A Project Report on

Enhancement of Machine Learning Classification Algorithms for Bank Loan Approval using Stacking

Submitted in partial fulfillment for award of

Bachelor of Technology

Degree

in

Computer Science & Engineering

By

SK. Sameer (Y18ACS577) **M. Avinash (Y18ACS512)**

P. V. Pavan Kalyan (Y18ACS531) **P. Jessi Pramodini (L19ACS589)**

Under the guidance of

Mr. J. Madhan Kumar M. Tech, (Ph.D)

Department of Computer Science and Engineering

Bapatla Engineering College

(Autonomous)

(Affiliated to Acharya Nagarjuna University)

BAPATLA – 522102, Andhra Pradesh, INDIA

2021-2022

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project report entitled **Enhancement of Machine Learning Classification Algorithms for Bank Loan Approval using Stacking** that is being submitted by A.Sameer SK (Y18ACS577), B.Avinash M (Y18ACS512), C.Pavan Kalyan P.V (Y18ACS531) and D.Jessi Pramodini P (L19ACS589) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science & Engineering to the Acharya Nagarjuna University is a record of bonafide work carried out by them under our guidance and supervision.

Date:

Signature of the Guide
Mr. J. Madhan Kumar
Assistant Professor

Signature of HOD
Dr. SK. Nazeer
Prof. & Head

DECLARATION

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

SK. Sameer (Y18ACS577)

M. Avinash (Y18ACS512)

P.V. Pavan Kalyan (Y18ACS531)

P. Jessi Pramodini (L19ACS589)

Acknowledgement

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide **Mr. J. Madhan Kumar**, Asst. Professor, Department of CSE, for his valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Dr. SK Nazeer**, Prof. & Head of the Dept. for extending his cooperation and providing the required resources.

We would like to thank our beloved Principal **Dr. V. Damodara Naidu** for providing the online resources and other facilities to carry out this work.

We would like to express our sincere thanks to our project coordinator **Dr. N. Sudhakar,** Prof., Dept. of CSE for his helpful suggestions in presenting this document.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly for their cooperation and encouragement.

SK. Sameer (Y18ACS577)

M. Avinash (Y18ACS512)

P.V. Pavan Kalyan (Y18ACS531)

P. Jessi Pramodini (L19ACS589)

Abstract

Machine learning is playing a prominent role in current era. In this modernized world almost all the applications are manipulated and controlled by machine learning algorithms. By the use of historical data there are possibilities to predict the future. Even though a number of researchers are working on various machine learning, the performance and exactness of the algorithms still remains as a challenge. This work focuses on the performance analysis of various classification algorithms like Logistic Regression, Gaussian Naïve Bayes, Decision Tree Classifier and Random Forest Classifier in terms of confusion matrix, accuracy, precision, recall, f-measure etc., and performance enhancement of those algorithms using stacking to predict the bank loan approval status so we can overcome the need to depend on a single machine learning model instead we can combine multiple models so to obtain a stacked model which provides better predictions as compared to each individual model.

Table of Contents

List of Figures

1. Introduction

Machine Learning is utilized to show machines how to handle the information all the more efficiently. Once in a while in the wake of review the information, we can't decipher the example or concentrate data from the information. All things considered, we apply machine learning. With the plenitude of data set accessible, the interest for machine learning is in rise. Numerous businesses from drug to military apply machine figuring out how to extricate applicable data. The present overwhelming worldview for ML is to run a ML calculation on an offered dataset to produce a model. The model is then connected, all things considered, and the assignments are executed. This is valid for both supervised and unsupervised learning.

Major Machine Learning Types:

1. Supervised Learning

2. Unsupervised Learning

3. Reinforcement Learning

1.1 Supervised Learning:

In training data set contains the features with target values. This is likewise called gaining from the models. This is generally composed as a lot of information (x_i, t_i), where the sources of info are x_i, the objectives are t_i , ordered by running from 1 to some maximum limit N.

1.1.1 Regression

Regression is a supervised learning approach considered when our target value(t_i) is a continuous value. It is a straight wat to deal with demonstrating the connection between a scalar reaction (or dependent variable) and at least one illustrate factors (or independent).

1.1.2 Classification

Regression is a supervised learning approach considered when our target value(t_i) is a categorical value.

1.2 Unsupervised Learning:

Unsupervised learning is a lot harder on the grounds that here the model need to figure out how to perform indicated assignments without revealing to it how to perform because the dataset will only have input features without any target values. Clustering is one of the approach to deal with unsupervised learning.

1.3. Reinforcement Learning(RL):

This learning lies between supervised and unsupervised learning. The calculation finds told when the solution is wrong, yet does not get advised how to right it. It needs to investigate and experiment with various potential outcomes until find the solution right. RL some time called learning with a commentator as a result of this screen scores the appropriate response, however does not recommend upgrades.

2. Literature Survey

This chapter contains a list of literature review of previous research where it is considered vital in development of this project. [1] With the plenitude of data set accessible, the interest for machine learning is in rise. Numerous businesses from drug to military apply machine figuring out how to extricate applicable data. Naive Bayes', which can be amazingly quick in respect to other order calculations. [5] It takes a shot at Bayes hypothesis of likelihood to foresee the class of unknown dataset. Regarding Logistic Regression Tabachnick and Fidell(2013) recommend that as long relationship coefficients among autonomous factors are under .90 the suspicion is met. At the Focal point of the logistic regression examination is undertaking evaluating the log chances of an occasion. In case of random forest, If one tree is extraordinary, various trees(a forest) should be better, given that there is adequate combination between them. [4] The most fascinating thing about the random forest is the habits in which that it makes randomness from a standard dataset. In the essence, stacking makes prediction by using a meta-model trained from a pool of base models — a pool of base models are first trained using training data and asked to give their prediction; a different meta model is then trained to use outputs from base models to give the final prediction. As machine learning is a research intensive field constant performance and behavioral analysis and striving to improve existing algorithms using innovative techniques or developing new algorithms or approaches are mandatory.

3. Problem Statement

In any machine learning based solution the basic approach involves taking the most suitable machine learning algorithm and obtaining a model through training on the well pre-processed dataset and then testing that model with the test data to analyze it's performance. Sometimes this model maybe underfitted or overfitted which requires complex tuning process which may or may not improve model's performance. Hence using more than one algorithm and preparing multiple models and then combining them using stacking approach in such a way that will improve prediction capabilities of a stacked model as compared to individual model is definitely a better choice.

The main objective of the proposed system is to ***analyze and enhance the performance of various machine learning classification algorithms through extensive stacking***.

To Achieve such an objective a problem statement satisfying being a classification problem is considered.

Problem Statement: Whether a person will be approved for personal bank loan?

Target value:	Yes (or) 1 if a person is approved to be granted by a personal loan.
	No (or) 0 if a person is not approved to be granted by a personal loan.

The above target values specifies that the problem statement is a binary classification problem, as there are only two possible categories or target labels.

4. Algorithms and Methodology

4.1 Classification Algorithms

1.1 Logistic Regression

1.2 Decision Tree Classifier

1.3 Gaussian Naïve Bayes Classifier

1.4 Random Forest Classifier

4.1.1 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 Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

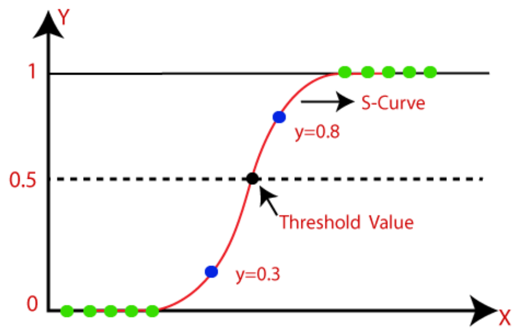


Figure 4.1 Sigmoid Function

Logistic Regression Equations

$P\ (Y = 0 / X) = 1 / 1 + e^z$ [max for class 0] **[**Equation 3.1]

$P\ (Y = 1/ X) = e^z/ 1 + e^z$ [max for class 1] **[**Equation 3.2]

where $z = w_o x_o + w_1 x_1 + w_2 x_2 + \ldots\ldots + w_n x_n$ [network sum] [Equation 3.3]

Y: target label [yes/no (or) 1/0]

X: sample

P(Y = 0 / X): probability that a sample X will belong to class 0

P(Y = 1 / X): probability that a sample X will belong to class 1

4.1.2 Decision Tree Classifier

- It is a Supervised Machine Learning where the data is continuously split according to a certain parameter.
- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

- For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

  Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

  Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

  Step-3: Divide the S into subsets that contains possible values for the best attributes.

  Step-4: Generate the decision tree node, which contains the best attribute.

  Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

- A Decision Tree consists of :

  i) Nodes : Test for the value of a certain attribute.

  ii) Edges/ Branch : Correspond to the outcome of a test and connect to the next node or leaf.

  iii) Leaf nodes : Terminal nodes that predict the outcome (represent class labels or class distribution).

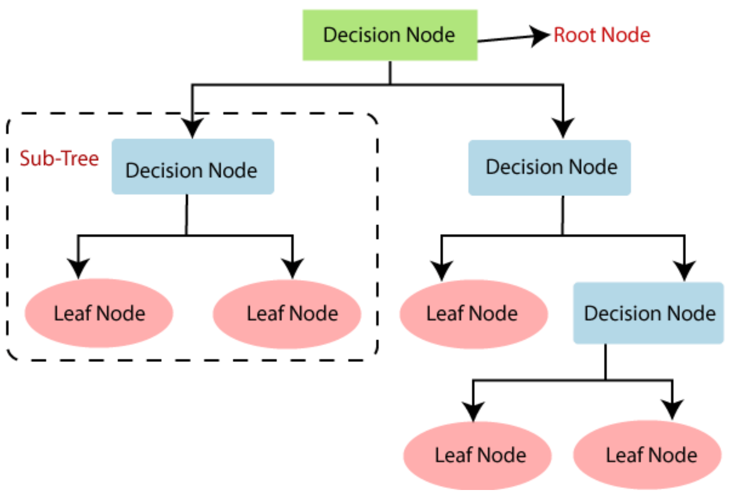


Figure 4.2 Decision Tree

ASM (Attribute Selection Measures)

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.**

i) Information Gain

Information gain is the measurement of changes In entropy after the segmentation of a dataset based on an attribute. It calculates how much information a feature provides us about a class. According to the value of information gain, we split the node and build the decision tree.A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

$$IG = Entropy(S) - [(Weighted\ Avg) * Entropy(each\ feature)] \qquad \text{[Equation 3.4]}$$

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

$$Entropy(s) = -P(yes)\log_2 P(yes) - P(no)\log_2 P(no) \quad \text{[Equation 3.5]}$$

4.1.3 Gaussian Naïve Bayes Classifier

Naïve Bayes classifier is constructed from Bayes Theorem with a assumption. Naïve Bayes Classifier is a multi-class classifier.

According to the Bayes Theorem: $P(Y/X) = P(X/Y)\,P(Y) / P(X)$ [Equation 3.6]

Where *$P(Y/X)$= probability of Y occurring given evidence X has already occurred[posterior]*

$P(X/Y)$= probability of X occurring given evidence Y has already occurred[likelihood]

$P(X)$= probability of X occurring [class prior probability]

$P(Y)$= probability of Y occurring [predictor prior probability]

Why Naïve?

In addition to the Naïve Bayes theorem we make a assumption that every feature/attribute is independent of other features and every feature/attribute contributes towards predicting target class. This is called as *Conditional Independence.*

Hence we have:

$P(Y/X) = P(x_1/Y) * P(x_2/Y) * P(x_3/Y) * \ldots\ldots\ldots P(x_n/Y) * P(Y)$ [Equation 3.7]

We consider Gaussian Naïve Bayes when our input features are of continuous values instead of categorical.

To determine $P(X_i/Y_i)$ we us e probability density function[conditional probability]:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$ [Equation 3.8]

Where $\mu = \sum X/N$ [mean] [Equation 3.9]

$\sigma_s = \sqrt{(\sum(X-\mu)^2 / N-1)}$ [Standard Deviation of samples] [Equation 3.10]

N= number of samples

4.1.4 Random Forest Classifier

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, *"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."* Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees In the forest leads to higher accuracy and prevents the problem of overfitting.

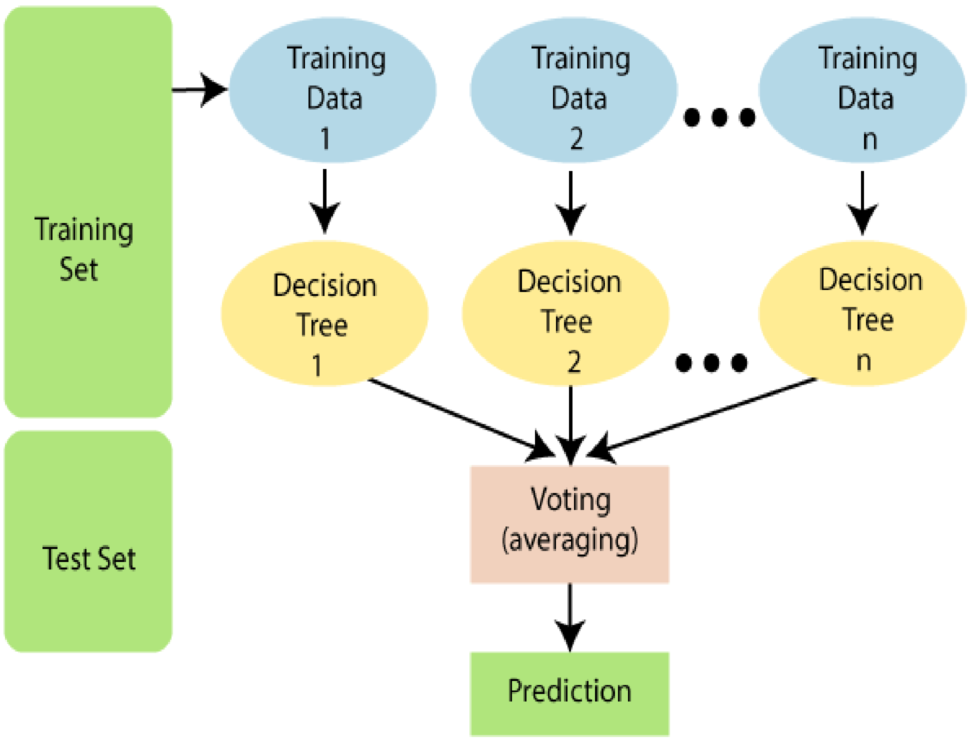


Figure 4.3 Random Forest Classifier

5. Stacking

Stacking is an ensemble machine learning algorithm. It uses a meta-learning algorithm to learn how best combine the predictions from two or more base machine learning algorithms. The benefit of stacking is that it harnesses the capabilities of a range of well-performing models on a classification or regression task and make predictions that have better performance than any single model in the ensemble.

Hence by using stacking we will be able to create a new classifier that hopefully will give better accuracy compared to all other base models.

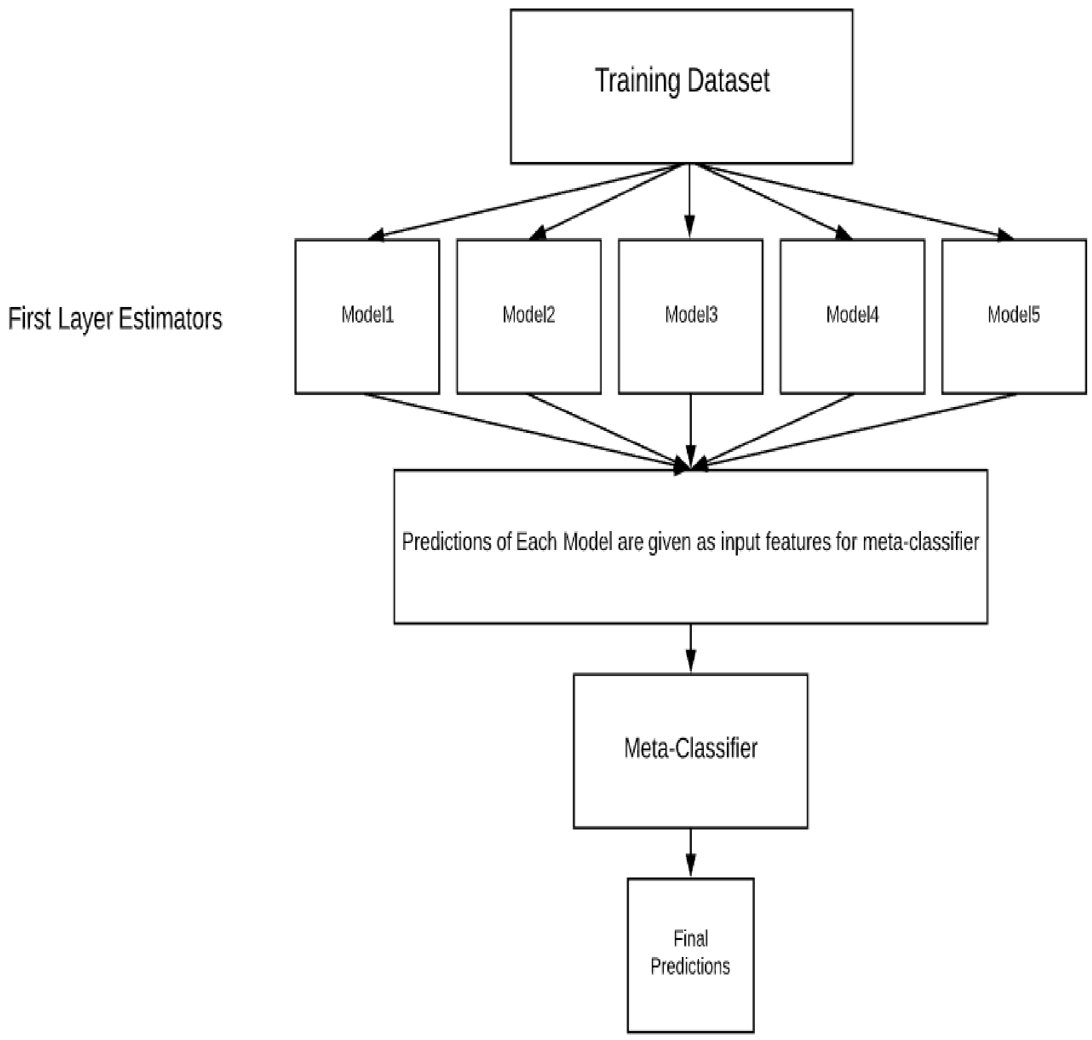


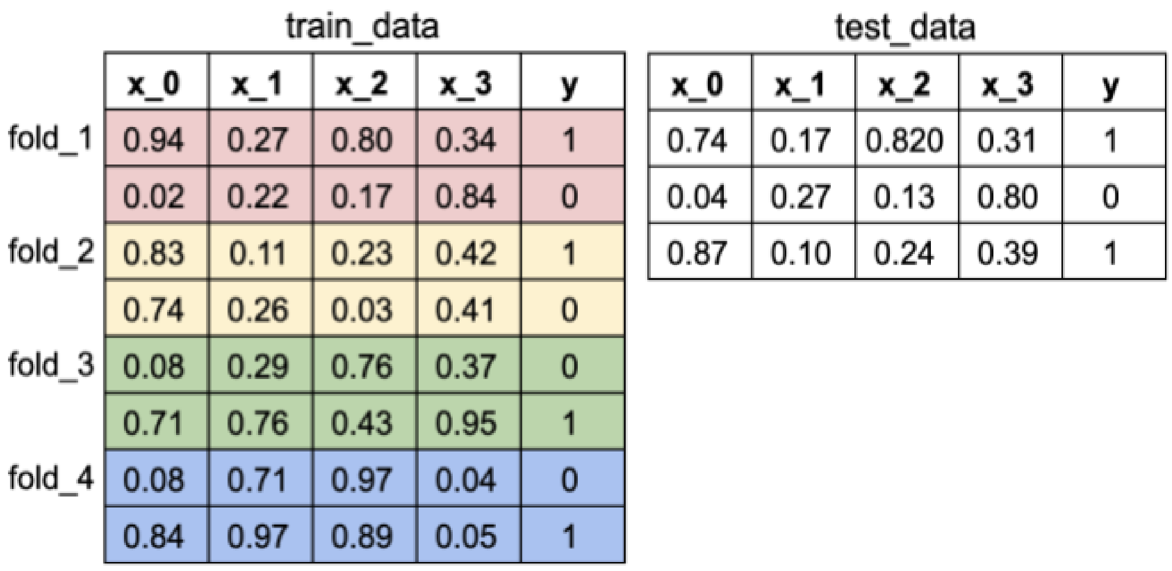
Figure 5.1 Stacking Architecture

5.1 UNDERSTANDING STACKING

In the essence, stacking makes prediction by using a meta-model trained from a pool of base models — a pool of base models are first trained using training data and asked to give their prediction; a different meta model is then trained to use outputs from base models to give the final prediction. The process is actually simple. To train a base model, K-fold cross validation technique is used.

We can understand the stacking in **7** steps:

Step 1: You have Train Data and Test Data as shown in Fig 3.5. Assume we are using 4-fold cross validation to train base models, the train_data is then divided into 4 parts.



train_data

	x_0	x_1	x_2	x_3	y
fold_1	0.94	0.27	0.80	0.34	1
	0.02	0.22	0.17	0.84	0
fold_2	0.83	0.11	0.23	0.42	1
	0.74	0.26	0.03	0.41	0
fold_3	0.08	0.29	0.76	0.37	0
	0.71	0.76	0.43	0.95	1
fold_4	0.08	0.71	0.97	0.04	0
	0.84	0.97	0.89	0.05	1

test_data

x_0	x_1	x_2	x_3	y
0.74	0.17	0.820	0.31	1
0.04	0.27	0.13	0.80	0
0.87	0.10	0.24	0.39	1

Figure 5.2 Train Data and Test Data

Step 2: Using the 4-part train_data, the 1st base model (assuming it's a decision tree) is fitted on 3 parts and predictions are made for the 4th part. This is done for each part of the training data. At the end, all instance from training data will have a prediction. This creates a new feature for train_data, call it pred_m1 (predictions model 1) as shown in fig 3.6.

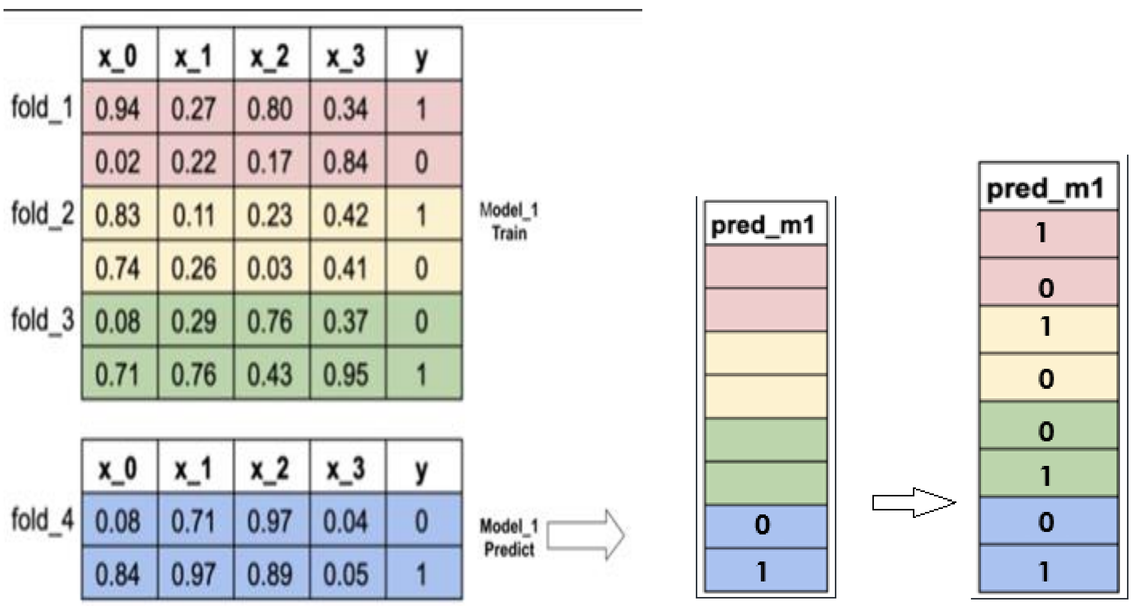


Figure 5.3 Prediction from model_1

Step 3: Step 2 is repeated for the 2nd model (e.g. GNB) and the 3rd model (e.g. Random Forest). These will give two more predictions, pred_m2 and pred_m3.

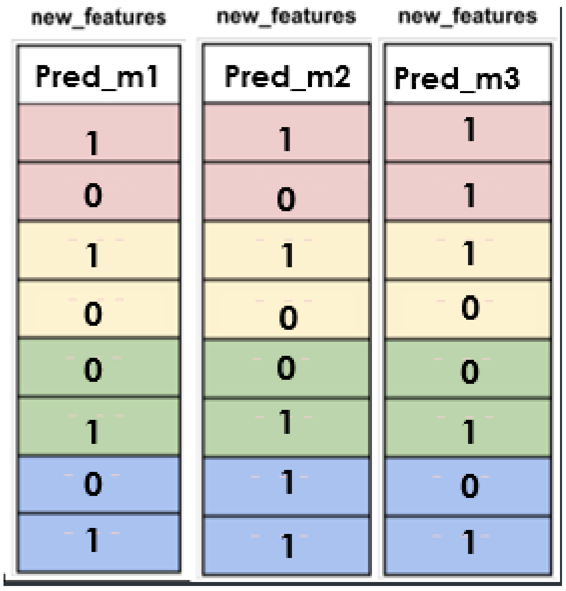
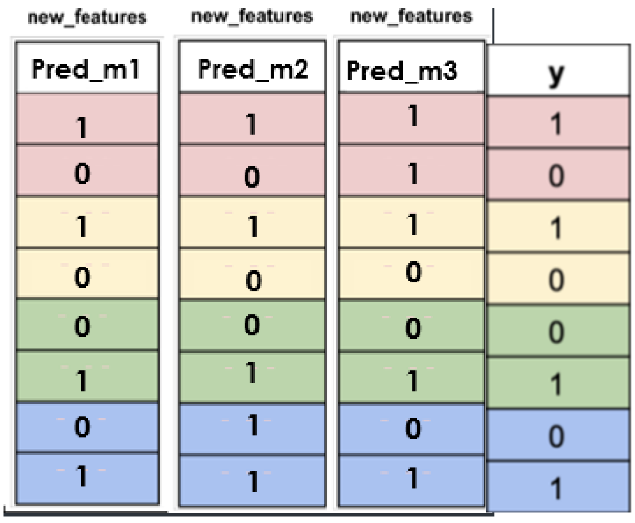


Figure 5.4 Prediction from model 1,2 and 3.

Step 4: Combine the above 3 prediction along with the actual target column ‘y’ from the train_data to obtain a dataset with predicted outcomes as input features (from each model) and actual outcome from ‘y’.



new_features	new_features	new_features	
Pred_m1	Pred_m2	Pred_m3	y
1	1	1	1
0	0	1	0
1	1	1	1
0	0	0	0
0	0	0	0
1	1	1	1
0	1	0	0
1	1	1	1

Figure 5.5 Train_Data for Meta-Model

Step 5: Select you desired meta model for ex: Logistic Regression, and train this model on the dataset created in step 4.

Step 6: The meta model(Logistic Regression) is now ready to predict if given a test sample but if we observe we can’t directly provide our real input features to the meta model as it’s trained on 3 features which are actually prediction from different models. So, first we need to freshly initialize our base models(GNB, Decision Tree, Random Forest) and train them on whole train_data without any k-folding.

Step 7: Predict the test data by using each base model and combine the predictions as a single data frame. Give this data frame as a input to our meta-model for prediction which finally will give our desired prediction.

6. Performance Measures

Analysis of performance of models is very important to ensure the correctness of model. Accuracy may provide good overview of how our model has performed but it isn’t enough to analyze where our model is lagging, so we can put effort to overcome those problems. Hence with using quality and different measures we can analyze our model much better.

Confusion Matrix

A confusion matrix is a table that is often used to *describe the performance of a classification model* (or "classifier") on a set of test data for which the true values are known.

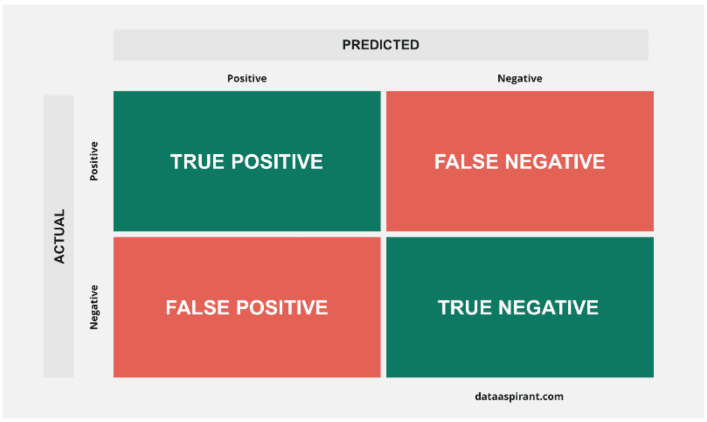


Figure 6.1 Confusion Matrix

In a confusion matrix the Actual values are represented along the rows whereas the Predicted Values are represented along the columns.

- **True Positive (TR) :** these are the no of sample which are predicted as positive and they are positive in real.

- **True Negatives (TN):** these are the no of sample which are predicted as negative and they are negative in real.
- **False Positive (FP):** these are the no of sample which are predicted as positive and they are negative in real.
- **False Negative (FN):** these are the no of sample which are predicted as negative and they are positive in real.

From the confusion matrix we obtain different performance measure to sufficiently analyze our model.

Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$ [Equation 3.11]

Sensitivity or Recall or True Positive Rate = $\frac{TP}{TP+FN}$ [Equation 3.12]

Specificity or True Negative Error = $\frac{TN}{TN+FP}$ [Equation 3.13]

Precision = $\frac{TP}{TP+FP}$ [Equation 3.14]

F measure = 2 $* \frac{Precision*Recall}{Precision+Recall}$ [Equation 3.15]

Classification Error = $\frac{Error}{Total} = \frac{FP+FN}{TP+TN+FP+FN}$ [Equation 3.16]

- Hence for the successful implementation of the proposed system three objectives must be reached as listed below and explained above :

  1. **Extensive Data Preprocessing Techniques.**
  2. **Stacking for Model Enhancement.**
  3. **Quality Performance Measures for proper model analysis.**

- At the we are optimistic that our new model (stacking) would provide better performance in most of the aspects (performance measures) compared to all individual base models.

7. Design

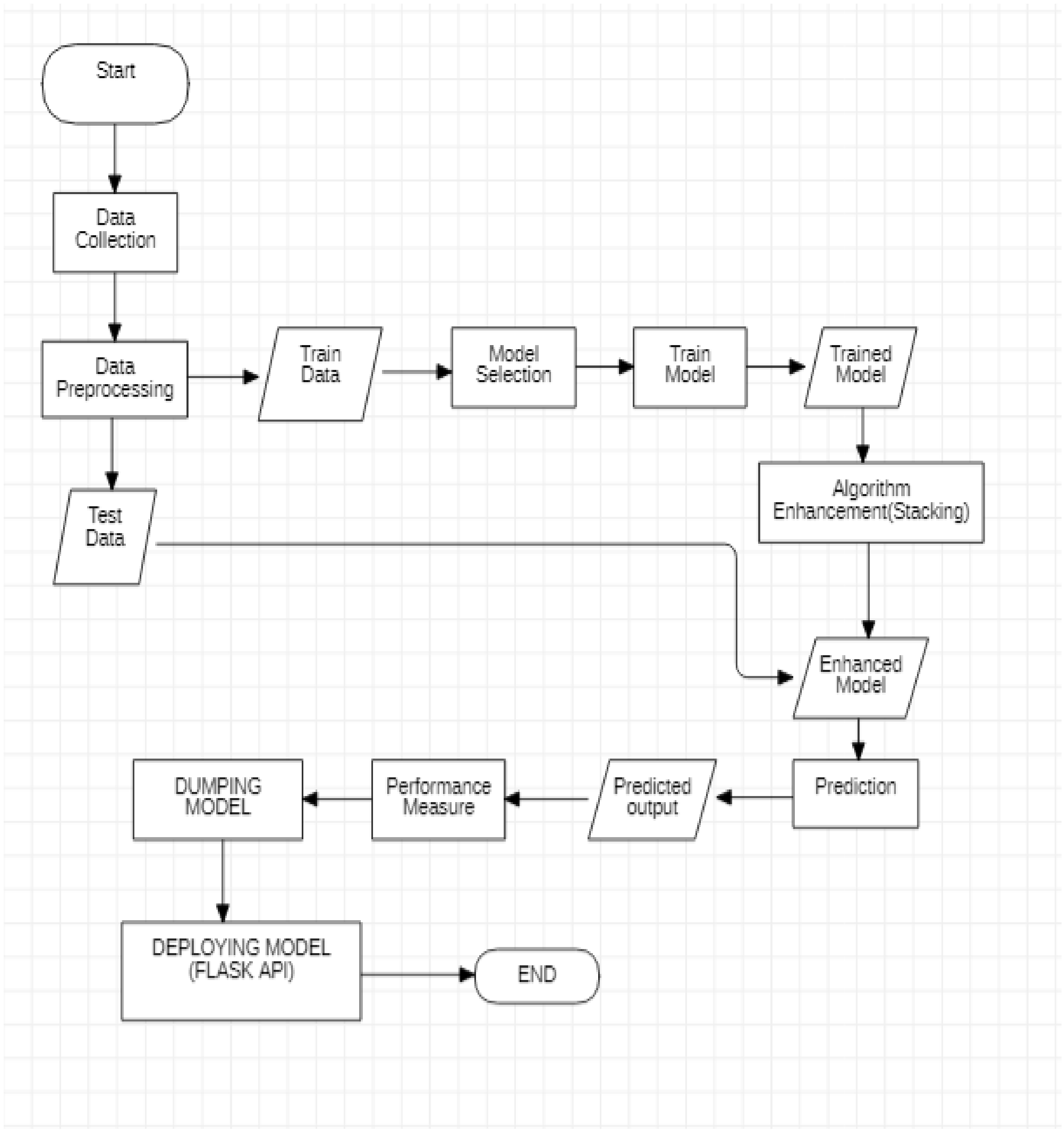


Figure 7.1 Flowchart of Design

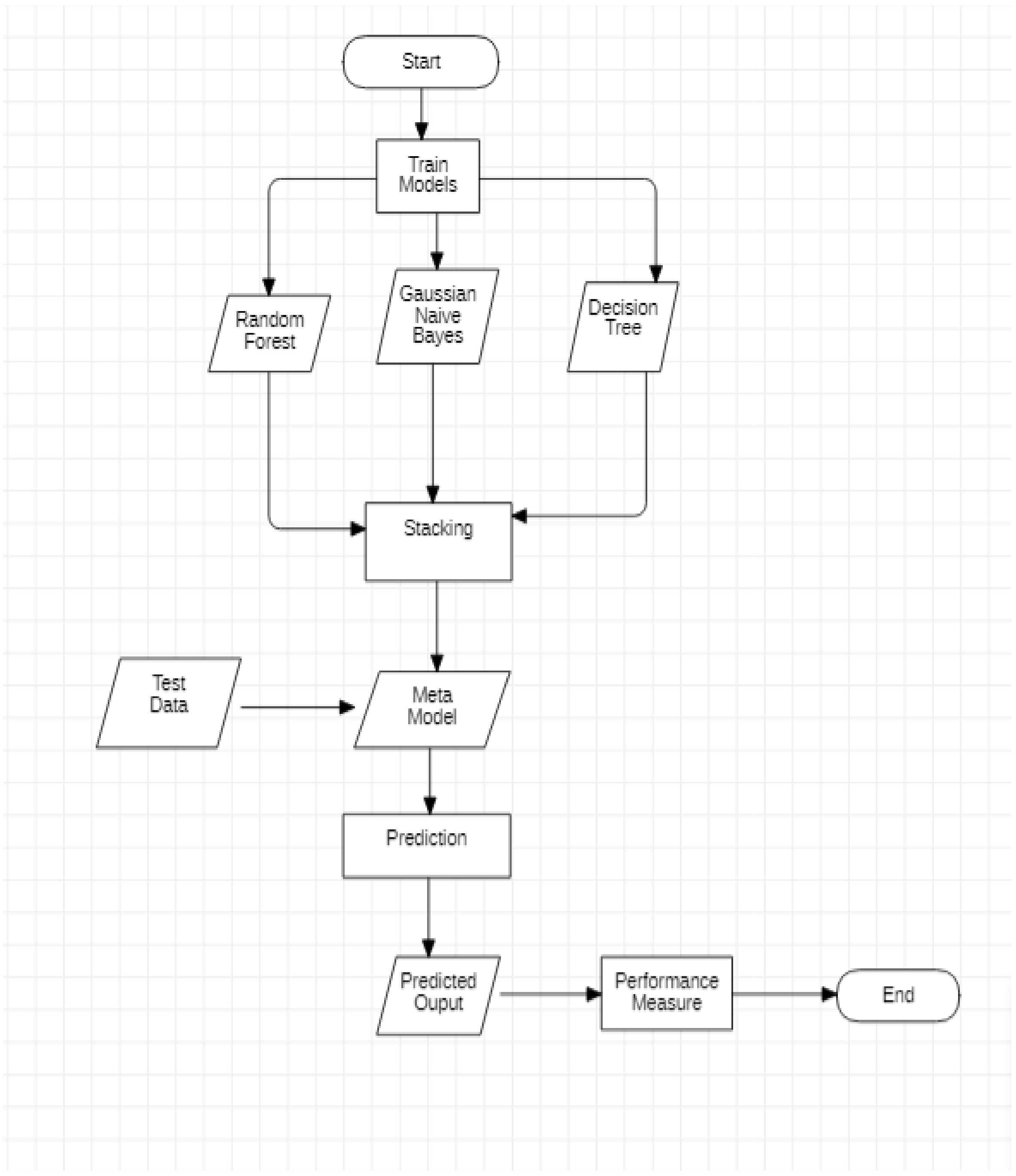


Figure 7.2 Flowchart of Stacking

8. Hardware and Software Requirements

8.1 Software Requirements

- In this project we have use Jupyter Notebook which is integrated in Anaconda Navigator Version 4.11.0.
- We have used Spyder IDE integrated in Anaconda Navigator to deploy Machine Learning models through Flask API.
- We also used a Web Browser to access Localhost port.
- We require basic machine learning classification algorithms (Random Forest Classifier, Gaussian Naïve Bayes Classifier, Decision Tree Classifier and Logistic Regression) and packages like (Pandas, NumPy, Matplotlib and Seaborn).

8.2 Hardware Requirements

- Operating System with Windows 7 or above.
- RAM of 2 GB (Minimum)
- Hard Disk : 1 GB
- Processor : i3 or Above

8.3 Dataset

- https://www.kaggle.com/itsmesunil/bank-loan-modelling

9. Code Implementation

In this project, you will learn how to combine various machine learning classification algorithms using stacking approach, so to obtain a better model that can predict better as compared to individual models.

As we require to adopt a problem statement to implement a classification algorithm we have taken "Bank Loan Approval" problem which a binary classification problem, where a person will be either be approved for a personal loan or not.

Initially we take our desired classification algorithms whom we want to combine which are called as our "Base Models". In this project we have taken Random Forest Classifier, Decision Tree Classifier, Gaussian Naïve Bayes Classifier and Gradient Boosting Classifier as our base models. We perform stacking approach explained in detail in section 6 on this 4 base models and hence obtain a train dataset for our meta model which is in our case is Logistic Regression.

After we obtain a train dataset for our meta model we train our meta-model(Logistic Regression) on that dataset. Now our meta model is trained and ready for prediction. But we can't give our initial dataset directly as test data to meta-model because meta-model is trained on four features that are the predictions from four base models, not on features of our actual dataset. So, firstly we need to freshly create our base model instances after training meta-model. Then, we have to train these four new instances of base models with train data. Now our base models are trained and ready for prediction.

To predict a sample first give it to base models as test sample. We get four predictions from our four base models. Combine these 4 prediction into a single dataframe and then give this new sample as test data to our meta-model to obtain final prediction.

To implement the project we will follow a general machine learning workflow.

9.1 Data Preprocessing

1. Import required packages.

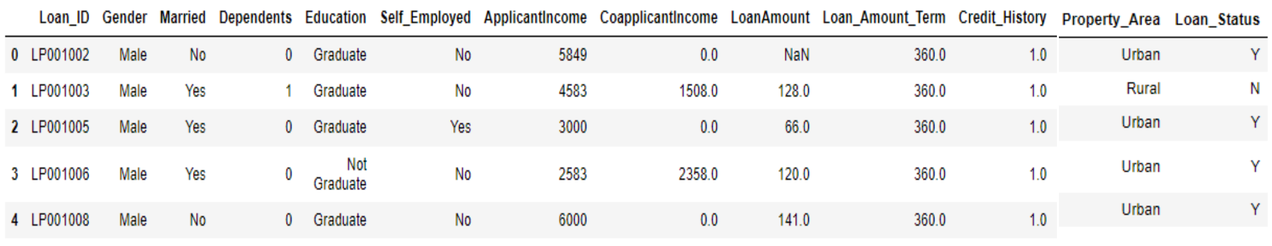
```
import pandas as pd

import numpy as np

import seaborn as sns
```

2. Load the Dataset

```
df = pd.read_csv('Bank_Data.csv')
```

3. Print the Dataframe

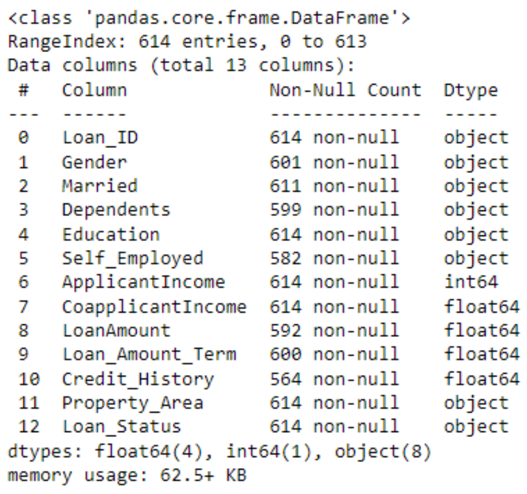
```
df.head()
```



	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

4. Check the Datatypes of all features

```
df.info()
```

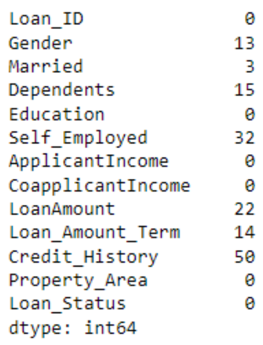


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

5. Data Cleaning

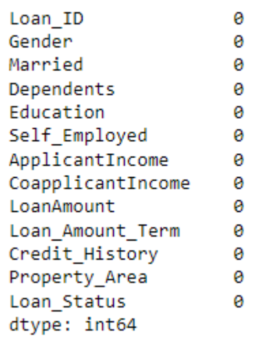
Finding out null values in each column and dropping those records as the count is less to affect the quality of dataset.

```
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
```

```
df=df.dropna()
df.isnull().sum()
```

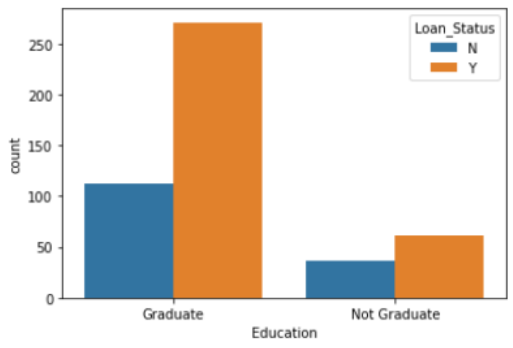


```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

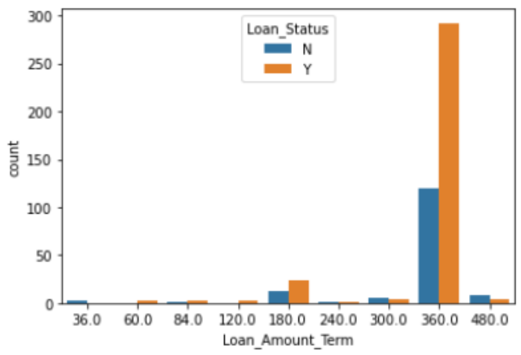
9.2. Data Visualization

By performing visualization we can get the intuition about how each feature is contributing.

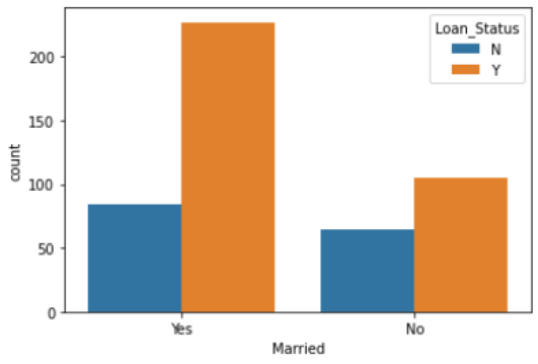
```
sns.countplot(x='Education',hue='Loan_Status',data=df)
sns.countplot(x='Loan_Amount_Term',hue='Loan_Status',data=df)
sns.countplot(x='Married',hue='Loan_Status',data=df)
sns.countplot(x='Gender',hue='Loan_Status',data=df)
sns.countplot(x='Dependents',hue='Loan_Status',data=df)
sns.countplot(x='Self_Employed',hue='Loan_Status',data=df)
sns.countplot(x='Credit_History',hue='Loan_Status',data=df)
sns.countplot(x='Property_Area',hue='Loan_Status',data=df)
sns.displot(df['ApplicantIncome’])
sns.displot(df['CoapplicantIncome'])
```



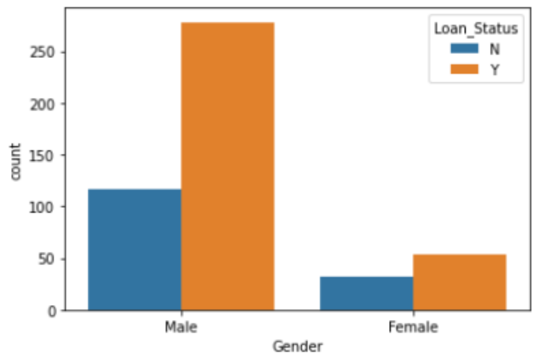
Education vs Loan Status



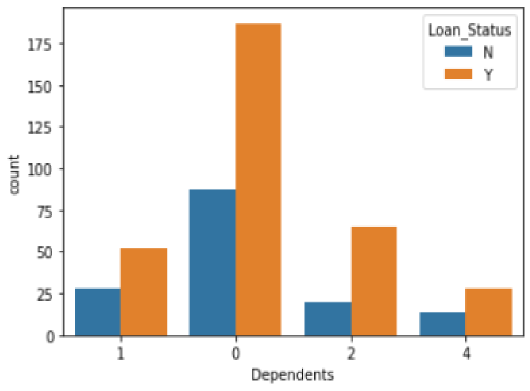
Loan_Term vs Loan Status



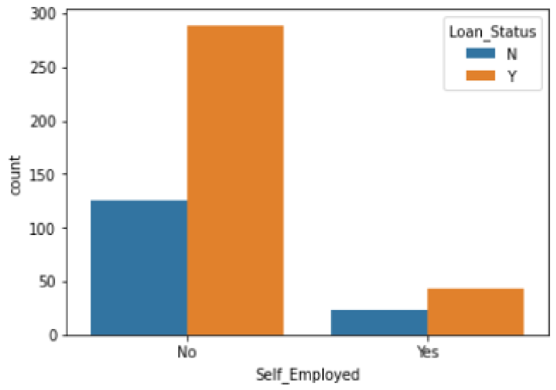
Married vs Loan Status



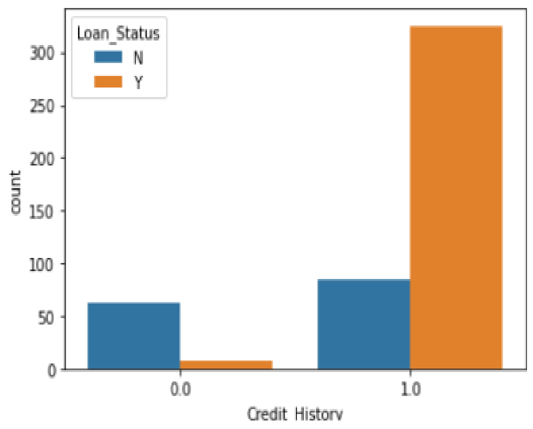
Gender vs Loan Status



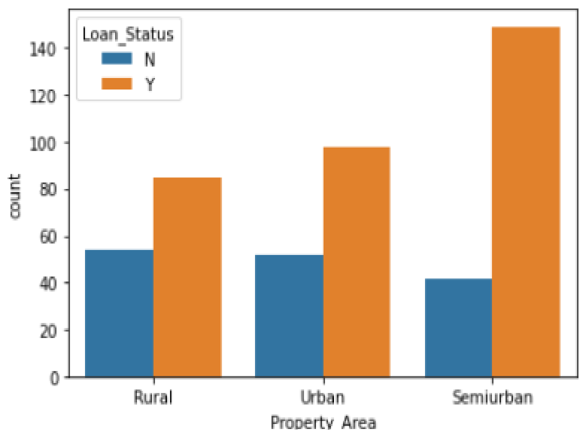
Dependents vs Loan Status



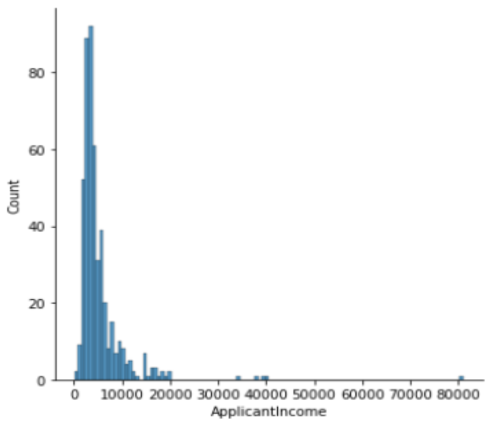
Self Employed vs Loan Status



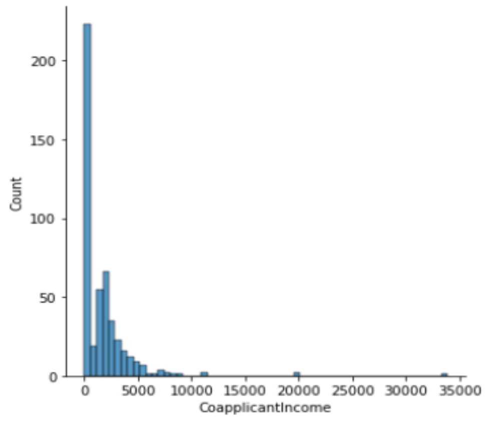
Credit History vs Loan Status



Property Area vs Loan Status



Applicant Income displot



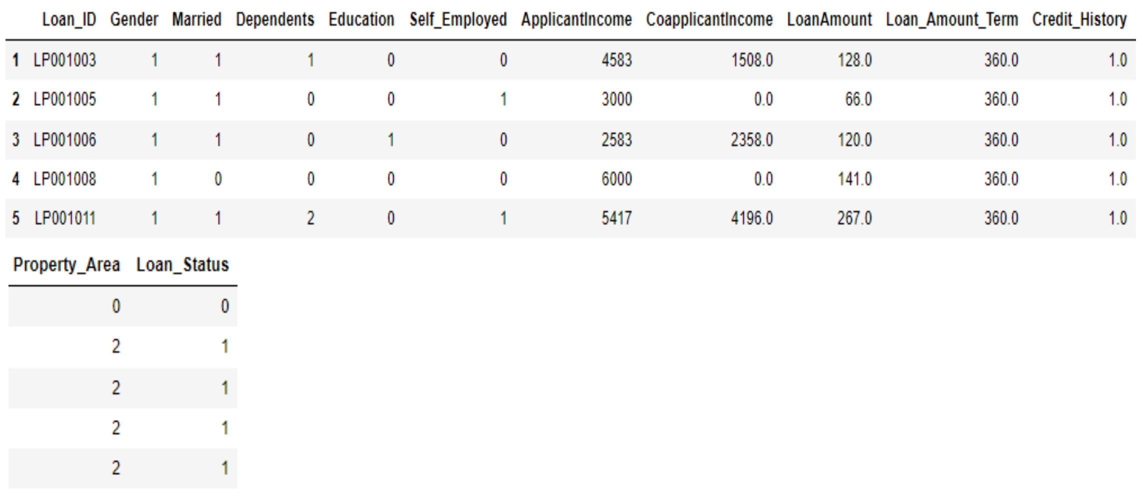
CoApplicant Income displot

From this visualization we get to know that all features play a prominent role in deciding the target label, so we don't need to drop any of the features except Loan_ID as it doesn't contribute.

9.3 Label Encoding

To make the work easy for the ML algorithms we will encode the values of feature labels that are categorical but in string format into numeric category. For ex: male as 0 and female as 1.

```
from sklearn.preprocessing import LabelEncoder
cols=['Gender','Married','Dependents','Education','Self_Employed',
          'Property_Area','Loan_Status']
le=LabelEncoder()
for col in cols:
   df[col]=le.fit_transform(df[col])
```



	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
1	LP001003	1	1	1	0	0	4583	1508.0	128.0	360.0	1.0
2	LP001005	1	1	0	0	1	3000	0.0	66.0	360.0	1.0
3	LP001006	1	1	0	1	0	2583	2358.0	120.0	360.0	1.0
4	LP001008	1	0	0	0	0	6000	0.0	141.0	360.0	1.0
5	LP001011	1	1	2	0	1	5417	4196.0	267.0	360.0	1.0

Property_Area	Loan_Status
0	0
2	1
2	1
2	1
2	1

9.4 Observing performace of Base Models

1. Splitting the data into features and target label.

```
X=df.drop(columns=['Loan_ID','Loan_Status'],axis=1)
y=df[['Loan_Status']]
```

2. Splitting the data into train-data and test-data.

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,y,test_size=0.2,stratify=y
,random_state=42)
```

3. Observe the performance of each base model before stacking.

i) Random Forest Classifier – Clearly overfitted as train accuracy is 100% and their is big difference between train and test accuracy.

```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier()
model.fit(X_train,Y_train)
yt_pred=model.predict(X_train)
accuracy_score(yt_pred,Y_train)//train accuracy
        1.0
y_pred=model.predict(X_test)
accuracy_score(y_pred,Y_test)//test accuracy
        0.8125
```

ii) Decision Tree Classifier - Clearly overfitted as train accuracy is 100% and their is big difference between train and test accuracy.

```
from sklearn.tree import DecisionTreeClassifier
model2=DecisionTreeClassifier()
model2.fit(X_train,Y_train)
yt_pred=model2.predict(X_train)
accuracy_score(yt_pred,Y_train)//train accurcay
        1.0
y_pred=model2.predict(X_test)
accuracy_score(y_pred,Y_test)//test accuracy

          0.7395833333333334
```

iii) Gaussian Naïve Bayes Classifier - Clearly not overfitted as train accuracy is 76% and test accuracy is 80% and their is no big difference between train and test accuracy but test accuracy is not that satisfactory.

```
from sklearn.naive_bayes import GaussianNB
model3=GaussianNB()
model3.fit(X_train,Y_train)
yt_pred=model3.predict(X_train)
accuracy_score(yt_pred,Y_train) //train accuracy
          0.796875

y_pred3=model3.predict(X_test)
accuracy_score(y_pred3,Y_test) //test accuracy

          0.8020833333333334
```

iv) Gradient Boosting Classifier - Clearly not overfitted as train accuracy is 90% and test accuracy is 84% and their is no big difference between train and test accuracy and test accuracy is satisfactory.

```
from sklearn.ensemble import GradientBoostingClassifier
model4=GradientBoostingClassifier()
model4.fit(X_train,Y_train)
yt_pred=model4.predict(X_train)
accuracy_score(yt_pred,Y_train)//train a
          0.90625

y_pred4=model4.predict(X_test)
accuracy_score(y_pred4,Y_test)
     0.8333333333333334
```

9.5. Perform Stacking.

i) Freshly Initialize Base Models.

```
model=RandomForestClassifier(n_estimators=5)
model2=DecisionTreeClassifier()
model3=GaussianNB()
model4=GradientBoostingClassifier()
```

ii) Split the data into Train data and Test data

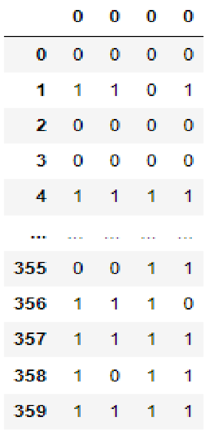
```
X_train,X_test,Y_train,Y_test=train_test_split(X,y,test_size=0.25,
stratify=y, random_state=42)
```

iii) Perform K-Folding

```
from sklearn.model_selection import StratifiedKFold
def Stacking(model,train,y,n_fold):
    folds=StratifiedKFold(n_splits=n_fold)
    train_pred=np.empty((0,1),int)
    for train_indices,val_indices in folds.split(train,y.values):
        x_train,x_val=train.iloc[train_indices],train.iloc[val_indices]
        print(x_train,x_val)#x_val=x_test
        y_train,y_val=y.iloc[train_indices],y.iloc[val_indices]
        print(y_train,y_val)#y_val=y_test
        model.fit(x_train,y_train)
        train_pred=np.append(train_pred,model.predict(x_val))
        print(train_pred)
    return train_pred

train_pred_1=Stacking(model=model,n_fold=5,train=X_train,y=Y_train)
train_pred_2=Stacking(model=model2,n_fold=5,train=X_train,y=Y_train)
train_pred_3=Stacking(model=model3,n_fold=5, train=X_train,y=Y_train)
train_pred_4=Stacking(model=model4,n_fold=5,  train=X_train,y=Y_train)
```

iv) Combine the Predictions of 4 Base Models to get the train data for meta model.

```
train_pred_1=pd.DataFrame(train_pred_1)
train_pred_2=pd.DataFrame(train_pred_2)
train_pred_3=pd.DataFrame(train_pred_3)
train_pred_4=pd.DataFrame(train_pred_4)
train_data_meta =
pd.concat([train_pred_1,train_pred_2,train_pred_3,train_pred_4], axis=1)
print(train_data-meta)
```



```
       0  0  0  0
  0    0  0  0  0
  1    1  1  0  1
  2    0  0  0  0
  3    0  0  0  0
  4    1  1  1  1
...  ... ... ... ...
355    0  0  1  1
356    1  1  1  0
357    1  1  1  1
358    1  0  1  1
359    1  1  1  1
```

v) Train the meta-model [Logistic Regression] with train_data_meta.

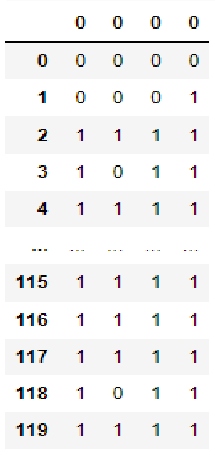
```
from sklearn.linear_model import LogisticRegression
meta_model = LogisticRegression()
meta_model.fit(train_data_meta,Y_train)
```

vi) Freshly train your base models on train_data as we used same names for models in checking performance before stacking and in K-Folding.

```
model= RandomForestClassifier()
model2= DecisionTreeClassifier()
model3= GaussianNB()
model4= GradientBoostingClassifier()

model.fit(X_train,Y_train)
model2.fit(X_train,Y_train)
model3.fit(X_train,Y_train)
model4.fit(X_train,Y_train)
```

vii) Predict the Test_data with trained based model and combine them to prepare the test_data for our meta model.

```
y1_pred= model.predict(X_test)
y2_pred= model2.predict(X_test)
y3_pred= model3.predict(X_test)
y4_pred= model4.predict(X_test)
y1_pred= pd.DataFrame(y1_pred)
y2_pred= pd.DataFrame(y2_pred)
y3_pred= pd.DataFrame(y3_pred)
y4_pred= pd.DataFrame(y4_pred)
test_res = pd.concat([y1_pred, y2_pred, y3_pred,y4_pred], axis=1)
print(test_res)
```



```
       0  0  0  0
  0    0  0  0  0
  1    0  0  0  1
  2    1  1  1  1
  3    1  0  1  1
  4    1  1  1  1
 ...  ... ... ... ...
115    1  1  1  1
116    1  1  1  1
117    1  1  1  1
118    1  0  1  1
119    1  1  1  1
```

viii) Test the meta-model (LR) with the test_res to get final predictions.

```
meta_model.score(test_res,Y_test)
```

```
0.8416666666666667
```

Now that we have trained and tested our base models as well as meta model a.k.a Stacked Model we can observe that Stacked model gave the better accuracy as compared to each base model.

9.6 Dump the base models and meta-model.

```
import pickle
pickle.dump(meta_model,open('StackModel.pkl','wb'))
pickle.dump(model,open('model.pkl','wb'))
pickle.dump(model2,open('model2.pkl','wb'))
pickle.dump(model3,open('model3.pkl','wb'))
pickle.dump(model4,open('model4.pkl','wb'))
```

9.7 Deploy the models using Flask-API.

i) Open Spyder IDE and create a project say ‘BankLoan’.

ii) Copy the dumped models from previous directory to this BankLoan Folder.

iii) Create two sub-folders inside BankLoan names ‘templates’ and ‘static’.

iv) The ‘static’ folder will contain the ‘index.html’ file, and ‘template’ will contain ‘CSS’ file for ‘index.html’.

v) Create a python file named ‘app.py’ inside the BankLoan Folder.

app.py

```
from flask import Flask, request, render_template
import pickle

app = Flask(__name__)

pickle_in=open('StackModel.pkl','rb')
p_model=open('model.pkl','rb')
p_model2=open('model2.pkl','rb')
p_model3=open('model3.pkl','rb')
p_model4=open('model4.pkl','rb')
stack_model=pickle.load(pickle_in)
model=pickle.load(p_model)
model2=pickle.load(p_model2)
model3=pickle.load(p_model3)
model4=pickle.load(p_model4)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    features = [ x for x in request.form.values()]
    Gender=features[5]
    Married=features[6]
    Dependents=features[7]
    Education=features[8]
    SelfEmployed=features[9]
    PropertyArea=features[10]
    CreditHistory=features[4]
    ApplicantIncome=float(features[0])/10.0
    CoapplicantIncome=float(features[1])/10.0
    LoanAmount=float(features[3])/1000.0
    Loan_Amount_Term=int(features[2])*30



p1=int(model.predict([[Gender,Married,Dependents,Education,SelfEmployed,Applicant
Income,CoapplicantIncome,LoanAmount,Loan_Amount_Term,CreditHistory,PropertyA
rea]]))

p2=int(model2.predict([[Gender,Married,Dependents,Education,SelfEmployed,Applican
tIncome,CoapplicantIncome,LoanAmount,Loan_Amount_Term,CreditHistory,Property
Area]]))
```

```
p3=int(model3.predict([[Gender,Married,Dependents,Education,SelfEmployed,Applica
ntIncome,CoapplicantIncome,LoanAmount,Loan_Amount_Term,CreditHistory,Propert
yArea]]))

p4=int(model4.predict([[Gender,Married,Dependents,Education,SelfEmployed,Applica
ntIncome,CoapplicantIncome,LoanAmount,Loan_Amount_Term,CreditHistory,Propert
yArea]]))

    prediction=stack_model.predict([[p1,p2,p3,p4]])



    return render_template('index.html',
prediction_text=prediction,ai=features[0],ci=features[1],la=features[3],lt=featu
        res[2],ch=int(features[4]),gd=int(features[5]),mr=int(features[6]),dp=int(featur
        es[7]),ed=int(features[8]),se=int(features[9]),pa=int(features[10]))

if __name__ == "__main__":
    app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
<html >
<head>
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Khand:wght@600&family=Monda&di
splay=swap" rel="stylesheet">
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Khand:wght@600&display=swap"
rel="stylesheet">
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
  <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
   <title>Bank Loan Approval</title>
</head>
```

```
<body>
 <div class="login">
            <h1>&emsp;&emsp;&ensp;Predict Loan Approval</h1><br/><br/>



   <div class="sam">
    <form action="{{ url_for('predict')}}" method="post" class="fom">
     <table >
     <tr>
     <td></td>
     </tr>
     <tr>
     <td></td>
     </tr>
     <tr>
            <td><label>Applicant_Income</label></td>
            <td class="colon"><label>:</label>&nbsp;&nbsp;</td>
            <td><input id="ai" type="text"  value="{{ai}}" name="Applicant_Income"
                     required="required" /></td>
     </tr>
      <tr>
            <td><label>CoApplicant_Income</label></td>
             <td class="colon"><label>:</label></td>
            <td><input id="ci" type="text" value="{{ci}}" name="CoApplicant_Income"
                     required="required" /></td>
     </tr>
        <tr>
            <td><label>Loan_Amount_Term</label></td>
            <td class="colon"><label>:</label></td>
            <td><input id="Term" type="text" value="{{lt}}" name="Term"
                     required="required" /></td>
        </tr>
        <tr>
            <td><label>Loan_Amount</label></td>
            <td class="colon"><label>:</label></td>
            <td><input id="la" type="text" value="{{la}}" name="Loan_Amount"
                     required="required" /></td>
        </tr>
        <tr>
            <td><label>Credit_History</label></td>
            <td class="colon"><label>:</label></td>
            <td><select id="Credit" name="Credit"  required="required" >
            {% if ch == 1 %}
                  <option value="1" selected >Have Credit History</option>
                 <option value="0">Doesn't have Credit History</option>
             {% else %}
                 <option value="1">Have Credit History</option>
                <option value="0" selected>Doesn't have Credit History</option>
             {% endif %}
            </select>
        </td>
     </tr>
     <tr>
         <td><label>Gender</label></td>
         <td class="colon"><label>:</label></td>
         <td><select id="Gender" name="Gender"  required="required" >
         {% if gd == 1 %}
        <option value="1" selected>Male</option>
        <option value="0">Female</option>
        {% else %}
        <option value="1">Male</option>
        <option value="0" selected>Female</option>
        {% endif %}
    </select></td>
     </tr>
      <tr>
         <td><label>Marital Status</label></td>
         <td class="colon"><label>:</label></td>
         <td><select id="Marital" name="Marital"  required="required" >
         {% if mr == 1 %}
                <option value="1" selected>Married</option>
                 <option value="0">Not Married</option>
         {% else %}
                 <option value="1">Married</option>
                 <option value="0" selected>Not Married</option>
         {% endif %}

    </select></td>
     </tr>
     <tr>
         <td><label>Dependents</label></td>
         <td class="colon"><label>:</label></td>
         <td><select id="Dept" name="Dept"  required="required" >
         {% if dp == 0 %}
                   <option value="0" selected>0</option>
                   <option value="1">1</option>
                   <option value="2">2</option>
                   <option value="4">3+</option>
          {% elif dp == 1 %}
                   <option value="0" >0</option>
                   <option value="1" selected>1</option>
                   <option value="2">2</option>
                   <option value="4">3+</option>
          {% elif dp == 2 %}
                   <option value="0" >0</option>
                   <option value="1">1</option>
                   <option value="2" selected>2</option>
                   <option value="4">3+</option>
           {% else %}
```

```
  </select></td>
</tr>
<tr>
   <td><label>Education</label></td>
   <td class="colon"><label>:</label></td>
   <td><select id="Edu" name="Edu"  required="required" >
          {% if ed == 0 %}
             <option value="0" selected>Graduate</option>
              <option value="1">Under Graduate</option>
          {% else %}
              <option value="0">Graduate</option>
              <option value="1" selected>Under Graduate</option>
           {% endif %}

            </select></td>
</tr>
<tr>
   <td><label>Employment</label></td>
   <td class="colon"><label>:</label></td>
   <td><select id="Emp" name="Emp"  required="required" >
            {% if se == 1 %}
                <option value="1" selected>Self Employed</option>
                <option value="0">Not Self Employment</option>
            {% else %}
                <option value="1">Self Employed</option>
               <option value="0" selected>Not Self Employment</option>
            {% endif %}
            </select></td>
  </tr>
   <tr>
   <td> <label>Property_Area</label></td>
   <td class="colon"><label>:</label></td>
   <td><select id="PA" name="PA"  required="required" >
            {% if pa == 0 %}
                 <option value="0" selected>Rural</option>
                 <option value="2">Urban</option>
                 <option value="1">Semi-Urban</option>
            {% elif pa == 2 %}
                 <option value="0">Rural</option>
                 <option value="2" selected>Urban</option>
                  <option value="1">Semi-Urban</option>
            {% else %}
                  <option value="0">Rural</option>
                  <option value="2">Urban</option>
                  <option value="1" selected>Semi-Urban</option>
            {% endif %}
```

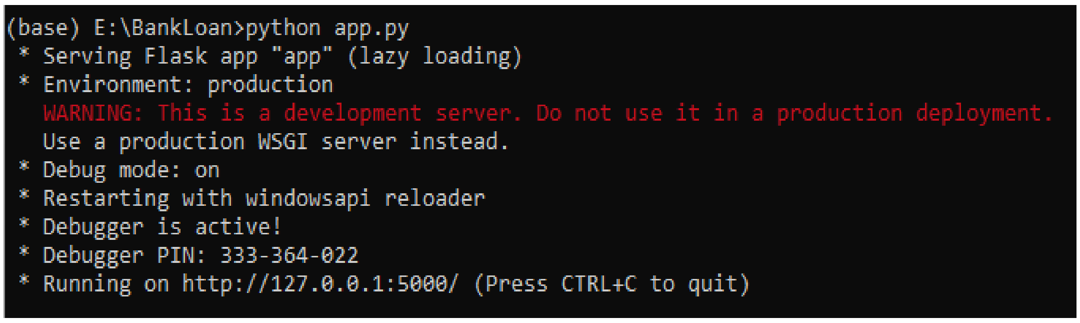
```
                        </select></td>
            </tr>
             <tr>
                        <td><button  type="submit" class="btn btn-primary btn-block btn-
                                    large" >Predict</button></td>
                        <td class="colon"><label></label></td>
                        <td><button  onclick="Reset()" class="btn btn-primary btn-block btn-
                                    large">Reset</button></td>
             </tr>
              <tr>
             {% if prediction_text == 1 %}
             <td style="text-align:center" colspan="3"><p class="tr">You are eligible for
                                                                                        Loan</p>
             </td>
             {% endif %}
             {% if prediction_text == 0 %}
             <td style="text-align:center" colspan="3"><p class="fr">You are not eligible
                                                                                        for Loan</p>
             </td>
             {% endif %}
              </tr>
        </table>
    </form>
   </div> <br/><br/>
    </div> <br/><br/>
  </body>
</html>
```

styles.css

```
html { width: 100%; height:100%;  }
label{
font-family:Monda;
}
body {
            width: 100%;
            height:100%;
            font-family: 'Open Sans', sans-serif;
            background-attachment:fixed;
            background: linear-gradient(to bottom, #cc0066 0%, #0000cc 100%);
            background-repeat:no-repeat;
            background-size:cover;
            background-attachment:fixed;
            color: #fff;
            font-size: 15px;
            text-align:center;
            letter-spacing:1.2px;
}
```

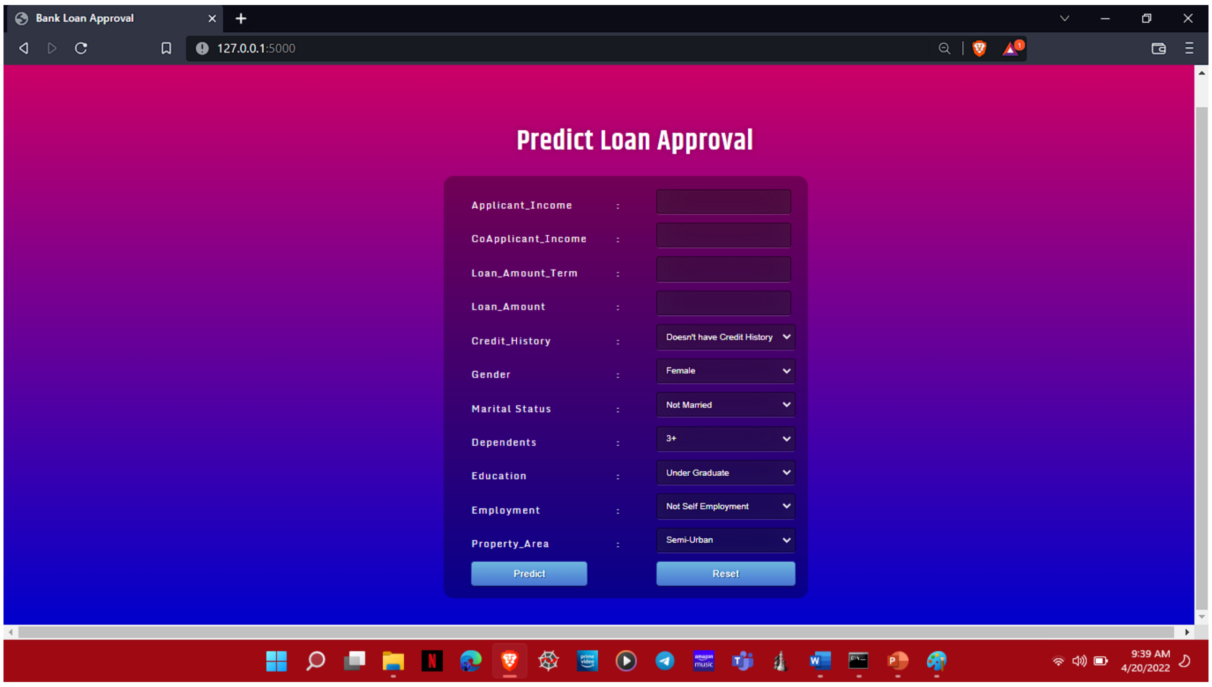
```
.fom{
   padding-top:10px;
   padding-bottom:15px;
}
.sam{
background:rgba(0,0,0,0.3);
width:540px;
border-radius:15px;
}
td {
  text-align: left;
  padding-left:40px;
}
.login {
         position: relative;
         top: 30%;
         left: 45%;
         margin: -150px 0 0 -150px;
         width:460px;
         height:500px;
         padding-top:20px;
}
.tr{
   color:rgb(78,212,78);
   font-size: 1.5em;
   font-weight: bold;
}
.fr{
   color:rgb(247,67,76);
   font-size: 1.5em;
   font-weight: bold;
}
.login h1 { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-spacing:1px;
text-align:center;display:inline;font-family:Khand;font-size:42px; }

input {
         width: 200px;
         margin-bottom: 10px;
         background: rgba(0,0,0,0.3);
         border: none;
         outline: none;
         padding: 10px;
         font-size: 13px;
         color: #fff;
         text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
         border: 1px solid rgba(0,0,0,0.3);
         border-radius: 4px;
         box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px
rgba(255,255,255,0.2);
         -webkit-transition: box-shadow .5s ease;
         -moz-transition: box-shadow .5s ease;
         -o-transition: box-shadow .5s ease;
-ms-transition: box-shadow .5s ease;
            transition: box-shadow .5s ease;
}select{
width: 100%;
            margin-bottom: 10px;
            background: rgba(0,0,0,0.3);
            border: none;
            outline: none;
            padding: 10px;
            font-size: 13px;
            color: #fff;
            text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
            border: 1px solid rgba(0,0,0,0.3);
            border-radius: 4px;
            box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px
rgba(255,255,255,0.2);
            -webkit-transition: box-shadow .5s ease;
            -moz-transition: box-shadow .5s ease;
            -o-transition: box-shadow .5s ease;
            -ms-transition: box-shadow .5s ease;
            transition: box-shadow .5s ease;
}
input:focus { box-shadow: inset 0 -5px 45px rgba(100,100,100,0.4), 0 1px 1px
                        rgba(255,255,255,0.2); }
select:focus{ box-shadow: inset 0 -5px 45px rgba(100,100,100,0.4), 0 1px 1px
                        rgba(255,255,255,0.2);}
```

vi) After writing the code we need to execute the project by opening the anaconda prompt and enter command "python app.py". The web application can be accessed at port http://127.0.0.1:5000/ by entering port address in an standard browser**.**



```
(base) E:\BankLoan>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
 * Debugger is active!
 * Debugger PIN: 333-364-022
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

9.8 Output

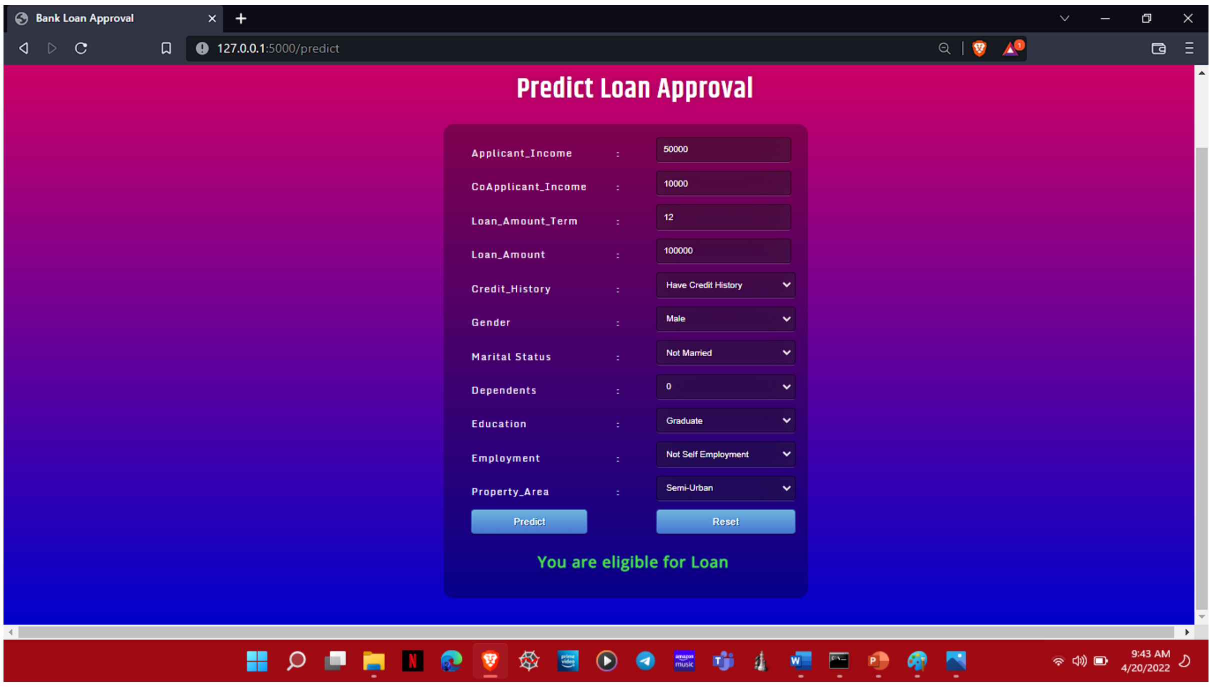

Bank Loan Approval

127.0.0.1:5000

Predict Loan Approval

Applicant_Income	:	
CoApplicant_Income	:	
Loan_Amount_Term	:	
Loan_Amount	:	
Credit_History	:	Doesn't have Credit History
Gender	:	Female
Marital Status	:	Not Married
Dependents	:	3+
Education	:	Under Graduate
Employment	:	Not Self Employment
Property_Area	:	Semi-Urban

Fig 9.1 Interface for Prediction


Bank Loan Approval

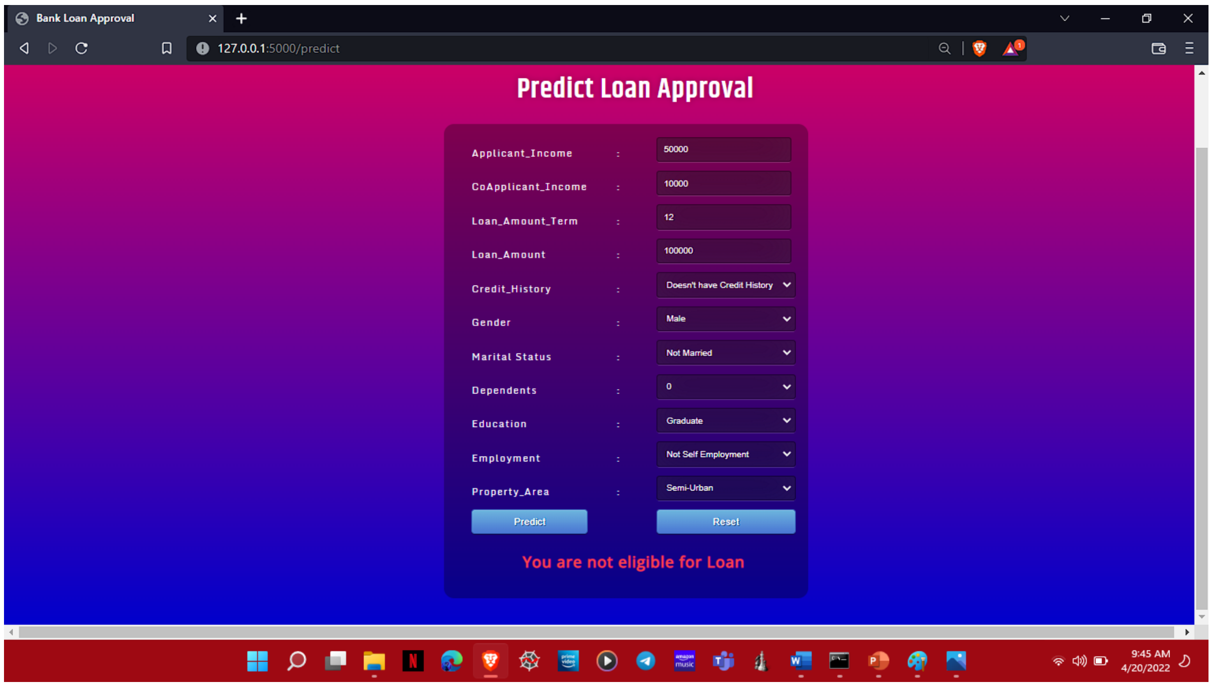
127.0.0.1:5000/predict

Predict Loan Approval

Applicant_Income	:	50000
CoApplicant_Income	:	10000
Loan_Amount_Term	:	12
Loan_Amount	:	100000
Credit_History	:	Have Credit History
Gender	:	Male
Marital Status	:	Not Married
Dependents	:	0
Education	:	Graduate
Employment	:	Not Self Employment
Property_Area	:	Semi-Urban

You are eligible for Loan

Fig 9.2 Prediction for Eligible Borrower



Predict Loan Approval

Applicant_Income	:	50000
CoApplicant_Income	:	10000
Loan_Amount_Term	:	12
Loan_Amount	:	100000
Credit_History	:	Doesn't have Credit History
Gender	:	Male
Marital Status	:	Not Married
Dependents	:	0
Education	:	Graduate
Employment	:	Not Self Employment
Property_Area	:	Semi-Urban

You are not eligible for Loan

Fig 9.3 Prediction for Ineligible Borrower

10. Conclusion

In this project we have tried to combine the knowledge of multiple machine learning model through stacking, where we obtained a meta model that can perform predictions on the predictions of base models. So, we have removed the requirement of using a single machine learning algorithms that gives best accuracy which may in turn be overfitted or underfitted as we have seen with our base models.
Hence adding a extra layer that can give us prediction using the knowledge of the meta-model that is gained from base models, we can do predictions with a model that are not underfitted or overfitted.

As a part of future work more techniques like stacking that can combine the knowledge of various machine learning models at more deep and core logical levels will lead to a very efficient and robust machine learning models.

11. Bibliography

[1] Ethem Alpaydin, “Introduction to Machine Learning” (Adaptive Computation and Machine Learning Series)”, Third Edition, MIT Press, 2014.

[2] Stephen Marsland, “Machine Learning – An Algorithmic Perspective”, Second Edition, Chapmanand Hall/CRC Machine Learning and Pattern Recognition Series, 2014.

[3] Leo Breiman, Jerome H. Friedman, Richard A.Olshen and Charles J. Stone, “Classification”.

[4] Tom M Mitchell, “Machine Learning”, First Edition, McGraw Hill Education, 2013.

[5] Min-Chun Yang, Chiun-Sheng Huang, “Whole breast lesion detection using naive Bayes classifier for portable ultrasound”, Elsevier , computer science section, vol. 38, No.11, pp.1870-1880, 2012.

[6] “Regression Trees”. Wadsworth & Brooks, 1984.

[7] J. Ross Quinlan, “C4.5: Programs for Machine Learning . Morgan Kaufmann”, 1993.

[8] S.B. Kotsiantis, “Supervised Machine Learning: A Review of Classification Techniques”, Informatica 31(2007).

[9] Jason Bell, “Machine learning – Hands on for Developers and Technical Professionals”, First Edition , Wiley, 2014.

[10] Stacking and Blending–An Intuitive Explanation by Steven Yu [Online]. https://medium.com/@stevenyu530_73989/stacking-and-blending-intuitive-explanation-of-advanced-ensemble-methods-46b295da413c