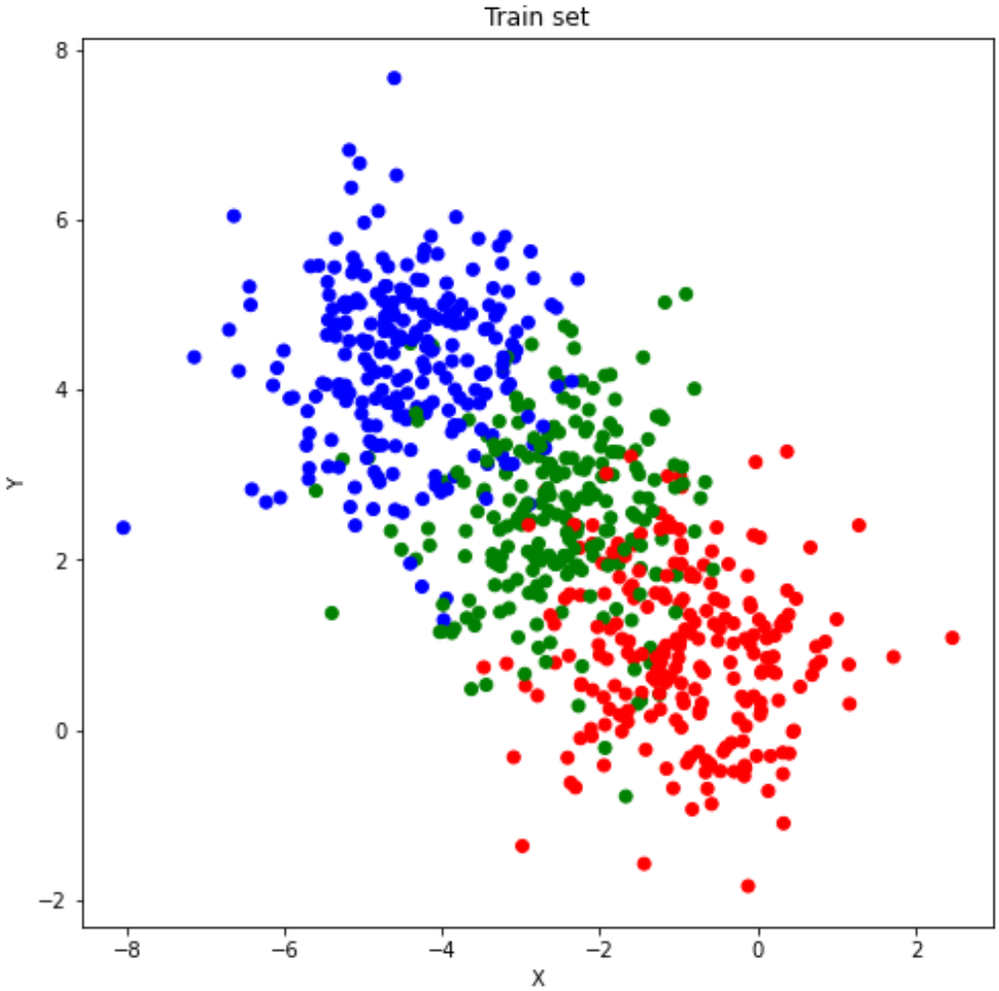
```
import matplotlib.pyplot as plt
import numpy as np
from numpy import random as rnd
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

```
# init

rnd.seed(2022)

colors = np.array(['red', 'green', 'blue'])

mu1 = (-1, 1)
mu2 = (-2.5, 2.5)
mu3 = (-4.5, 4.5)

sigma = np.eye(2)
```

```
# step 1

def get_train(N=700):
  trainX, trainY = list(), list()

  for i in range(N):
    temp = rnd.uniform(0, 1)

    if temp <= 1/3:
      trainX.append(rnd.multivariate_normal(mu1, sigma))
      trainY.append(0)
    elif temp <= 2/3:
      trainX.append(rnd.multivariate_normal(mu2, sigma))
      trainY.append(1)
    else:
      trainX.append(rnd.multivariate_normal(mu3, sigma))
      trainY.append(2)

  return np.array(trainX), np.array(trainY)

trainX, trainY = get_train()
```

```
# step 2

plt.figure(figsize=(8, 8))

plt.scatter(trainX[:, 0], trainX[:, 1], c=colors[trainY])
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Train set')
```

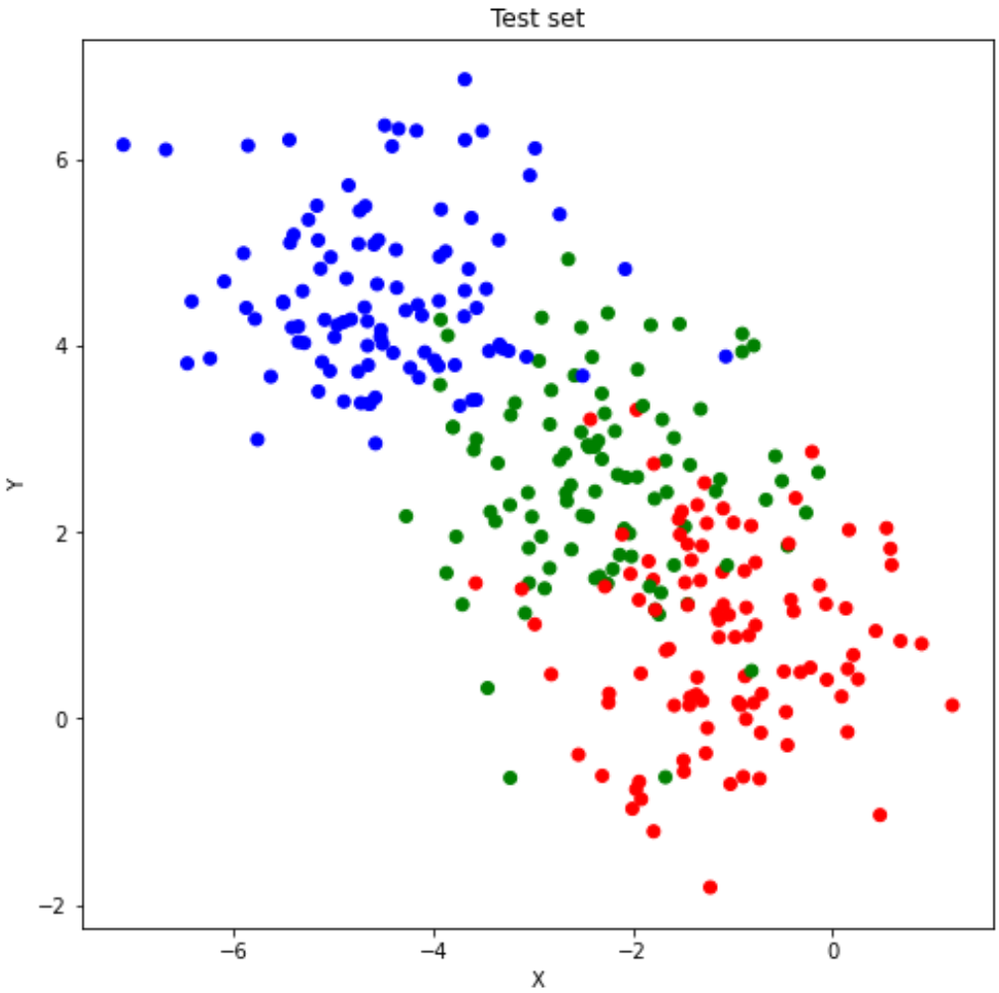
```
Text(0.5, 1.0, 'Train set')
```




```
# step 3

def get_test(T=300):
  testX, testY = list(), list()

  for i in range(T):
    temp = rnd.uniform(0, 1)

    if temp <= 1/3:
      testX.append(rnd.multivariate_normal(mu1, sigma))
      testY.append(0)
    elif temp <= 2/3:
      testX.append(rnd.multivariate_normal(mu2, sigma))
      testY.append(1)
    else:
      testX.append(rnd.multivariate_normal(mu3, sigma))
      testY.append(2)

  return np.array(testX), np.array(testY)

testX, testY = get_test()


# step 3 plot

plt.figure(figsize=(8, 8))
```

```
plt.scatter(testX[:, 0], testX[:, 1], c=colors[testY])
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Test set')
# plt.legend()
```

```
Text(0.5, 1.0, 'Test set')
```




```
# step 4

KNN = KNeighborsClassifier(n_neighbors=1)

KNN.fit(trainX, trainY)

y_train_pred = KNN.predict(trainX)
y_test_pred = KNN.predict(testX)

# CER = 1 - accuracy_score(testY, y_pred)

print('train CER:')
print(1 - accuracy_score(trainY, y_train_pred))
CER_train = sum([e1 != e2 for e1, e2 in zip(trainY, y_train_pred)]) / len(trainY)
print(CER_train, end='\n\n')

print('train CER:')
```

```
print(1 - accuracy_score(testY, y_test_pred))
CER_test = sum([e1 != e2 for e1, e2 in zip(testY, y_test_pred)]) / len(testY)
print(CER_test)
```

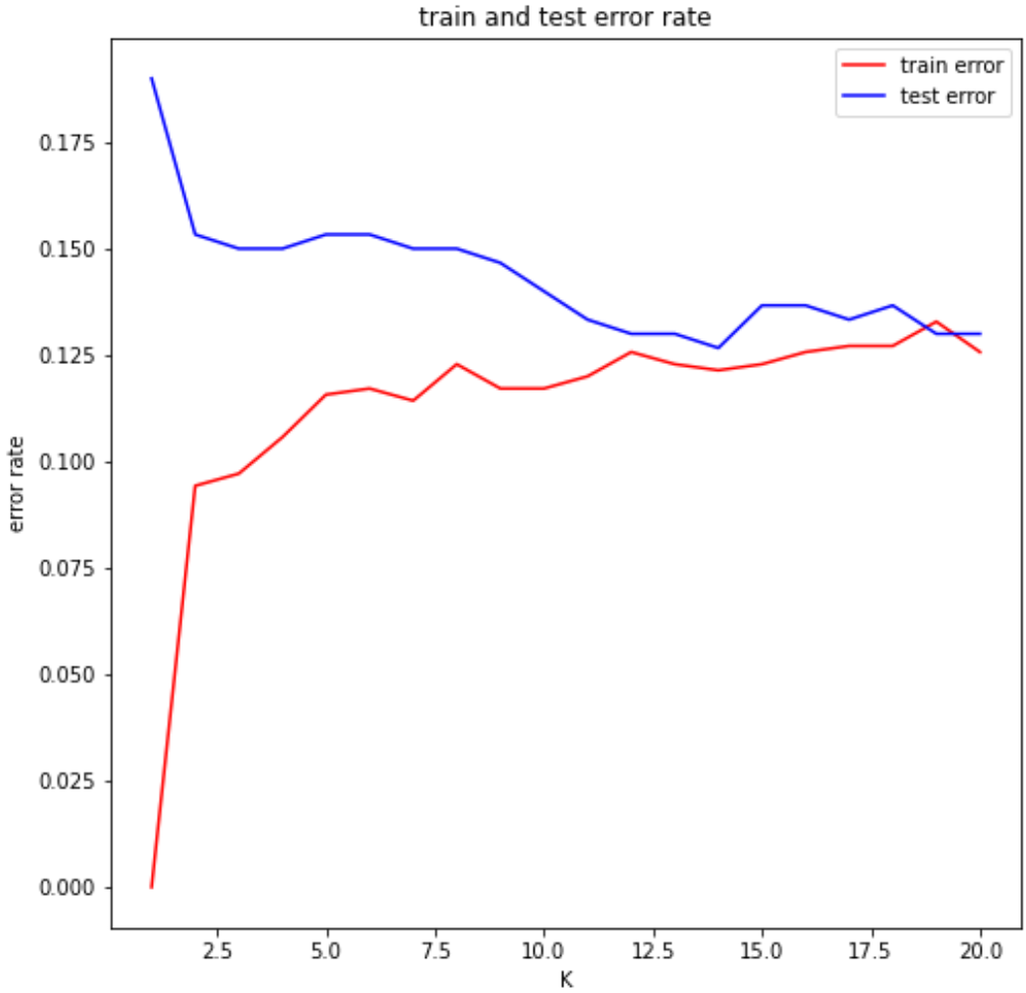
```
train CER:
0.0
0.0

train CER:
0.18999999999999995
0.19
```

1.4- classification error rate of 1-NN model on the train set :0
classification error rate of 1-NN model on the test set :0.19
As you can see there is a gap between the two values and this is due to the fact that the classification algorithm performs very well on the training set but not so much on the test set (error rate close to 20%) and as we have learned there's a suspicion that the model is overfited. It also depends on the size of the training set, larger training set descibes the distribution more clearly.

```
# step 5

cers_train = list()
cers_test = list()
K = list(range(1, 21))

for k in K:
  KNN = KNeighborsClassifier(n_neighbors=k)
  KNN.fit(trainX, trainY)

  y_train_pred = KNN.predict(trainX)
  y_test_pred = KNN.predict(testX)

  cers_train.append(1 - accuracy_score(trainY, y_train_pred))
  cers_test.append(1 - accuracy_score(testY, y_test_pred))


fig, ax = plt.subplots()

fig.set_size_inches(8, 8)

ax.plot(K, cers_train, label='train error', color='red')
ax.plot(K, cers_test, label='test error', color='blue')
ax.set_title('train and test error rate')
ax.set_xlabel('K')
```

```
ax.set_ylabel('error rate')
ax.legend()
```

```
<matplotlib.legend.Legend at 0x7fb1b202c590>
```



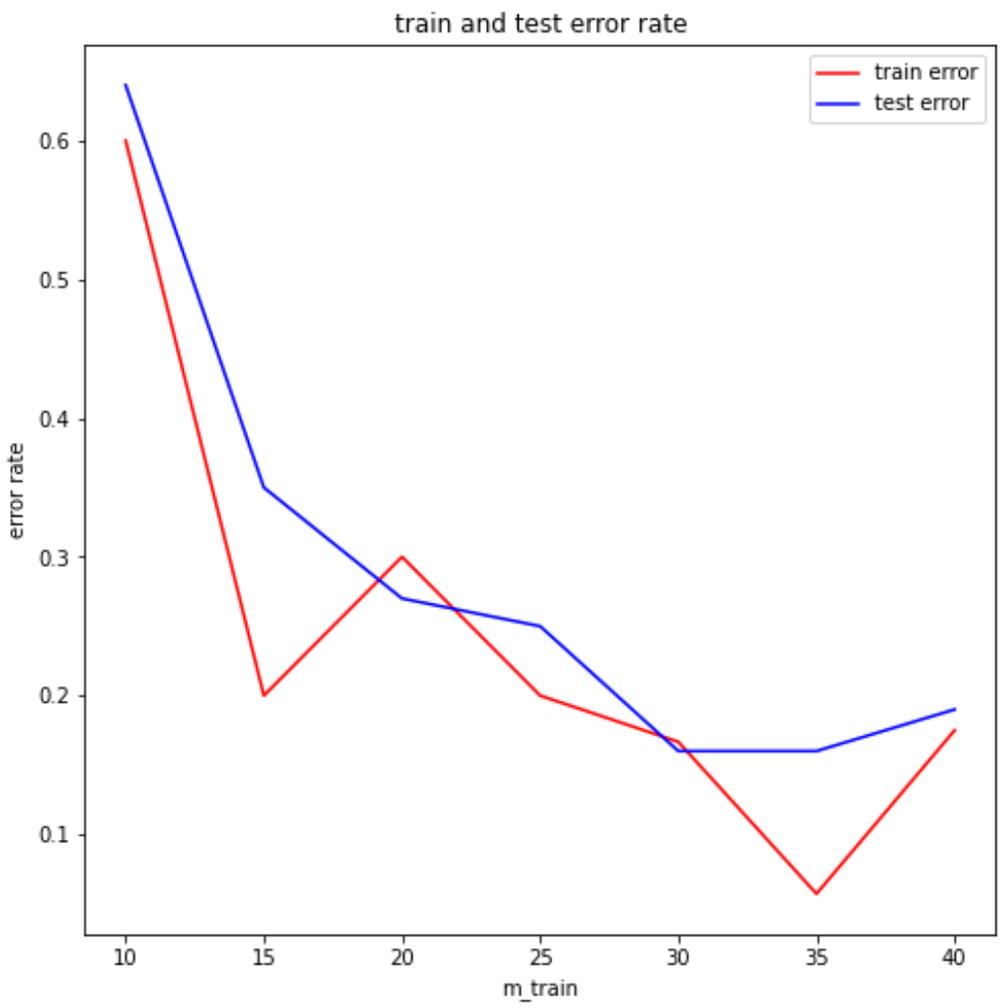

1.5- test error decreases with K in given field, but not always.

For example as shown in the drawing when K = 13 error 0.125 but when K = 15 error 0.145 ,specially it increased when we increased k.

```
# step 6

cers_train = list()
cers_test = list()

m_train = list(range(10, 41, 5))
m_test = 100
k = 10

testX, testY = get_test(m_test)

KNN = KNeighborsClassifier(n_neighbors=k)

for e in m_train:
  trainX, trainY = get_train(e)
  KNN.fit(trainX, trainY)


  y_train_pred = KNN.predict(trainX)
```

```
  y_test_pred = KNN.predict(testX)

  cers_train.append(1 - accuracy_score(trainY, y_train_pred))
  cers_test.append(1 - accuracy_score(testY, y_test_pred))


fig, ax = plt.subplots()

fig.set_size_inches(8, 8)

ax.plot(m_train, cers_train, label='train error', color='red')
ax.plot(m_train, cers_test, label='test error', color='blue')
ax.set_title('train and test error rate')
ax.set_xlabel('m_train')
ax.set_ylabel('error rate')
ax.legend()
```

```
<matplotlib.legend.Legend at 0x7f6f8cb9e210>
```




1.6- We expect that when we increase the size of the train set (m_train) the error rate will decrease in the train and test sets, but although we see a downward trend in the error rate, we see some exceptions for example on m_train=15 we see that the difference between the train

and the test error is relatively high which might indicate that the model is being overfitted to the

1.7-the line plots change between trials, This is because the size of the training set is small and because of the fact that the training set is being randomly generated, with that we see that all the plots usually indicated a decrease in the error rate of the training and test sets.

1.8- Suggestion version for k-NN that takes into account the distances of the neighbors to a new point : The labeling of the new point will be related to the labeling of all K neighbors So that weight is given to each label according to its distance from the new point The smaller the distance, the greater the weight ,According to the following equation:

$$\text{Prediction test}(x) = \frac{\sum_{i=1}^{k} W_i * \text{class}_i}{\sum_{i=1}^{k} W_i}$$

$$W_i = \frac{1}{\text{distance}(X_i, X_{test})}$$

## סעיף ב:

$\Leftarrow$ לכל $w_1, w_2 \in R^d$ קיים $w \in R^d$ כך ש:

$p = p_1 \quad , \quad 1-p = p_2$

נבחר $w = w_1 - w_2$ ונראה שמתקיים:

$$p = p_w(y_i = 1 \mid x_i) = \frac{e^{w^T x_i}}{1 + e^{w^T x_i}} = \frac{e^{(w_1 - w_2)^T x_i}}{1 + e^{(w_1 - w_2)^T x_i}}$$

$$= \frac{e^{w_1^T x_i - w_2^T x_i}}{1 + e^{w_1^T x_i - w_2^T x_i}} = \frac{\frac{e^{w_1^T x_i}}{e^{w_2^T x_i}}}{1 + \frac{e^{w_1^T x_i}}{e^{w_2^T x_i}}} = \frac{\frac{e^{w_1^T x_i}}{e^{w_2^T x_i}}}{\frac{e^{w_2^T x_i} + e^{w_1^T x_i}}{e^{w_2^T x_i}}}$$

$$= \frac{e^{w_1^T x_i}}{e^{w_1^T x_i} + e^{w_2^T x_i}} =: p_1$$

$$1 - p = 1 - \frac{e^{w_1^T x_i}}{e^{w_1^T x_i} + e^{w_2^T x_i}} = \frac{e^{w_2^T x_i}}{e^{w_1^T x_i} + e^{w_2^T x_i}} =: p_2$$

$\Rightarrow$ לכל $w \in R_d$ קיים $w_1, w_2 \in R^d$ כך ש:

$p = p_1 \ , \ 1-p = p_2 .$

בפרט עבור $w_1 = w$ ! $w_2 = 0$ שקיים

הלאנ:

$$p_1 = \frac{e^{w_1^T x}}{e^{w_1^T x} + e^{w_2^T x}} = \frac{e^{w^T x_i}}{e^{w^T x_i} + e^0} = \frac{e^{w^T x_i}}{e^{w^T x_i} + 1} = p$$

$$p_2 = \frac{e^{w_2^T x}}{e^{w_2^T x} + e^{w_1^T x}} = \frac{e^0}{e^{w^T x_i} + e^0} = \frac{1}{e^{w^T x_i} + 1}$$

$$= \frac{1 + e^{w^T x_i} - e^{w^T x_i}}{e^{w^T x_i} + 1} = 1 - \frac{e^{w^T x_i}}{e^{w^T x_i} + 1} =$$

$$1 - p$$

:

(2) כמו בהרצאה נניח ש

$$Y_i | X_i \sim Ber\left(p_i = \frac{e^{w^T x_i}}{1 + e^{w^T x_i}}\right)$$

פונקציית הנראות הנ"ל עבור אחת הינו:

$$L(Y_i | p_i) = p_i^{y_i} (1 - p_i)^{(1-y_i)}$$

ולכן ה log של הפונקציה: לכל $y_i \in \{1, \dots, K\}$

$$\log(L(Y_i = y_i | p_i)) = y_i \log p_i + (1-y_i)\log(1-p_i)$$

$$= y_i \log p_i - y_i \log(1-p_i) + \log(1-p_i)$$

$$y_i \log\left(\frac{p_i}{1-p_i}\right) + \log(1-p_i)$$

ולכן עבור כל הדגימות:

$$L(\omega) = \prod_{i=1} p_i^{y_i} (1-p_i)^{1-y_i}$$

$$\log(L(\omega)) = \sum_{i=1}^{K} \log(L(Y_i = K | X_i)) =$$

$$\sum_{i=1}^{K} I(y_i = K) \log\left(\frac{p_i}{1-p_i}\right) + \sum_{i=1}^{K} \log(1-p_i) =$$

$$\sum_{i=1} I(y_i = K)\, \omega_K^T x_i - \sum_{i=1}^{K} \log(1 + e^{\omega_K^T x_i})$$

$$\frac{\partial \log(L(\omega))}{\partial \omega} = \sum_{i=1}^{K} I(y_i = K)\, x_i - \sum_{i=1} \left(\frac{e^{\omega_K^T x_i}}{1 + e^{\omega_K^T x_i}}\right) x_i$$

w ס

לכן: לאפס מקבל:

$$\sum_{i=1}^{K} I(y_i=K)x_i - \sum_{i=1}^{K}\left(\frac{e^{w_K^T x_i}}{1+e^{w^T x_i}}\right)x_i = 0$$

$$\Rightarrow \sum_{i=1}^{K} I(y_i=K)\cdot x_i = \sum_{i=1}^{K}\left(\frac{e^{w_K^T x_i}}{1+e^{w_K^T x_i}}\right)x_i$$

③ $x_i \in R^2$ $y_i \in \{0,1,2\}$ —a

$w_1 = (8, -2.5, 2)$

$w_2 = (2, 0.5, -1.5)$

$w_3 (-10, 2, -0.5)$

in the multiclass formulation of logistic regression

אנו מחפשים את הנקודה הבינארי

לכן אותנו נצטרך לפחות ל 3 מחברים

כמו שנתנו בהגדלה רבים עם אלושה מגדים.

b— כפי שכתוב בהרצאה

$w = (w_0, w_1, w_2) \in R^3$

כדי ש $w_0$ ישחק תפקיד bias ולכן

$x$ נמצא קבוע כאחד בדוגמה $\Leftarrow$ $x \in R^2$.

(c) נניח $X_{i0}=1$ :

(1) $$X_i = \sum_{j=1}^{K} e^{w_K^T X_i}$$

(2) $$P_{iK} = P(Y_i = K \mid X_i) = \frac{e^{w_K^T X_i}}{\sum_{j=1}^{K} e^{w_K^T X_i}}$$

$$X_1 = \sum_{j=1}^{K} e^{w_K^T X_1} = e^{1\cdot 8 - 2\cdot 5\cdot 1 + 2\cdot 8} + e^{1\cdot 2 + 1\cdot 0\cdot 5 + 8\cdot -1\cdot 5} + e^{1\cdot -10 + 1\cdot 2 + 8\cdot \frac{1}{2}}$$

$$e^{21.5} + e^{-8.5} + e^{-12}$$

$$P_{11} = \frac{e^{w_1^T X_1}}{e^{21.5} + e^{-8.5} + e^{-12}} = \frac{e^{21.5}}{e^{21.5} + e^{-8.5} + e^{-12}} \approx 1$$

$$P_{12} = \frac{e^{w_2^T X_1}}{e^{21.5} + e^{-8.5} + e^{-12}} = \frac{e^{-8.5}}{e^{21.5} + e^{-8.5} + e^{-12}} \approx 0$$

$$P_{13} = \frac{e^{w_3^T X_1}}{e^{21.5} + e^{-8.5} + e^{-12}} = \frac{e^{-12}}{e^{21.5} + e^{-8.5} + e^{-12}} \approx 0$$

$\hat{y}_1 = 1$ : Label ⇐ מלא

$1\cdot 8 + 1\cdot -2\cdot 5 + 2\cdot -2$

$$x_2 = \sum_{j=1}^{K} e^{w_k^T x_2} = e^{\cdot}$$

$$+ e^{1\cdot 2 + 0.5\cdot 6 - 1\cdot 5\cdot -2}$$

$$+ e^{1\cdot -10 + 6\cdot + 2 - 0.5\cdot -2}$$

$$= e^{-11} + e^{8} + e^{3}$$

$$P_{2,1} = \frac{e^{-11}}{e^{-11}+e^{8}+e^{3}} \approx 0$$

$$P_{2,2} = \frac{e^{8}}{e^{-11}+e^{8}+e^{3}} \approx 1$$

$$P_{2,3} = \frac{e^{3}}{e^{-11}+e^{8}+e^{3}} \approx 0$$

לכן ה label :

$$\hat{y}_2 = 2$$

$$x_3 = \sum_{j=1}^{K} e^{w_k^T \cdot x_3} = e^{-14} + e^{2} + e^{12}$$

$$P_{3,1} = \frac{e^{-14}}{e^{-14} + e^{2} + e^{12}} \approx 0$$

$$P_{3,2} = \frac{e^{2}}{e^{-14} + e^{2} + e^{12}} \approx 0$$

$$P_{3,3} = \frac{e^{12}}{e^{-14} + e^{2} + e^{12}} \approx 1$$

$\hat{y}_3 = 3$ , Label $\Rightarrow$ [illegible]