Configuration Manual

MSc Research Project
Data Analytics

Soham Mohire
Student ID: x19225491

School of Computing
National College of Ireland

Supervisor: Vladimir Milosavljevic

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Soham Mohire
Student ID:	x19225491
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Vladimir Milosavljevic
Submission Due Date:	15/12/2022
Project Title:	Configuration Manual
Word Count:	1004
Page Count:	13

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	14th December 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Soham Mohire
x19225491

1 Introduction

One of the most undesirable and unforeseen event occurring to road user would be a road accident. In recent times, large number of such accidents has been witnessed in many parts of the USA. Occurrence of fatalities and injuries has created a huge impact on the economy of the country. This has not only led to untimely deaths, but also loss of property damage at social levels. In a survey conducted by WHO in 2017 [1] states that 1.5 million drivers die on a yearly basis due to road accidents and car crashes. In addition to this they stated, that due to negligence in traffic rules, the number of deaths is more likely to increase by 2030. The statistics given by the survey has concluded that 47 road users died on an everyday basis, which led to a 3 percent decrease in the GDP. In another survey report by Michigan Traffic [2] they have registered an estimation of 314,921 death occurrences due to road accidents in 2017. The numerical estimation led to a loss of 230 billion dollars with a massive drop in the economy of the country. Such devastating statistics has eventually become a matter of rising concern, for not only government officials but also research scholars and accident experts belonging to the same field. Extensive research has been conducted to determine all the factors that would highly be responsible for such statistics. Unfortunately, the surveys are conducted through questionnaires and not much implementation of statistical tools has been witnessed. Hence, the primary aim of research scholars is to not only to analyse the factors responsible for such accidents; but also resolve the severity of accidents from a behavioural angle of road users.

Since accidents are highly unpredictable and unforeseen; drawing observations from them is a significant challenge. In addition to the observations being made, drawing conclusions with 100 percent accurate data and results is quite an impossible task. To overcome this limitation, the thesis proposes to detect the severities that might lead to road accidents by using and implementing technical methods of Machine Learning. ML is observed to be as one of the most advanced and technical method to predict the occurrence of such mishaps. Hence, the aim is to build a classification model that can predict road accidents through the experimentation analysis of five classifiers. The classifiers would further undergo a set of processing techniques and make smart decision s by gaining insights from historical data. In addition to the implementation of the selected classifiers, the thesis also presents a conceptual theory of SMOTE; wherein the data would be balanced so that the classification algorithms can perform on them in a significant manner.

The Configuration Manual will be divided into six section excluding this introduction part as follows :

[1]https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries
[2]https://www.michigan.gov/msp/divisions/cjic/traffic-crash-reporting-unit

- Environmental Setup
- Libraries Required
- Dataset
- User Interface
- Implementation
- Code Repository

2 Environmental Setup

1. **Hardware Requirements**

- 8GB RAM.
- 250 GB HDD.
- 1.6 GHz Intel. Core i5

2. **Software Requirements**

- Windows 10.
- Python 3.6.3.

3. **Programming Prerequisites**

- Google Colab.
- Python.

For implementing the code, Google Colab has been used since they are hosted in cloud the code is centralized and can be accessed anytime anywhere, also they have all the packages readily available. Following are the dataset files uploaded on Colab. These dataset is taken from Kaggle.

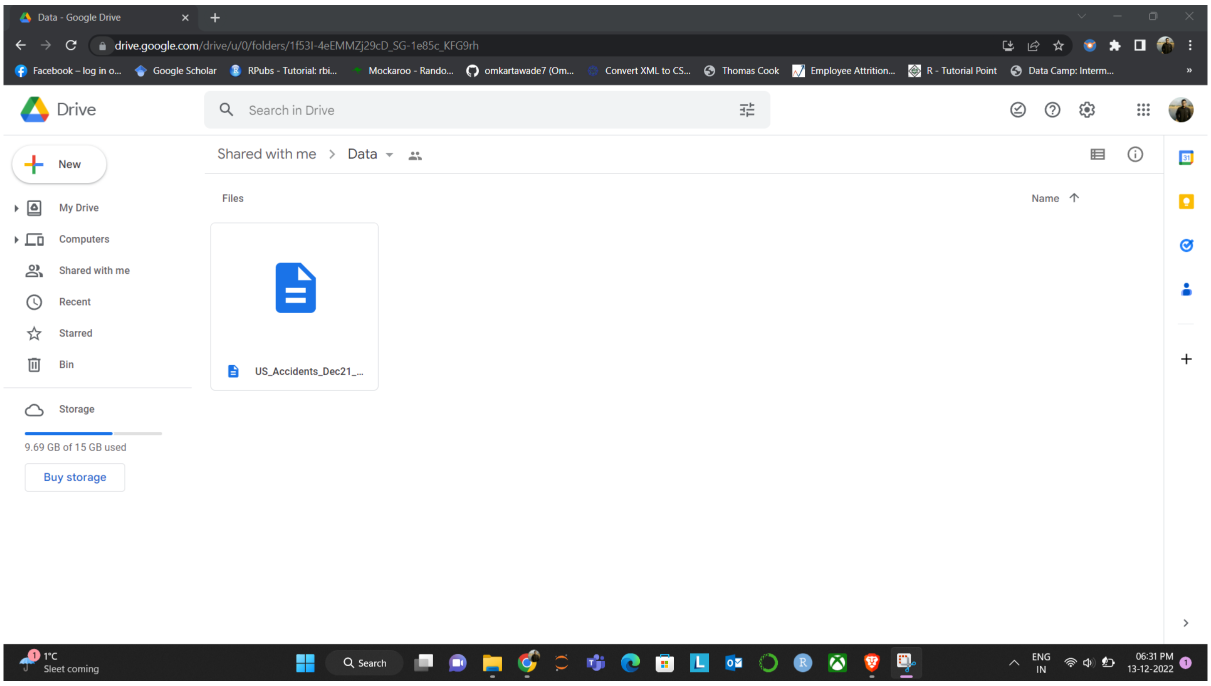
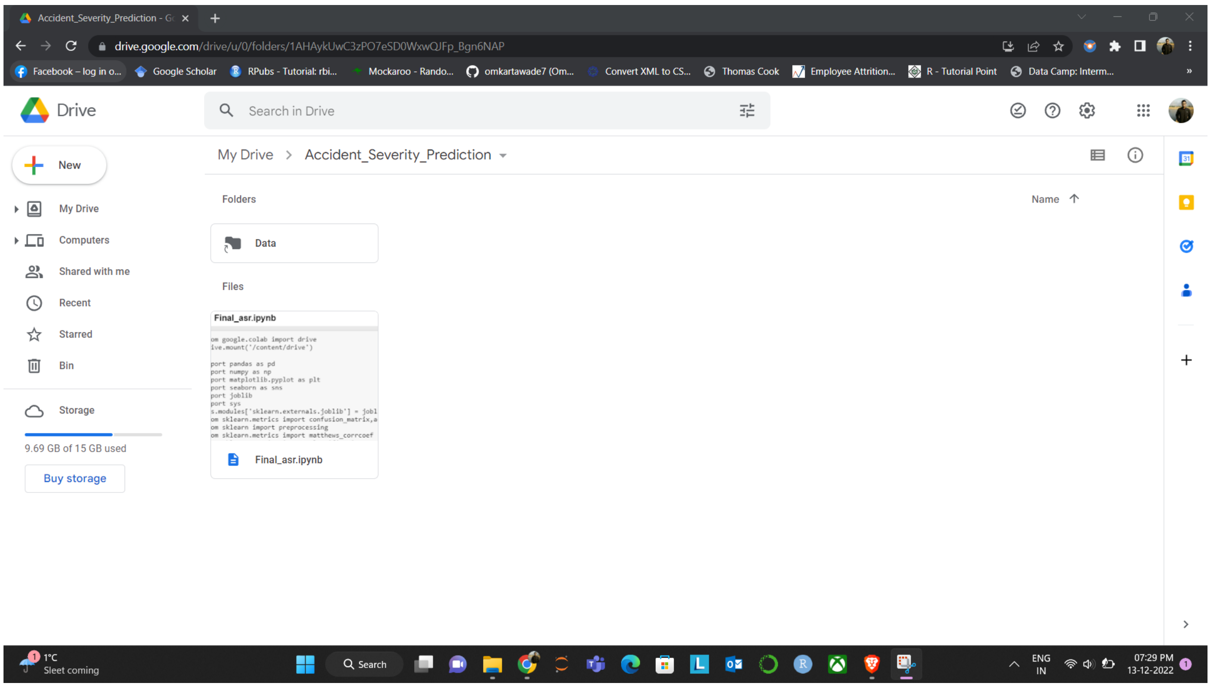


Figure 1: Dataset saved on Google Drive



```
om google.colab import drive
ive.mount('/content/drive')

port pandas as pd
port numpy as np
port matplotlib.pyplot as plt
port seaborn as sns
port joblib
port sys
s.modules['sklearn.externals.joblib'] = jobl
om sklearn.metrics import confusion_matrix,a
om sklearn import preprocessing
om sklearn.metrics import matthews_corrcoef
```

Figure 2: Google Drive Storage Structure

3 Libraries Required

All the libraries required for building this research project are mentioned in table 1 along with their usage:

- **Pandas :** It offers data structures and operations for manipulating numerical tables and time series.
- **Numpy :** It is used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices
- **SKLearn :** Used for implementing various machine learning algorithms : Gaussian Naiive Bayes, Support vector classifier, LogisticRegression
- **Matplotlib :** It is used for plotting various graphical visualisations
- **SKLearn.Metrics** : For generating various performance metrics: Accuracy, Confusion Matrix
- **SKLearn.Ensemble :** Implementing RandomForestClassifier, Decision Tree, Naive Bayes, Logistic Regression, Stacking Classifier.
- **IMBLearn :** For balancing the data using SMOTE Oversampling.

4 Data Set Details

This is a countrywide car accident dataset, which covers 49 states of the USA. The accident data are collected from February 2016 to Dec 2021, using multiple APIs that provide streaming traffic incident (or event) data. These APIs broadcast traffic data captured by a variety of entities, such as the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road-networks. Currently, there are about 2.8 million accident records in this dataset.

- **Link to Dataset** : https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents

The number of records against each target class is depicted in figure where in the value 1 indicates less severity whereas the value 4 indicates maximum severity.

5 Data Balance(Using SMOTE)

The data obtained from the repository is imbalanced in nature; and hence needs to be balanced so that respective algorithms can be performed on them. For this purpose, a simple approach of duplicating minority class is performed using SMOTE. SMOTE is abbreviated to Synthetic Mining Oversampling Technique. The problem of data imbalance as seen in fig 3 is resolved in the by oversampling the minority classes of the dataset as seen in fig 4.

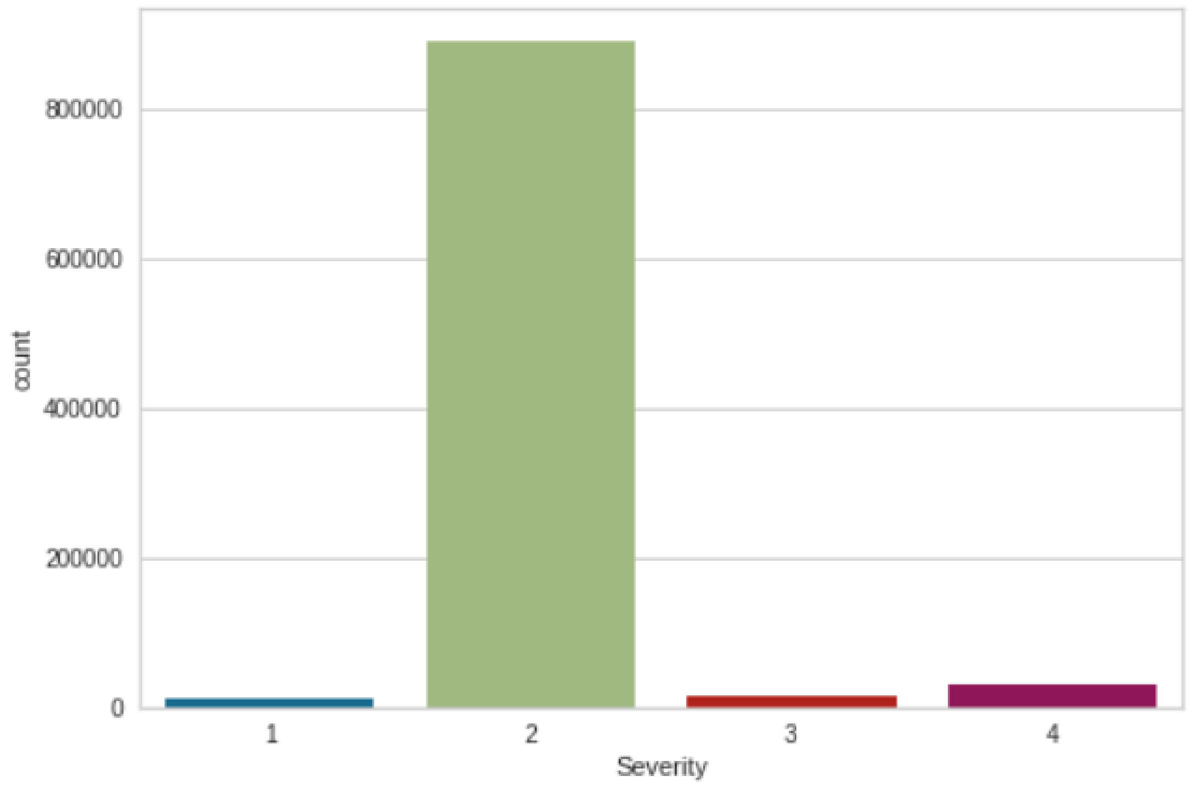


Figure 3: Dataset (Imbalanced Data) saved on Google Drive

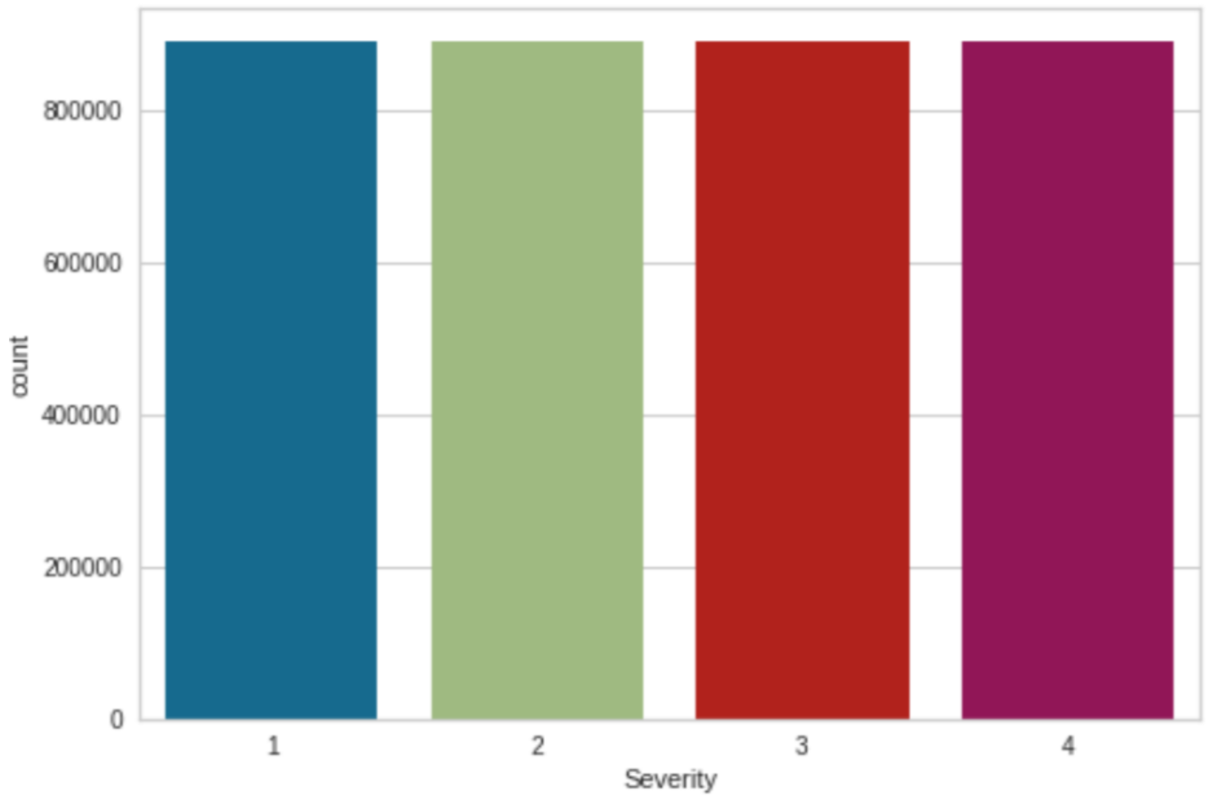
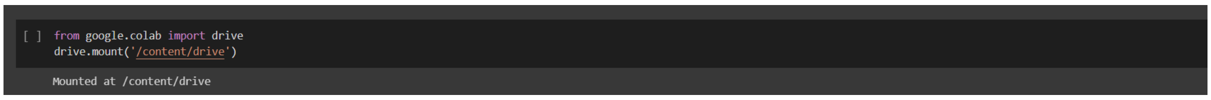


Figure 4: Balanced Dataset using SMOTE Technique

6 Code Implementation

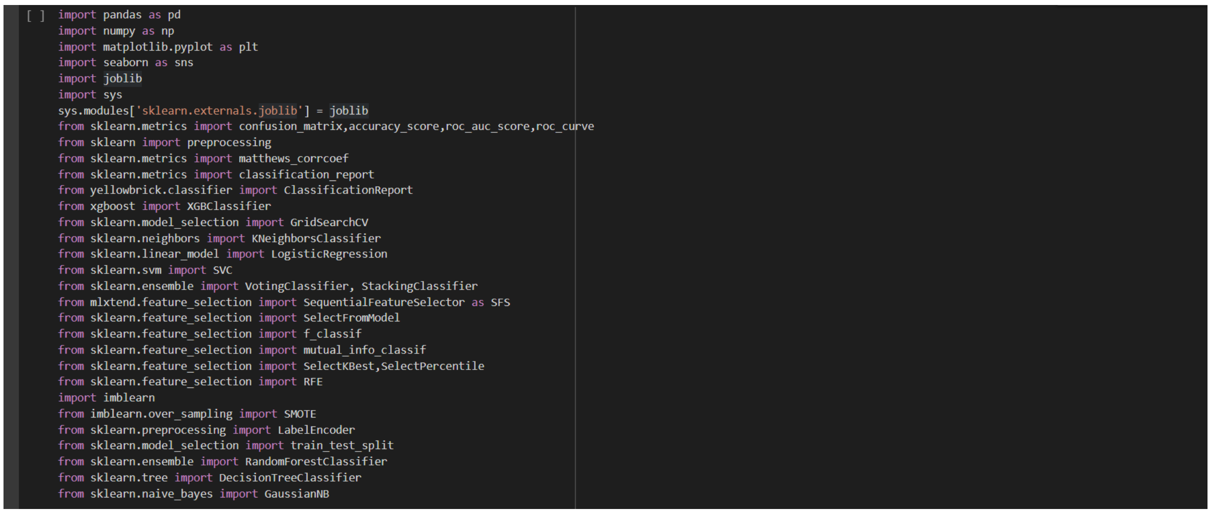
In this section all the steps followed to run the program along with code are described

- Mounting dataset from Google Drive onto Google Collab.



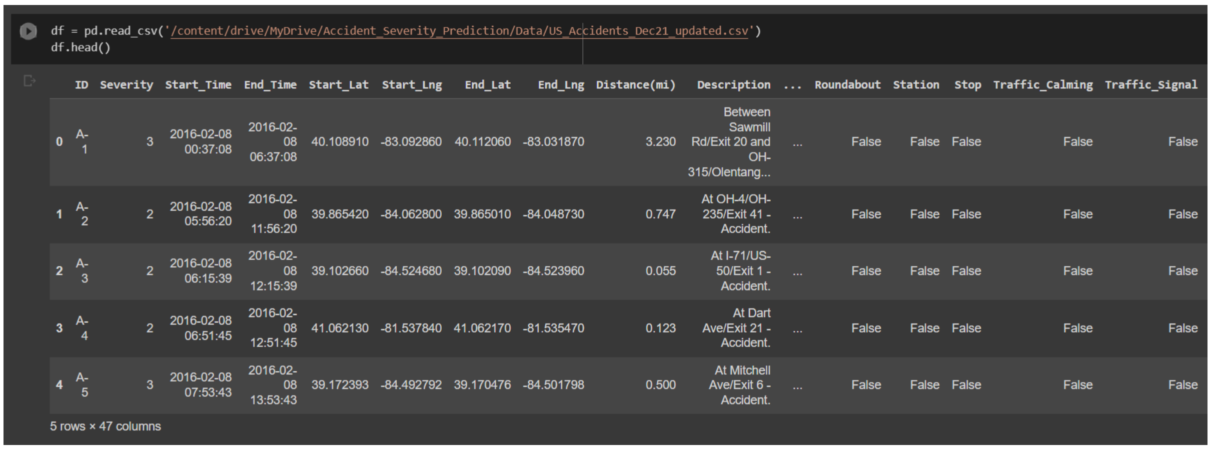
```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

- Importing the required libraries



```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    import joblib
    import sys
    sys.modules['sklearn.externals.joblib'] = joblib
    from sklearn.metrics import confusion_matrix,accuracy_score,roc_auc_score,roc_curve
    from sklearn import preprocessing
    from sklearn.metrics import matthews_corrcoef
    from sklearn.metrics import classification_report
    from yellowbrick.classifier import ClassificationReport
    from xgboost import XGBClassifier
    from sklearn.model_selection import GridSearchCV
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.linear_model import LogisticRegression
    from sklearn.svm import SVC
    from sklearn.ensemble import VotingClassifier, StackingClassifier
    from mlxtend.feature_selection import SequentialFeatureSelector as SFS
    from sklearn.feature_selection import SelectFromModel
    from sklearn.feature_selection import f_classif
    from sklearn.feature_selection import mutual_info_classif
    from sklearn.feature_selection import SelectKBest,SelectPercentile
    from sklearn.feature_selection import RFE
    import imblearn
    from imblearn.over_sampling import SMOTE
    from sklearn.preprocessing import LabelEncoder
    from sklearn.model_selection import train_test_split
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.naive_bayes import GaussianNB
```

- Loading Dataset.

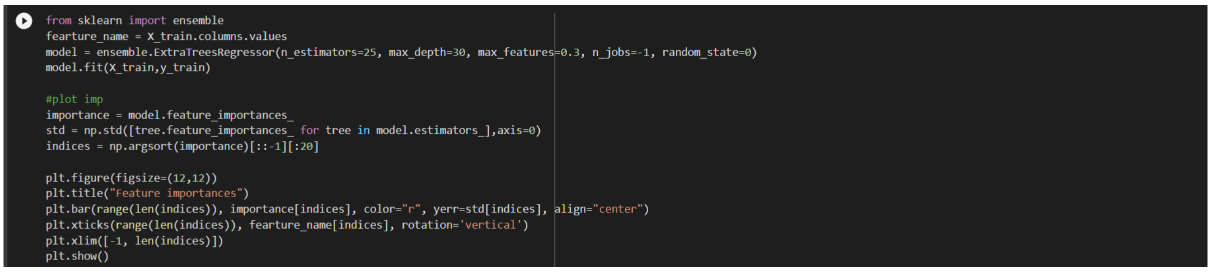


```
df = pd.read_csv('/content/drive/MyDrive/Accident_Severity_Prediction/Data/US_Accidents_Dec21_updated.csv')
df.head()
```

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Description	...	Roundabout	Station	Stop	Traffic_Calming	Traffic_Signal
0	A-1	3	2016-02-08 00:37:08	2016-02-08 06:37:08	40.108910	-83.092860	40.112060	-83.031870	3.230	Between Sawmill Rd/Exit 20 and OH-315/Olentang...	...	False	False	False	False	False
1	A-2	2	2016-02-08 05:56:20	2016-02-08 11:56:20	39.865420	-84.062800	39.865010	-84.048730	0.747	At OH-4/OH-235/Exit 41 - Accident.	...	False	False	False	False	False
2	A-3	2	2016-02-08 06:15:39	2016-02-08 12:15:39	39.102660	-84.524680	39.102090	-84.523960	0.055	At I-71/US-50/Exit 1 - Accident.	...	False	False	False	False	False
3	A-4	2	2016-02-08 06:51:45	2016-02-08 12:51:45	41.062130	-81.537840	41.062170	-81.535470	0.123	At Dart Ave/Exit 21 - Accident.	...	False	False	False	False	False
4	A-5	3	2016-02-08 07:53:43	2016-02-08 13:53:43	39.172393	-84.492792	39.170476	-84.501798	0.500	At Mitchell Ave/Exit 6 - Accident.	...	False	False	False	False	False

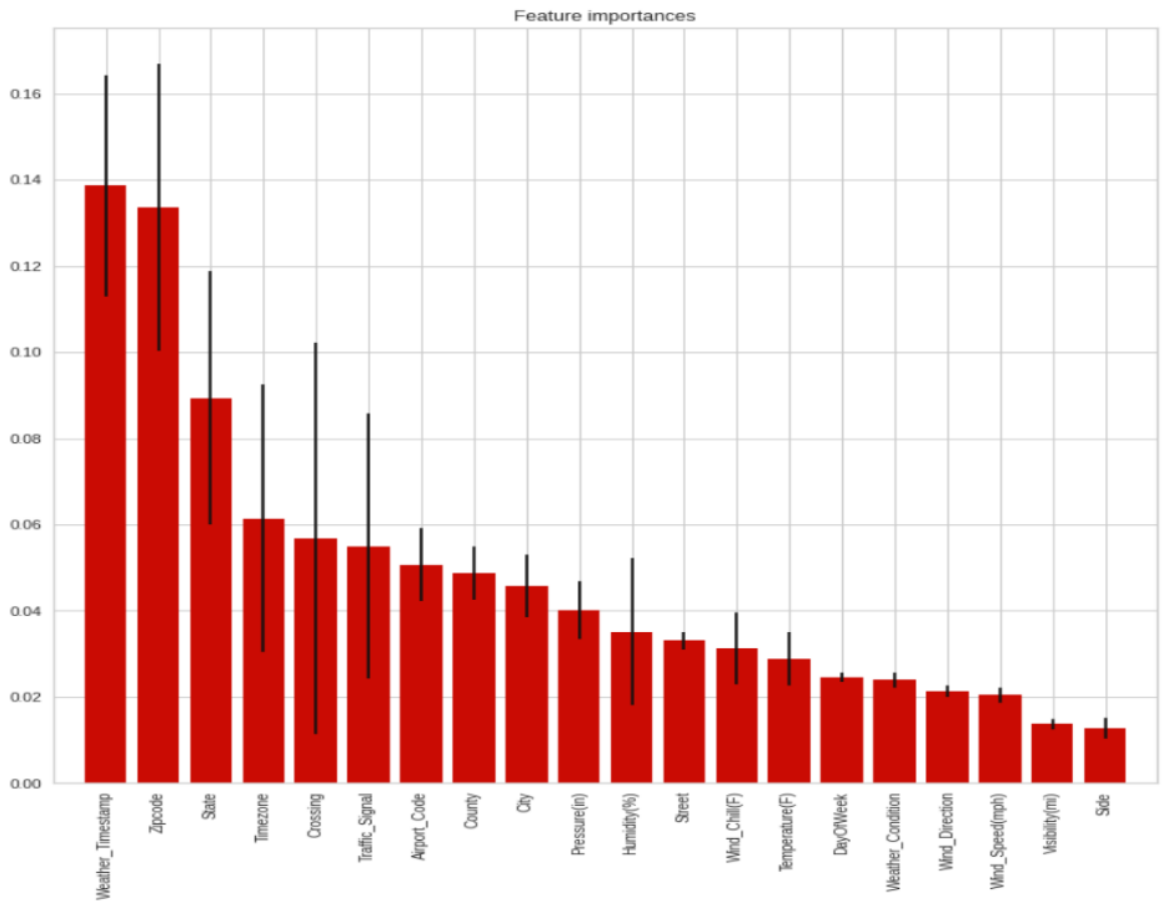
5 rows × 47 columns

- Selecting relevant parameters using Feature Selection.



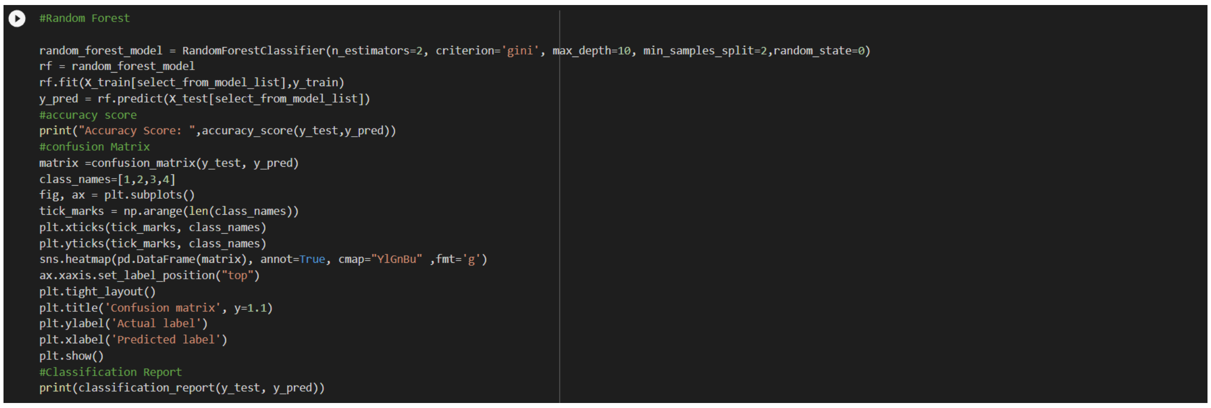
```
from sklearn import ensemble
fearture_name = X_train.columns.values
model = ensemble.ExtraTreesRegressor(n_estimators=25, max_depth=30, max_features=0.3, n_jobs=-1, random_state=0)
model.fit(X_train,y_train)

#plot imp
importance = model.feature_importances_
std = np.std([tree.feature_importances_ for tree in model.estimators_],axis=0)
indices = np.argsort(importance)[::-1][:20]

plt.figure(figsize=(12,12))
plt.title("Feature importances")
plt.bar(range(len(indices)), importance[indices], color="r", yerr=std[indices], align="center")
plt.xticks(range(len(indices)), fearture_name[indices], rotation='vertical')
plt.xlim([-1, len(indices)])
plt.show()
```

- 20 top influential features selected.


Feature importances
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Weather_Timestamp
Zipcode
State
Timezone
Crossing
Traffic_Signal
Airport_Code
County
City
Pressure(in)
Humidity(%)
Street
Wind_Chill(F)
Temperature(F)
DayOfWeek
Weather_Condition
Wind_Direction
Wind_Speed(mph)
Visibility(mi)
Side

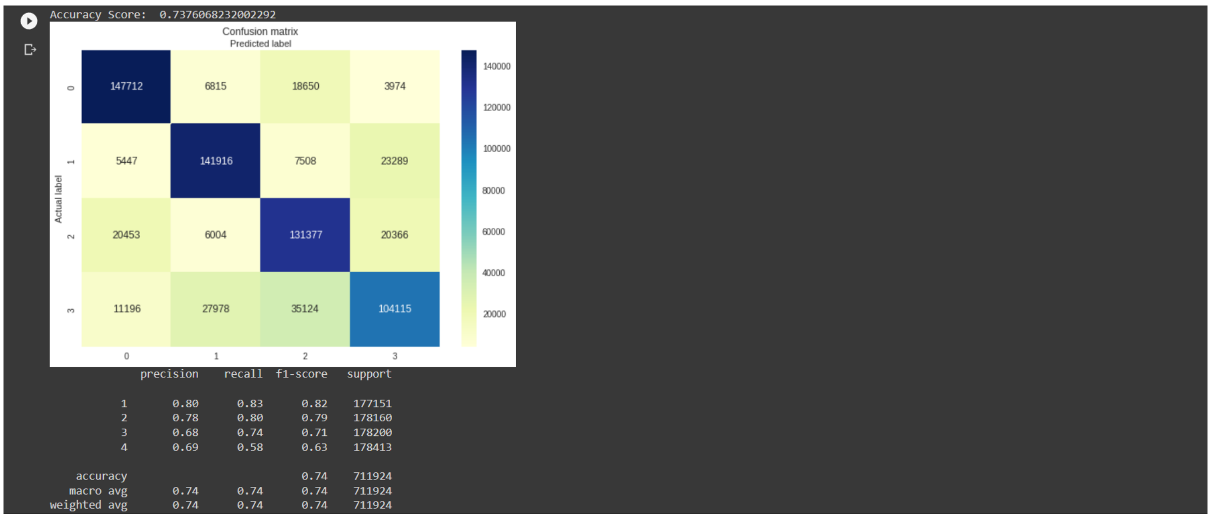
7 Experiment

7.1 Experiment 1 : Random Forest



```
#Random Forest

random_forest_model = RandomForestClassifier(n_estimators=2, criterion='gini', max_depth=10, min_samples_split=2,random_state=0)
rf = random_forest_model
rf.fit(X_train[select_from_model_list],y_train)
y_pred = rf.predict(X_test[select_from_model_list])
#accuracy score
print("Accuracy Score: ",accuracy_score(y_test,y_pred))
#confusion Matrix
matrix =confusion_matrix(y_test, y_pred)
class_names=[1,2,3,4]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
#Classification Report
print(classification_report(y_test, y_pred))
```

Figure 5: Random Forest (Code Snippet)



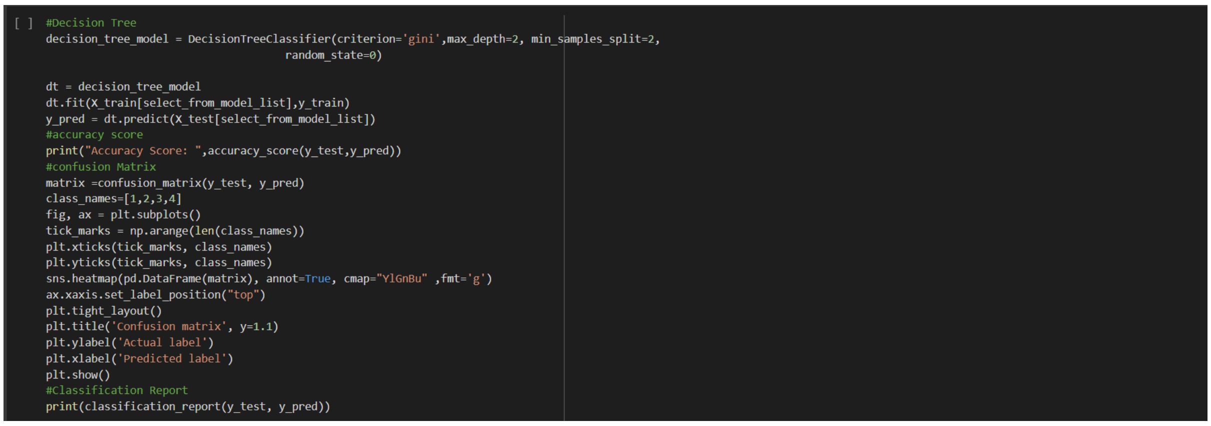
```
Accuracy Score:  0.7376068232002292
```

	0	1	2	3
0	147712	6815	18650	3974
1	5447	141916	7508	23289
2	20453	6004	131377	20366
3	11196	27978	35124	104115

```
              precision    recall  f1-score   support

           1       0.80      0.83      0.82    177151
           2       0.78      0.80      0.79    178160
           3       0.68      0.74      0.71    178200
           4       0.69      0.58      0.63    178413

    accuracy                           0.74    711924
   macro avg       0.74      0.74      0.74    711924
weighted avg       0.74      0.74      0.74    711924
```

Figure 6: Random Forest Confusion Matrix and Report Table

7.2 Experiment 2 : Decision Tree



```
[ ] #Decision Tree
    decision_tree_model = DecisionTreeClassifier(criterion='gini',max_depth=2, min_samples_split=2,
                                                 random_state=0)

    dt = decision_tree_model
    dt.fit(X_train[select_from_model_list],y_train)
    y_pred = dt.predict(X_test[select_from_model_list])
    #accuracy score
    print("Accuracy Score: ",accuracy_score(y_test,y_pred))
    #confusion Matrix
    matrix =confusion_matrix(y_test, y_pred)
    class_names=[1,2,3,4]
    fig, ax = plt.subplots()
    tick_marks = np.arange(len(class_names))
    plt.xticks(tick_marks, class_names)
    plt.yticks(tick_marks, class_names)
    sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
    ax.xaxis.set_label_position("top")
    plt.tight_layout()
    plt.title('Confusion matrix', y=1.1)
    plt.ylabel('Actual label')
    plt.xlabel('Predicted label')
    plt.show()
    #Classification Report
    print(classification_report(y_test, y_pred))
```

Figure 7: Decision Tree (Code Snippet)

```
Accuracy Score:  0.592230350430664
```

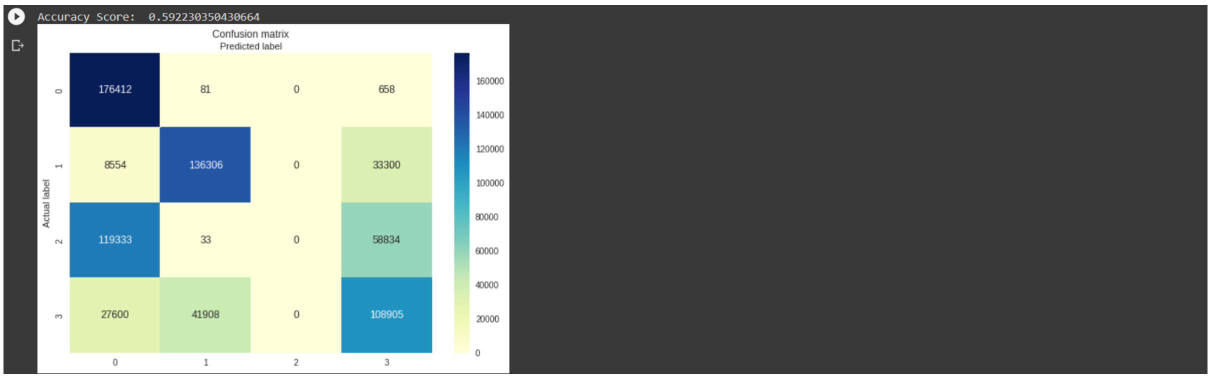
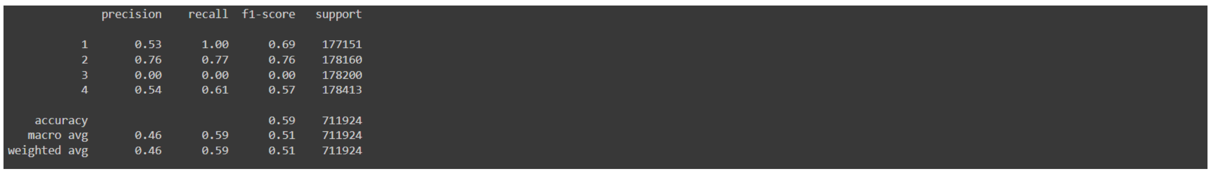


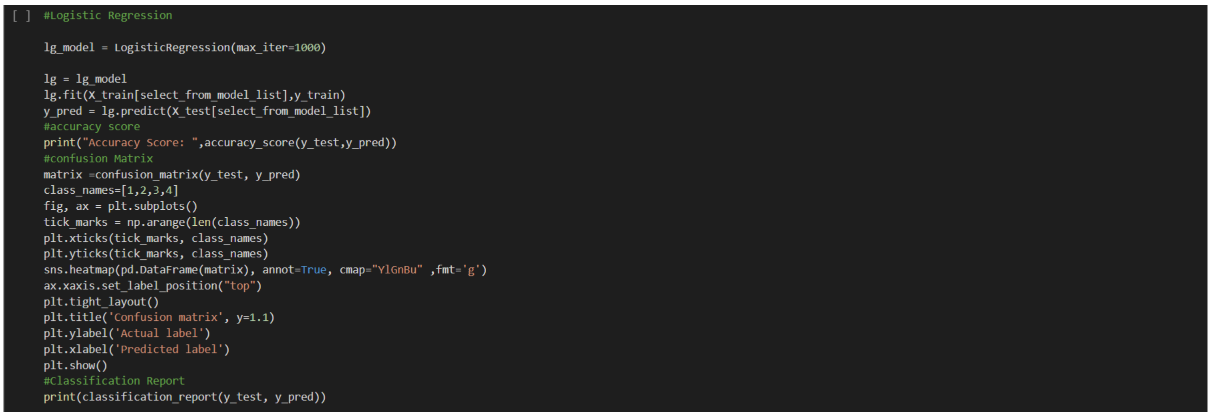
Figure 8: Decision Tree Confusion Matrix



```
              precision    recall  f1-score   support

           1       0.53      1.00      0.69    177151
           2       0.76      0.77      0.76    178160
           3       0.00      0.00      0.00    178200
           4       0.54      0.61      0.57    178413

    accuracy                           0.59    711924
   macro avg       0.46      0.59      0.51    711924
weighted avg       0.46      0.59      0.51    711924
```

Figure 9: Decision Tree Report Table

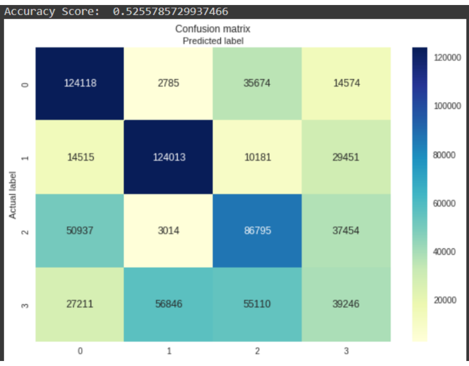
7.3 Experiment 3 : Logistic Regression



```
[ ] #Logistic Regression

    lg_model = LogisticRegression(max_iter=1000)

    lg = lg_model
    lg.fit(X_train[select_from_model_list],y_train)
    y_pred = lg.predict(X_test[select_from_model_list])
    #accuracy score
    print("Accuracy Score: ",accuracy_score(y_test,y_pred))
    #confusion Matrix
    matrix =confusion_matrix(y_test, y_pred)
    class_names=[1,2,3,4]
    fig, ax = plt.subplots()
    tick_marks = np.arange(len(class_names))
    plt.xticks(tick_marks, class_names)
    plt.yticks(tick_marks, class_names)
    sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
    ax.xaxis.set_label_position("top")
    plt.tight_layout()
    plt.title('Confusion matrix', y=1.1)
    plt.ylabel('Actual label')
    plt.xlabel('Predicted label')
    plt.show()
    #Classification Report
    print(classification_report(y_test, y_pred))
```

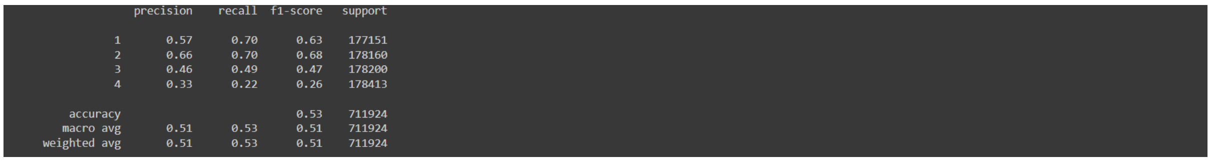
Figure 10: Logistic Regression (Code Snippet)

```
Accuracy Score:  0.5255785729937466
```



Actual label \ Predicted label	0	1	2	3
0	124118	2785	35674	14574
1	14515	124013	10181	29451
2	50937	3014	86795	37454
3	27211	56846	55110	39246

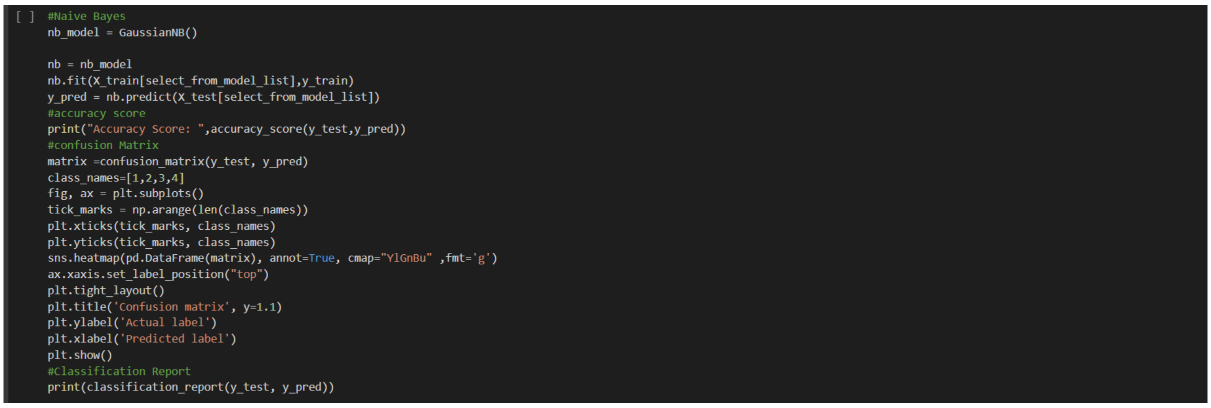
Figure 11: Logistic Regression Confusion Matrix



```
              precision    recall  f1-score   support

           1       0.57      0.70      0.63    177151
           2       0.66      0.70      0.68    178160
           3       0.46      0.49      0.47    178200
           4       0.33      0.22      0.26    178413

    accuracy                           0.53    711924
   macro avg       0.51      0.53      0.51    711924
weighted avg       0.51      0.53      0.51    711924
```

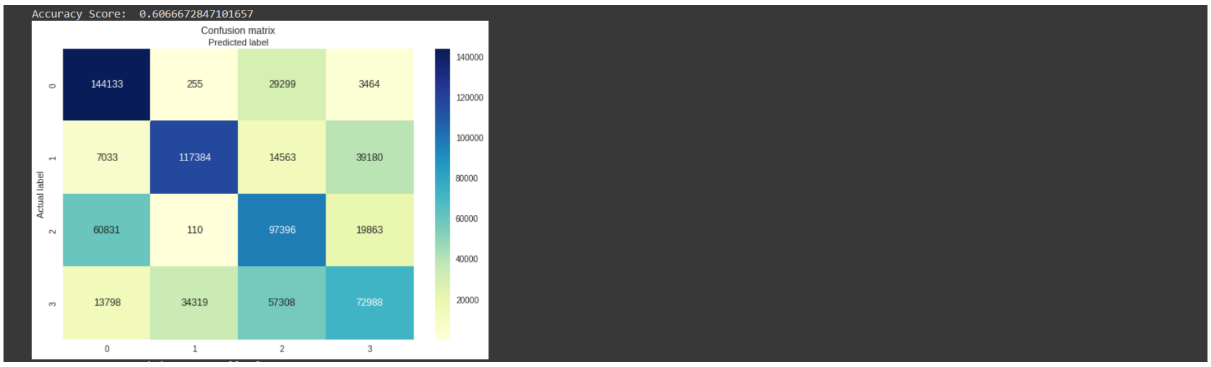
Figure 12: Logistic Regression Report Table

7.4 Experiment 4 : Naive Bayes



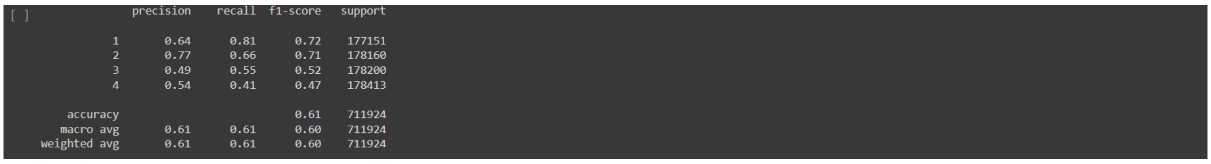
```
[ ]  #Naive Bayes
     nb_model = GaussianNB()

     nb = nb_model
     nb.fit(X_train[select_from_model_list],y_train)
     y_pred = nb.predict(X_test[select_from_model_list])
     #accuracy score
     print("Accuracy Score: ",accuracy_score(y_test,y_pred))
     #confusion Matrix
     matrix =confusion_matrix(y_test, y_pred)
     class_names=[1,2,3,4]
     fig, ax = plt.subplots()
     tick_marks = np.arange(len(class_names))
     plt.xticks(tick_marks, class_names)
     plt.yticks(tick_marks, class_names)
     sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
     ax.xaxis.set_label_position("top")
     plt.tight_layout()
     plt.title('Confusion matrix', y=1.1)
     plt.ylabel('Actual label')
     plt.xlabel('Predicted label')
     plt.show()
     #Classification Report
     print(classification_report(y_test, y_pred))
```

Figure 13: Naive Bayes (Code Snippet)



```
Accuracy Score:  0.6066672847101657
```

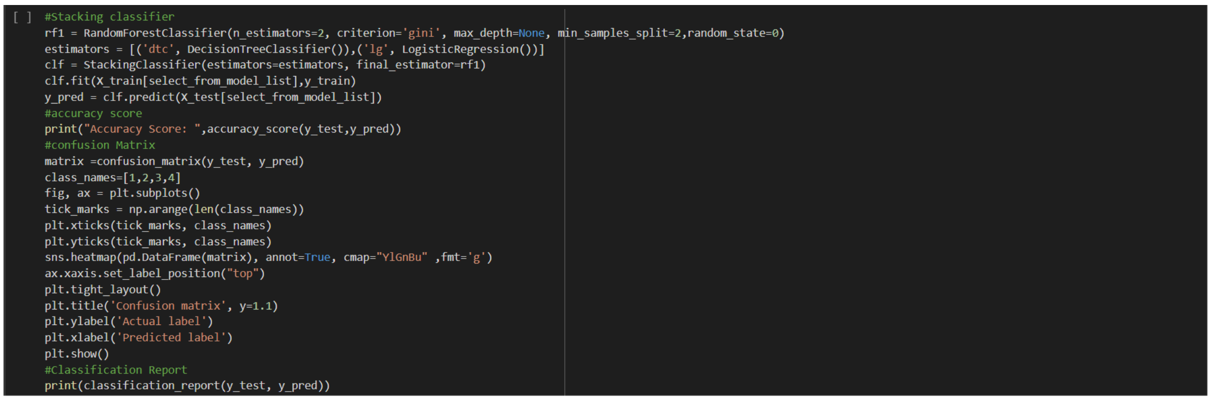
Figure 14: Naive Bayes Confusion Matrix



```
[ ]               precision    recall  f1-score   support

           1       0.64      0.81      0.72    177151
           2       0.77      0.66      0.71    178160
           3       0.49      0.55      0.52    178200
           4       0.54      0.41      0.47    178413

    accuracy                           0.61    711924
   macro avg       0.61      0.61      0.60    711924
weighted avg       0.61      0.61      0.60    711924
```

Figure 15: Naive Bayes Report Table

7.5 Experiment 5 : Ensemble Model (Stacking Classifier)



```
[ ] #Stacking classifier
    rf1 = RandomForestClassifier(n_estimators=2, criterion='gini', max_depth=None, min_samples_split=2,random_state=0)
    estimators = [('dtc', DecisionTreeClassifier()),('lg', LogisticRegression())]
    clf = StackingClassifier(estimators=estimators, final_estimator=rf1)
    clf.fit(X_train[select_from_model_list],y_train)
    y_pred = clf.predict(X_test[select_from_model_list])
    #accuracy score
    print("Accuracy Score: ",accuracy_score(y_test,y_pred))
    #confusion Matrix
    matrix =confusion_matrix(y_test, y_pred)
    class_names=[1,2,3,4]
    fig, ax = plt.subplots()
    tick_marks = np.arange(len(class_names))
    plt.xticks(tick_marks, class_names)
    plt.yticks(tick_marks, class_names)
    sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
    ax.xaxis.set_label_position("top")
    plt.tight_layout()
    plt.title('Confusion matrix', y=1.1)
    plt.ylabel('Actual label')
    plt.xlabel('Predicted label')
    plt.show()
    #Classification Report
    print(classification_report(y_test, y_pred))
```

Figure 16: Stacking Classifier (Code Snippet)

```
Accuracy Score:  0.9265469347851737
```

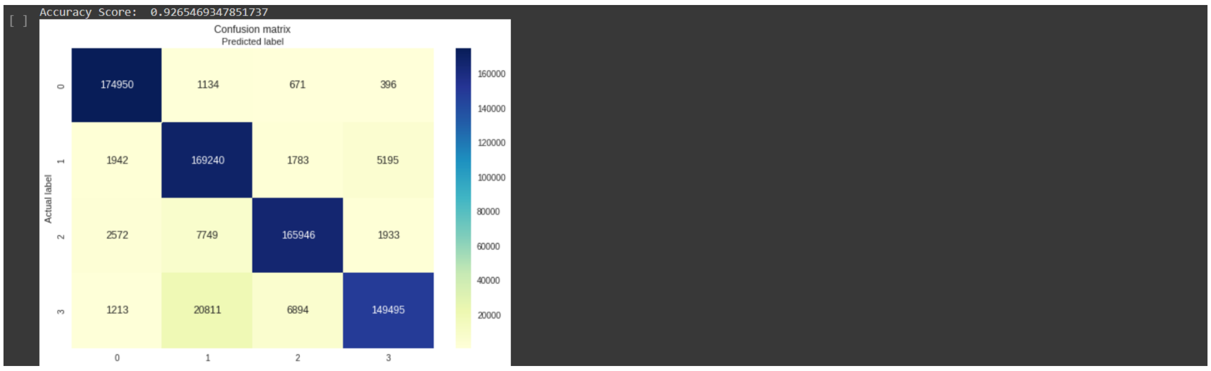
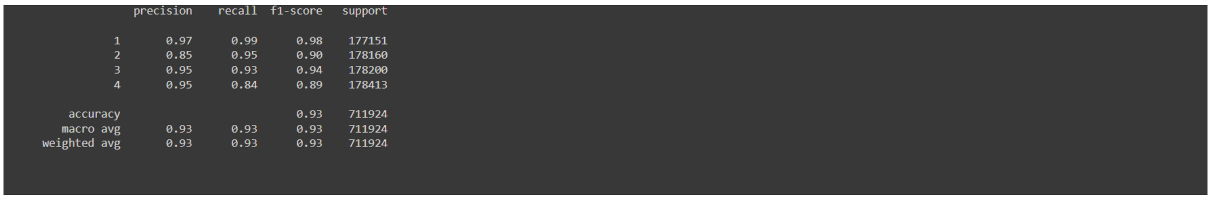


Figure 17: Stacking Classifier Confusion Matrix



```
              precision    recall  f1-score   support

           1       0.97      0.99      0.98    177151
           2       0.85      0.95      0.90    178160
           3       0.95      0.93      0.94    178200
           4       0.95      0.84      0.89    178413

    accuracy                           0.93    711924
   macro avg       0.93      0.93      0.93    711924
weighted avg       0.93      0.93      0.93    711924
```

Figure 18: Stacking Classifier Report Table