DALHOUSIE UNIVERSITY

Faculty of Computer Science

CSCI 6515 – Machine Learning for Big Data

Name: Arka Ghosh
Banner ID: B00911033
Assignment: 01

Task 1

In this assignment, I have worked with two datasets [1] [2] which contains object, integer and floating values as data. I have referred the dataset [1] as PM_dataset and dataset [2] as Traffic_dataset in my code. The explanation and the data types of both the dataset's features has been described in the following screenshots:

PM Dataset:

Features	Data Type	Explanation
Date & time	Object (MM/DD/YYYY HH:00:00 AM/PM)	Date & time of the data recorded
Pollutant	Object	Pollutant particle
Unit	Object	Unit of the pollutant particle
Station	Object	Place from where the data is recorded
Instrument	Object	Instrument collect the data
Average	Float	Average of the pollutant particle recorded for a particular hour of a day

```
In [1688]: PM_Dataset.dtypes

Out[1688]: Date & time    object
           Pollutant      object
           Unit           object
           Station        object
           Instrument     object
           Average       float64
           dtype: object
```

Figure 1: Data type of PM Dataset

Traffic Dataset:

Features	Data Type	Explanation
SECTION ID	Integer	Each section is allocated with an ID number
HIGHWAY	Integer	The name of the highway from where the data is recorded
SECTION	Integer	The section from where the data is recorded
SECTION LENGTH	Float	Length of the section
SECTION DESCRIPTION	Object	Description of the Sections
DATE	Object	Date of the data recorded
DESCRIPTION	Object	Distance away from the sections
GROUP	Object	Grouping of the ADT and AADT according to their seasonal patterns
TYPE	Object	Vehicle classification
COUNTY	Object	Different regions of Canada where the data is recorded

```
In [1532]: print(Traffic_Dataset.dtypes)

           SECTION ID               int64
           HIGHWAY                  int64
           SECTION                  int64
           SECTION LENGTH         float64
           SECTION DESCRIPTION     object
           Date                    object
           DESCRIPTION             object
           GROUP                   object
           TYPE                    object
           COUNTY                  object
           PTRUCKS                float64
           ADT                    float64
           AADT                   float64
           DIRECTION               object
           85PCT                  float64
           PRIORITY_POINTS        float64
           dtype: object
```

Figure 2: Data type of Traffic Dataset

Features	Data Type	Explanation
PTRUCKS	Float	Percentage of the trucks passing
ADT	Float	Average Daily Traffic based on vehicles passing the location in 24 hours
AADT	Float	Average Annual Daily Traffic based on vehicles passing the location in 24 hours, averaged on the basis of a year
DIRECTION	Object	Direction the vehicles are travelling
85PCT	Float	Speed at which 85% vehicles are passing
PRIORITY_POINTS	Float	Signal Analysis points

Task 2

Raw datasets have been collected from the mentioned resources. From the “Date & time” column of PM2.5 Dataset, the data of the year 2019 has been filtered first using the dt.year.eq() function. The “Pollutant”, “Unit”, “Station” and “Instrument” columns from this datasets have been filtered out as these values will not add any importance in case of training the model. Then, the average data has been computed using the mean() function for each date of 2019 which has been shown below:

Out[1526]:

Date	Average
01/01/2019	3.083333
01/02/2019	2.625000
01/03/2019	5.625000
01/04/2019	5.136364
01/05/2019	8.208333
...	...

After computing the average for each date, the average values have been normalized using min-max normalization which is:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \qquad [3]$$

Here, x’ represents the new normalized value of average and x is the average value of each date. x_{min} and x_{max} denotes the minimum and maximum value of the average respectively. **If the new normalized value is greater than 0.3 threshold, it will be labeled as “High” or “Low” otherwise.** The dataset is not uniformly balanced when threshold is set to 0.5. Most of normalized values are less than 0.5 for which the dataset becomes unbalanced and most of the data gets labeled as “Low” which has been showed in the following screenshot:

```
When threshold is set to 0.3:

Out[1697]: Low     245
           High    120
           Name: PM_Level, dtype: int64
```

```
When threshold is set to 0.5:

Out[1713]: Low     350
           High     15
           Name: PM_Level, dtype: int64
```

The visual summarization of the normalized average value of 2019 has been plotted in **Figure 3**.

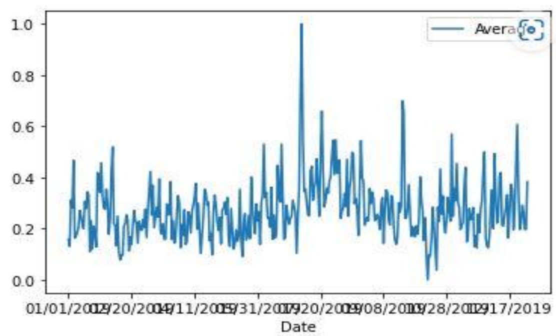


Figure 3: Normalized value of PM2.5 Average

From the Traffic dataset, I have also filtered the dataset only to represent the data recorded for the Halifax region in 2019. After filtration, the Traffic Dataset count came to 168 for Halifax region in 2019.

Task 3

From the PM Dataset, I have filtered out “Pollutant”, “Unit”, “Station” and “Instrument” as these columns consist Nominal Categorical data having at most two classes and there is no intrinsic ordering to the categories. I have only used the Average feature from the PM Dataset as it consists of numeric data and is directly related to deciding the “High” or “Low” threshold of PM level. I have used the “HIGHWAY”, “SECTION”, “SECTION LENGTH”, “ADT” and “AADT” from the Traffic Dataset as other features are mostly Nominal Categorical data which have a minimal chance of making a real impact on the model to fit. For example, the “COUNTY” feature has only one variable “HFX”. Due to its extremely low variance, features with such variables do not have any beneficial effect on model performance. In addition to that, most algorithms including ML libraries yield superior results with numerical variables [5]. For the final dataset, I have also dropped the Date from the dataset as it won’t add any significance in training the model. **Figure 4** demonstrates the final dataset that been created from the PM and Traffic dataset where “HIGHWAY”, “SECTION”, “SECTION LENGTH”, “ADT” and “AADT” are the Predictor Variables and “PM_Level” is the Target Variable. Target Variables are the values modeled and predicted by other variables, and variables whose values are used to forecast the value of the target value is called Predictor Variables [6].

	HIGHWAY	SECTION	SECTION LENGTH	ADT	AADT	Average	PM_Level
0	1	47	4.50	2566.0	2430.0	0.271403	Low
1	101	20	3.71	23205.0	22000.0	0.232488	Low
2	101	20	3.71	23385.0	22100.0	0.232488	Low

Figure 4: Final Dataset after Filtration and Merging

Classification tasks in ML refer to the prediction of the class of a set of data points. Approximating a mapping function (f) from input variables (X) to discrete output variables (Y) is the function of classification predictive modelling [5]. Moreover, decision tree is more suitable for the dataset which has been used for this assignment. Scatter plots have been generated in **Figure 5** to identify patterns and relationships among the variables in the dataset. Given these scatterplot figures, the X-axis is the independent variable which and the Y-axis presents the response [7]. Scatter plots have been generated for all of features columns individually to show the relationship between the feature and target variable. From the scatter plots generated in Figure 2, it is understandable that the dataset is non-linear as there is no clear pattern between the data points. Decision Trees are well suited for handling non-linear dataset effectively [8].

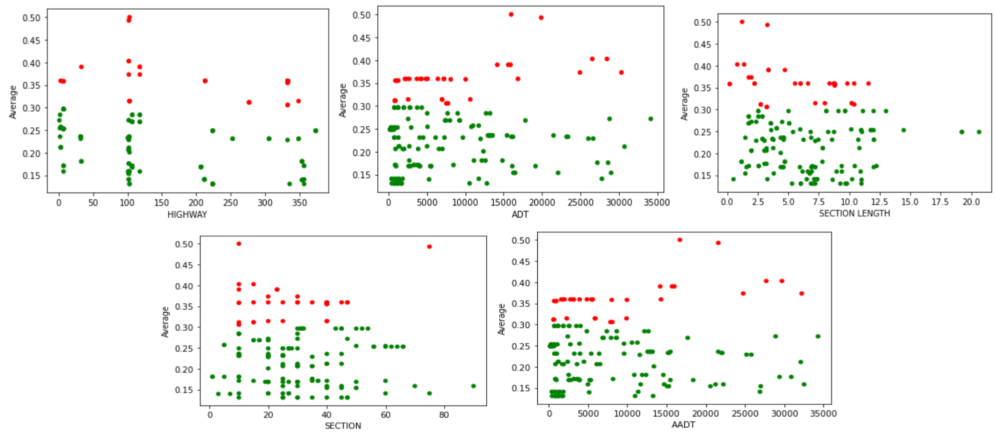


Figure 5: Scatter Plots of the Features

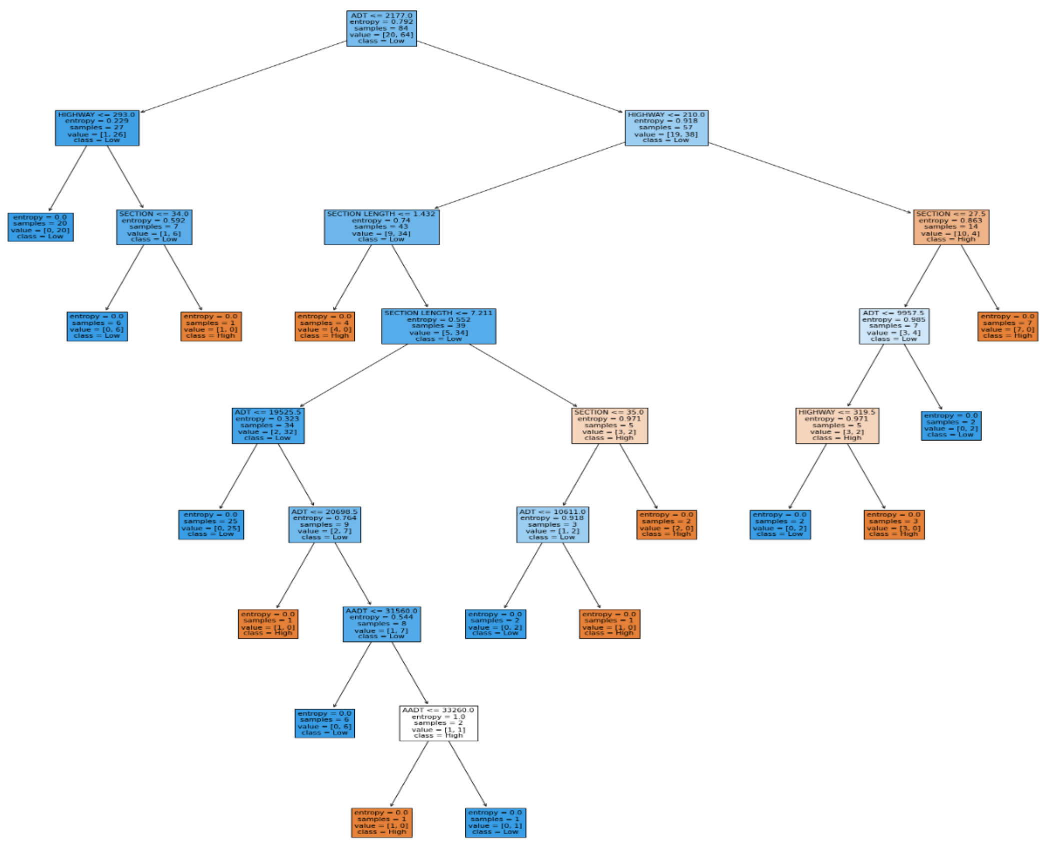


Figure 6: Generated Decision Tree

Task 4

(i) In the DecisionTreeClassifier() model, I used the criterion parameter “entropy” which is for Shanon Information Gain to measure the quality of the split and the splitter parameter is set to “best”. The most importance feature has been provided by the fitted attribute clf.feature_importances_ of python library Scikitlearn. According to clf.feature_importances_, the feature ADT is the most influential feature factor for PM2.5 level which is calculated as the mean and standard deviation of accumulation of the impurity decrease inside each tree [9]. The feature importance has been shown in the following **Figure 7.**

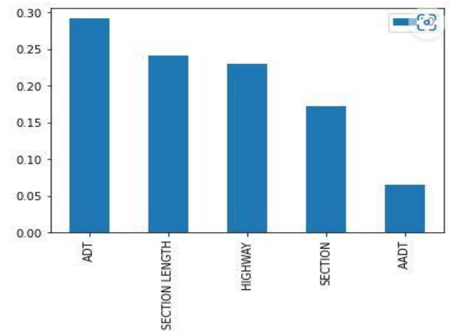


Figure 7: Feature Importance

(ii) The formula used for the calculating information gain of the root node is in the following:

$$Entropy, E = -\sum_{i=1}^{N} p_i log_2 p_i$$

$$Gain = E_{parent} - E_{Children}$$

First, the entropy of the root node needs to be calculated which has been denoted as E_{parent} in the following formula:

$$E_{Parent} = -\frac{20}{84} log_2 \frac{20}{84} - \frac{64}{84} log_2 \frac{64}{84} = 0.792$$

The parent has been split into two children node. The left children node has been denoted as E_{left} and the right children node has been denoted as E_{right} in the following formula to calculate the entropy:

$$E_{left} = -\frac{1}{27} log_2 \frac{1}{27} - \frac{26}{27} log_2 \frac{26}{27} = 0.229$$

$$E_{right} = -\frac{19}{57} log_2 \frac{19}{57} - \frac{38}{57} log_2 \frac{38}{57} = 0.918$$

$$Weighted\ Average\ Entropy\ of\ Children\ Nodes = \frac{27}{84} * 0.229 + \frac{57}{84} * 0.918 = 0.697$$

$$Gain = 0.792 - 0.697 = 0.095$$

So, the information gain for the root node (ADT) is **0.095**.

(iii) (a) The target variable “PM_Level” is categorical value for which it is encoded with a unique integer using the sckit-learn library. As label encoding uses the alphabetical order, “High” is encoded with 0 and “Low” is encoded with 1 in the target column.

After apportioning the data into train and test sets with 50-50 split, the classification report and the confusion matrix has been mentioned below:

```
In [436]: print(classification_report(y_test, y_testpred))

              precision    recall  f1-score   support

           0       0.41      0.44      0.42        16
           1       0.87      0.85      0.86        68

    accuracy                           0.77        84
   macro avg       0.64      0.65      0.64        84
weighted avg       0.78      0.77      0.78        84
```

```
In [435]: print("Testing Confusion Matrix: \n", confusion_matrix(y_test, y_testpred))

Testing Confusion Matrix:
 [[ 7  9]
 [10 58]]
```

The visualization for the Decision Tree has been added above in (ii) part.

(b) After applying 10 fold cross validation technique, the classification report for all 10 folds, average of the evaluation metrics and confusion matrix have been provided in the **Figure 8 and 9** respectively:

```
Fold 1
              precision    recall  f1-score   support

           0       0.22      0.67      0.33         3
           1       0.88      0.50      0.64        14

    accuracy                           0.53        17
   macro avg       0.55      0.58      0.48        17
weighted avg       0.76      0.53      0.58        17

Fold 2
              precision    recall  f1-score   support

           0       0.50      0.50      0.50         2
           1       0.93      0.93      0.93        15

    accuracy                           0.88        17
   macro avg       0.72      0.72      0.72        17
weighted avg       0.88      0.88      0.88        17

Fold 3
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.82      1.00      0.90        14

    accuracy                           0.82        17
   macro avg       0.41      0.50      0.45        17
weighted avg       0.68      0.82      0.74        17

Fold 4
              precision    recall  f1-score   support

           0       0.67      0.67      0.67         3
           1       0.93      0.93      0.93        14

    accuracy                           0.88        17
   macro avg       0.80      0.80      0.80        17
weighted avg       0.88      0.88      0.88        17

Fold 5
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.82      1.00      0.90        14

    accuracy                           0.82        17
   macro avg       0.41      0.50      0.45        17
weighted avg       0.68      0.82      0.74        17

Fold 6
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         5
           1       0.71      1.00      0.83        12

    accuracy                           0.71        17
   macro avg       0.35      0.50      0.41        17
weighted avg       0.50      0.71      0.58        17

Fold 7
              precision    recall  f1-score   support

           0       0.50      0.80      0.62         5
           1       0.89      0.67      0.76        12

    accuracy                           0.71        17
   macro avg       0.69      0.73      0.69        17
weighted avg       0.77      0.71      0.72        17

Fold 8
              precision    recall  f1-score   support

           0       0.56      0.50      0.53        10
           1       0.38      0.43      0.40         7

    accuracy                           0.47        17
   macro avg       0.47      0.46      0.46        17
weighted avg       0.48      0.47      0.47        17

Fold 9
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.86      0.86      0.86        14

    accuracy                           0.75        16
   macro avg       0.43      0.43      0.43        16
weighted avg       0.75      0.75      0.75        16

Fold 10
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        16

    accuracy                           1.00        16
   macro avg       1.00      1.00      1.00        16
weighted avg       1.00      1.00      1.00        16
```

Figure 8: Classification Report of 10 Folds

```
Out[450]: array([[ 3,  3],
                 [ 3, 11]], dtype=int64)

The average Precision is: 0.8388103318250376
The average Recall is: 0.6582417582417583
The average F1 score is: 0.6910952755831052
```

Figure 9: Average Confusion Matrix and Classification Report of 10 Folds

(c) The model makes sense. After apportioning the data into train and test sets with 50-50 split, the training accuracy becomes 100% and the testing accuracy becomes 77.38% for the Decision Tree Classifier. The model is overfitted as the model works well on the training dataset but does poorly with the testing test. It means the model can recall the data patterns from the training dataset but cannot generalize to new samples. One of the reasons of overfitting for the model might be the insufficient number of data that has been used for training the data as the initial number of data counts were only 168 after filtration and merging. Out of 168 records, only 84 is used for training the model which is not much.

Also, there are leaf nodes which are small.

(d) Both the Precision and Recall worked well for the target variable 1 (Low). It is because there are higher amount of instances in the final dataset labelled as "Low". After proportioning the data into train and test sets with 50-50 split and feeding the input through the classifier, it can be seen from the classification report that the recall and precision for the target variable 1 is 0.87 and 0.85 respectively which is comparatively higher. High scores for both indicate that the classifier is producing results that are accurate (high precision) and that are mostly positive (high recall).

(e) An experiment with five different values of max_depth, min_samples_leaf and min_samples_split has been conducted and the impact of these parameters has been described below:

max_depth

The default value of this parameter is None. Five different values (2, 4, 6, 8, 10) have been used for the parameter. This indicates how tall the tree may grow. The deeper the tree, the more splits it has and the more data it can hold. From the code, it can be seen that when the max_depth is set to 2, it gives the highest F1-score which can combat the overfitting problem [15].

min_samples_leaf

The default value of this parameter is 1. Five different values (5, 15, 25, 35, 45) have been used for the parameter. This parameter represents the bare minimum of samples that must be present at a leaf node [14]. It is noticeable from the code that increasing value of this parameter yields comparatively better Accuracy, Recall and Precision score which means increasing the value of min_samples_leaf can combat overfitting of the model.

min_samples_split

The default value of this parameter is 2. Five different values (5, 15, 25, 35, 45) have been used for the parameter. The minimal number of samples needed to separate an internal node is represented by this parameter. The tree is more limited when this value is increased since it must take into account more samples at each node [14]. It can be seen from the code that when the value of the parameter is increased, Recall and Precision score decreases gradually. This means increasing the value causes underfitting.

Summary of the Results

Data Pre-Processing

First, I have downloaded the datasets from the resources mentioned. For the pre-processing the dataset, I have filtered out the time from the "Date & time" column from the "PM_Dataset" and renamed it to "Date" as there are data of different hours of per day which will be troublesome in computing the average of each day. Then, I have took the records of year 2019 and filtered out the columns from the dataset which contain categorical nominal values. Then, I have computed the average of the each date by groupby("Date").mean() function. After that, I have normalized the data using min-max normalization and used 0.3 as a threshold value to categorize the average into "Low" and "High". After the filtration and processing of the dataset, the data count reaches to 365 where 245 records are labelled as "Low" and 120 records are labelled as "High".

From the traffic dataset, I have filtered the data first so that it only represents the data of Halifax Region of 2019. After that, I have filtered out the columns from the dataset which contains only categorical nominal values as these values won't have any significant importance in training the model. Then, both of the datasets has been merged using left merge where the rows of the PM_label have been merged with the traffic dataset as per day.

Model Evaluation

After filtration and merging, the "PM_Label" has been encoded into numerical values using label encoder of scikit-learn and stored the new numerical values in a new column named "label". The final dataset contains six headers ""HIGHWAY", "SECTION" , "SECTION LENGTH", "ADT", "AADT" and "label" among which the first five variables are the features and "label" is the target variable of the model. The dataset is apportioned into training and testing dataset with 50-50 split. Then, the DecisionTreeClassifier() model is trained and tested using the dataset which yields overall **100%** accuracy on the training dataset and **77.38%** accuracy on the testing dataset.

Another model was built applying the 10-fold cross validation with the dataset and the accuracy of the 10 folds is 0.82352941, 0.88235294, 0.82352941, 0.64705882, 0.82352941, 0.70588235, 0.70588235, 0.76470588, 0.875, and 1.0 respectively. The mean accuracy of 10 fold is **80.51%** with a standard deviation of **0.098**.

Experiments with Parameters

Five different values of max_depth, min_samples_leaf and min_samples_split parameters have been used to see how these parameters impact the model. It was observed that when the increasing value of the max_depth and min_samples_leaf had an overall better accuracy which can combat overfitting of the model. On the other hand, the model got underfitted when I have increased the value of min_samples_split.

References

[1] [Online]. Available: https://data.novascotia.ca/Environment-and-Energy/Nova-Scotia-Provincial-Ambient-Fine-Particulate-Ma/ddk3-mz42/data. [Accessed: 18- Sep- 2022].

[2] *Data.novascotia.ca*. [Online]. Available: https://data.novascotia.ca/Roads-Driving-and-Transport/Traffic-Volumes-Provincial-Highway-System/8524-ec3n. [Accessed: 18- Sep- 2022].

[3] A. Bhandari, "Feature Scaling | Standardization Vs Normalization", *Analytics Vidhya*. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/. [Accessed: 21- Sep- 2022].

[4] "DATA PREPROCESSING: Decreasing Categories in Categorical Data", *Medium*. [Online]. Available: https://medium.com/analytics-vidhya/data-preprocessing-decreasing-categories-in-categorical-data-132e8b4a4fd. [Accessed: 27- Sep- 2022].

[5] S. Ray, "How to Deal With Categorical Variable in Predictive Modeling", *Analytics Vidhya*, 2015. [Online]. Available: https://www.analyticsvidhya.com/blog/2015/11/easy-methods-deal-categorical-variables-predictive-modeling/. [Accessed: 28- Sep- 2022].

[6] "Classes and Types of Variables", *Dtreg.com*. [Online]. Available: https://www.dtreg.com/solution/classes-and-types-of-variables. [Accessed: 28- Sep- 2022].

[7] "Reading scatterplots - MathBootCamps", *MathBootCamps*. [Online]. Available: https://www.mathbootcamps.com/reading-scatterplots/. [Accessed: 28- Sep- 2022].

[8] "Decision Tree", *Corporate Finance Institute*. [Online]. Available: https://corporatefinanceinstitute.com/resources/knowledge/other/decision-tree/. [Accessed: 28- Sep- 2022].

[9] "Feature importances with a forest of trees", *scikit-learn*. [Online]. Available: https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html. [Accessed: 29- Sep- 2022].

[10] "sklearn.tree.DecisionTreeClassifier", *scikit-learn*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. [Accessed: 29- Sep- 2022].

[11] *A Simple Explanation of Information Gain and Entropy*, 2022. [Online]. Available: https://victorzhou.com/blog/information-gain/. [Accessed: 29- Sep- 2022].

[12] *Courses.cs.washington.edu*. [Online]. Available: https://courses.cs.washington.edu/courses/cse455/10au/notes/InfoGain.pdf. [Accessed: 29- Sep- 2022].

[13] "What is Overfitting in Deep Learning [+10 Ways to Avoid It]", *V7labs.com*, 2022. [Online]. Available: https://www.v7labs.com/blog/overfitting. [Accessed: 30- Sep- 2022].

[14] "InDepth: Parameter tuning for Decision Tree", *Medium*, 2017. [Online]. Available: https://medium.com/@mohtedibf/indepth-parameter-tuning-for-decision-tree-6753118a03c3. [Accessed: 03- Oct- 2022].

[15] "4 Useful techniques that can mitigate overfitting in decision trees", *Medium*, 2021. [Online]. Available: https://towardsdatascience.com/4-useful-techniques-that-can-mitigate-overfitting-in-decision-trees-87380098bd3c#:~:text=max_depth%3A%20Specifies%20the%20maximum%20depth,Decreasing%20this%20value%20prevents%20overfitting. [Accessed: 03- Oct- 2022].

CSCI 6515 - Assignment 1

In [238…

```
import numpy as np
import pandas as pd
```

Data Pre-Processing

In [239…

```
PM_Dataset = pd.read_csv("Nova_Scotia_Provincial_Ambient_Fine_Particulate_Matter__PM2.5__F
PM_Dataset
```

Out[239…

	Date & time	Pollutant	Unit	Station	Instrument	Average
0	01/25/2021 11:00:00 AM	PM2.5	µg/m3	Halifax Johnston	API T640	3.1
1	01/25/2021 11:00:00 PM	PM2.5	µg/m3	Halifax Johnston	API T640	3.4
2	01/25/2021 12:00:00 AM	PM2.5	µg/m3	Halifax Johnston	API T640	NaN
3	01/25/2021 12:00:00 PM	PM2.5	µg/m3	Halifax Johnston	API T640	3.3
4	01/26/2006 01:00:00 AM	PM2.5	µg/m3	Halifax	BAM 1020	NaN
...	...	...	...	...	...	...
140250	12/31/2021 10:00:00 PM	PM2.5	µg/m3	Halifax Johnston	API T640	5.1
140251	12/31/2021 11:00:00 AM	PM2.5	µg/m3	Halifax Johnston	API T640	6.3
140252	12/31/2021 11:00:00 PM	PM2.5	µg/m3	Halifax Johnston	API T640	4.7
140253	12/31/2021 12:00:00 AM	PM2.5	µg/m3	Halifax Johnston	API T640	6.1
140254	12/31/2021 12:00:00 PM	PM2.5	µg/m3	Halifax Johnston	API T640	6.8

140255 rows × 6 columns

In [240…

```
PM_Dataset.dtypes
```

Out[240…

```
Date & time      object
Pollutant        object
Unit             object
Station          object
Instrument       object
Average         float64
dtype: object
```

In [241…

```
#Pre-processing the PM_Dataset for the records of year 2019
PM_Dataset['Date & time'] = pd.to_datetime(PM_Dataset['Date & time'])
PM_Dataset=PM_Dataset[PM_Dataset['Date & time'].dt.year.eq(2019)]
PM_Dataset['Date & time']=PM_Dataset['Date & time'].dt.strftime('%m/%d/%Y')
PM_Dataset = PM_Dataset.rename(columns={"Date & time" : "Date"})
PM_Dataset
```

```
C:\Users\User\AppData\Local\Temp/ipykernel_12088/3151908078.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
```

```
ide/indexing.html#returning-a-view-versus-a-copy
  PM_Dataset['Date & time']=PM_Dataset['Date & time'].dt.strftime('%m/%d/%Y')
```

Out[241…

	Date	Pollutant	Unit	Station	Instrument	Average
326	01/01/2019	PM2.5	µg/m3	Halifax Johnston	BAM 1020	10.0
327	01/01/2019	PM2.5	µg/m3	Halifax Johnston	BAM 1020	0.0
328	01/01/2019	PM2.5	µg/m3	Halifax Johnston	BAM 1020	8.0
329	01/01/2019	PM2.5	µg/m3	Halifax Johnston	BAM 1020	1.0
330	01/01/2019	PM2.5	µg/m3	Halifax Johnston	BAM 1020	5.0
...	...	...	...	...	...	...
140202	12/31/2019	PM2.5	µg/m3	Halifax Johnston	API T640	6.2
140203	12/31/2019	PM2.5	µg/m3	Halifax Johnston	API T640	7.1
140204	12/31/2019	PM2.5	µg/m3	Halifax Johnston	API T640	9.2
140205	12/31/2019	PM2.5	µg/m3	Halifax Johnston	API T640	6.8
140206	12/31/2019	PM2.5	µg/m3	Halifax Johnston	API T640	7.4

8760 rows × 6 columns

In [242…

```
PM_Dataset.drop(PM_Dataset.iloc[:, 1:5], inplace = True, axis = 1)
PM_Dataset
```

Out[242…

	Date	Average
326	01/01/2019	10.0
327	01/01/2019	0.0
328	01/01/2019	8.0
329	01/01/2019	1.0
330	01/01/2019	5.0
...	...	...
140202	12/31/2019	6.2
140203	12/31/2019	7.1
140204	12/31/2019	9.2
140205	12/31/2019	6.8
140206	12/31/2019	7.4

8760 rows × 2 columns

In [243…

```
#Computing the average PM Level of each date
PM_Dataset = PM_Dataset.groupby("Date").mean()
PM_Dataset
```

Out[243…

	Average
Date	
01/01/2019	3.083333

	Average
Date	
01/02/2019	2.625000
01/03/2019	5.625000
01/04/2019	5.136364
01/05/2019	8.208333
...	...
12/27/2019	5.295833
12/28/2019	4.850000
12/29/2019	3.712500
12/30/2019	3.754167
12/31/2019	6.812500

365 rows × 1 columns

In [244…

```
#Normalizing the Average
PM_Dataset["Average"] = (PM_Dataset["Average"] - PM_Dataset["Average"].min()) / (PM_Datase
PM_Dataset
```

Out[244…

	Average
Date	
01/01/2019	0.158172
01/02/2019	0.130555
01/03/2019	0.311323
01/04/2019	0.281880
01/05/2019	0.466985
...	...
12/27/2019	0.291489
12/28/2019	0.264625
12/29/2019	0.196083
12/30/2019	0.198594
12/31/2019	0.382877

365 rows × 1 columns

In [245…

```
PM_Dataset["PM_Level"] = np.where(PM_Dataset["Average"]>=0.3, "High", "Low")
PM_Dataset
```

Out[245…

	Average	PM_Level
Date		
01/01/2019	0.158172	Low

	Average	PM_Level
Date		
01/02/2019	0.130555	Low
01/03/2019	0.311323	High
01/04/2019	0.281880	Low
01/05/2019	0.466985	High
...	...	...
12/27/2019	0.291489	Low
12/28/2019	0.264625	Low
12/29/2019	0.196083	Low
12/30/2019	0.198594	Low
12/31/2019	0.382877	High

365 rows × 2 columns

In [246…

```
PM_Dataset['PM_Level'].value_counts()
```

Out[246…

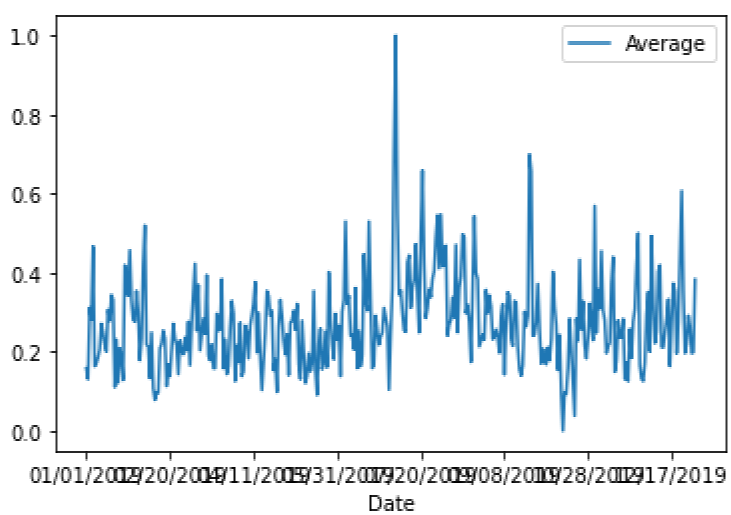
```
Low     245
High    120
Name: PM_Level, dtype: int64
```

In [247…

```
PM_Dataset[["Average"]].plot()
```

Out[247…

```
<AxesSubplot:xlabel='Date'>
```



In [248…

```
Traffic_Dataset = pd.read_csv("Traffic_Volumes_-_Provincial_Highway_System.csv", sep = ",'
Traffic_Dataset
```

Out[248…

	SECTION ID	HIGHWAY	SECTION	SECTION LENGTH	SECTION DESCRIPTION	Date	DESCRIPTION	GROUP	TYPE	COUNTY

	SECTION ID	HIGHWAY	SECTION	SECTION LENGTH	SECTION DESCRIPTION	Date	DESCRIPTION	GROUP	TYPE	COUNTY
0	1047	1	47	4.50	PATTON RD (SACKVILLE) TO MOUNT UNIACKE CONN	05/27/2021	0.5 KM EAST OF BRUSHY HILL RD	A	TC	HFX
1	1047	1	47	4.50	PATTON RD (SACKVILLE) TO MOUNT UNIACKE CONN	05/27/2021	JUST WEST OF PATTON RD (WB)	A	VC	HFX
2	1047	1	47	4.50	PATTON RD (SACKVILLE) TO MOUNT UNIACKE CONN	05/27/2021	JUST WEST OF PATTON RD (EB)	A	VC	HFX
3	1047	1	47	4.50	PATTON RD (SACKVILLE) TO MOUNT UNIACKE CONN	11/24/2020	0.5 KM EAST OF BRUSHY HILL RD	A	TC	HFX
4	1047	1	47	4.50	PATTON RD (SACKVILLE) TO MOUNT UNIACKE CONN	09/09/2019	0.5 KM EAST OF BRUSHY HILL RD	A	TC	HFX
...	...	...	...	...	...	...	...	...	...	...
9545	8050	8	50	5.36	RIVER RD (MILTON) TO LIVERPOOL TOWN LINE	06/17/2015	0.25 KM NORTH OF HWY 103 INTER/C (NB)	C	VC	QUE
9546	8050	8	50	5.36	RIVER RD (MILTON) TO LIVERPOOL TOWN LINE	07/19/2012	0.25 KM NORTH OF HWY 103 INTER/C	C	TC	QUE
9547	8050	8	50	5.36	RIVER RD (MILTON) TO LIVERPOOL TOWN LINE	08/19/2009	0.25 KM NORTH OF HWY 103 INTER/C	C	TC	QUE
9548	8050	8	50	5.36	RIVER RD (MILTON) TO LIVERPOOL TOWN LINE	06/21/2006	0.25 KM NORTH OF HWY 103 INTER/C (SB)	C	VC	QUE
9549	8050	8	50	5.36	RIVER RD (MILTON) TO LIVERPOOL TOWN LINE	06/21/2006	0.25 KM NORTH OF HWY 103 INTER/C (NB)	C	VC	QUE

9550 rows × 16 columns

In [249…

```
print(Traffic_Dataset.dtypes)
```

```
SECTION ID                  int64
HIGHWAY                     int64
```

```
SECTION                     int64
SECTION LENGTH            float64
SECTION DESCRIPTION        object
Date                       object
DESCRIPTION                object
GROUP                      object
TYPE                       object
COUNTY                     object
PTRUCKS                   float64
ADT                       float64
AADT                      float64
DIRECTION                  object
85PCT                     float64
PRIORITY_POINTS           float64
dtype: object
```

In [250…

```
Traffic_Dataset["Date"] = pd.to_datetime(Traffic_Dataset["Date"])
print(Traffic_Dataset.dtypes)
```

```
SECTION ID                              int64
HIGHWAY                                 int64
SECTION                                 int64
SECTION LENGTH                        float64
SECTION DESCRIPTION                    object
Date                           datetime64[ns]
DESCRIPTION                            object
GROUP                                  object
TYPE                                   object
COUNTY                                 object
PTRUCKS                               float64
ADT                                   float64
AADT                                  float64
DIRECTION                              object
85PCT                                 float64
PRIORITY_POINTS                       float64
dtype: object
```

In [251…

```
#Pre-processing the Data for the records of Halifax Region
Traffic_Dataset= Traffic_Dataset[Traffic_Dataset["COUNTY"].eq("HFX")]
Traffic_Dataset
```

Out[251…

	SECTION ID	HIGHWAY	SECTION	SECTION LENGTH	SECTION DESCRIPTION	Date	DESCRIPTION	GROUP	TYPE	COUNTY	PT
0	1047	1	47	4.50	PATTON RD (SACKVILLE) TO MOUNT UNIACKE CONN	2021-05-27	0.5 KM EAST OF BRUSHY HILL RD	A	TC	HFX	
1	1047	1	47	4.50	PATTON RD (SACKVILLE) TO MOUNT UNIACKE CONN	2021-05-27	JUST WEST OF PATTON RD (WB)	A	VC	HFX	
2	1047	1	47	4.50	PATTON RD (SACKVILLE) TO MOUNT UNIACKE CONN	2021-05-27	JUST WEST OF PATTON RD (EB)	A	VC	HFX	
3	1047	1	47	4.50	PATTON RD (SACKVILLE) TO MOUNT UNIACKE CONN	2020-11-24	0.5 KM EAST OF BRUSHY HILL RD	A	TC	HFX	

	SECTION ID	HIGHWAY	SECTION	SECTION LENGTH	SECTION DESCRIPTION	Date	DESCRIPTION	GROUP	TYPE	COUNTY	PT
4	1047	1	47	4.50	PATTON RD (SACKVILLE) TO MOUNT UNIACKE CONN	2019-09-09	0.5 KM EAST OF BRUSHY HILL RD	A	TC	HFX	
...	...	...	...	...	...	...	...	...	...	...	
9344	7066	7	66	11.77	MOOSEHEAD RD (MOOSEHEAD) TO GUYSBOROUGH-HALIFA...	2009-10-20	1 KM EAST OF NAUGLERS SETTLEMENT RD	D	TC	HFX	
9345	7066	7	66	11.77	MOOSEHEAD RD (MOOSEHEAD) TO GUYSBOROUGH-HALIFA...	2007-06-20	AT CIVIC # 28520 (MOSER RIVER) WESTBOUND	D	VC	HFX	
9346	7066	7	66	11.77	MOOSEHEAD RD (MOOSEHEAD) TO GUYSBOROUGH-HALIFA...	2007-06-20	AT CIVIC # 28520 (MOSER RIVER) EASTBOUND	D	VC	HFX	
9347	7066	7	66	11.77	MOOSEHEAD RD (MOOSEHEAD) TO GUYSBOROUGH-HALIFA...	2006-10-17	AT CIVIC # 28520 (MOSER RIVER)	D	TC	HFX	
9348	7066	7	66	11.77	MOOSEHEAD RD (MOOSEHEAD) TO GUYSBOROUGH-HALIFA...	2005-05-12	1 KM EAST OF MOOSEHEAD	D	TC	HFX	

2165 rows × 16 columns

In [252...

```
#Filtering out the traffic data of year 2019
Traffic_Dataset=Traffic_Dataset[Traffic_Dataset['Date'].dt.year.eq(2019)]
Traffic_Dataset['Date']=Traffic_Dataset['Date'].dt.strftime('%m/%d/%Y')
Traffic_Dataset
```

```
C:\Users\User\AppData\Local\Temp/ipykernel_12088/635188865.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  Traffic_Dataset['Date']=Traffic_Dataset['Date'].dt.strftime('%m/%d/%Y')
```

Out[252...

	SECTION ID	HIGHWAY	SECTION	SECTION LENGTH	SECTION DESCRIPTION	Date	DESCRIPTION	GROUP	TYPE	COUNT
4	1047	1	47	4.50	PATTON RD (SACKVILLE) TO MOUNT UNIACKE CONN	09/09/2019	0.5 KM EAST OF BRUSHY HILL RD	A	TC	HF

	SECTION ID	HIGHWAY	SECTION	SECTION LENGTH	SECTION DESCRIPTION	Date	DESCRIPTION	GROUP	TYPE	COUNT
554	101020	101	20	3.71	TK 1 OVERPASS (LOWER SACKVILLE) TO EXIT 2 (RTE...	09/12/2019	1.4 KM WEST OF TK 1 (EB) (LOOPS)	A	TC	HF
555	101020	101	20	3.71	TK 1 OVERPASS (LOWER SACKVILLE) TO EXIT 2 (RTE...	09/12/2019	1.4 KM WEST OF TK 1 (WB) (LOOPS)	A	TC	HF
584	101025	101	25	2.89	EXIT 2 (RTE 354 INTER/C) TO EXIT 2A (MARGESON ...	09/12/2019	2.04 KM WEST OF EXIT 2 (WB) (LOOPS)	A	TC	HF
585	101025	101	25	2.89	EXIT 2 (RTE 354 INTER/C) TO EXIT 2A (MARGESON ...	09/12/2019	1.02 KM EAST OF EXIT 2A (EB) (LOOPS)	A	TC	HF
...	...	...	...	...	...	...	...	...	...	
9290	7060	7	60	3.60	RTE 224 (SHEET HARBOUR) TO RTE 374	06/13/2019	2.5 KM EAST OF RTE 224	B	TC	HF
9304	7062	7	62	11.99	RTE 374 TO PORT DUFFERIN BRIDGE (SALMON RIVER)	06/13/2019	5 KM EAST OF RTE 374	D	TC	HF
9320	7064	7	64	14.46	PORT DUFFERIN BRIDGE (SALMON RIVER) TO MOOSEHE...	06/13/2019	3 KM EAST OF SMITH POINT RD	D	TC	HF
9334	7066	7	66	11.77	MOOSEHEAD RD (MOOSEHEAD) TO GUYSBOROUGH-HALIFA...	06/13/2019	1 KM EAST OF NAUGLERS SETTLEMENT RD (EB)	D	VC	HF
9335	7066	7	66	11.77	MOOSEHEAD RD (MOOSEHEAD) TO GUYSBOROUGH-HALIFA...	06/13/2019	1 KM EAST OF NAUGLERS SETTLEMENT RD (WB)	D	VC	HF

168 rows × 16 columns

In [409…

```
Combined_dataset = pd.merge(Traffic_Dataset, PM_Dataset, on = "Date", how = "left")
Combined_dataset
```

Out[409…

	SECTION ID	HIGHWAY	SECTION	SECTION LENGTH	SECTION DESCRIPTION	Date	DESCRIPTION	GROUP	TYPE	COUNTY
0	1047	1	47	4.50	PATTON RD (SACKVILLE) TO MOUNT UNIACKE CONN	09/09/2019	0.5 KM EAST OF BRUSHY HILL RD	A	TC	HFX

	SECTION ID	HIGHWAY	SECTION	SECTION LENGTH	SECTION DESCRIPTION	Date	DESCRIPTION	GROUP	TYPE	COUNTY
1	101020	101	20	3.71	TK 1 OVERPASS (LOWER SACKVILLE) TO EXIT 2 (RTE...	09/12/2019	1.4 KM WEST OF TK 1 (EB) (LOOPS)	A	TC	HFX
2	101020	101	20	3.71	TK 1 OVERPASS (LOWER SACKVILLE) TO EXIT 2 (RTE...	09/12/2019	1.4 KM WEST OF TK 1 (WB) (LOOPS)	A	TC	HFX
3	101025	101	25	2.89	EXIT 2 (RTE 354 INTER/C) TO EXIT 2A (MARGESON ...	09/12/2019	2.04 KM WEST OF EXIT 2 (WB) (LOOPS)	A	TC	HFX
4	101025	101	25	2.89	EXIT 2 (RTE 354 INTER/C) TO EXIT 2A (MARGESON ...	09/12/2019	1.02 KM EAST OF EXIT 2A (EB) (LOOPS)	A	TC	HFX
...	...	...	...	...	...	...	...	...	...	...
163	7060	7	60	3.60	RTE 224 (SHEET HARBOUR) TO RTE 374	06/13/2019	2.5 KM EAST OF RTE 224	B	TC	HFX
164	7062	7	62	11.99	RTE 374 TO PORT DUFFERIN BRIDGE (SALMON RIVER)	06/13/2019	5 KM EAST OF RTE 374	D	TC	HFX
165	7064	7	64	14.46	PORT DUFFERIN BRIDGE (SALMON RIVER) TO MOOSEHE...	06/13/2019	3 KM EAST OF SMITH POINT RD	D	TC	HFX
166	7066	7	66	11.77	MOOSEHEAD RD (MOOSEHEAD) TO GUYSBOROUGH-HALIFA...	06/13/2019	1 KM EAST OF NAUGLERS SETTLEMENT RD (EB)	D	VC	HFX
167	7066	7	66	11.77	MOOSEHEAD RD (MOOSEHEAD) TO GUYSBOROUGH-HALIFA...	06/13/2019	1 KM EAST OF NAUGLERS SETTLEMENT RD (WB)	D	VC	HFX

168 rows × 18 columns

In [410…

```
Combined_dataset['PM_Level'].value_counts()
```

Out[410…

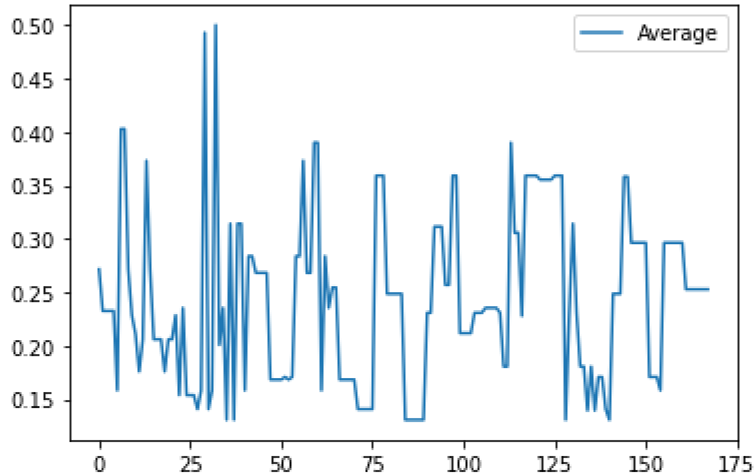
```
Low     132
High     36
Name: PM_Level, dtype: int64
```

In [411…

```
Combined_dataset[["Average"]].plot()
```

Out[411…

```
<AxesSubplot:>
```



In [412…

```
Combined_dataset = Combined_dataset[["HIGHWAY", "SECTION" , "SECTION LENGTH", "Date", "AD
Combined_dataset
```

Out[412…

	HIGHWAY	SECTION	SECTION LENGTH	Date	ADT	AADT	Average	PM_Level
0	1	47	4.50	09/09/2019	2566.0	2430.0	0.271403	Low
1	101	20	3.71	09/12/2019	23205.0	22000.0	0.232488	Low
2	101	20	3.71	09/12/2019	23385.0	22100.0	0.232488	Low
3	101	25	2.89	09/12/2019	16023.0	15200.0	0.232488	Low
4	101	25	2.89	09/12/2019	16204.0	15300.0	0.232488	Low
...	...	...	...	...	...	...	...	...
163	7	60	3.60	06/13/2019	2962.0	2760.0	0.252704	Low
164	7	62	11.99	06/13/2019	1176.0	1100.0	0.252704	Low
165	7	64	14.46	06/13/2019	784.0	730.0	0.252704	Low
166	7	66	11.77	06/13/2019	321.0	300.0	0.252704	Low
167	7	66	11.77	06/13/2019	329.0	310.0	0.252704	Low

168 rows × 8 columns

In [413…

```
Combined_dataset = Combined_dataset.drop(["Date"], axis = 1)
Combined_dataset
```

Out[413…

	HIGHWAY	SECTION	SECTION LENGTH	ADT	AADT	Average	PM_Level
0	1	47	4.50	2566.0	2430.0	0.271403	Low
1	101	20	3.71	23205.0	22000.0	0.232488	Low
2	101	20	3.71	23385.0	22100.0	0.232488	Low
3	101	25	2.89	16023.0	15200.0	0.232488	Low
4	101	25	2.89	16204.0	15300.0	0.232488	Low
...	...	...	...	...	...	...	...
163	7	60	3.60	2962.0	2760.0	0.252704	Low
164	7	62	11.99	1176.0	1100.0	0.252704	Low

	HIGHWAY	SECTION	SECTION LENGTH	ADT	AADT	Average	PM_Level
165	7	64	14.46	784.0	730.0	0.252704	Low
166	7	66	11.77	321.0	300.0	0.252704	Low
167	7	66	11.77	329.0	310.0	0.252704	Low

168 rows × 7 columns

In [414…

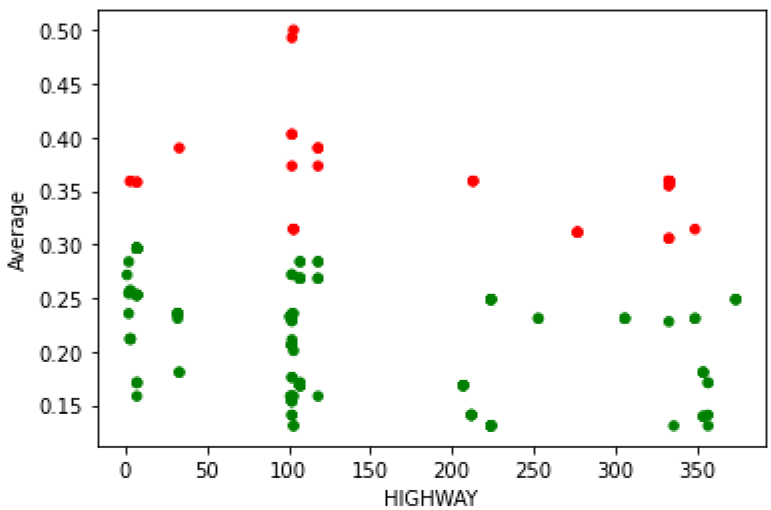
```
Plot_data = Combined_dataset
Plot_data["Color"] = Combined_dataset["PM_Level"].map({"Low": "Green", "High": "Red"})
```

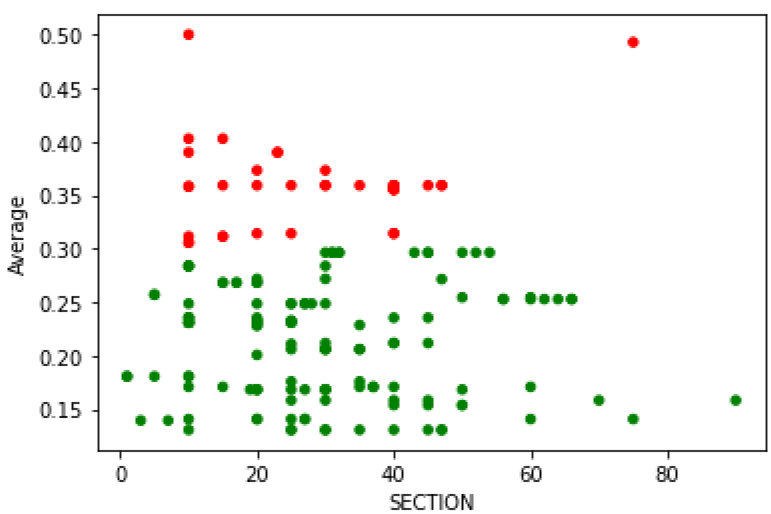
In [415…

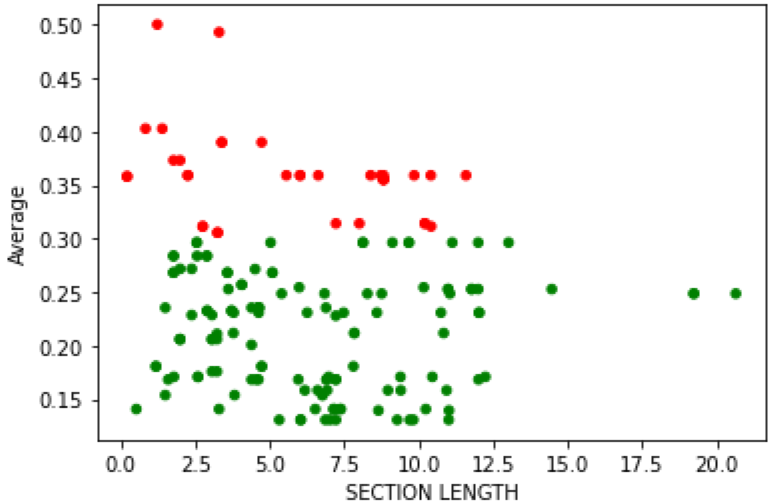
```
Combined_dataset.plot.scatter(x='HIGHWAY', y='Average', c='Color')
Combined_dataset.plot.scatter(x='SECTION', y='Average', c='Color')
Combined_dataset.plot.scatter(x='SECTION LENGTH', y='Average', c='Color')
Combined_dataset.plot.scatter(x='ADT', y='Average', c='Color')
Combined_dataset.plot.scatter(x='AADT', y='Average', c='Color')
```

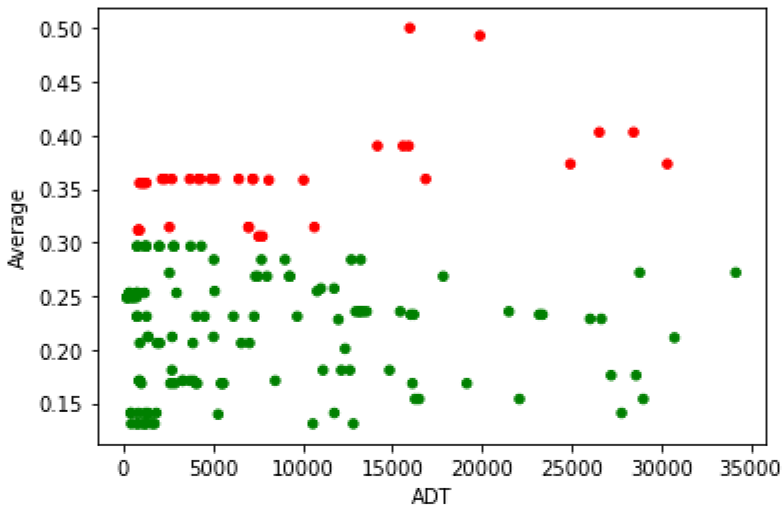
Out[415…

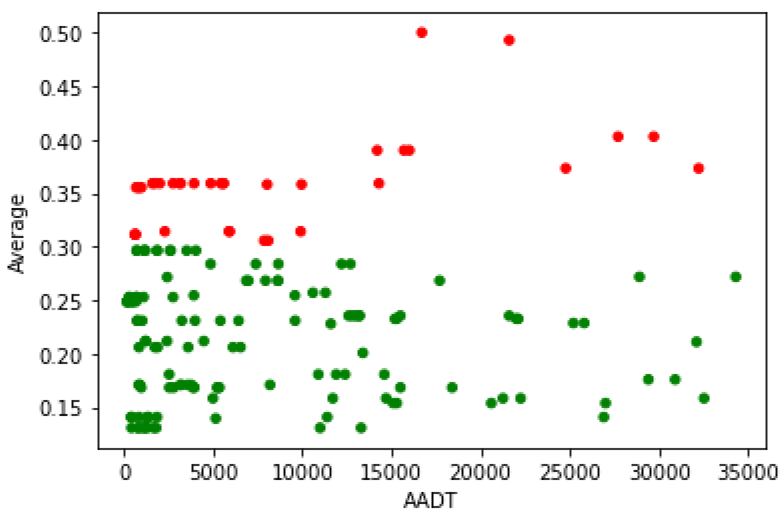
```
<AxesSubplot:xlabel='AADT', ylabel='Average'>
```











Model Evaluation

In [416…

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
```

In [417…

```
le_PM_Level = LabelEncoder()
```

In [418…

```
Combined_dataset["label"] = le_PM_Level.fit_transform(Combined_dataset["PM_Level"])
label_encoder_mapping = dict(zip(le_PM_Level.classes_, le_PM_Level.transform(le_PM_Level.c
print("Mapping of Label Encoded Classes", label_encoder_mapping, sep="\n")
```

```
Mapping of Label Encoded Classes
{'High': 0, 'Low': 1}
```

In [419…

```
Combined_dataset = Combined_dataset.drop(["Color", "Average", "PM_Level"], axis = 1)
Combined_dataset
```

Out[419…

	HIGHWAY	SECTION	SECTION LENGTH	ADT	AADT	label
0	1	47	4.50	2566.0	2430.0	1
1	101	20	3.71	23205.0	22000.0	1
2	101	20	3.71	23385.0	22100.0	1
3	101	25	2.89	16023.0	15200.0	1
4	101	25	2.89	16204.0	15300.0	1
...	...	...	...	...	...	...
163	7	60	3.60	2962.0	2760.0	1
164	7	62	11.99	1176.0	1100.0	1
165	7	64	14.46	784.0	730.0	1
166	7	66	11.77	321.0	300.0	1
167	7	66	11.77	329.0	310.0	1

168 rows × 6 columns

In [518…

```
Combined_dataset['label'].value_counts()
```

Out[518…

```
1    132
0     36
Name: label, dtype: int64
```

In [420…

```
Combined_dataset.isnull().sum()
```

Out[420…

```
HIGHWAY           0
SECTION           0
SECTION LENGTH    0
ADT               6
AADT              0
label             0
dtype: int64
```

In [421…

```
Combined_dataset = Combined_dataset.reset_index(drop=True)
```

In [422…

```
Combined_dataset = Combined_dataset.fillna(method ='pad')
```

In [423…

```
Combined_dataset.isnull().sum()
```

Out[423…

```
HIGHWAY           0
```

```
SECTION           0
SECTION LENGTH    0
ADT               0
AADT              0
label             0
dtype: int64
```

In [424…

```
feature_cols = ["HIGHWAY", "SECTION", "SECTION LENGTH", "ADT", "AADT"]
X = Combined_dataset[feature_cols]
Y = Combined_dataset.iloc[:,-1]
#X = Combined_dataset[:-1]
#y = Combined_dataset["label"]
```

In [425…

```
#Apportioning the dataset into training and testing dataset with 50-50 split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.5, random_state =
```

In [426…

```
print(y)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
163    1
164    1
165    1
166    1
167    1
Name: label, Length: 168, dtype: int32
```

In [427…

```
clf = DecisionTreeClassifier(criterion = "entropy")
clf = clf.fit(X_train, y_train)
```

In [428…

```
clf.get_params()
```

Out[428…

```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'entropy',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': None,
 'splitter': 'best'}
```

In [429…

```
y_trainpred = clf.predict(X_train)
```

In [430…

```
y_testpred = clf.predict(X_test)
```

In [431…

```
from sklearn import metrics
```

In [432…
```
#Acuracy of Training
print("Training Accuracy: ",metrics.accuracy_score(y_train, y_trainpred)*100)
```
```
Training Accuracy:  100.0
```

In [433…
```
#Acuracy of Testing
print("Testing Accuracy: ",metrics.accuracy_score(y_test, y_testpred)*100)
```
```
Testing Accuracy:  77.38095238095238
```

In [434…
```
print("Training Confusion Matrix: \n", confusion_matrix(y_train, y_trainpred))
```
```
Training Confusion Matrix:
 [[20  0]
 [ 0 64]]
```

In [435…
```
print("Testing Confusion Matrix: \n", confusion_matrix(y_test, y_testpred))
```
```
Testing Confusion Matrix:
 [[ 7  9]
 [10 58]]
```

In [458…
```
#Classification Report of Training
print(classification_report(y_train, y_trainpred))
```
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        20
           1       1.00      1.00      1.00        64

    accuracy                           1.00        84
   macro avg       1.00      1.00      1.00        84
weighted avg       1.00      1.00      1.00        84
```

In [436…
```
#Classification Report of Testing
print(classification_report(y_test, y_testpred))
```
```
              precision    recall  f1-score   support

           0       0.41      0.44      0.42        16
           1       0.87      0.85      0.86        68

    accuracy                           0.77        84
   macro avg       0.64      0.65      0.64        84
weighted avg       0.78      0.77      0.78        84
```

In [437…
```
clf.feature_importances_
```
Out[437…
```
array([0.23026643, 0.17181804, 0.24051069, 0.29202947, 0.06537538])
```

In [438…
```
feature_names = X.columns
Feature_importance = pd.DataFrame((clf.feature_importances_), index = X.columns).sort_valu
Feature_importance
```
Out[438…

	0
ADT	0.292029

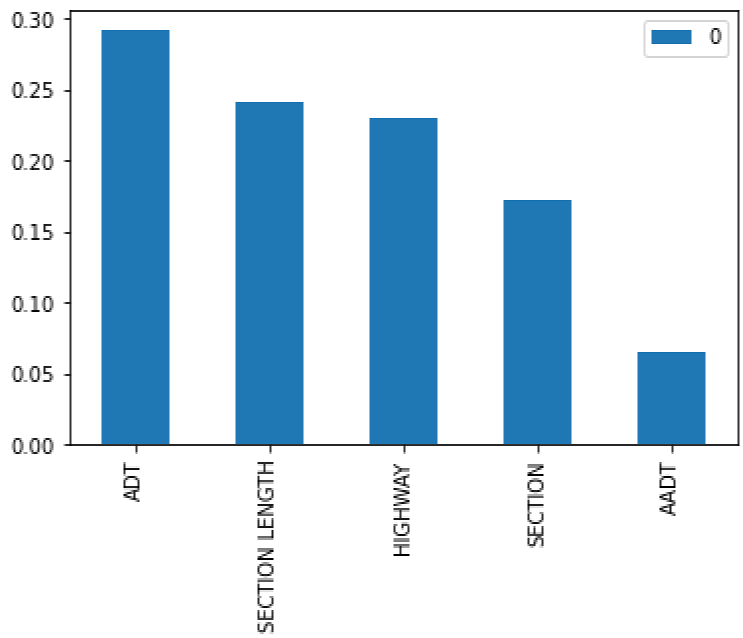
	0
SECTION LENGTH	0.240511
HIGHWAY	0.230266
SECTION	0.171818
AADT	0.065375

In [439…

```
Feature_importance.plot(kind="bar")
```

Out[439…

```
<AxesSubplot:>
```



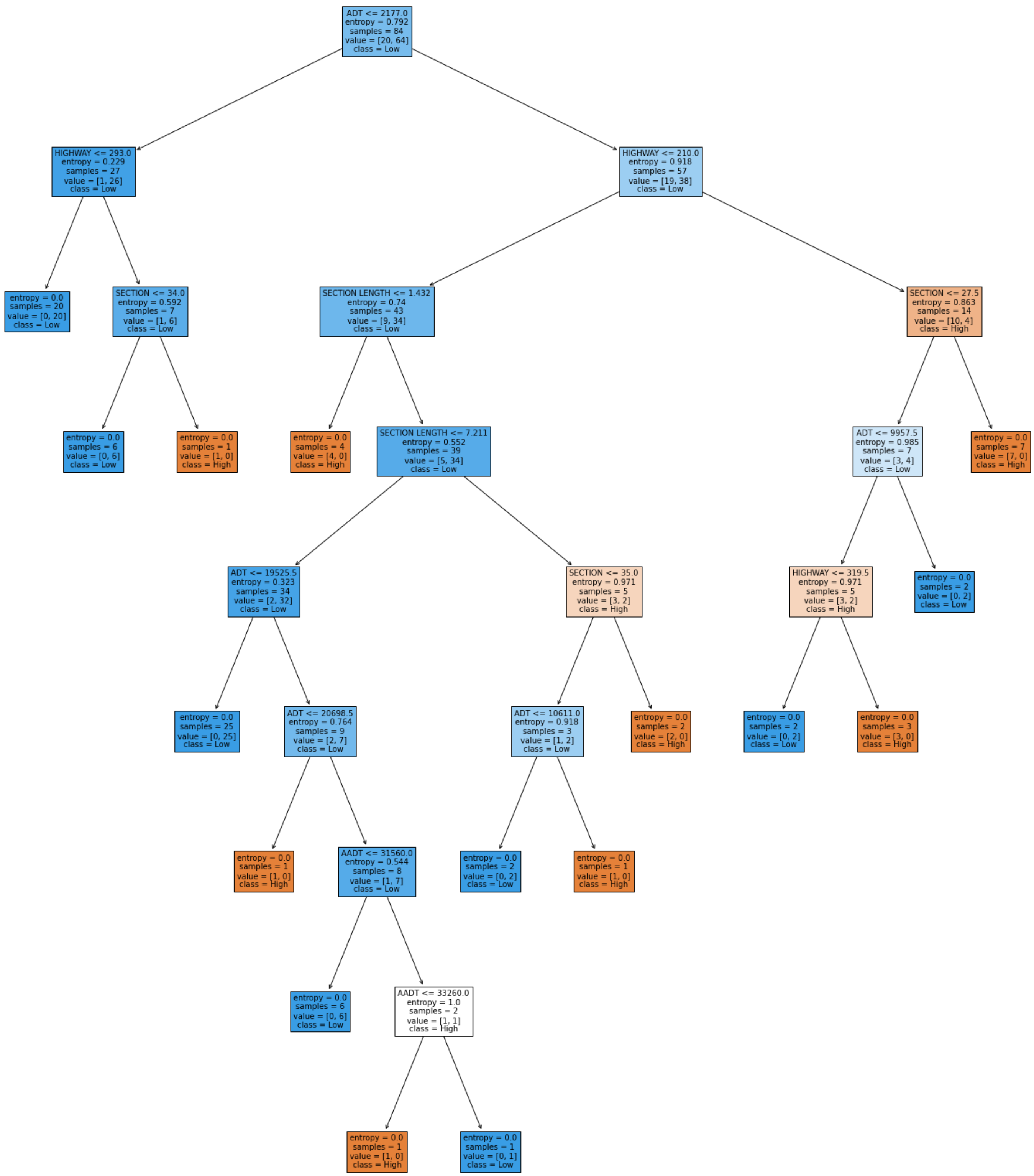
In [440…

```
from sklearn import tree
from matplotlib import pyplot as plt
fig = plt.figure(figsize=(25,30))
tree.plot_tree(clf, feature_names = feature_names, class_names = {0: "High", 1: "Low"}, fi
```

Out[440…

```
[Text(513.9473684210526, 1540.2, 'ADT <= 2177.0\nentropy = 0.792\nsamples = 84\nvalue = [2
0, 64]\nclass = Low'),
 Text(146.8421052631579, 1359.0, 'HIGHWAY <= 293.0\nentropy = 0.229\nsamples = 27\nvalue =
[1, 26]\nclass = Low'),
 Text(73.42105263157895, 1177.8, 'entropy = 0.0\nsamples = 20\nvalue = [0, 20]\nclass = Lo
w'),
 Text(220.26315789473682, 1177.8, 'SECTION <= 34.0\nentropy = 0.592\nsamples = 7\nvalue =
[1, 6]\nclass = Low'),
 Text(146.8421052631579, 996.6, 'entropy = 0.0\nsamples = 6\nvalue = [0, 6]\nclass = Lo
w'),
 Text(293.6842105263158, 996.6, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]\nclass = Hig
h'),
 Text(881.0526315789473, 1359.0, 'HIGHWAY <= 210.0\nentropy = 0.918\nsamples = 57\nvalue =
[19, 38]\nclass = Low'),
 Text(513.9473684210526, 1177.8, 'SECTION LENGTH <= 1.432\nentropy = 0.74\nsamples = 43\nv
alue = [9, 34]\nclass = Low'),
 Text(440.52631578947364, 996.6, 'entropy = 0.0\nsamples = 4\nvalue = [4, 0]\nclass = Hig
h'),
 Text(587.3684210526316, 996.6, 'SECTION LENGTH <= 7.211\nentropy = 0.552\nsamples = 39\nv
alue = [5, 34]\nclass = Low'),
 Text(367.10526315789474, 815.4, 'ADT <= 19525.5\nentropy = 0.323\nsamples = 34\nvalue =
[2, 32]\nclass = Low'),
 Text(293.6842105263158, 634.2, 'entropy = 0.0\nsamples = 25\nvalue = [0, 25]\nclass = Lo
```

```
w'),
 Text(440.52631578947364, 634.2, 'ADT <= 20698.5\nentropy = 0.764\nsamples = 9\nvalue =
[2, 7]\nclass = Low'),
 Text(367.10526315789474, 453.0, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]\nclass = Hig
h'),
 Text(513.9473684210526, 453.0, 'AADT <= 31560.0\nentropy = 0.544\nsamples = 8\nvalue =
[1, 7]\nclass = Low'),
 Text(440.52631578947364, 271.79999999999995, 'entropy = 0.0\nsamples = 6\nvalue = [0, 6]
\nclass = Low'),
 Text(587.3684210526316, 271.79999999999995, 'AADT <= 33260.0\nentropy = 1.0\nsamples = 2
\nvalue = [1, 1]\nclass = High'),
 Text(513.9473684210526, 90.60000000000014, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]\nc
lass = High'),
 Text(660.7894736842105, 90.60000000000014, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]\nc
lass = Low'),
 Text(807.6315789473684, 815.4, 'SECTION <= 35.0\nentropy = 0.971\nsamples = 5\nvalue =
[3, 2]\nclass = High'),
 Text(734.2105263157895, 634.2, 'ADT <= 10611.0\nentropy = 0.918\nsamples = 3\nvalue = [1,
2]\nclass = Low'),
 Text(660.7894736842105, 453.0, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]\nclass = Lo
w'),
 Text(807.6315789473684, 453.0, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]\nclass = Hig
h'),
 Text(881.0526315789473, 634.2, 'entropy = 0.0\nsamples = 2\nvalue = [2, 0]\nclass = Hig
h'),
 Text(1248.157894736842, 1177.8, 'SECTION <= 27.5\nentropy = 0.863\nsamples = 14\nvalue =
[10, 4]\nclass = High'),
 Text(1174.7368421052631, 996.6, 'ADT <= 9957.5\nentropy = 0.985\nsamples = 7\nvalue = [3,
4]\nclass = Low'),
 Text(1101.3157894736842, 815.4, 'HIGHWAY <= 319.5\nentropy = 0.971\nsamples = 5\nvalue =
[3, 2]\nclass = High'),
 Text(1027.8947368421052, 634.2, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]\nclass = Lo
w'),
 Text(1174.7368421052631, 634.2, 'entropy = 0.0\nsamples = 3\nvalue = [3, 0]\nclass = Hig
h'),
 Text(1248.157894736842, 815.4, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]\nclass = Lo
w'),
 Text(1321.578947368421, 996.6, 'entropy = 0.0\nsamples = 7\nvalue = [7, 0]\nclass = Hig
h')]
```



In [441…

```
Combined_dataset
```

Out[441…

	HIGHWAY	SECTION	SECTION LENGTH	ADT	AADT	label
0	1	47	4.50	2566.0	2430.0	1
1	101	20	3.71	23205.0	22000.0	1
2	101	20	3.71	23385.0	22100.0	1
3	101	25	2.89	16023.0	15200.0	1

	HIGHWAY	SECTION	SECTION LENGTH	ADT	AADT	label
4	101	25	2.89	16204.0	15300.0	1
...	...	...	...	...	...	...
163	7	60	3.60	2962.0	2760.0	1
164	7	62	11.99	1176.0	1100.0	1
165	7	64	14.46	784.0	730.0	1
166	7	66	11.77	321.0	300.0	1
167	7	66	11.77	329.0	310.0	1

168 rows × 6 columns

In [442…

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
```

In [443…

```
model = DecisionTreeClassifier()
```

In [444…

```
kf = KFold(n_splits=10)
```

In [445…

```
scores = cross_val_score(model, X, y, cv = kf)
scores
```

Out[445…

```
array([0.82352941, 0.88235294, 0.82352941, 0.64705882, 0.82352941,
       0.70588235, 0.70588235, 0.76470588, 0.875     , 1.        ])
```

In [446…

```
print("Accuracy:", np.mean(scores)*100, "\n Standard Deviation:", np.std(scores))
```

```
Accuracy: 80.51470588235293
 Standard Deviation: 0.09805912787983395
```

In [448…

```
conf_mat = []
i = 1
for train_index, test_index in kf.split(X):
    Xtrain, Xtest = X.iloc[train_index], X.iloc[test_index]
    ytrain, ytest = y.iloc[train_index], y.iloc[test_index]
    model = clf.fit(Xtrain, ytrain)
    ypred = clf.predict(Xtest)
    conf_matrix = confusion_matrix(ytest, ypred)
    conf_mat.append(conf_matrix)
    print("Fold", i)
    print(classification_report(ytest, ypred))
    print("-------------------------------------------------------")
    i += 1
```

```
Fold 1
              precision    recall  f1-score   support

           0       0.22      0.67      0.33         3
           1       0.88      0.50      0.64        14

    accuracy                           0.53        17
   macro avg       0.55      0.58      0.48        17
weighted avg       0.76      0.53      0.58        17
```

```
-------------------------------------------------------
Fold 2
              precision    recall  f1-score   support

           0       0.50      0.50      0.50         2
           1       0.93      0.93      0.93        15

    accuracy                           0.88        17
   macro avg       0.72      0.72      0.72        17
weighted avg       0.88      0.88      0.88        17

-------------------------------------------------------
Fold 3
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.82      1.00      0.90        14

    accuracy                           0.82        17
   macro avg       0.41      0.50      0.45        17
weighted avg       0.68      0.82      0.74        17

-------------------------------------------------------
Fold 4
              precision    recall  f1-score   support

           0       0.67      0.67      0.67         3
           1       0.93      0.93      0.93        14

    accuracy                           0.88        17
   macro avg       0.80      0.80      0.80        17
weighted avg       0.88      0.88      0.88        17

-------------------------------------------------------
Fold 5
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.82      1.00      0.90        14

    accuracy                           0.82        17
   macro avg       0.41      0.50      0.45        17
weighted avg       0.68      0.82      0.74        17

-------------------------------------------------------
Fold 6
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         5
           1       0.71      1.00      0.83        12

    accuracy                           0.71        17
   macro avg       0.35      0.50      0.41        17
weighted avg       0.50      0.71      0.58        17

-------------------------------------------------------
Fold 7
              precision    recall  f1-score   support

           0       0.50      0.80      0.62         5
           1       0.89      0.67      0.76        12

    accuracy                           0.71        17
   macro avg       0.69      0.73      0.69        17
weighted avg       0.77      0.71      0.72        17
```

```
-------------------------------------------------------
Fold 8
              precision    recall  f1-score   support

           0       0.56      0.50      0.53        10
           1       0.38      0.43      0.40         7

    accuracy                           0.47        17
   macro avg       0.47      0.46      0.46        17
weighted avg       0.48      0.47      0.47        17

-------------------------------------------------------
Fold 9
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.86      0.86      0.86        14

    accuracy                           0.75        16
   macro avg       0.43      0.43      0.43        16
weighted avg       0.75      0.75      0.75        16

-------------------------------------------------------
Fold 10
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        16

    accuracy                           1.00        16
   macro avg       1.00      1.00      1.00        16
weighted avg       1.00      1.00      1.00        16

-------------------------------------------------------
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
```

```
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [449…

```
from sklearn.model_selection import cross_validate
_scoring = ['precision', 'recall', 'f1']
results = cross_validate(estimator=model,
                               X=X,
                               y=y,
                               cv=10,
                               scoring=_scoring,
                               )

pre_avg = results['test_precision'].mean()
re_avg = results['test_recall'].mean()
F1_avg = results['test_f1'].mean()
print("The average Precision is:", pre_avg)
print("The average Recall is:", re_avg)
print("The average F1 score is:", F1_avg)
```

```
The average Precision is: 0.8388103318250376
The average Recall is: 0.6582417582417583
The average F1 score is: 0.6910952755831052
```

In [450…

```
average_conf = np.mean(conf_mat)
average_conf
```

```
C:\Users\User\anaconda3\lib\site-packages\numpy\core\_asarray.py:171: VisibleDeprecationWa
rning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists
-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do
this, you must specify 'dtype=object' when creating the ndarray.
  return array(a, dtype, copy=False, order=order, subok=True)
```

Out[450…

```
array([[ 3,  3],
       [ 3, 11]], dtype=int64)
```

Experiments with max_depth, min_samples_split and min_samples_leaf

In [523…

```
i = 1
for depth in [2,4,6,8,10]:
    tree = DecisionTreeClassifier(max_depth = depth)
    tree.fit(X_train,y_train)

    #y_exp_trainpred = tree.predict(X_train)
    y_exp_testpred = tree.predict(X_test)

    print("Classification Report", i)
    print(classification_report(y_test, y_exp_testpred))

    i += 1
```

```
Classification Report 1
              precision    recall  f1-score   support

           0       0.29      0.12      0.17        16
           1       0.82      0.93      0.87        68

    accuracy                           0.77        84
```

```
   macro avg       0.55      0.53      0.52        84
weighted avg       0.72      0.77      0.74        84

Classification Report 2
              precision    recall  f1-score   support

           0       0.40      0.50      0.44        16
           1       0.88      0.82      0.85        68

    accuracy                           0.76        84
   macro avg       0.64      0.66      0.65        84
weighted avg       0.78      0.76      0.77        84

Classification Report 3
              precision    recall  f1-score   support

           0       0.37      0.44      0.40        16
           1       0.86      0.82      0.84        68

    accuracy                           0.75        84
   macro avg       0.61      0.63      0.62        84
weighted avg       0.77      0.75      0.76        84

Classification Report 4
              precision    recall  f1-score   support

           0       0.40      0.38      0.39        16
           1       0.86      0.87      0.86        68

    accuracy                           0.77        84
   macro avg       0.63      0.62      0.62        84
weighted avg       0.77      0.77      0.77        84

Classification Report 5
              precision    recall  f1-score   support

           0       0.40      0.38      0.39        16
           1       0.86      0.87      0.86        68

    accuracy                           0.77        84
   macro avg       0.63      0.62      0.62        84
weighted avg       0.77      0.77      0.77        84
```

In [524…

```
i = 1
for values in [5,15,25,35,45]:
    tree1 = DecisionTreeClassifier(min_samples_split = values)
    tree1.fit(X_train,y_train)

    #y_exp_trainpred1 = tree1.predict(X_train)
    y_exp_testpred1 = tree1.predict(X_test)

    print("Classification Report", i)
    print(classification_report(y_test, y_exp_testpred1))

    i += 1
```

```
Classification Report 1
              precision    recall  f1-score   support

           0       0.46      0.38      0.41        16
           1       0.86      0.90      0.88        68

    accuracy                           0.80        84
   macro avg       0.66      0.64      0.65        84
```

```
weighted avg       0.78      0.80      0.79        84

Classification Report 2
              precision    recall  f1-score   support

           0       0.20      0.25      0.22        16
           1       0.81      0.76      0.79        68

    accuracy                           0.67        84
   macro avg       0.51      0.51      0.51        84
weighted avg       0.70      0.67      0.68        84

Classification Report 3
              precision    recall  f1-score   support

           0       0.20      0.25      0.22        16
           1       0.81      0.76      0.79        68

    accuracy                           0.67        84
   macro avg       0.51      0.51      0.51        84
weighted avg       0.70      0.67      0.68        84

Classification Report 4
              precision    recall  f1-score   support

           0       0.22      0.25      0.24        16
           1       0.82      0.79      0.81        68

    accuracy                           0.69        84
   macro avg       0.52      0.52      0.52        84
weighted avg       0.70      0.69      0.70        84

Classification Report 5
              precision    recall  f1-score   support

           0       0.29      0.12      0.17        16
           1       0.82      0.93      0.87        68

    accuracy                           0.77        84
   macro avg       0.55      0.53      0.52        84
weighted avg       0.72      0.77      0.74        84
```

In [525…

```
i = 1
for values in [5,15,25,35,45]:
    tree2 = DecisionTreeClassifier(min_samples_leaf = values)
    tree2.fit(X_train,y_train)

    #y_exp_trainpred2 = tree2.predict(X_train)
    y_exp_testpred2 = tree2.predict(X_test)

    print("Classification Report", i)
    print(classification_report(y_test, y_exp_testpred2))

    i += 1
```

```
Classification Report 1
              precision    recall  f1-score   support

           0       0.23      0.19      0.21        16
           1       0.82      0.85      0.83        68

    accuracy                           0.73        84
   macro avg       0.52      0.52      0.52        84
weighted avg       0.71      0.73      0.71        84
```

```
Classification Report 2
              precision    recall  f1-score   support

           0       0.25      0.12      0.17        16
           1       0.82      0.91      0.86        68

    accuracy                           0.76        84
   macro avg       0.53      0.52      0.51        84
weighted avg       0.71      0.76      0.73        84

Classification Report 3
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        16
           1       0.81      1.00      0.89        68

    accuracy                           0.81        84
   macro avg       0.40      0.50      0.45        84
weighted avg       0.66      0.81      0.72        84

Classification Report 4
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        16
           1       0.81      1.00      0.89        68

    accuracy                           0.81        84
   macro avg       0.40      0.50      0.45        84
weighted avg       0.66      0.81      0.72        84

Classification Report 5
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        16
           1       0.81      1.00      0.89        68

    accuracy                           0.81        84
   macro avg       0.40      0.50      0.45        84
weighted avg       0.66      0.81      0.72        84

C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:

In [ ]: