DCU School of Computing

Assignment Submission

Student Name(s):	Fawaz Alsafadi, Ayman El Gendy
Student Number(s):	1538087, 15395461
Programme:	BSc in Computer Applications
Project Title:	Predicting traffic offences
Module code:	CA4010
Lecturer:	Mark Roantree
Project Due Date:	18/12/2019

Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying is a grave and serious offence in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion, or copying. I have read and understood the Assignment Regulations set out in the module documentation. I have identified and included the source of all facts, ideas, opinions, viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references.

I have not copied or paraphrased an extract of any length from any source without identifying the source and using quotation marks as appropriate. Any images, audio recordings, video or other materials have likewise been originated and produced by me or are fully acknowledged and identified.

This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study. I have read and understood the referencing guidelines found at

http://www.library.dcu.ie/citing&refguide08.pdf and/or recommended in the assignment guidelines.

I understand that I may be required to discuss with the module lecturer/s the contents of this submission.

I/me/my incorporates we/us/our in the case of group work, which is signed by all of us.

Signed: Fawaz Alsafadi & Ayman El Gendy

TABLE OF CONTENTS

1. INTRODUCTION

This report outlines the details of the dataset that we chose, the idea and how we implemented our algorithms for our predictions. We also discuss and build on the work and advice we received from our workshop. Finally, we show results and analysis and we highlight what we learned and what we could do better next time in order to improve our prediction and data preparation for similar projects in the future.

2. IDEA AND DATASET DESCRIPTION

2.1 IDEA

The inspiration for our idea came from our fellow students and peers who were all at the age were people began getting their driving licences. We were very interested in attempting to discover for ourselves whether at a particular age are drivers more likely to commit a certain type of driving offence more than other ages and was it possible to predict the most likely ticket based on age alone. We decided to utilize techniques and skills learned in order to attempt to make this prediction or analyse if it is possible.

Real world applications of our idea are plentiful, an example application is that Road safety authorities could use such a system to determine what offence a certain age group is likely to commit and target them with safety advertisement campaigns based on the offence they are likely to cause

2.2 DATASET DESCRIPTION

In order to carry out this project we decided to use a Kaggle data set for which a link is provided in the appendix. The data set originally contained all Traffic tickets issued in New York City (NYC) from the year 2014 to 2017. The data set had over 14 million lines in total each representing a ticket. Originally, we wanted to user the entirety of the dataset to make our prediction and build our algorithms however the work we had to carry out for our workshop was extremely useful in showing us that working with such a large data set can be very difficult and highly cumbersome. During preparation for the workshop we decided that simply selecting the year 2014 to do out project on would suffice as it would not restrict us carrying out the project successfully and would greatly aid us in running our algorithms faster and carrying out much more tests.

After successfully choosing our dataset we took some time to simply study and understand the data. We observed that the data was a mix of categorical and numerical data. The target for our data set was an attribute called *Violation description* which is categorical data an example entry in this attribute is “Speed in zone” which is the name of a speeding offence. This was a clear classification problem. As such we realized we had to come up with a way to manipulate some of our categorical data in a way as to allow us to use many of the standard classification algorithms.

Our dataset going into the data preparation phase had the following information.

- Traffic tickets issued in NYC for the year 2014
- The attributes were
    - Violation Charge code; Violation description; Violation Year; Violation month; Violation day of the week; Age at violation; Gender; State of license; Police agency; Court; Source
- ~2.5 Million lines
- CSV format

3. DATA PREPARATION

3.1 DATA CLEANING

Immediately after selecting the dataset we wanted to use and identifying all the current attributes we began formatting and cleaning the data, before we got any further as we had already noticed some discrepancies and errors in the dataset. All the cleaning and formatting was done with python using pandas library and scikit-learn library.

We began with the simplest items to clean which were the null values in our dataset, removing these would insure that they did not affect the algorithms and made the data slightly easier to manage and process. Using the below function allowed us to efficiently remove null values.

```
#Deal with missing data: Use fillna or dropna()
dataset = dataset.dropna()
```

The next step was to split our data into features and target. The target was to be Violation description and the features were to be everything else. We simply added the relevant data to two lists as shown below.

```
# Split into features and targets
targets = dataset.iloc[:,:1]
features = dataset.iloc[:, dataset.columns != 'ViolationDescription']
```

The next step was mentioned earlier which was to manipulate the categorical data to insure it worked with the various classification algorithm which require continuous data. This was quite a challenge and we had to attempt multiple things in order to get it right. The first method we attempted was to simply change the categorical data to numerical data. We changed the following data.

- "Violation day of the week" attributes had the values "Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday". We mapped each textual value respectively to "1, 2, 3, 4, 5, 6, 7".
- We adopted a similar approach to the "Genders" attribute where we mapped "Male/Female" to 1 or 0.

This approach however we felt wasn't correct and in the process of mapping the values we were compromising their meaning and usefulness in the classification algorithms. Namely, the values for the "Violation day of the week" attribute would be misinterpreted to have certain days represented as a higher value than other days (e.g. Monday < Tuesday < Wednesday). It assumes that the higher the value is the better the category is.

The next approach was to user Label Encoder and One Hot Encoder provided by SciKit Learn library in python. This handled the conversion of categorical data into numbers which the algorithms could better understand and process. One hot encoder library creates a column for each individual day in the week and assigns it either a 0 if it does not appear in the row, or a 1 if it appears in the row. This is demonstrated in the following code.

```
#Deal with categorical data
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
#LabelEncode categorical data
DayLabelEncoder = LabelEncoder()
features['ViolationDayofWeek'] = DayLabelEncoder.fit_transform(features['ViolationDayofWeek'])
genderEncoder = LabelEncoder()
features['Gender'] = genderEncoder.fit_transform(features['Gender'])
#Onehotencoding for Days of week cuz there are more than 2 options
features = pd.concat([features,pd.get_dummies(features['ViolationDayofWeek'], prefix='day')],axis=1)
features = features.drop('ViolationDayofWeek',axis=1)
```

3.2 THE DISPERSION OF DATA

The workshop we completed was the first time we began to use our dataset and begins to understand what is expected of us in this module and project. Evidence of the knowledge we gained in preparing our dataset and understanding it can be found throughout this report. However, it most importantly allowed us to see how dispersed our dataset it.

All the calculations we performed as part of the workshop such as the standard deviation, variance and range showed us high dispersion and variance values in our data. Due to our inexperience and naivety we began trimming many values that we thought were outliers as shown by our calculations and boxplots.

Upon further research and study in the module and our lecturers' multiple warnings about trimming the data too much we realized that we may have made a mistake as our outliers were important in our final prediction. For example, "Speed in zone" traffic offence was the most frequent occurring ticket which we thought was an outlier that would need to be removed however without this traffic offence we would have an invalid prediction due to the over manipulation of the dataset. As such we reverted most of the changes we made in relation to removing outliers during the workshop and kept as much of the data as possible.

3.3 ATTRIBUTE ANALYSIS

As mentioned above the original dataset contained the following attributes

- Violation Charge code; Violation description; Violation Year; Violation month; Violation day of the week; Age at violation; Gender; State of license; Police agency; Court; Source

During our workshop and afterwards as we were preparing the algorithms we quickly realized that in fact not every attribute was necessary for the final prediction and removing them would increase the speed of our algorithms and allow us to analyse a cleaner more compact dataset. In order to select the attributes, we wanted to keep, we had to perform an attribute analysis.

The list of attributes we decided to remove are as follows.

- Source – 100% of this attribute contained the value "TSLED" which is the abbreviation for the traffic safety law enforcement agency in NYC. This had no bearing on the target for our prediction and was removed.
- Court – The values in this attribute contained the name of the court the ticket was answerable in, this also had no effect on the final prediction and was removed.
- Police agency – The values in this attribute contained the name of the police agency that issued the tickets. This had no effect on the final prediction either and was removed.
- State of license – Originally, we thought that the sate a person was issued their license may have a bearing on the likelihood of a driver receiving a ticket however upon further analysis this proved not to be the case and 91% of values in this attribute were "NEW YORK" as such this attribute was removed.
- Violation year – This attribute simply contained the year of the traffic offence, as we were only using the year 2014 this was unnecessary and was removed.

Once we further cleaned our dataset and cut the attributes that were no longer necessary, we had to analyse which attributes we were going to use for our prediction. Initially our project was going to simply attempt to attempt to see If given a drivers age alone could we predict the most likely traffic offence. However, we quickly realized this was not very possible and our workshop and algorithms highlighted this immediately. There were simply too many different traffic offences and the range of ages was 18 – 94, we then had to adjust our course and attempt to analyse and choose the attributes that we could use to make a correct and useful prediction. The following attributes were selected after analysis and work carried out during our workshop.

- Age at violation – This attribute contains the age of drivers at the day the driving offence was committed the range here is 18-94 and it is represented as numerical data. While this attribute alone was not enough to accurately predict and yielded low accuracy scores it is an integral attribute in the final algorithms as the accuracy would dramatically decrease without the age of the driver.
- Gender – This attribute contained the gender of the driver split into Male/Female with 64% of tickets issued to male and 36% of tickets issued to females. We found that the inclusion of this attribute affected our accuracy score very positively as there is a significant difference between the number of male and female drivers received tickets.
- Violation month – This attribute contained the 12 months as categorical data, initially we didn't pay too much attention to this attribute as we didn't suspect that it would affect the final prediction accuracy however upon doing further research we found that there were significant increases in the number of traffic tickets issued in the Christmas periods and summer periods. We then included this attribute in the dataset and this positively impacted the accuracy scores.
- Violation day of the week – This attribute contains the values of the 7 days of the week as categorical data. This was like the violation month attribute as we simply didn't expect it to have a bearing on the results. However, research into the topic showed us that in fact on certain days drivers are more likely to commit offences. For example, during weekends there is a slight increase in the percentage of tickets issued. Due to this discovery we decided to include the violation day of the week into our final dataset.
- Violation description and Violation charged code – Finally this is the target attribute in our algorithm it contains categorical data of the description of the traffic offence such as "Speed in zone" or "Turn without signal". For each violation description there is an equivalent charged code which is in a separate attribute, we kept the violation charged code attribute for cross refencing between the description and the code in case there were any discrepancies.

3.4 DATA NORMALISATION

The final step before we were happy that our dataset was in its final form, we carried out some relevant normalisation techniques. We insured that our data had not been corrupted and no incorrect or null values were introduced as part of the cleaning. We observed that when removing the Violation year attribute, it had created multiple null values and errors in other attributes.

In order to address this, we had to restore our dataset to a previous version and remove the attribute correctly. Once this was completed and our final checks were done, we were happy to begin finalizing our dataset.

3.5 FINAL DATA SET

The final dataset turned out to be quite different from the dataset we started with. We started with a dataset containing 4 years of traffic offences at over 14 million entries. When we first attempted to process this, it was impossible due to the sheer size and our limited computing resources. The workshop first exposed us to some of the challenges we faced and the work for this report allowed us to create this final data set which was useable and relevant to our final prediction. The dataset contains the 6 attributes that will be used by our algorithms, it contains ~2.5 million lines which are all valid and non-corrupted. The dataset was finally complete and ready for our classification program.

4. ALGORITHM DESCRIPTION AND IMPLEMENTATION

Before choosing what type of algorithm we would use we had to first identify what type of problem we were dealing with. We saw that we were dealing with a large dataset with labelled, categorical attributes. Following guidelines for choosing the right algorithm we decided that we were dealing with a classification algorithm, and that supervised learning was the way to go. This was further reinforced by the attribute analysis we had carried out and the knowledge we gained from our workshop.

After identifying that our problem was a classification problem, we began researching numerous classification algorithms so we could find one that would best suit our dataset. The algorithms we shortlisted to try and implement were Naïve Bayes and K nearest neighbour (KNN). We also decided to split for testing and training purposes, 80% was to be for training and 20% was to be for testing.

4.1 NAÏVE BAYES IMPLEMENTATION

The first algorithm we decided to implement was Naïve Bayes. Using the SciKit library allowed us to quickly implement all three variations of the Naïve Bayes algorithm, namely Bernoulli, Multinomial and Gaussian. We decided to first implement this algorithm as compared to KNN it is much faster and initially we over looked the fact that Naïve Bayes assumes conditional independence between he attributes and this would negatively impact our result as shown later in the analysis of the results.

```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score
Bernoulli = BernoulliNB()
Bernoulli.fit(X_train, y_train.values.ravel())
y_pred = Bernoulli.predict(X_test)
print("The accuracy using Bernoulli is: ")
print(accuracy_score(y_test,y_pred))

Multinomial = MultinomialNB()
Multinomial.fit(X_train, y_train.values.ravel())
y_pred = Multinomial.predict(X_test)
print("The accuracy using Multinomial is: ")
print(accuracy_score(y_test,y_pred))

Gaussian = GaussianNB()
Gaussian.fit(X_train, y_train.values.ravel())
y_pred = Gaussian.predict(X_test)
print("The accuracy using Gaussian is: ")
print(accuracy_score(y_test,y_pred))
```

4.2 KNN IMPLEMENTATION

The next classification algorithm we implemented was KNN classifier also using the SciKit python library as shown below. This implantation was the more difficult of the two as we had to carry out multiple tests each of which took a long time due to the nature of the KNN algorithm. Finally, we settled on an appropriate K value to avoid overfitting. This algorithm was not our first choice initially as we felt it would take too long to test due to the large dimensions of our dataset and we thought that our attributes did not need to be dependent.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=9)
knn.fit(X_train, y_train.values.ravel())
y_pred = knn.predict(X_test)
print("The accuracy using KNN is: ")
print(accuracy_score(y_test,y_pred))
```

5. RESULTS AND ANALYSIS

5.1 FINAL RESULTS

Following on from the above section detailing the implementation of the two classification algorithms we chose to use the results are as follows.

```
The accuracy using Bernoulli is:
0.37370461544135275
The accuracy using Multinomial is:
0.3834453541332448
The accuracy using Gaussian is:
0.0019177223359333224
```

MULTINOMIAL NB

This variation of Naïve Bayes gave us the best accuracy with ~38% accuracy. This can be attributed to the fact that it works best with discrete data. This is the form most of our data takes on after it has been converted from categorical to numerical.

GAUSSIAN NB

This variation gives us the lowest accuracy which was much lower than we anticipated at ~0% accuracy. This can be attributed to the fact that Gaussian works best with continuous data that follows a normal distribution, and our data is lacking in any continuous attributes. This is generally a relatively simple algorithm and struggles with very large datasets while we did expect a lower result, this was much lower, and we needed to further analyse the results.

Resulting in an accuracy score below 60% was not what we wanted, nor what we expected. After getting such low results we decided to analyse our choices and see where we went wrong. Some questions that were immediately obvious were;

- Did the existence of outliers negatively affect the classification? - It is possible that outliers lead to unseen observations, thus negatively affecting our accuracy. This was one of the confusing parts of our data preparation phase as we were struggling to determine whether we should remove outliers.
- Was there simply too many unique target values and not enough features to accurately classify an offence.
- Did we need to increase the training size and decrease the testing size?

Before we answered these questions, we made sure that we did not simply choose the wrong algorithm for our dataset, so we decided to backtrack and re-evaluate our choices.

KNN

We wanted to test our hypothesis by running our data through a KNN algorithm. After researching this classifier, we found that the complex decision trees and large number of data points created by KNN could potentially yield a much higher accuracy sore. We ran it with k = 9(adjusting it to this helped with KNNs likelihood to overfit) and this almost doubled the accuracy of the prediction resulting in ~68% accuracy. This proved to us that we had initially chosen the wrong algorithm to begin with.

This result may still be considered low in terms of being an effective classification, getting just over 30% of classifications wrong. Perhaps this was due to the complexity of the data and not enough training data to match that complexity. We also took in to consideration the fact that we had outliers, which in turn is not great for KNN as it is sensitive to outliers. If we removed them we might have seen a higher prediction but we initially aired on the side of caution when dealing with them as it can be bad practice to remove them a lot of the time. Nevertheless, we were happy with this new accuracy score as it is more like what we expected going into this project.

```
The accuracy using KNN is:
0.6840229757887555
```

5.2 WHAT WE LEARNED

After we performed all the calculations and ran out dataset through both the different variations of the Bayes algorithm and the KNN algorithm we are happy that KNN proved to be the best fit for our dataset. Achieving a relatively high result showed us that we should never settle with the first algorithm and continue to trial different algorithms and methods. A downside to using the KNN algorithm for us was its known problem of taking a great amount of time to complete its calculations. While we experienced this issue in our repeated tests to find the optimum value for K in the over all scheme of things it was still the correct algorithm for our dataset providing us an accuracy of 68%.

As we were relatively new to the study of data science and the theory of classifying and predicting events we naively chose 1 attribute to predict 1 outcome. This was out first mistake and the most important lesson, we realized that attempting to predict a traffic offence by simple using the drivers age was unfeasible due to the sheer number of different possible traffic offences and the 2.5 million cases in our dataset. However, committing this mistake allowed us to reflect on the material of the module and study classification in more detail which reinforced our learning. We were fortunate that the dataset also contained more relevant attributes that could be used to provide a meaningful classification and saved us from having to begin again.

During the cleaning phase of the project, we also had issues with over trimming of our dataset. As mentioned above we initially removed outliers in the top percentile such as most frequent traffic offence (Speeding) as we initially thought including this ticket would skew our results and decrease our accuracy. However, we realized that this was simply a characteristic of the data and reverting that change increased our accuracy.

We also had issues with the different data types we were dealing with e.g. Categorical, numerical. Our initial attempts into converting all categorical data to numerical proved to be a mistake as this changed their meaning and made the algorithms behave differently as data values were no longer correctly represented. We then found our solution in the form of a label encoder library provided by SciKit learn which allowed us to convert our data without compromising its meaning. Choosing data sets in the future we would insure that we had less variation of data types as this was quite a difficult problem to overcome and having large number of attributes would make this an even bigger issue. We also believe our accuracy could have been improved if we used a dataset with only continuous data or only categorical and the mix reduced our accuracy.

Finally, we were happy with the size of our data set being over 2.5 million lines allowed us to feed a great amount of data into our algorithms we decided to choose a dataset as large as possible given advice from previous students. However, downsides were that in the beginning it was very difficult to work with the data as the file was quite large and wait times for algorithms and applications using our dataset were lengthy but as it resulted in a favourable accuracy we were happy with the dataset choice and subsequently the choice of the K nearest neighbour classification algorithm.

6. APPENDIX

1. Dataset - https://www.kaggle.com/new-york-state/nys-traffic-tickets-issued-four-year-window
2. SciKitlearn - https://scikit-learn.org/stable/
3. Pandas python library - https://pandas.pydata.org/
4. https://medium.com/@contactsunny/label-encoder-vs-one-hot-encoder-in-machine-learning-3fc273365621

ALGORTIHIM IMPLEMENTATION

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
pd.options.mode.chained_assignment = None

#Import DataSet and drop unused Columns
dataset = pd.read_csv("Tickets2014.csv")
dataset = dataset.drop("ViolationChargedCode",axis=1)
dataset = dataset.drop("ViolationYear",axis=1)

#Deal with missing data: Use fillna or dropna() or imputer version from Udemy
dataset = dataset.dropna()

# Split into features and targets
targets = dataset.iloc[:,:1]
features = dataset.iloc[:, dataset.columns != 'ViolationDescription']

#Deal with categorical data
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
#LabelEncode categorical data
DayLabelEncoder = LabelEncoder()
features['ViolationDayofWeek'] = DayLabelEncoder.fit_transform(features['ViolationDayofWeek'])
genderEncoder = LabelEncoder()
features['Gender'] = genderEncoder.fit_transform(features['Gender'])
#Onehotencoding for Days of week cuz there are more than 2 options
features = pd.concat([features,pd.get_dummies(features['ViolationDayofWeek'], prefix='day')],axis=1)
features = features.drop('ViolationDayofWeek',axis=1)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, targets, test_size=0.20)

from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=9)
knn.fit(X_train, y_train.values.ravel())
y_pred = knn.predict(X_test)
print("The accuracy using KNN is: ")
print(accuracy_score(y_test,y_pred))
```

FINAL DATASET SAMPLE

	ViolationDescription	ViolationMonth	ViolationDay	Age	Gender
2	INSUFF TURN SIGNAL-LESS THAN 100 FEET	12	SUNDAY	56	F
3	DRIVING WHILE INTOXICATED	12	SUNDAY	56	F
4	DRIVING W/.08 OF 1 PERCENT OF ALCO/BLD	12	SUNDAY	56	F
5	SPEED OVER 55 ZONE	5	MONDAY	24	F
6	OPERATING MV MOBILE PHONE	4	MONDAY	35	M
7	UNLICENSED OPERATOR	12	WEDNESDAY	23	M
8	SPEED NOT REASONABLE AND PRUDENT	12	WEDNESDAY	23	M
9	MOVED FROM LANE UNSAFELY/WEAVING	12	WEDNESDAY	23	M
10	NO SEAT BELT ADULT	12	SATURDAY	48	M
11	NO SEAT BELT ADULT	12	SATURDAY	43	M
12	SPEED OVER 55 ZONE	12	SATURDAY	39	M
13	OPERATING MV MOBILE PHONE	12	MONDAY	30	M
14	SPEED OVER 55 ZONE	12	MONDAY	50	F
15	NO/INADEQUATE HEADLAMPS	12	MONDAY	53	F
16	INADEQUATE OR NO STOP LAMPS	12	FRIDAY	41	F
17	SPEED NOT REASONABLE AND PRUDENT	11	TUESDAY	60	M
18	SPEED IN ZONE	11	SUNDAY	22	M
19	SPEED IN ZONE	11	SUNDAY	26	M
20	NO/INADEQUATE HEADLAMPS	11	SUNDAY	40	M
21	FLD DUE CARE FOR EMERG VEH STOPPED OR STA	11	SATURDAY	63	F
22	FLD DUE CARE FOR EMERG VEH STOPPED OR STA	11	SATURDAY	50	F
23	SPEED IN ZONE	11	SATURDAY	24	F
24	OPERATING MV MOBILE PHONE	11	SATURDAY	66	F
25	INADEQUATE OR NO STOP LAMPS	11	FRIDAY	78	M
26	NO SEAT BELT ADULT	10	WEDNESDAY	30	M
27	NO SEAT BELT ADULT	10	WEDNESDAY	46	F
28	SPEED IN ZONE	10	TUESDAY	18	M