Question:

1. Implement Linear Regression and k-Nearest Neighbor Classifier without using Scikit-learn.

<u>Solution to Linear Regression</u>

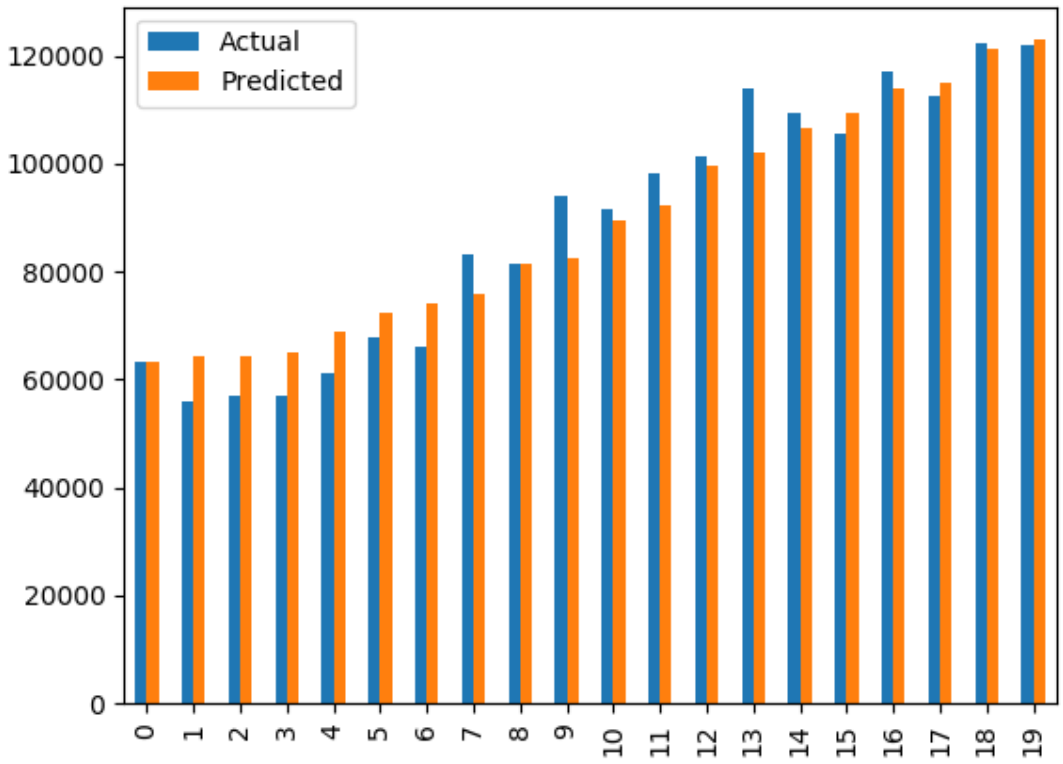
The demonstrated Python code:

```
# Import the necessary libraries
import matplotlib.pyplot as plot
import pandas
import numpy as np

def getAlpha():
    global xTrain,yTrain,xMean,yMean

    n = 0
    for i in range(len(xTrain)):
        n = n + (xTrain[i] - xMean) * (yTrain[i] - yMean)

    dn = 0
    for i in range(len(xTrain)):
        dn = dn + pow((xTrain[i] - xMean),2)

    return n/dn

def meanAbsoluteError():
    global yTest, pred

    n = 0
    for i in range(len(yTest)):
        nom = n + abs(yTest[i] - pred[i])

    return n/len(yTest)

def meanSquaredError():
    global yTest, pred

    n = 0
    for i in range(len(yTest)):
        n = n + pow((yTest[i] - pred[i]),2)

    return n/len(yTest)


# Import the dataset
```

```
dataset = pandas.read_csv('salaryData.csv')

x = dataset['YearsExperience'].values
y = dataset['Salary'].values

# Reshape x y
X = x.reshape(len(x),1)
Y = y.reshape(len(y),1)

# Spliting dataset
xTrain, yTrain, xTest, yTest, pred = ([] for i in range(5))
for i in range(int(len(X)*1/3)):
    xTrain.append(X[i])
    yTrain.append(Y[i])

for i in range(int(len(X)*1/3), len(X)):
    xTest.append(X[i])
    yTest.append(Y[i])

# Calculation of mean values and alpha, beta
xMean = np.mean(xTrain)
yMean = np.mean(yTrain)

alpha = getAlpha()
beta = yMean - alpha*xMean

# Prediction on Training data
for i in range(len(xTest)):
    pred.append( alpha* xTest[i] + beta)

print(np.asarray(pred).shape)
df = pandas.DataFrame({'Actual': np.asarray(yTest).flatten(), 'Predicted': np.asarray(pred).flatten()})
print(df)
df1 = df
df1.plot(kind='bar')
plot.show()

print('Mean Absolute Error:', meanAbsoluteError())
print('Mean Squared Error:', meanSquaredError())
print('Root Mean Squared Error:', np.sqrt(meanSquaredError()))
```

A sample of input and output is as below:

```
"E:\4.1\ai lab\term project 3\ass5\venv\Scripts\python.
(20, 1)
      Actual      Predicted
0    63218.0   63397.347042
1    55794.0   64299.410602
2    56957.0   64299.410602
3    57081.0   65201.474162
4    61111.0   68809.728402
5    67938.0   72417.982641
6    66029.0   74222.109761
7    83088.0   76026.236881
8    81363.0   81438.618240
9    93940.0   82340.681800
10   91738.0   89557.190279
11   98273.0   92263.380959
12  101302.0   99479.889438
13  113812.0  102186.080118
14  109431.0  106696.397917
15  105582.0  109402.588597
16  116969.0  113912.906396
17  112635.0  114814.969956
18  122391.0  121129.414875
19  121872.0  122933.541995
Mean Absolute Error: 0.0
Mean Squared Error: [37067616.12767456]
Root Mean Squared Error: [6088.31800481]

Process finished with exit code 0
```



```
"E:\4.1\ai lab\term project 3\ass5\venv\Scripts\python.exe" "E:/4.1/ai lab/term project
3/ass5/regressionAssignment.py"
(20, 1)
     Actual      Predicted
0    63218.0   63397.347042
1    55794.0   64299.410602
2    56957.0   64299.410602
3    57081.0   65201.474162
4    61111.0   68809.728402
5    67938.0   72417.982641
6    66029.0   74222.109761
7    83088.0   76026.236881
8    81363.0   81438.618240
9    93940.0   82340.681800
10   91738.0   89557.190279
11   98273.0   92263.380959
12  101302.0   99479.889438
13  113812.0  102186.080118
14  109431.0  106696.397917
15  105582.0  109402.588597
16  116969.0  113912.906396
17  112635.0  114814.969956
18  122391.0  121129.414875
19  121872.0  122933.541995
Mean Absolute Error: 0.0
Mean Squared Error: [37067616.12767456]
Root Mean Squared Error: [6088.31800481]
```

<u>Solution to K nearest neighbor</u>

The demonstrated Python code:

```
# Import the necessary libraries
import matplotlib.pyplot as plot
import pandas
import numpy as np

def getAlpha():
    global xTrain,yTrain,xMean,yMean

    n = 0
    for i in range(len(xTrain)):
        n = n + (xTrain[i] - xMean) * (yTrain[i] - yMean)

    dn = 0
```

```
    for i in range(len(xTrain)):
        dn = dn + pow((xTrain[i] - xMean),2)

    return n/dn

def meanAbsoluteError():
    global yTest, pred

    n = 0
    for i in range(len(yTest)):
        nom = n + abs(yTest[i] - pred[i])

    return n/len(yTest)

def meanSquaredError():
    global yTest, pred

    n = 0
    for i in range(len(yTest)):
        n = n + pow((yTest[i] - pred[i]),2)

    return n/len(yTest)


# Import the dataset
dataset = pandas.read_csv('salaryData.csv')


# Differentiate attribute and target columns
x = dataset['YearsExperience'].values
y = dataset['Salary'].values

# Reshaping
X = x.reshape(len(x),1)
Y = y.reshape(len(y),1)

# Spliting dataset into test and training data
xTrain, yTrain, xTest, yTest, pred = ([] for i in range(5))
for i in range(int(len(X)*1/3)):
    xTrain.append(X[i])
    yTrain.append(Y[i])

for i in range(int(len(X)*1/3), len(X)):
    xTest.append(X[i])
    yTest.append(Y[i])
```

```
# Calculating the mean values and alpha, beta
xMean = np.mean(xTrain)
yMean = np.mean(yTrain)

alpha = getAlpha()
beta = yMean - alpha*xMean

# Prediction on Training data
for i in range(len(xTest)):
    pred.append( alpha* xTest[i] + beta)

print(np.asarray(pred).shape)
df = pandas.DataFrame({'Actual': np.asarray(yTest).flatten(), 'Predicted': np.asarray(pred).flatten()})
print(df)
df1 = df
df1.plot(kind='bar')
plot.show()

print('Mean Absolute Error:', meanAbsoluteError())
print('Mean Squared Error:', meanSquaredError())
print('Root Mean Squared Error:', np.sqrt(meanSquaredError()))
```

A sample of input and output is as below:

```
"E:\4.1\ai lab\term project 3\ass5\venv\Scripts\python.exe" "E:/4.1/ai lab/term project 3/ass5/knn.py"
Accuracy for  1  nearest neighbour is  0.92
Accuracy for  2  nearest neighbour is  0.9
Accuracy for  3  nearest neighbour is  0.92
Accuracy for  4  nearest neighbour is  0.92
Accuracy for  5  nearest neighbour is  0.94

Process finished with exit code 0
```

"E:\4.1\ai lab\term project 3\ass5\venv\Scripts\python.exe" "E:/4.1/ai lab/term project 3/ass5/knn.py"
Accuracy for 1 nearest neighbour is 0.92
Accuracy for 2 nearest neighbour is 0.9
Accuracy for 3 nearest neighbour is 0.92
Accuracy for 4 nearest neighbour is 0.92
Accuracy for 5 nearest neighbour is 0.94