Q1. Write a Python code to implement the KNN classifier algorithm on load_iris dataset in sklearn.datasets.

In [2]:
```
import seaborn as sns
df = sns.load_dataset('iris')
```

In [3]:
```
df
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [5]:
```
x = df.drop('species',axis = 1)
y = df['species']
```

In [6]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 24 ,te
```

In [8]:
```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3,algorithm='auto')
neigh.fit(x, y)
```

Out[8]:
```
KNeighborsClassifier(n_neighbors=3)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [10]:
```
y_pred = neigh.predict(x_test)
y_pred
```

Out[10]:
```
array(['virginica', 'virginica', 'versicolor', 'versicolor', 'virginica',
       'setosa', 'setosa', 'virginica', 'setosa', 'virginica', 'setosa',
       'setosa', 'setosa', 'virginica', 'virginica', 'setosa',
       'virginica', 'setosa', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'versicolor', 'virginica', 'setosa',
       'virginica', 'virginica', 'setosa', 'virginica'], dtype=object)
```

Q2. Write a Python code to implement the KNN regressor algorithm on load_boston dataset in sklearn.datasets.

In [18]:
```
import pandas as pd

df = pd.read_csv('Boston.csv')
```

In [19]:
```
df
```

Out[19]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	3
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	3
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	3
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	3
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	3
...	...	...	...	...	...	...	...	...	...	...	...	...	
501	502	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	3
502	503	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	3
503	504	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	3
504	505	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	3
505	506	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	3

506 rows × 15 columns

In [26]:
```
x = df.drop('medv',axis = 1)
y = df['medv']
```

In [27]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 24 ,te
```

In [31]:
```
from sklearn.neighbors import KNeighborsRegressor
regressor = KNeighborsRegressor(n_neighbors=5,algorithm='auto')
regressor.fit(x_train,y_train)
regressor.predict(x_test)
```

Out[31]:
```
array([39.16, 19.84, 37.28, 17.9 , 22.9 , 33.14, 24.78, 16.98, 14.86,
       26.36, 23.5 , 14.88, 28.7 , 21.46, 35.22, 19.96, 31.  , 38.64,
       33.82, 26.24, 17.24, 21.  , 15.68, 18.28, 20.62, 35.2 , 19.96,
       33.08, 13.76, 20.92, 28.8 , 22.54, 41.38, 13.38, 17.  , 18.28,
       15.3 , 17.04, 36.28, 19.7 , 21.52, 28.8 , 11.24, 17.68, 27.6 ,
       15.4 , 19.4 , 19.7 , 21.3 , 17.68, 22.64, 33.72, 41.38, 16.62,
       23.16, 20.26, 19.36, 19.48, 22.34, 17.18, 13.82, 10.28, 28.02,
       14.86, 15.74,  7.64, 23.54, 13.82, 15.98,  9.76, 35.34, 21.2 ,
       20.54, 20.08, 10.2 , 34.24, 35.48, 21.54, 24.9 , 24.12, 13.3 ,
       19.68, 13.82, 20.  , 19.84, 21.46,  7.88, 10.36, 12.6 , 20.72,
       13.84, 23.52, 19.66, 20.36, 38.7 , 30.8 , 19.7 , 21.18, 24.84,
       35.34, 11.14, 28.76])
```

Q3. Write a Python code snippet to find the optimal value of K for the KNN classifier algorithm using cross-validation on load_iris dataset in sklearn.datasets

In [33]:
```
from sklearn.model_selection import GridSearchCV
params = {'n_neighbors' :[1,2,3,4,5,6,7,8,9,10]}
classifier = GridSearchCV(KNeighborsRegressor(),param_grid = params,cv = 5
classifier.fit(x_train,y_train)
classifier.best_params_
```

Out[33]:
```
{'n_neighbors': 3}
```

Q4. Implement the KNN regressor algorithm with feature scaling on load_boston dataset in sklearn.datasets.

In [34]:
```
import pandas as pd

df = pd.read_csv('Boston.csv')

x = df.drop('medv',axis = 1)
y = df['medv']

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 24 ,te
```

In [37]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(x_train)
X_test = scaler.transform(x_test)
X_train
```

Out[37]:
```
array([[ 1.43038514,  0.20157003, -0.49782577, ...,  0.83441922,
        -3.75056866,  0.88804601],
       [-0.93457371, -0.43986066, -0.49782577, ..., -0.2672323 ,
        -0.183528  ,  0.38429857],
       [ 1.57732045,  1.66507225, -0.49782577, ...,  0.83441922,
        -0.06558684,  1.70455972],
       ...,
       [-0.41680165, -0.44607109,  1.37597805, ..., -1.46068812,
         0.38019813, -1.34560336],
       [ 0.94759768,  2.73552972, -0.49782577, ...,  0.83441922,
         0.4498758 ,  2.68437618],
       [ 1.16450219,  9.93490283, -0.49782577, ...,  0.83441922,
        -3.68567385,  1.11085738]])
```

In [38]:
```
from sklearn.neighbors import KNeighborsRegressor
regressor = KNeighborsRegressor(n_neighbors=5,algorithm='auto')
regressor.fit(X_train,y_train)
regressor.predict(X_test)
```

Out[38]:
```
array([29.62, 19.84, 23.38, 19.04, 23.8 , 33.14, 23.78, 16.  , 13.54,
       30.84, 22.36, 17.68, 19.88, 24.04, 22.92, 21.42, 23.86, 33.48,
       34.08, 28.26, 18.18, 20.96,  9.32, 19.04, 20.04, 26.82, 26.72,
       23.96, 13.12, 21.58, 26.04, 21.8 , 29.24, 12.5 , 16.04, 21.06,
       12.52, 19.1 , 41.26, 19.7 , 20.86, 30.34, 11.52, 17.68, 27.94,
        7.88, 18.94, 21.1 , 21.66, 18.32, 23.34, 33.92, 43.84, 15.88,
       23.9 , 23.78, 19.36, 17.32, 22.04, 11.04, 14.54, 13.84, 31.26,
       14.56, 15.62,  8.28, 23.74, 14.1 , 15.18,  9.1 , 25.64, 20.08,
       20.16, 19.52,  8.28, 24.04, 29.06, 20.36, 24.38, 21.4 ,  8.28,
       19.56, 14.54, 21.28, 19.84, 20.64,  8.28, 10.36, 11.44, 19.3 ,
       15.66, 23.24, 19.66, 21.48, 31.62, 45.16, 19.7 , 21.24, 22.68,
       41.26, 12.36, 24.22])
```

Q5. Write a Python code snippet to implement the KNN classifier algorithm with weighted voting on load_iris dataset in sklearn.datasets

In [41]:
```
import seaborn as sns
df = sns.load_dataset('iris')
x = df.drop('species',axis = 1)
y = df['species']
```

In [42]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 24 ,te
```

In [43]:
```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3,weights='distance')
neigh.fit(x, y)
```

Out[43]:
```
KNeighborsClassifier(n_neighbors=3, weights='distance')
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [44]:
```
y_pred = neigh.predict(x_test)
y_pred
```

Out[44]:
```
array(['virginica', 'virginica', 'versicolor', 'versicolor', 'virginica',
       'setosa', 'setosa', 'versicolor', 'setosa', 'virginica', 'setosa',
       'setosa', 'setosa', 'virginica', 'virginica', 'setosa',
       'virginica', 'setosa', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'versicolor', 'virginica', 'setosa',
       'virginica', 'virginica', 'setosa', 'virginica'], dtype=object)
```

Q6. Implement a function to standardise the features before applying KNN classifier.

In [45]:
```
import seaborn as sns
df = sns.load_dataset('iris')
x = df.drop('species',axis = 1)
y = df['species']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 24 ,te
```

In [47]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(x_train)
X_test = scaler.transform(x_test)
```

In [50]:
```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3,weights='distance')
neigh.fit(X_train, y_train)
```

Out[50]:
```
KNeighborsClassifier(n_neighbors=3, weights='distance')
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [51]:
```
y_pred = neigh.predict(X_test)
y_pred
```

Out[51]:
```
array(['virginica', 'virginica', 'versicolor', 'versicolor', 'virginica',
       'setosa', 'setosa', 'virginica', 'setosa', 'virginica', 'setosa',
       'setosa', 'setosa', 'virginica', 'virginica', 'setosa',
       'virginica', 'setosa', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'versicolor', 'virginica', 'setosa',
       'virginica', 'virginica', 'setosa', 'virginica'], dtype=object)
```

Q7. Write a Python function to calculate the euclidean distance between two points.

In [4]:
```
import numpy as np

P1 = np.array((1, 1))
P2 = np.array((4, 3))

temp = P1 - P2

euclid_dist = np.sqrt(np.dot(temp.T, temp))

print(euclid_dist)
```

```
3.605551275463989
```

Q8. Write a Python function to calculate the manhattan distance between two points.

In [2]:
```
def get_manhattan_distance(p, q):

    distance = 0
    for p_i,q_i in zip(p,q):
        distance += abs(p_i - q_i)

    return distance

a = (1, 1)
b = (4, 3)

d = get_manhattan_distance(a, b)

print(d)
```

```
5
```

In [ ]:
```

```