Q1: Find the correlation between ‘survived’ (target column) and ‘sex’ column for the Titanic use case in class.

Explanation:

Input file: train.csv

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S

- Imported the necessary libraries. Importing pandas library as we will be doing data frame manipulation and will be using different pandas functions for analysis.

```
import pandas as pd
import matplotlib.pyplot as plt # visualization
import seaborn as sns # statistical visualizations and aesthetics
```

- Importing train csv file into data frame. We are using read_csv method of Pandas for that.

```
train_df = pd.read_csv('./train.csv')
# Importing test csv file. It can be combined with train if required
test_df = pd.read_csv('./test.csv')
```

- We are using pivoting to analyze and find correlation between different columns. Below code is to get the correlation between **Survived** and **Pclass** columns.

```
# Analysis by pivoting features
print('Correlation between Survived and Pclass')
print(train_df[['Survived','Pclass']].groupby(['Pclass'],as_index=False).mean().sort_values(by='Pclass', ascending=True))
```

- Below pivoting is done to find correlation between Survived and Sex columns. Here GROUPBY on Sex column is used so that mean aggregation function can be used on Survived data categorized by Sex values.

```
print('Correlation between Survived and Sex')
print(train_df[['Sex','Survived']].groupby(['Sex'],as_index=False).mean().sort_values(by='Survived', ascending=False))
```

- For visualizing the correlation we are using Seaborn data visualization library. We are using Faceting which is act of breaking data variables (here Survived column) up across multiple subplots. Afterwards we are combining those subplots into single figure.

```
g = sns.FacetGrid(train_df, col='Survived')
```

- Calling map function on FacetGrid object built above to graph Sex column.

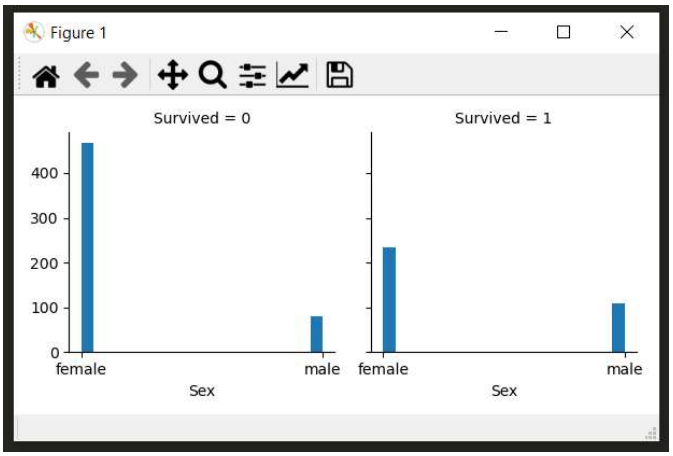
```
g.map(plt.hist, 'Sex', bins=20)
plt.show()
```

OUTPUT:

```
ICP4.1
C:\python\Anaconda\envs\env_full\python.exe "D:/Harshita/PycharmProjects/ClassICP/ICP 4/ICP4.1.py"
Correlation between Survived and Pclass
   Pclass  Survived
0       1  0.629630
1       2  0.472826
2       3  0.242363


Correlation between Survived and Sex
      Sex  Survived
0  female  0.742038
1    male  0.188908
```

GRAPH:



Q1: Do you think we should keep this feature?

Answer: As from the Output we can see correlation is great(> 0.7) between Sex = Female and Survived we should keep this feature for our model.

Q2: Implement Naïve Bayes method using scikit-learn library
Use dataset available in https://umkc.box.com/s/ea6wn1cidukan67t02j60nmp1ljln3kd
Use **train_test_split** to create training and testing part. Evaluate the model on testing part using score and

```
classification_report(y_true, y_pred)
```

EXPLAINATION:

Input file: glass.csv

	A	B	C	D	E	F	G	H	I	J	K
	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type	
	1.52101	13.64	4.49	1.1	71.78	0.06	8.75	0	0	1	
	1.51761	13.89	3.6	1.36	72.73	0.48	7.83	0	0	1	
	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0	0	1	
	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0	0	1	
	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0	0	1	

- Imported the necessary libraries. Importing pandas library as we will be doing data frame manipulation.
- To implement the naive bayes classifier model, we will use Scikit-learn library and will import our **GaussianNB** from **sklearn.naive_bayes**

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

- Importing glass csv input file into dataframe. We are using read_csv method of Pandas for that

```
df = pd.read_csv("glass.csv")
```

- To Correct the dataframes we need to drop unwanted columns and configure another dataframe to use the required column.

```
x_train = df.drop("Type",axis=1)
y_train = df["Type"]
```

- Here we are using the train_test_split function of Scikit-learn to split the data into training and test sets.
- In below code **test_size 0.4** indicates we have used 4% of data for testing and random_state ensures reproducibility

```
x_train, x_test, y_train, y_test= train_test_split(x_train, y_train, test_size=0.4, random_state=40)
```

- Create Gaussian Naive Bayes classifier.

```
classifier = GaussianNB()
```

- Train the model using train sets created above

```
classifier.fit(x_train,y_train)
```

- Using predict method on classifier and pass x_test as a parameter to get prediction output

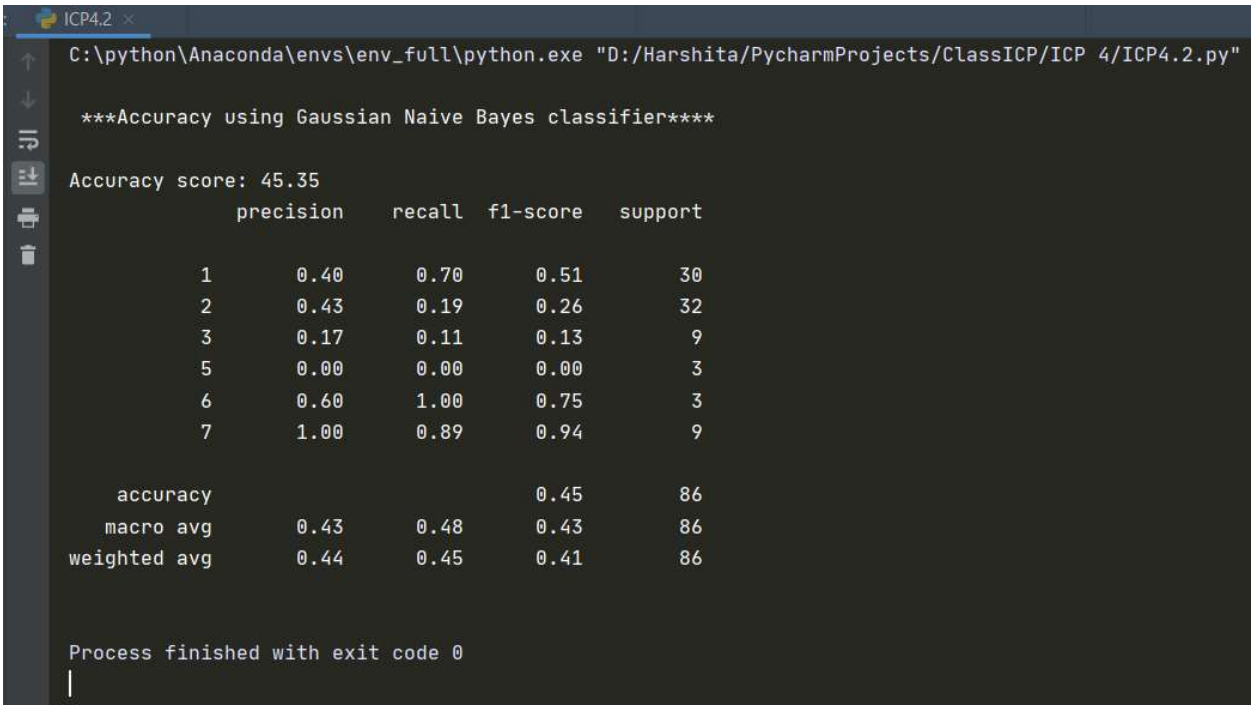
```
y_pred = classifier.predict(x_test)
```

- Model Accuracy: how often is the classifier correct?

```
accuracy = round(accuracy_score(y_test,y_pred)*100, 2)

print('\n', '***Accuracy using Gaussian Naive Bayes classifier****' , '\n')
# Print the accuracy score
print("Accuracy score: " + str(accuracy))
print(classification_report(y_test, y_pred))
```

OUTPUT:



Q3: Implement linear SVM method using scikit library. Use the same dataset above
Use **train_test_split** to create training and testing part. Evaluate the model on testing part using score and

```
classification_report(y_true, y_pred)
```

EXPLAINATION:

Input file: glass.csv

RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
1.52101	13.64	4.49	1.1	71.78	0.06	8.75	0	0	1
1.51761	13.89	3.6	1.36	72.73	0.48	7.83	0	0	1
1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0	0	1
1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0	0	1
1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0	0	1

- Imported the necessary libraries. Importing pandas library as we will be doing data frame manipulation.
- To implement the Support Vector Machines we will use Scikit-learn and will import our SVM

```
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

- Importing glass csv input file into dataframe. We are using read_csv method of Pandas for that

```
df = pd.read_csv("glass.csv")
```

- To Correct the dataframes we need to drop unwanted columns and configure another dataframe to use the required column.

```
x_train = df.drop("Type",axis=1)
y_train = df["Type"]
```

- Here we are using the train_test_split function of Scikit-learn to split the data into training and test sets.
- In below code **test_size 0.4** indicates we have used 4% of data for testing and random_state ensures reproducibility

```
x_train, x_test, y_train, y_test= train_test_split(x_train, y_train, test_size=0.4, random_state=40)
```

- Create SVM classifier

```
svc = svm.SVC(kernel="linear")
```

- Train the model using train sets created above

```
svc.fit(X_train, Y_train)
```

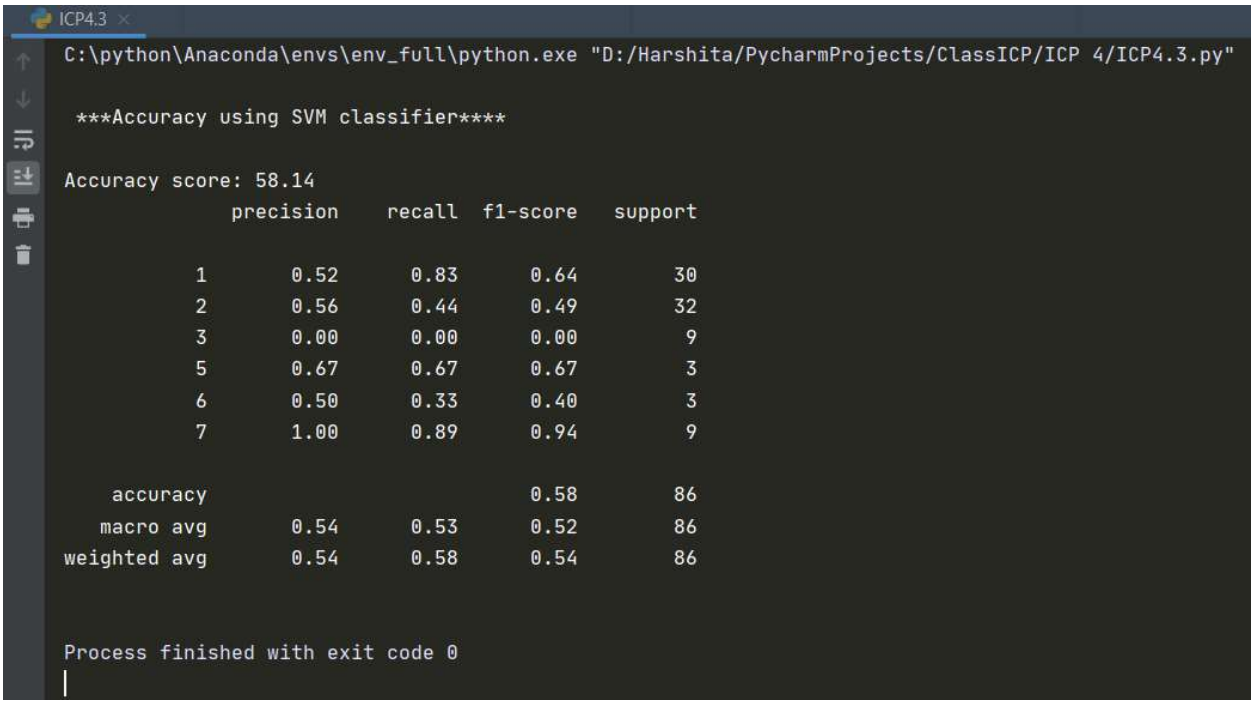
- Using predict method on classifier and pass x_test as a parameter to get prediction output

```
y_pred = classifier.predict(x_test)
```

- Model Accuracy: how often is the classifier correct?

```
print('\n', '***Accuracy using SVM classifier****' , '\n')
# Print the accuracy score
print("Accuracy score: " + str(acc_svc))
print(classification_report(Y_test,Y_pred1))
```

OUTPUT:



Q3: Which algorithm you got better accuracy? Can you justify why?

Answer: As we can see in outputs above, accuracy using Naive Baise is 45.35 while that for SVM is 58.14, clearly SVM has better accuracy.

Justification: - SVM works on constructive linear regression logic that is why it is fast and most accurate.