DATA9001

FUNDAMENTALS OF DATA SCIENCE

Problem 3

In this report, various features of cars are analysed to predict the quality of cars through the implementation of ML model. Two different classifiers are selected and implemented: Decision Tree Classifier and Naive Bayes which are compared to identify the better model for the evaluation of the overall car quality.

I. Data Preprocessing

By processing data, we make sure that the given data set contains all features on a similar scale by cleaning and normalising the data. The following steps were taken to pre-process the data to make it ready for model training and evaluation:

a. Converted all columns to lowercase

```
# Load the data
df = pd.read_csv('/Users/CHIPPY/Documents/car_problem3.csv')
df
```

	buying	maint	doors	persons	lug_boot	safety	class
0	HIGH	high	5more	2	med	LOW	unacc
1	high	med	3	more	BIG	LOW	unacc
2	vhigh	MED	3	more	small	med	unacc
3	vhigh	med	4	2	small	LOW	unacc
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...
2068	LOW	high	3	2	SMALL	low	unacc
2069	HIGH	high	4	2	big	med	UNACC
2070	0	0	NaN	NaN	0	0	NaN
2071	high	high	3	4	MED	high	acc
2072	med	med	5more	4	med	med	acc

2073 rows × 7 columns

```
# Convert all columns to lowercase
df = df.map(lambda s:s.lower() if type(s) == str else s)
```

```
df
```

	buying	maint	doors	persons	lug_boot	safety	class
0	high	high	5more	2	med	low	unacc
1	high	med	3	more	big	low	unacc
2	vhigh	med	3	more	small	med	unacc
3	vhigh	med	4	2	small	low	unacc
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...
2068	low	high	3	2	small	low	unacc
2069	high	high	4	2	big	med	unacc
2070	0	0	NaN	NaN	0	0	NaN
2071	high	high	3	4	med	high	acc
2072	med	med	5more	4	med	med	acc

2073 rows × 7 columns

Figure 8: Before and after converting all columns to lower space

b. Replaced ‘more’ and ‘5more’ values in the columns ‘doors’ and ‘persons’ with the highest number (in both cases ‘5’)

```
# Replace 'more' and '5more' with a high number (e.g., 5)
df.replace(['more', '5more'], 5, inplace=True)
df
```

	buying	maint	doors	persons	lug_boot	safety	class
0	high	high	5	2	med	low	unacc
1	high	med	3	5	big	low	unacc
2	vhigh	med	3	5	small	med	unacc
3	vhigh	med	4	2	small	low	unacc
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...
2068	low	high	3	2	small	low	unacc
2069	high	high	4	2	big	med	unacc
2070	0	0	NaN	NaN	0	0	NaN
2071	high	high	3	4	med	high	acc
2072	med	med	5	4	med	med	acc

2073 rows × 7 columns

Figure 9: Data frame after cleaning the unclear values

c. Converted 'doors' and 'persons' to numeric values
d. Replaced ‘NaN’ with ‘0’
e. Deleted the rows where all values are ‘NaN’

```
# Drop the rows where all values are 'NaN'
df.dropna(how='all', inplace=True)
df
```

	buying	maint	doors	persons	lug_boot	safety	class
0	high	high	5.0	2.0	med	low	unacc
1	high	med	3.0	5.0	big	low	unacc
2	vhigh	med	3.0	5.0	small	med	unacc
3	vhigh	med	4.0	2.0	small	low	unacc
5	high	med	5.0	2.0	med	high	unacc
...	...	...	...	...	...	...	...
2067	med	med	4.0	2.0	big	high	unacc
2068	low	high	3.0	2.0	small	low	unacc
2069	high	high	4.0	2.0	big	med	unacc
2071	high	high	3.0	4.0	med	high	acc
2072	med	med	5.0	4.0	med	med	acc

1901 rows × 7 columns

Figure 10: Data frame after deletion of 'NaN Value rows

f. Converted categorical data into numerical data

```
# Convert categorical data to numerical data
le = LabelEncoder()
df = df.apply(le.fit_transform)
df
```

	buying	maint	doors	persons	lug_boot	safety	class
0	0	0	4	1	1	1	2
1	0	2	2	3	0	1	2
2	3	2	2	3	2	2	2
3	3	2	3	1	2	1	2
5	0	2	4	1	1	0	2
...	...	...	...	...	...	...	...
2067	2	2	3	1	0	0	2
2068	1	0	2	1	2	1	2
2069	0	0	3	1	0	2	2
2071	0	0	2	2	1	0	0
2072	2	2	4	2	1	2	0

1901 rows × 7 columns

Figure 11: Data Frame after conversion to numerical data

II. Data Splitting, Parameter Tuning, Model Selection, Training and Implementation

The data frame is split into the 2 sets: one for training the model (80%) and the other for testing the data (20%). After splitting the data frame into 'X_train' and 'X_test', the Decision Tree Classifier function and the Naïve Bayes Classifier functions are trained and evaluated to find the most accurate model.

```
# Split the data into features and target variable
X = df.drop('class', axis=1)
Y = df['class']

# Split the dataset into a training set and a test set
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# Create the Naive Bayes classifier
clfr = GaussianNB()

# Train the model
clfr.fit(X_train, Y_train)
```

GaussianNB

GaussianNB()

Figure 12: Data Splitting and Naive Bayes Classifier Model

```
# Split the data into features and target variable
X = df.drop('class', axis=1)
Y = df['class']

# Split the dataset into a training set and a test set
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# Create the Decision Tree classifier
clfr = DecisionTreeClassifier()

# Train the model
clfr.fit(X_train, Y_train)
```

DecisionTreeClassifier

DecisionTreeClassifier()

Figure 13: Data Splitting and Decision Tree Classifier Model Training

```
# Make predictions on the test set
Y_prediction = clfr.predict(X_test)
```

```
# Evaluate the model
print("Accuracy:", accuracy_score(Y_test, Y_prediction))
print('Confusion Matrix:')
print(confusion_matrix(Y_test, Y_prediction))
print("Classification Report:")
print(classification_report(Y_test,Y_prediction))
```

```
Accuracy: 0.9763779527559056
Confusion Matrix:
[[ 86   1   1   2   0]
 [  0  11   1   0   0]
 [  3   0 246   0   0]
 [  1   0   0  12   0]
 [  0   0   0   0  17]]
```

Figure 14: Accuracy of Decision Tree Classifier

```
# Make predictions on the test set
Y_prediction = clfr.predict(X_test)
```

```
# Evaluate the model
print("Accuracy:", accuracy_score(Y_test, Y_prediction))
print('Confusion Matrix:')
print(confusion_matrix(Y_test, Y_prediction))
print("Classification Report:")
print(classification_report(Y_test,Y_prediction))
```

```
Accuracy: 0.6377952755905512
Confusion Matrix:
[[ 10   1  30  49   0]
 [  1   0   3   8   0]
 [  6   0 203  40   0]
 [  0   0   0  13   0]
 [  0   0   0   0  17]]
```

Figure 15: Accuracy of Naive Bayes Method

From the accuracy values of 97% for Decision Tree and 63% for Naïve Bayes, it is clear that the Decision Tree Classifier provides the most accurate model among the two and hence, the model chosen for evaluation is Decision Tree Classifier.

III. Model Evaluation

The Decision Tree model provided an accuracy of 97% which means 97 cars out of 100 cars categorised on the basis of quality were correct. For further evaluation, Confusion matrix and classification report were analysed for assessing the ability of the model to accurately rate the quality of the cars.

```
# Evaluate the model
print("Accuracy:", accuracy_score(Y_test, Y_prediction))
print('Confusion Matrix:')
print(confusion_matrix(Y_test, Y_prediction))
print("Classification Report:")
print(classification_report(Y_test,Y_prediction))
```

```
Accuracy: 0.9763779527559056
Confusion Matrix:
[[ 86   1   1   2   0]
 [  0  11   1   0   0]
 [  3   0 246   0   0]
 [  1   0   0  12   0]
 [  0   0   0   0  17]]
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        90
           1       0.92      0.92      0.92        12
           2       0.99      0.99      0.99       249
           3       0.86      0.92      0.89        13
           4       1.00      1.00      1.00        17

    accuracy                           0.98       381
   macro avg       0.94      0.96      0.95       381
weighted avg       0.98      0.98      0.98       381
```

Figure 16: Confusion matrix and Classification Report

- The overall high precision, recall and F1 scores indicates a balanced performance which implies that the model is reliable in predicting various classes for the cars with few false positives and negatives.
- The values for Class 2 ('good') indicates that the model works well for this class which is the majority class in the dataset, hence can be considered really good at identifying 'good' cars.
- The perfect score '1' for class 4(NaN) indicates that the model handles the incomplete data effectively without error.
- Although the model performs well, the slightly lower precision and F1 score for the Class 3('vgood') with other classes, which might be due to a smaller number of training data compared to that of others. This can be thus be improved by providing a significant number of instances to train the model.

Overall, the model is reliable for evaluating the cars into various classes with a very few instances of error.