Amir Hossein Eyvazkhani
1747696
Ex07

Task 1)

ⓐ Hamming Distance: $d(x, x') = |x \setminus x' \cup x' \setminus x|$

x_n	y_n	$H_D(\{a,d,b\})$	$H_D(\{f\})$
$\{a,b,d,e\}$	3	$\|\{e\}\| = 1$	$\|\{f,a,b,d,e\}\| = 5$
$\{c,d\}$	1	$\|\{a,b,c\}\| = 3$	$\|\{c,d,f\}\| = 3$
$\{a\}$	2	$\|\{d,b\}\| = 2$	$\|\{a,f\}\| = 2$
$\{a,f,g\}$	3	$\|\{d,b,f,g\}\| = 4$	$\|\{a,g\}\| = 2$
$\{e,f\}$	4	$\|\{e,f,a,d,b\}\| = 5$	$\|\{e\}\| = 1$

Sorted Nearest Neighbors:

$\{3, 2, 1, 3, 4\}$ $\qquad$ $\{4, 3, 2, 1, 3\}$

k=1 : $\bar{y} = 3$ ✗. $\qquad$ $\bar{y} = 4$ ✗.

k=2 : $\bar{y} = \frac{1}{2}(3+2) = 2.5$ $\qquad$ $\bar{y} = \frac{1}{2}(4+3) = 3.5$ ✗.

k=3 : $\bar{y} = \frac{1}{3}(3+2+1) = 2$ ✓ $\qquad$ $\bar{y} = \frac{1}{3}(4+3+2) = 3$ ✓

$\Rightarrow$ k=3 was obtained

CamScanner

ⓑ Jaccard Similarity: $S(x, x') = \frac{|x \cap x'|}{|x \cup x'|}$

x_n	y_n	$J_D(\{a,b,d\})$	$J_D(\{f\})$
$\{a,b,d,e\}$	3	$\frac{3}{4}$	$\frac{0}{5}$
$\{c,d\}$	1	$\frac{1}{4}$	$\frac{0}{5}$
$\{a\}$	2	$\frac{1}{3}$	$\frac{0}{2}$
$\{a,c,f,g\}$	3	$\frac{1}{5}$	$\frac{1}{3}$
$\{e,f\}$	4	$\frac{0}{5}$	$\frac{1}{2}$

Sorted Nearest Neighbours:

$\{3,2,1,3,4\}$ $\quad$ $\{3,4,2,1,3\}$

$k=1 \rightarrow \bar{y} = 3$ (X) $\quad$ $\bar{y} = 3$ (X)

$k=2 \rightarrow \bar{y} = \frac{1}{2}(3+2) = 2.5$ (✓) $\quad$ $\bar{y} = \frac{1}{2}(3+4) = 3.5$ (✓)

$\Rightarrow$ $k=2$ was obtained

Task 3) The algorithm:

i) First we initialise $D(0,j)=j$ and $D(i,0)=i$

ii) Recursively for $D(i,j)$: $\begin{cases} D(i-1,j-1) & \text{if } x_i = y_j \\ 1+\min\begin{cases} D(i-1,j) \\ D(i,j-1) \\ D(i-1,j-1) \end{cases} & \text{if } x_i \neq y_i \end{cases}$

we do two inner loops $i=1$ to L, $j=1$ to k

iii) For the answer we select the buttom right element

CamScanner

		e	x	e	c	u	t	i	o	n
	0	1	2	3	4	5	6	7	8	9
i	1	1	2	3	4	5	6	6	7	8
n	2	2	2	3	4	5	6	7	7	7
t	3	3	3	3	4	5	$\boxed{5}$	6	7	8
e	4	3	4	3	4	5	6	6	7	8
n	5	4	4	4	4	5	6	7	7	7
t	6	5	5	$\boxed{5}$	5	5	5	6	7	8
i	7	6	6	6	6	6	6	5	6	7
o	8	7	7	7	7	7	7	6	5	6
n	9	8	8	8	8	8	8	7	6	$\boxed{5}$

The final distance is $\boxed{5}$

The distance between "intent" and "exe" is $\boxed{5}$

The distance between "execut" and "int" is $\boxed{5}$

Task 2 The code given below:

In [ ]:

```python
# We import necessary libraries
# Please do not use scikit-learn or any other package. Implement K-NN classification yourself.
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.colors import ListedColormap
```

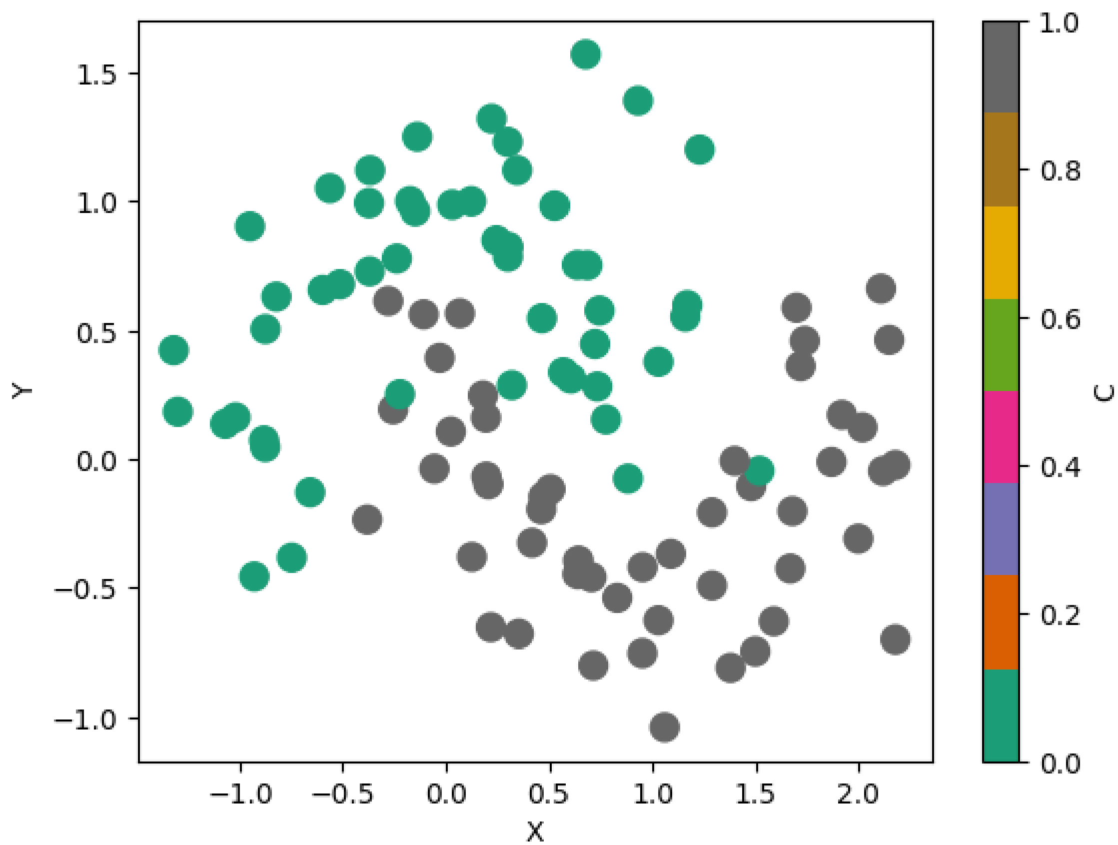
In [ ]:

```python
# Here we read the provided ushape.csv file
# We have retained only a small number of rows to ensure computational easiness and clear visualization
df = pd.read_csv('ushape.csv',names=['X','Y', 'C'], header=0, index_col=None)
```

In [ ]:

```python
# Let us see how the data looks like
df.plot.scatter('X','Y',c='C', s=100, colormap='Dark2')
```

Out[ ]: <Axes: xlabel='X', ylabel='Y'>



In [ ]:
```
# This function implements the L2 distance between two sets of points
def l2(x1,y1, x2, y2):
    distance = np.sqrt((x1-x2)**2 + (y1-y2)**2)
    return distance
```

In [ ]:
```
# The following function computes the distances between a test point and all the training points
def distance(x_test, y_test):
    distances = list()
    for index, row in df.iterrows():
        d = l2(row['X'], row['Y'], x_test, y_test)
        distances.append(d)
    return np.array(distances)
```

In [ ]:

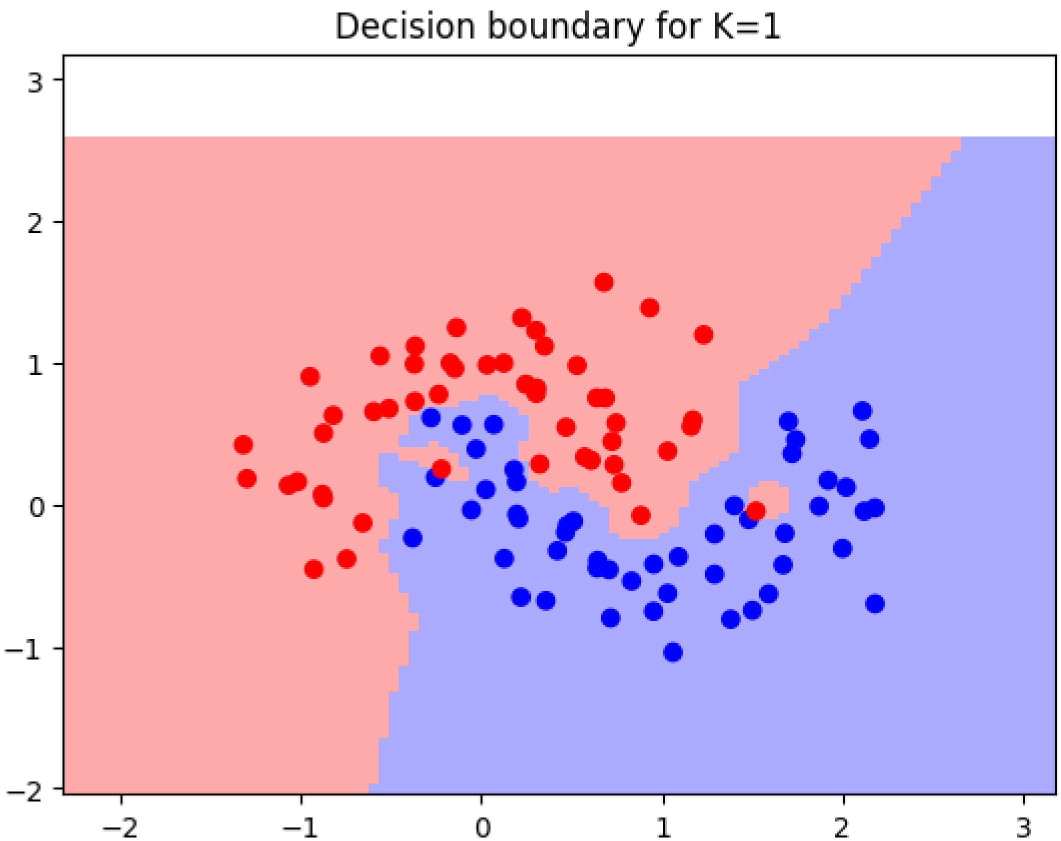
```python
# You need to complete the following function.
# The following function should assign the class to a point (x_test, y_test) using K-NN classification
def knn_classification(x_test, y_test, k):
    dist=distance(x_test,y_test)
    sorted_indices = np.argsort(dist)
    k_nearest_labels = df['C'][sorted_indices[:k]]
    unique_labels, counts = np.unique(k_nearest_labels, return_counts=True)
    return unique_labels[np.argmax(counts)]
```

In [ ]:

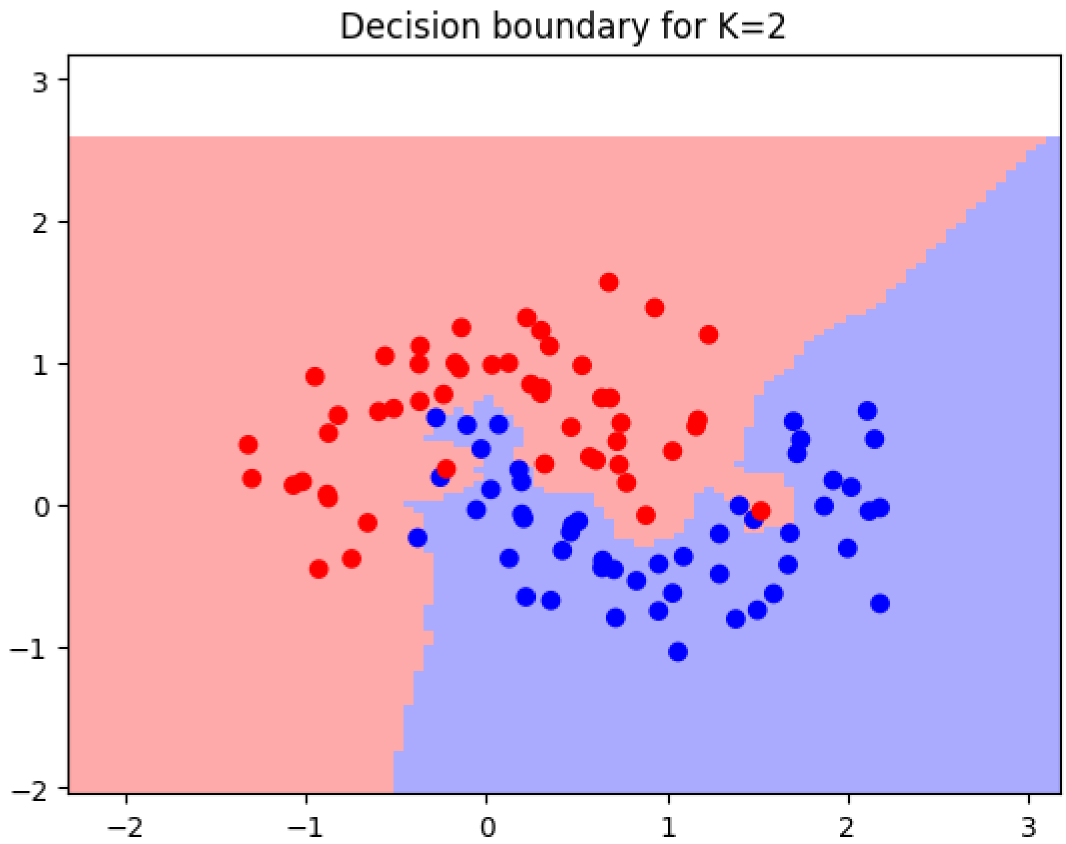
```python
# You need to complete the following function.
# The following function should plot the decision surface for the two classes given the value of K.
# You need to test all the points between df.X.min() and df.X.max() and also df.Y.min() and df.Y.max().
def plot_decision_surface(k):
   light=ListedColormap(['#FFAAAA', '#AAAAFF'])
   bold=ListedColormap(['#FF0000','#0000FF'])
   x=df[['X','Y']].to_numpy()
   xmin,xmax=x[:, 0].min()-1,x[:, 0].max()+1
   ymin,ymax=x[:, 1].min()-1,x[:, 1].max()+1
   XX,YY=np.meshgrid(np.linspace(xmin,xmax,100),np.linspace(ymin,ymax,100))
   ZZ=[]
   for xx,yy in zip(XX.ravel(),YY.ravel()):
      ZZ.append(knn_classification(xx,yy,k))
   ZZ=np.array(ZZ).reshape(XX.shape)
   plt.pcolormesh(XX, YY, ZZ,cmap=light,shading='auto')
   plt.scatter(x[:,0], x[:,1],c=df['C'].to_numpy(),cmap=bold)
   plt.xlim(XX.min(), XX.max())
   plt.ylim(YY.min(), XX.max())
   plt.title('Decision boundary for K=%d'%k)
   plt.show()
```

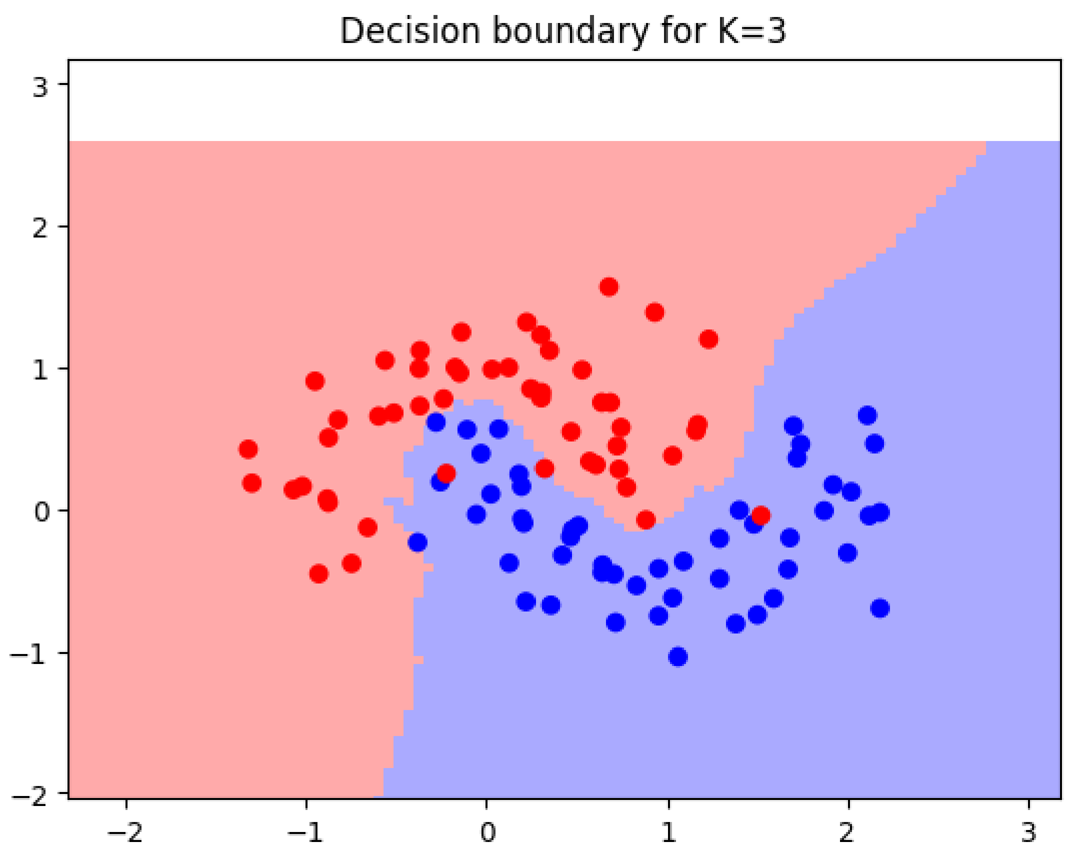
In [ ]:

```python
K=[1,2,3]
for k in range(1, 4):
    plot_decision_surface(k)
```



Decision boundary for K=1
3
2
1
0
−1
−2
−2
−1
0
1
2
3


Decision boundary for K=2
3
2
1
0
−1
−2
−2
−1
0
1
2
3



With smaller values of k, the decision boundary tends to be more sensitive to noise or outliers in the data. A single outlier can have a significant impact on the classification of a point. Smaller values of k result in more complex decision boundaries that follow the fluctuations in the training data more closely. This can lead to overfitting, especially if the dataset has noise.

s k increases, the decision boundary becomes smoother and less sensitive to local variations in the data. The model becomes more robust to noise and outliers. However, if the value of k is too large (which we do not have here in our example; just worth mentioning), the model might underfit the data, meaning it may fail to capture the underlying patterns and relationships in the dataset.