CAPSTONE PROJECT INTERIM REPORT

Batch details	PGP – DSE ONLINE FEB 21 (CHENNAI)
Team members	DHANESH K MANOJ KUMARAN R ANIRUDH RAMACHANDRA SUDVEEP KUMAR REDDY VENKATRAMANA S NAGARAJU ARVETI
Domain of Project	FRAUD DETECTION
Proposed project title	CREDIT CARD FRAUD DETECTION
Group Number	4
Team Leader	DHANESH K
Mentor Name	MS ANJANA AGRAWAL

TABLE OF CONTENTS

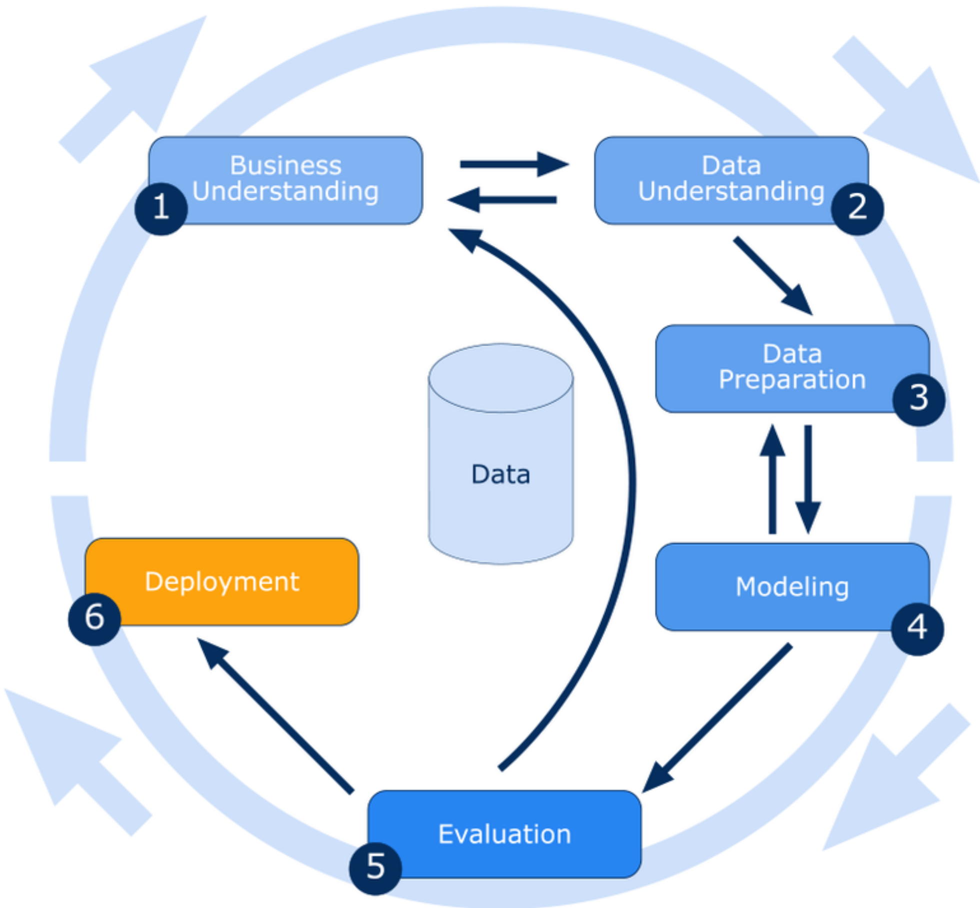
OVERVIEW

In today’ s world, high dependency on internet technology has enjoyed increased credit card transactions but, credit card fraud has also accelerated as online and offline transaction. As credit card transactions become a widespread mode of payment, focus has been given to recent computational methodologies to handle the credit card fraud problem.

Data mining technique is one notable and popular methods used in solving credit fraud detection problem. It is impossible to be sheer certain about the true intention and rightfulness behind an application or transaction. Credit card transaction datasets are rarely available, highly imbalanced and skewed. Optimal feature (variables) selection for the models, suitable metric is most important part of data mining to evaluate performance of techniques on skewed credit card fraud data.

From the experiments, the result that has been concluded is that Logistic regression has an accuracy of 0.99, KNN with 0.99. The boosting methods like Gradient boosting, AdaBoost and XGBoost shows an accuracy of 1.00.

METHODOLOGY



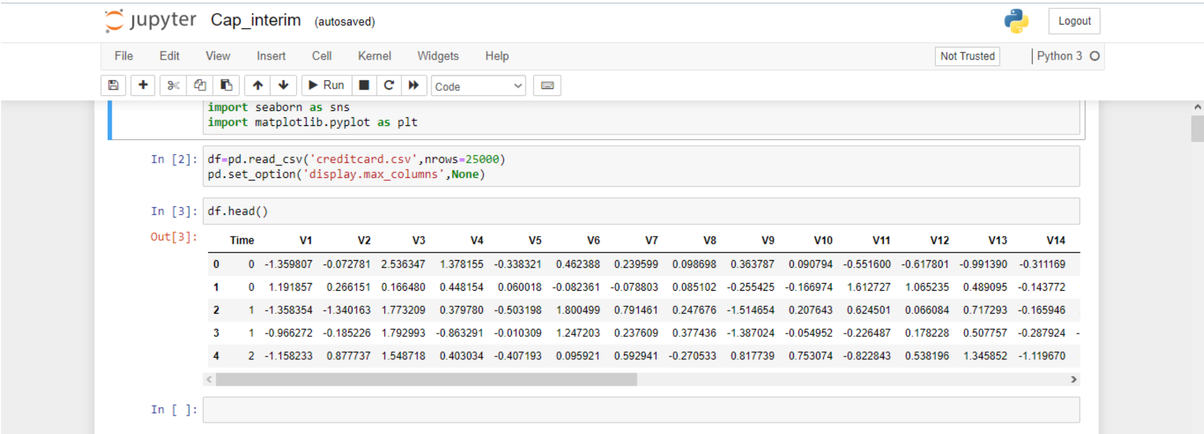
BUSINESS UNDERSTANDING

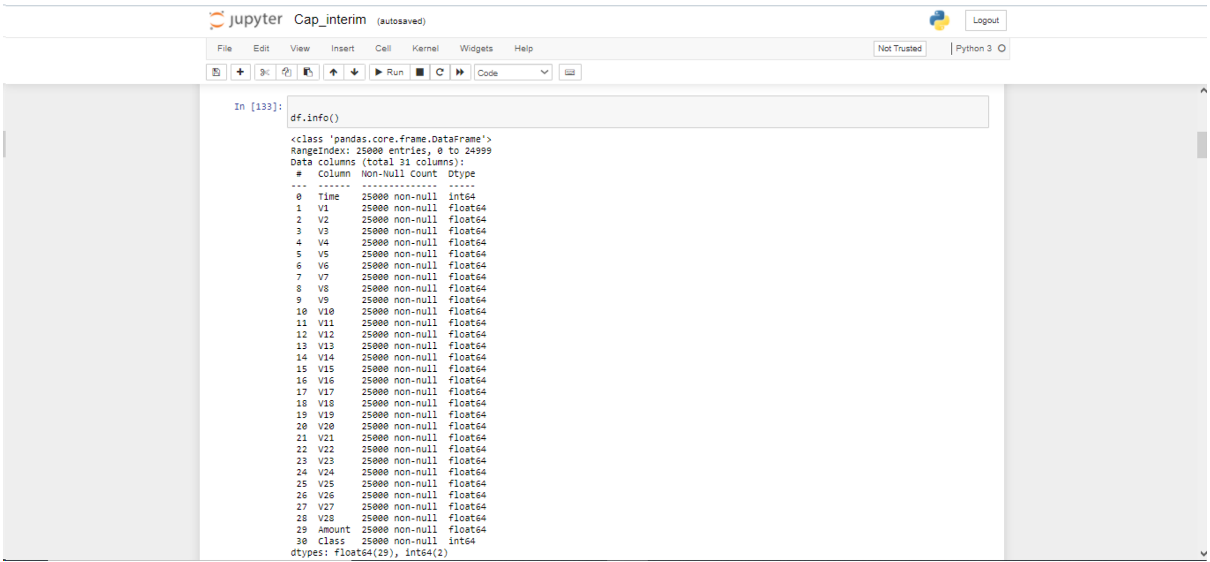
As online transactions increased over a period of time online fraud also subsequently increased. Although the proportion of fraud to nonfraud transactions is very low the chances of online fraud to grow is increasing gradually.

The goal is to identify and reduce the online fraud transactions without disturbing the flow of legal transactions.

By reducing the number of frauds, this will increase the trust of the customers in approaching the bank for the future.

DATA UNDERSTANDING





- The dataset contains 284807 rows and 31 columns.
- There is no categorical columns present in the dataset as it contains only numerical columns.

DATA PREPARATION

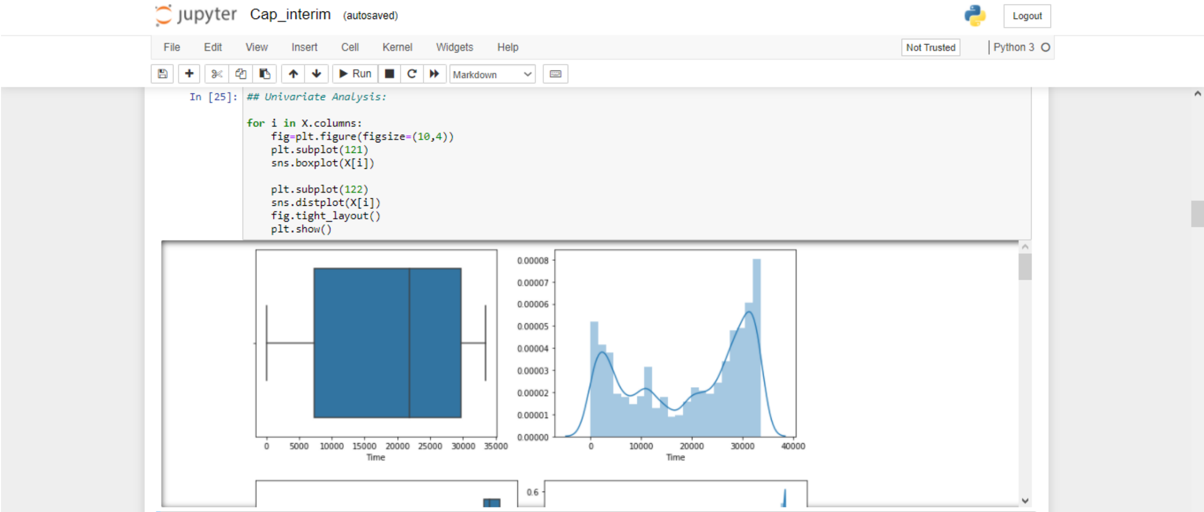
No of rows / columns	284807 rows / 31 columns	
Missing values	No missing values	
Categorical features	None	
Skewness value (numerical data)	V1 = -3.280667	V10 = 1.187141
	V2 = -4.624866	V12 = -2.278401
	V3 = -2.240155	V14 = -1.995176
	V5 = -2.425901	V16 = -1.100966
	V6 = 1.826581	V17 = -3.844914
	V7 = 2.553907	V28 = 11.192091
	V8 = -8.521944	Amount = 16.977724

MISSING VALUES

```
In [5]: df.isnull().sum()

Out[5]: Time      0
        V1        0
        V2        0
        V3        0
        V4        0
        V5        0
        V6        0
        V7        0
        V8        0
        V9        0
        V10       0
        V11       0
        V12       0
        V13       0
        V14       0
        V15       0
        V16       0
        V17       0
        V18       0
        V19       0
        V20       0
        V21       0
        V22       0
        V23       0
        V24       0
        V25       0
        V26       0
        V27       0
        V28       0
        Amount    0
        Class     0
```

SKEWNESS

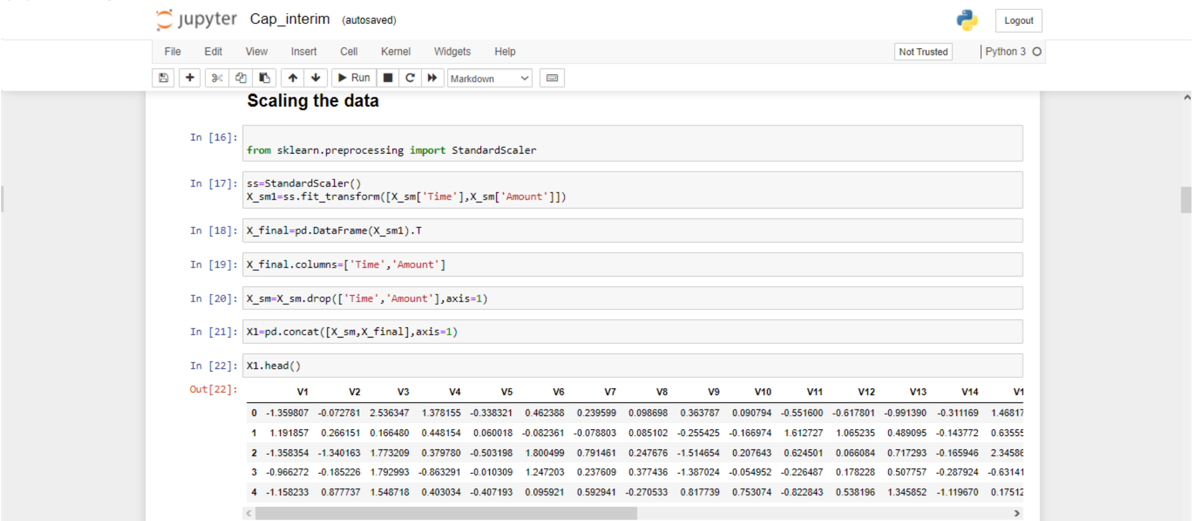
- Column time does not have outliers in it.
- Other than Time all other columns have extreme values or outliers which can't be denoted due to PCA.
- So, a distplot is also used to see the distribution and check skewness.
- Columns V1, V2, V3, V5, V6, V7, V8, V12, V14, V16, V17, V20, V21, V23, V27, V28, Amount all seem to be skewed.
- So, the skewness has to be removed from these columns.



OUTLIER TREATMENT

- Outlier treatment can't be done as this is a dataset which has undergone Principal Component Analysis.
- Capping method using IQR is not preferred because the information will be lost.
- Transformation techniques like log, sqrt, exponential, box-cox won't work as there are values that are very close to zero and negative values as well.
- So, it is better not to do outlier treatment here.

SCALING



MODELLING

SIGNIFICANT VARIABLES

Out[76]:

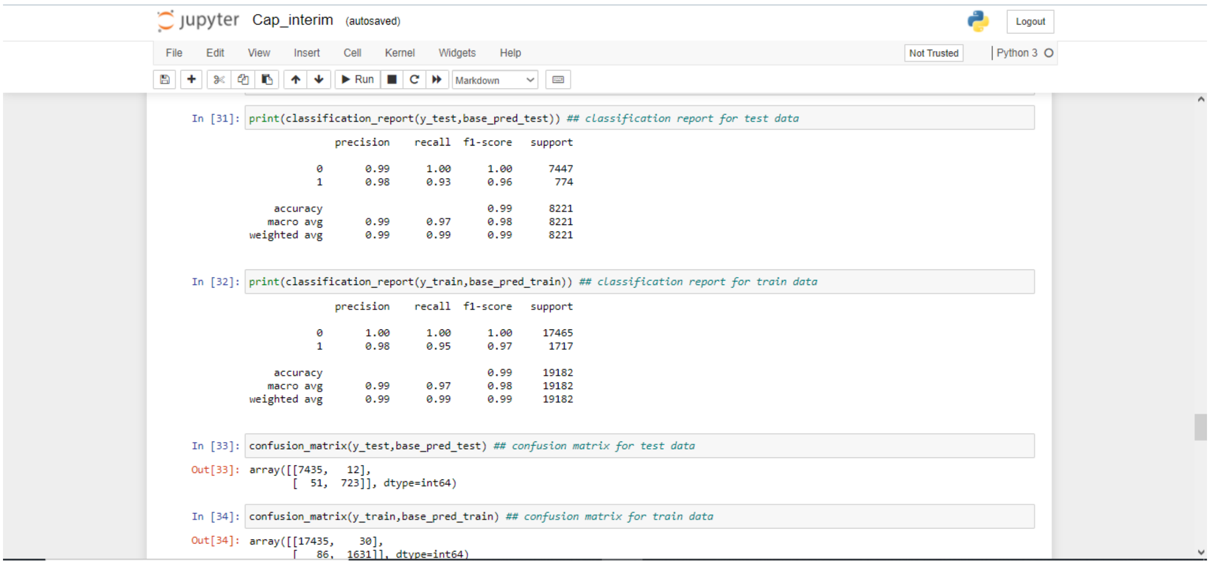
Logit Regression Results

Dep. Variable:	Class	**No. Observations:**	284807
Model:	Logit	**Df Residuals:**	284776
Method:	MLE	**Df Model:**	30
Date:	Thu, 09 Sep 2021	**Pseudo R-squ.:**	0.6922
Time:	22:57:58	**Log-Likelihood:**	-1114.8
converged:	True	**LL-Null:**	-3621.2
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-8.3917	0.249	-33.652	0.000	-8.880	-7.903
Time	-3.742e-06	2.26e-06	-1.659	0.097	-8.16e-06	6.79e-07
V1	0.0960	0.042	2.264	0.024	0.013	0.179
V2	0.0094	0.058	0.161	0.872	-0.104	0.123
V3	-0.0079	0.053	-0.149	0.881	-0.112	0.096
V4	0.6986	0.074	9.454	0.000	0.554	0.843
V5	0.1295	0.067	1.944	0.052	-0.001	0.260
V6	-0.1198	0.074	-1.626	0.104	-0.264	0.025
V7	-0.0969	0.067	-1.453	0.146	-0.228	0.034
V8	-0.1739	0.030	-5.711	0.000	-0.234	-0.114
V9	-0.2843	0.111	-2.561	0.010	-0.502	-0.067
V10	-0.8176	0.097	-8.432	0.000	-1.008	-0.628
V11	-0.0621	0.081	-0.762	0.446	-0.222	0.098
V12	0.0909	0.087	1.045	0.296	-0.080	0.261
V13	-0.3312	0.082	-4.058	0.000	-0.491	-0.171
V14	-0.5571	0.062	-8.949	0.000	-0.679	-0.435
V15	-0.1141	0.086	-1.330	0.183	-0.282	0.054
V16	-0.1908	0.125	-1.526	0.127	-0.436	0.054
V17	-0.0216	0.070	-0.309	0.757	-0.159	0.116
V18	-0.0131	0.129	-0.102	0.919	-0.266	0.240
V19	0.0963	0.097	0.993	0.321	-0.094	0.286
V20	-0.4582	0.082	-5.607	0.000	-0.618	-0.298
V21	0.3898	0.060	6.494	0.000	0.272	0.507
V22	0.6297	0.134	4.707	0.000	0.367	0.892
V23	-0.0951	0.058	-1.629	0.103	-0.209	0.019
V24	0.1289	0.147	0.874	0.382	-0.160	0.418
V25	-0.0761	0.131	-0.582	0.560	-0.332	0.180
V26	0.0195	0.190	0.103	0.918	-0.352	0.392
V27	-0.8188	0.122	-6.686	0.000	-1.059	-0.579
V28	-0.2937	0.088	-3.332	0.001	-0.467	-0.121
Amount	0.0009	0.000	2.449	0.014	0.000	0.002

- For a column to be considered significant statistically, the pvalue should be less than 0.05.
- Else it is insignificant.
- So, the values considered significant are v1, v4, v5, v8, v9, v10, v20, v21, v22, v27, v28, Amount.

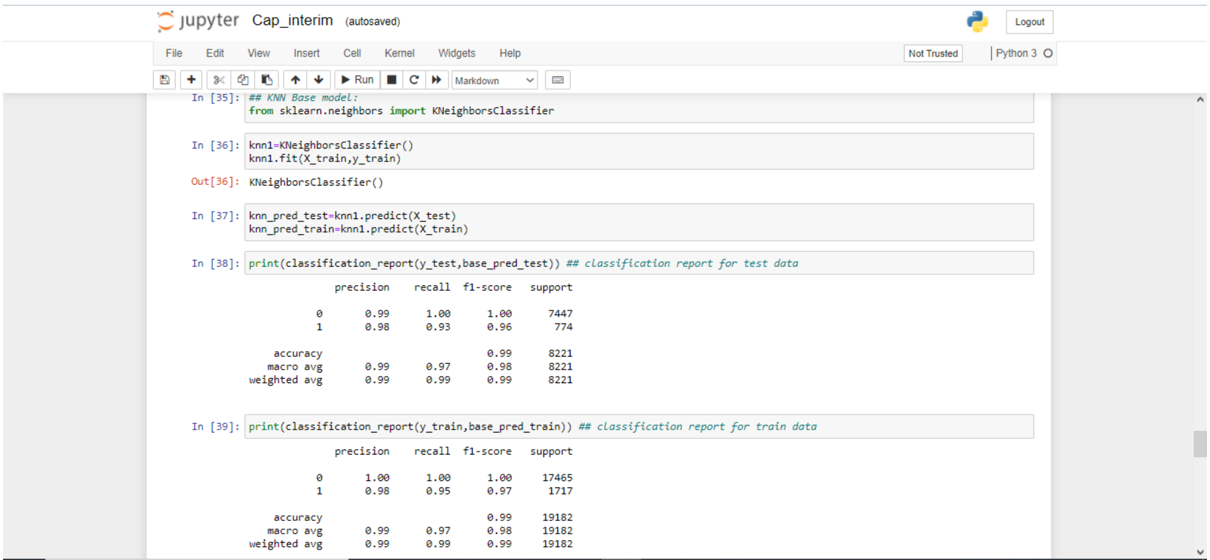
BASE MODEL



```
In [31]: print(classification_report(y_test,base_pred_test)) ## classification report for test data

              precision    recall  f1-score   support

           0       0.99      1.00      1.00      7447
           1       0.98      0.93      0.96       774

    accuracy                           0.99      8221
   macro avg       0.99      0.97      0.98      8221
weighted avg       0.99      0.99      0.99      8221

In [32]: print(classification_report(y_train,base_pred_train)) ## classification report for train data

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     17465
           1       0.98      0.95      0.97      1717

    accuracy                           0.99     19182
   macro avg       0.99      0.97      0.98     19182
weighted avg       0.99      0.99      0.99     19182

In [33]: confusion_matrix(y_test,base_pred_test) ## confusion matrix for test data
Out[33]: array([[7435,   12],
                [  51,  723]], dtype=int64)

In [34]: confusion_matrix(y_train,base_pred_train) ## confusion matrix for train data
Out[34]: array([[17435,    30],
                [   86,  1631]], dtype=int64)
```

The classification report shows the accuracy as 0.99 for the test data and also for the train data the accuracy is 0.99.

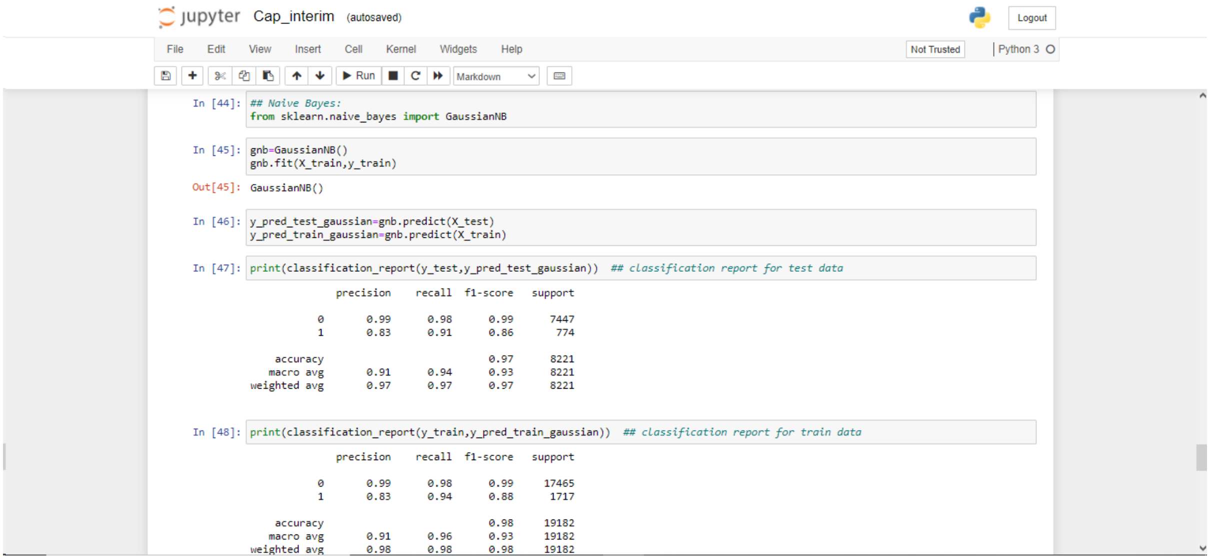
KNN MODEL



```
In [35]: ## KNN Base model:
         from sklearn.neighbors import KNeighborsClassifier

In [36]: knn1=KNeighborsClassifier()
         knn1.fit(X_train,y_train)
Out[36]: KNeighborsClassifier()

In [37]: knn_pred_test=knn1.predict(X_test)
         knn_pred_train=knn1.predict(X_train)

In [38]: print(classification_report(y_test,base_pred_test)) ## classification report for test data

              precision    recall  f1-score   support

           0       0.99      1.00      1.00      7447
           1       0.98      0.93      0.96       774

    accuracy                           0.99      8221
   macro avg       0.99      0.97      0.98      8221
weighted avg       0.99      0.99      0.99      8221

In [39]: print(classification_report(y_train,base_pred_train)) ## classification report for train data

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     17465
           1       0.98      0.95      0.97      1717

    accuracy                           0.99     19182
   macro avg       0.99      0.97      0.98     19182
weighted avg       0.99      0.99      0.99     19182
```

For the KNN model, the accuracy for the test data is showing as 0.99 and for the train data it is showing as 0.99.

NAÏVE BAYES MODEL

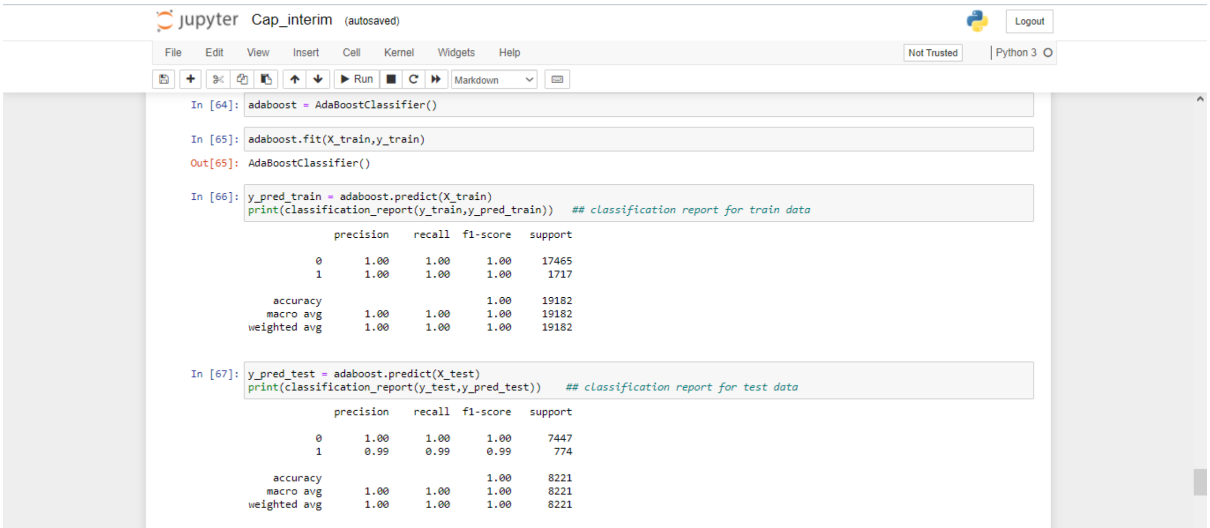


jupyter Cap_interim (autosaved)

```
In [44]: ## Naive Bayes:
         from sklearn.naive_bayes import GaussianNB

In [45]: gnb=GaussianNB()
         gnb.fit(X_train,y_train)

Out[45]: GaussianNB()

In [46]: y_pred_test_gaussian=gnb.predict(X_test)
         y_pred_train_gaussian=gnb.predict(X_train)

In [47]: print(classification_report(y_test,y_pred_test_gaussian))  ## classification report for test data

              precision    recall  f1-score   support

           0       0.99      0.98      0.99      7447
           1       0.83      0.91      0.86       774

    accuracy                           0.97      8221
   macro avg       0.91      0.94      0.93      8221
weighted avg       0.97      0.97      0.97      8221

In [48]: print(classification_report(y_train,y_pred_train_gaussian))  ## classification report for train data

              precision    recall  f1-score   support

           0       0.99      0.98      0.99     17465
           1       0.83      0.94      0.88      1717

    accuracy                           0.98     19182
   macro avg       0.91      0.96      0.93     19182
weighted avg       0.98      0.98      0.98     19182
```

For the Naïve Bayes model, the accuracy for the test data is showing as 0.97 and for the train data it is showing as 0.98.

AdaBoost MODEL

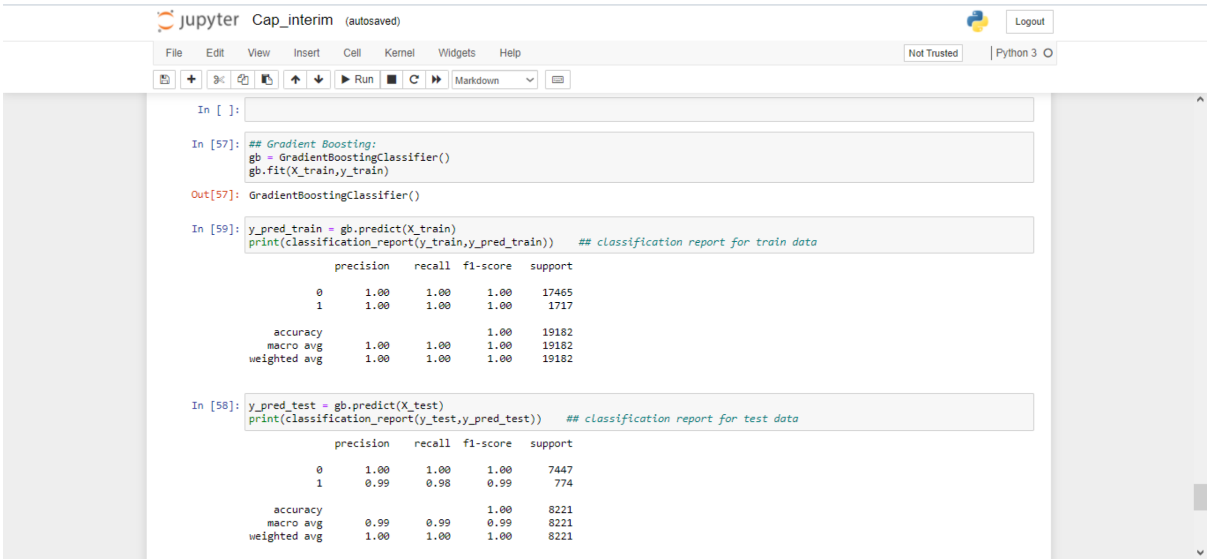


jupyter Cap_interim (autosaved)

```
In [64]: adaboost = AdaBoostClassifier()

In [65]: adaboost.fit(X_train,y_train)

Out[65]: AdaBoostClassifier()

In [66]: y_pred_train = adaboost.predict(X_train)
         print(classification_report(y_train,y_pred_train))   ## classification report for train data

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     17465
           1       1.00      1.00      1.00      1717

    accuracy                           1.00     19182
   macro avg       1.00      1.00      1.00     19182
weighted avg       1.00      1.00      1.00     19182

In [67]: y_pred_test = adaboost.predict(X_test)
         print(classification_report(y_test,y_pred_test))    ## classification report for test data

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7447
           1       0.99      0.99      0.99       774

    accuracy                           1.00      8221
   macro avg       1.00      1.00      1.00      8221
weighted avg       1.00      1.00      1.00      8221
```

For the Ada Boost model, the accuracy for the test data is showing as 1.00 and for the train data it is showing as 1.00.

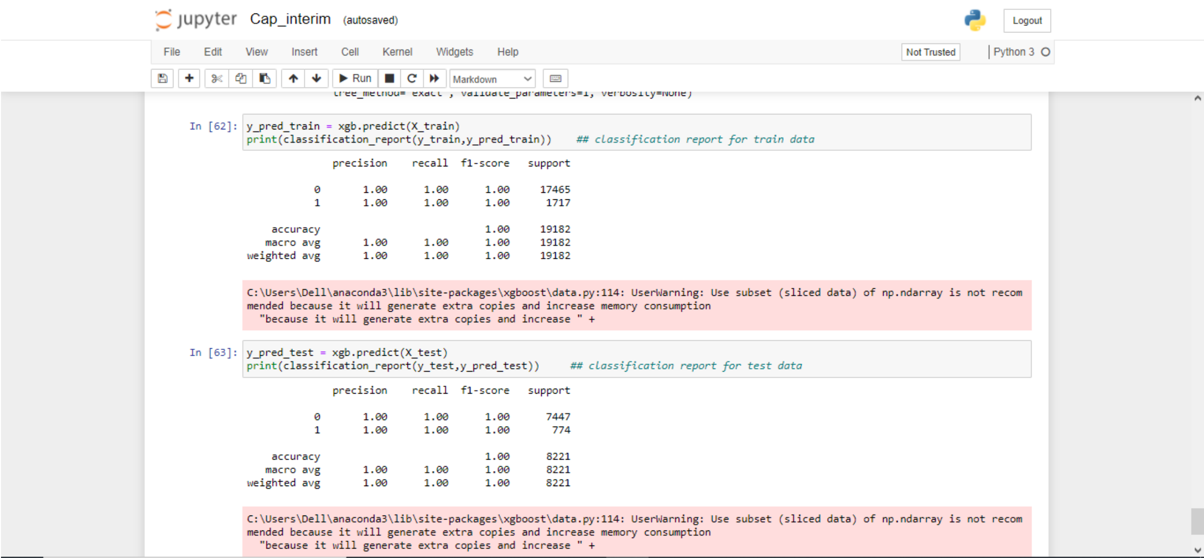
GRADIENT BOOSTING MODEL



```
jupyter Cap_interim (autosaved)                                    Logout
File  Edit  View  Insert  Cell  Kernel  Widgets  Help    Not Trusted | Python 3

In [ ]:

In [57]: ## Gradient Boosting:
         gb = GradientBoostingClassifier()
         gb.fit(X_train,y_train)

Out[57]: GradientBoostingClassifier()

In [59]: y_pred_train = gb.predict(X_train)
         print(classification_report(y_train,y_pred_train))    ## classification report for train data

                      precision    recall  f1-score   support

                   0       1.00      1.00      1.00     17465
                   1       1.00      1.00      1.00      1717

            accuracy                           1.00     19182
           macro avg       1.00      1.00      1.00     19182
        weighted avg       1.00      1.00      1.00     19182

In [58]: y_pred_test = gb.predict(X_test)
         print(classification_report(y_test,y_pred_test))    ## classification report for test data

                      precision    recall  f1-score   support

                   0       1.00      1.00      1.00      7447
                   1       0.99      0.98      0.99       774

            accuracy                           1.00      8221
           macro avg       0.99      0.99      0.99      8221
        weighted avg       1.00      1.00      1.00      8221
```

For the Gradient Boosting model, the accuracy for the test data is showing as 1.00 and for the train data it is showing as 1.00.

XG BOOST MODEL



```
jupyter Cap_interim (autosaved)                                    Logout
File  Edit  View  Insert  Cell  Kernel  Widgets  Help    Not Trusted | Python 3

              tree_method='exact', validate_parameters=1, verbosity=None)

In [62]: y_pred_train = xgb.predict(X_train)
         print(classification_report(y_train,y_pred_train))    ## classification report for train data

                      precision    recall  f1-score   support

                   0       1.00      1.00      1.00     17465
                   1       1.00      1.00      1.00      1717

            accuracy                           1.00     19182
           macro avg       1.00      1.00      1.00     19182
        weighted avg       1.00      1.00      1.00     19182

C:\Users\Dell\anaconda3\lib\site-packages\xgboost\data.py:114: UserWarning: Use subset (sliced data) of np.ndarray is not recom
mended because it will generate extra copies and increase memory consumption
  "because it will generate extra copies and increase " +

In [63]: y_pred_test = xgb.predict(X_test)
         print(classification_report(y_test,y_pred_test))    ## classification report for test data

                      precision    recall  f1-score   support

                   0       1.00      1.00      1.00      7447
                   1       1.00      1.00      1.00       774

            accuracy                           1.00      8221
           macro avg       1.00      1.00      1.00      8221
        weighted avg       1.00      1.00      1.00      8221

C:\Users\Dell\anaconda3\lib\site-packages\xgboost\data.py:114: UserWarning: Use subset (sliced data) of np.ndarray is not recom
mended because it will generate extra copies and increase memory consumption
  "because it will generate extra copies and increase " +
```

For the XG Boost model, the accuracy for the test data is showing as 1.00 and for the train data it is showing as 1.00.

REFERENCE

The references used in the project are given below:

1.) Calibrating Probability with Under sampling for Unbalanced Classification by Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi.

2.) Learned lessons in credit card fraud detection from a practitioner perspective by Dal Pozzolo, Andrea; Caelen, Olivier; Le Borgne, Yann-Ael; Waterschoot, Serge; Bontempi, Gianluca.

3.) Combining Unsupervised and Supervised Learning in Credit Card Fraud Detection by Fabrizio Carcillo, Yann-Aël Le Borgne, Olivier Caelen, Frederic Oblé, Gianluca Bontempi.