log-car-predcition-1

November 5, 2024

[ ]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```python
df = pd.read_csv('/content/cars.csv')
```

[ ]:
```python
df.head()
```

[ ]:

	buying	maint	doors	persons	lug_boot	safety	class
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc

[ ]:
```python
df.isnull().sum()
```

[ ]:
```
buying      0
maint       0
doors       0
persons     0
lug_boot    0
safety      0
class       0
dtype: int64
```

[ ]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   buying    1728 non-null   object
 1   maint     1728 non-null   object
 2   doors     1728 non-null   object
 3   persons   1728 non-null   object
```

```
 4   lug_boot  1728 non-null   object
 5   safety    1728 non-null   object
 6   class     1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

1 Data Preparation

[ ]:
```
# encode the categorical columns

from sklearn.preprocessing import LabelEncoder

label_encoder = {}
for column in df.columns:
  le = LabelEncoder()
  df[column] = le.fit_transform(df[column])
  label_encoder[column] = le

df.head()
```

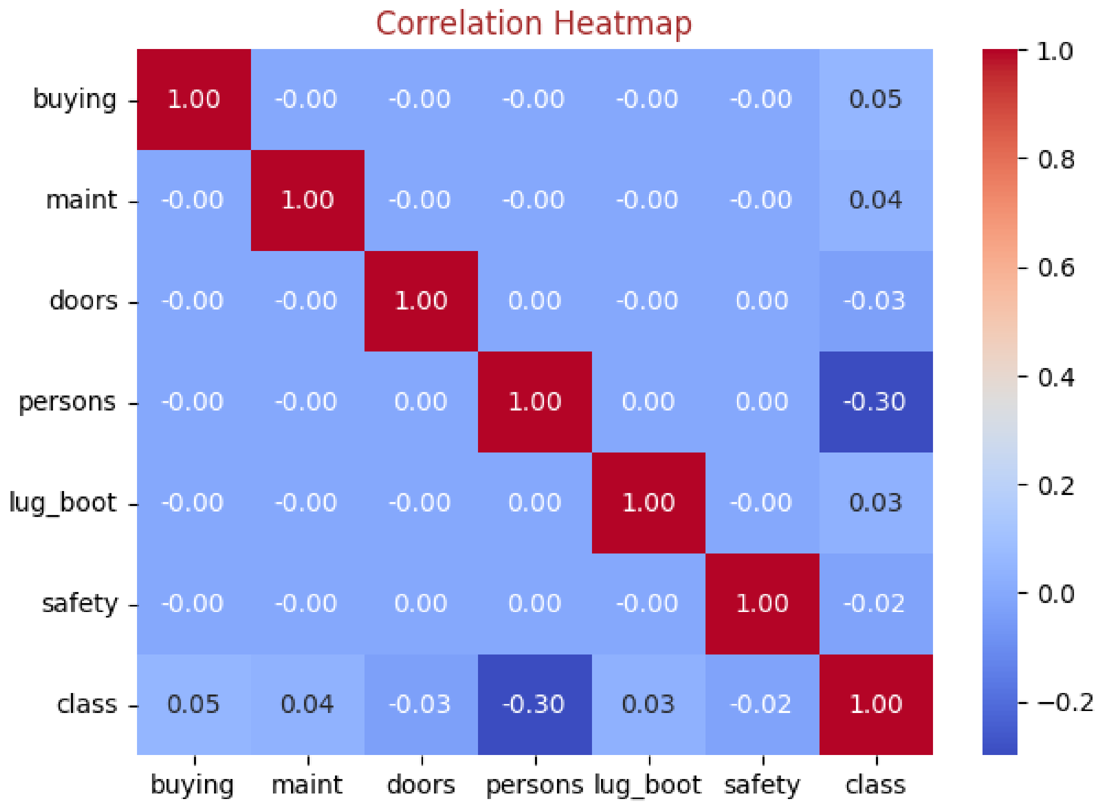
[ ]:

	buying	maint	doors	persons	lug_boot	safety	class
0	3	3	0	0	2	1	2
1	3	3	0	0	2	2	2
2	3	3	0	0	2	0	2
3	3	3	0	0	1	1	2
4	3	3	0	0	1	2	2

[ ]:
```
# check the correlation between the features to see if any strong relationship␣
↪exists.

# Correlation heatmap

plt.figure(figsize=(7,5))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap', color = 'brown')
plt.show()
```



1.0.1 The purpose of this plot is to visualize the co-relation between each feature -

1. buying & maint - they have strong positive co-relation. as the buying price increases, maintenance cost also tends to increase.

2. buying & class - they both indicating the strong relationship as the buying price is increasing, the car class is also likely to tends to go up.

3. While doors, persons, lug_boot, safety have the weaker correlations with each other and with the buying and maint. but we have oberserved the their relationships with the class.

2 Data Preparation

[ ]:
```
X = df.drop('class', axis = 1)
X
```

[ ]:

	buying	maint	doors	persons	lug_boot	safety
0	3	3	0	0	2	1
1	3	3	0	0	2	2

```
2           3      3      0       0         2       0
3           3      3      0       0         1       1
4           3      3      0       0         1       2
…      …    …    …     …      …     …
1723        1      1      3       2         1       2
1724        1      1      3       2         1       0
1725        1      1      3       2         0       1
1726        1      1      3       2         0       2
1727        1      1      3       2         0       0

[1728 rows x 6 columns]
```

[ ]:
```
y = df[['class']]
y
```

[ ]:
```
      class
0         2
1         2
2         2
3         2
4         2
…     …
1723      1
1724      3
1725      2
1726      1
1727      3

[1728 rows x 1 columns]
```

[ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 45)
X_train
```

[ ]:
```
      buying  maint  doors  persons  lug_boot  safety
1151       2      2      2        1         0       0
342        3      1      0        2         2       1
1698       1      1      2        2         0       1
472        0      3      1        1         1       2
1264       2      1      2        2         1       2
…      …    …    …     …      …     …
607        0      0      2        1         1       2
1568       1      2      2        0         2       0
1667       1      1      1        2         2       0
414        3      1      3        1         2       1
```

```
971        2      3      3       2        0       0

[1382 rows x 6 columns]
```

[ ]:
```
X_test
```

[ ]:
```
      buying  maint  doors  persons  lug_boot  safety
467        0      3      1        0         0       0
617        0      0      2        2         1       0
229        3      2      0        1         1       2
1039       2      0      2        1         1       2
1426       1      0      0        2         1       2
…       …    …    …      …       …     …
762        0      1      0        0         0       1
1224       2      1      1        1         2       1
681        0      2      1        0         0       1
1110       2      2      1        0         1       1
632        0      0      3        1         2       0

[346 rows x 6 columns]
```

3 Feature Scaling

[ ]:
```python
from sklearn.preprocessing import StandardScaler
```

[ ]:
```python
scaler = StandardScaler()
```

[ ]:
```python
X_train_scaled = scaler.fit_transform(X_train)
X_train_scaled
```

[ ]:
```
array([[ 0.43237595,  0.44876822,  0.41164022, -0.00354744, -1.24878941,
        -1.21862035],
       [ 1.32690226, -0.44617793, -1.37731195,  1.22209232,  1.20441167,
         0.01247126],
       [-0.46215035, -0.44617793,  0.41164022,  1.22209232, -1.24878941,
         0.01247126],
       …,
       [-0.46215035, -0.44617793, -0.48283586,  1.22209232,  1.20441167,
        -1.21862035],
       [ 1.32690226, -0.44617793,  1.30611631, -0.00354744,  1.20441167,
         0.01247126],
       [ 0.43237595,  1.34371437,  1.30611631,  1.22209232, -1.24878941,
        -1.21862035]])
```

[ ]:
```python
X_test_scaled = scaler.transform(X_test)
X_test_scaled
```

[ ]:
```
array([[-1.35667665,  1.34371437, -0.48283586, -1.2291872 , -1.24878941,
        -1.21862035],
       [-1.35667665, -1.34112408,  0.41164022,  1.22209232, -0.02218887,
        -1.21862035],
       [ 1.32690226,  0.44876822, -1.37731195, -0.00354744, -0.02218887,
         1.24356288],
       …,
       [-1.35667665,  0.44876822, -0.48283586, -1.2291872 , -1.24878941,
         0.01247126],
       [ 0.43237595,  0.44876822, -0.48283586, -1.2291872 , -0.02218887,
         0.01247126],
       [-1.35667665, -1.34112408,  1.30611631, -0.00354744,  1.20441167,
        -1.21862035]])
```

4 Model Training

[ ]:
```python
from sklearn.linear_model import LogisticRegression
```

[ ]:
```python
lg = LogisticRegression()
lg
```

[ ]:
```
LogisticRegression()
```

[ ]:
```python
lg.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1339:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

[ ]:
```
LogisticRegression()
```

[ ]:
```python
log_reg_predict = lg.predict(X_test)
log_reg_predict
```

[ ]:
```
array([2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       3, 2, 0, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0, 0, 2, 2, 2, 2, 2, 2, 2,
       0, 2, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 0, 3, 0, 2, 2, 0,
       2, 2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2,
```

```
0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 0, 2,
2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 3, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2,
2, 2, 2, 0, 2, 2, 0, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
0, 2, 2, 2, 3, 3, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2,
0, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2])
```

5 Model Evaluation

[ ]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,␣
↪recall_score, f1_score
```

[ ]:
```
accuracy_log_reg  =  accuracy_score(y_test, log_reg_predict)
precision_log_reg  =  precision_score(y_test, log_reg_predict, average='macro'␣
↪, zero_division = 1)
recall_log_reg  =  recall_score(y_test, log_reg_predict, average='macro',␣
↪zero_division = 1)
f1_log_reg  =  f1_score(y_test, log_reg_predict, average='macro')

print("Accuracy :" ,  accuracy_log_reg)
print("Precision :" ,  precision_log_reg)
print("Recall :" ,  recall_log_reg)
print("F1 Score :" ,  f1_log_reg)
```

```
Accuracy : 0.7254335260115607
Precision : 0.6788002980625931
Recall : 0.36078737467626354
F1 Score : 0.3726813809698044
```

6 Random Forest Classifier

[ ]:
```
from sklearn.ensemble import  RandomForestClassifier
```

[ ]:
```
rf = RandomForestClassifier(random_state = 45, class_weight = 'balanced')
```

[ ]:
```
rf.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  return fit_method(estimator, *args, **kwargs)
```

[ ]:
```
RandomForestClassifier(class_weight='balanced', random_state=45)
```

[ ]:
```
rf_predict = rf.predict(X_test_scaled)
rf_predict
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does
not have valid feature names, but RandomForestClassifier was fitted with feature
names
  warnings.warn(
```

[ ]:
```
array([2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 0, 2, 2, 2, 0, 2, 2, 2,
       0, 2, 0, 2, 2, 2, 0, 0, 2, 2, 2, 2, 2, 2, 3, 3, 2, 0, 2, 2, 2, 2,
       0, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2,
       2, 1, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 0, 2, 2, 0, 2, 2, 0,
       2, 2, 2, 2, 0, 2, 0, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0,
       2, 2, 2, 2, 2, 2, 0, 2, 2, 0, 2, 2, 3, 2, 0, 2, 0, 2, 0, 2, 2, 2,
       0, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0,
       2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 0, 0, 2, 2, 2, 0,
       2, 2, 2, 0, 3, 2, 2, 2, 2, 0, 2, 2, 2, 2, 0, 0, 2, 0, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 0, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 0, 0, 3, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 0, 2, 2, 2, 2, 0, 0, 2, 2, 2, 2,
       0, 2, 2, 2, 0, 0, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2,
       2, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 0, 2, 2, 2, 2, 2, 2,
       0, 0, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2])
```

[ ]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,␣
↪recall_score, f1_score
```

[ ]:
```
accuracy_rf  =  accuracy_score(y_test, log_reg_predict)
precision_rf  =  precision_score(y_test, log_reg_predict, average='macro',␣
↪zero_division = 1)
recall_rf  =  recall_score(y_test, log_reg_predict, average='macro',␣
↪zero_division = 1)
f1_rf  =  f1_score(y_test, log_reg_predict, average='macro')

print("Accuracy :" ,  accuracy_rf)
print("Precision :" ,  precision_rf)
print("Recall :" ,  recall_rf)
print("F1 Score :" ,  f1_rf)
```

```
Accuracy : 0.7254335260115607
Precision : 0.6788002980625931
Recall : 0.36078737467626354
F1 Score : 0.3726813809698044
```

[ ]:
```
results = {
    'Model'  :  ['Logistic Regression',  'Random Forest'],
```

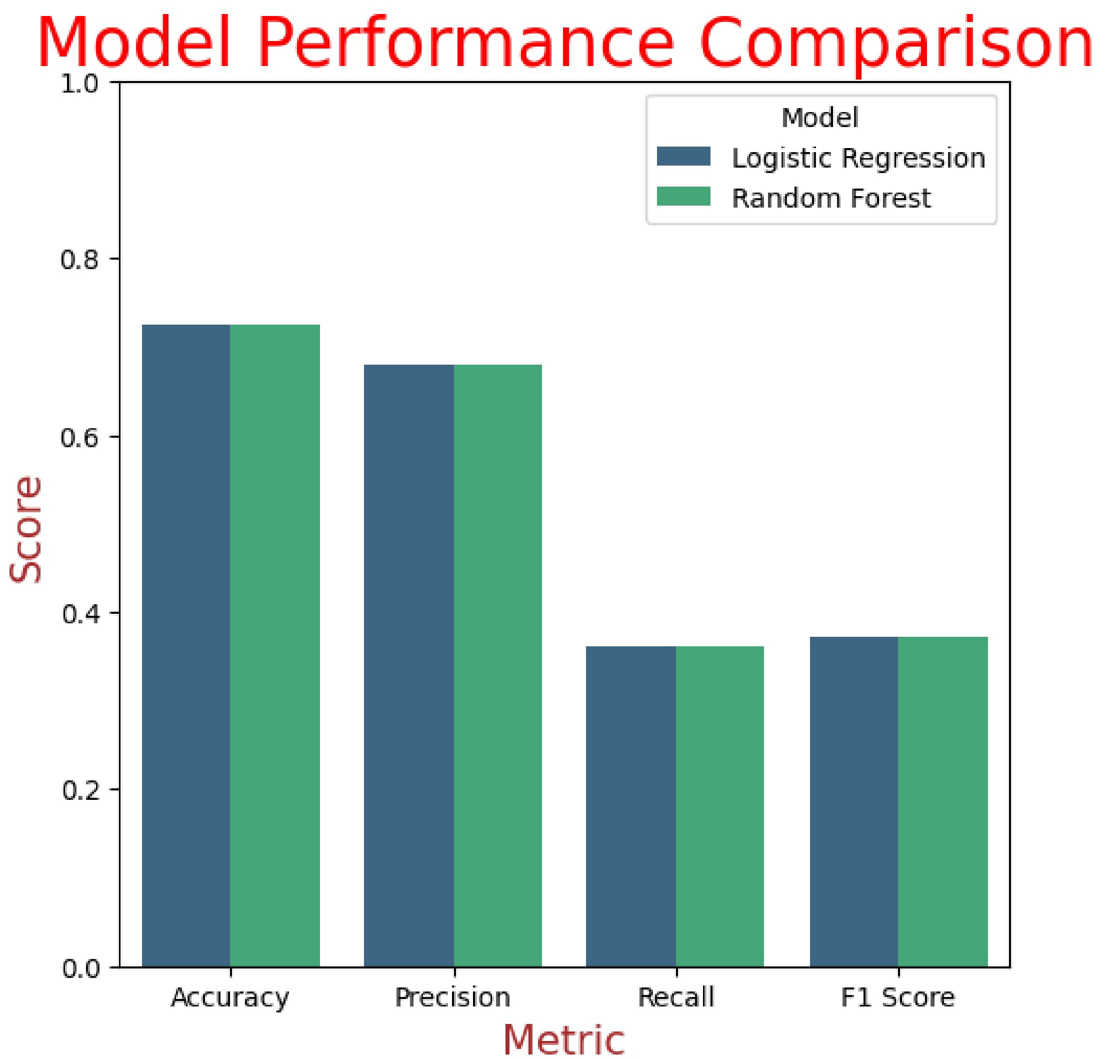
```
    'Accuracy' :  [ accuracy_log_reg,  accuracy_log_reg],
    'Accuracy' :  [ accuracy_log_reg,  accuracy_rf],
    'Precision' :  [ precision_log_reg,  precision_rf],
    'Recall' :  [ recall_log_reg,  recall_rf],
    'F1 Score' :  [ f1_log_reg, f1_rf]

}
comparison_df = pd.DataFrame(results)
print(comparison_df)
```

```
                 Model  Accuracy  Precision    Recall  F1 Score
0  Logistic Regression  0.725434     0.6788  0.360787  0.372681
1        Random Forest  0.725434     0.6788  0.360787  0.372681
```

[ ]:
```
comparison_melted = comparison_df.melt(id_vars="Model", var_name="Metric",␣
 ↪value_name="Score")

plt.figure(figsize=(6,6))
sns.barplot(data=comparison_melted, x="Metric", y="Score", hue="Model",␣
 ↪palette="viridis")
plt.title("Model Performance Comparison", color = "red", size = 25)
plt.ylabel("Score", color = "brown", size = 15)
plt.xlabel("Metric", color = "brown", size = 15)
plt.ylim(0, 1)
plt.legend(title="Model")
plt.show()
```



7 Feature Selection for the logistic classification model

[ ]:
```python
# for logistic regression

from sklearn.feature_selection import SelectKBest, f_classif

X = df.drop('class', axis = 1)
y = df['class']
```

[ ]:
```python
selector = SelectKBest(f_classif, k = 6)
X_new = selector.fit_transform(X,y)
```

[ ]:
```python
selected_features = X.columns[selector.get_support()]
```

```
[ ]: # get the scores of the all features
feature_scores = selector.scores_
```

```
[ ]: feature_score_df = pd.DataFrame({'Feature' : X.columns, 'Score' :␣
      ↪feature_scores})
feature_score_df
```

[ ]:

	Feature	Score
0	buying	2.624840
1	maint	4.143259
2	doors	2.764439
3	persons	105.050194
4	lug_boot	17.616190
5	safety	38.858267

8 Feature selection for the Random Forest classifier

```
[ ]: from sklearn.feature_selection import RFE
```

```
[ ]: selector = RFE(rf, n_features_to_select = 3, step = 1)
X_new = selector.fit_transform(X,y)
```

```
[ ]: selected_features = X.columns[selector.get_support()]
print(selected_features)
```

```
Index(['buying', 'maint', 'safety'], dtype='object')
```

```
[ ]: # get the feature ranking
feature_ranking = selector.ranking_
```

```
[ ]: feature_ranking_df = pd.DataFrame({'Feature' : X.columns, 'Ranking' :␣
      ↪feature_ranking})
feature_ranking_df
```

[ ]:

	Feature	Ranking
0	buying	1
1	maint	1
2	doors	4
3	persons	3
4	lug_boot	2
5	safety	1

9 Conclusion

###The analysis of the car dataset revealed significant insights into the factors influencing car acceptability

The heatmap displays how strongly different feature is related to each other.Suggesting that "Buying Price", "Maintenance Cost" and "Car Class" have strong positive co-relation between them.

9.0.1 Feature Selection

Feature Selection using both Filter and Wrapping Method reveals different sets of crucial features for predicting the target feature(class).

Logistic Regression identified - persons, lug_boot & safety are the most influential features suggesting that passengers capacity, luggage space and safety are the key factors in this model's predictions.

While, Random Forest through Recursive Feature Elimination(RFE) highlighted buying, maint, & safety suggesting that purchase price, maintenance cost & safety are the key factors for this model's predictions.

9.0.2 Model Performance

9.0.3 Logistic Regression

1) It is a multi-classification model,The purpose of this model is to predict acceptability class of a car(unacc, acc, good, vgood) based on its features.

2) The model has achieved an accuracy of 72.54%, indicating a relatively good performance in classifying cars.

3) While this accuracy is promising, there is potential for further improvement. The model's performance was also evaluated using precision, recall, and F1-score, which provided insights into its strengths and limitations."

9.0.4 Random Forest Classifier

1) The model was employed to predict the acceptability class of a cars(unacc, acc, good, vgood) based on the same set of features.

2) The ensemble learning method combines multiple decision trees to improve prediction accuracy and robustness.

3) Trained with a balance class weight to address potential class imbalance in the dataset.

4) The model has achieved 72.54% accuracy, comparable to the Logistic Regression Model. This indicates that both models, despite their different approaches, demonstrate similar effectiveness in predicting car acceptabilty.

9.0.5 This analysis explored the use of Logistic Regression and Random Forest models to predict car acceptability based on various features. Both models demonstrated comparable performance, achieving similar accuracy, precision, recall, and F1-score.

[ ]:
```

```