NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

SPRING 2025

MACHINE LEARNING

ASSIGNMENT – 01

Name: Wania Naeem

Roll No: 22I-2369

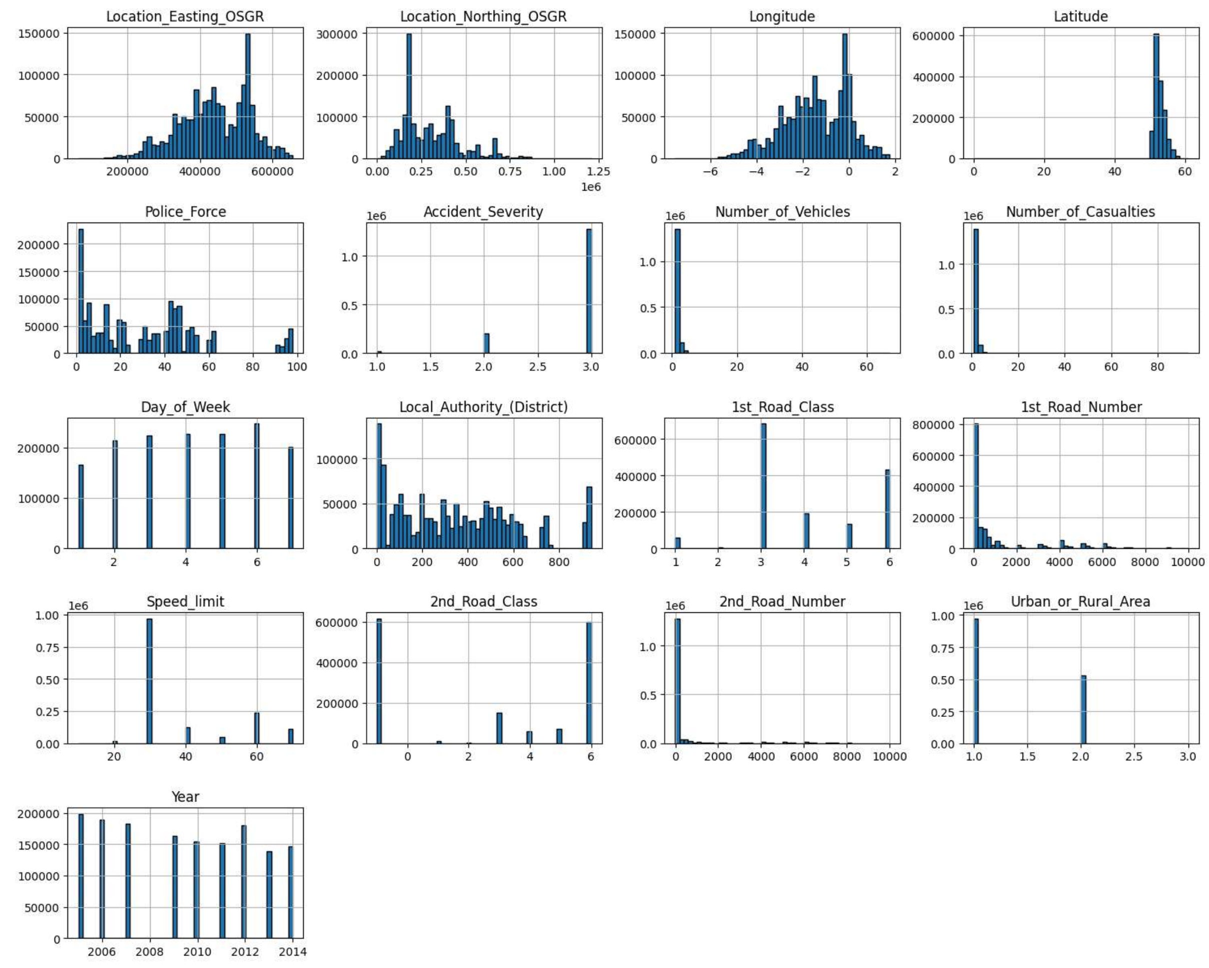
Section: CS-Z

Task – 1 (Dataset Selection and Understanding):

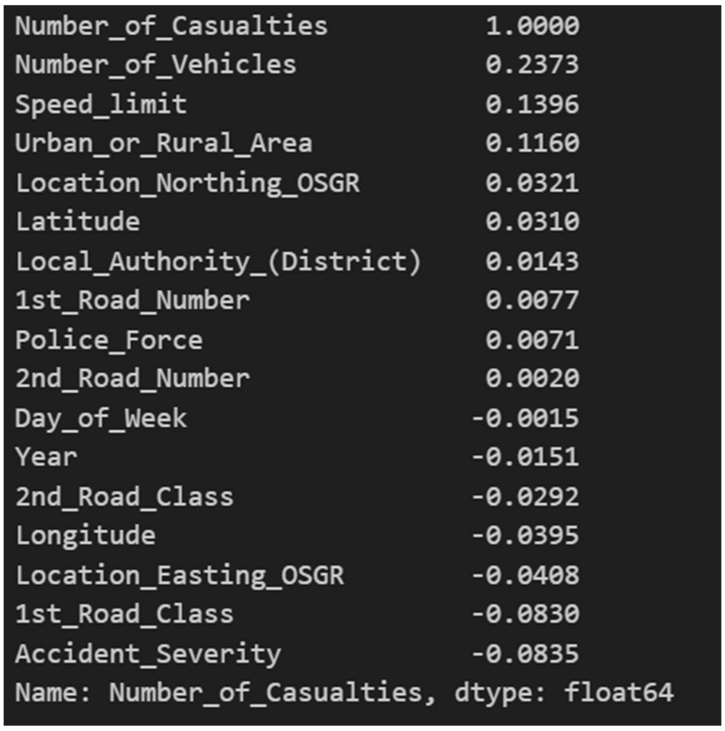
The dataset I choose, is about the Road Accidents in UK from 2000 to 2018 from Kaggle. It has 1.8 million records with 33 features. The domain of this dataset is about safety of transportation. The target variable is the 'Accident_Severity', which is a categorical feature, predicting 3 classes for the severity of accidents. The problem I am going to solve is a Classification problem, which is predicting the severity of the accident.

Task – 2 (Exploratory Data Analysis (EDA)):

For the numerical features, I generated a histogram which shows the trend in data. One thing to keep in mind, this dataset have some features that are categorical in nature, but are stored in int64, so they are displayed in the histograms as well. Such features are later catered in the processing part.

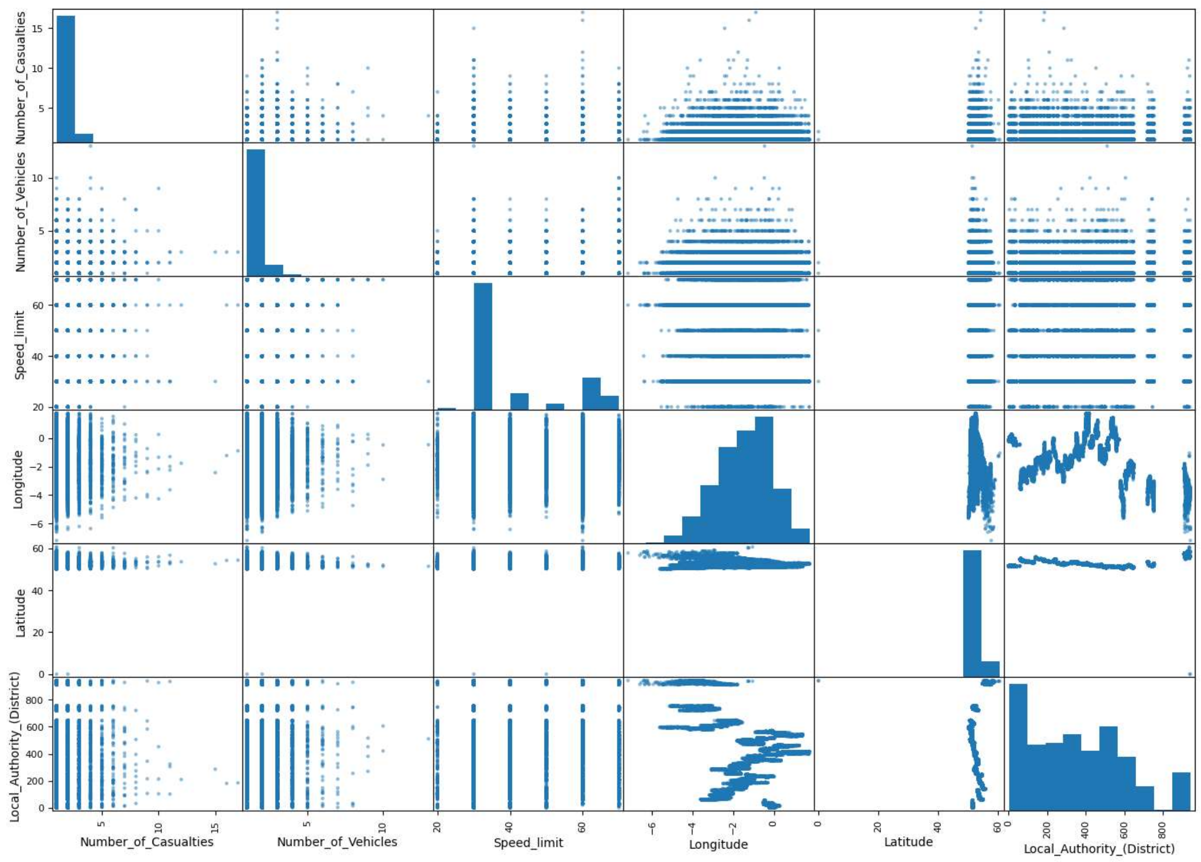


Now the correlation matrix was created using a numeric feature ‘Number_of_Casualties’, and these were the results.

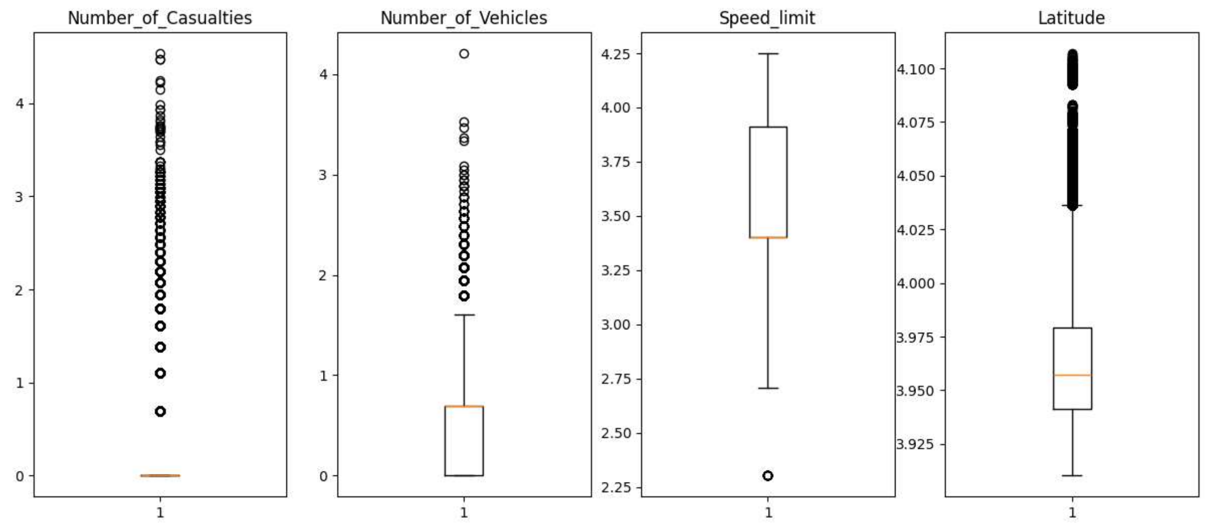


```
Number_of_Casualties          1.0000
Number_of_Vehicles            0.2373
Speed_limit                   0.1396
Urban_or_Rural_Area           0.1160
Location_Northing_OSGR        0.0321
Latitude                      0.0310
Local_Authority_(District)    0.0143
1st_Road_Number               0.0077
Police_Force                  0.0071
2nd_Road_Number               0.0020
Day_of_Week                  -0.0015
Year                         -0.0151
2nd_Road_Class               -0.0292
Longitude                    -0.0395
Location_Easting_OSGR        -0.0408
1st_Road_Class               -0.0830
Accident_Severity            -0.0835
Name: Number_of_Casualties, dtype: float64
```

Scatter plots were also made, but this led a shocking discovery, that some of the prominent features to use, majorly consisted of discrete-like values. Hence some of the plots are dotted, straight lines etc.



Box plots were also created to identify outliers, and there have been many extreme outliers in the dataset. The most important features considered are Number_of_Casualties, Number_of_Vehicles, Speed_limit and Latitude.



Task – 3 (Date Preprocessing and Feature Engineering):

Now there were some null values in the dataset as well, specifically in 5 features: Location_Easting_OSGR, Longitude, Time, Junction_Control and Pedestrian_Crossing-Physical_Facilites.

Now using the pipeline from sklearn and Column Transformer, the null values were imputing in the features. The numerical features had median values imputed while the categorical features had most frequent values imputed.

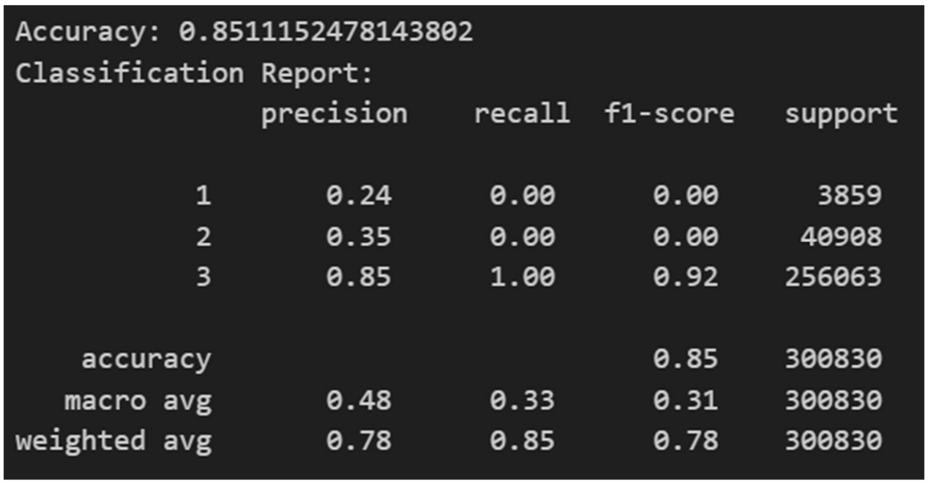
OneHotEncoder was used to convert the categorical features into numerical and also the supposedly numerical features which were meant to be categorical (Day_of_Week, Accident_Severity, Year etc).

The numerical features were also standardized using the StandardScaler to make sure that their mean values were 1 and std. deviation is 0. This makes its easier for model to create relationships between features.

Task – 4 (Model Selection and Training):

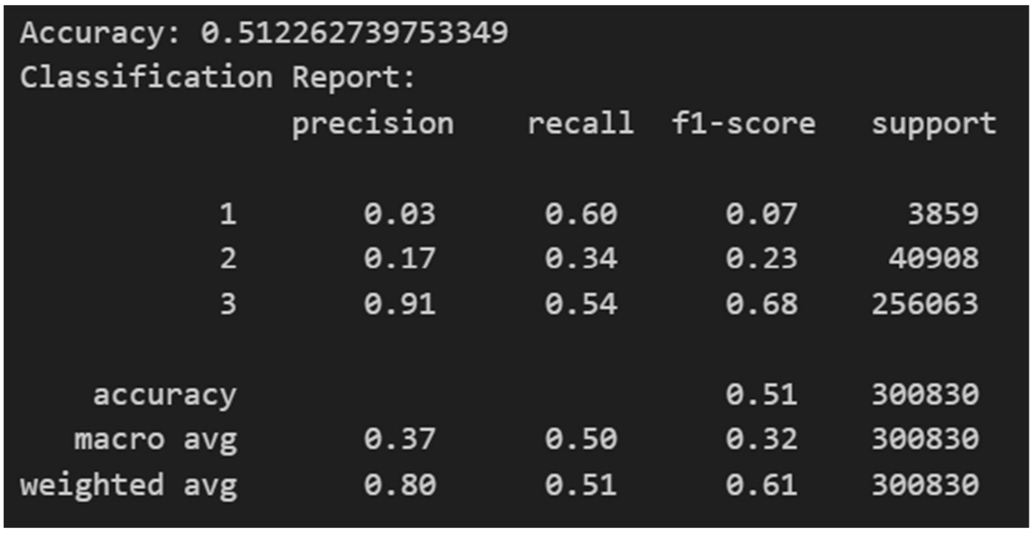
I choose 3 different models to train my dataset on: Logistic Regression, Random Forest, Gradient Boosting. Since I have a classification problem, so I displayed accuracy, recall, precision and F1-score.

1. Model 1: Logistic Regression



```
Accuracy: 0.8511152478143802
Classification Report:
              precision    recall  f1-score   support

           1       0.24      0.00      0.00      3859
           2       0.35      0.00      0.00     40908
           3       0.85      1.00      0.92    256063

    accuracy                           0.85    300830
   macro avg       0.48      0.33      0.31    300830
weighted avg       0.78      0.85      0.78    300830
```

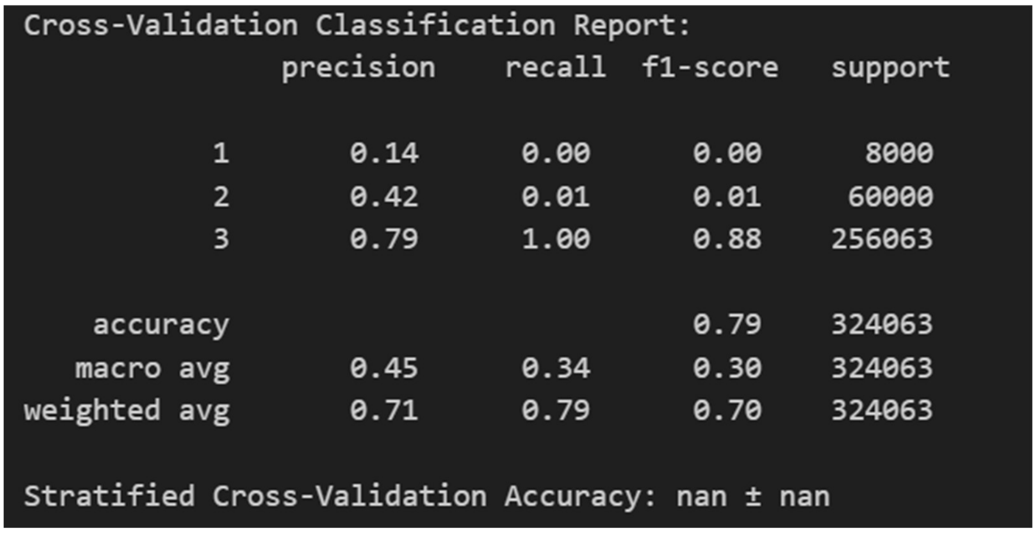
Now from the dataset, it became evident, that the data is highly biased towards one class (class 3), so because of this precision and recall were not balanced. To counter this, I used class_balancing and SMOTE, which generates more samples of the less represented classes. The below report is the improved one.



```
Accuracy: 0.512262739753349
Classification Report:
              precision    recall  f1-score   support

           1       0.03      0.60      0.07      3859
           2       0.17      0.34      0.23     40908
           3       0.91      0.54      0.68    256063

    accuracy                           0.51    300830
   macro avg       0.37      0.50      0.32    300830
weighted avg       0.80      0.51      0.61    300830
```

However, the accuracy decreased majorly while trying to improve the precision and recall.

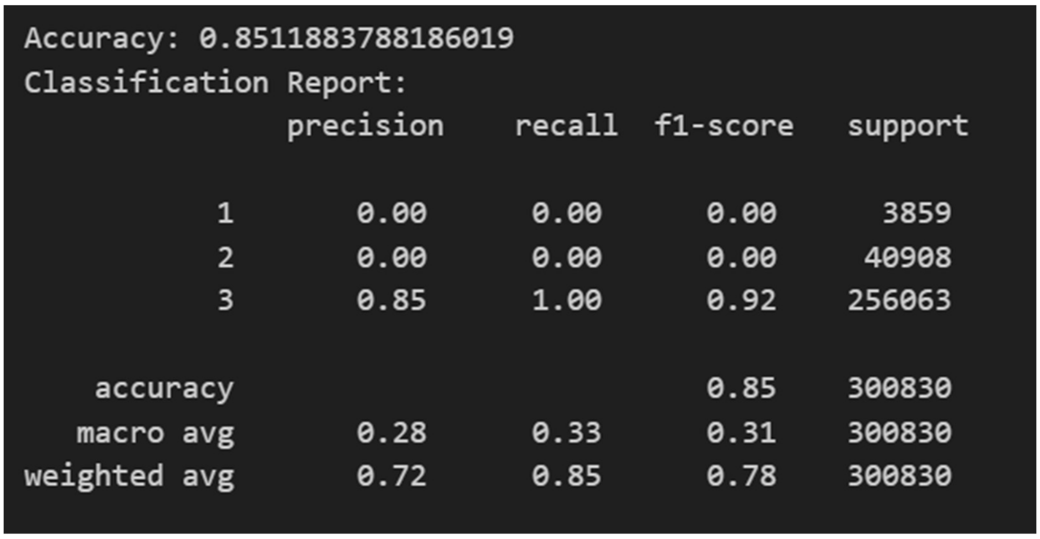
Now using Cross-Validation on Logistic Regression:



```
Cross-Validation Classification Report:
              precision    recall  f1-score   support

           1       0.14      0.00      0.00      8000
           2       0.42      0.01      0.01     60000
           3       0.79      1.00      0.88    256063

    accuracy                           0.79    324063
   macro avg       0.45      0.34      0.30    324063
weighted avg       0.71      0.79      0.70    324063

Stratified Cross-Validation Accuracy: nan ± nan
```

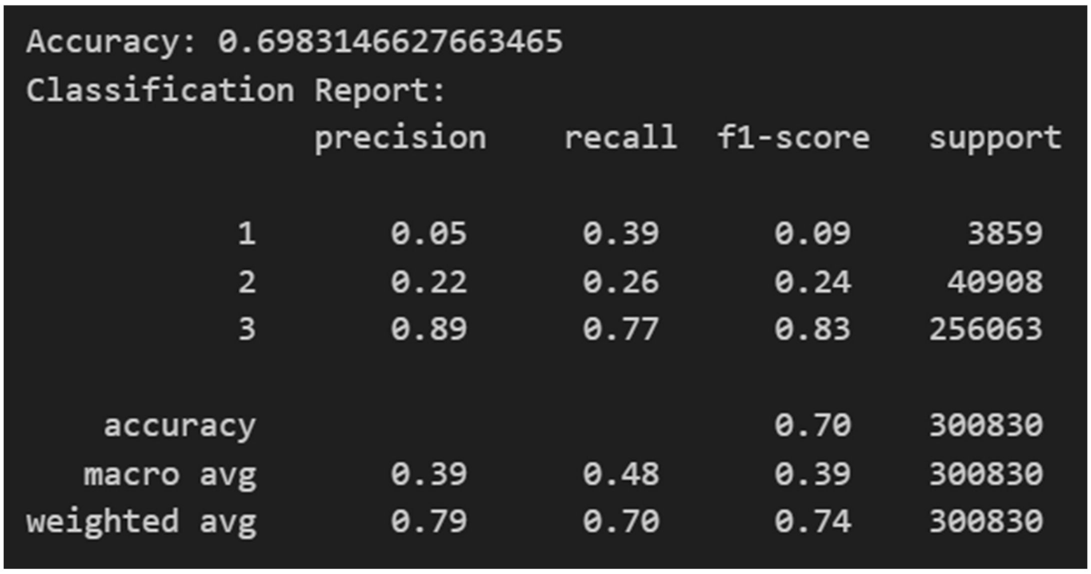
The accuracy could not be determined, due to huge class imbalance, resulting in nan values for the accuracy, and poor recall.

2. Model 2: Random Forest



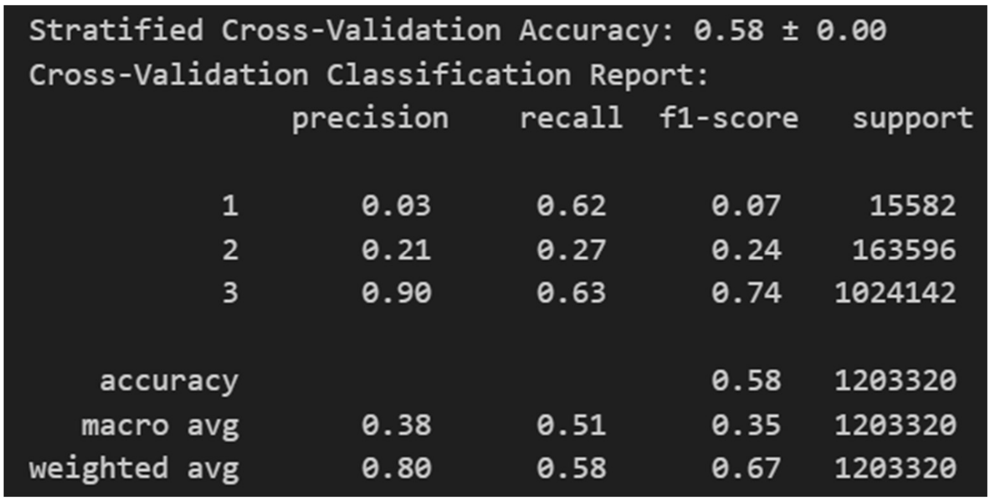
```
Accuracy: 0.8511883788186019
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00      3859
           2       0.00      0.00      0.00     40908
           3       0.85      1.00      0.92    256063

    accuracy                           0.85    300830
   macro avg       0.28      0.33      0.31    300830
weighted avg       0.72      0.85      0.78    300830
```

Again same as the logistic regression, only 1 class was detected, while the others weren't able to be generalized by the model. So applying SMOTE and class balancing on this.



```
Accuracy: 0.6983146627663465
Classification Report:
              precision    recall  f1-score   support

           1       0.05      0.39      0.09      3859
           2       0.22      0.26      0.24     40908
           3       0.89      0.77      0.83    256063

    accuracy                           0.70    300830
   macro avg       0.39      0.48      0.39    300830
weighted avg       0.79      0.70      0.74    300830
```

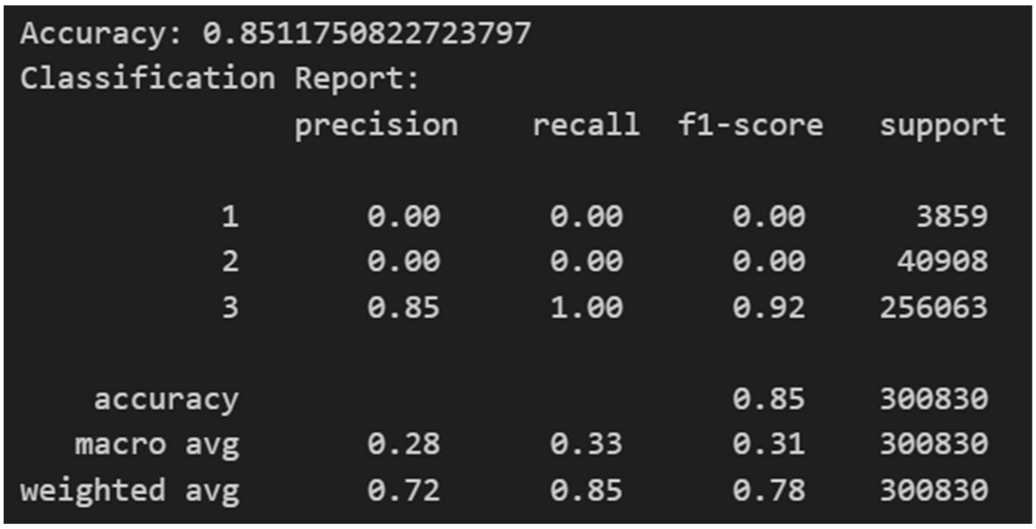
The accuracy decreased a bit, but not as much as logistic regression. The precision and recall also improved significantly. Now if I were to use stratified cross-validation here, the report would be like this:


```
Stratified Cross-Validation Accuracy: 0.58 ± 0.00
Cross-Validation Classification Report:
              precision    recall  f1-score   support

           1       0.03      0.62      0.07     15582
           2       0.21      0.27      0.24    163596
           3       0.90      0.63      0.74   1024142

    accuracy                           0.58   1203320
   macro avg       0.38      0.51      0.35   1203320
weighted avg       0.80      0.58      0.67   1203320
```

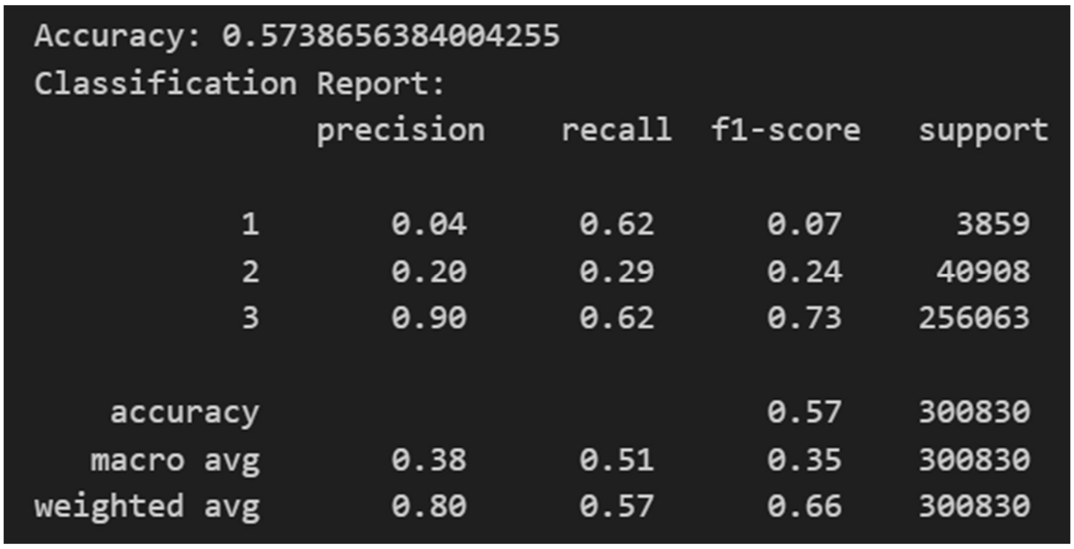
The accuracy of the model decreased majorly, but the precision and recall have improved significantly.

3. Model 3: Histogram Gradient Boosting

GradientBoostingClassifier was used before, but it was then replaced by HistGradientBoostingClassifier, because the former had a very large execution time, and caused vs code to crash every time the cells were run. The latter is designed for large datasets, so this was preferred.

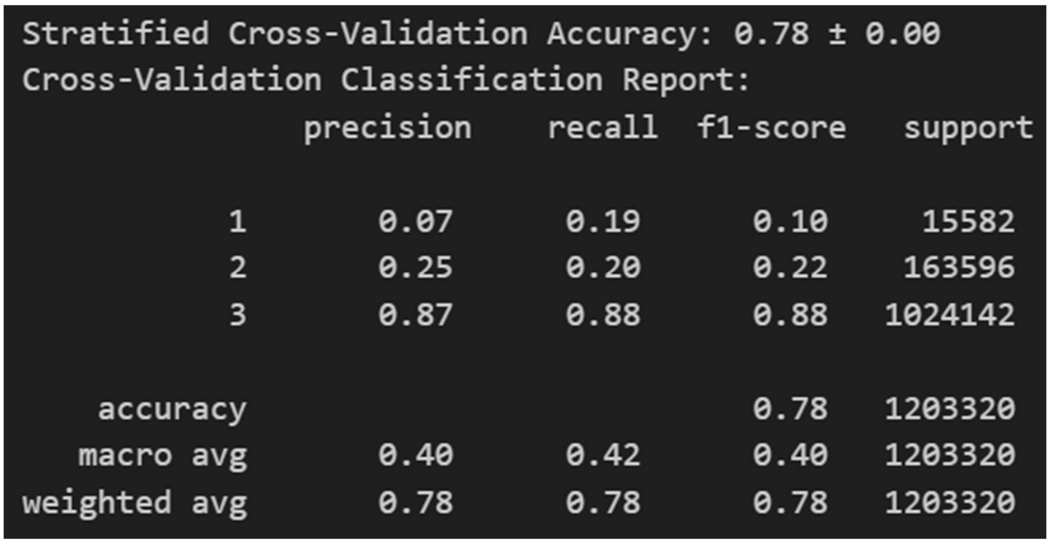

```
Accuracy: 0.8511750822723797
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00      3859
           2       0.00      0.00      0.00     40908
           3       0.85      1.00      0.92    256063

    accuracy                           0.85    300830
   macro avg       0.28      0.33      0.31    300830
weighted avg       0.72      0.85      0.78    300830
```

The same results are shown as it was shown in logistic regression and random forest, Applying SMOTE and class balancing here, gives:



```
Accuracy: 0.5738656384004255
Classification Report:
              precision    recall  f1-score   support

           1       0.04      0.62      0.07      3859
           2       0.20      0.29      0.24     40908
           3       0.90      0.62      0.73    256063

    accuracy                           0.57    300830
   macro avg       0.38      0.51      0.35    300830
weighted avg       0.80      0.57      0.66    300830
```

Now using stratified cross-validation here:



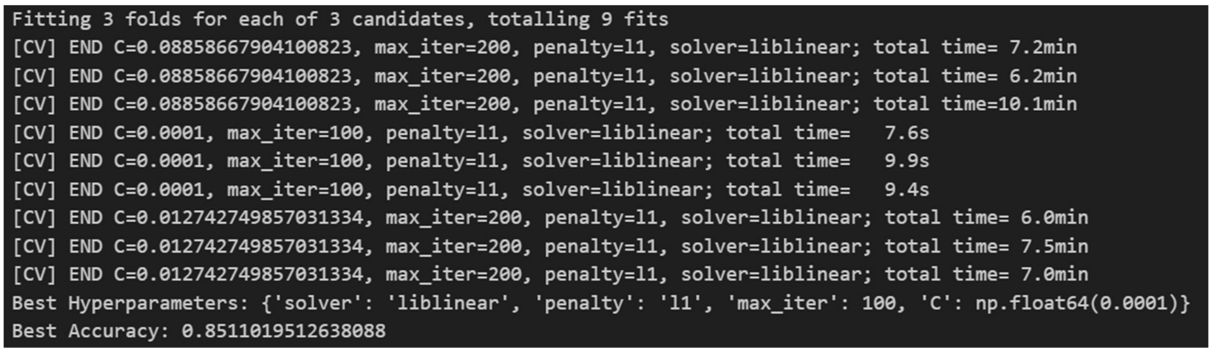
```
Stratified Cross-Validation Accuracy: 0.78 ± 0.00
Cross-Validation Classification Report:
              precision    recall  f1-score   support

           1       0.07      0.19      0.10     15582
           2       0.25      0.20      0.22    163596
           3       0.87      0.88      0.88   1024142

    accuracy                           0.78   1203320
   macro avg       0.40      0.42      0.40   1203320
weighted avg       0.78      0.78      0.78   1203320
```

This showed the best results out of any of the models and their variations. With an accuracy more than 75% and good recall and precision according to the dataset. This specific model became the best model out of all.

<u>**Task – 5 (Hyperparameter Tuning):**</u>

I used RandomizedSearchCV to find the best hyper parameters for my model, because this uses less resources and is computations cheaper than gridsearchCV.

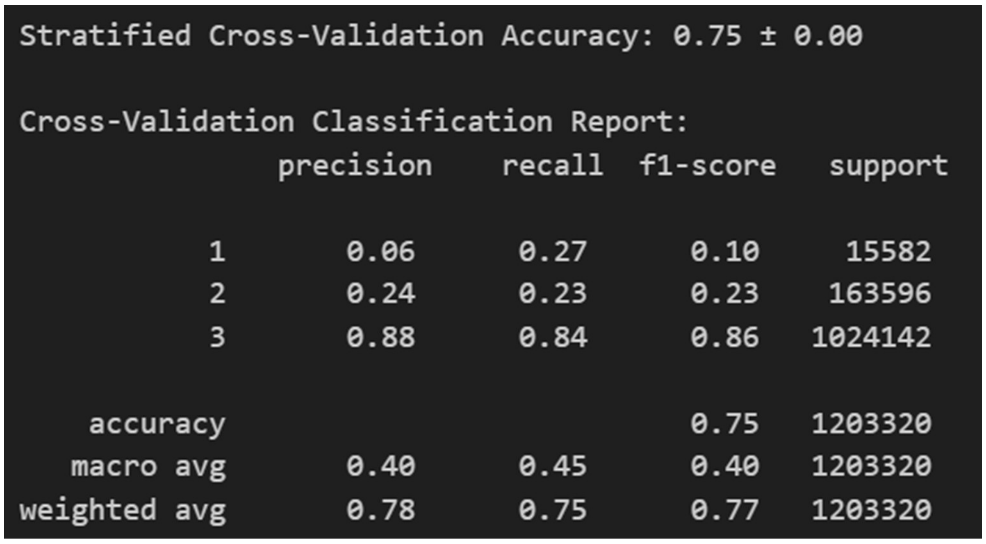
One thing to keep in mind, this took very long (30-40 mins!), also the system kept crashing due to hugh amounts of data unable to be processed by the my laptop’s 8 GBs. So only the logistic regression was fine tuned.



```
Fitting 3 folds for each of 3 candidates, totalling 9 fits
[CV] END C=0.08858667904100823, max_iter=200, penalty=l1, solver=liblinear; total time= 7.2min
[CV] END C=0.08858667904100823, max_iter=200, penalty=l1, solver=liblinear; total time= 6.2min
[CV] END C=0.08858667904100823, max_iter=200, penalty=l1, solver=liblinear; total time=10.1min
[CV] END C=0.0001, max_iter=100, penalty=l1, solver=liblinear; total time=   7.6s
[CV] END C=0.0001, max_iter=100, penalty=l1, solver=liblinear; total time=   9.9s
[CV] END C=0.0001, max_iter=100, penalty=l1, solver=liblinear; total time=   9.4s
[CV] END C=0.012742749857031334, max_iter=200, penalty=l1, solver=liblinear; total time= 6.0min
[CV] END C=0.012742749857031334, max_iter=200, penalty=l1, solver=liblinear; total time= 7.5min
[CV] END C=0.012742749857031334, max_iter=200, penalty=l1, solver=liblinear; total time= 7.0min
Best Hyperparameters: {'solver': 'liblinear', 'penalty': 'l1', 'max_iter': 100, 'C': np.float64(0.0001)}
Best Accuracy: 0.8511019512638088
```

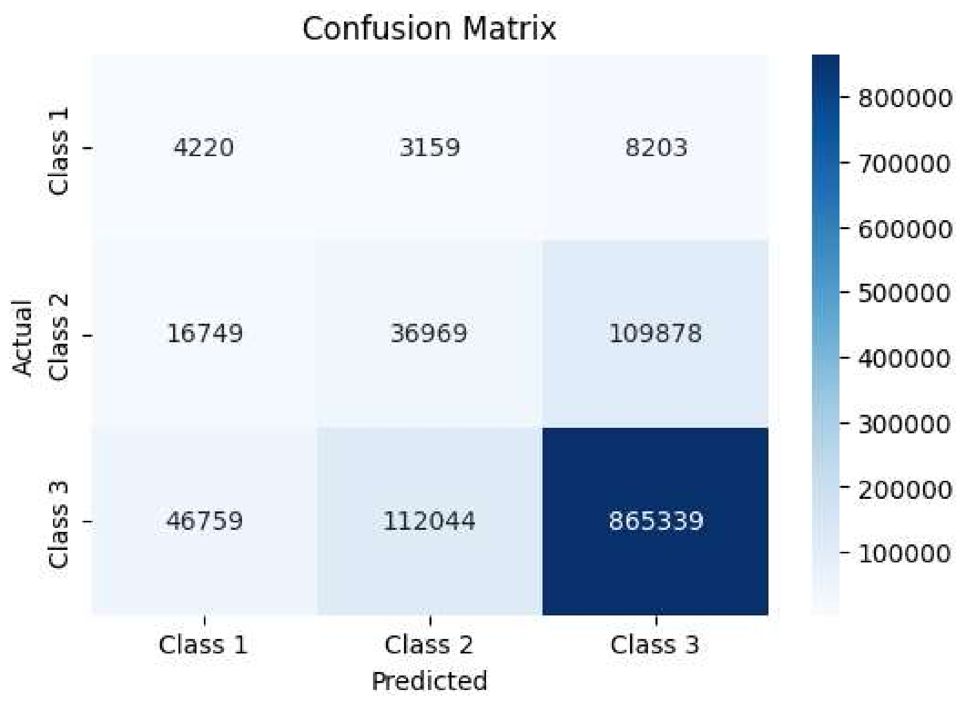
Task – 6 (Final Model Evaluation):

As discussed in the previous section, the stratified cross-validation of gradient boosting, gave the best results out of the all the model performed. With their final metrics being this:



```
Stratified Cross-Validation Accuracy: 0.75 ± 0.00

Cross-Validation Classification Report:
              precision    recall  f1-score   support

           1       0.06      0.27      0.10     15582
           2       0.24      0.23      0.23    163596
           3       0.88      0.84      0.86   1024142

    accuracy                           0.75   1203320
   macro avg       0.40      0.45      0.40   1203320
weighted avg       0.78      0.75      0.77   1203320
```

This is slightly less than the report shared above, this is due to some of the parameters changed I had to make, in order for my system to not crash and run properly. A confusion matrix was also generated showing the performance of the matrix.



For the rest of the tasks 8, 9, 10, all of them have been achieved using this dataset and all of the points have been mentioned in the report.

Video Link:

https://www.dropbox.com/scl/fi/x9wyqdrdo41i0hw7635xx/i222369_ML_VideoExplanation.mp4?rlkey=4sksnyhd1req2l0fijfwf9ese&st=8af0l492&dl=0