Machine Learning Assignment -3
Student: Sowjanya Sunkavalli
ID: 700731896

Programming elements:

Classification

Question 1

1. Find the correlation between 'survived' (target column) and 'sex' column for the Titanic use case in class.

In [30]:
```
import numpy as np # Importing numpy as np
import pandas as pd # Importing pandas as pd
import warnings # Importing warnings to exclude warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
train = pd.read_csv('train.csv') # Reading train data set as train dataframe
test = pd.read_csv('test.csv') # Reading test data set as test dataframe
```

In [17]:
```
train['Sex'] = train['Sex'].replace(["female", "male"], [1, 0]) # Replacing Female and Male values with 1 , 0
train['Embarked'] = train['Embarked'].replace(['S','C','Q'],[0,1,2]) # Replacing S, C, QA with 0, 1, 2
```

In [18]:
```
train.fillna(train.mean(),axis=0,inplace=True) # Filling null values with mean
```

In [19]:
```
train['Survived'].corr(train['Sex']) # Correlation of train for Survived and Sex
```

Out[19]: 0.5433513806577552

Imported numpy as np, pandas as pd and Warnings to ignore the warning statements.

Reading train data as train dataframe and test data as test data frame.

Replacing Female and Male values to equivalent 1 , 0 in the Sex column using **replace function**

And also replacing S, C, Q with 0,1, 2 values respectively in Embarked Column.

Filling Null values in train data set with mean values of corresponding columns using **mean**() function

Finding the correlation value between Survived and Sex column using **corr**() function.

a. Do you think we should keep this feature?

In [ ]:
```
# Yes
```

Yes, I believe we should maintain this functionality. We may include this column when providing input to the model because we changed the values of the Sex column from female and male to 1 and 0

accordingly.

2. Do at least two visualizations to describe or show correlations

Showing correlations

In [40]:
```
# showing correlations for trained data set with background style and color
train.corr().style.background_gradient(cmap="Greys")
```

Out[40]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
PassengerId	1.000000	-0.005007	-0.035144	-0.042939	0.033207	-0.057527	-0.001652	0.012658	-0.030479
Survived	-0.005007	1.000000	-0.338481	0.543351	-0.069809	-0.035322	0.081629	0.257307	0.108472
Pclass	-0.035144	-0.338481	1.000000	-0.131900	-0.331339	0.083081	0.018443	-0.549500	0.043714
Sex	-0.042939	0.543351	-0.131900	1.000000	-0.084153	0.114631	0.245489	0.182333	0.118347
Age	0.033207	-0.069809	-0.331339	-0.084153	1.000000	-0.232625	-0.179191	0.091566	0.009464
SibSp	-0.057527	-0.035322	0.083081	0.114631	-0.232625	1.000000	0.414838	0.159651	-0.060591
Parch	-0.001652	0.081629	0.018443	0.245489	-0.179191	0.414838	1.000000	0.216225	-0.079300
Fare	0.012658	0.257307	-0.549500	0.182333	0.091566	0.159651	0.216225	1.000000	0.063396
Embarked	-0.030479	0.108472	0.043714	0.118347	0.009464	-0.060591	-0.079300	0.063396	1.000000

Finding the Correlations of the train data using corr() function and also here we can use style.background_gradient(cmap=”Greys”) to provide the background style and color to the output visulalizations.

3. Implement Naïve Bayes method using scikit-learn library and report the accuracy.

In [21]:
```
X = train.drop(columns = ['Name','Survived','Ticket','Cabin']) # Dropping alphnumeric values
Y = train[['Survived']] # assigning Survived column
```

In [22]:
```
# Splitting dataset into traning and testing data sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
```

In [23]:
```
# Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB() #creating the model
classifier.fit(X_train, Y_train) #Feeding training data to the model

Y_pred = classifier.predict(X_test) # Predicting the test variables

# Summary of the predictions made by the classifier
print(classification_report(Y_test, Y_pred)) # Classification report
print(confusion_matrix(Y_test, Y_pred)) # confusion matrix
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(Y_pred,Y_test)) # Accuracy of the model
```

```
              precision    recall  f1-score   support

           0       0.84      0.82      0.83       110
           1       0.72      0.75      0.74        69

    accuracy                           0.79       179
   macro avg       0.78      0.79      0.78       179
weighted avg       0.80      0.79      0.79       179

[[90 20]
 [17 52]]
accuracy is 0.7932960893854749
```

Dropping Name, Survived, Ticket, Cabin from training data set and assigning Data set to X using **drop ()** function.

Assigning training Data Set containing only Survived column to Y

Splitting data set into training and testing data sets using **train_test_split() method**

Creating the model using Classifier = GaussianNB()

Feeding the training data to the model using **fit(data parameters)** function.

Predicting the test variables using **predict() function**.

Printing Classificationreport using **classification_report()** function.

Printing Confusion Matrox using **confusion_matrix()** function.

Getting Accuracy using **accuracy_score()** function

Question 2

Implement Naïve Bayes method using scikit-learn library.

a. Use the glass dataset available in Link also provided in your assignment

In [24]:
```
data = pd.read_csv('glass.csv') # reading glass data set as data
```

In [25]:
```
X = data.drop(columns=['Type']) # dropping Type coloumn
Y = data['Type'] # assigning type column
```

Reading glass data file using read_csv() method

Dropping Type columns and assigning remaining data set to X.

Assigning Type Column data set to Y.

2. Evaluate the model on testing part using score and classification_report

In [31]:
```
# Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB # importing classifier
classifier = GaussianNB() # assigning classifier with model
classifier.fit(X_train, Y_train) # feeding training data to the model

Y_pred = classifier.predict(X_test) # Predicting the dependant variable
# Summary of the predictions made by the classifier
print(classification_report(Y_test, Y_pred)) # Classification Report
print(confusion_matrix(Y_test, Y_pred)) # Confusion matrix
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(Y_pred,Y_test)) # Finding the Accuracy
```

```
              precision    recall  f1-score   support

           1       0.19      0.44      0.27         9
           2       0.33      0.16      0.21        19
           3       0.33      0.20      0.25         5
           5       0.00      0.00      0.00         2
           6       0.67      1.00      0.80         2
           7       1.00      1.00      1.00         6

    accuracy                           0.37        43
   macro avg       0.42      0.47      0.42        43
weighted avg       0.40      0.37      0.36        43

[[ 4  3  1  0  1  0]
 [14  3  1  1  0  0]
 [ 3  1  1  0  0  0]
 [ 0  2  0  0  0  0]
 [ 0  0  0  0  2  0]
 [ 0  0  0  0  0  6]]
accuracy is 0.37209302325581395
```

Creating the model using Classifier = GaussianNB()

Feeding the training data to the model using **fit(data parameters)** function.

Predicting the test variables using **predict() function**.

Printing Classificationreport using **classification_report()** function.

Printing Confusion Matrox using **confusion_matrix()** function.

Getting Accuracy using **accuracy_score()** function

Implement SVM method using scikit-learn library.

a. Use the glass dataset available in Link also provided in your assignment

In [24]:
```
data = pd.read_csv('glass.csv') # reading glass data set as data
```

In [25]:
```
X = data.drop(columns=['Type']) # dropping Type coloumn
Y = data['Type'] # assigning type column
```

Reading glass data file using read_csv() method

Dropping Type columns and assigning remaining data set to X.

Assigning Type Column data set to Y.

1. Implement linear SVM method using scikit library

In [37]:
```
# Support Vector Machine's
from sklearn.svm import SVC # importing SVC
classifier = SVC() # creating the model
classifier.fit(X_train, Y_train) # feeding model with training dataset
Y_pred = classifier.predict(X_test) # predicting the dependent variable in the test dataset

```

2. Evaluate the model on testing part using score and Classification

In [38]:
```
# Summary of the predictions made by the classifier
print(classification_report(Y_test, Y_pred)) # printing the classification report
print(confusion_matrix(Y_test, Y_pred)) # printing the confusion matrix
# Accuracy score
from sklearn.metrics import accuracy_score # importing the accuracy_score
print('accuracy is',accuracy_score(Y_pred,Y_test)) # printing the accuracy of predicted values with true values
```

```
              precision    recall  f1-score   support

           1       0.21      1.00      0.35         9
           2       0.00      0.00      0.00        19
           3       0.00      0.00      0.00         5
           5       0.00      0.00      0.00         2
           6       0.00      0.00      0.00         2
           7       0.00      0.00      0.00         6

    accuracy                           0.21        43
   macro avg       0.03      0.17      0.06        43
weighted avg       0.04      0.21      0.07        43

[[ 9  0  0  0  0  0]
 [19  0  0  0  0  0]
 [ 5  0  0  0  0  0]
 [ 2  0  0  0  0  0]
 [ 2  0  0  0  0  0]
 [ 6  0  0  0  0  0]]
accuracy is 0.20930232558139536
```

Importing svm from sklearn library.

Assiging model svc() to classifier

Feeding the training data to the model using **fit(data parameters)** function.

Predicting the test variables using **predict() function**

Printing Classificationreport using **classification_report()** function.

Printing Confusion Matrox using **confusion_matrix()** function.

Getting Accuracy using **accuracy_score()** function

Do at least two visualizations to describe or show correlations in the Glass Dataset.

Showing Correlations

In [39]:
```
1 data.corr().style.background_gradient(cmap="Greys") # Correlation for trained data set
```

Out[39]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
RI	1.000000	-0.191885	-0.122274	-0.407326	-0.542052	-0.289833	0.810403	-0.000386	0.143010	-0.164237
Na	-0.191885	1.000000	-0.273732	0.156794	-0.069809	-0.266087	-0.275442	0.326603	-0.241346	0.502898
Mg	-0.122274	-0.273732	1.000000	-0.481799	-0.165927	0.005396	-0.443750	-0.492262	0.083060	-0.744993
Al	-0.407326	0.156794	-0.481799	1.000000	-0.005524	0.325958	-0.259592	0.479404	-0.074402	0.598829
Si	-0.542052	-0.069809	-0.165927	-0.005524	1.000000	-0.193331	-0.208732	-0.102151	-0.094201	0.151565
K	-0.289833	-0.266087	0.005396	0.325958	-0.193331	1.000000	-0.317836	-0.042618	-0.007719	-0.010054
Ca	0.810403	-0.275442	-0.443750	-0.259592	-0.208732	-0.317836	1.000000	-0.112841	0.124968	0.000952
Ba	-0.000386	0.326603	-0.492262	0.479404	-0.102151	-0.042618	-0.112841	1.000000	-0.058692	0.575161
Fe	0.143010	-0.241346	0.083060	-0.074402	-0.094201	-0.007719	0.124968	-0.058692	1.000000	-0.188278
Type	-0.164237	0.502898	-0.744993	0.598829	0.151565	-0.010054	0.000952	0.575161	-0.188278	1.000000

Finding the Correlations of the train data using **corr**() function and also here we can use style.background_gradient(cmap=”Greys”) to provide the background style and color to the output visulalizations.

Which algorithm you got better accuracy? Can you justify why?

In [ ]:
```
1 # After analyzing results got from training data with Naives Bayes and SVM model, from the above results of accuracy
2 # We can say Naives Bayes Algorithm is better than SVM
3 # Accuracy of Naive Bayes i.e 37.2% > Accuracy of SVM i.e 20.9%
```

GitHub: https://github.com/sunkavallisowjanya/MachineLearningAssignment3

Video Link: https://youtu.be/7bFOO9fPqQ4