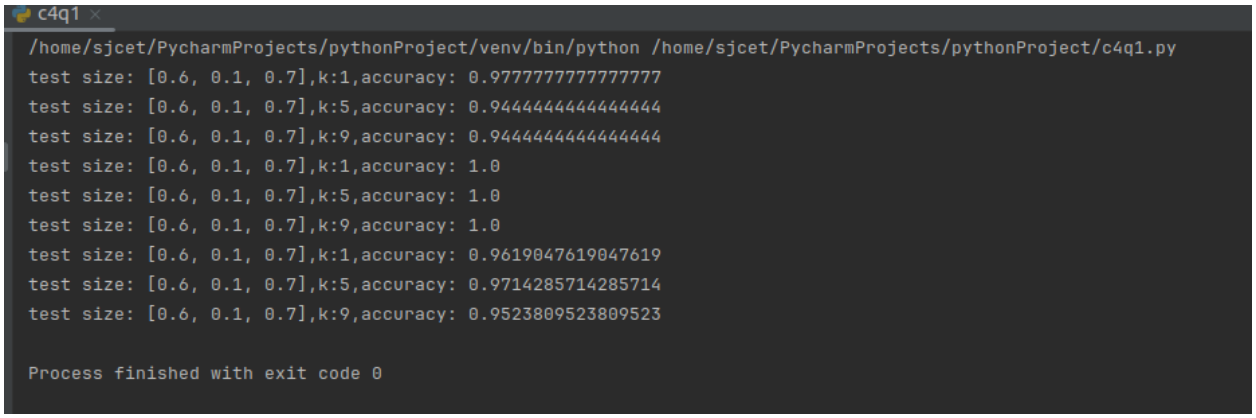
1. Using the iris data set implement the KNN algorithm. Take different values for Test and training data set .Also use different values for k. Also find the accuracy level.

Code

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
iris_data = pd.read_csv("iris.csv")
X = iris_data.iloc[:, :-1].values
y = iris_data.iloc[:, -1].values

test_sizes = [0.6, 0.1, 0.7]
k_values = [1,5,9]
for test_size in test_sizes:
    for k in k_values:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size,
random_state=42)
        knn = KNeighborsClassifier(n_neighbors=k)
        knn.fit(X_train, y_train)
        y_pred = knn.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        print(f'test size: {test_sizes},k:{k},accuracy: {accuracy}')
```

Output

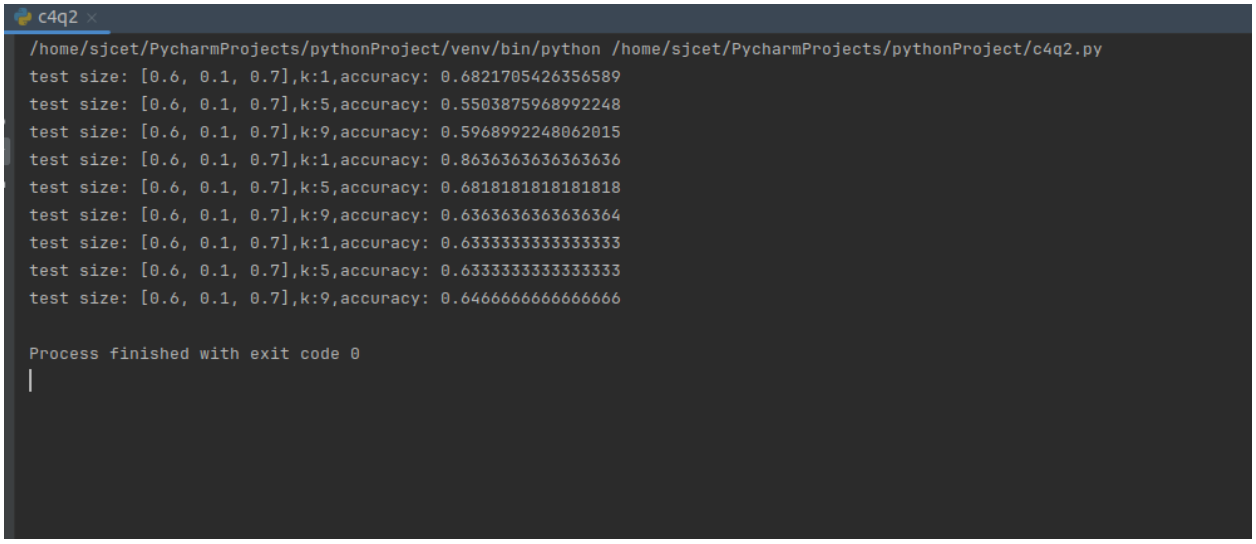


2. Download another data set suitable for the KNN and implement the KNN algorithm. Take different values for Test and training data set .Also use different values for k.

Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
data = pd.read_csv("glass.csv")
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

test_sizes = [0.6, 0.1, 0.7]
k_values = [1,5,9]
for test_size in test_sizes:
    for k in k_values:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size,
random_state=42)
        knn = KNeighborsClassifier(n_neighbors=k)
        knn.fit(X_train, y_train)
        y_pred = knn.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        print(f'test size: {test_sizes},k:{k},accuracy: {accuracy}')
```

Output



3. Using iris data set, implement naive bayes classification for different naive Bayes classification algorithms.((i) gaussian (ii) bernoulli etc)

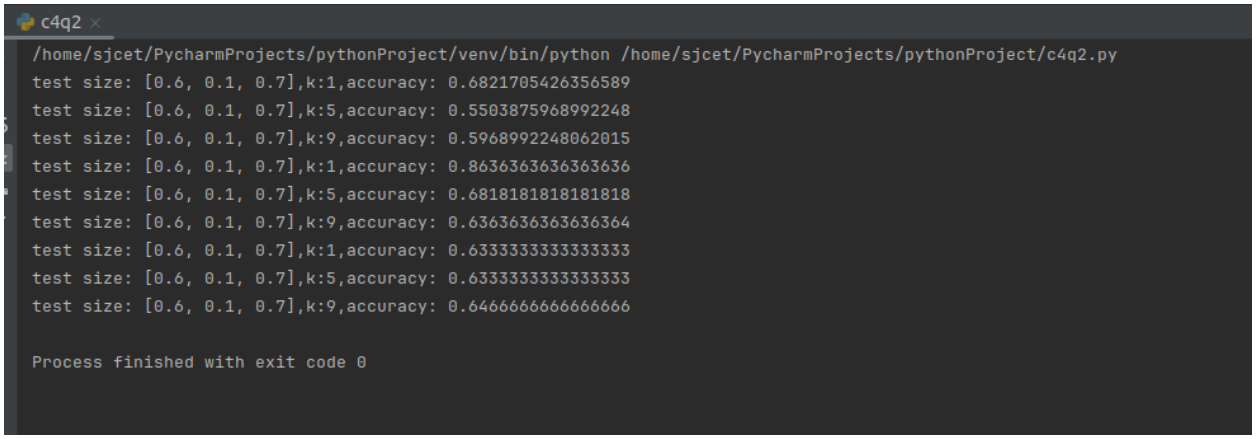
- Find out the accuracy level w.r.t to each algorithm
- Display the no:of mislabeled classification from test data set
- List out the class labels of the mismatching records

Code

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('iris.csv')
X = dataset.iloc[:,:4].values
y = dataset['variety'].values
dataset.head (5)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.5)
from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB ()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print(y_pred)

from sklearn.metrics import confusion_matrix
cm =confusion_matrix(y_test, y_pred)
print(cm)

from sklearn.metrics import accuracy_score
print ("Accuracy: ", accuracy_score (y_test, y_pred))
df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
print(df)
print()
from sklearn.naive_bayes import BernoulliNB
classif = BernoulliNB ()
classif.fit(X_train, y_train)
y_pred = classif.predict(X_test)
print(y_pred)

from sklearn.metrics import confusion_matrix
cmx =confusion_matrix(y_test, y_pred)
```

```
print(cmx)

from sklearn.metrics import accuracy_score
print ("Accuracy: ", accuracy_score (y_test, y_pred))
fd = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
print(fd)
```

Output



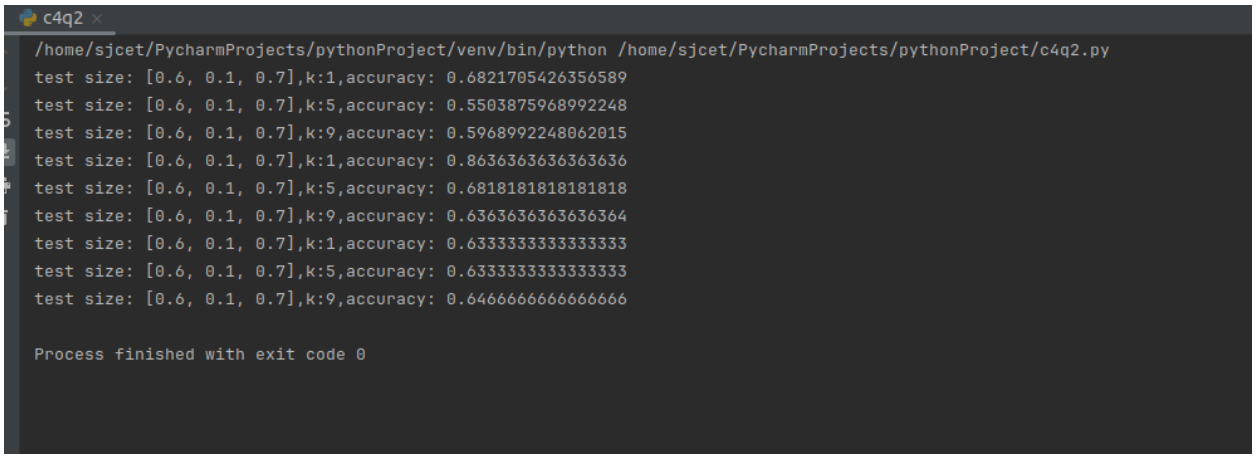
4. Use car details CSV file and implement decision tree algorithm
   - Find out the accuracy level.
   - Display the no: of mislabelled classification from test data set
   - List out the class labels of the mismatching records

Code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

data = pd.read_csv('car.csv')
print(data.head())
col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety',
'class']
data.columns = col_names
print(col_names)

data['class'],class_names = pd.factorize(data['class'])
data['buying'],_ = pd.factorize(data['buying'])
data['maint'],_ = pd.factorize(data['maint'])
```

```
data['doors'],_ = pd.factorize(data['doors'])
data['persons'],_ = pd.factorize(data['persons'])
data['lug_boot'],_ = pd.factorize(data['lug_boot'])
data['safety'],_ = pd.factorize(data['safety'])

print(data.head())
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

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

tree1 = DecisionTreeClassifier()

tree1.fit(X_train, y_train)
y_pred = tree1.predict(X_test)
# how did our model perform?
count_misclassified = (y_test != y_pred).sum()
print('Misclassified samples count:', count_misclassified)

accuracy = accuracy_score (y_test, y_pred)
print("Accuracy:", accuracy)
```

Output

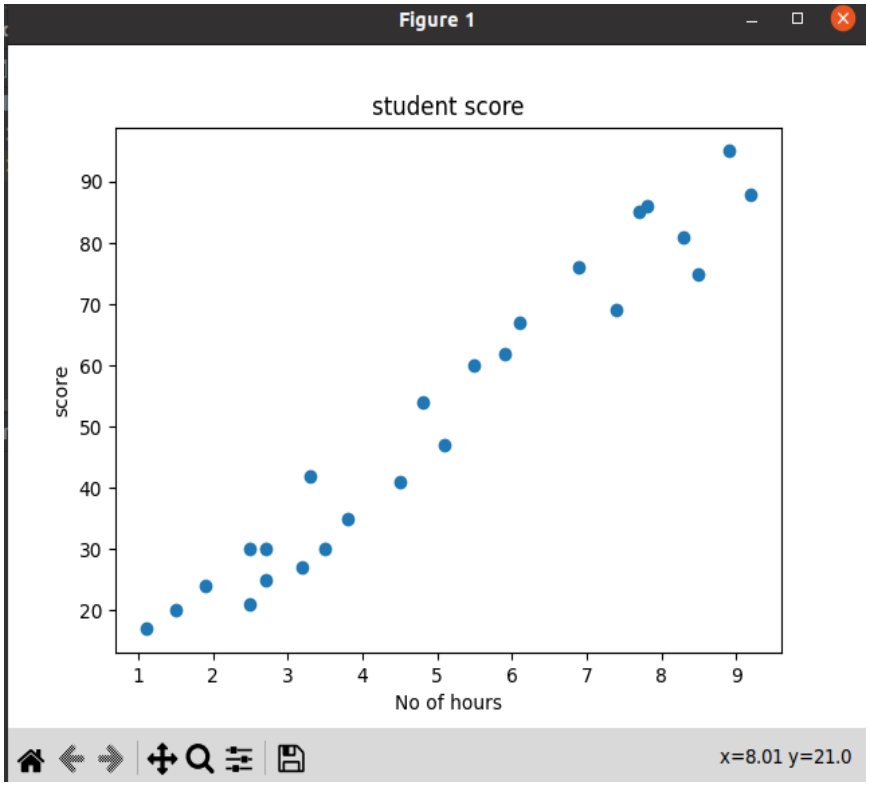


5. Implement Simple and multiple linear regression for the data sets ‘student_score.csv’ and ‘company_data .csv’ respectively

Code

```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score
```

```
student = pd.read_csv('student_scores.csv')
print(student.head())
student.describe()
student.info()
import matplotlib.pyplot as plt
Xax=student.iloc[:, 0]
yax=student.iloc[:, 1]
plt.scatter(Xax,yax)
plt.xlabel("No of hours")
plt.ylabel("score")
plt.title("student score")
plt.show()
X = student.iloc[:, :-1]
y = student.iloc[:, 1]
print('X values:')
print(X)
print(' y values :')
print(y)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3)
print(X_train)
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
print('INTERCEPT=', regressor.intercept_)
print('CO_EFFICIENT=', regressor.coef_)
y_pred = regressor.predict(X_test)
for(i,j) in zip(y_test, y_pred):
    if(i!=j):
        print("Actual value :", i, "predicted value :",j)
        print("number of misslabelled points from test data set  :", (y_test != y_pred).sum())
        from sklearn import metrics
        print("mean absolute error :", metrics.mean_absolute_error(y_test, y_pred))
        print("mean squared error :", metrics.mean_squared_error(y_test, y_pred))
        print("root mean squared error :", np.sqrt(metrics.mean_squared_error(y_test,
y_pred))
```

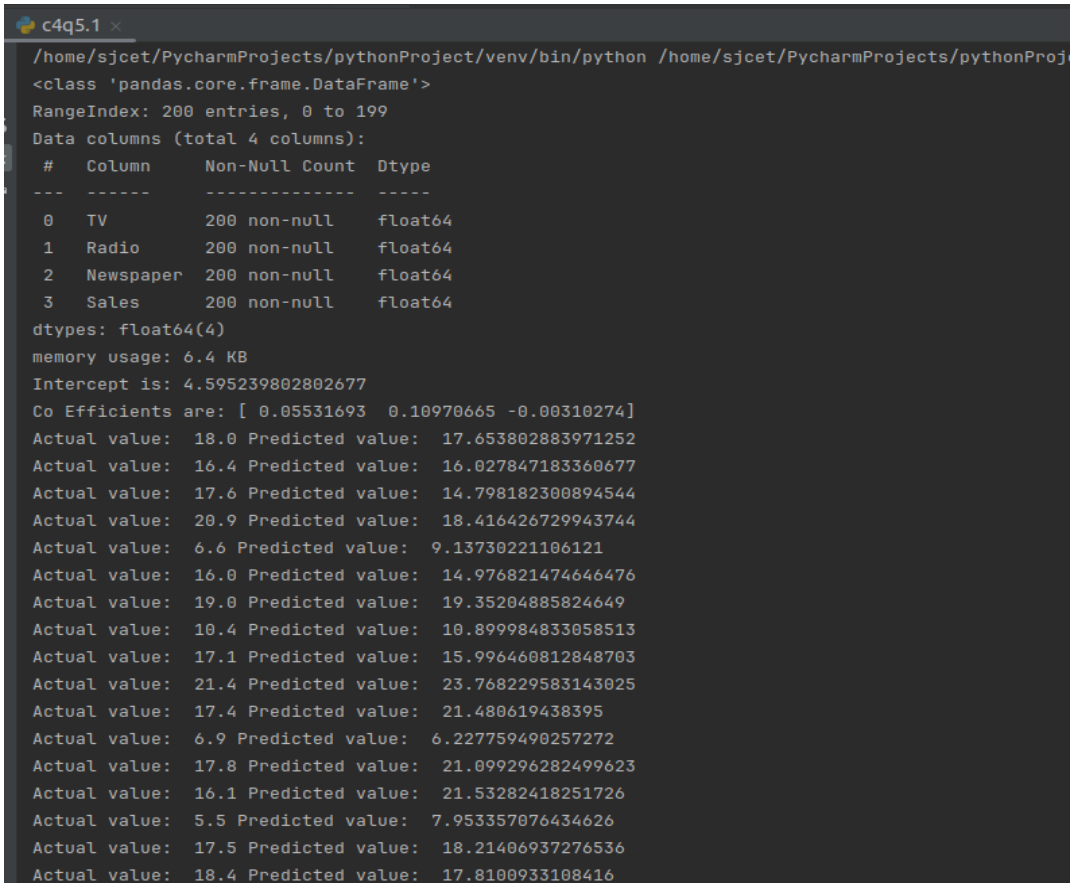


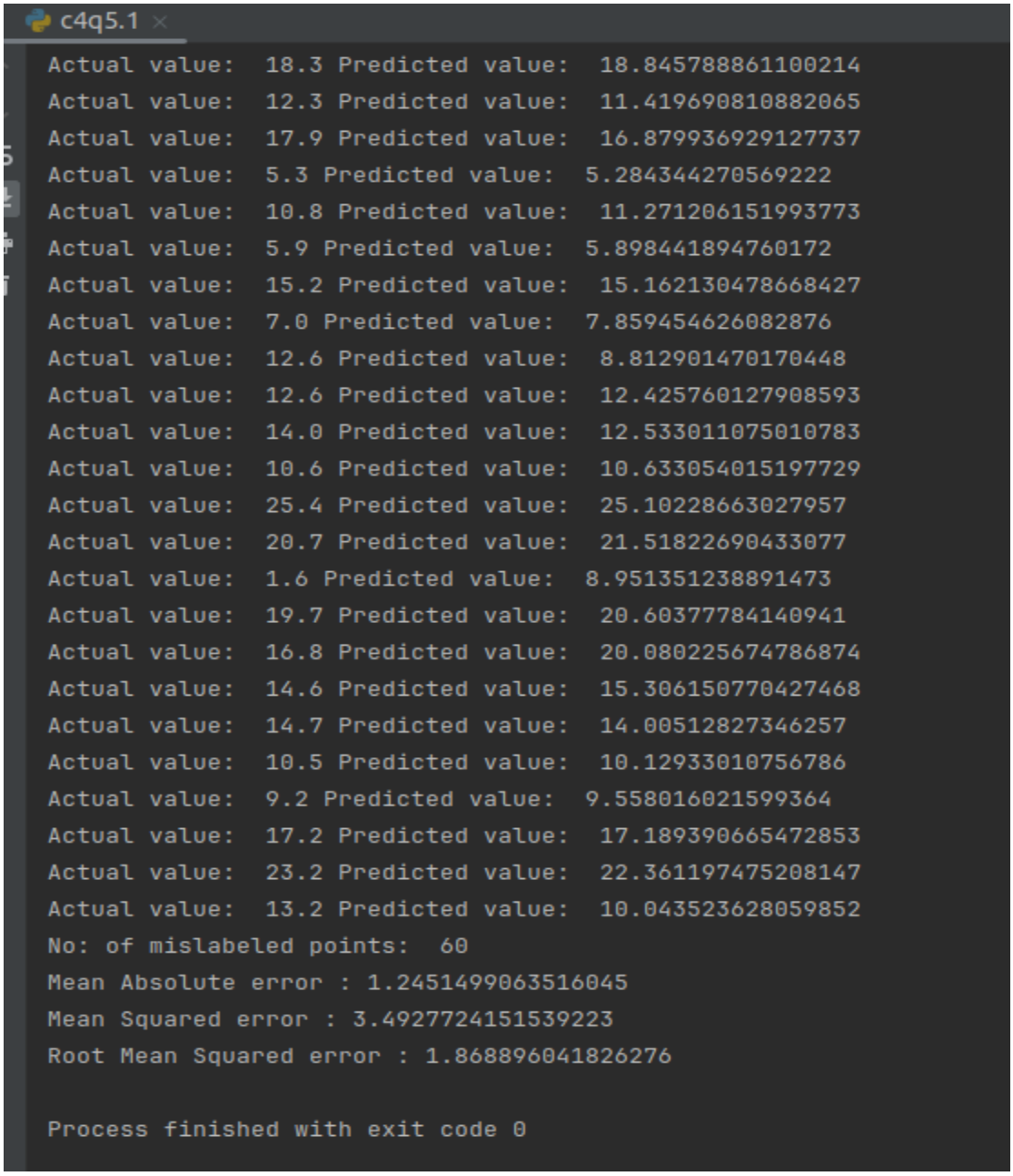
Multiple Regression

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics

advertising = pd.read_csv('Company_data.csv')
advertising.head()
advertising.describe()
advertising.info()

X = advertising.iloc[:, :-1]
y = advertising.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
reg = LinearRegression()
reg.fit(X_train, y_train)
print("Name: Aswathy Chandran")
```

```
print("Reg No: SJC22MCA-2016")
print("Batch: 22-24")
print()
print('Intercept is:', reg.intercept_)
print('Co Efficients are:', reg.coef_)
y_pred = reg.predict(X_test)

for(i,j) in zip(y_test, y_pred):
    if(i!=j):
        print('Actual value: ', i, 'Predicted value: ', j)
print('No: of mislabeled points: ', (y_test != y_pred).sum())

print("Mean Absolute error :", metrics.mean_absolute_error(y_test,y_pred))
print("Mean Squared error :", metrics.mean_squared_error(y_test,y_pred))
print("Root Mean Squared error :",
np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

Output




c4q5.1
Actual value: 18.3 Predicted value: 18.845788861100214
Actual value: 12.3 Predicted value: 11.419690810882065
Actual value: 17.9 Predicted value: 16.879936929127737
Actual value: 5.3 Predicted value: 5.284344270569222
Actual value: 10.8 Predicted value: 11.271206151993773
Actual value: 5.9 Predicted value: 5.898441894760172
Actual value: 15.2 Predicted value: 15.162130478668427
Actual value: 7.0 Predicted value: 7.859454626082876
Actual value: 12.6 Predicted value: 8.812901470170448
Actual value: 12.6 Predicted value: 12.425760127908593
Actual value: 14.0 Predicted value: 12.533011075010783
Actual value: 10.6 Predicted value: 10.633054015197729
Actual value: 25.4 Predicted value: 25.10228663027957
Actual value: 20.7 Predicted value: 21.51822690433077
Actual value: 1.6 Predicted value: 8.951351238891473
Actual value: 19.7 Predicted value: 20.60377784140941
Actual value: 16.8 Predicted value: 20.080225674786874
Actual value: 14.6 Predicted value: 15.306150770427468
Actual value: 14.7 Predicted value: 14.00512827346257
Actual value: 10.5 Predicted value: 10.12933010756786
Actual value: 9.2 Predicted value: 9.558016021599364
Actual value: 17.2 Predicted value: 17.189390665472853
Actual value: 23.2 Predicted value: 22.361197475208147
Actual value: 13.2 Predicted value: 10.043523628059852
No: of mislabeled points: 60
Mean Absolute error : 1.2451499063516045
Mean Squared error : 3.4927724151539223
Root Mean Squared error : 1.868896041826276
Process finished with exit code 0