```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import LabelEncoder


# Load the dataset
data = pd.read_csv('car_data.csv')
data.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

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
# Encode categorical features and target variable
le = LabelEncoder()
for col in data.columns:
    data[col] = le.fit_transform(data[col])


# Seperate features and targete
x= data.drop('class',axis=1)
y= data['class']


# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
len(X_train), len(X_test)
```

```
(1382, 346)
```

```
# Create and train Decision Tree Classifier
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

```
# Predict on Test Set
y_pred = clf.predict(X_test)


# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9739884393063584

Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.92      0.94        83
           1       0.62      0.91      0.74        11
           2       1.00      1.00      1.00       235
           3       1.00      0.94      0.97        17

    accuracy                           0.97       346
   macro avg       0.90      0.94      0.91       346
weighted avg       0.98      0.97      0.98       346
```

```
!pip install graphviz
```

```
Requirement already satisfied: graphviz in /usr/local/lib/python3.12/dist-packages (0.21)
```

```
from sklearn.tree import export_graphviz
import graphviz
```

```
# Export the decision tree to a DOT file
dot_data = export_graphviz(clf, out_file=None, feature_names=x.columns, class_names=[str(x) for x in le.classes_], filled=True, rounded=

# Visualize the tree
graph = graphviz.Source(dot_data)
graph.render("decision_tree", format="png", cleanup = True)
```

```
'decision_tree.png'
```

Ques 2

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
import matplotlib.pyplot as plt


# Load the dataset
url = "https://raw.githubusercontent.com/ageron/handson-ml2/master/datasets/housing/housing.csv"
data = pd.read_csv(url)


# Select important features and target variable
features = [
    'longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms',
    'population', 'households', 'median_income', 'ocean_proximity']
X = data[features]
y = data['median_house_value']



# Handle missing values by filling with median for numeric columns
X['total_bedrooms'] = X['total_bedrooms'].fillna(X['total_bedrooms'].median())
```

```
/tmp/ipython-input-1250734058.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  X['total_bedrooms'] = X['total_bedrooms'].fillna(X['total_bedrooms'].median())
```

```
# One-hot encode categorical feature 'ocean_proximity'
cat_features = ['ocean_proximity']
num_features = [f for f in features if f != 'ocean_proximity']

preprocessor = ColumnTransformer([
    ("onehot", OneHotEncoder(), cat_features)
], remainder='passthrough')

X_processed = preprocessor.fit_transform(X)


# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X_processed, y, test_size=0.2, random_state=42)


# Initialize and train Decision Tree Regressor
regressor = DecisionTreeRegressor(random_state=42, max_depth=4)  # limit depth for readability in plot
regressor.fit(X_train, y_train)

# Predict on test set
y_pred = regressor.predict(X_test)


# Evaluate model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared Score: {r2:.2f}")
```

```
Mean Squared Error: 5448146831.17
R-squared Score: 0.58
```

```
# Plot the tree
plt.figure(figsize=(20,10))
plot_tree(
```

```
    regressor,
    filled=True,
    rounded=True,
    feature_names=preprocessor.get_feature_names_out(),
    fontsize=10
)
plt.show()
```