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

def calculate_accuracy(y_test, y_pred):
    correct = (y_pred == y_test).sum()
    total_instances = len(y_test)
    accuracy = correct / total_instances
    return correct, total_instances, accuracy

def euclidean_distance(x1, x2):
    distance = np.sqrt(np.sum((x1 - x2) ** 2))
    return distance

def count_zeros_ones(k_nearest, distances):
    count_0 = 0
    count_1 = 0
    for element in k_nearest:
        if element == 0:
            count_0 += 1
        elif element == 1:
            count_1 += 1

    if count_0 > count_1:
        return 0
    elif count_0 < count_1:
        return 1
    else:
        distances_0 = 0
        distances_1 = 0
        for i in range(len(k_nearest)):
            if k_nearest[i] == 0:
                distances_0 += 1/distances[i]
            elif k_nearest[i] == 1:
                distances_1 += 1/distances[i]
        if distances_0 > distances_1:
            return 0
        else:
            return 1


class KNN:
    def _init_(self):
        self.k = 1

    def setK(self, k):
        self.k = k

    def fit(self, X, y):
```

```
        self.X_train = X
        self.y_train = y

    def predict(self, X_test):
        predictions = [self._predict(X_test[i]) for i in
range(X_test.shape[0])]
        return predictions

    def _predict(self, x_test):
        distances = [euclidean_distance(x_test, x_train) for x_train
in self.X_train]
        # sort and return nums of rows before sort
        rows_num = np.argsort(distances)[:self.k]
        k_nearest = [self.y_train[i][0] for i in rows_num]
        distances = sorted(distances)
        predict_value = count_zeros_ones(k_nearest, distances)
        return predict_value



path = 'diabetes.csv'
diabetes = pd.read_csv(path)
print('diabetes')
print(diabetes)
```

```
diabetes
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
\
0              6      148             72             35        0  33.6

1              1       85             66             29        0  26.6

2              8      183             64              0        0  23.3

3              1       89             66             23       94  28.1

4              0      137             40             35      168  43.1

..           ...      ...            ...            ...      ...   ...

762            9       89             62              0        0  22.5

763           10      101             76             48      180  32.9

764            2      122             70             27        0  36.8

765            5      121             72             23      112  26.2

766            1      126             60              0        0  30.1
```

```
     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
762                     0.142   33        0
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1

[767 rows x 9 columns]
```

```
# the features and targets are separated
num_of_cols = diabetes.shape[1]
X = diabetes.iloc[:, 0:num_of_cols-1]
y = diabetes.iloc[:, num_of_cols-1:num_of_cols]
print("features")
print(X)
print("---------------------------------------------")
print("targets")
print(y)
```

```
features
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
\
0              6      148             72             35        0  33.6

1              1       85             66             29        0  26.6

2              8      183             64              0        0  23.3

3              1       89             66             23       94  28.1

4              0      137             40             35      168  43.1

..           ...      ...            ...            ...      ...   ...

762            9       89             62              0        0  22.5

763           10      101             76             48      180  32.9

764            2      122             70             27        0  36.8

765            5      121             72             23      112  26.2

766            1      126             60              0        0  30.1
```

```
     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
762                     0.142   33
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47

[767 rows x 8 columns]
-----------------------------------------------
targets
     Outcome
0          1
1          0
2          1
3          0
4          1
..       ...
762        0
763        0
764        0
765        0
766        1

[767 rows x 1 columns]
```

```python
# divide data into 70% for training and 30% for testing
X_train, X_test, y_train, y_test = train_test_split(X, y,
shuffle=True, test_size=0.30, random_state=45)
print("X_train")
print(X_train)
print("-----------------------------------------------")
print("X_test")
print(X_test)
print("-----------------------------------------------")
print("y_train")
print(y_train)
print("-----------------------------------------------")
print("y_test")
print(y_test)
```

```
X_train
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
\
172            2       87              0             23        0  28.9
```

```
405            2      123             48             32       165  42.1
357           13      129              0             30         0  39.9
483            0       84             82             31       125  38.2
737            8       65             72             23         0  32.0
..           ...      ...            ...            ...       ...   ...
725            4      112             78             40         0  39.4
607            1       92             62             25        41  19.5
544            1       88             78             29        76  32.0
643            4       90              0              0         0  28.0
414            0      138             60             35       167  34.6

     DiabetesPedigreeFunction  Age
172                     0.773   25
405                     0.520   26
357                     0.569   44
483                     0.233   23
737                     0.600   42
..                        ...  ...
725                     0.236   38
607                     0.482   25
544                     0.365   29
643                     0.610   31
414                     0.534   21

[536 rows x 8 columns]
------------------------------------------------
X_test
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
\
195            5      158             84             41      210  39.4
51             1      101             50             15       36  24.2
66             0      109             88             30        0  32.5
437            5      147             75              0        0  29.9
664            6      115             60             39        0  33.7
..           ...      ...            ...            ...       ...   ...
```

```
38              2       90              68              42         0  38.2

354             3       90              78               0         0  42.7

18              1      103              30              38        83  43.3

229             0      117              80              31        53  45.2

540             8      100              74              40       215  39.4


     DiabetesPedigreeFunction  Age
195                     0.395   29
51                      0.526   26
66                      0.855   38
437                     0.434   28
664                     0.245   40
..                        ...  ...
38                      0.503   27
354                     0.559   21
18                      0.183   33
229                     0.089   24
540                     0.661   43

[231 rows x 8 columns]
-----------------------------------------------
y_train
     Outcome
172        0
405        0
357        1
483        0
737        0
..       ...
725        0
607        0
544        0
643        0
414        1

[536 rows x 1 columns]
-----------------------------------------------
y_test
     Outcome
195        1
51         0
66         1
437        0
664        1
```

```
..       ...
38         1
354        0
18         0
229        0
540        1

[231 rows x 1 columns]
```

```
# Normalize each feature column separately for training and test
objects using Min-Max Scaling.
X_train = (X_train-X_train.min()) / (X_train.max()-X_train.min())

X_test = (X_test-X_test.min()) / (X_test.max()-X_test.min())

print("X_train after standardization")
print(X_train)
print("-----------------------------------------------")
print("X_test after standardization")
print(X_test)
```

```
X_train after standardization
     Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin
BMI  \
172     0.117647  0.437186       0.000000       0.232323  0.000000
0.430700
405     0.117647  0.618090       0.393443       0.323232  0.195035
0.627422
357     0.764706  0.648241       0.000000       0.303030  0.000000
0.594635
483     0.000000  0.422111       0.672131       0.313131  0.147754
0.569300
737     0.470588  0.326633       0.590164       0.232323  0.000000
0.476900
..           ...       ...            ...            ...       ...
...
725     0.235294  0.562814       0.639344       0.404040  0.000000
0.587183
607     0.058824  0.462312       0.508197       0.252525  0.048463
0.290611
544     0.058824  0.442211       0.639344       0.292929  0.089835
0.476900
643     0.235294  0.452261       0.000000       0.000000  0.000000
0.417288
414     0.000000  0.693467       0.491803       0.353535  0.197400
0.515648

     DiabetesPedigreeFunction       Age
172                  0.296755  0.078431
405                  0.188728  0.098039
```

```
357                   0.209650  0.450980
483                   0.066183  0.039216
737                   0.222886  0.411765
..                         ...       ...
725                   0.067464  0.333333
607                   0.172502  0.078431
544                   0.122545  0.156863
643                   0.227156  0.196078
414                   0.194705  0.000000

[536 rows x 8 columns]
-----------------------------------------
X_test after standardization
     Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin
BMI  \
195     0.357143  0.802030       0.736842       0.788462  0.282258
0.687609
51      0.071429  0.512690       0.438596       0.288462  0.048387
0.422339
66      0.000000  0.553299       0.771930       0.576923  0.000000
0.567190
437     0.357143  0.746193       0.657895       0.000000  0.000000
0.521815
664     0.428571  0.583756       0.526316       0.750000  0.000000
0.588133
..           ...       ...            ...            ...       ...
...
38      0.142857  0.456853       0.596491       0.807692  0.000000
0.666667
354     0.214286  0.456853       0.684211       0.000000  0.000000
0.745201
18      0.071429  0.522843       0.263158       0.730769  0.111559
0.755672
229     0.000000  0.593909       0.701754       0.596154  0.071237
0.788831
540     0.571429  0.507614       0.649123       0.769231  0.288978
0.687609

     DiabetesPedigreeFunction       Age
195                   0.138530  0.133333
51                    0.196882  0.083333
66                    0.343430  0.283333
437                   0.155902  0.116667
664                   0.071715  0.316667
..                         ...       ...
38                    0.186637  0.100000
354                   0.211581  0.000000
18                    0.044098  0.200000
229                   0.002227  0.050000
```

```
540                  0.257016  0.366667

[231 rows x 8 columns]
```

```
# Convert X to numpy array
X_train = X_train.to_numpy().reshape((-1, num_of_cols - 1))
X_test = X_test.to_numpy().reshape((-1, num_of_cols - 1))

# Convert y to numpy array
y_train = y_train.to_numpy()
y_test = y_test.to_numpy()

knn = KNN()
knn.fit(X_train, y_train)
avg = 0
iterations = 8
for k in range(2, iterations+2):
    print('k value:', k)
    knn.setK(k)
    y_pred = knn.predict(X_test)
    y_pred = np.array(y_pred).reshape(len(y_pred), 1)
    correct, total_instances, accuracy = calculate_accuracy(y_test,
y_pred)
    avg += accuracy
    print('Number of correctly classified instances:', correct)
    print('Total number of instances:', total_instances)
    print('Accuracy:', accuracy)
    print()
```

```
k value: 2
Number of correctly classified instances: 155
Total number of instances: 231
Accuracy: 0.670995670995671

k value: 3
Number of correctly classified instances: 167
Total number of instances: 231
Accuracy: 0.7229437229437229

k value: 4
Number of correctly classified instances: 162
Total number of instances: 231
Accuracy: 0.7012987012987013

k value: 5
Number of correctly classified instances: 160
Total number of instances: 231
Accuracy: 0.6926406926406926

k value: 6
Number of correctly classified instances: 160
```

```
Total number of instances: 231
Accuracy: 0.6926406926406926

k value: 7
Number of correctly classified instances: 163
Total number of instances: 231
Accuracy: 0.7056277056277056

k value: 8
Number of correctly classified instances: 165
Total number of instances: 231
Accuracy: 0.7142857142857143

k value: 9
Number of correctly classified instances: 166
Total number of instances: 231
Accuracy: 0.7186147186147186


avg = avg/iterations
print('The average accuracy across all iterations:', avg)

The average accuracy across all iterations: 0.7023809523809524
```