Applied machine learning
Group assignment 2

__Team members-Group 8:__

- Abdelrhman Gaber Youssef Saad Rezkallah
- Eman Metwally Mohammed Abood
- Basma Reda Shaban Abd-Elsalam Abd-Elwahab

Part 1: Calculations

Problem1:

P(Yes)	9/15
P(No)	6/15
Total	15

Colors			
		P(Yes)	**P(No)**
Red	5/15	2/9	3/6
Green	5/15	3/9	2/6
Yellow	5/15	4/9	1/6
Total	15	9	6

Gender			
		P(Yes)	**P(No)**
Female	7/15	6/9	1/6
Male	8/15	3/9	5/6
Total	15	9	6

Price			
		P(Yes)	**P(No)**
High	4/15	2/9	2/6
Medium	5/15	2/9	3/6
Low	6/15	5/9	1/6
Total	15	9	6

P (C = Yes | Green, Female, High) =

$$\frac{\mathrm{p(Green|\ C=Yes).p(Female|\ C=Yes).p(High|C=Yes)}.p(yes)}{p\mathrm{(Green|C=Yes).p(Female|C=Yes).p(High|\ C=Yes)}.p(yes) + \mathrm{(Green|C=No).p(Female|\ C=No).p(High|\ C=No)}}$$

P (C = Yes | Green, Female, High). p(Yes) = $\frac{3}{9}.\frac{6}{9}.\frac{2}{9}.\frac{9}{15} = \frac{4}{135}$

P (C = No | Green, Female, High). p(No) = $\frac{2}{6}.\frac{1}{6}.\frac{2}{6}.\frac{6}{15} = \frac{1}{135}$

P (color = Green, Gender = Female, price = High) = P (C = Yes | Green, Female, High) * (Yes) + P (C = No | Green, Female, High) * P(No)

P (C = Yes | Green, Female, High) = $\frac{3}{9}.\frac{6}{9}.\frac{2}{9} = \frac{4}{81}$
P (C = No | Green, Female, High) = $\frac{2}{6}.\frac{1}{6}.\frac{2}{6} = \frac{1}{54}$
P (color = Green, Gender = Female, price = High) = $\frac{4}{81} . \frac{9}{15} + \frac{1}{54} . \frac{6}{15} = \frac{1}{27}$
P (C = Yes | Green, Female, High) = $\frac{\frac{4}{135}}{\frac{1}{27}} = \frac{4}{5} = 80\%$
P (C = No | Green, Female, High) = $\frac{\frac{1}{135}}{\frac{1}{27}} = \frac{1}{5} = 20\%$

problem 2:

Target	Class1	Class2
A1(choose class2)	5	2
A2(choose class1)	0	5
A3(Reject)	4	4

P(α_1|X) = 0P(C1|X) + 5P(C2|X)
= 5 – (1- P(C1|X)
= 5 -5(P(C1|X)

P(α_2|X) = 5P(C1|X) + 2P(C2|X)
= 5P(C1|X) + 2(1-P(C1|X))
= **2 + 3P(C1|X)**

P (α_r | x) = 4
R ($\boldsymbol{\alpha_1}$ | X) < 4
5 – 5P (C1 | X) < 4
P (C1 | X) > $\mathbf{\frac{1}{5}}$

R ($\boldsymbol{\alpha_2}$ | X) < 4
2 + 3P(C1|X) < 4
P(C1|X) < $\mathbf{\frac{2}{3}}$

But, there are no intersection between the two classes, so no rejection area exists.

Part 2: Programming

Problem1: Implementation steps:

1- **Load wine dataset:**
These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

```
[2] #load the dataset
    data = load_wine()
```

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

```
[3] df = pd.DataFrame(data.data,columns=data.feature_names)
```

```
#print the head of the data
df.head()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	

2- **Extracting the most two important features:**

• The pair plot diagram shows the most important features to select, A pair plot plots a pairwise relationships in a data set.

• The pair plot function creates a grid of Axes such that each variable in data will be shared in the y-axis across a single row and in the x-axis across a single column.

The pair Plot between the features Diagram:

1- train test split function in scikitlearn to split the dataset into a training set and testing set:

```
[ ] from sklearn.model_selection import train_test_split
    x_train ,x_test, y_train , y_test = train_test_split(df.drop(0,axis = 1),df[0],test_size = 0.2 ,random_state=42)
```

```
[ ] #print the x_train data
    x_train
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_c
158	14.34	1.68	2.70	25.0	98.0	2.80	1.31	0.53	2.70	13.00	0.57	
137	12.53	5.51	2.64	25.0	96.0	1.79	0.60	0.63	1.10	5.00	0.82	
98	12.37	1.07	2.10	18.5	88.0	3.52	3.75	0.24	1.95	4.50	1.04	
159	13.48	1.67	2.64	22.5	89.0	2.60	1.10	0.52	2.29	11.75	0.57	
38	13.07	1.50	2.10	15.5	98.0	2.40	2.64	0.28	1.37	3.70	1.18	
...	...	...	...	...	...	...	...	...	...	...	...	
71	13.86	1.51	2.67	25.0	86.0	2.95	2.86	0.21	1.87	3.38	1.36	
106	12.25	1.73	2.12	19.0	80.0	1.65	2.03	0.37	1.63	3.40	1.00	
14	14.38	1.87	2.38	12.0	102.0	3.30	3.64	0.29	2.96	7.50	1.20	
92	12.69	1.53	2.26	20.7	80.0	1.38	1.46	0.58	1.62	3.05	0.96	
102	12.34	2.45	2.46	21.0	98.0	2.56	2.11	0.34	1.31	2.80	0.80	

142 rows × 13 columns

2- Implement GaussianNB classifier:

Gaussian Naive Bayes is a variant of Naive Bayes that follows Gaussian normal.

```
[14] from sklearn.naive_bayes import GaussianNB #to import GaussianNB classifier library
     nb = GaussianNB()
     nb.fit(x_train,y_train)
```

```
[15] #define y_pred to make the prediction
     y_pred = nb.predict(x_test)
```

3- use the classification report to calculate precision, recall and F1 score:

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives, and False Negatives are used to predict the metrics of a classification report.

```
[16] from sklearn.metrics import classification_report #to use the function of classification report
     print(classification_report(y_train,nb.predict(x_train)))
     print(classification_report(y_test,y_pred))
```

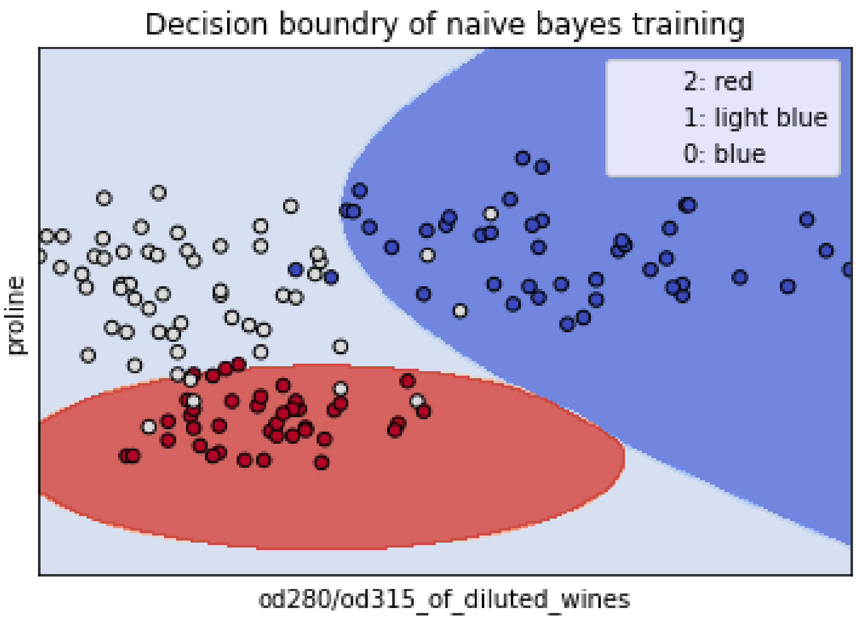
```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        45
           1       0.96      0.96      0.96        57
           2       0.95      1.00      0.98        40

    accuracy                           0.97       142
   macro avg       0.97      0.97      0.97       142
weighted avg       0.97      0.97      0.97       142

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00         8

    accuracy                           1.00        36
   macro avg       1.00      1.00      1.00        36
weighted avg       1.00      1.00      1.00        36
```

The accuracy on training is 0.97
The accuracy on the testing phase is 1.00

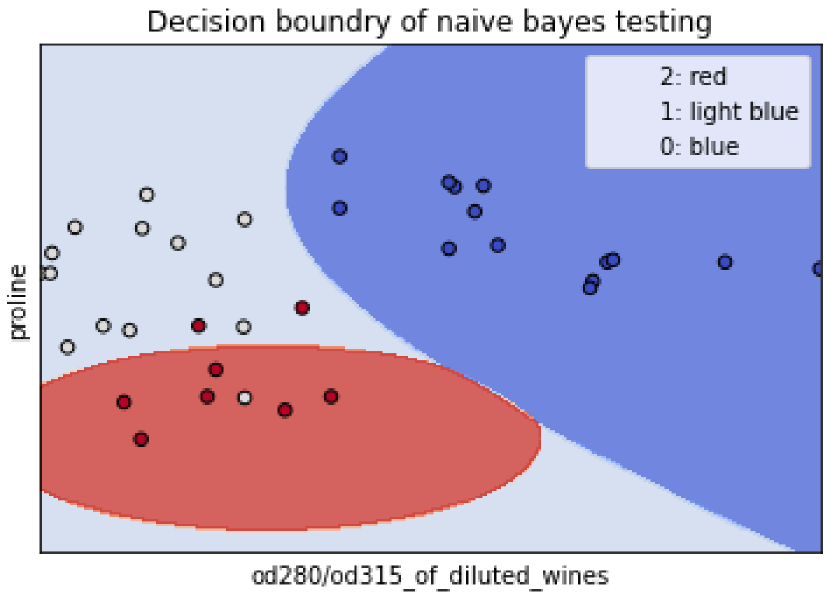
4- Plot the decision boundary:

• we choose The Two features ‘od280/od315_of_diluted_wines’ and ‘proline’ because the plot shows that we can separate between these classes.

6.1 plot the decision boundary on train data between 'proline' and 'od280/od315_of_diluted_wines’:



6.2 plot the decision boundary on test data between 'proline' and 'od280/od315_of_diluted_wines':



The decision boundary diagram shows that we can separate between the 2 classes **'proline' and 'od280/od315_of_diluted_wines'.**

Problem 2: Implementation of KNN Classifier

1- Load car_evaluation dataset:

```
#read the dataset
car = pd.read_csv('/content/car_evaluation.csv')
car.head()
```

	vhigh	vhigh.1	2	2.1	small	low	unacc
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc

2- perform label encoding:

```
#perform label encoding
car['vhigh'] = car['vhigh'].map({'low':0,'med':1,'high':2,'vhigh':3})
car['vhigh.1'] = car['vhigh.1'].map({'low':0,'med':1,'high':2,'vhigh':3})
car['2'] = car['2'].map({'2':0,'3':1,'4':2,'5more':3})
car['2.1'] = car['2.1'].map({'2':0,'4':1,'more':2})
car['small'] = car['small'].map({'small':0,'med':1,'big':2})
car['low'] = car['low'].map({'low':0,'med':1,'high':2})
car['unacc'] = car['unacc'].map({'unacc':0,'acc':1,'good':2,'vgood':3})
```

The result of label encoding is:

```
[33] #print the head of the data after label encoding
car.head()
```

	vhigh	vhigh.1	2	2.1	small	low	unacc
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc

a- split the dataset to training and testing: data preparation step:

```python
from sklearn.model_selection import train_test_split
x_train, x ,y_train, y = train_test_split(car.drop('unacc',axis = 1) , car['unacc'],train_size = 1000,random_state = 42,stratify=car['unacc'])
x_val,x_test , y_val , y_test = train_test_split(x,y,test_size = 428,random_state = 42,stratify=y)
```

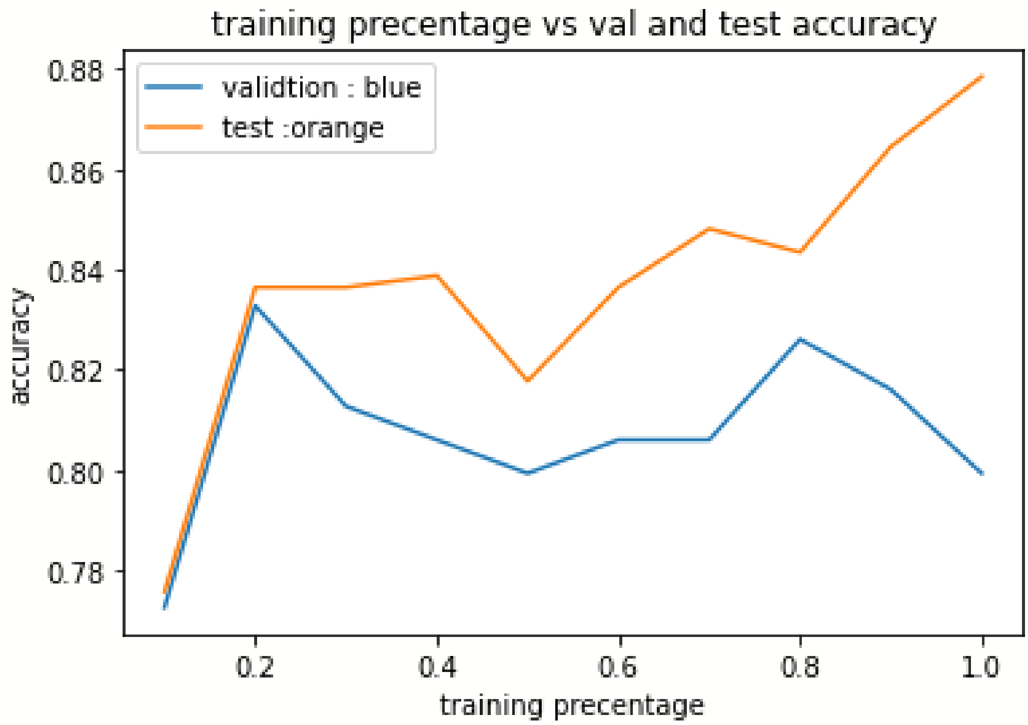
b-Define the percentage of x_train and y_train:

```python
def percentage(x_train , y_train ,precentage):
    x_train_prc = x_train.iloc[:int(x_train.shape[0]*precentage)]
    y_train_prc = y_train.iloc[:int(y_train.shape[0]*precentage)]
    return x_train_prc , y_train_prc
```

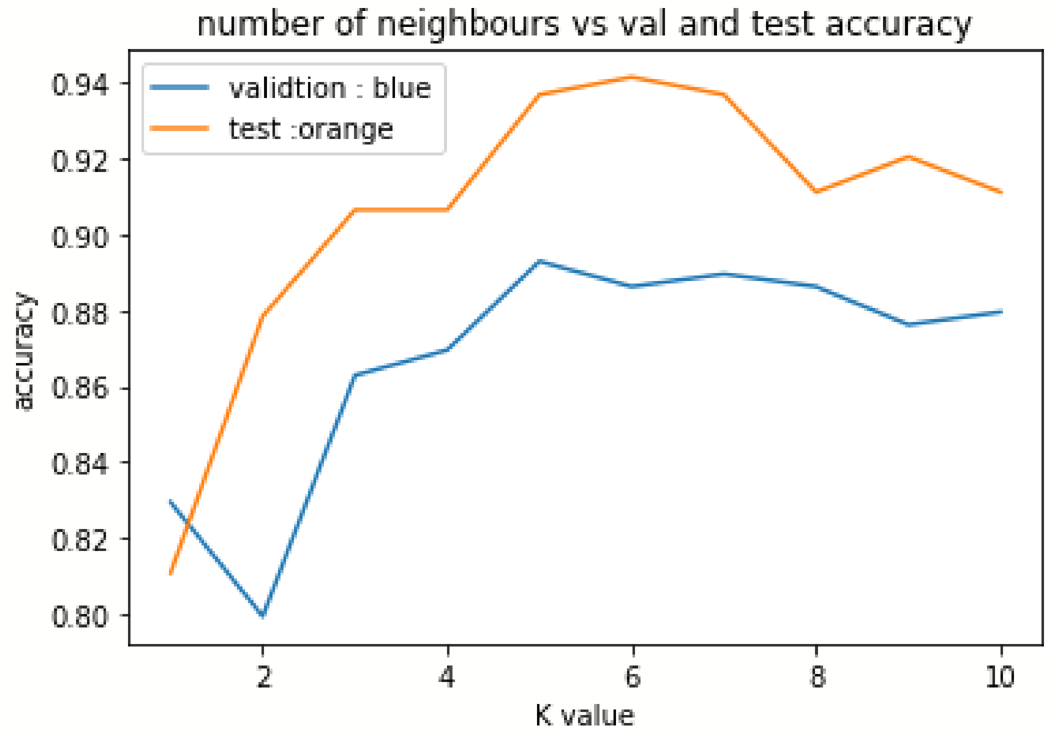
```python
prc = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]
```

c- plot the training set and accuracy score:

The maximum accuracy of validation is 83%
The maximum accuracy of test is 88%

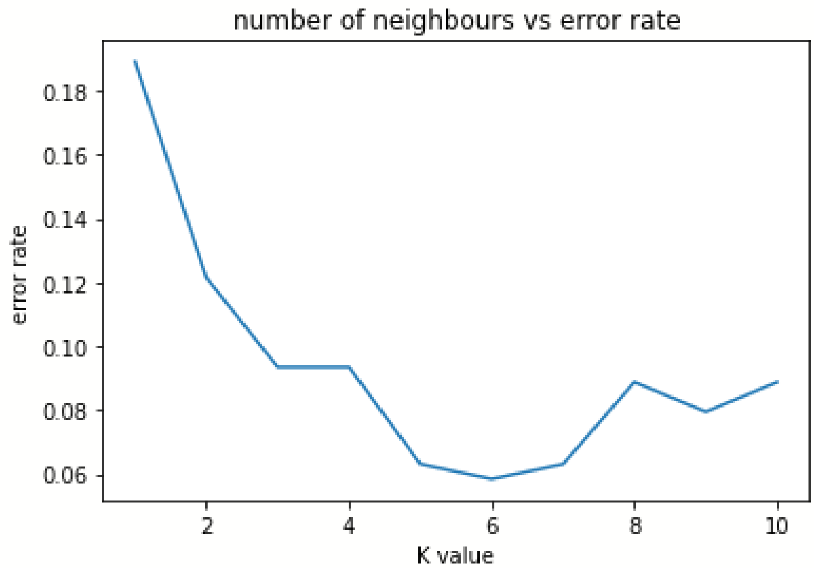


d- accuracy curve on the validation set when K varies from 1 to 10



The maximum accuracy of validation is 89%.
The maximum accuracy of test is 94%.
The maximum K-value is 5.
The minimum error rate is 0.058.

e-
Analysis the training time when use different number of training sampl es.



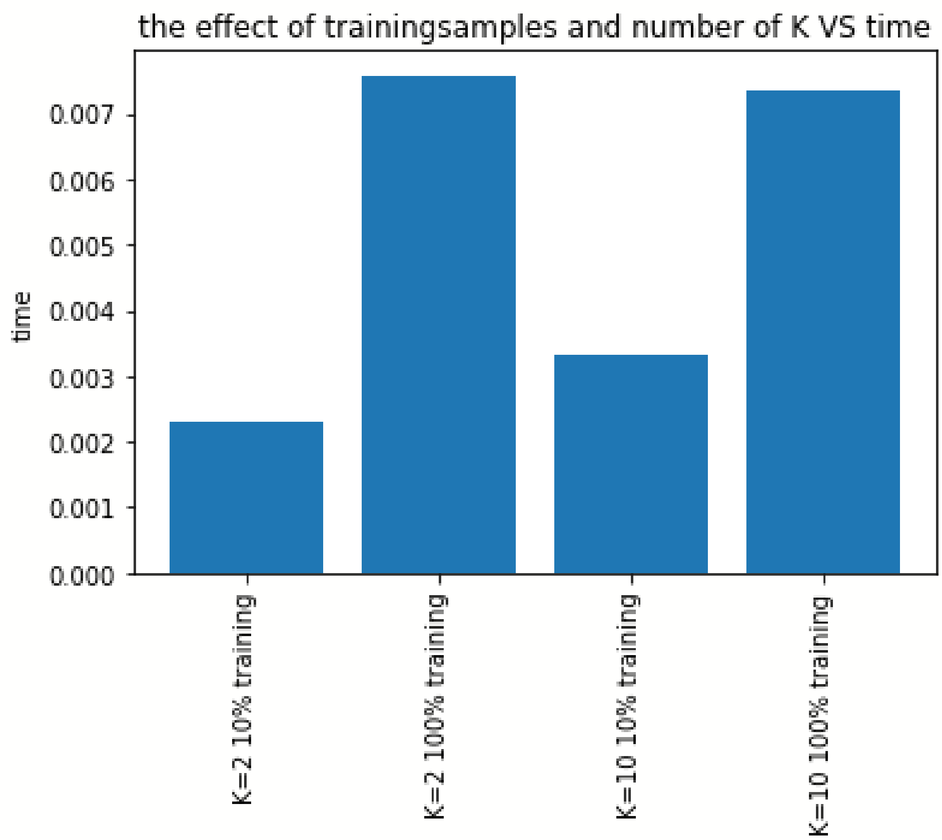
Text (0, 0.5, 'error rate')

Analysis the training time when use different number of training samples.

•10% of the whole training set and K = 2
•100% of the whole training set and K = 2
•10% of the whole training set and K = 10
•100% of the whole training set and K = 10

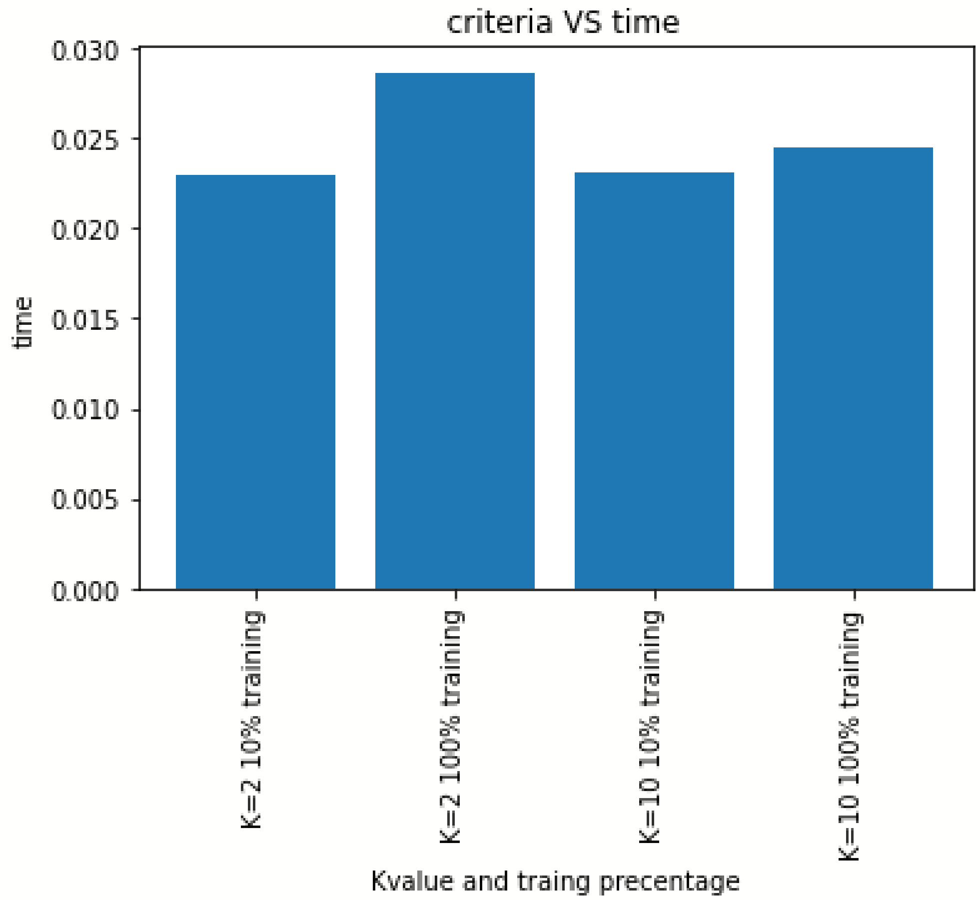
Case:	**10% training and k=2**	**100% training and k=2**	**10% training and k=10**	**100% training and k=10**
Start time	1655224913.2653127	1655224913.3789108	1655224913.491071	1655224913.5868094
End time	1655224913.2882903	1655224913.4075544	1655224913.5141633	1655224913.6113305
Total time	0.022977590560913086	0.0286436080932617 2	0.023092269897460938	0.0245211124420166

The bar chart figure shows the prediction time on the training set.



So, the training time when K = 2 and 100% training is the highest time.

bar chart figure to show prediction time on the testing set:



The testing time when k = 2 and with 100% training is the highest time.

f- from the experiments on points c, d, e we can say that:

1- when the number of k decreased, it can lead to an overfitting.
2- When the number of k increased, it can lead to an underfitting.
3- We must choose the best number of K to prevent overfitting or underfitting.
4- when the number of the training data increased, the time will increase because it is non-parametric model, and it is not limited to that, so it causes better results.

Conclusion:

During this assignment we learned more about the calculation process of Bayesian Rule Based Classifier to make prediction on a dataset, and how to calculate the risk and rejection area of the model.

We learned more about the Naïve Bayes classifier, and how to use the GaussinNB classification algorithm, how to extract the most important features and show them in the decision boundary figure, after that we learned more about the KNN classification algorithm and how to apply different numbers of training set and different numbers of K, so we realized that the KNN algorithm require more time when the number of K increased.