MACHINE LEARNING ENGINEER NANODEGREE

Capstone Project

TANUKULA PRASANTH BABU

April-22-2019

SUPERVISED CLASSIFICATION

BANK TELEMARKETING DATASET

Definition

Project Overview

This project provides a way to predict the success of telemarketing calls for subscribing bank long term deposits. Term deposits play a vital role in financial institutions. Banks invest clients deposits on other financial assets which would earn better return than what the banks pay as interest on deposits to customers.

There is a growing competition among financial institutions in the retail segment. Banks invest massively on marketing products and service, with the latest technological advancements in data science and machine learning, banks are starting to adapt data driven decisions.

Project allows banks to identify customers that are more likely to subscribe for term deposits based on the given attributes so banks would target such customers, reason being term deposit customers are more likely to enroll into other products and services like insurance or pension funds so banks revenue would Increase massively. Banks would be able to invest in other financial products and gain higher interest rates on debt securities since the money is withheld for a certain duration. The goal is to predict how likely clients would enroll into a term deposit for a given set of variables.

Project applies machine learning algorithms and build a predictive model of the data set to provide suggestion for marketing campaign team. This is a real dataset collected from a Portuguese bank that used its own contact center to do tele- marketing campaigns and be able to attract clients for term deposits.

Problem Statement:

Financial instititutions invest heavily on marketing products and services. Banks would like to spend their money wisely and make the best use of time and resources to be able to identify customers that are more likely to subscribe for term deposits and maximize revenue.

Here I used Binary classification using supervised Learning algorithm to predict whether he subscribes the term deposit or not.

Metrics

The Precision is one of the evaluation metrics I have used in the benchmark model and also as well as in the proposed Project.

Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

Precision= True Positive/(True Positive+ False Positive)

	Predicted No	Predicted Yes
Actual No	TN	FN
Actual Yes	FP	TP

Precision=TP/ (TP+FP), Recall=TP/ (TP+FN), where

TP=True Positive

FP=False Positive

FN=False Negative

Her the data is imbalanced so, we use to apply precision to the data using the below formula because, we have to know how much we have predicted from the actual data. And we have to know how much interest the people are wishing to make a term deposits.

Analysis

Data Exploration

The dataset I chosen is from UCI Machine Learning Repository. The data is about Direct Marketing campaigns of a Portuguese banking Institution.

https://archive.ics.uci.edu/ml/datasets/Bank%2BMarketing

The dataset contains the following features.

Input variables:

bank client data:
1 - age (numeric)
2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5 - default: has credit in default? (categorical: 'no','yes','unknown')
6 - housing: has housing loan? (categorical: 'no','yes','unknown')
7 - loan: has personal loan? (categorical: 'no','yes','unknown')
related with the last contact of the current campaign:
8 - contact: contact communication type (categorical: 'cellular','telephone')
9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.
other attributes:
12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14 - previous: number of contacts performed before this campaign and for this client (numeric)

15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')
social and economic context attributes
16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)
17 - cons.price.idx: consumer price index - monthly indicator (numeric)
18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
20 - nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):
21 - y - has the client subscribed a term deposit? (binary: 'yes','no')

In the below screenshot, i can show the features and instances of data. I used head() function to display the 5 rows of data.

```
#print the first 5 rows
data.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no

And there are 45211 rows and 17 columns and I provided the picture below.

```
#To ensure the no.of rows and columns`
data.shape
```

```
(45211, 17)
```

Target class Variables:

The target column we need to predict is Yes or No using our classification.

```
: # TO count the no.of yes
  data.y.value_counts()
```

```
: no     39922
  yes     5289
  Name: y, dtype: int64
```

Using the value_counts() on feature, we can see the target variable, and we can count the yes values and no values as seen above.

Data types:

```
data.dtypes
```

```
age           int64
job          object
marital      object
education    object
default      object
balance       int64
housing      object
loan         object
contact      object
day           int64
month        object
duration      int64
campaign      int64
pdays         int64
previous      int64
poutcome     object
y            object
dtype: object
```

We have to know whether the data is integer data or object data(if any strings) in above picture.

Statistical Parameters :

Statistical parameters of dataset are as follows.

```
data.describe()
```

	age	balance	day	duration	campaign	pdays	previous
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0.580323
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2.303441
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0.000000
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

Correlation :

I checked the correlation factor using the corr function and the data is as follows.

```
[51]: data.corr()
```

t[51]:

	age	balance	day	duration	campaign	pdays	previous
age	1.000000	0.097783	-0.009120	-0.004648	0.004760	-0.023758	0.001288
balance	0.097783	1.000000	0.004503	0.021560	-0.014578	0.003435	0.016674
day	-0.009120	0.004503	1.000000	-0.030206	0.162490	-0.093044	-0.051710
duration	-0.004648	0.021560	-0.030206	1.000000	-0.084570	-0.001565	0.001203
campaign	0.004760	-0.014578	0.162490	-0.084570	1.000000	-0.088628	-0.032855
pdays	-0.023758	0.003435	-0.093044	-0.001565	-0.088628	1.000000	0.454820
previous	0.001288	0.016674	-0.051710	0.001203	-0.032855	0.454820	1.000000

Mapping values:

I mapped the yes to ‘1’ and No values to ‘0’ to get the better convenience.

```
output=output.map({'yes':1,'no':0})
```

```
output[:500]
```

```
0      0
1      0
2      0
3      0
4      0
5      0
6      0
7      0
8      0
9      0
10     0
11     0
12     0
13     0
14     0
15     0
16     0
17     0
18     0
19     0
```

Exploratory Visualization:

We can see the different levels of a categorical variable by the color of plot elements.

```
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x19003f20c50>
```

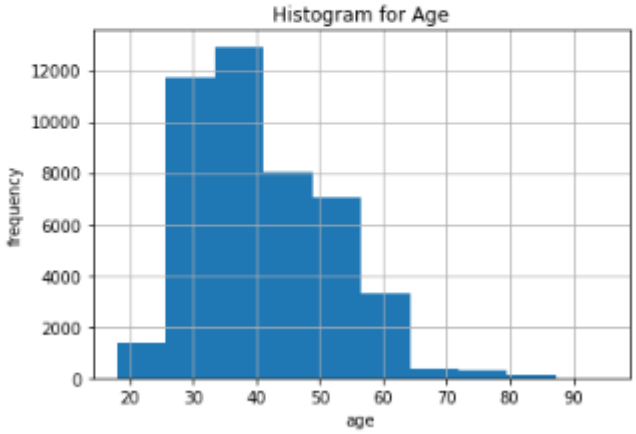
Histograms:

The histograms drawn for this telemarketing dataset is plotted below. The below visuals show the correlation features between features and target class. We can see Every feature can be in relation with the target class.

In [18]:
```
data.age.hist()

plt.xlabel("age")
plt.ylabel("frequency")
plt.title("Histogram for Age")
```

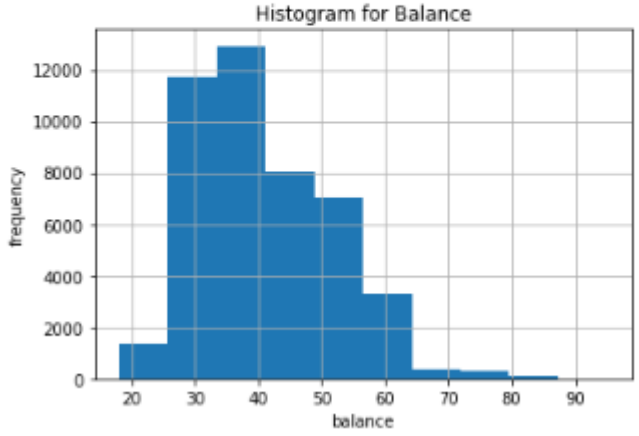
Out[18]:
```
Text(0.5,1,'Histogram for Age')
```



```
data.age.hist()

plt.xlabel("balance")
plt.ylabel("frequency")
plt.title("Histogram for Balance")
```

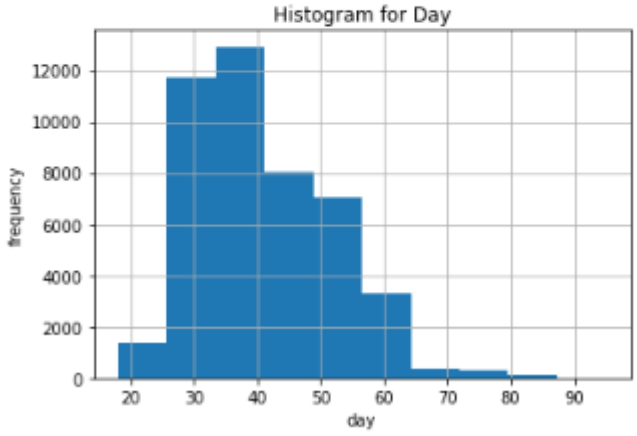
```
Text(0.5,1,'Histogram for Balance')
```




```
data.age.hist()

plt.xlabel("day")
plt.ylabel("frequency")
plt.title("Histogram for Day")
```

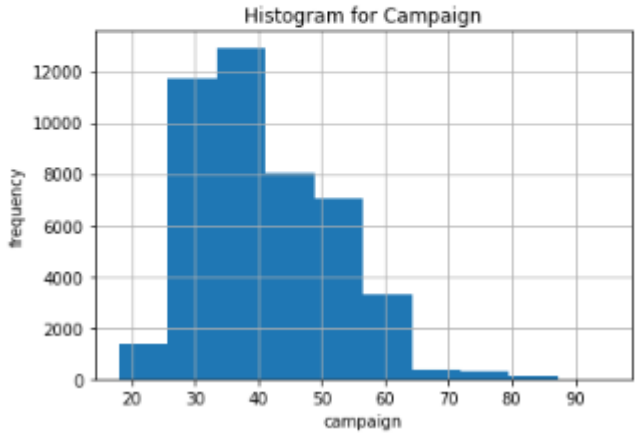
```
Text(0.5,1,'Histogram for Day')
```



```
data.age.hist()

plt.xlabel("campaign")
plt.ylabel("frequency")
plt.title("Histogram for Campaign")
```

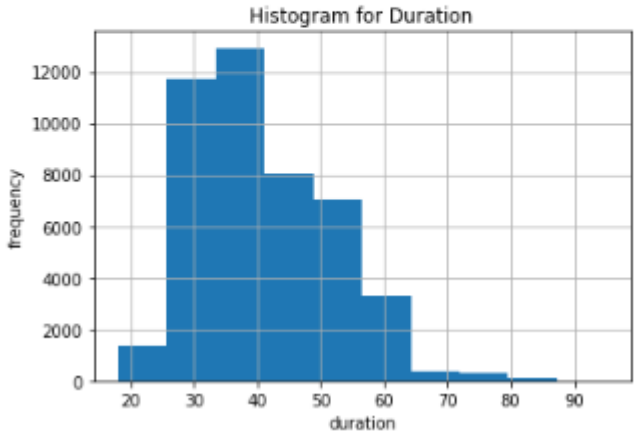
```
Text(0.5,1,'Histogram for Campaign')
```




```
data.age.hist()

plt.xlabel("duration")
plt.ylabel("frequency")
plt.title("Histogram for Duration")
```

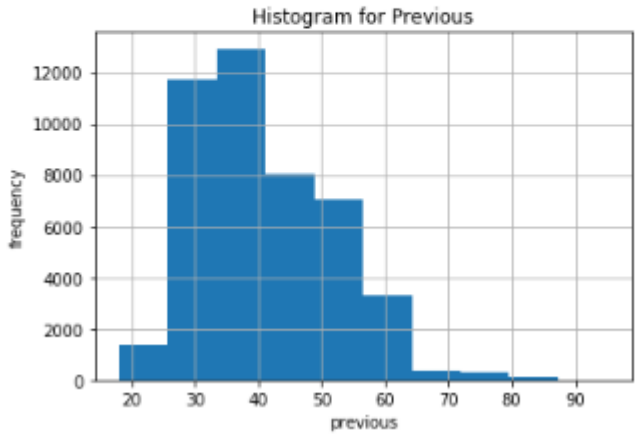
```
Text(0.5,1,'Histogram for Duration')
```



```
data.age.hist()

plt.xlabel("previous")
plt.ylabel("frequency")
plt.title("Histogram for Previous")
```

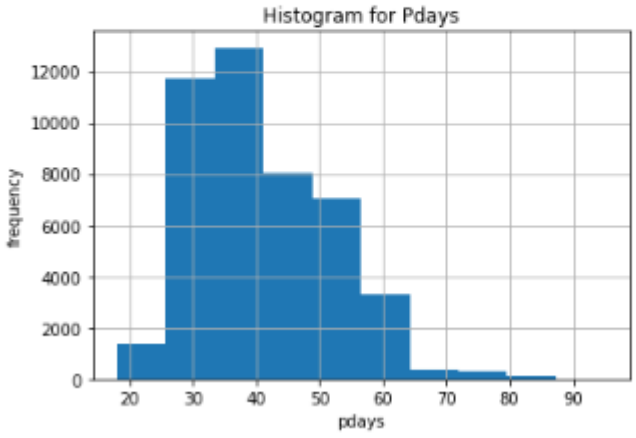
```
Text(0.5,1,'Histogram for Previous')
```




```
data.age.hist()

plt.xlabel("pdays")
plt.ylabel("frequency")
plt.title("Histogram for Pdays")
```

```
Text(0.5,1,'Histogram for Pdays')
```



Algorithm and Techniques:

The classification algorithms used for this dataset are:

- Naïve bayes GaaussianNB – Benchmark model
- RandomForest classifier
- DecisionTree classifier

- LGBMClassifier

Naïve Bayes GaussianNb:

As all we know, Naïve Bayes machine learning algorithm is a classification algorithm and it works effectively for classification tasks. As we can browse, naïve bayes is not a single algorithm, which have a family of algorithms. It says every pair of features being classified is independent of each other. It can make probabilistic predictions well.

The guassianNB algorithm is used when the dataset is too large or too big. And also this method is good at performance and it is good in numerical stability. This GuassianNB algorithm has a simple approach, fast and works accurate for prediction.

sklearn.naive_bayes.GaussianNB

class sklearn.naive_bayes. **GaussianNB** (*priors=None, var_smoothing=1e-09*)

Gaussian Naive Bayes (GaussianNB)

This Naïve Bayes algorithm works on Bayes Theorem to model the conditional relationship of each attribute to the class variable. I will mention the formula below.

$$P(H \mid E) = \frac{P(E \mid H) * P(H)}{P(E)}$$

where

- P(H) is the probability of hypothesis H being true. This is known as the prior probability.
- P(E) is the probability of the evidence(regardless of the hypothesis).
- P(E|H) is the probability of the evidence given that hypothesis is true.
- P(H|E) is the probability of the hypothesis given that the evidence is there.

Random Forest:

Random Forest algorithm uses tree models and classifies as high variance and low bias models. This algotithm can able to handle missing values and gives the good accuracy for a huge dataset. It splits the data according to features upto the instances are there for the value of target feature .The only thing to keep in mind is time, it takes much time to process. It handles by producing leaves to the data. The main advantage with this is it won't allow the trees to be overfit the model. It helps to overcome overfitting by averaging many trees and gives good output.

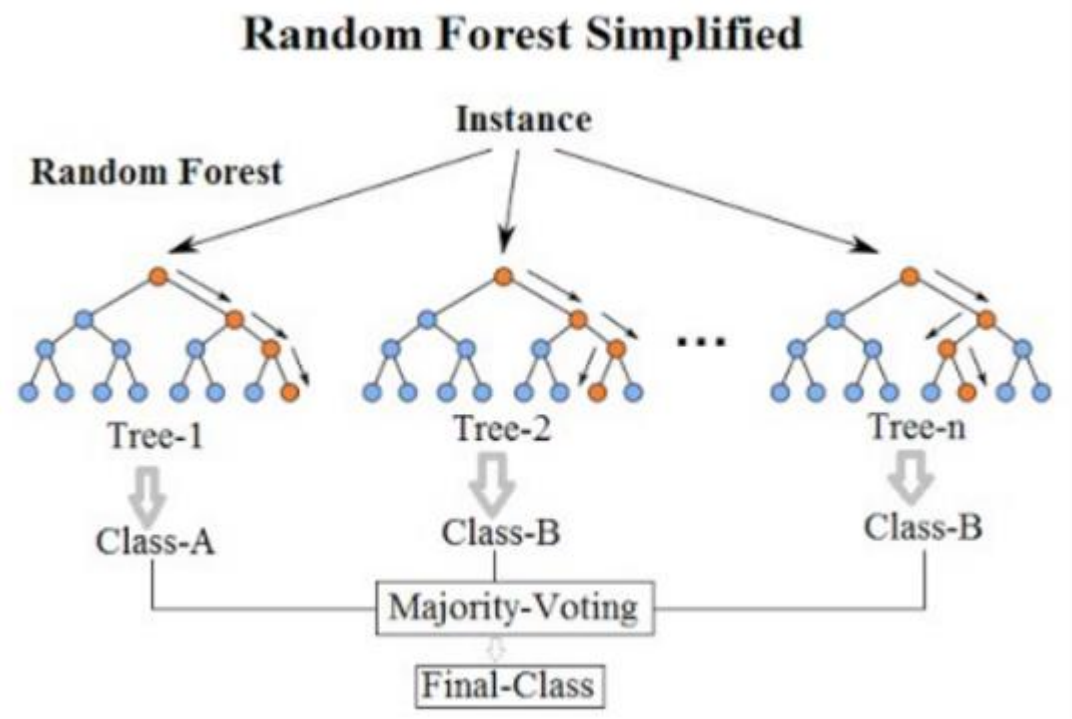
3.2.4.3.1. `sklearn.ensemble`.RandomForestClassifier

class `sklearn.ensemble.` **`RandomForestClassifier`** (*n_estimators='warn', criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None*) ¶ [source]

A random forest classifier.

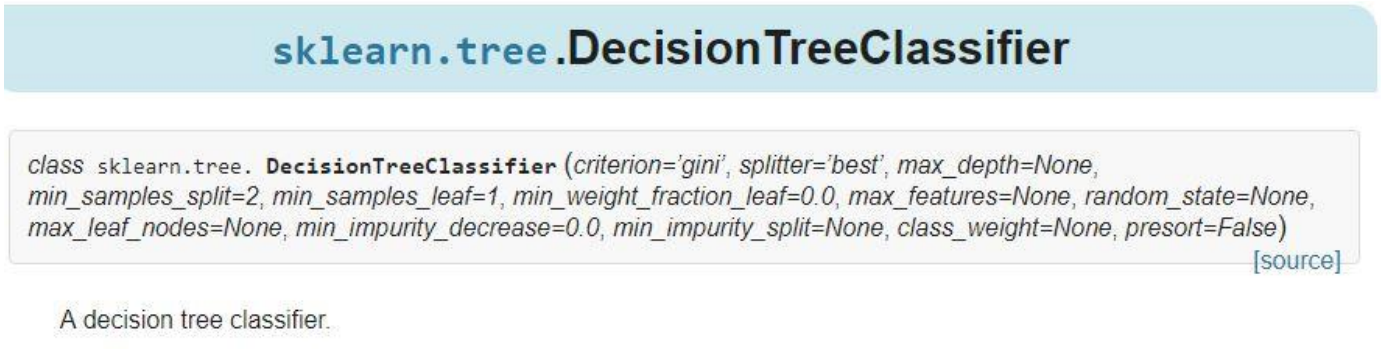
A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

Feature importance or Random Forest can be calculated by decrease in node impurity weighted by probability of reaching that node.



DecisionTree

As all we know, decision trees are used for classification and regression.This algorithm helps in numerical and categorical data efficiently. This algorithm is good at removing blank values and can perform tasks as dummy variables. It is also good at normalizing the large data . And it uses a non parametric supervised learning. It can also helpful n analysing datasets which have one type of variable also. It works well even its assumptions disturbed by true model of data also. And DecisionTree works well by training with more trees, and it sometimes unstable even to bear small variations in data leads to generate complete different tree.



Advantages and disadvantages:

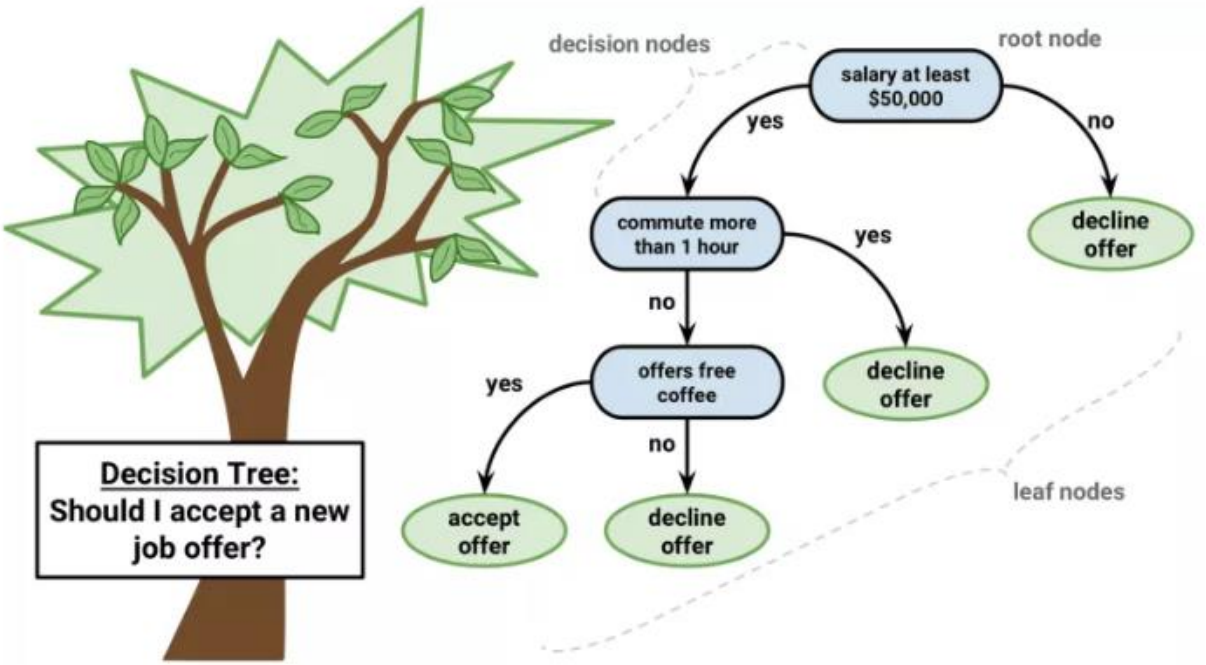
This decision tree classifier builds the fastest and shortest tree.

For creating a tree it works to search the entire dataset.

It prefers for a limited attributes until entire data is classified.

It can't handle missing values and numeric attributes.

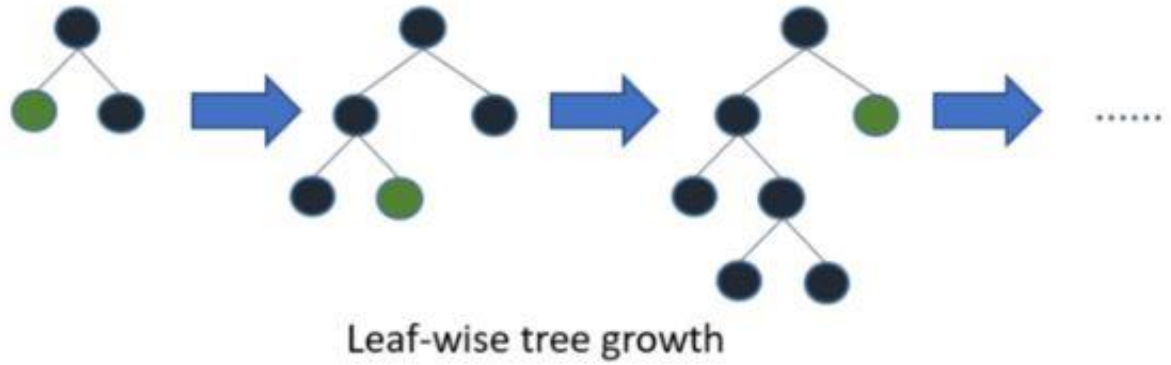
While making a decision, it tests only one attribute for a while.



LGBMClassifier:

Light GBM classifier is gradient boosting framework and it uses tree based learning algorithm. It helps in binary classification and by supervised learning.

The Light GBM classifier tree grows in a vertical in shape such that leaf shape. Other algorithms works as a horizontally grown algorithm. While growing same leaf, it helps in reducing more loss comparing to other algorithms.

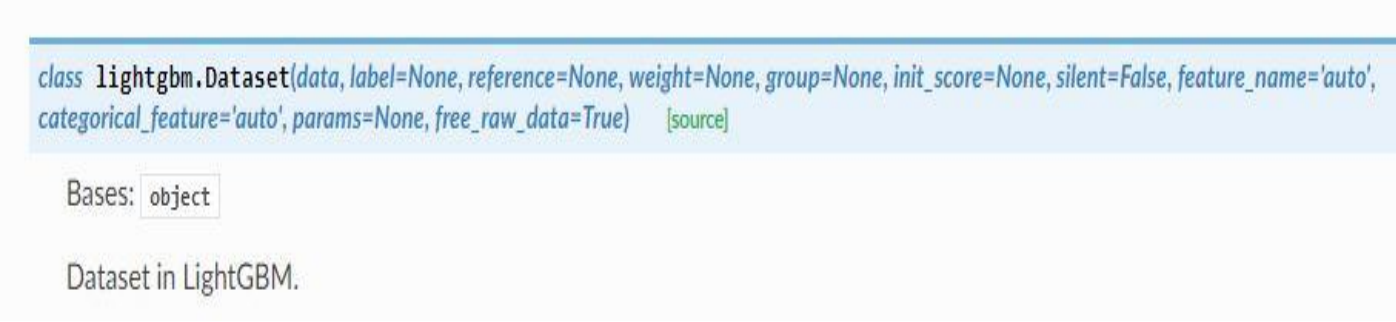


Leaf-wise tree growth

Explains how LightGBM works

Comparing to other algorithms, LGBM is gives faster results. It is famous for its accuracy of results. As we can know that day by day, data is increasing a lot(traditional data), LGBM works very fast even it supports GPU learning.

The main drawback to use this is it is not a good model for small datasets, it may overfit the data. As our project is 45000 rows, it works very efficient.



Benchmark:

Here,I used NaïveBased GuassianNB as my benchmark model and it given precision_score of “0.40”. While I keep applying different models to my dataset, if it gives better result than this benchmark results, Then I consider the best of all.

Methodology:

Data pre-processing:

Here, in this step, we will pre-process the data. This step is the foremost step and has to be done initially after the dataset. We can load the dataset by using the fuction “read_csv”.

As the data is semicolumn separated values, I set an variable separation as sep=”;” then it is set to separate using the semicolumn. We can able to view the no.of rows and columns by using shape function. I checked for the null values then, and made it sum of values, then I found no null values in the features by using isnull() function. And checked for the dummy values by using get_dummies function(),And I mapped the yes to ‘1’ and No values to ‘0’ to get the better convenience. all the functions I used will provide proofs below:

```
#csv file loaded into data
data=pd.read_csv("bank-full.csv",sep=';')
```

```
#To check the no.of  null values
data.isnull().sum()
```

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

```
[22]: data1=pd.get_dummies(data1)
```

```
[23]: data1.head()
```

t[23]:

	age	balance	day	duration	campaign	pdays	previous	job_admin.	job_blue-collar	job_entrepreneur	...	month_jun	month_mar	month_may	month_nov	mc
0	58	2143	5	261	1	-1	0	0	0	0	...	0	0	1	0	
1	44	29	5	151	1	-1	0	0	0	0	...	0	0	1	0	
2	33	2	5	76	1	-1	0	0	0	1	...	0	0	1	0	
3	47	1506	5	92	1	-1	0	0	1	0	...	0	0	1	0	
4	33	1	5	198	1	-1	0	0	0	0	...	0	0	1	0	

5 rows × 51 columns

```
output=output.map({'yes':1,'no':0})
```

```
output[:500]
```

```
0      0
1      0
2      0
3      0
4      0
5      0
6      0
7      0
8      0
9      0
10     0
11     0
12     0
13     0
14     0
15     0
16     0
17     0
18     0
19     0
```

Implementation:

First of all , I imported all the algorithms and random states and hyper parameters, was imported to evaluate training and testing of every algorithm. The evaluation metrics – precision_score is imported as we know it helps the percent of all relevant documents which is returned by search. I splitted the data to training and testing.

```
In [18]: from sklearn.model_selection import train_test_split
```

```
In [19]:  X_train, X_test, y_train, y_test = train_test_split(d
ata1, output, test_size=0.2, random_state=42)
```

Then the second attempt is applying the algorithms. I used GuassianNB and found precision score and recorded. Then after, I used Random Forest algorithm to check whether the good precision score is obtained or not. Then I used Decision Tree classifier and I set the random state to 3 and applied. The obtained result is stored, and compared with the benchmark model. Later, I used LGBMClassifier and applied the same training and testing data with random state, and I found precision score and I observed it is the best precision score of all. Out of all the algorithms applied I have observed that LGBMClassifier have gained the good result among all. Coming to complications, I faced as the data is imbalanced data. And I faced a bit complexity in tuning the parameters in the LGBMClassifier.

Refinement:

Coming to this Refinement step, I applied Decision Tree Classifier and selected random state as 3 , the result obtained is 0.48. Then for the GuassianNB the precision score obtained is 0.40, and for the Random forest classifier , the precision score obtained is 0.64. Then I applied LGBM classifier and obtained a high precision score of 0.66 which is best value obtained from all the algorithm's. Theni applied GridSearchCV to obtain the best parameters, I altered the parameter values num_leaves, max_depth,learning_rate,n_estimators by parameter tuning and obtained a bit high value of 0.67. I will picturized above statements in below clippings.

The parameters were

```
params = {'num_leaves':[11,8,9], 'max_depth':[3,1,2], 'learning_rate':[0.1,0.5,0.3], 'n_estimators':[100,130,150] }
```

and after applying the grid.best_params_ I got the following as best parameters.

```
In [48]: grid.best_params_
Out[48]: {'learning_rate': 0.1, 'max_depth': 1, 'n_estimators': 100, 'num_leaves': 11}
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import fbeta_score

clf_B = DecisionTreeClassifier(random_state = 3)
clf_B.fit(X_train,y_train)
pred=clf_B.predict(X_test)
precision_score(y_test,pred)
```

```
0.48455428067078554
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import fbeta_score

clf_E = GaussianNB()
clf_E.fit(X_train,y_train)
pred=clf_E.predict(X_test)
precision_score(y_test,pred)
```

```
0.40375586854460094
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_score

clf_G=RandomForestClassifier(random_state = 3)
clf_G.fit(X_train,y_train)
pred=clf_G.predict(X_test)
precision_score(y_test,pred)
```

```
C:\Users\roy\Anaconda3\lib\site-packages\sklearn\ensemble\weight_boosting.py:29: DeprecationWarning: nump
an internal NumPy module and should not be imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```

```
0.6461038961038961
```

```
from lightgbm import LGBMClassifier
clf_H=LGBMClassifier(random_state = 3)
clf_H.fit(X_train,y_train)
pred=clf_H.predict(X_test)
precision_score(y_test,pred)
```

```
C:\Users\roy\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: Dep
rray is ambiguous. Returning False, but in future this will result in an error.
not empty.
  if diff:
```

```
0.6607142857142857
```

```
from sklearn.metrics import make_scorer
from sklearn.model_selection import GridSearchCV

params = {'num_leaves':[11,8,9], 'max_depth':[3,1,2], 'learning_rate':[0.1,0.5,0.3], 'n_estimators':[100,130,150]

scoring_fnc = make_scorer(precision_score)

grid = GridSearchCV(estimator=clf_H,param_grid=params,scoring=scoring_fnc,cv=10)

grid = grid.fit(X_train, y_train)
var2=grid.predict(X_test)
print(precision_score(y_test, var2))
```

```
is not empty.
  if diff:
C:\Users\roy\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: DeprecationWarning: The truth valu
array is ambiguous. Returning False, but in future this will result in an error. Use `array.size > 0` to check
is not empty.
  if diff:
C:\Users\roy\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: DeprecationWarning: The truth valu
array is ambiguous. Returning False, but in future this will result in an error. Use `array.size > 0` to check
is not empty.
  if diff:
C:\Users\roy\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: DeprecationWarning: The truth valu
array is ambiguous. Returning False, but in future this will result in an error. Use `array.size > 0` to check
is not empty.
  if diff:

0.6775956284153005
```

Results:

Model Evaluation and Validation:

The final model I have chosen I LGBMClassifier which had given more accuracy 0.66. While applying the make_scorer function to precision score and applying GridSearCV, I got the 0.67 value, which is a bit much satisfying such that a bit more than 0.66.

And Tried with different random states such as 3, 43, 66, 23… then I came to know confidently that it is going to work for new data and then the result obtained is 0.67 which is same.

Justification:

The Benchmark model(GuassianNB) have a precision score of 0.40% ,best model we applied (LGBClassifier) obtained precision score of 0.66% . So that we came to know that the model is performing well.

And the Classification report shows the good results of precision score.

There is a reasonable difference we can observe between benchmark and final model of 0.40 % and 0.66% , we can see the difference in precision score and improved by 1 . And the final score is 67% which is a bit higher than 66%.

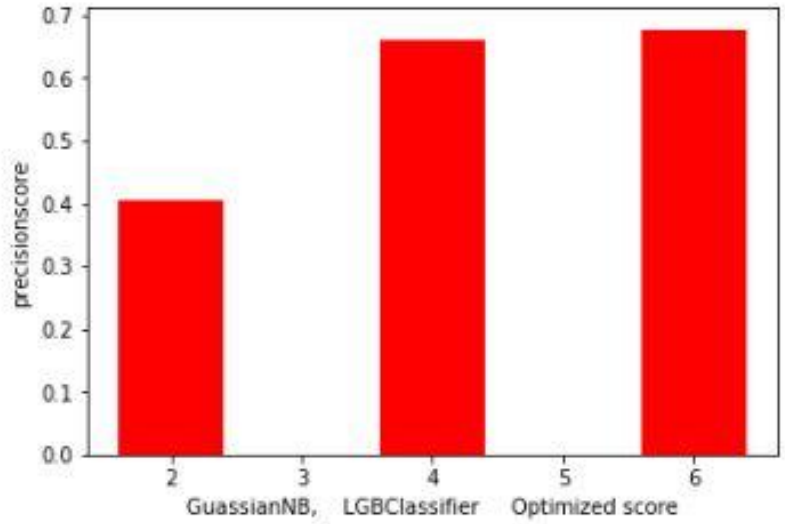
Conclusion:

Free-Form Visualization:

For better visualization, I used bar graphs. I have shown the Benchmark model, LGBClassifier and optimized model in the below screen shot.

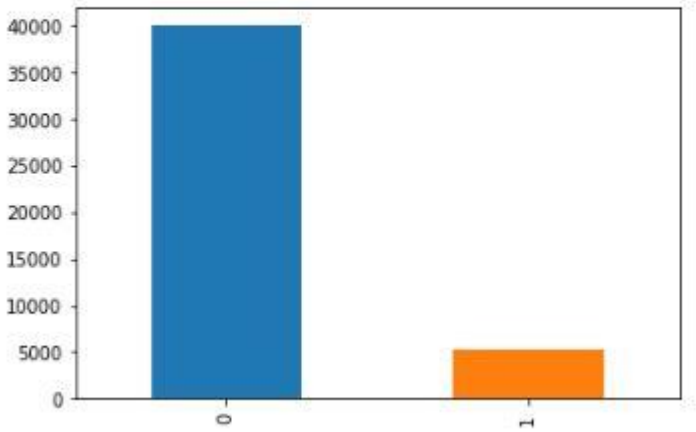
I have taken x-axis as the three model names, and y- axis as precision score and the scores given as arguments to variable and plotted.

```
precisionscore=[0.4037558685,0.6607142,0.6775956284153005
]
plt.bar(lab,f1scores,color='red')
plt.xlabel("GuassianNB,    LGBClassifier     Optimized score")
plt.ylabel("precisionscore")
plt.show()
```



```
output.value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x223063d3b70>
```



Reflection:

In this project ,the main goal is to build a classifier which can able to correctly classify whether the client subscribes the term deposit or not.

- Initially, the data preprocessing and data cleaning is made for the dataset such as identifying dummy variables, removing null values, counting yes or no values and other operaitions.
- Various Supervised machine learning algorithms have been used to get the best results.
- Seperating the data to training and testing, and performing random states.
- The metrics obtained from using these algorithm are evaluated each other, and the model which gives the best model is taken to consideration.
- The difficult part I faced is tuning the parameters in the algorithm and using the bestsearchcv makes more time to process the data and result is delayed as it is the somewhat big dataset.
- I felt every part which I don't know as interesting and made me to learn using websites and performing with the new matter I learnt.
- Considering the quality of dataset, the model performed well on predicting yes or no values correctly, and precision score is good even in random states.

Improvement:

- As it is unbalanced data, we have more No predictions and less Yes cases. Further the yes cases can be increased by giving the more training data and considering the people with more yes cases.
- We can try to somewhat to gain score for this dataset by further improving to Ensemble algorithms from my research.