Report

ELG5255[EG] APPLIED MACHINE LEARNING [LEC] 20225

Assignment 2

GROUP 11
Mostafa Mahmoud Abd Elwahab Nofal
Nada Abdellatef Shaker Seddik
Hadeer Mamouh Abdelfattah Mohammed

Part One: Calculations

1. P(buy=yes) = 9/15 = 0.6
   P(buy=No) = 6/15=0.4

Color	Yes	No
R	2/9	3/6
G	3/9	2/6
Y	4/9	1/6

Price	Yes	No
H	2/9	2/6
M	2/9	3/6
L	5/9	1/6

Gender	Yes	No
M	3/9	5/6
F	6/9	1/6

Color = G, Gender = F, Price=H

P(yes)= P(yes)*P(G|yes) *P(F|yes) *P(H|yes) $= 0.6 * \frac{3}{9} * \frac{6}{9} * \frac{2}{9} = \frac{4}{135}$

P(No)=P(No)*P(G|No) *P(F| No) *P(H| No) $=0.4 * \frac{2}{6} * \frac{1}{6} * \frac{2}{6} = \frac{1}{135}$

P(Yes)= $\frac{\frac{4}{135}}{\frac{4}{135}+\frac{1}{135}} = \frac{4}{5}$

P(No)= $1 - \frac{4}{5} = \frac{1}{5}$

Max $(\frac{4}{5}, \frac{1}{5}) = \frac{4}{5}$

At last yes, the consumer will buy the cloth

2. Loss table

Target	Class 1	Class 2
A1 (Choose class 2)	5	2
A2(Choose class 1)	0	5
A3(Rejection)	4	4

$\mathrm{R}(\alpha 1|x) = P(C1)P(C1|x) + P(C2)P(C2|x) < 4$

$$5P(C1|x) + 2 * 1 - P(C1|x) < 4$$

$$5P(C1|x) + 2 - 2P(C1|x) < 4$$

$$3P(C1|x) + 2 < 4$$

$$P(C1|x) < \frac{2}{3}$$

$R(\alpha 2|x) = 0P(C1|x) + 5P(C2|x) < 4$

$5 * 1 - P(C1|x) < 4$

$1 - P(C1|x) < \frac{4}{5}$

$P(C1|x) > \frac{1}{5}$

We choose $\alpha 1$ if:

- $\mathrm{R}(\alpha 1|x) < 4 \;\; => \;\; P(C1|X) < \frac{2}{3}$

We choose $\alpha 2$ if:

- $\mathrm{R}(\alpha 2|x) < 4 \;\; => \;\; P(C1|X) > \frac{1}{5}$

No Rejection area $\frac{1}{5} > \mathrm{P(C1|x)} > \frac{2}{3}$

Problem Two: Programming (Naïve Bayes)

Import libraries

```
[1] from sklearn.datasets import load_wine
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    from mlxtend.plotting import plot_decision_regions
    %matplotlib inline
    import time

    from sklearn.model_selection import train_test_split
    from sklearn.naive_bayes import GaussianNB
    from sklearn.metrics import classification_report, ConfusionMatrixDisplay, accuracy_score, confusion_matrix
    from sklearn.feature_selection import SelectKBest
    from sklearn.feature_selection import chi2
```

Read the load_wine dataset

```
[19] dataset = load_wine()
     x = pd.DataFrame(dataset.data, columns = ["Alcohol",  "Malic acid", "Ash",  "Alcalinity of ash",  "Magnesium",  "Total phenols",  "Flavanoids", "Nonflavanoid phe
     y = pd.DataFrame(dataset.target, columns = ["Target"])
```

```
[20] x.head()
```

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065.0
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050.0
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185.0
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480.0
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735.0

Naïve Bayes

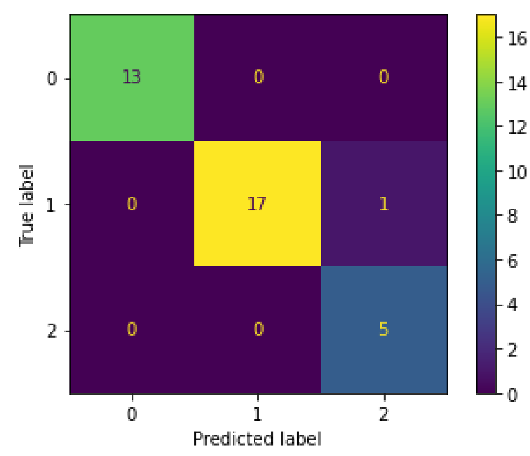
Train the Naïve Bayes model after splitting the dataset with train_test_split function

```
[50] model = GaussianNB()
     model = model.fit(x_train, y_train)
     y_predict = model.predict(x_test)
```

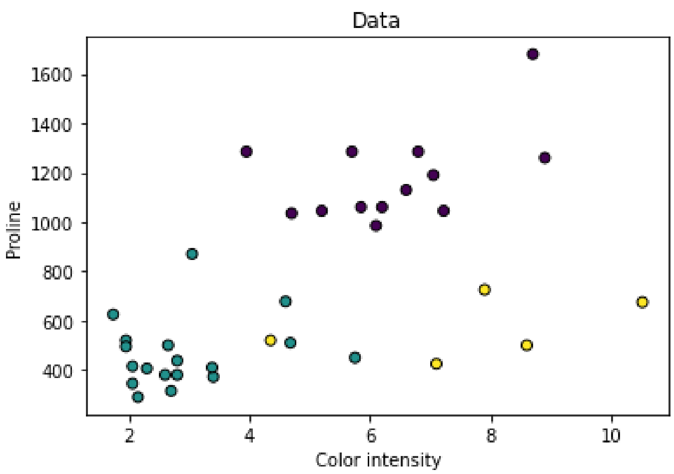
Display the classification report of the model

```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       1.00      0.94      0.97        18
           2       0.83      1.00      0.91         5

    accuracy                           0.97        36
   macro avg       0.94      0.98      0.96        36
weighted avg       0.98      0.97      0.97        36
```

Confusion Matrix



Decision boundary without the model



We want to plot the decision boundary with the model so we need to train it again on just two features and chose chi-square to select our most two important features

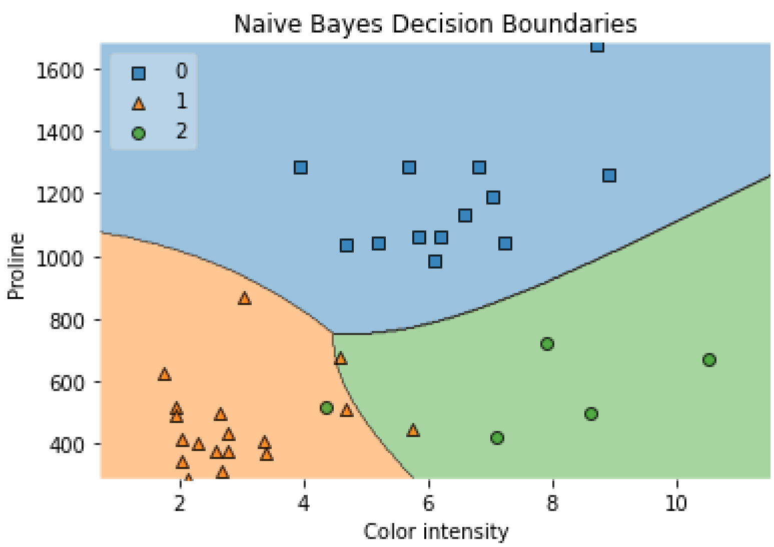
```
[52] chi2_features = SelectKBest(chi2, k = 2)
     X_kbest_features = chi2_features.fit_transform(x_train, y_train)
     X_kbest_features
```

We ended up with that "Color intensity" and "Proline" columns are the most important two features so we trained the model again with them

```
[27] x_train2 = x_train.loc[:,["Color intensity","Proline"]]
     x_test2 = x_test.loc[:,["Color intensity","Proline"]]
```

```
model2 = GaussianNB()
model2 = model2.fit(x_train2, y_train)
y_predict2 = model2.predict(x_test2)
```

Decision boundary with the model



Problem Two: Programming (KNeighborsClassifier)

Import libraries

```
[59] import pandas as pd
     import os
     import numpy as np
     from sklearn.preprocessing import LabelEncoder,OrdinalEncoder
     import matplotlib.pyplot as plt
     from sklearn import metrics
     import time
     from sklearn.neighbors import KNeighborsClassifier
     import matplotlib.pyplot as plt

     !pip install fast_ml
     from fast_ml.model_development import train_valid_test_split
```

Read the car_evaluation dataset

```
[60] car=pd.read_csv(r"car_evaluation.csv",names=['price', 'maint', 'doors', 'ppl', 'lug_boot', 'safety', 'acceptable'])
```

```
[61] car.head()
```

	price	maint	doors	ppl	lug_boot	safety	acceptable
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc

(b) Encoders

We used ordinal encoder for the features to transform the string values to numbers

```
[62] car['price'].unique()

     array(['vhigh', 'high', 'med', 'low'], dtype=object)

[63] category_price=['low', 'med', 'high','vhigh']
     or_prince=OrdinalEncoder(categories=[category_price])
     car[['price']] =or_prince.fit_transform(car[['price']])
     car.head()
```

	price	maint	doors	ppl	lug_boot	safety	acceptable
0	3.0	vhigh	2	2	small	low	unacc
1	3.0	vhigh	2	2	small	med	unacc
2	3.0	vhigh	2	2	small	high	unacc
3	3.0	vhigh	2	2	med	low	unacc
4	3.0	vhigh	2	2	med	med	unacc

Label encoder was also used for the label column

```
en=LabelEncoder()
car['acceptable']=en.fit_transform(car['acceptable'])
car.head()
```

	price	maint	doors	ppl	lug_boot	safety	acceptable
0	3.0	3.0	0.0	0.0	0.0	0.0	2
1	3.0	3.0	0.0	0.0	0.0	1.0	2
2	3.0	3.0	0.0	0.0	0.0	2.0	2
3	3.0	3.0	0.0	0.0	1.0	0.0	2
4	3.0	3.0	0.0	0.0	1.0	1.0	2

(a) Split the data

We used the train_valid_test_split function from fast_ml library to split the data but we implemented the split ratio for each one manually in it and data shuffle is the default option

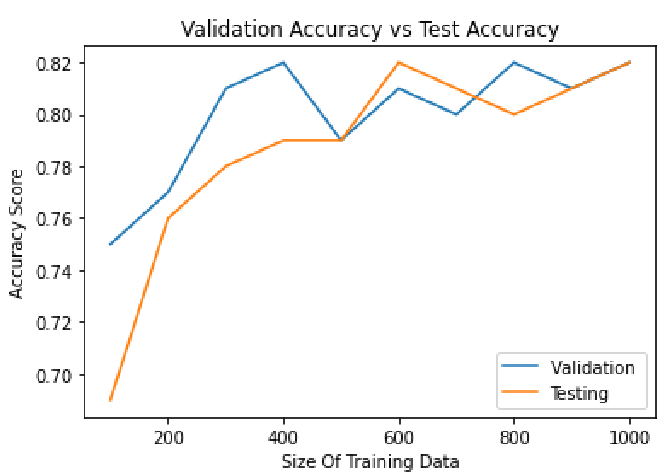
```
[30] X_train, y_train, X_valid, y_valid, X_test, y_test = train_valid_test_split(
         car,
         target = 'acceptable',
         train_size=1000/1728,
         valid_size=300/1728,
         test_size=428/1728,
         random_state=42)
```

```
[31] print(X_train.shape, X_valid.shape, X_test.shape)
     print(y_train.shape, y_valid.shape, y_test.shape)

     (1000, 6) (300, 6) (428, 6)
     (1000,) (300,) (428,)
```

(c) Train the 10 KNN models

We trained the models with the mentioned ratios of training sets. Here are the validation and test accuracies and their plot

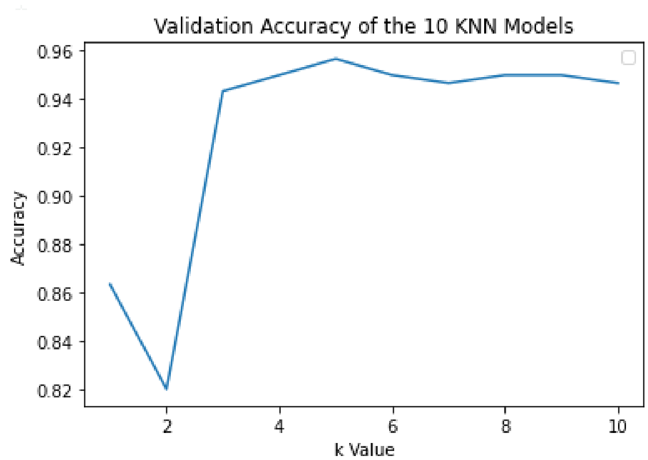
```
[43] print(validation_accuracy)
     print(testing_accuracy)

     [0.75, 0.77, 0.81, 0.82, 0.79, 0.81, 0.8, 0.82, 0.81, 0.82]
     [0.69, 0.76, 0.78, 0.79, 0.79, 0.82, 0.81, 0.8, 0.81, 0.82]
```

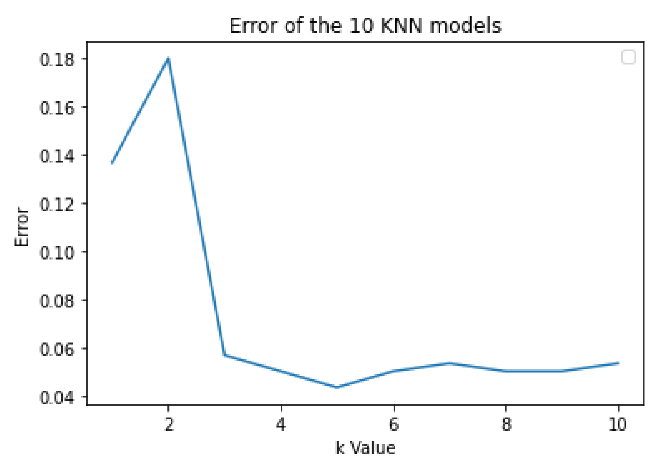


(c) Conclusion

We can notice that the accuracy is increasing rapidly at first when we train the model with higher ratios of training set but it reaches a point where it keeps increasing and decreasing

(d) Selecting the best k from 1 to 10





(d) Conclusion

We used the accuracy and analyzing error methods to determine and the best k from 1 to 10 for the KNN model which was 5.

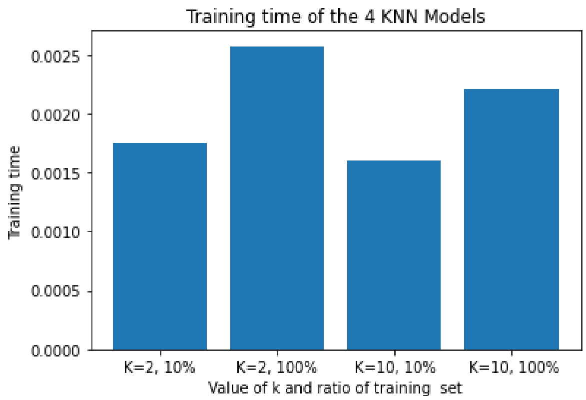
(e) Training and prediction time for the 4 KNN models

- 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.

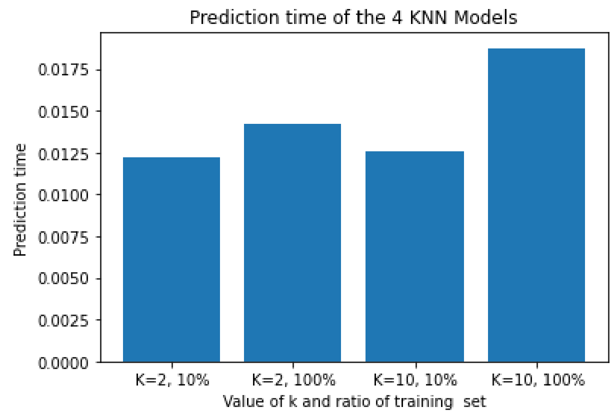
Analysis of the training and prediction time

```
[89] print(training_time)
     print(prediction_time)

     [0.0017571449279785156, 0.002576589584350586, 0.0016086101531982422, 0.0022115707397460938]
     [0.012192487716674805, 0.014163494110107422, 0.012578725814819336, 0.018758535385131836]
```

Training time bar chart



Prediction time bar chart



(e) Conclusion

Regarding the training time, every time we run the program, it varies but most of the times we noticed the following:

• When k = 2, the model takes more time than when we set the k equal to 10

• When we train the model with our full training data, it takes time more than the model that was trained with just 10% of the data