this project. Valuable discussions with him not only made our work smooth but also encouraged us to think more professionally in the field of research.

Our heartiest thanks to **Prof.(Dr.) Anand SinghJalal**, Head of Dept. , Department of CEA
For providing us with an encouraging platform to develop this project , which thus helped
us in shaping our abilities towards a constructive goal.
After doing this project we can confidently say that this experience has not only enriched us with technical knowledge but also has unparsed the maturity of thought and vision. The attributes required being a successful professional.

We are also thankful to all teaching and non-teaching staff for their support and cooperation.

Sign ____________________	Sign ____________________
Name of Candidate : Sanjana Gupta	Name of Candidate: Kratika Upadhyay
University Roll No. : 171500290	University Roll No.: 171500163

ABSTRACT

Distribution of the loans is the core business part of almost every banks. The main portion of 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.

The Loan Prediction System can automatically calculate each features taking part in loan processing and on new test data same features are processed with respect to their associated features .A time limit can be set for the applicant to check whether his/her loan can be sanctioned or not.

Contents

Declaration

Certificate

Acknowledgement

Abstract

CHAPTER 1. INTRODUCTION

1.1 Problem Statement

1.2 Motivation

1.3 Scope

CHAPTER 2 ALGORITHM

2.1 Decision Tree

2.2Logistic Regression

2.3 k - Nearest Neighbors

2.4 Naïve Bayes

2.5 Random forest

2.6 SVM

CHAPTER 3 SOURCE CODE

3.1 Code

CHAPTER 4 CONCLUSION

REFERENCE

CHAPTER 1

INTRODUCTION

1.1.Problem Statement

A 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 data set.

1.2 Motivation

The loan is one of the most important products of the banking. All the banks are trying to figure out effective business strategies to persuade customers to apply their loans. However, there are some customers behave negatively after their application are approved. To prevent this situation, banks have to find some methods to predict customers' behaviours. Machine learning algorithms have a pretty good performance on this purpose, which are widely-used by the banking. Here, we will work on loan behaviours prediction using machine learning models.

1.3 Future Scope

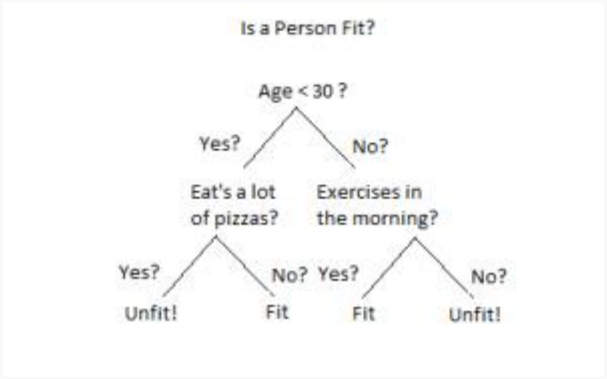
The aim is to provide quick, immediate and easy way to choose the deserving applicants. It can provide special advantages to the bank. So in the near future the so –called software could be made more secure, reliable and dynamic.In near future this module of prediction can be integrate with the module of automated processing system. The system is trained on old training dataset in future software can be made such that new testing data should also take part in training data after some fix time.

CHAPTER 2

ALGORITHM

2.1 Decision Tree

Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.



An example of a decision tree can be explained using above binary tree. Let's say you want to predict whether a person is fit given their information like age, eating habit, and physical activity, etc. The decision nodes here are questions like 'What's the age?', 'Does he exercise?', 'Does he eat a lot of pizzas'? And the leaves, which are outcomes like either 'fit', or 'unfit'. In this case this was a binary classification problem (a yes no type problem).

There are two main types of Decision Trees:

1. **Classification trees** (Yes/No types)

What we've seen above is an example of classification tree, where the outcome was a variable like 'fit' or 'unfit'. Here the decision variable is **Categorical**.

2. **Regression trees** (Continuous data types)

Here the decision or the outcome variable is **Continuous**, e.g. a number like 123.

Working

Now that we know what a Decision Tree is, we'll see how it works internally. There are many algorithms out there which construct Decision Trees, but one of the best is called as **ID3 Algorithm**. ID3 Stands for **Iterative Dichotomiser 3**.

Before discussing the ID3 algorithm, we'll go through few definitions.

Entropy

Entropy, also called as Shannon Entropy is denoted by H(S) for a finite set S, is the measure of the amount of uncertainty or randomness in data.

$$H(S) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

Intuitively, it tells us about the predictability of a certain event. Example, consider a coin toss whose probability of heads is 0.5 and probability of tails is 0.5. Here the entropy is the highest possible, since there's no way of determining what the outcome might be. Alternatively, consider a coin which has heads on both the sides, the entropy of such an event can be predicted perfectly since we know beforehand that it'll always be heads. In other words, this event has **no randomness** hence it's entropy is zero.

In particular, lower values imply less uncertainty while higher values imply high uncertainty.

Information Gain

Information gain is also called as Kullback-Leibler divergence denoted by IG(S,A) for a set S is the effective change in entropy after deciding on a particular attribute A. It measures the relative change in entropy with respect to the independent variables.

$$IG(S, A) = H(S) - H(S, A)$$

Alternatively,

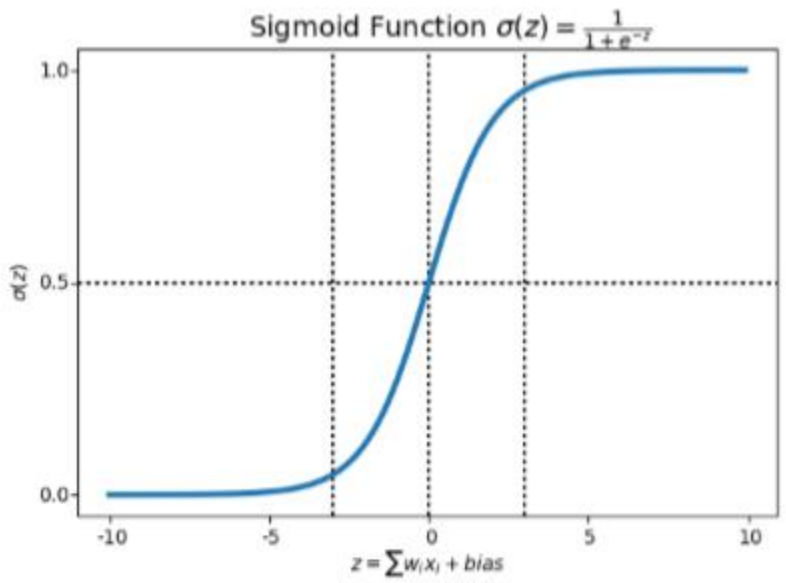
$$IG(S,A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

2.2 Logistic Regression

Logistic Regression is a Machine Learning algorithm which is used for the classification problems. it is a predictive analysis algorithm and based on the concept of probability. Logistic regression is probably the most widely used general-purpose classifier. It is very scalable and can be very fast to train.

Logistic regression extends the ideas of linear regression to the situation where the dependent variable, Y, is categorical. Logistic regression is designed as a binary classifier (output say {0,1}) but actually outputs the probability that the input instance is in the “1” class. Unlike ordinary linear regression, logistic regression does not assume that the relationship between the independent variables and the dependent variable is a linear one. Nor does it assume that the dependent variable or the error terms are distributed normally.

Logistic Regression uses a more complex cost function; this cost function can be defined as the ‘Sigmoid function’ or also known as the ‘logistic function’ instead of a linear function. The Sigmoid function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.



2.3 k - Nearest Neighbors

The k-nearest neighbors (kNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. However, it is more widely used in classification problems in the industry.

“kNN which stand for K Nearest Neighbor is a Supervised Machine Learning algorithm that classifies a new data point into the target class, depending on the features of its neighboring data points.”

Features of kNN Algorithm-

The kNN algorithm has the following features:

- kNN is a Supervised Learning algorithm that uses labeled input data set to predict the output of the data points.
- It is one of the simplest Machine learning algorithms and it can be easily implemented for a varied set of problems.
- It is mainly based on feature similarity. kNN checks how similar a data point is to its neighbor and classifies the data point into the class it is most similar to.
- Unlike most algorithms, kNN is a non-parametric model which means that it does not make any assumptions about the data set. This makes the algorithm more effective since it can handle realistic data.
- kNN is a lazy algorithm, this means that it memorizes the training data set instead of learning a discriminative function from the training data.
- kNN can be used for solving both classification and regression problems.

The kNN Algorithm -

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. K-nearest neighbors (kNN) algorithm uses ‘feature similarity’ to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps:

Step 1 − For implementing any algorithm, we need dataset. So during the first step of kNN, we must load the training as well as test data.

Step 2 − Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

Step 3 − For each point in the test data do the following –

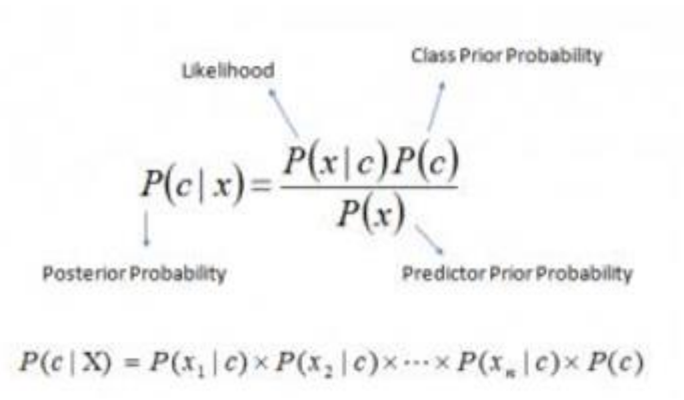
- 3.1 − Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.
- 3.2 − Now, based on the distance value, sort them in ascending order.
- 3.3 − Next, it will choose the top K rows from the sorted array.
- 3.4 − Now, it will assign a class to the test point based on most frequent class of these rows.

Step 4-End

2.4 Naïve Bayes

Naive Bayes classifier assumes that the presence of a feature in a class is unrelated to any other feature. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that a particular fruit is an apple or an orange or a banana and that is why it is known as “Naive”.

A Naive Bayes classifier is a probabilistic machine learning model that’s used for classification task. The crux of the classifier is based on the Bayes theorem. Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



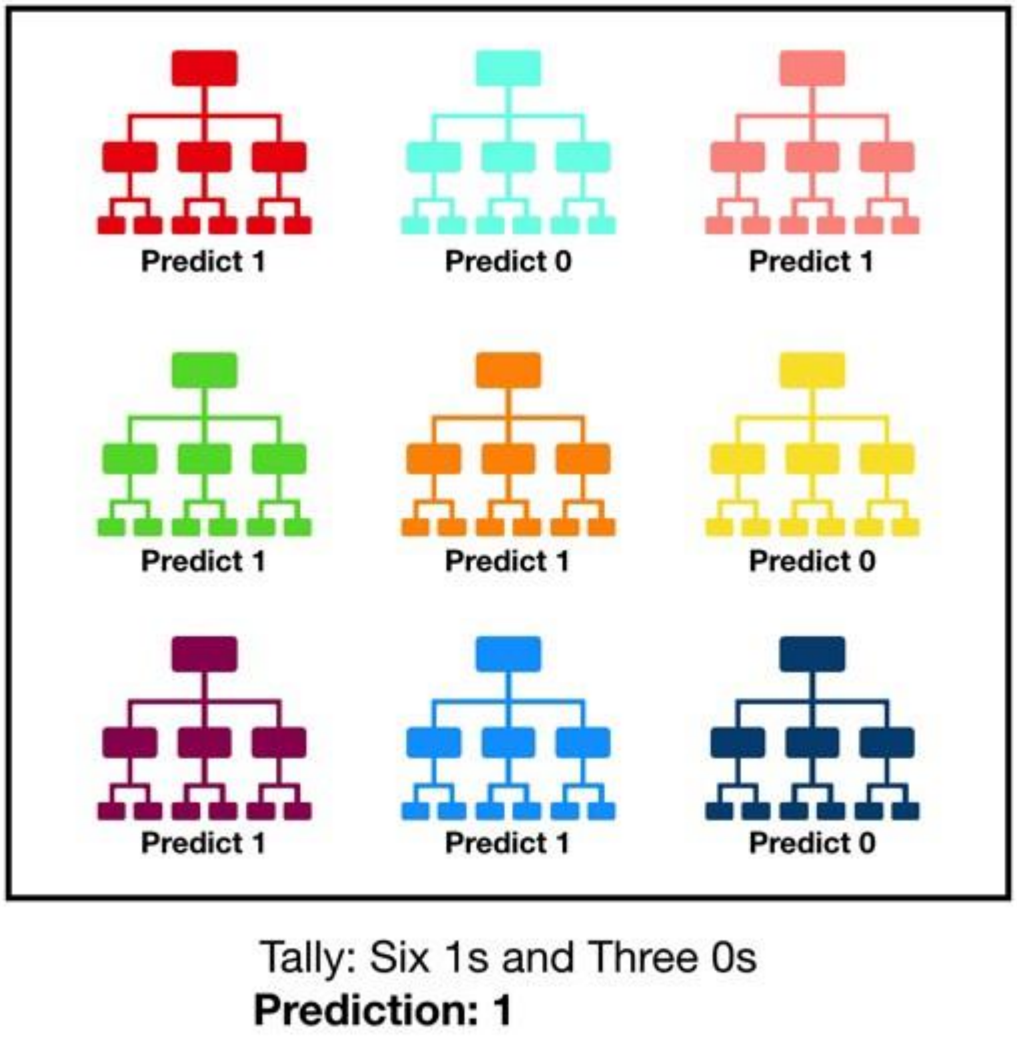
$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Above, P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes). P(c) is the prior probability of class. P(x|c) is the likelihood which is the probability of predictor given class. P(x) is the prior probability of predictor.

2.5 Random forest

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction (see figure below).



Visualization of a Random Forest Model Making a Prediction

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is:

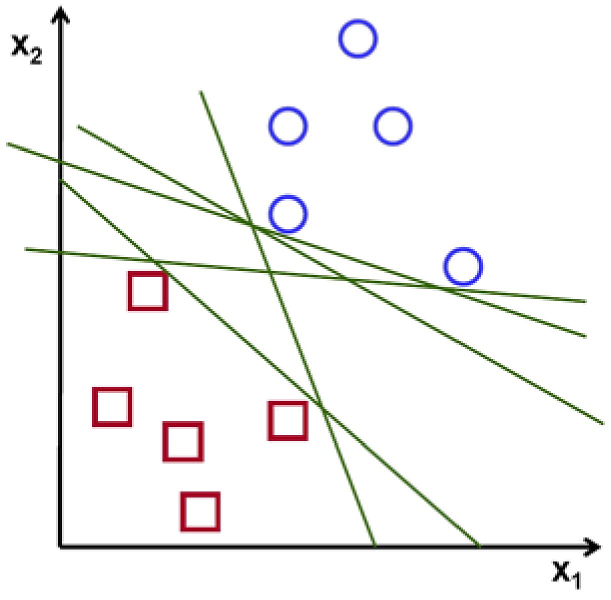
A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

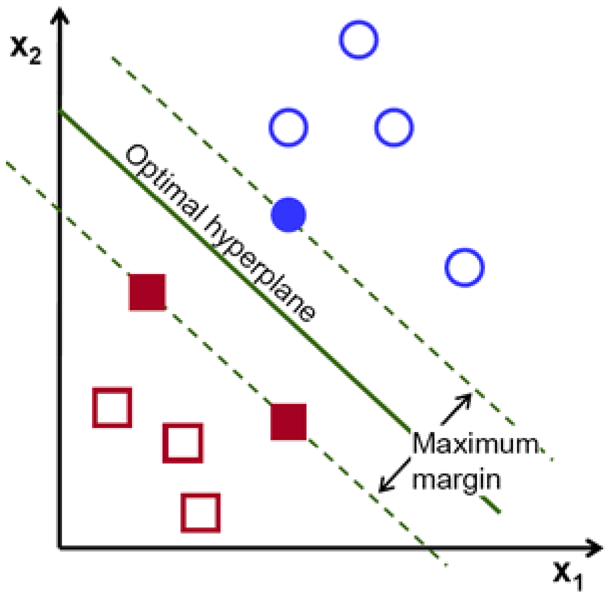
The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for random forest to perform well are:

1. There needs to be some actual signal in our features so that models built using those features do better than random guessing.
2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

2.6SVM

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.

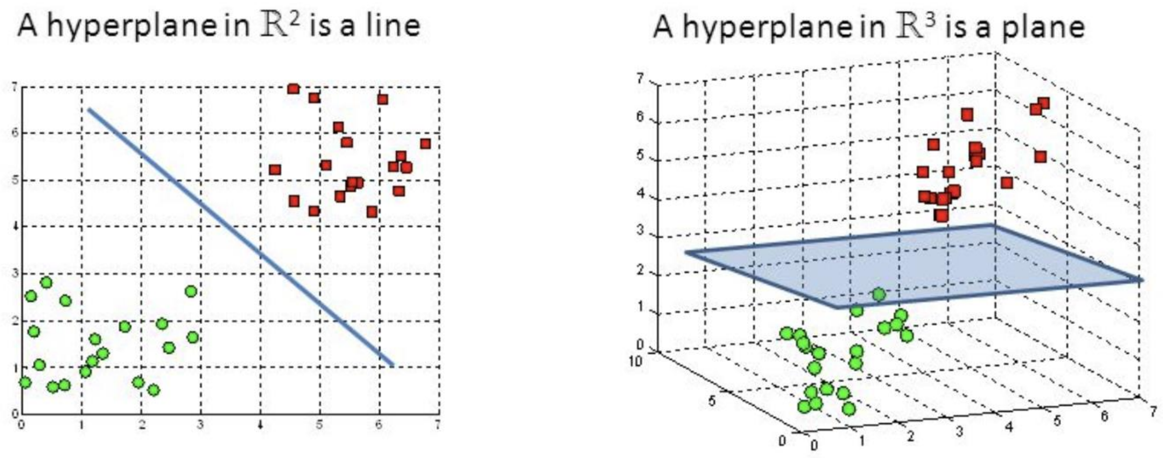




Possible hyperplanes

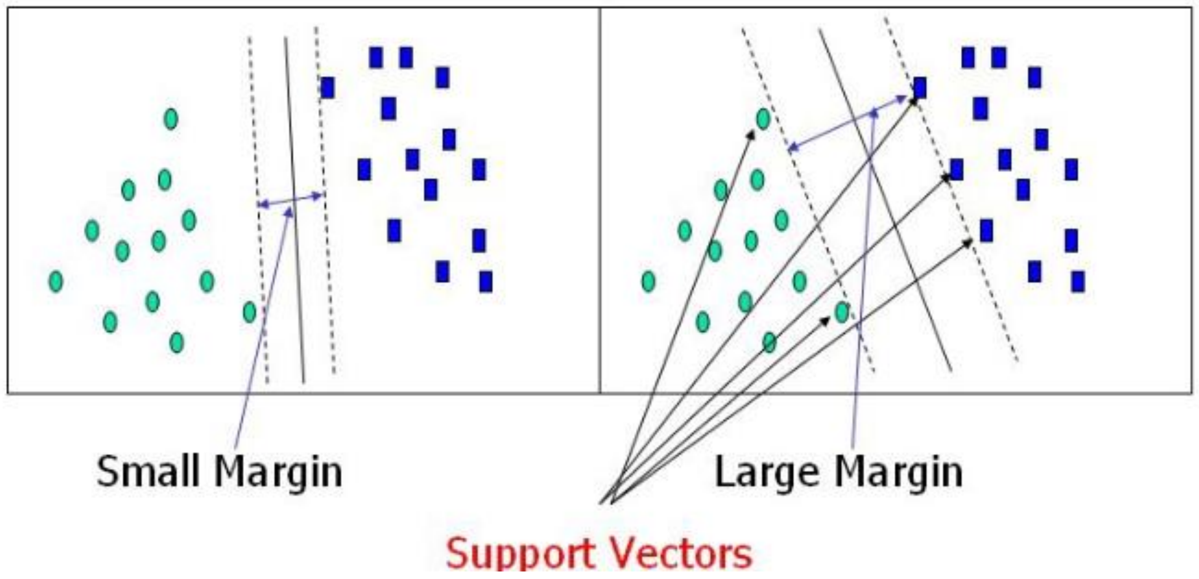
To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Hyperplanes and Support Vectors



Hyperplanes in 2D and 3D feature space

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.



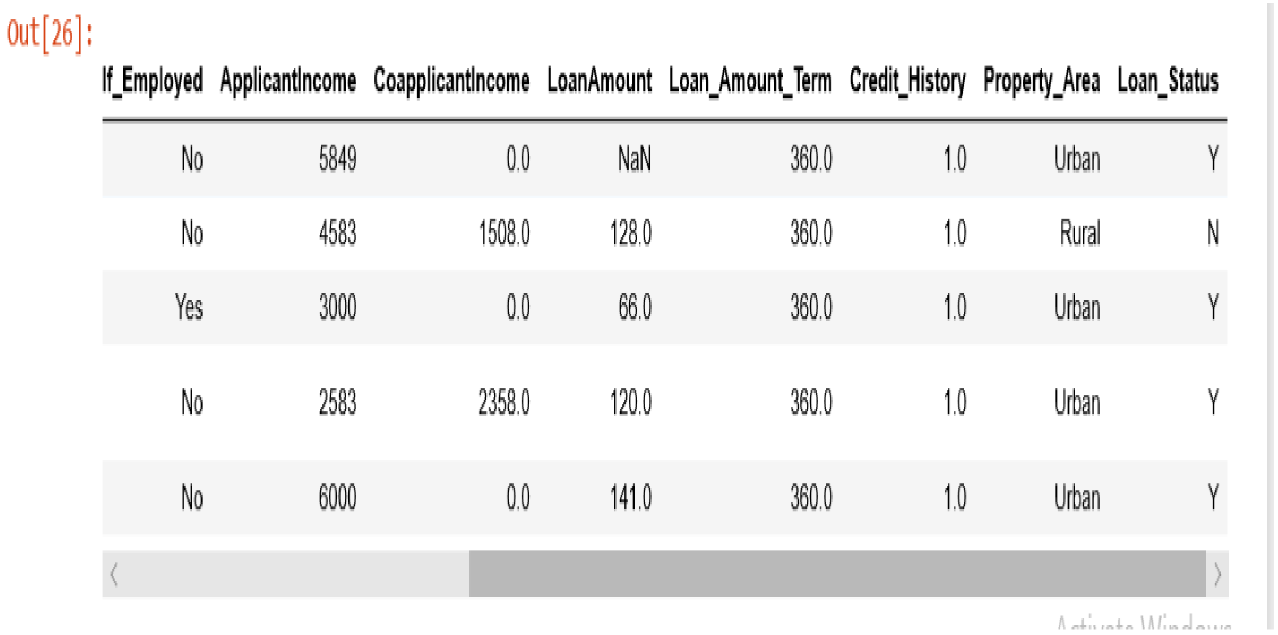
Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

CHAPTER 3

SOURCE CODE

3.1 Code

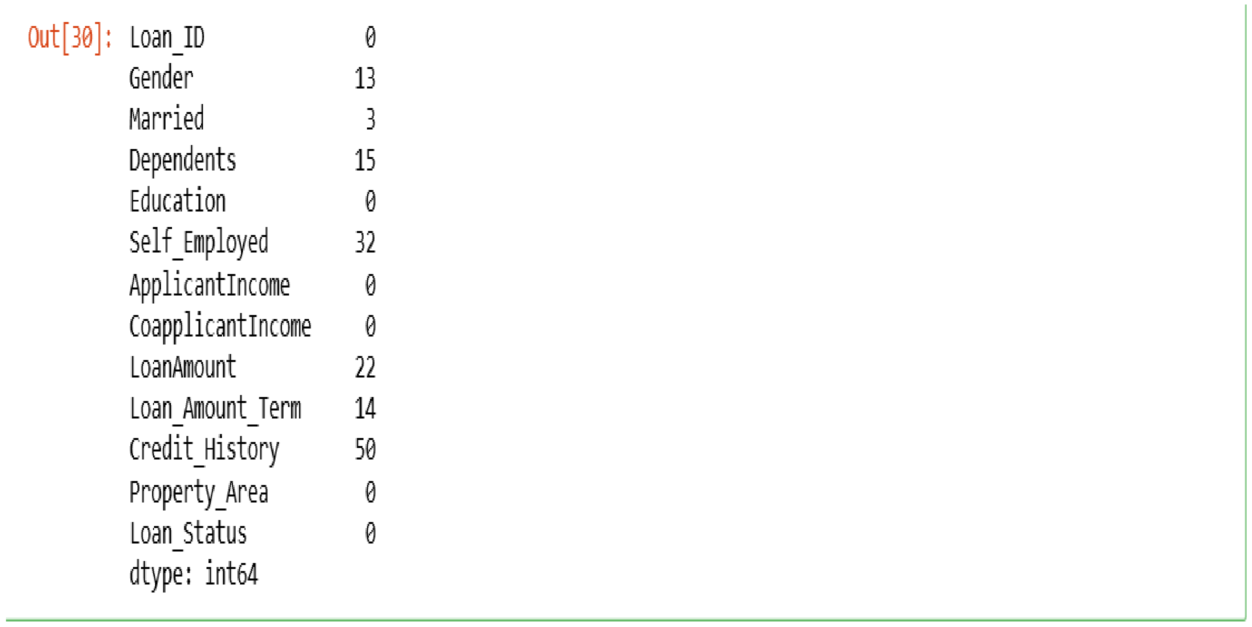
```
# Importing Library

import pandas as pd

import numpy as np

from sklearn import preprocessing

from sklearn.preprocessing import LabelEncoder

data = pd.read_csv("E:/datasets/dataa.csv")

data.head()
```



Out[26]:

lf_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
No	5849	0.0	NaN	360.0	1.0	Urban	Y
No	4583	1508.0	128.0	360.0	1.0	Rural	N
Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
No	2583	2358.0	120.0	360.0	1.0	Urban	Y
No	6000	0.0	141.0	360.0	1.0	Urban	Y

data.isnull().sum()



```
Out[30]: 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
```

data['LoanAmount'].fillna(data['LoanAmount'].mean(),inplace=True)

data['Self_Employed'].fillna('No',inplace=True)

data['Gender'].fillna(data['Gender'].mode()[0], inplace=True)

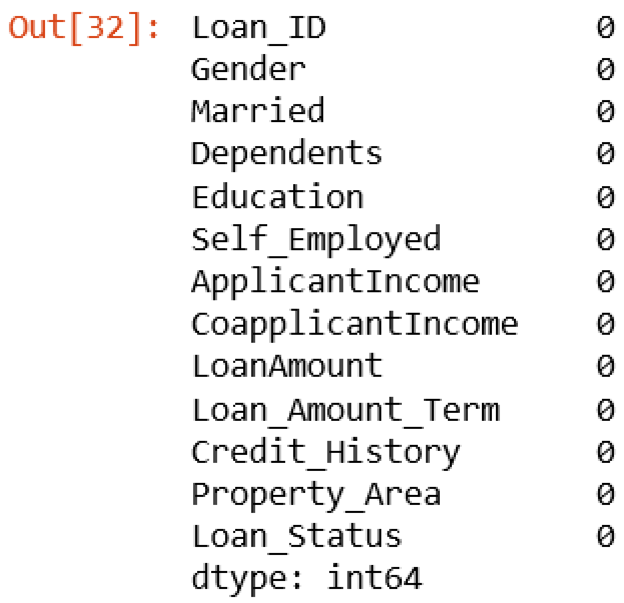
data['Married'].fillna(data['Married'].mode()[0], inplace=True)

data['Dependents'].fillna(data['Dependents'].mode()[0], inplace=True)

data['Loan_Amount_Term'].fillna(data['Loan_Amount_Term'].mode()[0], inplace=True)

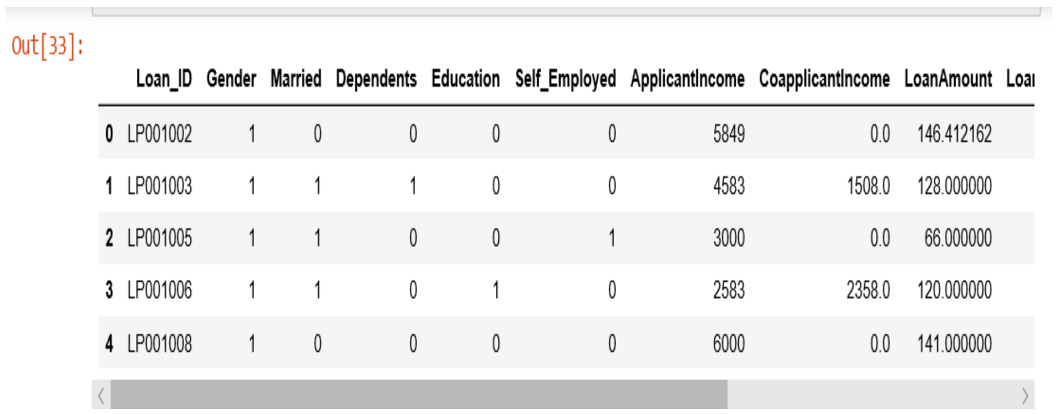
data['Credit_History'].fillna(data['Credit_History'].mode()[0], inplace=True)

```
data.isnull().sum()
```



```
Out[32]: 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
```

```
from sklearn.preprocessing import LabelEncoder

var_mod = ['Gender','Married','Dependents','Education','Self_Employed','Property_Area','Loan_Status']

     le = LabelEncoder()

     for i in var_mod:

       data[i] = le.fit_transform(data[i])

     data.head()
```



```
Out[33]:
   Loan_ID  Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome  CoapplicantIncome  LoanAmount  Loa
0  LP001002      1        0           0          0              0             5849                0.0  146.412162
1  LP001003      1        1           1          0              0             4583             1508.0  128.000000
2  LP001005      1        1           0          0              1             3000                0.0   66.000000
3  LP001006      1        1           0          1              0             2583             2358.0  120.000000
4  LP001008      1        0           0          0              0             6000                0.0  141.000000
```

X = data[['Credit_History','Gender','Married','Education']]

y = data['Loan_Status']

from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random_state = 0)

#Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)

classifier.fit(X_train, Y_train)

predictions = model.predict(X)

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(Y_test, Y_pred)

cm

from sklearn.metrics import accuracy_score

accuracy = accuracy_score(Y_test,Y_pred)

accuracy

#Accuracy

0.83116883116883122


from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state = 0)

classifier.fit(X_train, Y_train)

predictions = model.predict(X)

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(Y_test, Y_pred)cm

from sklearn.metrics import accuracy_score

accuracy = accuracy_score(Y_test,Y_pred)accuracy

#Accuracy

0.83015883116883121
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, Y_train)
predictions = model.predict(X)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, Y_pred)
cm
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_test,Y_pred)
accuracy
#Accuracy
0.82116003116883125

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, Y_train)
```

```
predictions = model.predict(X)

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(Y_test, Y_pred)

cm

from sklearn.metrics import accuracy_score

accuracy = accuracy_score(Y_test,Y_pred)

accuracy

#Accuracy

0.82266883116883158
```

```
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)

classifier.fit(X_train, Y_train)

predictions = model.predict(X)

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(Y_test, Y_pred)

cm
```

```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(Y_test,Y_pred)

accuracy

#Accuracy

0.83102688311688310



from sklearn.svm import SVC

classifier = SVC(kernel = 'linear', random_state = 0)

classifier.fit(X_train, Y_train)

predictions = model.predict(X)

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(Y_test, Y_pred)

cm

from sklearn.metrics import accuracy_score

accuracy = accuracy_score(Y_test,Y_pred)

accuracy

#Accuracy

0.83016883116842310
```

CHAPTER 4

CONCLUSION

After applying the different classification models, we have got below accuracies with different models:

1. Logistic Regression — 83.0%

2. Nearest Neighbor — 82.1%

3. Support Vector Machines — 83.0%

4. Naive Bayes — 82.2%

5. Decision Tree Algorithm — 83.1%

6. Random Forest Classification — 83.1%

So finally we have built our classification model and we can see that Random Forest Classification algorithm and . Decision Tree Algorithm give the best results for our dataset. Well its not always applicable to every dataset. To choose our model we always need to analyze our dataset and then apply our machine learning model.

REFERENCES

www.towardsdatascience.com

www.kaggle.com

www.geeksforgeeks.com