



## FLIP ROBO

*Project Name: -*

## MICRO-CREDIT DEFaulter MODEL



*Submitted By: -*

DIPTIRANJAN PRADHAN

## **ACKNOWLEDGMENT:-**

I would like to express my special gratitude to the “Flip Robo” team, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analysing skills. Also, I want to express my huge gratitude to Mrs . Khushboo Garg (SME FlipRobo), who has helped me overcome difficulties within this project and others. and also encouraged me a lot with his valuable words and with his unconditional support I have ended up with a beautiful Project.

A huge thanks to my academic team “Data trained” who are the reason behind what I am today. Last but not least my parents who have been my backbone in every step of my life.

And also thank you for many other persons who has helped me directly or indirectly to complete the project

## **INTRODUCTION:-**

- **Business Problem Framing:-**

A Microfinance Institution (MFI) is an organization that offers financial services to low income populations. MFS becomes very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans and so on.

Many microfinance institutions (MFI), experts and donors are supporting the idea of using mobile financial services (MFS) which they feel are more convenient and efficient, and cost saving, than the traditional high-touch model used since long for the purpose of delivering microfinance services. Though, the MFI industry is primarily focusing on low income families and is very useful in such areas, the implementation of MFS has been uneven with both significant challenges and successes.

Today, microfinance is widely accepted as a poverty-reduction tool, representing \$70 billion in outstanding loans and a global outreach of 200 million clients.

We are working with one such client that is in Telecom Industry. They are a fixed wireless telecommunications network provider. They have launched various products and have developed its business and organization based on the budget operator model, offering better products at Lower Prices to all value conscious customers through a strategy of disruptive innovation that focuses on the subscriber.

They understand the importance of communication and how it affects a person's life, thus, focusing on providing their services and products to low income families and poor customers that can help them in the need of hour.

They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be 6 (in Indonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah).

The sample data is provided to us from our client database. It is hereby given to you for this exercise. In order to improve the selection of customers for the credit, the client wants some

predictions that could help them in further investment and improvement in selection of customers.

Describe the business problem and how this problem can be related to the real world.

- **Motivation for the Problem Undertaken:-**

Build a model which can be used to predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan. In this case, Label '1' indicates that the loan has been paid i.e. Non- defaulter, while, Label '0' indicates that the loan has not been paid i.e. defaulter.

## **Analytical Problem Framing:-**

- **Mathematical/ Analytical Modeling of the Problem:-**

Machine Learning is defined by Tom Mitchell in his book as "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". Supervised learning is when the output is known for the corresponding inputs, and is also provided for the machine to learn.

- **Data Sources and their formats:-**

The data is provided to us from our client database. It is hereby given to us for the exercise to improve the selection of

customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers. It is given in the csv file format.

### importing Dataset:-

```
In [2]: df=pd.read_csv('Data file.csv')
df
```

Out[2]:

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	r
0	1	0	21408170789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0	
1	2	1	76462170374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0	
2	3	1	17943170372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0	
3	4	1	55773170781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0	
4	5	1	03813182730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	...	6.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
209588	209589	1	22758185348	404.0	151.872333	151.872333	1089.19	1089.19	1.0	0.0	...	6.0	
209589	209590	1	95583184455	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	0.0	...	6.0	
209590	209591	1	28556185350	1013.0	11843.111667	11904.350000	5861.83	8893.20	3.0	0.0	...	12.0	
209591	209592	1	59712182733	1732.0	12488.228333	12574.370000	411.83	984.58	2.0	38.0	...	12.0	
209592	209593	1	65061185339	1581.0	4489.362000	4534.820000	483.92	631.20	13.0	0.0	...	12.0	

209593 rows x 37 columns

1. Label = Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}
2. Msisdn = mobile number of user
3. Aon = age on cellular network in days
4. daily\_decr30 = Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)
5. daily\_decr90 = Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)
6. rental30 = Average main account balance over last 30 days
7. rental90 = Average main account balance over last 90 days
8. last\_rech\_date\_ma = Number of days till last recharge of main account
9. last\_rech\_date\_da = Number of days till last recharge of data account
10. last\_rech\_amt\_ma = Amount of last recharge of main account (in Indonesian Rupiah)
11. cnt\_ma\_rech30 = Number of times main account got recharged in last 30 days
12. fr\_ma\_rech30 = Frequency of main account recharged in last 30 days
13. sumamnt\_ma\_rech30 = Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)

14. medianamnt\_ma\_rech30 = Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)
15. medianmarechprebal30 = Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)
16. cnt\_ma\_rech90 = Number of times main account got recharged in last 90 days
17. fr\_ma\_rech90 = Frequency of main account recharged in last 90 days
18. sumamnt\_ma\_rech90 = Total amount of recharge in main account over last 90 days (in Indonesian Rupiah)
19. medianamnt\_ma\_rech90 = Median of amount of recharges done in main account over last 90 days at user level (in Indonesian Rupiah)
20. medianmarechprebal90 = Median of main account balance just before recharge in last 90 days at user level (in Indonesian Rupiah)
21. cnt\_da\_rech30 = Number of times data account got recharged in last 30 days
22. fr\_da\_rech30 = Frequency of data account recharged in last 30 days
23. cnt\_da\_rech90 = Number of times data account got recharged in last 90 days
24. fr\_da\_rech90 = Frequency of data account recharged in last 90 days
25. cnt\_loans30 = Number of loans taken by user in last 30 days
26. amnt\_loans30 = Total amount of loans taken by user in last 30 days
27. maxamnt\_loans30 = maximum amount of loan taken by the user in last 30 days
28. medianamnt\_loans30 = Median of amounts of loan taken by the user in last 30 days
29. cnt\_loans90 = Number of loans taken by user in last 90 days
30. amnt\_loans90 = Total amount of loans taken by user in last 90 days

31.        maxamnt\_loans90 = maximum amount of loan taken by the user in last 90 days
32.        medianamnt\_loans90 = Median of amounts of loan taken by the user in last 90 days
33.        payback30 = Average payback time in days over last 30 days
34.        payback90 = Average payback time in days over last 90 days
35.        pcircle = telecom circle
36.        pdate = date

- Data Preprocessing Done:-

The dataset that will be used to train the model has some challenges. In particular it has been subjected to the imbalanced dataset problem, whereby not all classes have similar number of instances .

After removing classes with too few instance, to handle the remaining classes there were a few possible solutions and these are discussed in more detail below. One way to deal with imbalanced dataset is by downsampling which involves reducing the number of instances by tossing away some instances of classes with high number of instances to match the classes with lower number of instances. This has the advantage that it is simple to implement and has proven to be more effective than upsampling. However this method has the major disadvantage that useful and valuable instances are thrown away upon training. Another common solution to deal with imbalanced dataset is by upsampling which involves generating synthetic data or making multiple copies of the instances for classes with low number of instances (minority class)

to match the majority class . This has the advantage that it does not throw away useful instances. However this method has proved in the past to cause overfitting and can be computationally intensive if the number of instances and classes are very large. A common solution that changes how the model learns instead of the dataset itself is known as Cost Sensitive Learning. It works by changing the loss/error function to give a different error value to False Positive and False Negatives depending on which is more significant, and hence model will avoid the error with a higher error value. The advantage of this is that it changes the model instead of the data. However it can be ineffective on severely imbalanced datasets and does not guarantee convergence to optimal model. Boosting like Cost Sensitive Learning is another technique that changes the model instead of the data. It trains multiple classifiers that improve on the errors of the previous classifiers. Again this technique has the advantage that it does not change the dataset itself. However it is prone to overfit the dataset Having considered each of the possible solution, a final decision was reached to not consider classes that have too few instances for the machine learning algorithm to learn anything useful.

Tree based models was decided as the solution to be applied on the dataset, this was chosen so that valuable data are not to change by sampling. The dataset now consisted of only

- Data Inputs- Logic- Output Relationships:-



Correlation Matrix:

	label	aon	daily_dec30	rental30	last_rech_date_ma	last_rech_date_ds	last_rech_amt_ma
label	1.000000	0.000403	-0.002048	0.002739	0.003077	-0.003906	-0.003459
aon	0.000403	1.000000	-0.003785	0.168298	0.196150	0.058085	0.075521
daily_dec30	-0.002048	-0.003785	1.000000	0.001104	0.000374	-0.000960	-0.000790
rental30	0.002739	0.168298	0.001104	1.000000	0.977704	0.442066	0.458977
last_rech_date_ma	0.003077	0.196150	0.000374	0.977704	1.000000	0.434685	0.471730
last_rech_date_ds	-0.003906	0.058085	-0.000960	0.442066	0.434685	1.000000	0.955237
last_rech_amt_ma	-0.003459	0.075521	-0.000790	0.458977	0.471730	0.955237	1.000000

Correlation Matrix (s x 34 columns):

	namnt_ma_rech30	anmarechprebal30	cnt_ma_rech30	fr_ma_rech30	namnt_ma_rech90	namnt_ma_rech90	anmarechprebal90	cnt_da_rech30	fr_da_rech30	cnt_da_rech90	fr_da_rech90	cnt_loans30	amnt_loans30	maxamnt_loans30	dianamnt_loans30	cnt_loans90	amnt_loans90	maxamnt_loans90	dianamnt_loans90	payback30	payback90
namnt_ma_rech30	1.000000	0.000403	-0.002048	0.002739	0.003077	-0.003906	-0.003459	-0.001133	0.001711	-0.001693	-0.001636	-0.001886	0.003261	0.002794	0.001790	1.000000	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
anmarechprebal30	0.000403	1.000000	-0.003785	0.168298	0.196150	0.058085	0.075521	0.000374	0.000908	-0.001191	-0.001191	-0.001191	0.000471	0.000471	0.000471	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
cnt_ma_rech30	-0.002048	-0.003785	1.000000	0.001104	0.000374	-0.000960	-0.000790	0.001104	0.000374	-0.000960	-0.000790	0.001104	0.000374	-0.000960	-0.000790	0.001104	0.000374	-0.000960	-0.000790	0.001104	0.000374
fr_ma_rech30	0.002739	0.168298	0.001104	1.000000	0.977704	0.442066	0.458977	0.000374	0.000908	-0.001191	-0.001191	-0.001191	0.000471	0.000471	0.000471	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
namnt_ma_rech90	0.003077	0.196150	0.000374	0.977704	1.000000	0.434685	0.471730	0.000374	0.000908	-0.001191	-0.001191	-0.001191	0.000471	0.000471	0.000471	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
namnt_ma_rech90	-0.003906	0.058085	-0.000960	0.442066	0.434685	1.000000	0.955237	-0.001191	-0.001191	-0.001191	-0.001191	0.000471	0.000471	0.000471	0.000471	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
anmarechprebal90	-0.003459	0.075521	-0.000790	0.458977	0.471730	0.955237	1.000000	-0.001191	-0.001191	-0.001191	-0.001191	0.000471	0.000471	0.000471	0.000471	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
cnt_da_rech30	-0.001133	0.001711	-0.001693	-0.001636	-0.001886	0.003261	0.002794	0.001790	1.000000	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
fr_da_rech30	0.001711	-0.001693	-0.001636	-0.001886	0.003261	0.002794	0.001790	1.000000	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
cnt_da_rech90	-0.001693	-0.001636	-0.001886	0.003261	0.002794	0.001790	1.000000	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
fr_da_rech90	-0.001636	-0.001886	0.003261	0.002794	0.001790	1.000000	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
cnt_loans30	0.003261	0.002794	0.001790	1.000000	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
amnt_loans30	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
maxamnt_loans30	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
dianamnt_loans30	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
cnt_loans90	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
amnt_loans90	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
maxamnt_loans90	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
dianamnt_loans90	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
payback30	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149
payback90	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149	-0.000149

- State the set of assumptions (if any) related to the problem under consideration:-

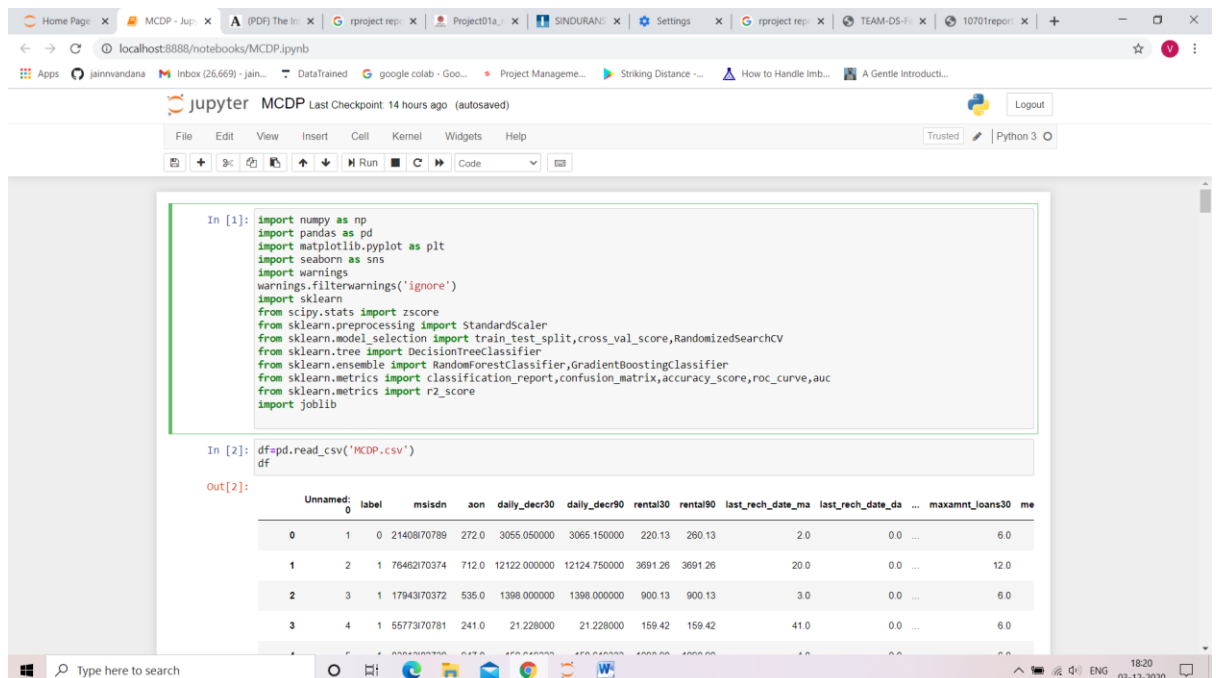
Here, you can describe any presumptions taken by you.

- Hardware and Software Requirements and Tools Used:-

**Hardware :** Since the computational aspect of the project is of importance to PANDA, it is important to know the hardware that was used in the evaluation process. The training and evaluation of the neural network model has been done on a Windows 10 computer using a quad-core CPU at i3.

**Software :** anaconda 3 , windows 10 ,Microsoft office.

**Tools used :** python , machine learning libraries.



The screenshot shows a Jupyter Notebook running in a web browser. The notebook has two cells. The first cell contains a list of imports for various machine learning and data manipulation libraries. The second cell contains code to read a CSV file named 'MCDP.csv' and display the first few rows of the resulting DataFrame.

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import sklearn
from scipy.stats import zscore
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score, RandomizedSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_curve, auc
from sklearn.metrics import r2_score
import joblib
```

```
In [2]: df = pd.read_csv('MCDP.csv')
df
```

Out[2]:

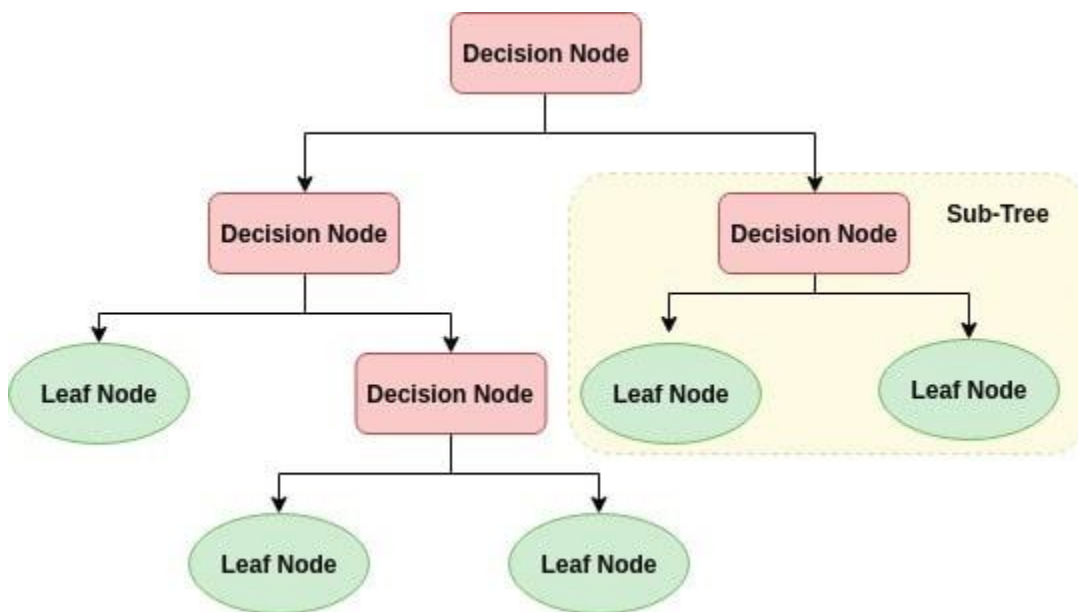
Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_data_ma	last_rech_data_da	...	maxamnt_loans30	me
0	1	0	2140870789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0
1	2	1	76462170374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0
2	3	1	17943170372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0
3	4	1	55773170781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0

## Model/s Development and Evaluation :-

- Identification of possible problem-solving approaches (methods)
-

## 1. Decision Tree Based Method:-

The decision tree classifiers organized a series of test questions and conditions in a tree structure. In the decision tree, the root and internal nodes contain attribute test conditions to separate records that have different characteristics. All the terminal node is assigned a class label Yes or No. The decision tree induction algorithm works by recursively selecting the best attribute to split the data and expanding the leaf nodes of the tree until the stopping criterion is met. The choice of best split test condition is determined by comparing the impurity of child nodes and also depends on which impurity measurement is used. After building the decision tree, a tree-pruning step can be performed to reduce the size of decision tree. Decision trees that are too large are susceptible to a phenomenon known as overfitting. Pruning helps by trimming the branches of the initial tree in a way that improves the generalization capability of the decision tree.

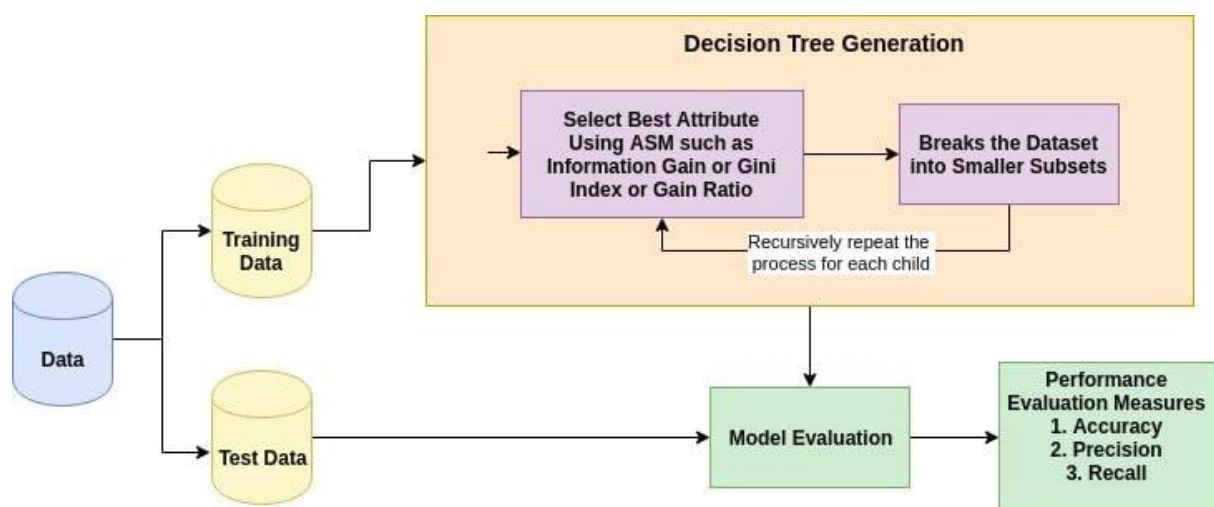


## How does the Decision Tree algorithm work?

The basic idea behind any decision tree algorithm is as follows:

1. Select the best attribute using Attribute Selection Measures(ASM) to split the records.

2. Make that attribute a decision node and breaks the dataset into smaller subsets.
3. Starts tree building by repeating this process recursively for each child until one of the condition will match:
  - All the tuples belong to the same attribute value.
  - There are no more remaining attributes.
  - There are no more instances.



## Pros:-

- Decision trees are easy to interpret and visualize.
- It can easily capture Non-linear patterns.
- It requires fewer data preprocessing from the user, for example, there is no need to normalize columns.
- It can be used for feature engineering such as predicting missing values, suitable for variable selection.
- The decision tree has no assumptions about distribution because of the non-parametric nature of the algorithm. ([Source](#))

## Cons:-

- Sensitive to noisy data. It can overfit noisy data.
- The small variation(or variance) in data can result in the different decision tree. This can be reduced by bagging and boosting algorithms.

## 2. Gradient Boosting Based Method:-

The idea of boosting came out of the idea of whether a weak learner can be modified to become better. Gradient boosting involves three elements:

A loss function to be optimized.

A weak learner to make predictions.

An additive model to add weak learners to minimize the loss function.

- Loss Function

The loss function used depends on the type of problem being solved. It must be differentiable, but many standard loss functions are supported and you can define your own. For example, regression may use a squared error and classification may use logarithmic loss. A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used.

- Weak Learner

Decision trees are used as the weak learner in gradient boosting. Specifically regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and “correct” the residuals in the predictions. Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize the loss. Initially, such as in the case of AdaBoost, very short decision trees were used that only had a single split, called a decision stump. Larger trees can be used generally with 4-to-8 levels. It is common to constrain the weak learners in specific ways, such as a maximum number of layers, nodes, splits or leaf nodes. This is to ensure that the learners remain weak, but can still be constructed in a greedy manner.

- Additive Model

Trees are added one at a time, and existing trees in the model are not changed. A gradient descent procedure is used to minimize the loss when adding trees.

Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error. Instead of parameters, we have weak learner sub-models or more specifically decision trees. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss (i.e. follow the gradient). We do this by parameterizing the tree, then

modify the parameters of the tree and move in the right direction by (reducing the residual loss).

Gradient boosting is a greedy algorithm and can overfit a training dataset quickly. It can benefit from regularization methods that penalize various parts of the algorithm and generally improve the performance of the algorithm by reducing overfitting.

### 3. Random Forest Classifier Base Method:-

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random forests has a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

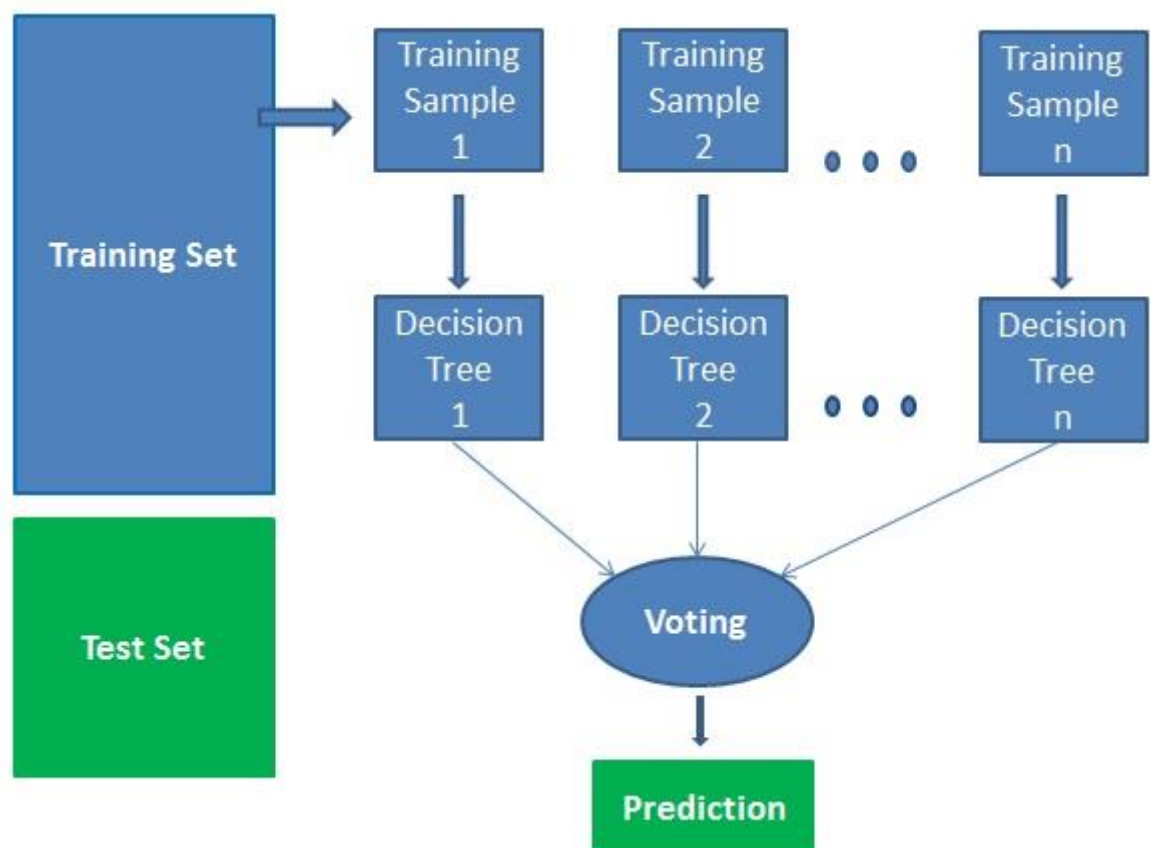
It technically is an ensemble method (based on the divide-and-conquer approach) of decision trees generated on a randomly split dataset. This collection of decision tree classifiers is also known as the forest. The individual decision trees are generated using an attribute selection indicator such as information gain, gain ratio, and Gini index for each attribute. Each tree depends on an independent random sample. In a classification problem, each tree votes and the most popular class is chosen as the final result. In the case of regression, the average of all the tree outputs is considered as the final result. It is simpler and more powerful compared to the other non-linear classification algorithms.

### How does the algorithm work?

It works in four steps:

1. Select random samples from a given dataset.

2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.
4. Select the prediction result with the most votes as the final prediction.



- Run and Evaluate selected models:-



```
In [29]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.33,random_state=42)

In [30]: print(x_train.shape,x_test.shape)
         print(y_train.shape,y_test.shape)

(108712, 31) (53546, 31)
(108712,) (53546,)

In [31]: DT=DecisionTreeClassifier()
         XGB=GradientBoostingClassifier()
         RFC=RandomForestClassifier()

In [32]: models = []
         models.append(('DecisionTreeClassifier', DT))
         models.append(('RandomForestClassifier', RFC))
         models.append(('GradientBoostingClassifier', XGB))

In [33]: Model = []
         score = []
         cvs=[]
         rocscore=[]
         for name,model in models:
             print('*****',name,'*****')
             print('\n')
             Model.append(name)
             model.fit(x_train,y_train)
             print(model)
             pre=model.predict(x_test)
             print('\n')
             AS=accuracy_score(y_test,pre)
             print('Accuracy_score = ',AS)
             score.append(AS*100)
             print('\n')
```

```
print('\n')
Model.append(name)
model.fit(x_train,y_train)
print(model)
pre=model.predict(x_test)
print('\n')
AS=accuracy_score(y_test,pre)
print('Accuracy_score = ',AS)
score.append(AS*100)
print('\n')
sc = cross_val_score(model, x, y, cv=10, scoring='accuracy').mean()
print('cross_val_score = ',sc)
cvs.append(sc*100)
print('\n')
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,pre)
roc_auc = auc(false_positive_rate, true_positive_rate)
print('roc_auc_score = ',roc_auc)
rocscore.append(roc_auc*100)
print('\n')
print('classification_report\n',classification_report(y_test,pre))
print('\n')
cm=confusion_matrix(y_test,pre)
print(cm)
print('\n')
plt.figure(figsize=(10,40))
plt.subplot(911)
plt.title(name)
print('sns.heatmap(cm,annot=True)')
plt.subplot(912)
plt.title(name)
plt.plot(false_positive_rate, true_positive_rate, label='AUC = %0.2f'% roc_auc)
plt.plot([0,1],[0,1],'-')
plt.legend(loc='lower right')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
print('\n\n')
```

- Key Metrics for success in solving problem under consideration:-



Home Page - X MCDP - Jupyter X (PDF) The Intel X Project01a\_re X SINDURANSIV X rproject report X TEAM-DS-Fin X random forest X Random Forest X + -

localhost8888/notebooks/MCDP.ipynb

jupyter MCDP Last Checkpoint: 17 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
print('****')

***** DecisionTreeClassifier *****

DecisionTreeClassifier()

Accuracy_score = 0.8588129832293728

Cross_Val_Score = 0.8596679186916241

roc_auc_score = 0.7154758275812634

classification_report
precision    recall  f1-score   support

   0       0.49    0.52    0.50     7342
   1       0.92    0.91    0.92     46204

 accuracy
macro avg    0.70    0.72    0.71     53546
weighted avg    0.86    0.86    0.86     53546

[[ 3803  3539]
 [ 4021 42183]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)
```

Type here to search

2035 03-12-2020

Home Page - X MCDP - Jupyter X (PDF) The Intel X Project01a\_re X SINDURANSIV X rproject report X TEAM-DS-Fin X random forest X Random Forest X + -

localhost8888/notebooks/MCDP.ipynb

jupyter MCDP Last Checkpoint: 17 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
***** RandomForestClassifier *****

RandomForestClassifier()

Accuracy_score = 0.9074627423150189

Cross_Val_Score = 0.9073266026467062

roc_auc_score = 0.7279713730783203

classification_report
precision    recall  f1-score   support

   0       0.76    0.48    0.59     7342
   1       0.92    0.98    0.95     46204

 accuracy
macro avg    0.84    0.73    0.77     53546
weighted avg    0.90    0.91    0.90     53546

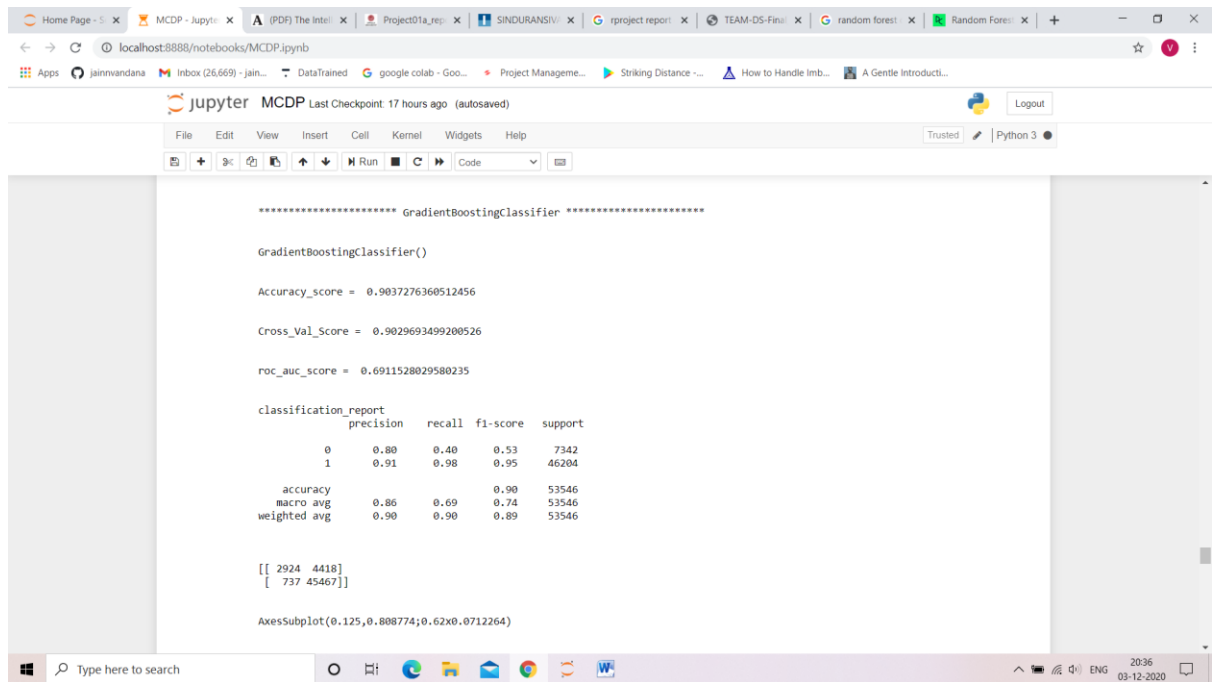
[[ 3529  3813]
 [ 1142 45862]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)

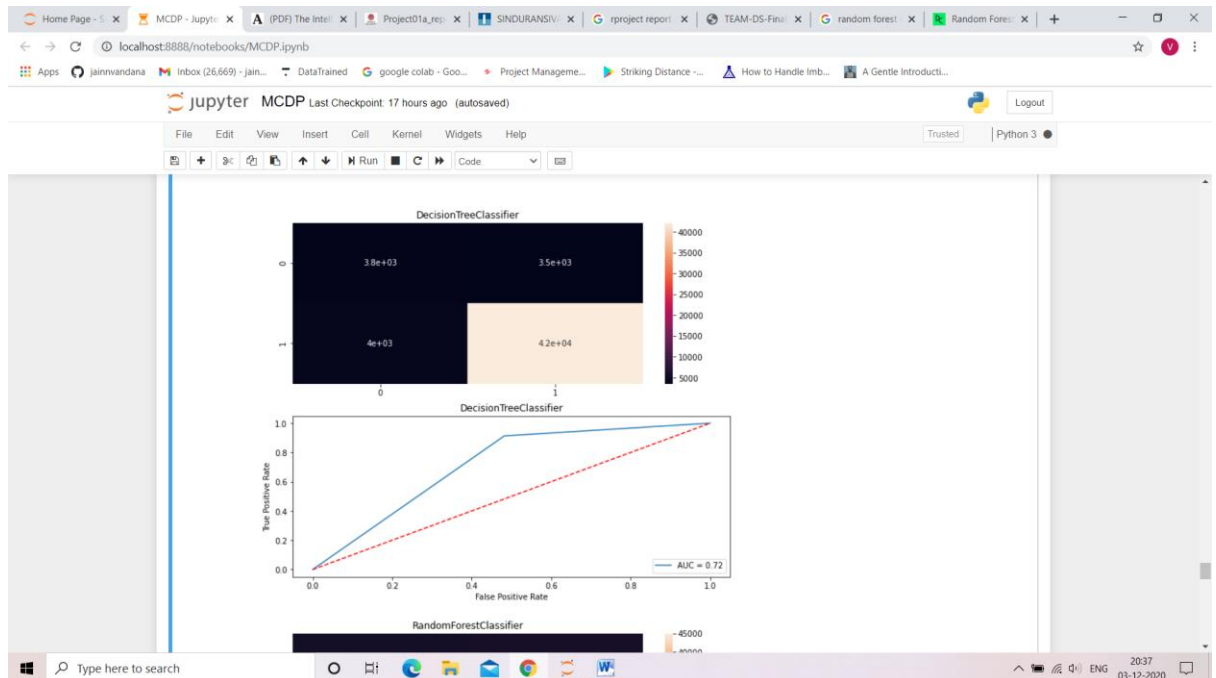
***** GradientBoostingClassifier *****
```

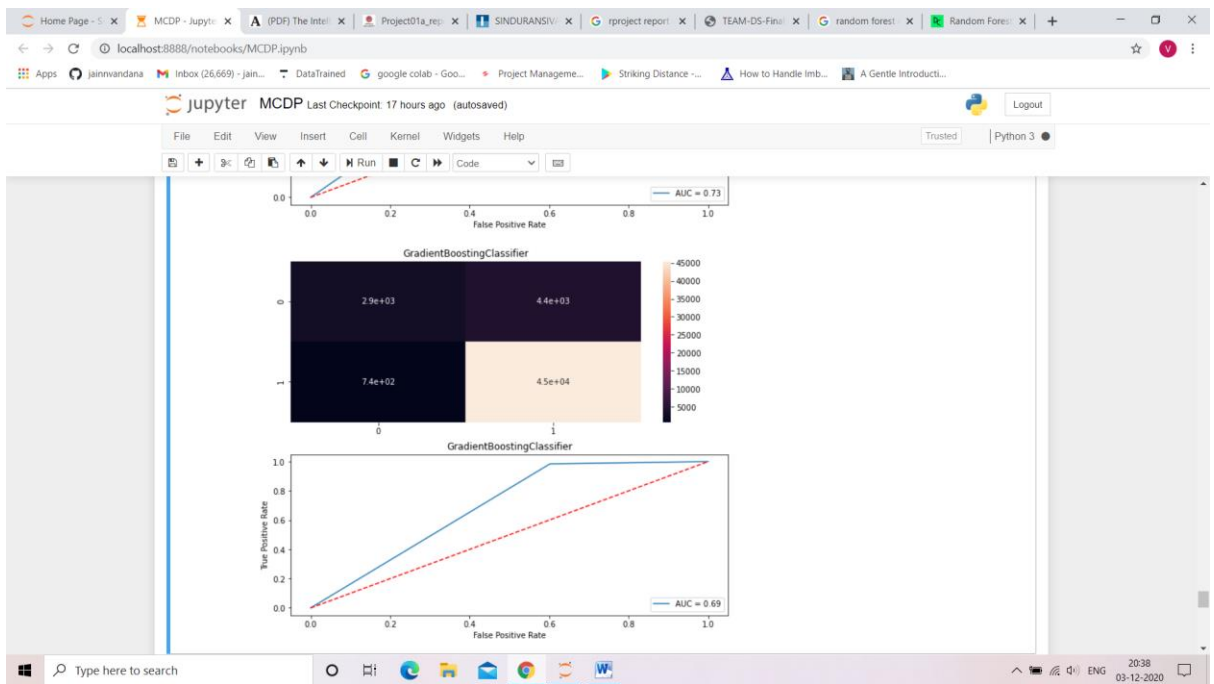
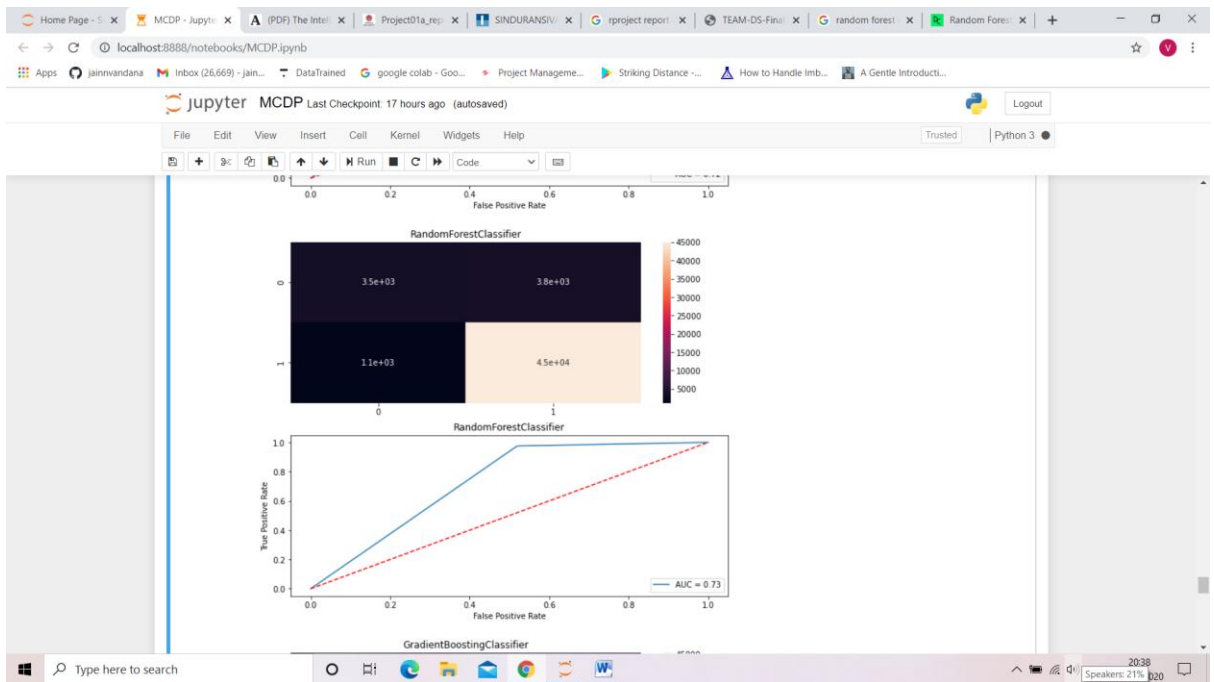
Type here to search

2036 03-12-2020

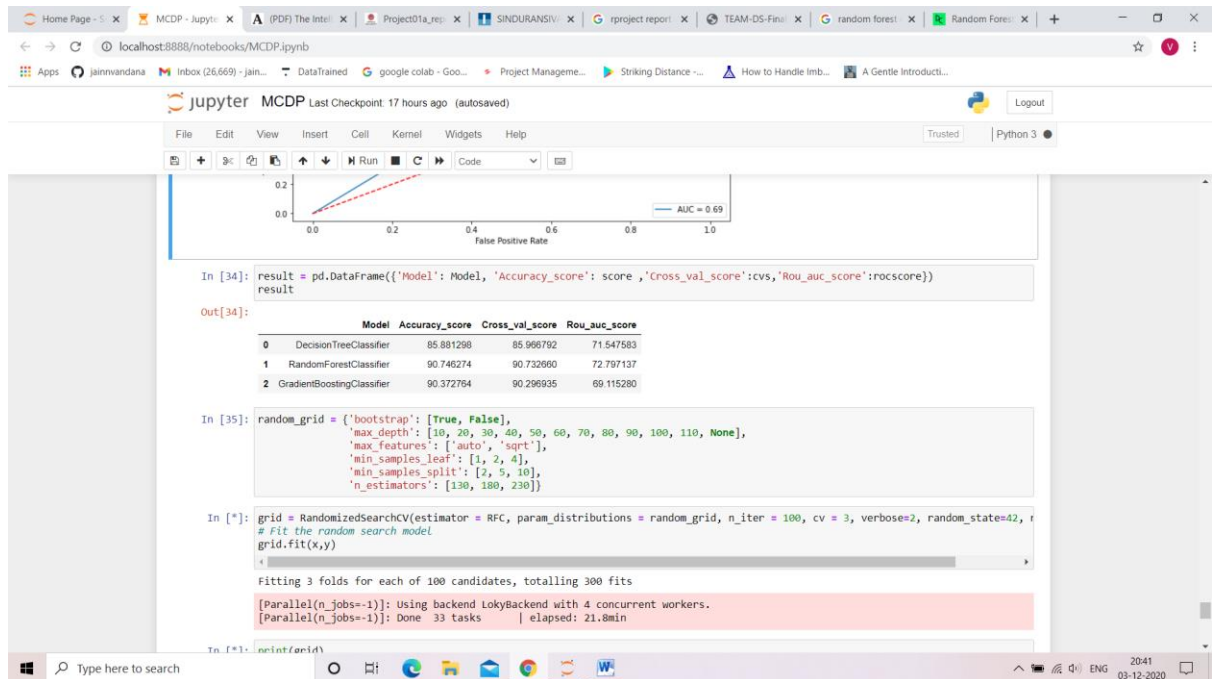


- Visualizations:-



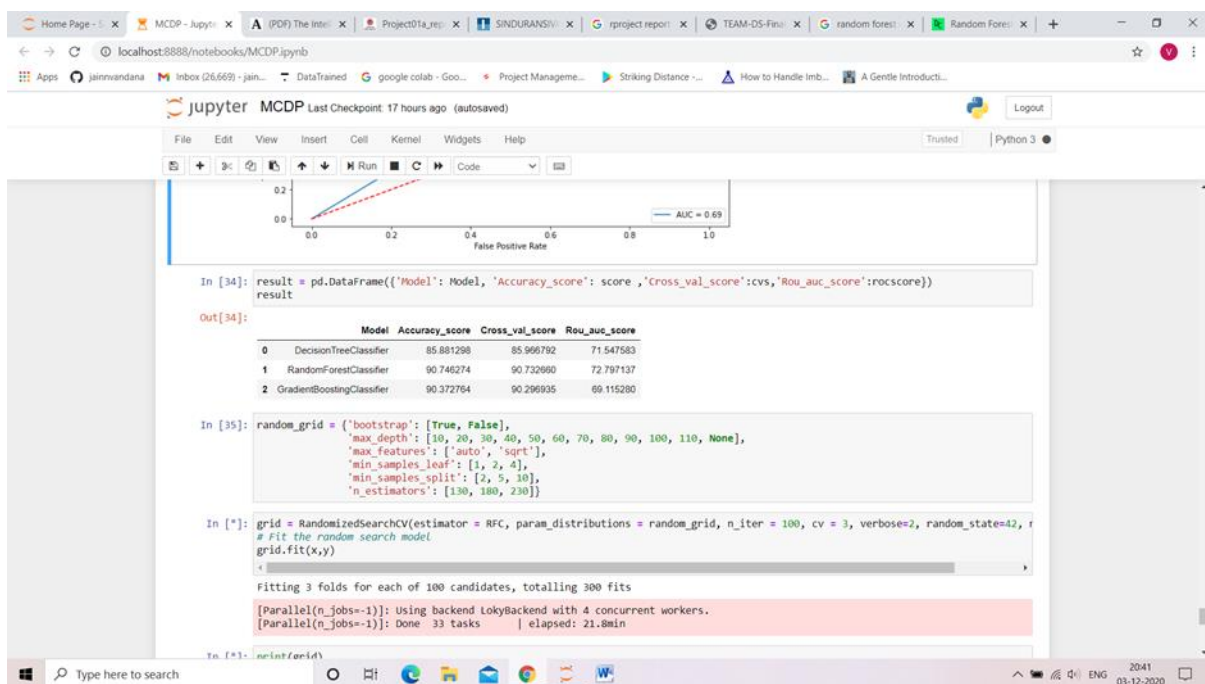


- Interpretation of the Results:-



## CONCLUSION:-

Now that we have implemented all the main classifiers Tree based machine learning algorithms due to our imbalanced dataset. Label '1' has approximately 87.5% records, while, label '0' has approximately 12.5% records. Let's compare the results. We have implemented the same project using Decision Tree classifier, Random Forest Classifier and Gradient Boosting Classifier.



We can see that although all of them have made good predictions and have high accuracy scores, Random Forest Classifier have performed slightly better.

*Thank You!*