```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
import math
import seaborn as sns
```

```
df = pd.read_csv('/content/sample_data/Iris.csv')
```

```
df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
df.isna().sum().sum()
```

```
0
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Species'] = le.fit_transform(df['Species'])
```

```
df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0

```
X = df.iloc[:,:-1].values
y = df.iloc[:,-1].values
```

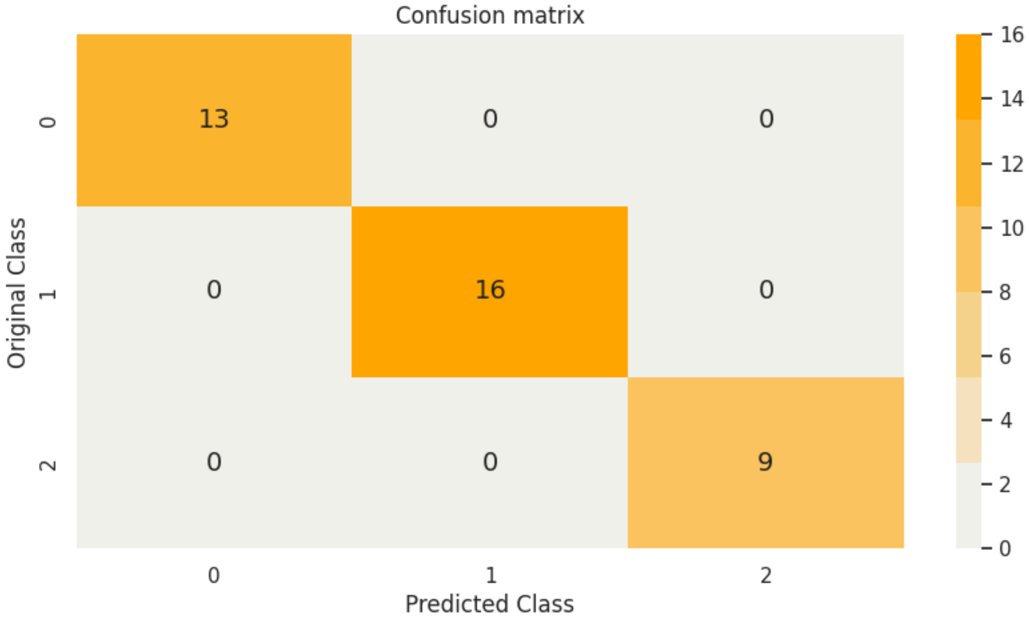
```
def plot_confusion_matrix(test_y, predict_y):
    C = confusion_matrix(test_y, predict_y)
    sns.set(font_scale=1)
    plt.figure(figsize=(10,5))
    labels = [0,1,2]
    # representing A in heatmap format
    cmap1=sns.light_palette("orange")
    sns.heatmap(C, annot=True, cmap=cmap1, fmt=".0f", xticklabels=labels,
yticklabels=labels,annot_kws={"size":14})
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Confusion matrix")
    plt.show()
```

```
def model_evaluations(X_train,y_train,X_test,y_test):
    gb = GaussianNB()
    knn = KNeighborsClassifier(round(math.sqrt(X_train.shape[0])))
    dt = DecisionTreeClassifier()
    gb.fit(X_train,y_train)
    knn.fit(X_train,y_train)
    dt.fit(X_train,y_train)
    y_pred = gb.predict(X_test)
    print(f"\nmodel: Naive bayes \naccuracy:{accuracy_score(y_test,y_pred)}")
    plot_confusion_matrix(y_test,y_pred)
    y_pred = knn.predict(X_test)
    print(f"\n\nmodel: k nearest neighbors \naccuracy:{accuracy_score(y_test,y_pred)}")
    plot_confusion_matrix(y_test,y_pred)
    y_pred = dt.predict(X_test)
    print(f"\n\nmodel:decision tree \naccuracy:{accuracy_score(y_test,y_pred)}")
    plot_confusion_matrix(y_test,y_pred)
```

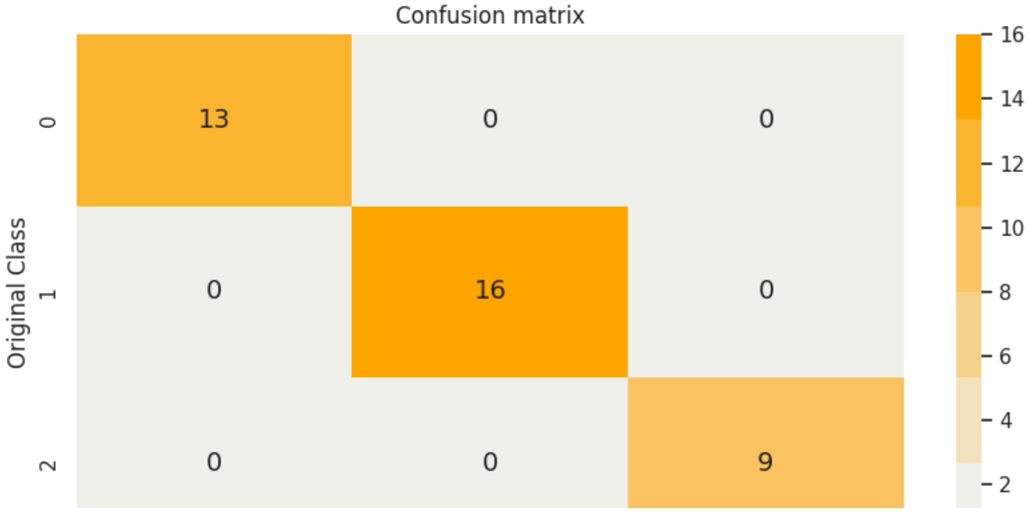
```
# Different sizes of train and test sets
# train = 0.75   test = 0.25
seed = 100
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=0)
print("accuracy score for models with train set = 0.75 and test set = 0.25 ")
model_evaluations(X_train, y_train, X_test, y_test)
```

```
accuracy score for models with train set = 0.75 and test set = 0.25

model: Naive bayes
accuracy:1.0
```



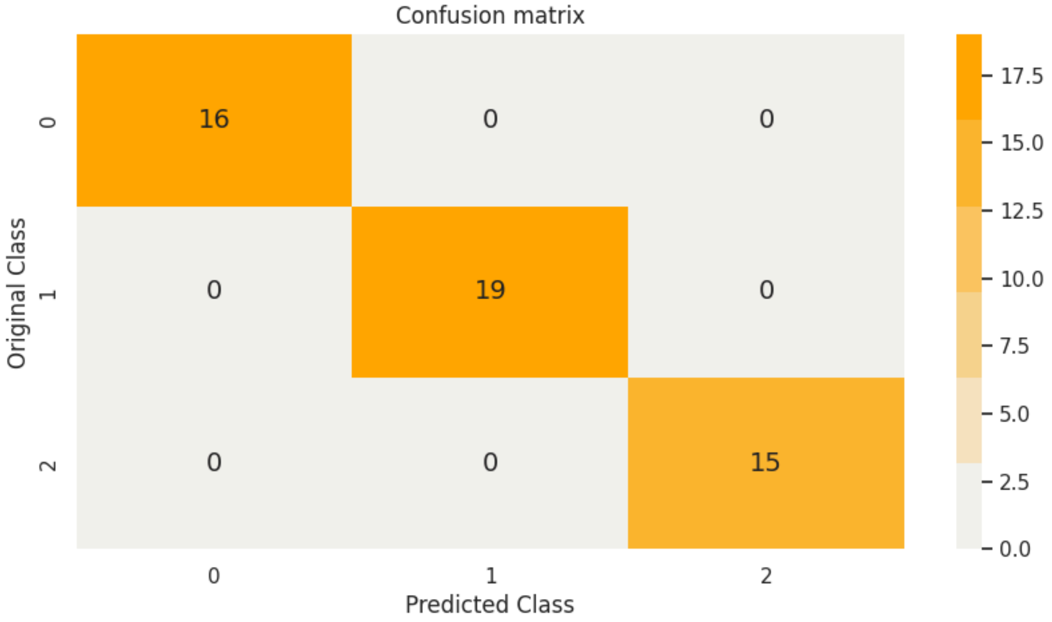
```
model: k nearest neighbors
accuracy:1.0
```



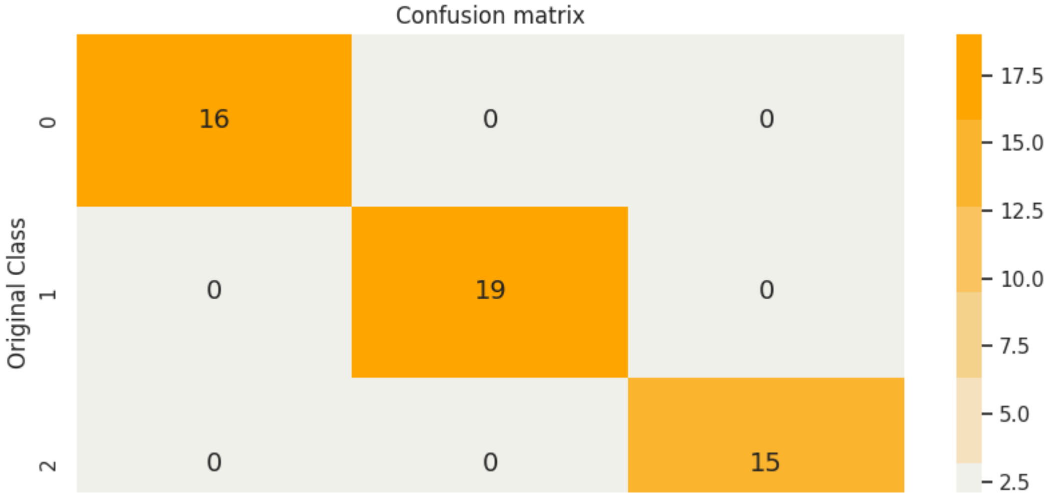
```
# train = 0.667   test = 0.333
seed = 100
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.333,random_state=0)
print("accuracy score for models with train set = 0.667 and test set = 0.333 ")
model_evaluations(X_train, y_train, X_test, y_test)
```

```
accuracy score for models with train set = 0.667 and test set = 0.333

model: Naive bayes
accuracy:1.0
```



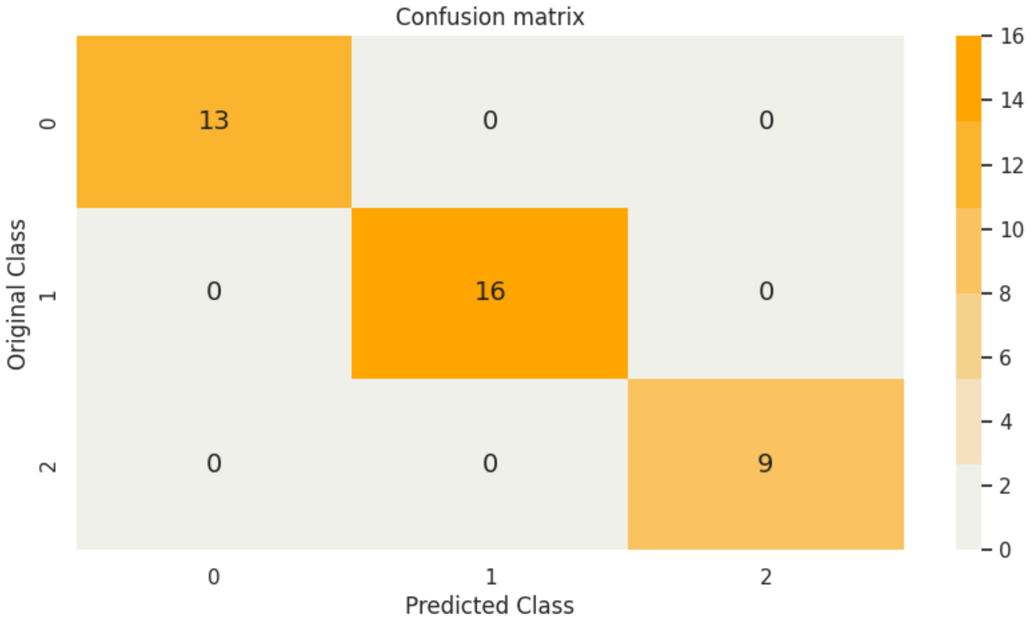
```
model: k nearest neighbors
accuracy:1.0
```



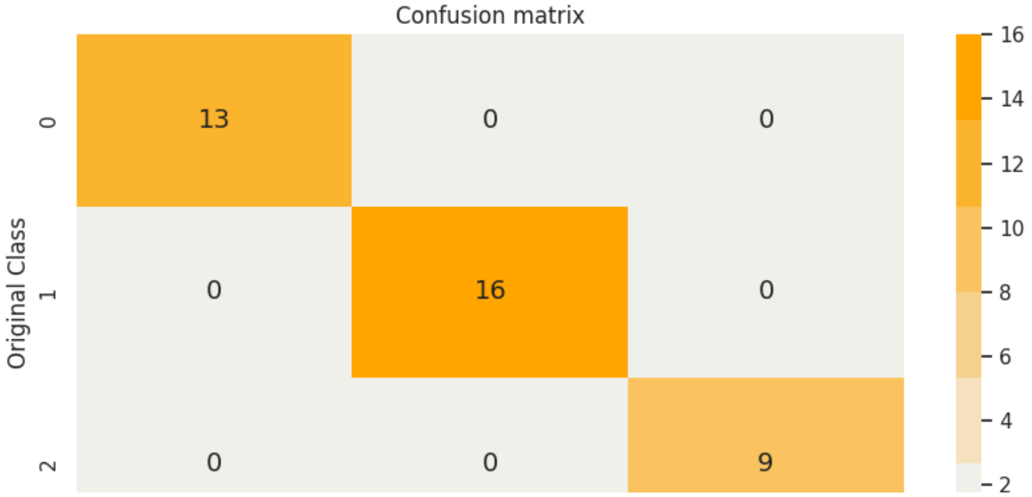
```
# Training set choosing method
#  holdout method
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=0)
print("accuracy score for models with train set = 0.75 and test set = 0.25 ")
model_evaluations(X_train, y_train, X_test, y_test)
```

```
accuracy score for models with train set = 0.75 and test set = 0.25

model: Naive bayes
accuracy:1.0
```



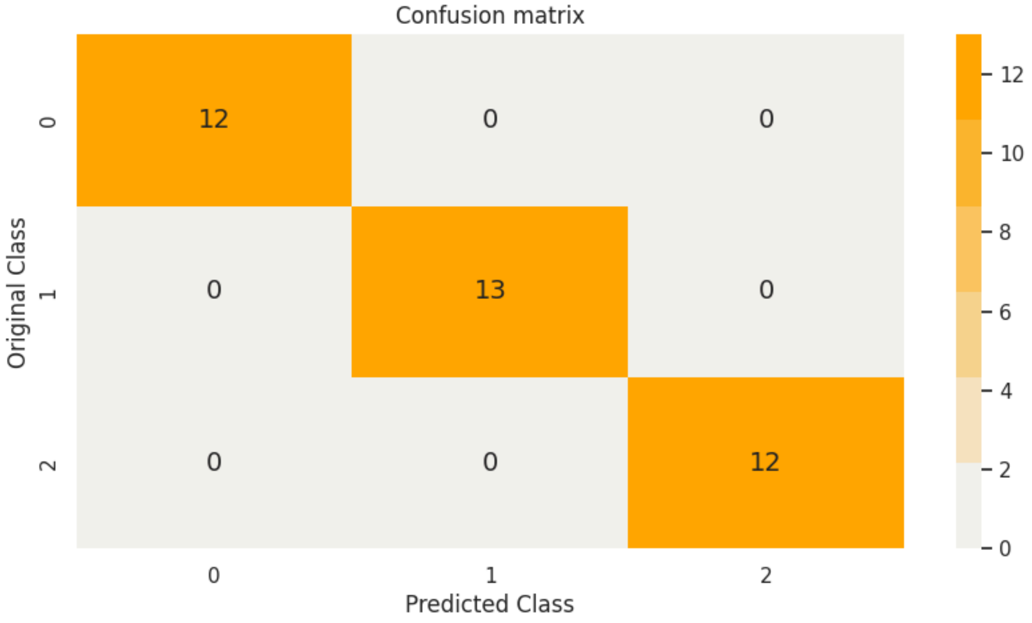
```
model: k nearest neighbors
accuracy:1.0
```



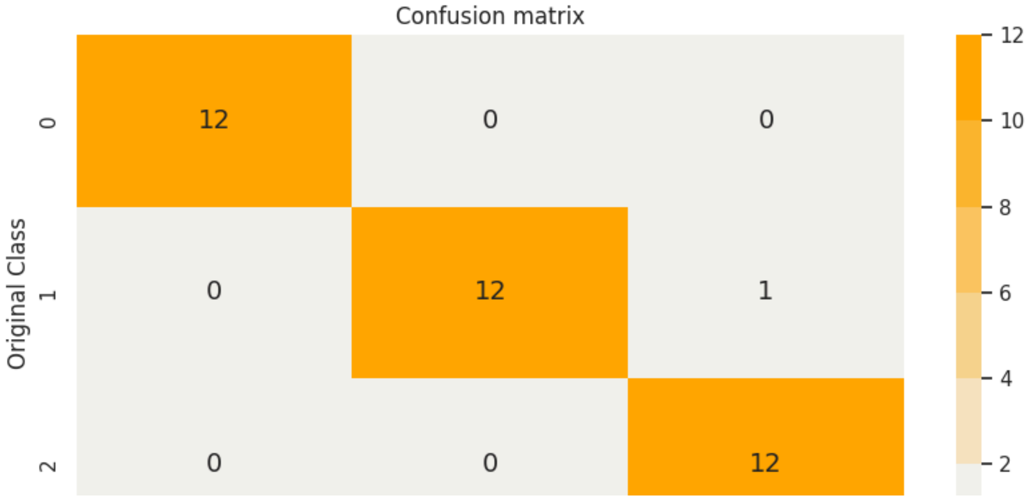
```
# random subsampling
import random
def plot_c(C):
    sns.set(font_scale=1)
    plt.figure(figsize=(10,5))
    labels = [0,1,2]
    # representing A in heatmap format
    cmap1=sns.light_palette("orange")
    sns.heatmap(C, annot=True, cmap=cmap1, fmt=".0f", xticklabels=labels, yticklabels=labels,annot_kws={"size":14})
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Confusion matrix")
    plt.show()
acc1, acc2, acc3 = list(),list(),list()
cf1, cf2, cf3 = [[0,0,0],[0,0,0],[0,0,0]],[[0,0,0],[0,0,0],[0,0,0]],[[0,0,0],[0,0,0],[0,0,0]]
for _ in range(10):
    rd = random.randint(0,1000)
    X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.75,random_state=rd)
    gb = GaussianNB()
    knn = KNeighborsClassifier(round(math.sqrt(X_train.shape[0])))
    dt = DecisionTreeClassifier()
    gb.fit(X_train,y_train)
    knn.fit(X_train,y_train)
    dt.fit(X_train,y_train)
    y_pred = gb.predict(X_test)
    acc1.append(accuracy_score(y_test,y_pred))
    cm = confusion_matrix(y_test,y_pred)
    cf1 =  [[cf1[k][j] + cm[k][j]  for j in range(3)] for k in range(3)]
    y_pred = knn.predict(X_test)
    acc2.append(accuracy_score(y_test,y_pred))
    cm = confusion_matrix(y_test,y_pred)
    cf2 =  [[cf2[k][j] + cm[k][j]  for j in range(3)] for k in range(3)]
    y_pred = dt.predict(X_test)
    acc3.append(accuracy_score(y_test,y_pred))
    cm = confusion_matrix(y_test,y_pred)
    cf3 =  [[cf3[k][j] + cm[k][j]  for j in range(3)] for k in range(3)]
print("accuracy score for models with train set = 0.75 and test set = 0.25 ")
print(f"\nmodel: Naive bayes \naccuracy:{sum(acc1)/10}")
cf1 = [[round(cf1[k][j]/10)  for j in range(3)] for k in range(3)]
plot_c(cf1)
print(f"\n\nmodel: k nearest neighbors \naccuracy:{sum(acc2)/10}")
cf2 = [[round(cf2[k][j]/10)  for j in range(3)] for k in range(3)]
plot_c(cf2)
print(f"\n\nmodel:decision tree \naccuracy:{sum(acc3)/10}")
cf3 = [[round(cf3[k][j]/10)  for j in range(3)] for k in range(3)]
plot_c(cf3)
```

```
accuracy score for models with train set = 0.75 and test set = 0.25

model: Naive bayes
accuracy:0.994736842105263
```



```
model: k nearest neighbors
accuracy:0.9763157894736842
```



```
# cross validation
from sklearn.model_selection import cross_val_score

DT = cross_val_score(DecisionTreeClassifier(), X,y )
print("DecisionTree :",DT.mean())

KNN = cross_val_score(KNeighborsClassifier(), X,y )
print("KNeighborsClassifier :",KNN.mean())

NB = cross_val_score(GaussianNB(), X,y)
print("GaussianNB : ",NB.mean())
```

```
DecisionTree : 0.9
KNeighborsClassifier : 0.8733333333333333
GaussianNB :  0.9933333333333334
```

```
# Data is scaled to standard format.
df.describe()
```

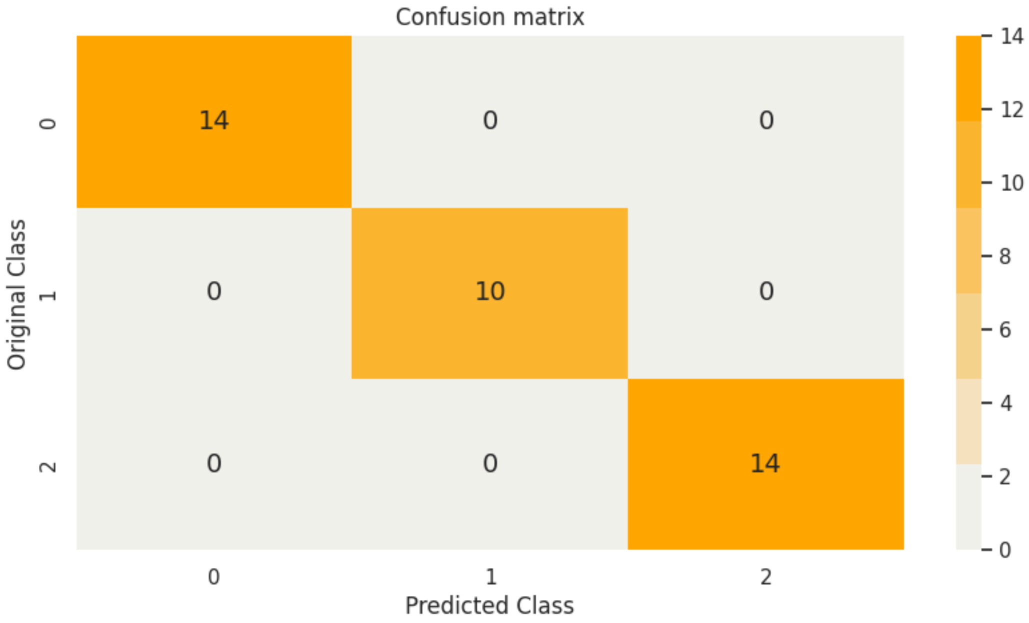
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

```
# need standardization
from sklearn.preprocessing import StandardScaler
X = df.iloc[:,:-1].values
y = df.iloc[:,-1].values
ss = StandardScaler()
X = ss.fit_transform(X)
```

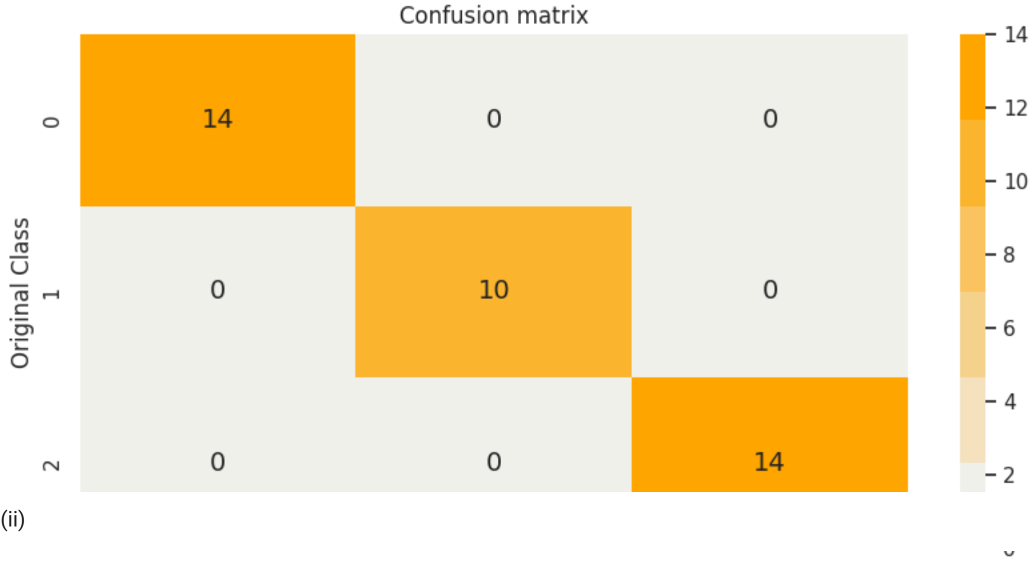
```
seed = 100
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=seed)
print("accuracy score for models with train set = 0.75 and test set = 0.25 and all the data is standardized")
model_evaluations(X_train, y_train, X_test, y_test)
```

```
accuracy score for models with train set = 0.75 and test set = 0.25 and all the data is standardized

model: Naive bayes
accuracy:1.0
```

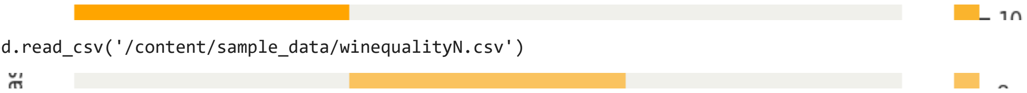


```
model: k nearest neighbors
accuracy:1.0
```



Lab 5(ii)

```
def plot_confusion_matrix(test_y, predict_y):
    C = confusion_matrix(test_y, predict_y)
    sns.set(font_scale=1)
    plt.figure(figsize=(10,5))
    labels = [3,4,5,6,7,8,9]
    # representing A in heatmap format
    cmap1=sns.light_palette("orange")
    sns.heatmap(C, annot=True, cmap=cmap1, fmt=".0f", xticklabels=labels, yticklabels=labels,annot_kws={"size":14})
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Confusion matrix")
    plt.show()
```



```
df = pd.read_csv('/content/sample_data/winequalityN.csv')
```

```
df.head()
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6

```
df.isna().sum().sum()
```

```
38
```

```
df.isna().sum()
```

	0
type	0
fixed acidity	10
volatile acidity	8
citric acid	3
residual sugar	2
chlorides	2
free sulfur dioxide	0
total sulfur dioxide	0
density	0
pH	9
sulphates	4
alcohol	0
quality	0

dtype: int64

```
df.shape
```

```
(6497, 13)
```

```
df.dropna(inplace=True)
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['type'] = le.fit_transform(df['type'])
```

```
df.head()
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	1	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	1	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6

```
df['quality'].unique()
```

```
array([6, 5, 7, 8, 4, 3, 9])
```

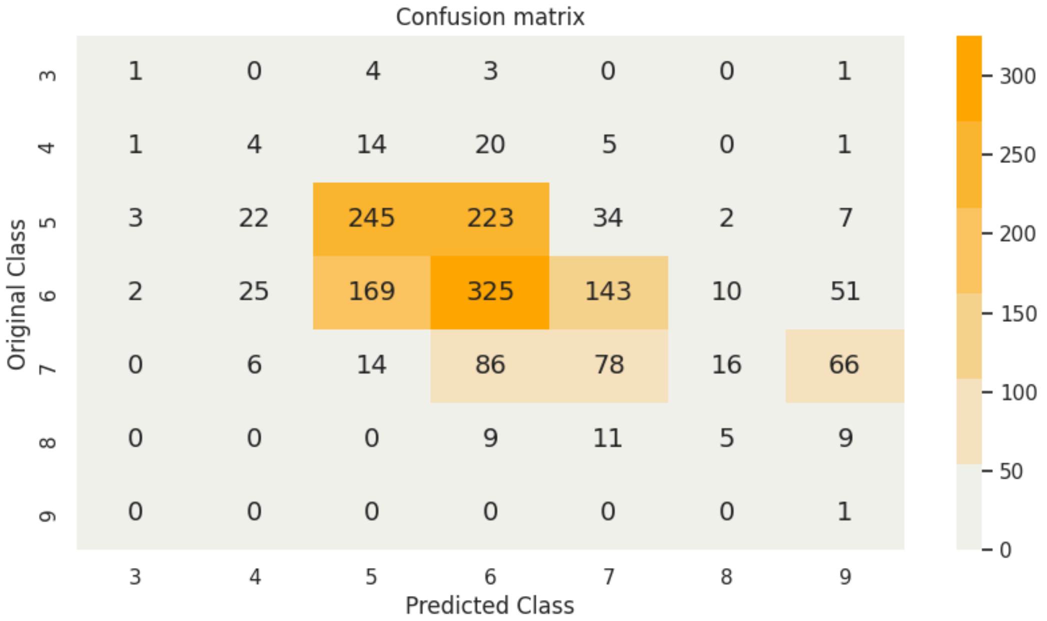
```
X = df.iloc[:,:-1].values
y = df.iloc[:,-1].values
```

```
def model_evaluations(X_train,y_train,X_test,y_test):
    gb = GaussianNB()
    knn = KNeighborsClassifier(round(math.sqrt(X_train.shape[0])))
    dt = DecisionTreeClassifier()
    gb.fit(X_train,y_train)
    knn.fit(X_train,y_train)
    dt.fit(X_train,y_train)
    y_pred = gb.predict(X_test)
    print(f"\nmodel: Naive bayes \naccuracy:{accuracy_score(y_test,y_pred)}")
    plot_confusion_matrix(y_test,y_pred)
    y_pred = knn.predict(X_test)
    print(f"\n\nmodel: k nearest neighbors\naccuracy:{accuracy_score(y_test,y_pred)}")
    plot_confusion_matrix(y_test,y_pred)
    y_pred = dt.predict(X_test)
    print(f"\n\nmodel:decision tree \naccuracy:{accuracy_score(y_test,y_pred)}")
    plot_confusion_matrix(y_test,y_pred)
```

```
seed = 100
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=0)
print("accuracy score for models with train set = 0.75 and test set = 0.25 ")
model_evaluations(X_train, y_train, X_test, y_test)
```

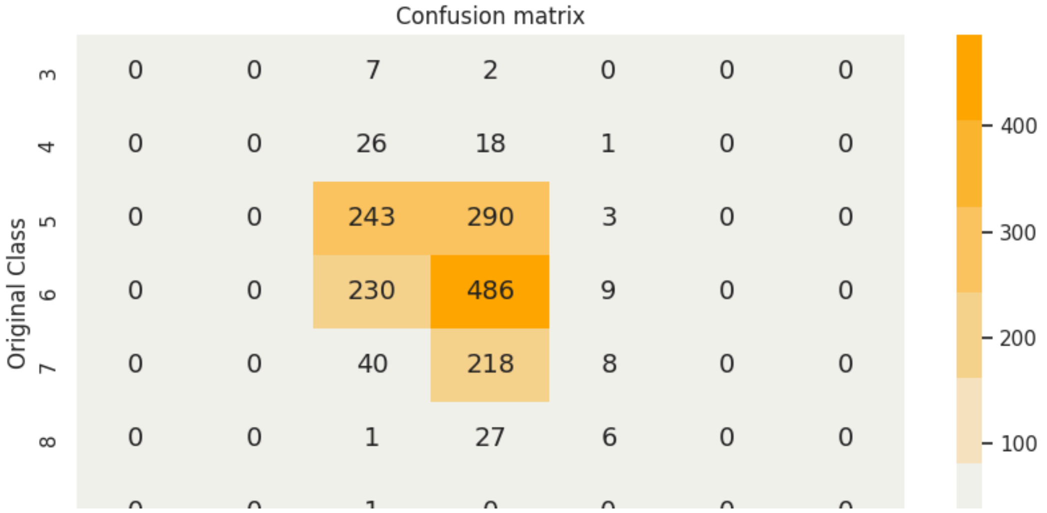
```
accuracy score for models with train set = 0.75 and test set = 0.25

model: Naive bayes
accuracy:0.40779702970297027
```


Confusion matrix

	3	4	5	6	7	8	9
3	1	0	4	3	0	0	1
4	1	4	14	20	5	0	1
5	3	22	245	223	34	2	7
6	2	25	169	325	143	10	51
7	0	6	14	86	78	16	66
8	0	0	0	9	11	5	9
9	0	0	0	0	0	0	1

```
model: k nearest neighbors
accuracy:0.4560643564356436
```

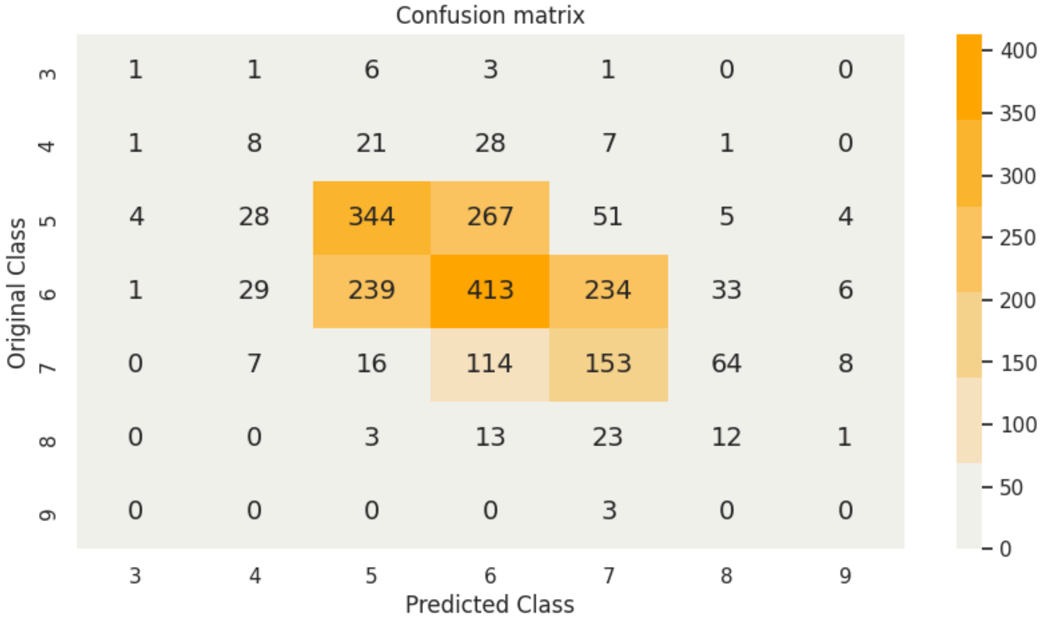

Confusion matrix

	3	4	5	6	7	8	9
3	0	0	7	2	0	0	0
4	0	0	26	18	1	0	0
5	0	0	243	290	3	0	0
6	0	0	230	486	9	0	0
7	0	0	40	218	8	0	0
8	0	0	1	27	6	0	0
	0	0	1	0	0	0	0

```
seed = 100
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.333,random_state=0)
print("accuracy score for models with train set = 0.667 and test set = 0.333 ")
model_evaluations(X_train, y_train, X_test, y_test)
```

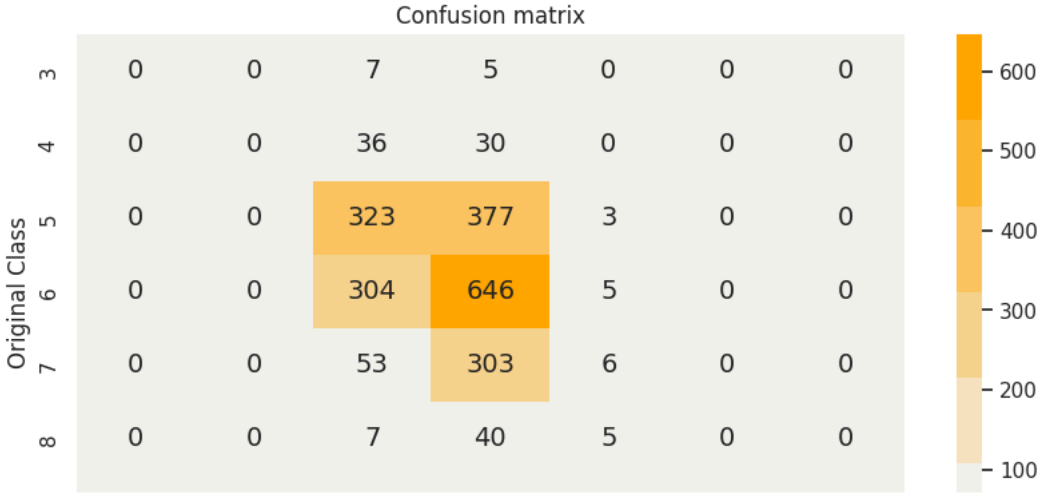
```
accuracy score for models with train set = 0.667 and test set = 0.333

model: Naive bayes
accuracy:0.4324198792382722
```



	3	4	5	6	7	8	9
3	1	1	6	3	1	0	0
4	1	8	21	28	7	1	0
5	4	28	344	267	51	5	4
6	1	29	239	413	234	33	6
7	0	7	16	114	153	64	8
8	0	0	3	13	23	12	1
9	0	0	0	0	3	0	0

```
model: k nearest neighbors
accuracy:0.4528564793311658
```

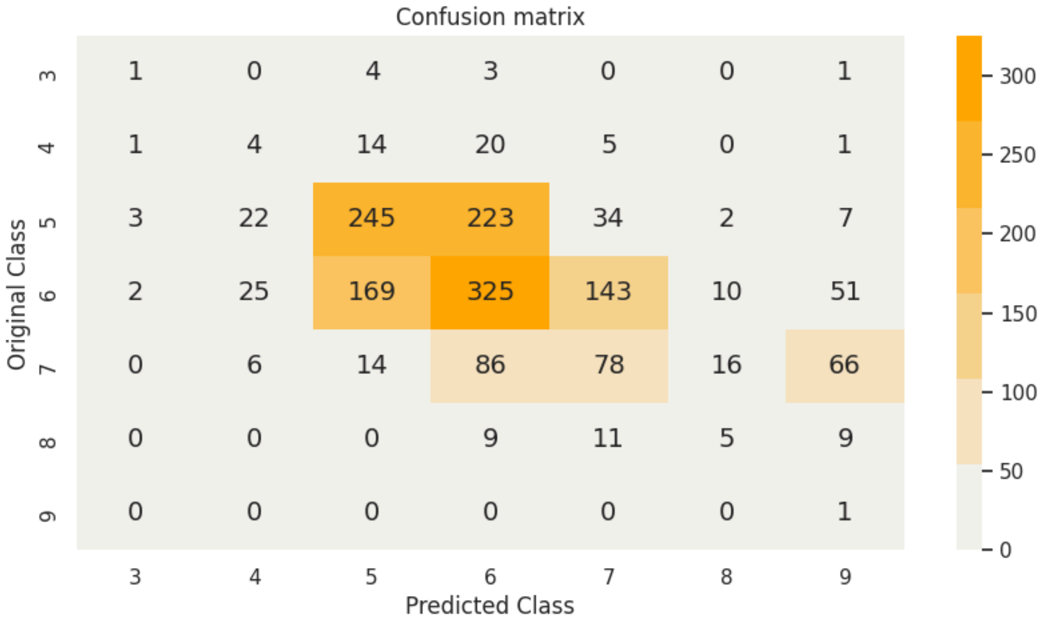


3	0	0	7	5	0	0	0
4	0	0	36	30	0	0	0
5	0	0	323	377	3	0	0
6	0	0	304	646	5	0	0
7	0	0	53	303	6	0	0
8	0	0	7	40	5	0	0

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=0)
print("accuracy score for models with train set = 0.75 and test set = 0.25 ")
model_evaluations(X_train, y_train, X_test, y_test)
```

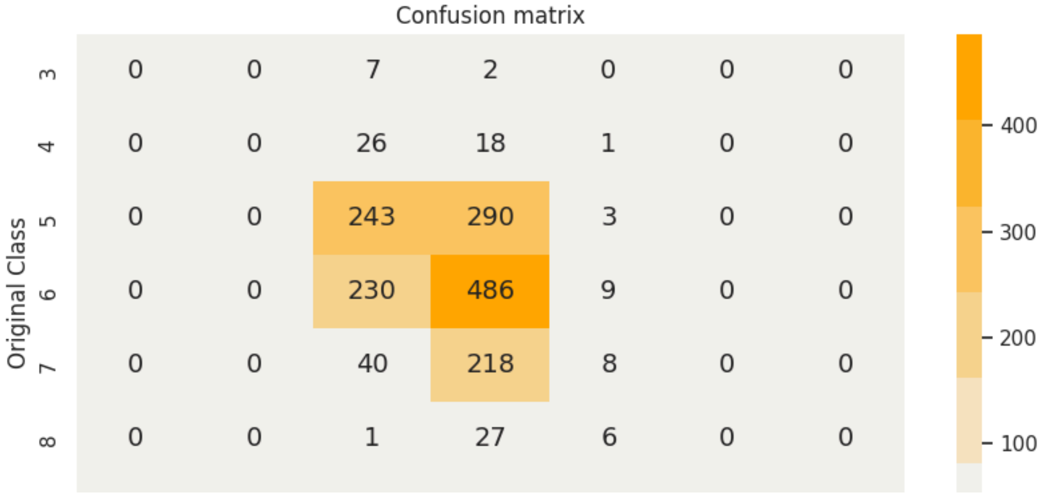
```
accuracy score for models with train set = 0.75 and test set = 0.25

model: Naive bayes
accuracy:0.40779702970297027
```



	3	4	5	6	7	8	9
3	1	0	4	3	0	0	1
4	1	4	14	20	5	0	1
5	3	22	245	223	34	2	7
6	2	25	169	325	143	10	51
7	0	6	14	86	78	16	66
8	0	0	0	9	11	5	9
9	0	0	0	0	0	0	1

```
model: k nearest neighbors
accuracy:0.4560643564356436
```



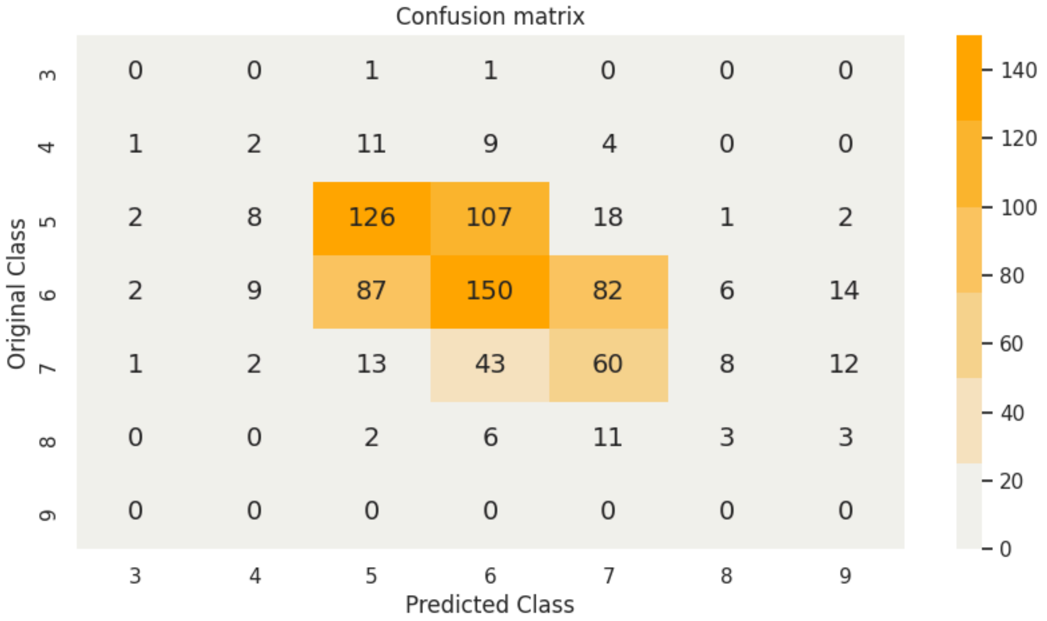
3	0	0	7	2	0	0	0
4	0	0	26	18	1	0	0
5	0	0	243	290	3	0	0
6	0	0	230	486	9	0	0
7	0	0	40	218	8	0	0
8	0	0	1	27	6	0	0

```
import random
def plot_c(C):
    sns.set(font_scale=1)
    plt.figure(figsize=(10,5))
    labels = [3,4,5,6,7,8,9]
    # representing A in heatmap format
    cmap1=sns.light_palette("orange")
    sns.heatmap(C, annot=True, cmap=cmap1, fmt=".0f", xticklabels=labels, yticklabels=labels,annot_kws={"size":14})
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Confusion matrix")
    plt.show()
acc1, acc2, acc3 = list(),list(),list()
cf1, cf2, cf3 =np.zeros((7,7),dtype=np.int64).tolist(),np.zeros((7,7),dtype=np.int64).tolist(),np.zeros((7,7),dtype=np.int64).tolist(
for _ in range(5):
    rd = random.randint(0,1000)
    X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.75,random_state=rd)
    gb = GaussianNB()
    knn = KNeighborsClassifier(round(math.sqrt(X_train.shape[0])))
    dt = DecisionTreeClassifier()
    gb.fit(X_train,y_train)
    knn.fit(X_train,y_train)
    dt.fit(X_train,y_train)
    y_pred = gb.predict(X_test)
    acc1.append(accuracy_score(y_test,y_pred))
    cm = confusion_matrix(y_test,y_pred)
    cf1 =  [[cf1[k][j] + cm[k][j]  for j in range(7)] for k in range(7)]
    y_pred = knn.predict(X_test)
    acc2.append(accuracy_score(y_test,y_pred))
```

```
        cm = confusion_matrix(y_test,y_pred)
        cf2 =  [[cf2[k][j] + cm[k][j]  for j in range(7)] for k in range(7)]
        y_pred = dt.predict(X_test)
        acc3.append(accuracy_score(y_test,y_pred))
        cm = confusion_matrix(y_test,y_pred)
        cf3 =  [[cf3[k][j] + cm[k][j]  for j in range(7)] for k in range(7)]
    print("accuracy score for models with train set = 0.75 and test set = 0.25 ")
    print(f"\nmodel: Naive bayes \naccuracy:{sum(acc1)/10}")
    cf1 = [[round(cf1[k][j]/10)  for j in range(7)] for k in range(7)]
    plot_c(cf1)
    print(f"\n\nmodel: k nearest neighbors \naccuracy:{sum(acc2)/10}")
    cf2 = [[round(cf2[k][j]/10)  for j in range(7)] for k in range(7)]
    plot_c(cf2)
    print(f"\n\nmodel:decision tree \naccuracy:{sum(acc3)/10}")
    cf3 = [[round(cf3[k][j]/10)  for j in range(7)] for k in range(7)]
    plot_c(cf3)
```

```
accuracy score for models with train set = 0.75 and test set = 0.25

model: Naive bayes
accuracy:0.2110148514851485
```



	3	4	5	6	7	8	9
3	0	0	1	1	0	0	0
4	1	2	11	9	4	0	0
5	2	8	126	107	18	1	2
6	2	9	87	150	82	6	14
7	1	2	13	43	60	8	12
8	0	0	2	6	11	3	3
9	0	0	0	0	0	0	0