

▼ 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)

```

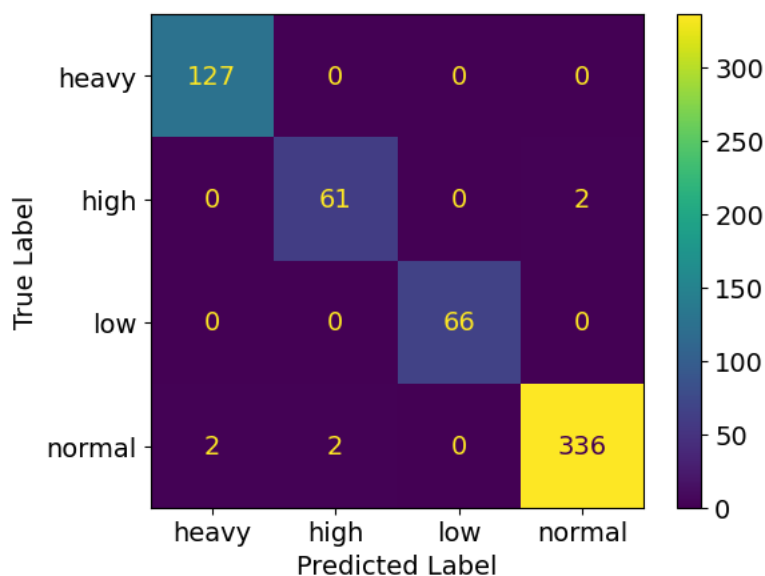
```

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)
```

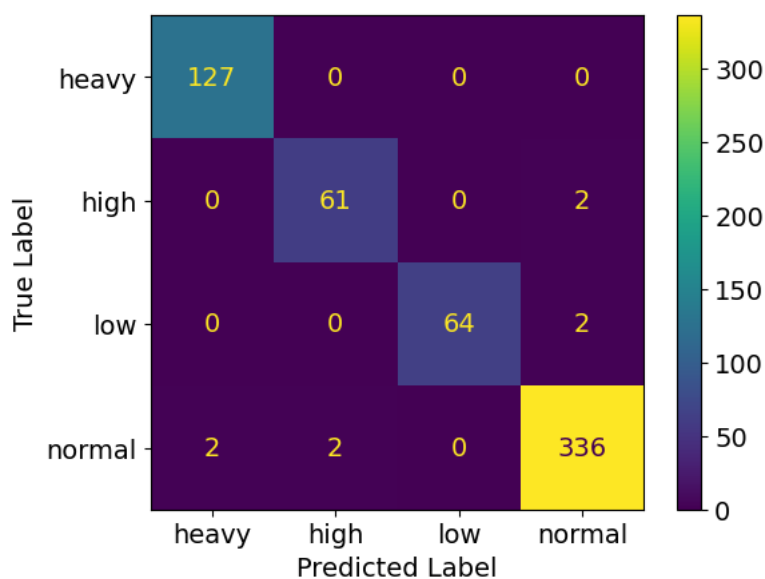
```
{'normal': 1669, 'heavy': 682, 'high': 321, 'low': 304}
```

```
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([0.04492375, 0.0107539 , 0.01454279, 0.18717177, 0.05708706,
       0.14582189, 0.20602223, 0.33367662])
```

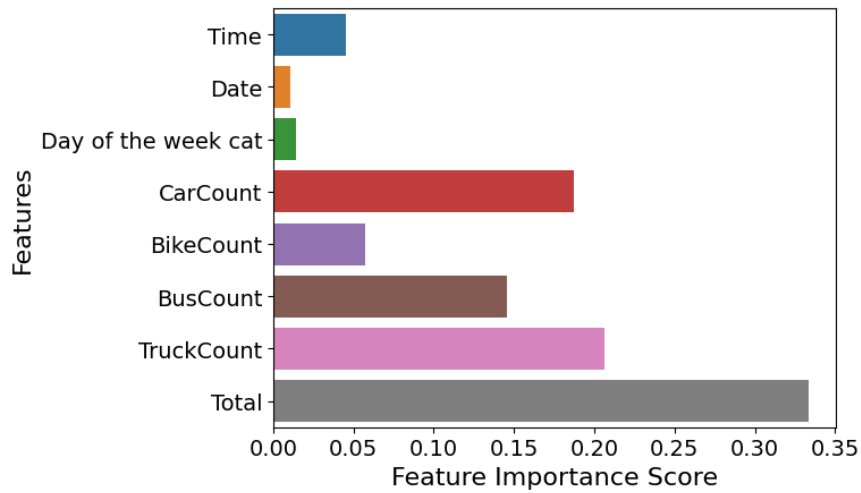
```
# Model 1
feature_scores = pd.Series(rf.feature_importances_, index = X_train.columns)
print(feature_scores)
```

```
Time          0.044924
Date          0.010754
Day of the week cat  0.014543
CarCount      0.187172
BikeCount     0.057087
BusCount      0.145822
```

```
TruckCount    0.206022
Total         0.333677
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')
```



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.