MACHINE LEARNINIG PROJECT REPORT

SUBMITTED BY:

Sankari BALASUBRAMANIYAN

PACKAGES USED

For data manipulation

- Numpy
- Pandas

For data visualisation

- Matplotlib
- Seaborn

For Statistical computation

- Scipy
- Statsmodels

For machine learning

- Sklearn

DATA OVERVIEW

STATISTICAL ANALYSIS OF THE DATA

- An overview of the data.

	a0	a1	a2	a3	a4	a5	a6	a7	a8	a9	...	a21	a22	a23	a24	a25	a26	a27	a28	a29	a30
0	Campagne	0	342	2015	Juillet	27	1	0	0	2	...	3	ND	NaN	NaN	0	T	0.0	0	0	2015-07-01
1	Campagne	0	737	2015	Juillet	27	1	0	0	2	...	4	ND	NaN	NaN	0	T	0.0	0	0	2015-07-01
2	Campagne	0	7	2015	Juillet	27	1	0	1	1	...	0	ND	NaN	NaN	0	T	75.0	0	0	2015-07-02
3	Campagne	0	13	2015	Juillet	27	1	0	1	1	...	0	ND	304.0	NaN	0	T	75.0	0	0	2015-07-02
4	Campagne	0	14	2015	Juillet	27	1	0	2	2	...	0	ND	240.0	NaN	0	T	98.0	0	1	2015-07-03

- Statistical description of the data.

	a1	a2	a3	a5	a6	a7	a8	a9	a10
count	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119386.000000
mean	0.370416	104.011416	2016.156554	27.165173	15.798241	0.927599	2.500302	1.856403	0.103890
std	0.482918	106.863097	0.707476	13.605138	8.780829	0.998613	1.908286	0.579261	0.398561
min	0.000000	0.000000	2015.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	18.000000	2016.000000	16.000000	8.000000	0.000000	1.000000	2.000000	0.000000
50%	0.000000	69.000000	2016.000000	28.000000	16.000000	1.000000	2.000000	2.000000	0.000000
75%	1.000000	160.000000	2017.000000	38.000000	23.000000	2.000000	3.000000	2.000000	0.000000
max	1.000000	737.000000	2017.000000	53.000000	31.000000	19.000000	50.000000	55.000000	10.000000

- *isnull().sum* function used to check for null elements in the data, to be removed or replaced later.

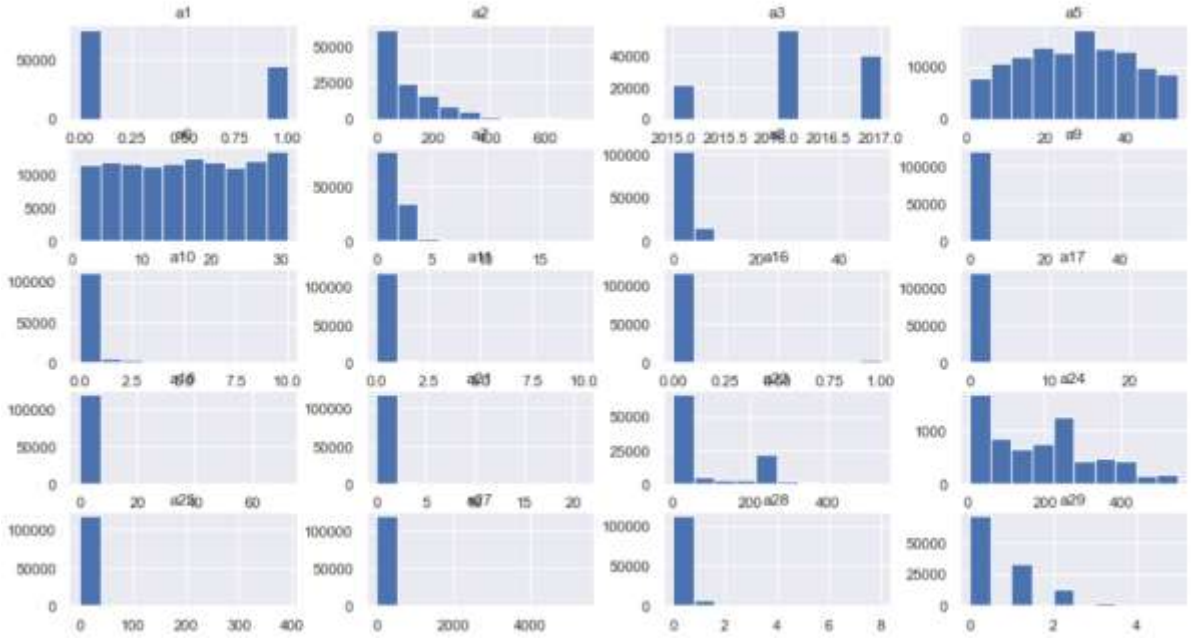
```
a0          0
a1          0
a2          0
a3          0
a4          0
a5          0
a6          0
a7          0
a8          0
a9          0
a10         4
a11         0
a12         0
a13       488
a14         0
a15         0
a16         0
a17         0
a18         0
a19         0
a20         0
a21         0
a22         0
a23     16340
a24    112593
a25         0
a26         0
a27         0
a28         0
a29         0
a30         0
dtype: int64
```

- Mean of numerical data to get an understanding of a label's typical value.

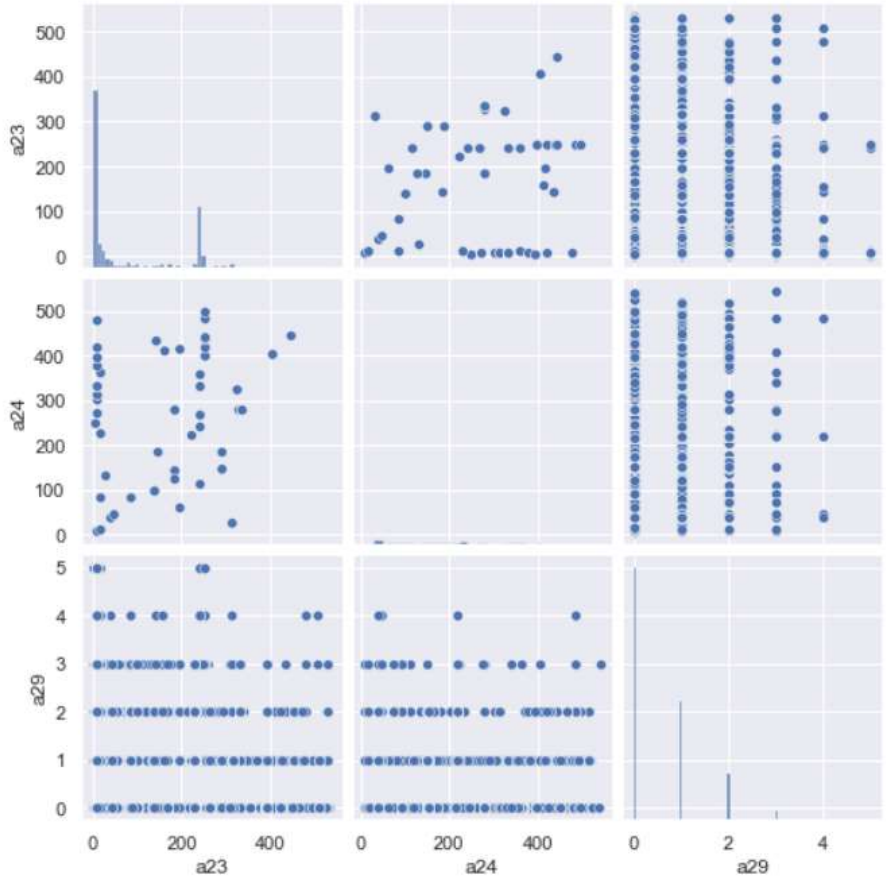
```
a1        0.370416
a2      104.011416
a3     2016.156554
a5       27.165173
a6       15.798241
a7        0.927599
a8        2.500302
a9        1.856403
a10       0.103890
a11       0.007949
a16       0.031912
a17       0.087118
a18       0.137097
a21       0.221124
a23      86.693382
a24     189.266735
a25       2.321149
a27     101.831122
a28       0.062518
a29       0.571363
dtype: float64
```

GRAPHICAL ILLUSTRATION

- **Histogram analysis of the data**



- **Pairplot on a23,a24,a29 columns**

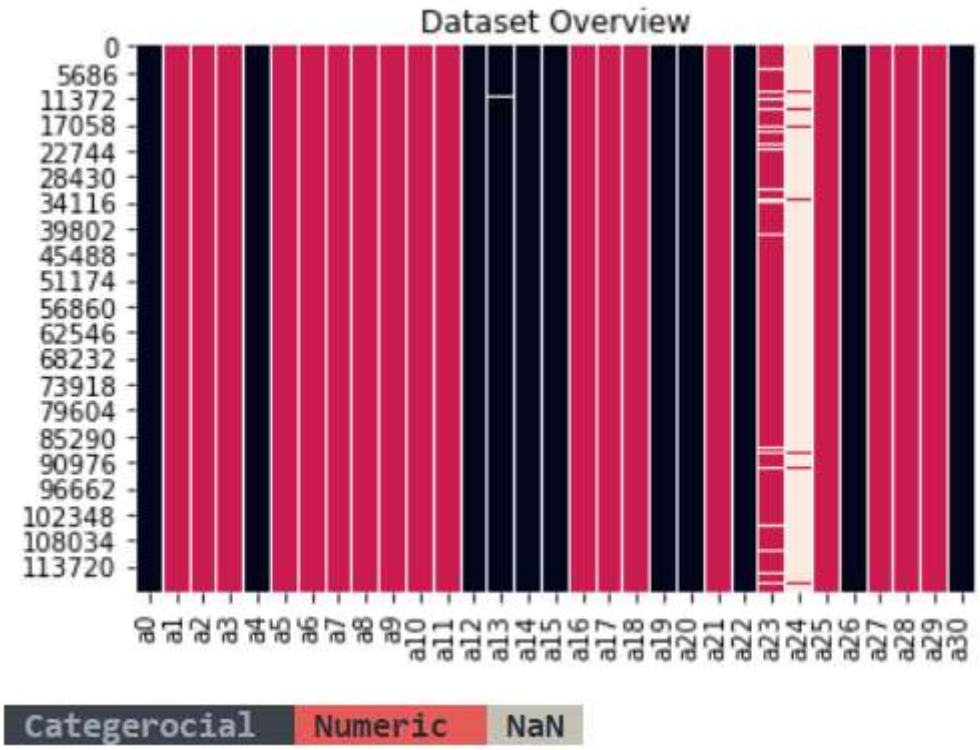


- **Correlation plot**

DATA INTERPRETATION

There are 119390 rows and 31 columns

- Each row of the table represents an observation
- A1 and A26 are dependent variable that we want to learn about and predict.
- a1 a2 a3 a5 a6 a7 a8 a9 a10 a11 a16 a17 a18 a21 a23 a24 a25 a27 a28 a29 are numerical variable while others are categorical.
- a10 a13 a23 a24 contains missing data.
- Clearly this is a classification problem with discrete variables
- Dataset overview to visualise column type and missing values using heat map



DATA PREPARATION FOR LEARNING

DATA PREPROCESSING

- Partitioning of data to train and test to feed the model.

```
X_train shape: (79991, 30) | X_test shape: (39399, 30)
y_train mean: 0.37 | y_test mean: 0.37
30 features: ['a0', 'a2', 'a3', 'a4', 'a5', 'a6', 'a7', 'a8', 'a9', 'a10', 'a11', 'a12', 'a13', 'a14', 'a15', 'a16', 'a17', 'a1
8', 'a19', 'a20', 'a21', 'a22', 'a23', 'a24', 'a25', 'a26', 'a27', 'a28', 'a29', 'a30']
```

- Data with missing values must be replaced/dropped and unnecessary information to be removed. For better learning, it is good to split the data as train and test and then replace NAs with average of training set.

  In this case a10 a23 contains missing data, and those have to filled with mean of training set.

  It's better to remove a24 column since it more than 50% of it's data is missing and hence becomes irrelevant information and a13 since it has missing string values.

- Categorical data must be encoded, so the labels (Strings) are converted to integers before feeding the data to the model.

This is achieved using **One-Hot-Encoding method**, transforming all categorical columns with n unique values into n-1 dummies. Every row which had the particular categorical content is given a value '1' and rest '0'

The original columns are dropped after transformation.

A sample of how a4 is transformed:

	a4	a4_Avril	a4_Decembre	a4_Fevrier	a4_Janvier	a4_Juillet	\
67587	Mai	0	0	0	0	0	
56208	Septembre	0	0	0	0	0	
14523	Fevrier	0	0	1	0	0	
111556	Mai	0	0	0	0	0	
107291	Mars	0	0	0	0	0	

	a4_Juin	a4_Mai	a4_Mars	a4_Novembre	a4_Octobre	a4_Septembre
67587	0	1	0	0	0	0
56208	0	0	0	0	0	1
14523	0	0	0	0	0	0
111556	0	1	0	0	0	0
107291	0	0	1	0	0	0

TEST SEVERAL MODELS

MODEL DESIGN

We used some of the best classifier models to train and test our dataset. All the data went through a data preprocessing methods before being fed into the model. The models chosen for the purpose of this classification are given below :

- GaussianNB
- Decision Tree Classifier
- Stochastic Gradient Descent Classifier
- KNeighbors Classifier
- Linear Regression
- Logistic Regression

Processing time for training and predicting of models

```
               GaussianNB fit : 2.050 ; Pred 1.685
--------------------------------------------------------------------------
   DecisionTreeClassifier fit : 7.828 ; Pred 0.328
--------------------------------------------------------------------------
            SGDClassifier fit : 12.250 ; Pred 0.293
--------------------------------------------------------------------------
     KNeighborsClassifier fit : 0.241 ; Pred 250.096
--------------------------------------------------------------------------
C:\Users\bsank\anaconda3\lib\site-packages\sklearn\linear_model\_logisti
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-r
  n_iter_i = _check_optimize_result(
       LogisticRegression fit : 6.167 ; Pred 0.337
--------------------------------------------------------------------------
```

Prediction and score on TRAIN dataset

```
          GaussianNB: 0.734
DecisionTreeClassifier: 0.888
       SGDClassifier: 0.447
KNeighborsClassifier: 0.851
  LogisticRegression: 0.765
```

Prediction and score on TEST dataset

```
          GaussianNB: 0.614
DecisionTreeClassifier: 0.820
       SGDClassifier: 0.450
KNeighborsClassifier: 0.773
  LogisticRegression: 0.765
```

REAL ERROR OF MODELS

Evaluation of the model is mostly using the following common metrics: R squared, **mean absolute error (MAE)**, and **root mean squared error (RMSD)**. The last two are measures of error between paired observations expressing the same phenomenon. Since errors can be both positive (actual > prediction) and negative (actual < prediction), you can measure the absolute value and the squared value of each error.

- **The error is model is mostly due to overfitting of the model since the Train error score is higher than the test score.**
- **This can be avoided by training the model with subsamples or removing some data that can be considered trivial from the original dataset.**
- **In this problem, since the data is labelled in a generalised way, it is quite difficult to conclude which is more imminent than the other.**

CLASSIFIER	GaussianNB	Descision Tree	SGD	KNN	Logistic Regression
Accuracy	**0.61**	**0.82**	**0.45**	**0.77**	**0.77**
Score	**0.614**	**0.82**	**0.45**	**0.773**	**0.765**
Explained Variance	**-0.65**	**0.23**	**-1.36**	**0.03**	**0**
Absolute mean error	**0.39**	**0.18**	**0.55**	**0.23**	**0.23**
Root mean squared error	**0.62**	**0.42**	**0.74**	**0.48**	**0.48**
Mean squared log error	**0.19**	**0.09**	**0.26**	**0.11**	**0.11**

```
GaussianNB()
Accuracy: 0.61
R2 (explained variance): -0.65
Mean Absolute Error (Σ|y-pred|/n): 0.39
Root Mean Squared Error (sqrt(Σ(y-pred)^2/n)): 0.62
Mean squared log error: 0.19
----------------------------------------------------------------------
DecisionTreeClassifier(max_depth=20)
Accuracy: 0.82
R2 (explained variance): 0.23
Mean Absolute Error (Σ|y-pred|/n): 0.18
Root Mean Squared Error (sqrt(Σ(y-pred)^2/n)): 0.42
Mean squared log error: 0.09
----------------------------------------------------------------------
SGDClassifier()
Accuracy: 0.45
R2 (explained variance): -1.36
Mean Absolute Error (Σ|y-pred|/n): 0.55
Root Mean Squared Error (sqrt(Σ(y-pred)^2/n)): 0.74
Mean squared log error: 0.26
----------------------------------------------------------------------
KNeighborsClassifier()
Accuracy: 0.77
R2 (explained variance): 0.03
Mean Absolute Error (Σ|y-pred|/n): 0.23
Root Mean Squared Error (sqrt(Σ(y-pred)^2/n)): 0.48
Mean squared log error: 0.11
----------------------------------------------------------------------
LogisticRegression()
Accuracy: 0.77
R2 (explained variance): -0.0
Mean Absolute Error (Σ|y-pred|/n): 0.23
Root Mean Squared Error (sqrt(Σ(y-pred)^2/n)): 0.48
Mean squared log error: 0.11
----------------------------------------------------------------------
```

MODEL COMPARATIVE ANALYSIS

- **Classification report of the models :** To do a comparative study of models used and measure their performance evaluation using the metrics precision, recall, F1 Score, and support.

GAUSSION NB

	precision	recall	f1-score	support
0	0.76	0.67	0.71	27854
1	0.37	0.48	0.42	11545
accuracy			0.61	39399
macro avg	0.56	0.57	0.56	39399
weighted avg	0.64	0.61	0.63	39399

DECISION TREE

	precision	recall	f1-score	support
0	0.91	0.83	0.86	27150
1	0.68	0.81	0.74	12249
accuracy			0.82	39399
macro avg	0.79	0.82	0.80	39399
weighted avg	0.83	0.82	0.82	39399

SGD

	precision	recall	f1-score	support
0	0.14	0.90	0.24	3843
1	0.97	0.40	0.57	35556
accuracy			0.45	39399
macro avg	0.56	0.65	0.40	39399
weighted avg	0.89	0.45	0.54	39399

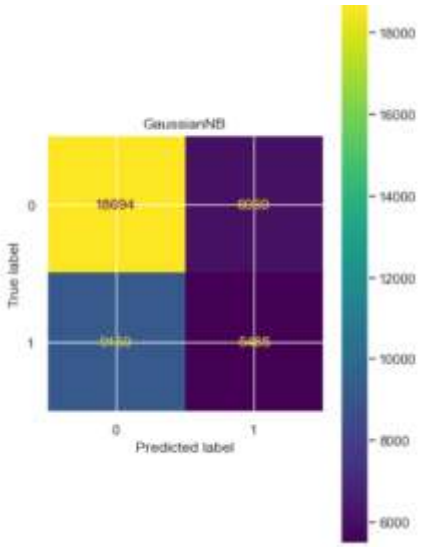
KNN

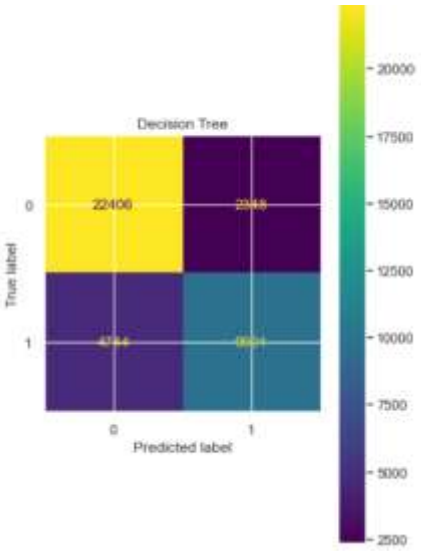
	precision	recall	f1-score	support
0	0.85	0.80	0.83	26333
1	0.64	0.72	0.68	13066
accuracy			0.77	39399
macro avg	0.75	0.76	0.75	39399
weighted avg	0.78	0.77	0.78	39399

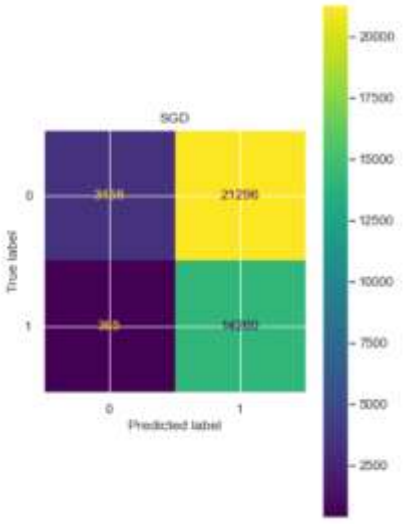
LOGISTIC REGRESSION

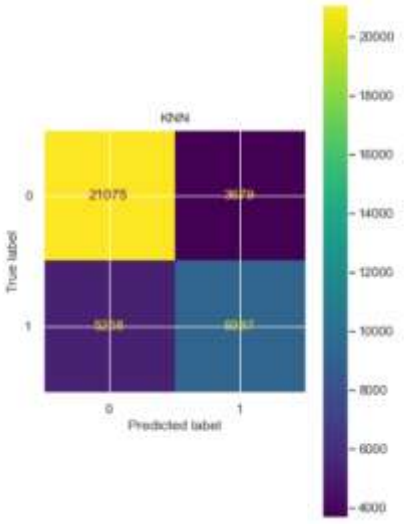
	precision	recall	f1-score	support
0	0.90	0.77	0.83	29087
1	0.54	0.76	0.63	10312
accuracy			0.77	39399
macro avg	0.72	0.76	0.73	39399
weighted avg	0.81	0.77	0.78	39399

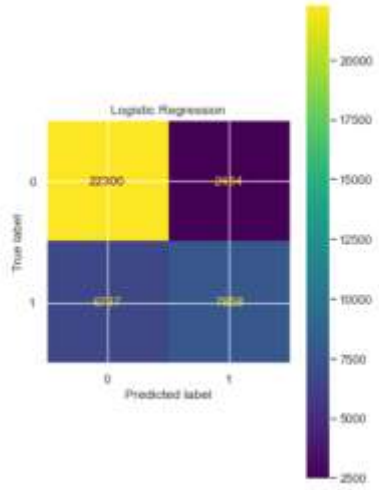
- **Confusion correlation Heatmap of the models**











HYPERPARAMETER SELECTION

- The best hyperparameter is found for Decision Tree classifier since it produced better results than other models.

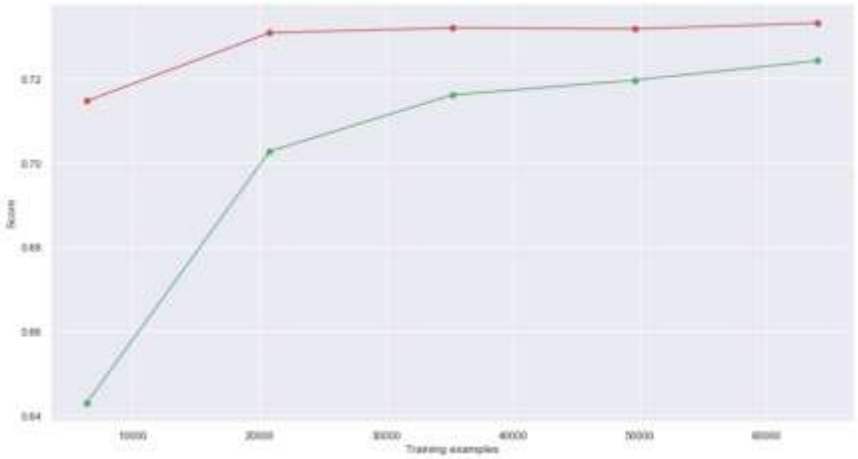
```
Best Criterion: gini
Best max_depth: 12
DecisionTreeClassifier(max_depth=12)
```

LEARNING CURVE

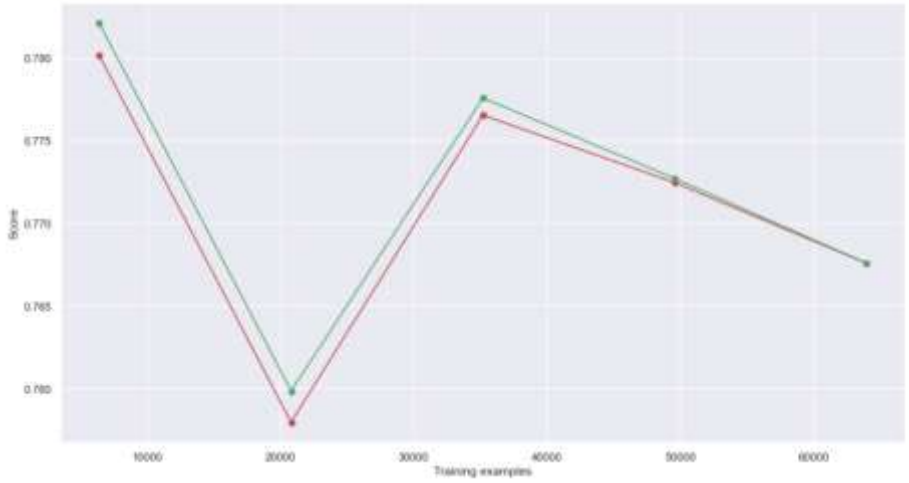
Learning curve is plotted for classifier.

- Estimator: In this we have to pass the models or functions on which we want to use GridSearchCV. We plotted it for GaussianNB and Logistic regression estimators
- Train_sizes: Relative or absolute numbers of training examples that will be used to generate the learning curve. We have set a limited value since it took a lot of processing time.
- Training accuracy is the accuracy score when the trained model is tested on the data it was trained on. Essentially it's tested on data it has already seen
- In cross validation the data is randomly split into training and testing sets. The model is trained on the training set, and tested on the testing set. The accuracy score is a reflection of how closely the testing set is predicted.
- n_jobs : This signifies the number of jobs to be run in parallel, and we have mentioned as 4

GAUSSIAN NB



LOGISTIC REGRESSION



- The lines coinciding might as well suggest that the model is trained well, it's just as good at predicting things it hasn't seen before as it is on things it was trained on.