Experiment - 3

Implement k - nearest neighbor's classification using python.

Source code:

```
# Import necessary modules
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# Loading data
  irisData = load_iris()

# create feature and target arrays
  X = irisData.data
  y = irisData.target

# split into training and test set
  X_train, X_test, y_train, y_test = train_test_
          split(X, y, test_size=0.2, random_state=42)

  knn = KNeighborsClassifier(n_neighbors=7)
  knn.fit(X_train, y_train)
# predict on dataset which model has not
seen before
  print(knn.predict(X_test))
```

Output:

[1 0 2 1 1 0 1 2 2 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]

Example of Calculating Euclidean distance

```
from math import sqrt
# calculate the Euclidean distance between
two vectors
def euclidean_distance(row1, row2):
    distance = 0.0
    for i in range(len(row1)-1):
        distance += (row1[i]-row2[i])**2
    return sqrt(distance)

# Test distance function
dataset =
[[2.7810836, 2.550537003, 0],
 [1.465489372, 2.362125076, 0],
 [3.396561688, 4.400293529, 0],
 [1.38807019, 1.850220317, 0],
 [3.06407232, 3.005305973, 0],
 [7.627531214, 2.088626775, 1],
 [5.332441248, 2.088626775, 1],
 [6.922596716, 1.77106367, 1],
 [8.675418651, -0.242068655, 1],
 [7.673756466, 3.508563011, 1]]

row0 = dataset[0]
for row in dataset: distance = euclidean_
                        distance(row0, row)

Print(distance)
```

Output:

0.0

1.3290173915275787

1.9494646655653247

1.5591439385540549

0.5356280721938492

4.850940186986411

2.592833759950511

4.214227042632867

6.5224099882228337

4.98558538244497 95

Knn example:

Implement knn algorithm and plot the result using python.

Source code:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import
~~Listed Co~~
ListedColormap
from sklearn import neighbors, datasets
n-neighbors =15
#import some data to play with
iris = datasets.load_iris()
# we only take the first two features. We could avoid this ugly
# slicing by using a two-dim dataset
X = iris.data [:, :2]
y = iris.target
h = 0.02 # step size in the mesh
# create color maps
cmap_light = ListedColormap(["orange","cyan",
                 "cornflowerblue"])

cmap_bold = ["darkorange","c","darkblue"]
for weights in ["uniform", "distance"]:
```

we create an instance of Neighbours classifier and fit the data.

```
clf = neighbors.KNeighborsClassifier(n_neighbors,
                                     weights= weights)

clf.fit(X,y)
```

plot decision boundary. for that, we will assign a color to each.

point in the mesh [x_min, x_max]x[y_min, y_max].

```
x_min, x_max = X[:,0].min() - 1, X[:,0].
                                    max() + 1

y_min, y_max = X[:,1].min() - 1, X[:,1].max()+1

xx,yy = np.meshgrid(np.arrange(x_min, x_max,
            h), np.arrange(y_min, y_max,
                                    h))

Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
```

put the results into a color plot

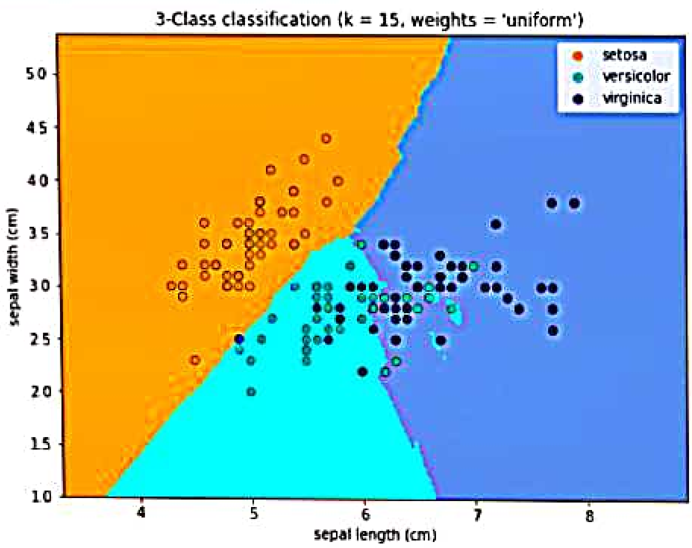
```
Z = Z.reshape(xx.shape)
plt.figure(figsize = (8,6))
plt.contourf(xx, yy, Z, cmap = cmap_
                              light)
```

```
# plot also the training points
sns.scatterplot(
    x= X[:,0],
    y= X[:,1],
    hue= iris.target_names[y],
    palette= cmap_bold,
    alpha =1.0,
    edgecolor = "black",
)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title(
    "3-class classification (k=%i,weights='%s')"
        % (n_neighbors, weights)
    )
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.show()
```

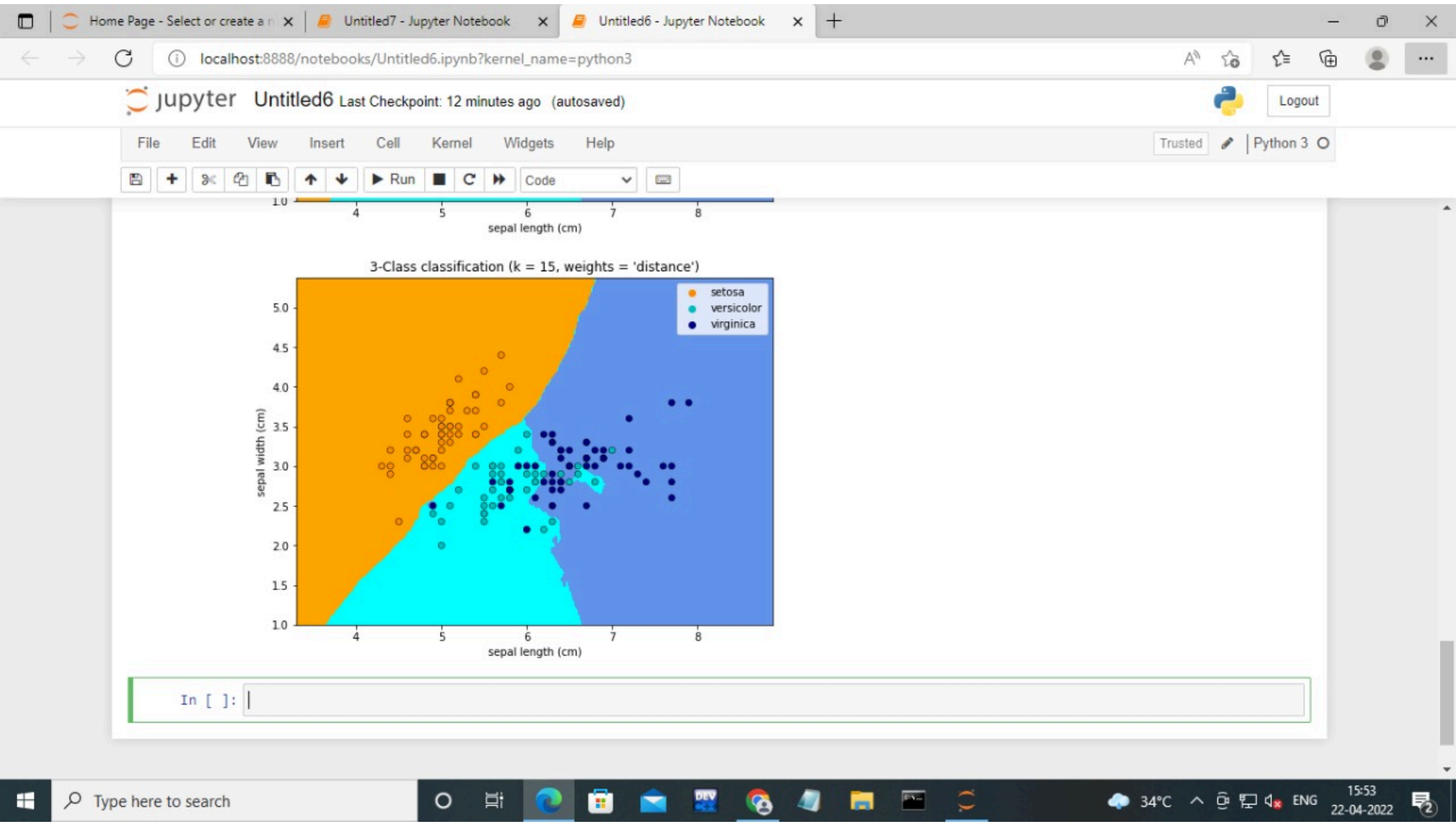
Run Code

```
    plt.ylim(yy.min(), yy.max())
    plt.title(
        "3-Class classification (k = %i, weights = '%s')" % (n_neighbors, weights)
    )
    plt.xlabel(iris.feature_names[0])
    plt.ylabel(iris.feature_names[1])

plt.show()
```



pe here to search


Home Page - Select or create a n
Untitled7 - Jupyter Notebook
Untitled6 - Jupyter Notebook
localhost:8888/notebooks/Untitled6.ipynb?kernel_name=python3
jupyter Untitled6 Last Checkpoint: 12 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
sepal length (cm)
3-Class classification (k = 15, weights = 'distance')
setosa
versicolor
virginica
sepal width (cm)
sepal length (cm)
In [ ]:
Type here to search
34°C
ENG
15:53
22-04-2022