Practical 5

Q5. Use Naive bayes, K-nearest, and Decision tree classification algorithms and build classifiers. Divide the data set into training and test set. Compare the accuracy of the different classifiers under the following situations:

1. a) Training set = 75% Test set = 25%
   b) Training set = 66.6% (2/3rd of total), Test set = 33.3%
2. Training set is chosen by
   i) hold out method
   ii) Random subsampling
   iii) Cross-Validation.
   Compare the accuracy of the classifiers obtained.
3. Data is scaled to standard format.

Code:

lab 5.py

```
# %%
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('iris.csv')

# %%
df.head()

# %%
df.isna().sum().sum()

# %%
```

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

# %%
df.head()

# %%
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)
```

```
# %% [markdown]
# Different sizes of train and test sets

# %% [markdown]
# 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)

# %% [markdown]
# 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)

# %% [markdown]
# Training set choosing method

# %% [markdown]
# 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)

# %% [markdown]
# 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)

# %% [markdown]
# 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())

# %% [markdown]
# Data is scaled to standard format.

# %%
df.describe()

# %% [markdown]
# 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)
```

lab 5ii.py

```
# %%

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,f1_score
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
import math
import seaborn as sns
from IPython.display import display

# %%
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('winequalityN.csv')

# %%
df.head()

# %%
df.isna().sum().sum()

# %%
df.isna().sum()

# %%
```

```
df.shape

# %% [markdown]
# dropping 38 coulmn wont make much difference

# %%
df.dropna(inplace=True)

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

# %%
df.head()

# %%
df['quality'].unique()

# %%
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)

# %% [markdown]
# Different sizes of train and test sets
```

```
# %% [markdown]
# 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)

# %% [markdown]
# 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)

# %% [markdown]
# Training set choosing method

# %% [markdown]
# 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)

# %% [markdown]
# random subsampling

# %%
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(),n
p.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)

# %% [markdown]
# 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())

# %% [markdown]
# Data is scaled to standard format.

# %%
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 and
all the data is standardized")
model_evaluations(X_train, y_train, X_test, y_test)
```

Code with output:

lab 5.ipynb

```
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
```

[1] ✓ 4.7s Python

```
df = pd.read_csv('iris.csv')
```

[2] ✓ 0.7s Python

```
df.head()
```

[3] ✓ 0.8s Python

	Sepal_length	Sepal_width	Petal_length	Petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

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

[4] ✓ 0.2s Python

```
0
```

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

[5] ✓ 0.2s Python

```
df.head()
```

[6] ✓ 0.2s Python

	Sepal_length	Sepal_width	Petal_length	Petal_width	class
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

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

[7] ✓ 0.1s Python

```python
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()
```

[8] ✓ 0.6s Python

```python
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)
```

[9] ✓ 0.1s Python

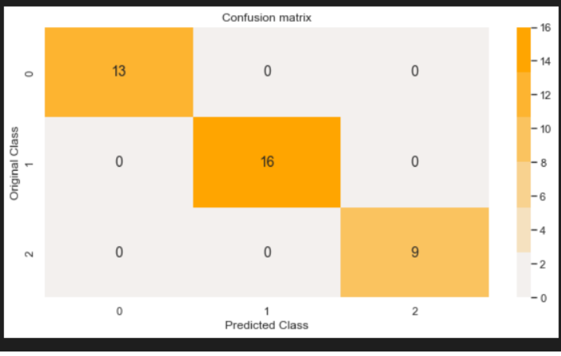
Different sizes of train and test sets

train = 0.75 test = 0.25

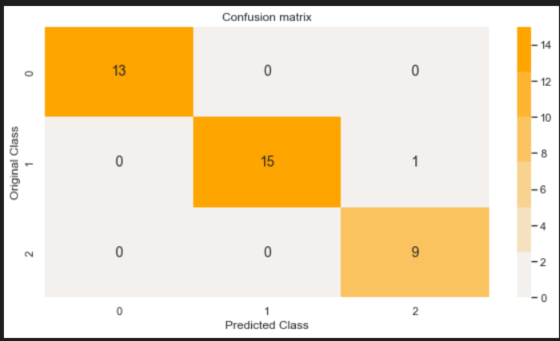
```python
seed = 100
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=0)
print("accuracy score for models with train set = 0.75 and test set = 0.25 ")
model_evaluations(X_train, y_train, X_test, y_test)
```

[10] ✓ 1.3s Python

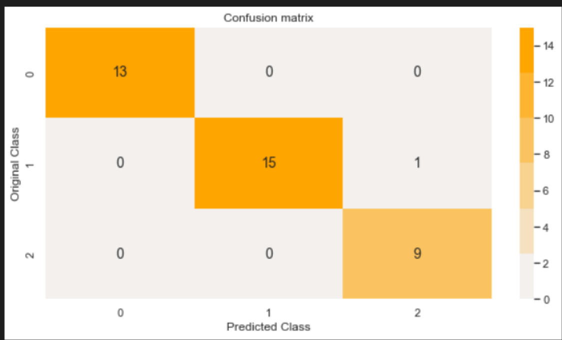
```
accuracy score for models with train set = 0.75 and test set = 0.25

model: Naive bayes
accuracy:1.0
```



```
model: k nearest neighbors
accuracy:0.9736842105263158
```




```
model:decision tree
accuracy:0.9736842105263158
```

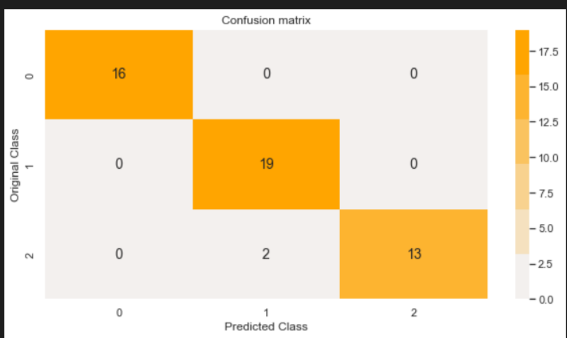


train = 0.667 test = 0.333

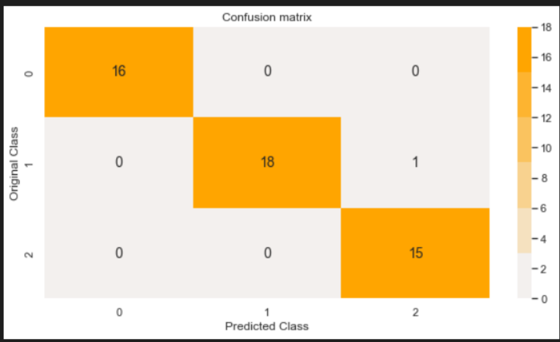
```python
seed = 100
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.333,random_state=0)
print("accuracy score for models with train set = 0.667 and test set = 0.333 ")
model_evaluations(X_train, y_train, X_test, y_test)
```

[11] ✓ 1.3s Python

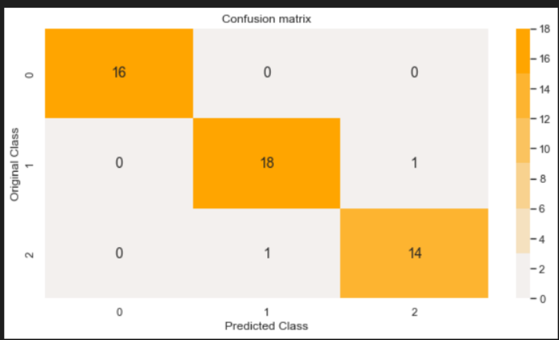
```
accuracy score for models with train set = 0.667 and test set = 0.333

model: Naive bayes
accuracy:0.96
```



```
model: k nearest neighbors
accuracy:0.98
```




```
model:decision tree
accuracy:0.96
```



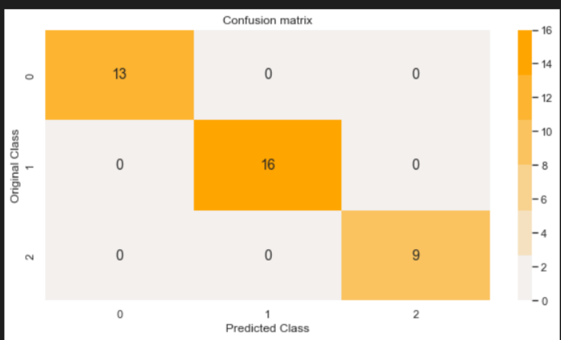
Training set choosing method

holdout method

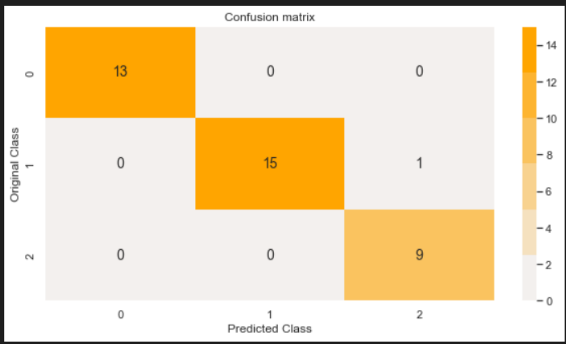
```python
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=0)
print("accuracy score for models with train set = 0.75 and test set = 0.25 ")
model_evaluations(X_train, y_train, X_test, y_test)
```

[29] 0.6s Python

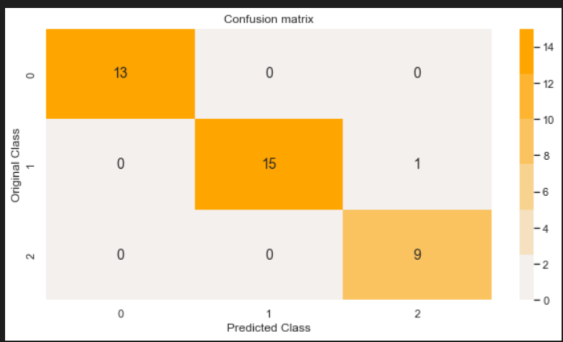
```
accuracy score for models with train set = 0.75 and test set = 0.25

model: Naive bayes
accuracy:1.0
```



```
model: k nearest neighbors
accuracy:0.9736842105263158
```




```
model:decision tree
accuracy:0.9736842105263158
```

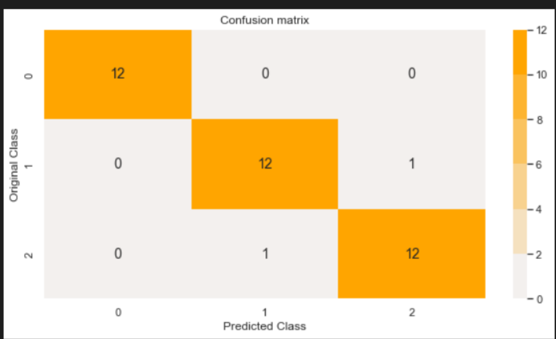


random subsampling

```
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)
```

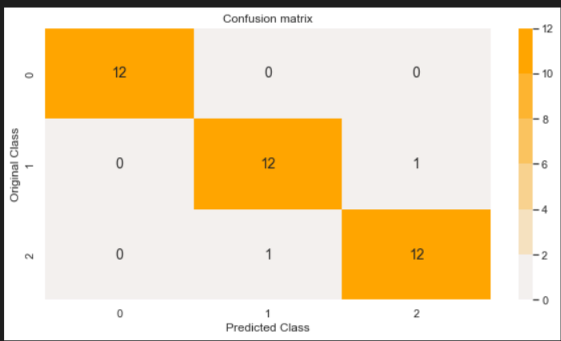
[47] ✓ 1.6s Python

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

model: Naive bayes
accuracy:0.9526315789473683
```



```
model: k nearest neighbors
accuracy:0.968421052631579
```


```
model:decision tree
accuracy:0.9473684210526315
```



cross validation

+ Code + Markdown

```python
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())
```

[48] ✓ 0.9s Python

```
DecisionTree : 0.9666666666666668
KNeighborsClassifier : 0.9733333333333334
GaussianNB :  0.9533333333333334
```

Data is scaled to standard format.

```
df.describe()
```

[49] ✓ 0.8s Python

	Sepal_length	Sepal_width	Petal_length	Petal_width	class
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667	1.000000
std	0.828066	0.433594	1.764420	0.763161	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

need standardization

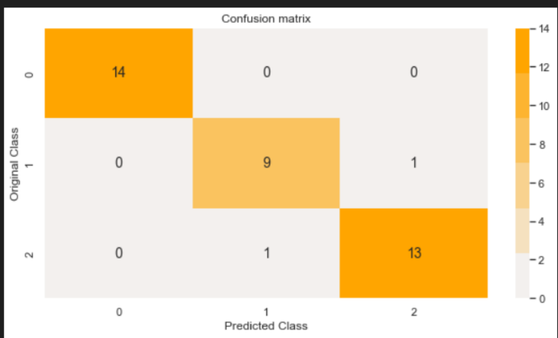
```
from sklearn.preprocessing import StandardScaler
X = df.iloc[:,:-1].values
y = df.iloc[:,-1].values
ss = StandardScaler()
X = ss.fit_transform(X)
```

[50] ✓ 0.7s Python

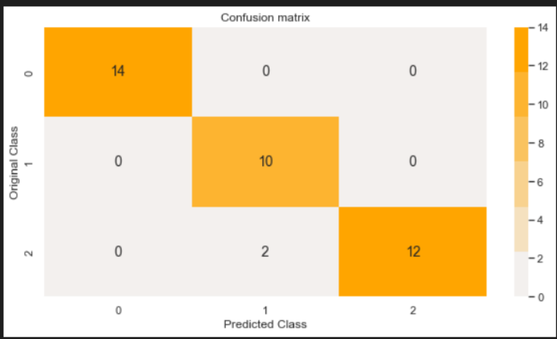
```
seed = 100
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=seed)
print("accuracy score for models with train set = 0.75 and test set = 0.25 and all the data is standardized")
model_evaluations(X_train, y_train, X_test, y_test)
```

[51] ✓ 1.1s Python

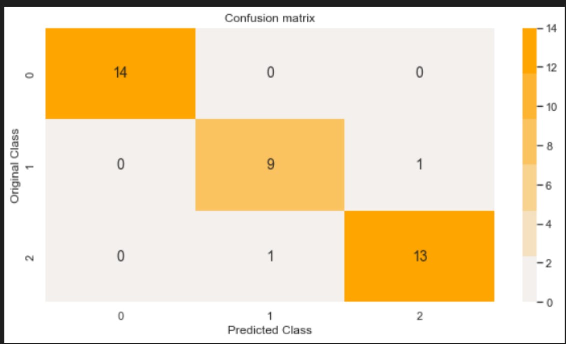
```
accuracy score for models with train set = 0.75 and test set = 0.25 and all the data is standardized

model: Naive bayes
accuracy:0.9473684210526315
```



```
model: k nearest neighbors
accuracy:0.9473684210526315
```




```
model:decision tree
accuracy:0.9473684210526315
```



lab 5ii.ipynb

```python
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,f1_score
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
import math
import seaborn as sns
from IPython.display import display
```

[244] ✓ 0.1s Python

```python
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()
```

[245] ✓ 0.6s Python

```python
df = pd.read_csv('winequalityN.csv')
```

[246] ✓ 0.8s Python

```python
df.head()
```

[247] ✓ 0.7s Python

	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
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

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

[248] ✓ 0.6s Python

38

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

[249] ✓ 0.6s Python

```
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
```

```python
df.shape
```

[250] ✓ 0.6s Python

```
(6497, 13)
```

dropping 38 coulmn wont make much difference

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

[251] ✓ 0.6s Python

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

[252] ✓ 0.4s Python

```python
df.head()
```

[253] ✓ 0.6s Python

	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
3	1	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	1	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

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

[254] ✓ 0.7s Python

```
array([6, 5, 7, 8, 4, 3, 9], dtype=int64)
```

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

[255] ✓ 0.6s Python

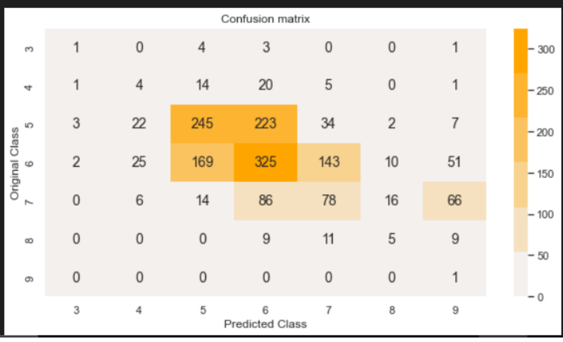
Different sizes of train and test sets

train = 0.75 test = 0.25

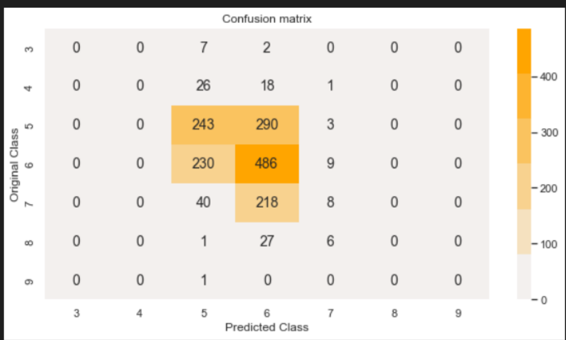
```
seed = 100
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=0)
print("accuracy score for models with train set = 0.75 and test set = 0.25 ")
model_evaluations(X_train, y_train, X_test, y_test)
```

[257] ✓ 1.3s Python

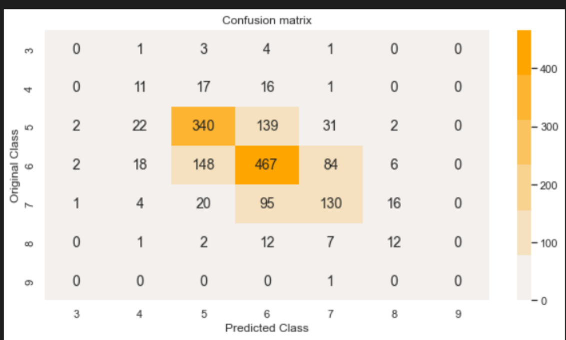
```
accuracy score for models with train set = 0.75 and test set = 0.25

model: Naive bayes
accuracy:0.40779702970297027
```




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




```
model:decision tree
accuracy:0.594059405940594
```

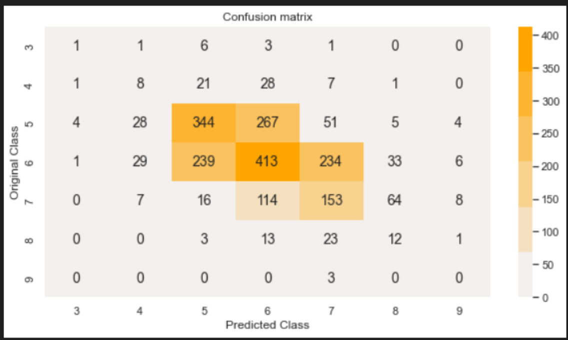


train = 0.667 test = 0.333

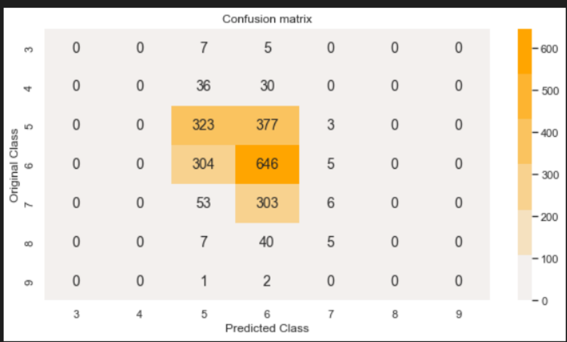
```python
seed = 100
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.333,random_state=0)
print("accuracy score for models with train set = 0.667 and test set = 0.333 ")
model_evaluations(X_train, y_train, X_test, y_test)
```

[258] ✓ 1.6s Python

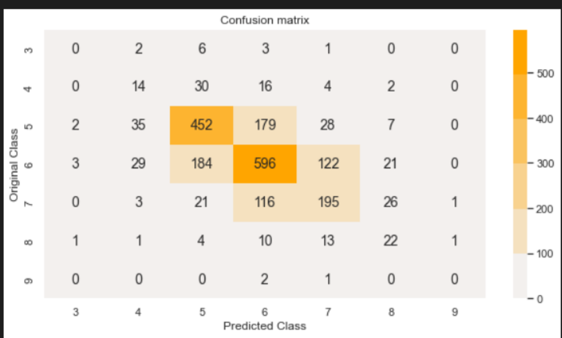
```
accuracy score for models with train set = 0.667 and test set = 0.333

model: Naive bayes
accuracy:0.4324198792382722
```




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




```
model:decision tree
accuracy:0.5940548072457037
```



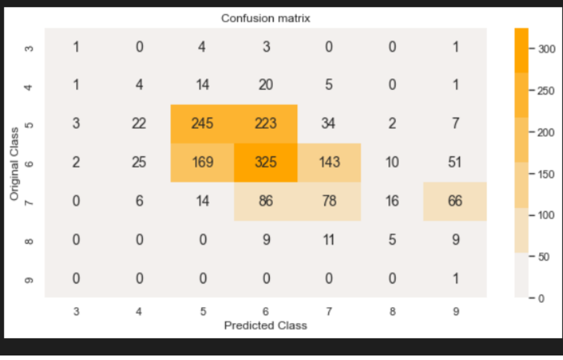
Training set choosing method

holdout method

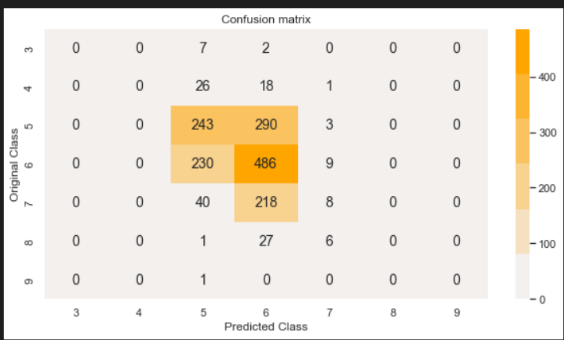
```python
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=0)
print("accuracy score for models with train set = 0.75 and test set = 0.25 ")
model_evaluations(X_train, y_train, X_test, y_test)
```

[259] ✓ 1.3s Python

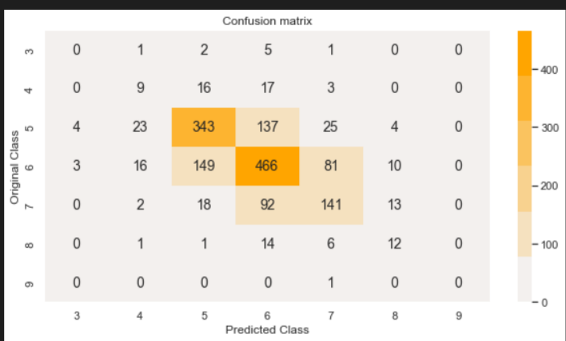
```
accuracy score for models with train set = 0.75 and test set = 0.25

model: Naive bayes
accuracy:0.40779702970297027
```




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




```
model:decision tree
accuracy:0.6008663366336634
```

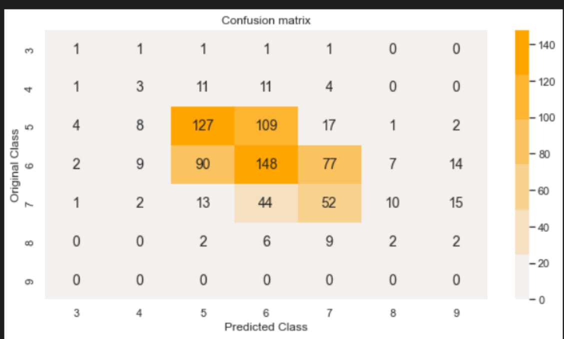


random subsampling

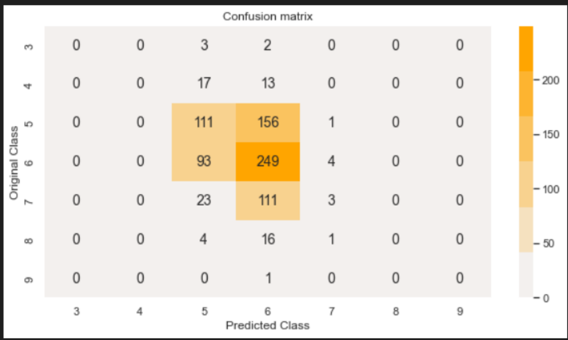
```python
import random
def plot_c(C):
    sns.set(font_scale=1)
    plt.figure(figsize=(10,5))
    labels = [3,4,5,6,7,8,9]
    # representing A in heatmap format
    cmap1=sns.light_palette("orange")
    sns.heatmap(C, annot=True, cmap=cmap1, fmt=".0f", xticklabels=labels, yticklabels=labels,annot_kws={"size":14})
    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)
```

[260] ✓ 2.4s Python

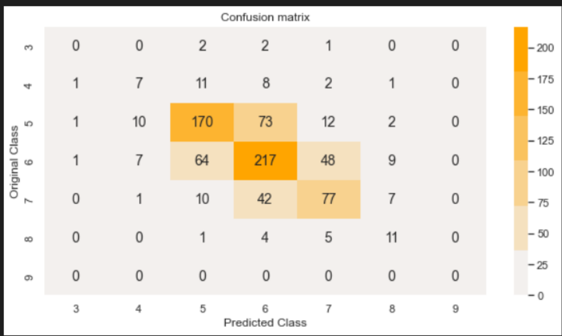
```
accuracy score for models with train set = 0.75 and test set = 0.25

model: Naive bayes
accuracy:0.20618811881188118
```



```
model: k nearest neighbors
accuracy:0.22462871287128716
```




```
model:decision tree
accuracy:0.29870049504950497
```



cross validation

```python
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())
```

[261] ✓ 0.7s Python

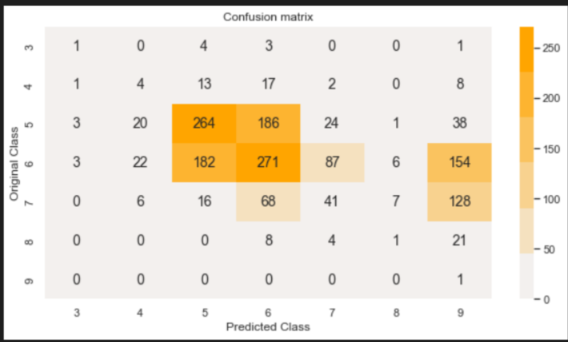
```
DecisionTree : 0.39207497384104456
KNeighborsClassifier : 0.3953235928636933
GaussianNB :  0.3668472053615683
```

Data is scaled to standard format.

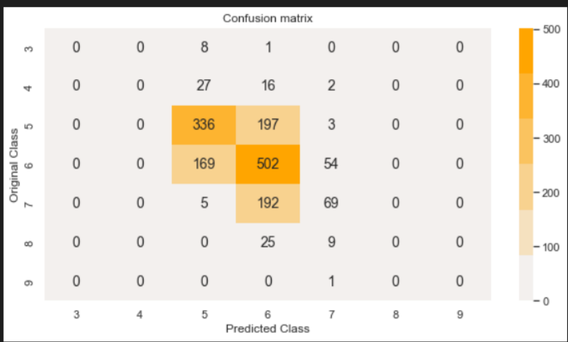
```python
seed = 100
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=0)
print("accuracy score for models with train set = 0.75 and test set = 0.25 and all the data is standardized")
model_evaluations(X_train, y_train, X_test, y_test)
```

265] ✓ 2.4s Python

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

model: Naive bayes
accuracy:0.36076732673267325
```




```
model: k nearest neighbors
accuracy:0.5612623762376238
```




```
model:decision tree
accuracy:0.5928217821782178
```