Assignment 4

1) To predict if the student will ace the end sem exam, based on mid semester & assignment submission

From data, (Exam = End Sem)

→ Aced Exam → 5 False, 6 True

→ Aced Exam and Submitted assignment → 5 True, 1 False

Aced Exam and not submitted assignment → 4 False, 1 True

→ Aced Exam and Mid-Sem Performance

- Below Average → 3 False, 1 True
- Average → 2 False, 3 True
- Above Average → 2 True, 0 False

Formulae:

→ Entropy $= -\left[P_T \log P_T + P_F \log P_F\right]$ {for our problem}

→ Average Entropy $= \frac{n_1}{n} E_1 + \frac{n_2}{n} E_2$

↳ weighted avg. of all the subnodes that a parent node split.

→ Information Gain : E_{parent} - Avg. E_{child}

Part 1 Parent Node Calculation: (Accd Expcl Sem)

→ $P_T = \frac{6}{11}$, $P_F = \frac{5}{11}$

→ Entropy $(\mathcal{E}_P) = -\left[\frac{6}{11}\ln(6/11) + \frac{5}{11}\ln(\frac{5}{11})\right]$

$= 0.689$

Root Node Calculation:

a) Submitted Assignments

Entropy - submitted $(\mathcal{E}_s) = -\left[\frac{5}{6}\ln(\frac{5}{6}) + \frac{1}{6}\ln(1/6)\right]$

$= 0.451$

Entropy - not submitted $(\mathcal{E}_{ns}) = -\left[\frac{1}{5}\ln(\frac{1}{5}) + \frac{4}{5}\ln(\frac{4}{5})\right]$

$= 0.5$

Avg. Entropy - submission $= \frac{6}{11}(0.451) + \frac{5}{11}(0.5)$

$= 0.473$

IG - submitted Assignments $= 0.689 - 0.473 = 0.216$

b) Mid - Sem Performance

Entropy - BA $= -\left[\frac{1}{4}\ln(\frac{1}{4}) + \frac{3}{4}\ln(\frac{3}{4})\right] = 0.562$

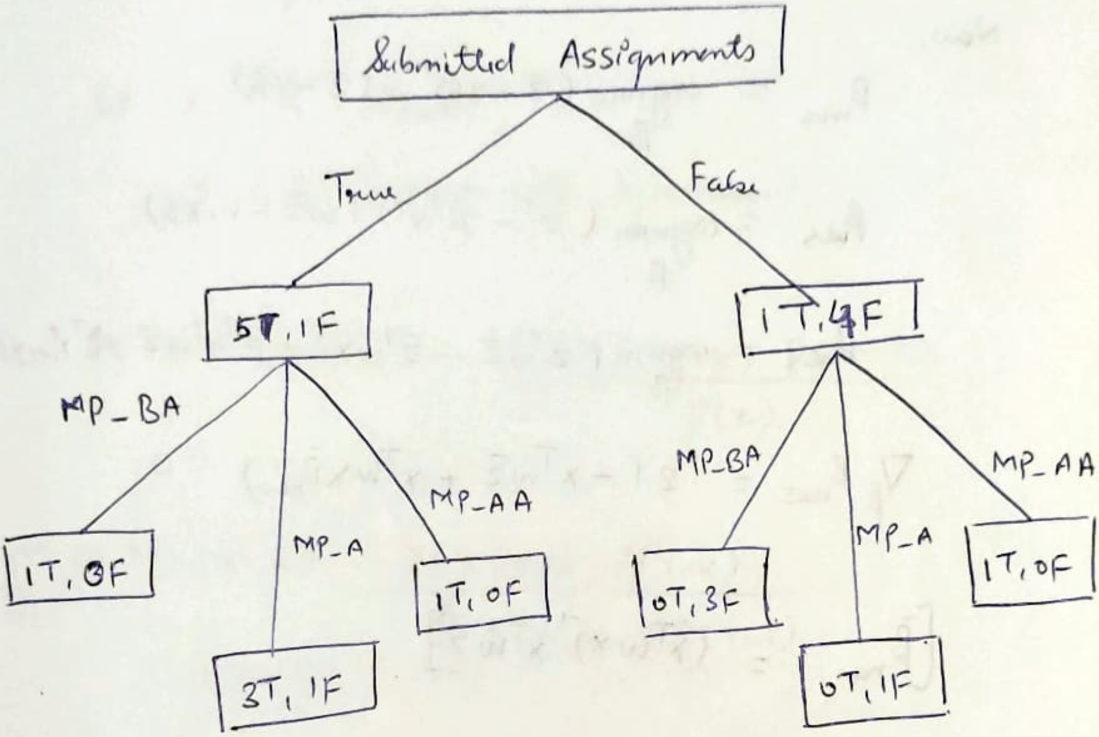
Entropy - A $= -\left[\frac{3}{5}\ln(3/5) + \frac{2}{5}\ln(\frac{2}{5})\right] = 0.673$

Entropy - AA $= -\left[\frac{2}{2}\ln(\frac{2}{2}) + 0\right] = 0$

Avg. Entropy - Mid Sem $= 0.510$

IG - mid sem performance $= 0.689 - 0.510 = 0.179$

⇒ Based on the information gain the root node would be submitted assignments, since we have only two variables so no need for further calculation



where, MP_BA = midsem performance below average

MP_A = midsem performance average

MP_AA = midsem performance above average

→ Here only one node is not terminal node i.e. $\boxed{3T, 1F}$, rest all are terminal nodes.

but since we don't have any other variable, we can't split it further.

Problem 27 Given

$$WLSE : (\bar{z} - X\beta)^T W(\bar{z} - X\bar{\beta})$$

where, $\bar{z} = X\bar{\beta}_{old} - W^{-1}(\bar{y} - \bar{p})$

Now,

$$\beta_{WLS} = \arg\min_{\beta} (\bar{z} - X\bar{\beta})^T W(\bar{z} - X\bar{\beta})$$

$$\beta_{WLS} = \arg\min_{\beta} (\bar{z}^T - \bar{\beta}^T X^T)(W\bar{z} - WX\bar{\beta})$$

$$\beta_{WLS} = \arg\min_{\beta} (\bar{z}^T W \bar{z} - \bar{z}^T W X \bar{\beta} - \bar{\beta}^T X^T W \bar{z} + \bar{\beta}^T X^T W X \bar{\beta})$$

$$\nabla_{\bar{\beta}} \beta_{WLS} = 2(-X^T W \bar{z} + X^T W X \bar{\beta}_{new}) = 0$$

$$\left[\bar{\beta}_{new} = (X^T W X)^{-1} X^T W \bar{z}\right]$$

Substituting $\bar{z} = X\bar{\beta}_{old} + W^{-1}(\bar{y} - \bar{p})$

$$\bar{\beta}_{new} = (X^T W X)^{-1} X^T W (X\bar{\beta}_{old} + W^{-1}(\bar{y} - \bar{p}))$$

$$\bar{\beta}_{new} = (X^T W X)^{-1}((X^T W X)\bar{\beta}_{old} + X^T(\bar{y} - \bar{p}))$$

$$\boxed{\bar{\beta}_{new} = \bar{\beta}_{old} + (X^T W X)^{-1} X^T (\bar{y} - \bar{p})}$$

3) In LDA we assume homogenous covariance matrices but is it not applicable in many case. The quadratic discriminant analysis is for heterogenous variance-covariance matrices.

let $X \subseteq \mathbb{R}^p$ (input)

$Y \subseteq \Gamma$ (output with k classes)

$$\rightarrow P(Y=K \mid X=x) = \frac{P(x \mid Y=k)\, P(Y=k)}{P(x)}$$

$$\rightarrow p(Y=k \mid X=x) = \frac{P(x \mid Y=k)\, P(Y=k)}{\sum_{i=1}^{K} P(x \mid Y=i)\, P(Y=i)}$$

let $f_k(x) = P(x \mid Y=k)$

$\pi_k = P(Y=k)$

$$\rightarrow P(Y=k \mid X=x) = \frac{f_k(x) \cdot \pi_k}{\sum_{i=1}^{K} f_i(x) \cdot \pi_i}$$

assuming $f_k(x)$ be a gaussian distribution

$$f_k(x) = \frac{1}{(\sqrt{2\pi})^{1/p} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\bar{\mu}_k)^T \Sigma_k^{-1} (x-\bar{\mu}_k)\right)$$

→ Boundary condition (between class l & k)

$$\log\left(\frac{P(Y=k|X=x)}{P(Y=l|X=x)}\right) = 0$$

$$\rightarrow \log(P(Y=k|X=x)) - \log(P(Y=l|X=x)) = 0$$

$$\rightarrow \log(f_k(x)\,\pi_k) - \log(f_l(x)\,\pi_l) = 0$$

$$\rightarrow \log\left(\frac{f_k(x)}{f_l(x)}\right) + \log\left(\frac{\pi_k}{\pi_l}\right) = 0$$

$$\rightarrow \log\left(\left(\frac{\Sigma_l}{\Sigma_k}\right)^{1/2}\right) + \cancel{\log} -\frac{1}{2}\left((x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right)$$
$$-\frac{1}{2}\left((x-\mu_l)^T\Sigma_l^{-1}(x-\mu_l)\right) + \log\left(\frac{\pi_k}{\pi_l}\right) = 0$$

$$\rightarrow -\frac{1}{2}\log\left(\frac{\Sigma_k}{\Sigma_l}\right) + \log\left(\frac{\pi_k}{\pi_l}\right) - \frac{1}{2}\Big[x^T(\Sigma_k^{-1}-\Sigma_l^{-1})x - x^T(\mu_k\Sigma_k^{-1}-\mu_l\Sigma_l^{-1})$$
$$- \cancel{x}(\mu_k\Sigma_k^{-1}-\mu_l\Sigma_l^{-1})x$$
$$+ (\mu_k^2\Sigma_k^{-1}-\mu_l^2\Sigma_l^{-1})\Big]$$

So. $$\left[\delta_k(x) = -\frac{1}{2}\log(\Sigma_k) - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log(\pi_k)\right]$$

~~Hence, it~~ ∴ $\forall k$ no two class have same covariance

hence the quadratic term will not cancel out

resulting in quadratic boundary.

Finally the objective function can be written as

$$\left[\delta_k(x) = \log(\pi_k) - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}x^T\mu_k x - \frac{1}{2}\log|\Sigma_k|\right]$$

→ Class Prediction: $\hat{y} = \underset{k}{\arg\max}\ \delta_k(x)$

- basically a point lies in k class if

$$\log\left(\frac{P(Y=k \mid X=\bar{x})}{P(Y=l \mid X=\bar{x})}\right) > 0 \quad \forall\, l \neq k$$

$$\rightarrow \hat{y} = \underset{k}{\arg\max}\left[\log(\pi_k) - \frac{1}{2}\log(|\Sigma_k|) - \frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)\right]$$

4) to find probability of passing endsem given the midsem performance was average and all assignment were submitted.

X_1 = 'average' midsem performance

X_2 = all assignments were submitted

Y = passing endsem.

$X = \langle X_1, X_2 \rangle$

assuming conditional independence of features

$$P(Y|X) = \frac{P(X|Y)\,P(Y)}{P(X)}$$

$$P(Y|X) = \frac{P(X_1|Y)\,P(X_2|Y)\,P(Y)}{\sum_{i=1}^{2} P(X_1|Y_i)\,P(X_2|Y_i)\,P(Y_i)}$$

$$P(Y|X_1,X_2) = \frac{\left(\frac{3}{6}\right)\left(\frac{5}{6}\right)\left(\frac{6}{11}\right)}{\left(\frac{2}{5}\right)\left(\frac{1}{5}\right)\left(\frac{5}{11}\right) + \left(\frac{3}{6}\right)\left(\frac{5}{6}\right)\left(\frac{6}{11}\right)}$$

$$P(Y|X_1,X_2) = \frac{\frac{15}{6}}{\frac{2}{5} + \frac{15}{6}} = \frac{25}{29}$$

5) Given $p(x|y=0) \sim N(0, 1/4)$ | two classes 0, 1

$p(x|y=1) \sim N(0, 1/2)$

$p(y=1) = 0.5$

$$L = \begin{bmatrix} 0 & \sqrt{2} \\ 1 & 0 \end{bmatrix}$$

Posterior for class 0

$$P(Y=0|x) = \frac{P(x|Y=0)\,P(Y=0)}{P(x)}$$

Posterior for class 1

$$P(Y=1|x) = \frac{P(x|Y=1)\,P(Y=1)}{P(x)}$$

→ The loss for class 0 & 1

$$R(Y=0|x) = l_{00}\,P(Y=0|x) + l_{01}\,P(Y=1|x)$$

$$R(Y=1|x) = l_{10}\,P(Y=0|x) + l_{11}\,P(Y=1|x)$$

→ We say a point belongs to class 1 if

$R(Y=1|x) < R(Y=0|x)$ & to class 0 if

$\cancel{l_{10}P(Y}$ $R(Y=0|x) < R(Y=1|x)$

So, the decision boundary is

$$R(Y=0|x) = R(Y=1|x)$$

$$l_{00}P(Y=0|x) + l_{01}P(Y=1|x) = l_{10}P(Y=0|x) + l_{11}P(Y=1|x)$$

$$\sqrt{2}\,P(Y=1|x) = P(Y=0|x)$$

$$\sqrt{2}\left(\frac{P(x \mid y=1)\,P(y=1)}{P(x)}\right) = \left(\frac{P(x \mid y=0)\,P(y=0)}{P(x)}\right)$$

$$\sqrt{2}\,P(x \mid y=1) = P(x \mid y=0)$$

$$\sqrt{2}\left(\frac{1}{\sqrt{2\pi\left(\frac{1}{2}\right)}}\exp\left(\frac{-x^2}{2(1/2)}\right)\right) = \left(\frac{1}{\sqrt{2\pi\left(\frac{1}{4}\right)}}\exp\left(\frac{-x^2}{2(1/4)}\right)\right)$$

$$\left(\sqrt{\frac{2}{\pi}}\exp\left(-x^2\right)\right) = \sqrt{\frac{2}{\pi}}\exp\left(-2x^2\right)$$

$$\cancel{\exp(x^2) = 1}$$

$$\cancel{x^2 = \ln 2}$$

~~Decision Rule~~ $\cancel{\left[x = \pm\sqrt{\frac{\ln 2}{12}}\right]}$

$$\exp(x^2) = 1$$

Decision Rule $\boxed{x = 0}$

6) Given the prediction function

$$\hat{y} = \text{sign}(w^T x + b) \quad , \quad w^T \in R^d, b \in R$$

let $t_i \in \{-1, 1\}$

$$y_i = \text{sign}(w^T x_i + b)$$

a) In hard margin SVM we try to ~~minimize~~ maximize the distance of the closest point from the decision boundary.

$$\arg\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{such that} \quad t_i(w^T x_i + b) \geq 1$$

Using lagrangian we can write

$$L(w, b, \alpha) = \min_{w,b}\left[\max\left\{\frac{1}{2}\|w\|^2 - \sum_{i=1}^{d} \alpha_i \{t_i(w^T x_i + b) - 1\}\right\}\right]$$

using dual formulation

$$L(w, b, \alpha) = \max_{\alpha \geq 0}\left[\min_{w,b}\left\{\frac{1}{2}\|w\|^2 - \sum_{i=1}^{d} \alpha_i (t_i(w^T x_i + b) - 1)\right\}\right]$$

Now

$$\frac{\partial L}{\partial w} = 0 \quad \Rightarrow \quad \left[w = \sum_{i=1}^{d} \alpha_i t_i x_i \right]$$

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \left[\sum_{i=1}^{d} \alpha_i t_i = 0 \right]$$

So, we have

$$L(\alpha) = \max_{\alpha \geq 0}\left[\alpha^T \mathbb{I} - \frac{1}{2}\alpha^T t^T x^T x t \alpha \right]$$

We can write

$$y_i = \left(\sum_{i=1}^{d} \alpha_i t_i x_i\right) X_i + b$$

- we have the following constraints

$$\begin{cases} \alpha_i \geq 0 \quad \forall i \\ t_i y_i \geq 1 \quad \forall i \\ \alpha_i (t_i (y_i) - 1) = 0 \quad \forall i \end{cases}$$

by KKT, implies either $\alpha_i = 0$ or

$$t_i y_i - 1 = 0$$

So, only $\alpha \neq 0$ play role in making prediction

Now, to get value of b

$$t_n \left[\left(\sum_{i=1}^{d} \alpha_i t_i x_i\right) x_n + b \right] = 1 \qquad \Bigg| \quad \begin{matrix} t_n = \{-1, 1\} \\ t_n^2 = 1 \end{matrix}$$

$$t_n^2 \left[\left(\sum_{i=1}^{d} (\alpha_i t_i x_i)\right) x_n + b \right] = t_n$$

$$b = t_n - \left(\sum_{i=1}^{d} \alpha_i t_i x_i\right) x_n$$

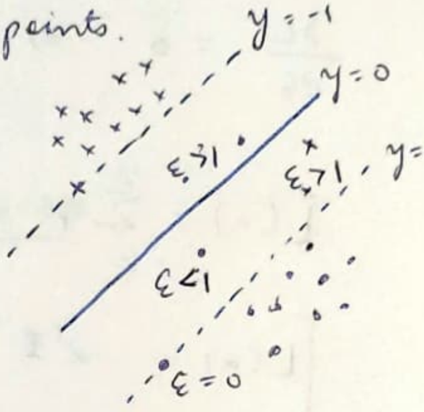
averaging over all support vectors

$$\left[\text{Bias} = b = \frac{1}{N_S} \sum_{n \in S} \left[t_n - \left(\sum_{i=1}^{d} \alpha_i t_i x_i\right) x_n \right] \right]$$

b) For soft margin SVM, we will introduce slack variables ξ_i $\forall i \in d$

$\xi_i = 0$ for datapoints that are on or inside the correct margin boundary

$\xi_i = | t_i - y_i |$ for other points.



Now we have new condition

$$t_i y_i + \xi_i \geq 1 \quad , i = 1, 2 \ldots d$$

such that $\xi_i \geq 0$

→ Maximize the margin while penalizing the points that lie on the wrong side.

$$\underset{w,\,\xi}{\arg\min} \; \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{d} \xi_i \quad , \quad \text{such that } \xi_i \geq 0 \, , \, i = 1, 2 \ldots N \, , \; t_i y_i + \xi_i \geq 1$$

→ Using lagrangian

$$\left[L(w, b, \xi, \alpha, \beta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{d} \xi_i - \sum_{i=1}^{d} \alpha_i \left(t_i (w^T x_i + b) + \xi_i - 1 \right) - \sum_{i=1}^{d} \beta_i \xi_i \right]$$

$$\min_{w,b} \left[\max_{\alpha, \beta \geq 0} L(w, b, \xi, \alpha, \beta) \right] \equiv \max_{\substack{\alpha \geq 0 \\ \beta \geq 0}} \left[\min_{w,b} L(w, b, \xi, \alpha, \beta) \right]$$

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{d} \alpha_i t_i x_i$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{d} \alpha_i t_i = 0$$

$$\frac{\partial L}{\partial \xi} = 0 \Rightarrow \alpha_i = C - \beta_i$$

$$\tilde{L}(\alpha) = \sum_{i=1}^{d} \alpha_i - \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} \alpha_i \alpha_j t_i t_j x_i x_j$$

$$\left[L(\alpha) = \alpha^T I - \frac{1}{2} \alpha^T t^T X^T X t \alpha \right]$$

maximize over
$\alpha \geq 0, \beta \geq 0$
$\alpha + \beta = C$

$\therefore \beta \geq 0 \longrightarrow \alpha \leq C$

So, $\left[0 \leq \alpha \leq C \quad \& \quad \sum_{i=1}^{d} \alpha_i t_i = 0 \right]$ new set of constraints

Now, $\left[y_i = t_i (w^T x_i + b) + \xi_i \right]$

$$t_n (w^T x_n + b) + \xi_n = 1$$

$$t_n \left(\left(\sum_{i=1}^{d} \alpha_i t_i x_i \right) x_n + b \right) + \xi_n = 1$$

$$b = t_n - \left(\sum_{i=1}^{d} \alpha_i t_i x_i \right) x_n$$

Now, for $0 < \alpha < C$ the support vectors will going to have $\xi_n = 0$ so that $t_n y_n = 1$

averaging over data points having $0 < \alpha < C$

bias $= \left[b = \frac{1}{N_M} \sum_{m \in M} \left(t_n - \left(\sum_{i=1}^{d} \alpha_i t_i x_i \right) x_n \right) \right]$

7) a) Because the hinge loss for SVM only consider support vectors and give zero weight to other points but this is not true for logistic model where all points are considered hence SVM's decision boundary remains unaffected by distant points.

b) Kernel tricks allow to operate in the original feature without computing the coordinate of data in higher dimensional space, In SVM Kernel is basically the dot product of the features, which is computationally less expensive to calculate.

Problem 8

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder

path = '/content/drive/MyDrive/sem 7/ID5055/Assignment 4/Traffic_data.csv'
```

```
data = pd.read_csv(path)
```

```
data.head(10)
```

	Time	Date	Day of the week	CarCount	BikeCount	BusCount	TruckCount	Total	Traffic Situation
0	12:00:00 AM	10	Tuesday	31	0	4	4	39	low
1	12:15:00 AM	10	Tuesday	49	0	3	3	55	low
2	12:30:00 AM	10	Tuesday	46	0	3	6	55	low
3	12:45:00 AM	10	Tuesday	51	0	2	5	58	low
4	1:00:00 AM	10	Tuesday	57	6	15	16	94	normal
5	1:15:00 AM	10	Tuesday	44	0	5	4	53	low
6	1:30:00 AM	10	Tuesday	37	0	1	4	42	low
7	1:45:00 AM	10	Tuesday	42	4	4	5	55	low
8	2:00:00 AM	10	Tuesday	51	0	9	7	67	low
9	2:15:00 AM	10	Tuesday	34	0	4	7	45	low

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2976 entries, 0 to 2975
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Time               2976 non-null   object
 1   Date               2976 non-null   int64 
 2   Day of the week    2976 non-null   object
 3   CarCount           2976 non-null   int64 
 4   BikeCount          2976 non-null   int64 
 5   BusCount           2976 non-null   int64 
 6   TruckCount         2976 non-null   int64 
 7   Total              2976 non-null   int64 
 8   Traffic Situation  2976 non-null   object
dtypes: int64(6), object(3)
memory usage: 209.4+ KB
```

```
df = data
```

```
# Converting string values to numeric, Monday = 0 and Sunday = 6

df['Day of the week cat'] = LabelEncoder().fit_transform(df['Day of the week'])
col = df.pop('Day of the week cat')
data.insert(2, col.name, col)
```

```
df['Time'] = pd.to_datetime(df['Time']).apply(lambda x: x.hour)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2976 entries, 0 to 2975
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Time                 2976 non-null   int64 
 1   Date                 2976 non-null   int64 
 2   Day of the week cat  2976 non-null   int64 
```

```
 3   Day of the week      2976 non-null   object
 4   CarCount             2976 non-null   int64
 5   BikeCount            2976 non-null   int64
 6   BusCount             2976 non-null   int64
 7   TruckCount           2976 non-null   int64
 8   Total                2976 non-null   int64
 9   Traffic Situation    2976 non-null   object
dtypes: int64(8), object(2)
memory usage: 232.6+ KB
```

```
X = df.drop(['Traffic Situation', 'Day of the week'], axis = 1)
Y = df['Traffic Situation']
```

1. Split the dataset into train and test set (train size= 0.8, random state = 42) and train a random forest classifier using the train set. Plot a confusion matrix using the test set for prediction.

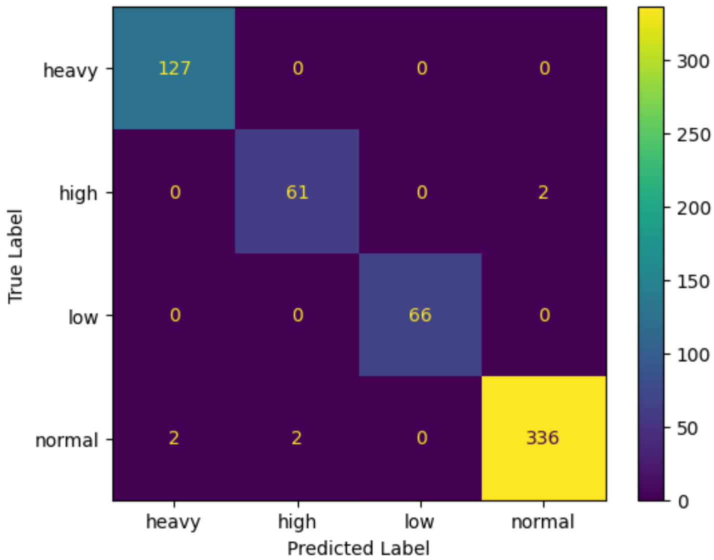
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size= 0.8, random_state = 42)
```

```
rf = RandomForestClassifier()
rf.fit(X_train, Y_train)
Y_pred = rf.predict(X_test)
```

```
cm = confusion_matrix(Y_test, Y_pred)
cm_display = ConfusionMatrixDisplay(cm, display_labels = rf.classes_).plot()
plt.rcParams.update({'font.size': 14})
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

print('\n*****************************************************************************\n')

print(classification_report(Y_test, Y_pred))
```




```
*****************************************************************************

              precision    recall  f1-score   support

       heavy       0.98      1.00      0.99       127
        high       0.97      0.97      0.97        63
         low       1.00      1.00      1.00        66
      normal       0.99      0.99      0.99       340

    accuracy                           0.99       596
   macro avg       0.99      0.99      0.99       596
weighted avg       0.99      0.99      0.99       596
```

2. Use a weighted random forest classifier with weights based on the frequency of the corresponding class. Plot a confusion matrix and report your observation by comparing the results with the previous results.

```
print(f'The classes are:\n{rf.classes_}')
```

```
The classes are:
['heavy' 'high' 'low' 'normal']
```

```
freq = dict(Y.value_counts())
print(freq)
```

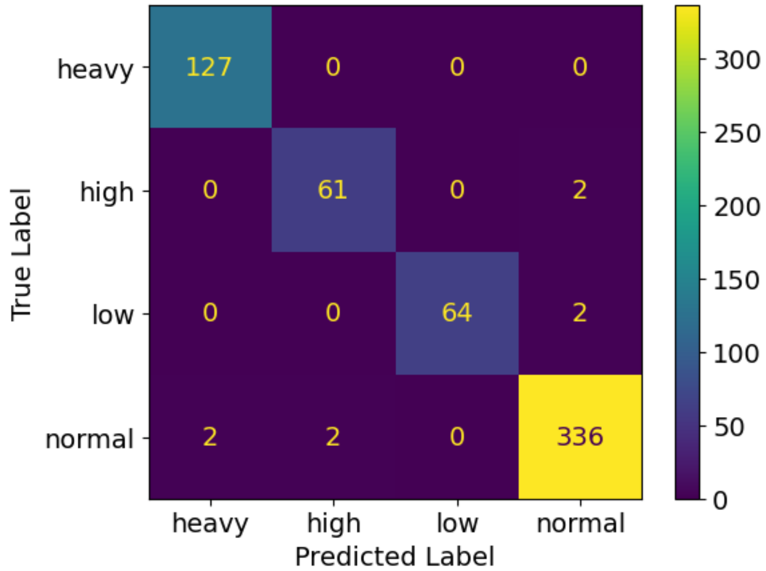
```
{'normal': 1669, 'heavy': 682, 'high': 321, 'low': 304}
```

```
rf_weighted = RandomForestClassifier(class_weight = freq)
rf_weighted.fit(X_train, Y_train)
Y_pred_weighted = rf_weighted.predict(X_test)
```

```
cm = confusion_matrix(Y_test, Y_pred_weighted)
cm_display = ConfusionMatrixDisplay(cm, display_labels = rf_weighted.classes_).plot()
plt.rcParams.update({'font.size': 14})
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

print('\n****************************************************************************\n')

print(classification_report(Y_test, Y_pred_weighted))
```




```
****************************************************************************

              precision    recall  f1-score   support

       heavy       0.98      1.00      0.99       127
        high       0.97      0.97      0.97        63
         low       1.00      0.97      0.98        66
      normal       0.99      0.99      0.99       340

    accuracy                           0.99       596
   macro avg       0.99      0.98      0.98       596
weighted avg       0.99      0.99      0.99       596
```

3. Use the trained classifier model to report the important features based on impurity metric

```
rf.feature_importances_
```

```
array([0.04223248, 0.0091661 , 0.01331356, 0.14465027, 0.05917621,
       0.15528884, 0.22902063, 0.3471519 ])
```

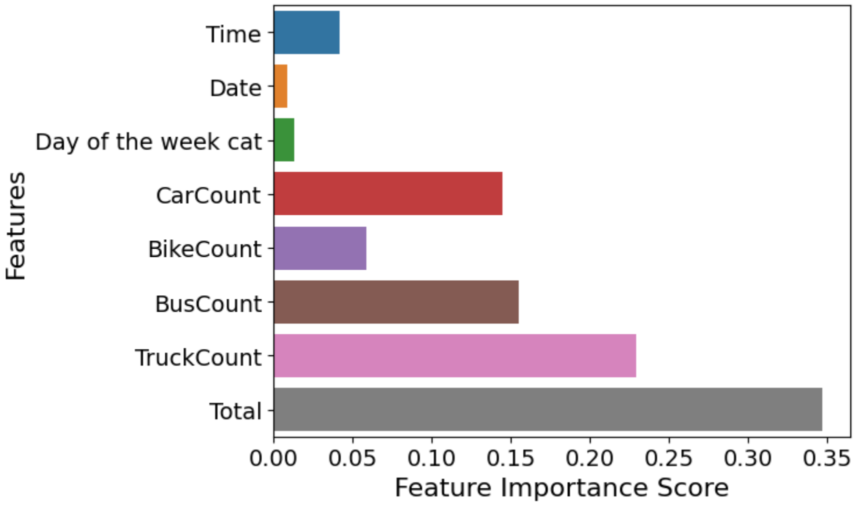
```
# Model 1
feature_scores = pd.Series(rf.feature_importances_, index = X_train.columns)
print(feature_scores)
```

```
Time                   0.042232
Date                   0.009166
Day of the week cat    0.013314
CarCount               0.144650
BikeCount              0.059176
BusCount               0.155289
```

```
TruckCount              0.229021
Total                   0.347152
dtype: float64
```

```
sns.barplot(x = feature_scores, y = feature_scores.index)
plt.xlabel('Feature Importance Score', fontsize = 16)
plt.ylabel('Features', fontsize = 16)
```

```
Text(0, 0.5, 'Features')
```

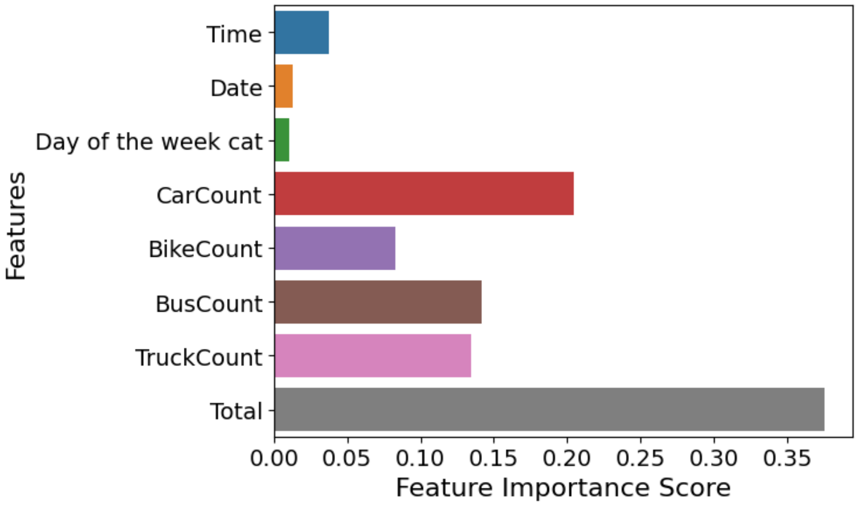



```
# Model 2
feature_scores_w = pd.Series(rf_weighted.feature_importances_, index = X_train.columns)
print(feature_scores)
```

```
Time                    0.037775
Date                    0.012922
Day of the week cat     0.010682
CarCount                0.204475
BikeCount               0.082924
BusCount                0.141365
TruckCount              0.134151
Total                   0.375706
dtype: float64
```

```
sns.barplot(x = feature_scores_w, y = feature_scores.index)
plt.xlabel('Feature Importance Score', fontsize = 16)
plt.ylabel('Features', fontsize = 16)
```

```
Text(0, 0.5, 'Features')
```



Observations

1. Since we are predicting the traffic condition, so it is reasonable why 'Total' feature has the highest feature importance.

2. Other important features are the count of heavy vehicles like cars, buses, and trucks, as they are main reason for traffic.
3. Time, Date, and Day of week got low feature score, however time is also an important feature in determining the traffic condition.