

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

```
data = pd.read_csv('cars.csv')
```

```
data.head()
```

	vhigh	vhigh.1	2	2.1	small	low	unacc
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   vhigh       1727 non-null   object
1   vhigh.1     1727 non-null   object
2   2           1727 non-null   object
3   2.1         1727 non-null   object
4   small       1727 non-null   object
5   low         1727 non-null   object
6   unacc       1727 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

```
data.columns = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'acceptability']
```

```
for col in data.columns:
    if data[col].dtype == 'object':
        print(f"{col}: {data[col].value_counts()}")
```

```
buying: high      432
      med       432
      low       432
      vhigh     431
      Name: buying, dtype: int64
maint: high      432
      med       432
      low       432
      vhigh     431
      Name: maint, dtype: int64
doors: 3         432
      4         432
      5more     432
      2         431
      Name: doors, dtype: int64
persons: 4       576
      more     576
      2       575
      Name: persons, dtype: int64
lug_boot: med     576
      big       576
      small     575
      Name: lug_boot, dtype: int64
safety: med      576
      high     576
      low     575
      Name: safety, dtype: int64
acceptability: unacc  1209
      acc       384
      good      69
      vgood     65
      Name: acceptability, dtype: int64
```

```
data.isnull().sum()
```

```

buying      0
maint       0
doors       0
persons     0
lug_boot    0
safety      0
acceptability 0
dtype: int64

```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(data.drop(['acceptability'], axis=1), data['acceptability'], test_size=0.4, random_state
```

```

print("X_train", X_train.shape)
print("X_test", X_test.shape)

```

```

X_train (1036, 6)
X_test (691, 6)

```

```

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X_train = X_train.apply(le.fit_transform)
X_test = X_test.apply(le.fit_transform)

```

```

from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
rf_100 = RandomForestClassifier(n_estimators=100)
rf_100.fit(X_train, y_train)

```

```

▼ RandomForestClassifier
RandomForestClassifier()

```

```
importance_scores = rf_100.feature_importances_
```

```

feature_importance_df = pd.DataFrame({'Feature': X_train.columns, 'Importance': importance_scores})
feature_importance_df = feature_importance_df.sort_values('Importance', ascending=False)

```

```
print(feature_importance_df)
```

```

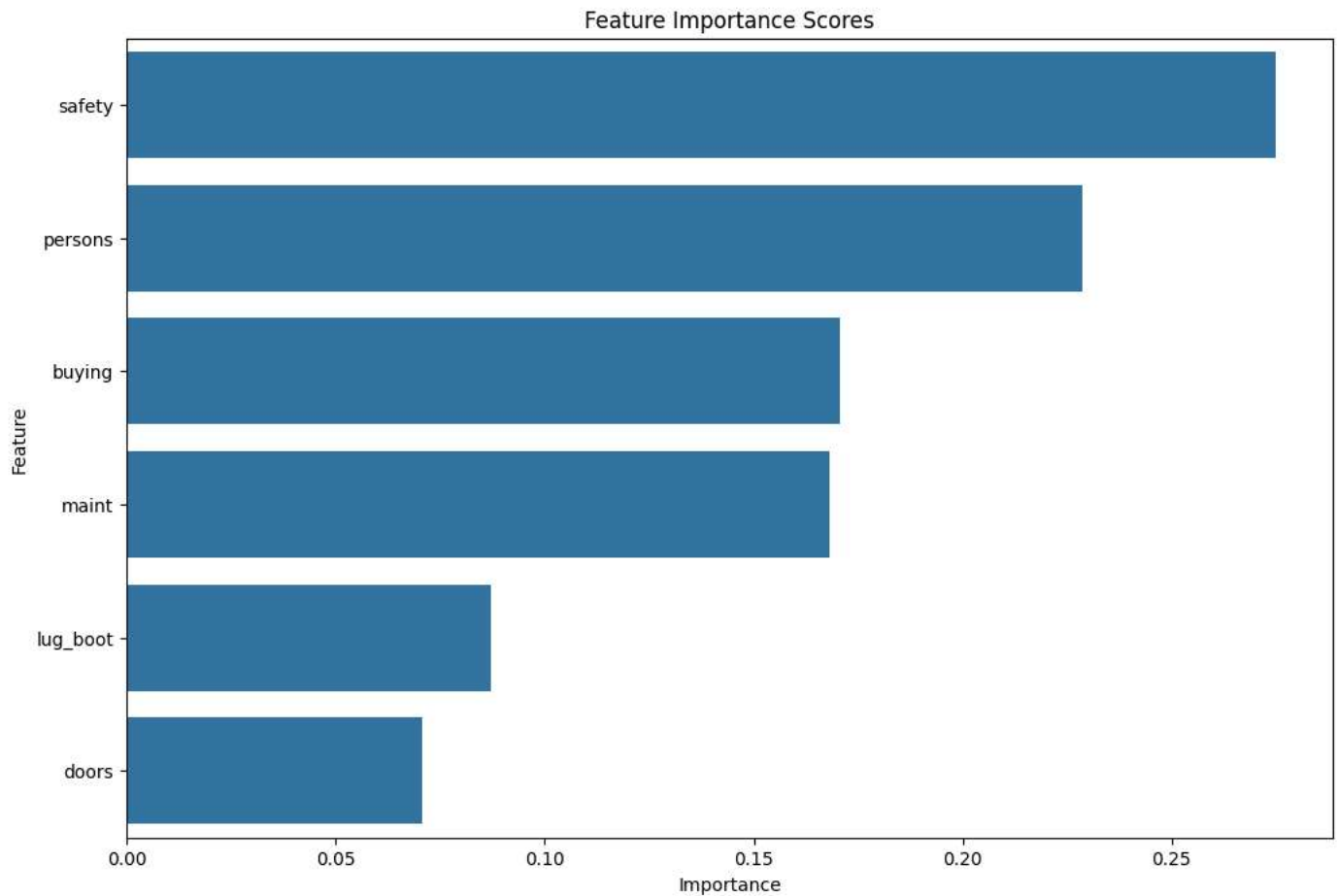
   Feature  Importance
5  safety    0.274815
3  persons    0.228581
0  buying    0.170653
1  maint     0.168176
4  lug_boot    0.086986
2  doors     0.070789

```

```

plt.figure(figsize=(12, 8))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
plt.title('Feature Importance Scores')
plt.show()

```



```
selected_features = feature_importance_df['Feature'][:3]
```

```
X_train_selected = X_train[selected_features]
```

```
X_test_selected = X_test[selected_features]
```

```
rf_selected = RandomForestClassifier(n_estimators=100)
```

```
rf_selected.fit(X_train_selected, y_train)
```

▼ RandomForestClassifier

RandomForestClassifier()

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

```
print(confusion_matrix(y_test, rf_selected.predict(X_test_selected)))
```

```
[[115  0  35  7]
 [ 24  0  0  4]
 [ 32  0 440  1]
 [ 24  0  0  9]]
```

```
print(classification_report(y_test, rf_selected.predict(X_test_selected)))
```

	precision	recall	f1-score	support
acc	0.59	0.73	0.65	157
good	0.00	0.00	0.00	28
unacc	0.93	0.93	0.93	473
vgood	0.43	0.27	0.33	33
accuracy			0.82	691
macro avg	0.49	0.48	0.48	691
weighted avg	0.79	0.82	0.80	691

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
_warn_prf(average, modifier, msg_start, len(result))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
```

```
_warn_prf(average, modifier, msg_start, len(result))  
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d  
_warn_prf(average, modifier, msg_start, len(result))
```

```
print(accuracy_score(y_test, rf_selected.predict(X_test_selected)))
```

```
0.8162083936324168
```