NAME:BHARATH M H

USN :1SV21CS010

TEAM:05

```
 import numpy as np import
 pandas as pd
from sklearn.model_selection import train_test_split from
sklearn.neighbors import KNeighborsClassifier from
sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt

from google.colab import drive drive.mount('/content/drive')

data = {
     'mileage':[10,20,30,40,50,60,70,80,90,100],
     'engine':[1000,1200,1300,1400,1500,1600,1700,1800,1900,2000],
       'max_power': [100,200,300,400,500,600,700,800,900,1000],
       'quality': [0,0,0,0,0,1,1,1,1,1],
}

df = pd.DataFrame(data)

x = df[['mileage','engine','max_power']] y =
df[['quality']]

k = 4
knn = KNeighborsClassifier(n_neighbors=k) knn.fit(x,y)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/
_classification.py:215: 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 self._fit(X, y)
```

```
KNeighborsClassifier(n_neighbors=4)
```

```
new_data = np.array([[60,1700,700]]) prediction =
knn.predict(new_data)

if prediction[0] == 1:
  print("good")
elif prediction[0] == 0: print("bad")
```

```
good
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not
have valid feature names, but KNeighborsClassifier was fitted with feature names
  warnings.warn(
```

```
import pandas as pd
from sklearn.model_selection import train_test_split from
sklearn.preprocessing import StandardScaler
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

data = {
    'mileage':[10,20,30,40,50,60,70,80,90,100],
    'engine':[1000,1200,1300,1400,1500,1600,1700,1800,1900,2000],
      'max_power': [100,200,300,400,500,600,700,800,900,1000],
      'quality': [0,0,0,0,0,1,1,1,1,1],
}

df = pd.DataFrame(data)

x = df[['mileage','engine','max_power']]
y = df[['quality']]

X_train, X_test, y_train, y_test = train_test_split(x, y,
test_size=0.3,
random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

lr = LinearRegression()
lr.fit(X_train, y_train)

LinearRegression()

y_pred = lr.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')

Mean Squared Error: 0.11996825469502365
R-squared: 0.4601428538723936

print('Predictions for the test set:')
for i, (true, pred) in enumerate(zip(y_test, y_pred)):
    print(f'Sample {i}: True value = {true}, Predicted value =
{pred[0]:.2f}') # Extract the float value from the array

Predictions for the test set:
Sample 0: True value = quality, Predicted value = 1.03

import numpy as np # Import the NumPy library and give it the alias
'np'
from sklearn.neighbors import KNeighborsClassifier # Import the
KNeighborsClassifier
```

```
# Assuming you have your training data X_train and y_train ready
knn = KNeighborsClassifier(n_neighbors=5) # Create a KNN classifier
with 5 neighbors (adjust as needed)
knn.fit(X_train, y_train) # Train the KNN model on your training data

new_data = np.array([[60,1700,700]])
prediction = knn.predict(new_data)

if prediction[0] == 1:
  print("good")
elif prediction[0] == 0:
  print("bad")
```

```
good
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/
_classification.py:215: 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 self._fit(X, y)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

data = {
    'mileage':[10,20,30,40,50,60,70,80,90,100],
    'engine':[1000,1200,1300,1400,1500,1600,1700,1800,1900,2000],
      'max_power': [100,200,300,400,500,600,700,800,900,1000],
      'quality': [0,0,0,0,0,1,1,1,1,1],
}

df = pd.DataFrame(data)

x = df[['mileage','engine','max_power']]
y = df[['quality']]

X_train, X_test, y_train, y_test = train_test_split(x, y,
test_size=0.3,
random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

lr = LogisticRegression(multi_class='multinomial', solver='lbfgs',
max_iter=1000)
lr.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/
validation.py:1143: 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(max_iter=1000, multi_class='multinomial')
```

```
y_pred = lr.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

```
print(f'Accuracy: {accuracy}')
print('Confusion Matrix:')
print(conf_matrix)
print('Classification Report:')
print(class_report)
```

```
Accuracy: 0.6666666666666666
Confusion Matrix:
[[1 0]
 [1 1]]
Classification Report:
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       1.00      0.50      0.67         2

    accuracy                           0.67         3
   macro avg       0.75      0.75      0.67         3
weighted avg       0.83      0.67      0.67         3
```

```
print('Predictions for the test set:')
for i, (true, pred) in enumerate(zip(y_test, y_pred)):
    print(f'Sample {i}: True value = {true}, Predicted value =
{pred}')
```

```
Predictions for the test set:
Sample 0: True value = quality, Predicted value = 1
```

```
knn = KNeighborsClassifier(n_neighbors=5) # Create a KNN classifier
with 5 neighbors (adjust as needed)
knn.fit(X_train, y_train) # Train the KNN model on your training data
```

```
new_data = np.array([[60,1700,700]])
prediction = knn.predict(new_data)
```

```
if prediction[0] == 1:
  print("good")
```

```
elif prediction[0] == 0:
  print("bad")
```

```
good
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/
_classification.py:215: 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 self._fit(X, y)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

data = {
    'mileage':[10,20,30,40,50,60,70,80,90,100],
    'engine':[1000,1200,1300,1400,1500,1600,1700,1800,1900,2000],
      'max_power': [100,200,300,400,500,600,700,800,900,1000],
      'quality': [0,0,0,0,0,1,1,1,1,1],
}

df = pd.DataFrame(data)

x = df[['mileage','engine','max_power']]
y = df[['quality']]

X_train, X_test, y_train, y_test = train_test_split(x, y,
test_size=0.3)

random_state=42

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
```

```
DecisionTreeClassifier(random_state=42)
```

```
y_pred = dt.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print('Confusion Matrix:')
```

```
print(conf_matrix)
print('Classification Report:')
print(class_report)
```

```
Accuracy: 1.0
Confusion Matrix:
[[1 0]
 [0 2]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         2

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3
```

```
knn = KNeighborsClassifier(n_neighbors=5) # Create a KNN classifier
with 5 neighbors (adjust as needed)
knn.fit(X_train, y_train) # Train the KNN model on your training data

new_data = np.array([[60,1700,700]])
prediction = knn.predict(new_data)

if prediction[0] == 1:
  print("good")
elif prediction[0] == 0:
  print("bad")
```

```
good

/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/
_classification.py:215: 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 self._fit(X, y)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

data = {
    'mileage':[10,20,30,40,50,60,70,80,90,100],
    'engine':[1000,1200,1300,1400,1500,1600,1700,1800,1900,2000],
      'max_power': [100,200,300,400,500,600,700,800,900,1000],
      'quality': [0,0,0,0,0,1,1,1,1,1],
```

```
}

df = pd.DataFrame(data)

x = df[['mileage','engine','max_power']]
y = df[['quality']]

X_train, X_test, y_train, y_test = train_test_split(x, y,
test_size=0.3, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

rfc = RandomForestClassifier(n_estimators=100, random_state=42)
rfc.fit(X_train, y_train)
```

```
<ipython-input-55-e15c9536b5bf>:2: 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().
  rfc.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```

```
y_pred = rfc.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print('Confusion Matrix:')
print(conf_matrix)
print('Classification Report:')
print(class_report)
```

```
Accuracy: 0.6666666666666666
Confusion Matrix:
[[1 0]
 [1 1]]
Classification Report:
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       1.00      0.50      0.67         2

    accuracy                           0.67         3
   macro avg       0.75      0.75      0.67         3
weighted avg       0.83      0.67      0.67         3
```

```
print('Predictions for the test set:')
```

```
Predictions for the test set:

for i, (true, pred) in enumerate(zip(y_test, y_pred)):
    print(f'Sample {i}: True value = {true}, Predicted value =
{pred}')

Sample 0: True value = quality, Predicted value = 1

knn = KNeighborsClassifier(n_neighbors=5) # Create a KNN classifier
with 5 neighbors (adjust as needed)
knn.fit(X_train, y_train) # Train the KNN model on your training data

new_data = np.array([[60,1700,700]])
prediction = knn.predict(new_data)

if prediction[0] == 1:
  print("good")
elif prediction[0] == 0:
  print("bad")

good

/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/
_classification.py:215: 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 self._fit(X, y)
```