AI assignment

Submitted By :

Adwayith K S

B210664EC

I. Data Visualisation

1) Write and execute Python scripts to do the followings:

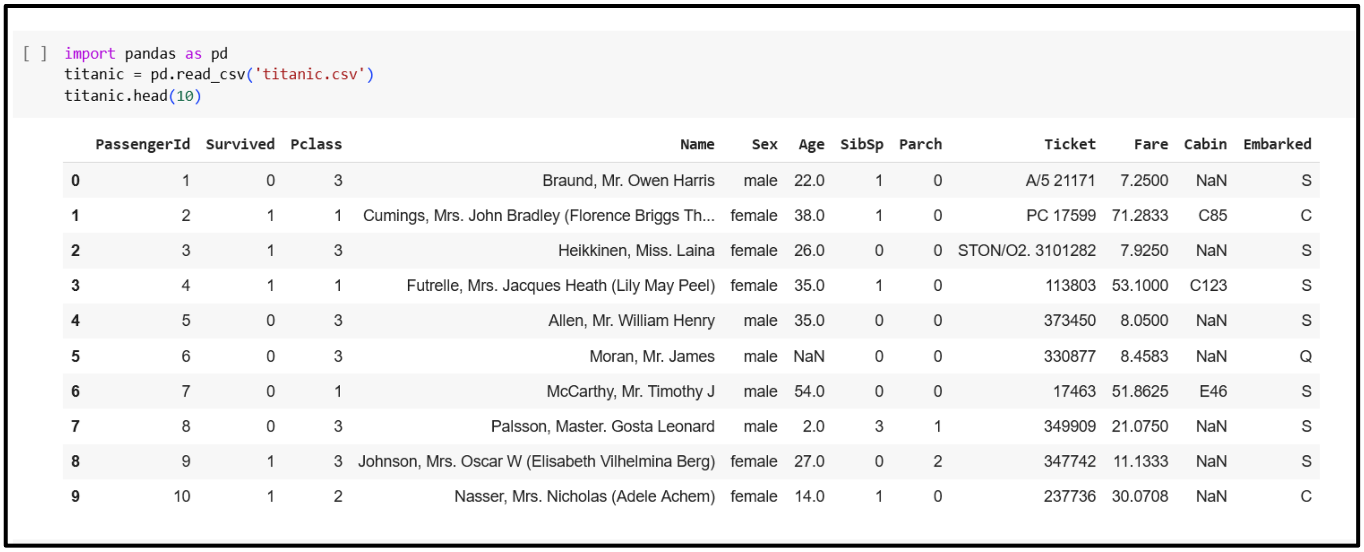
(i) Read CSV file & display information on the dataframe.

Hints: read_csv(), info() method

(ii) Display first 10 rows of the data.

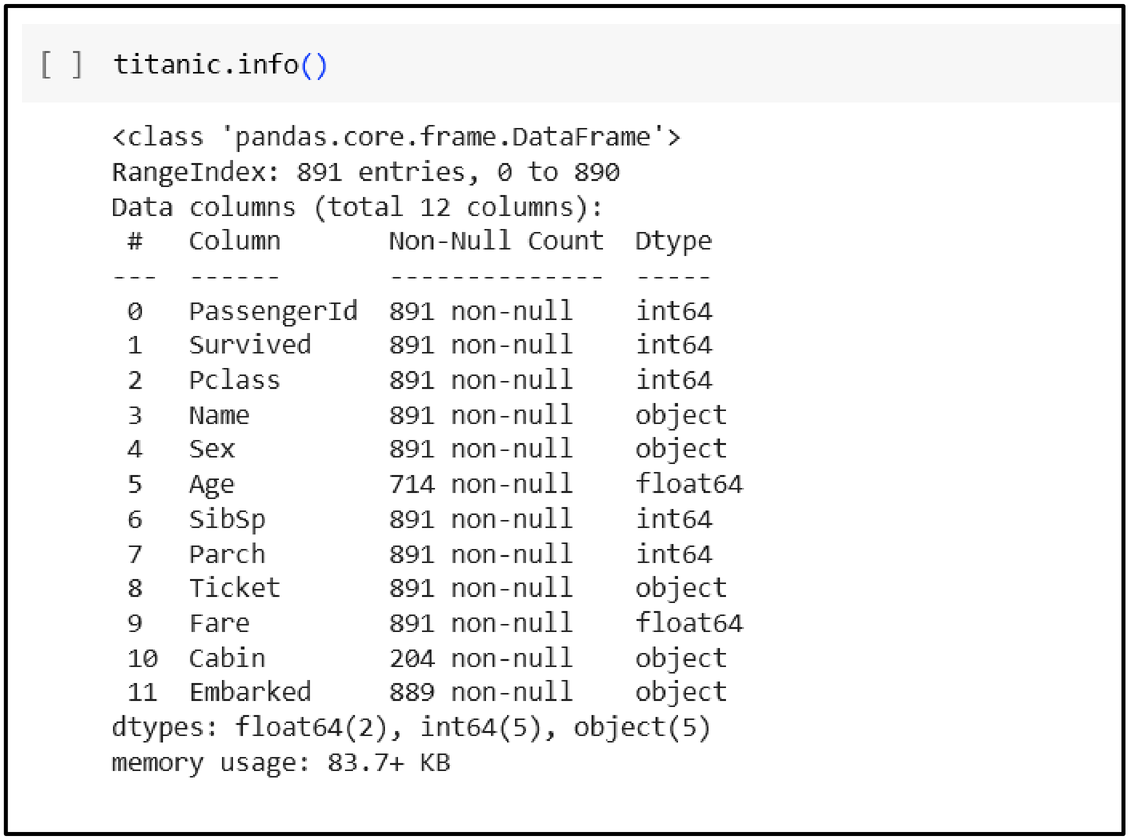
(iii) Display first 5 rows of the data having the given columns only.

'PassengerID', 'Name', 'Age', 'Sex'



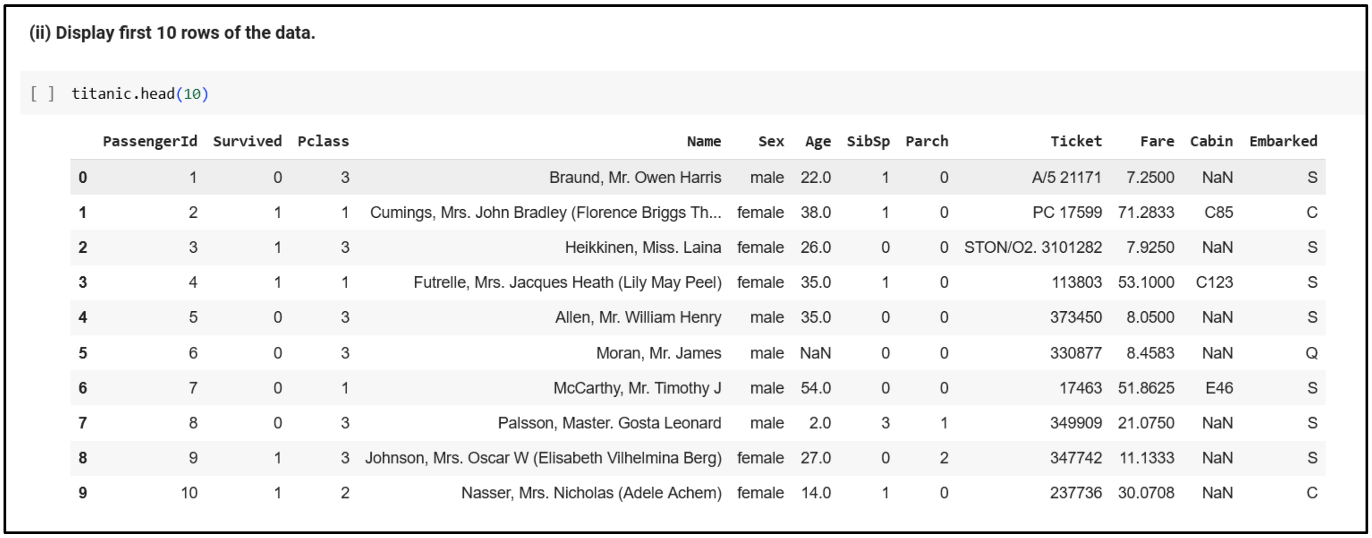
```
[ ] import pandas as pd
    titanic = pd.read_csv('titanic.csv')
    titanic.head(10)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C



```
[ ]  titanic.info()
```

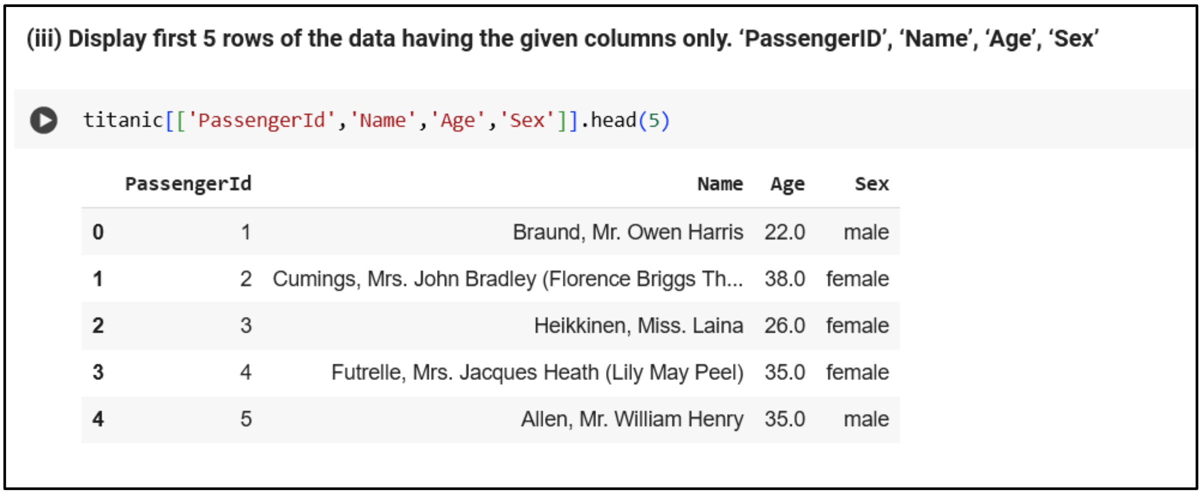
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```



(ii) Display first 10 rows of the data.

```
[ ]  titanic.head(10)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C



(iii) Display first 5 rows of the data having the given columns only. ‘PassengerID’, ‘Name’, ‘Age’, ‘Sex’

```
titanic[['PassengerId','Name','Age','Sex']].head(5)
```

	PassengerId	Name	Age	Sex
0	1	Braund, Mr. Owen Harris	22.0	male
1	2	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	female
2	3	Heikkinen, Miss. Laina	26.0	female
3	4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	female
4	5	Allen, Mr. William Henry	35.0	male

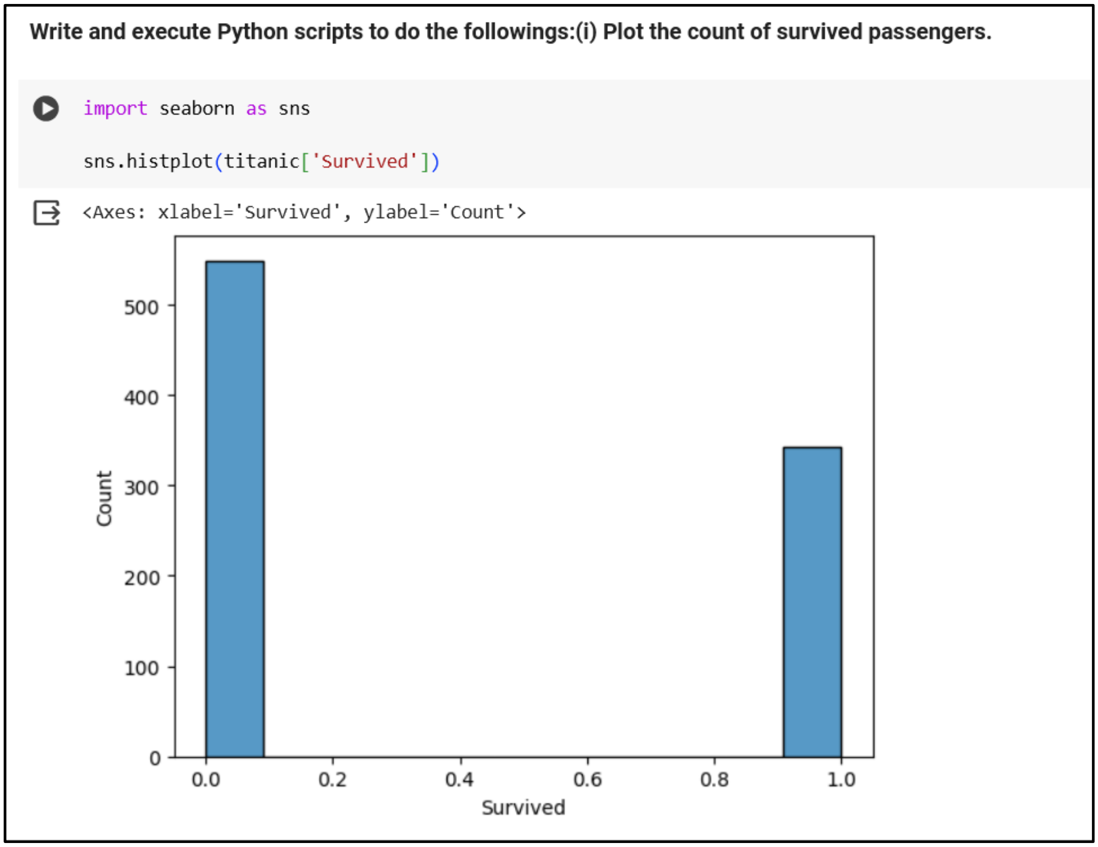
II. Data Analysis

1) Write and execute Python scripts to do the followings:

(i) Plot the count of survived passengers.

(ii) Plot histogram of ‘Age’ column

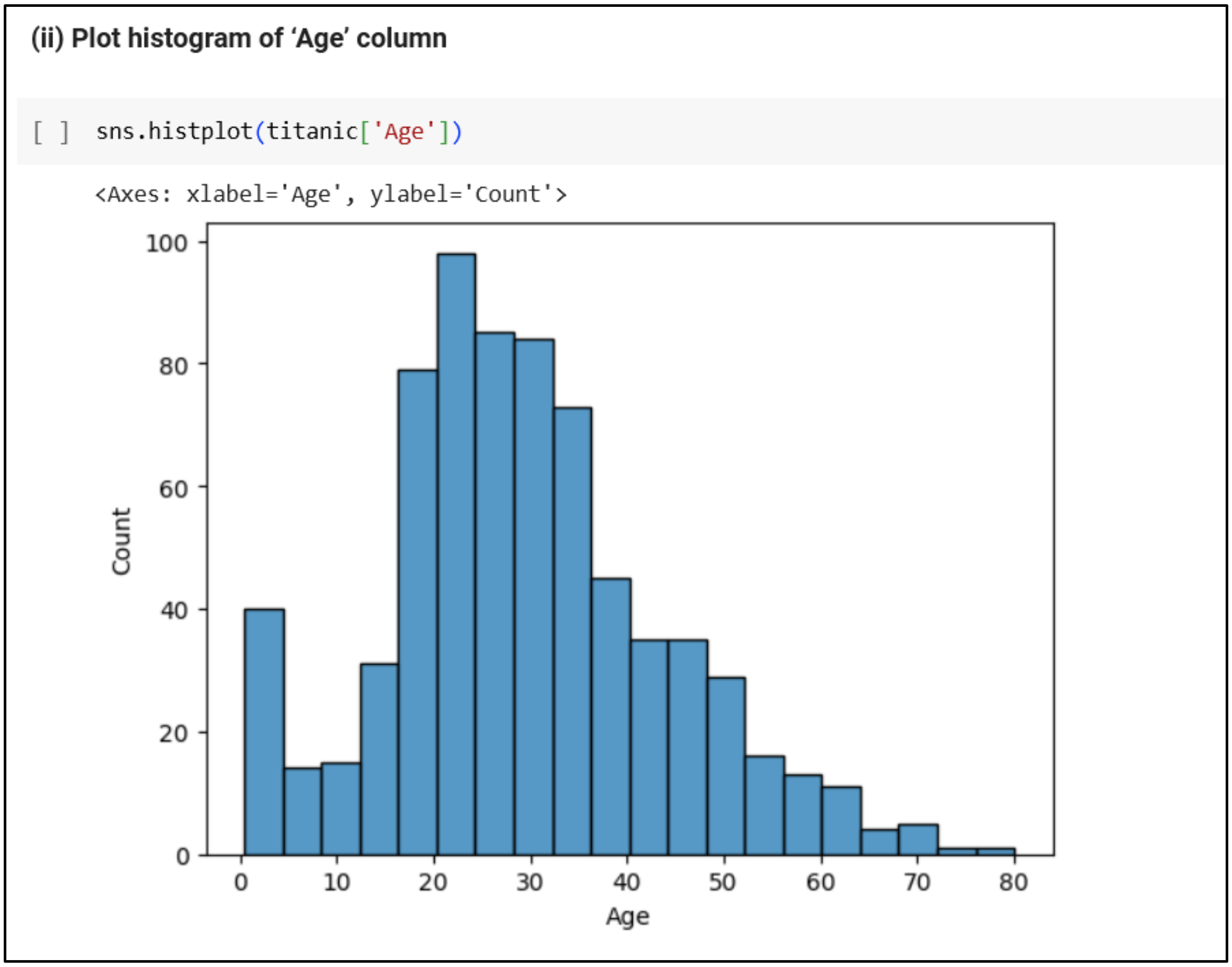
Hints: hist() method



Write and execute Python scripts to do the followings:(i) Plot the count of survived passengers.

```
import seaborn as sns
sns.histplot(titanic['Survived'])
```

```
<Axes: xlabel='Survived', ylabel='Count'>
```



(ii) Plot histogram of 'Age' column

```
[ ]  sns.histplot(titanic['Age'])
```

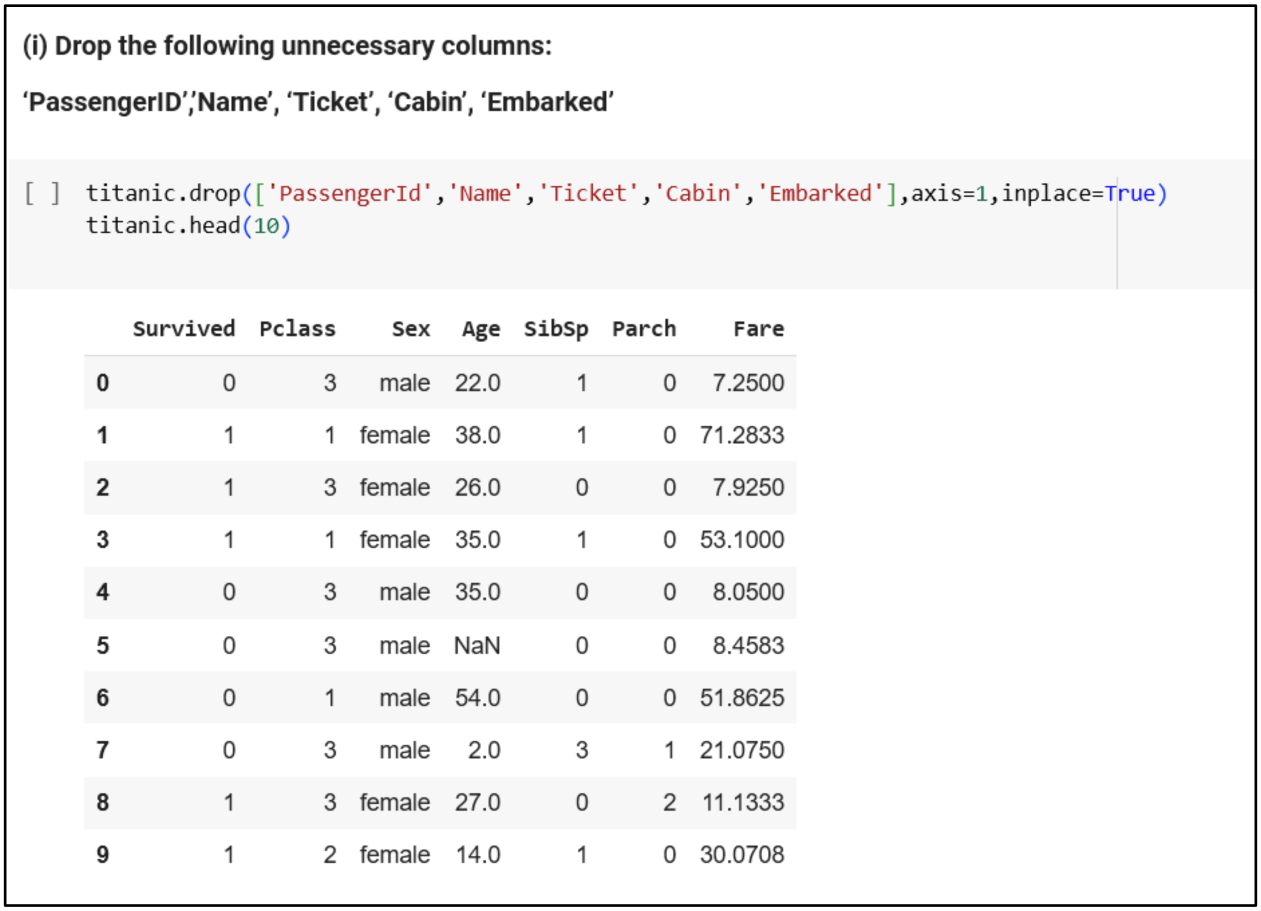
```
<Axes: xlabel='Age', ylabel='Count'>
```

III. Data Wrangling and feature selection

1) Write and execute Python scripts to do the followings:

(i) Drop the following unnecessary columns.

'PassengerID','Name', 'Ticket', 'Cabin', 'Embarked'



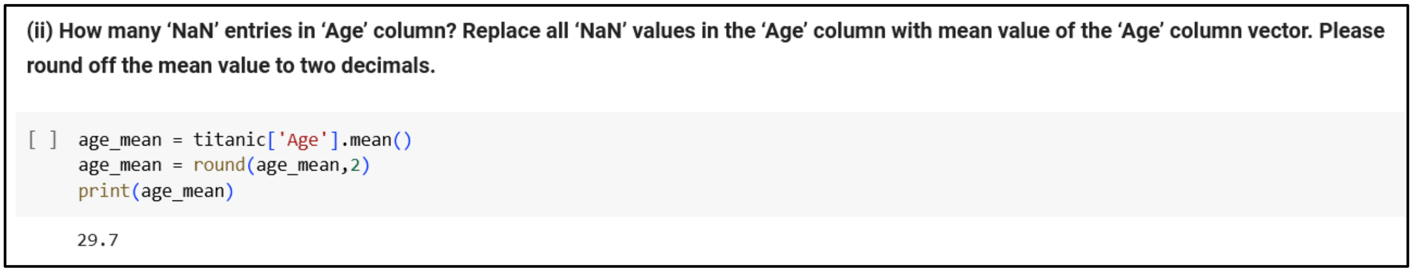
(i) Drop the following unnecessary columns:

‘PassengerID’,‘Name’, ‘Ticket’, ‘Cabin’, ‘Embarked’

```
[ ]  titanic.drop(['PassengerId','Name','Ticket','Cabin','Embarked'],axis=1,inplace=True)
     titanic.head(10)
```

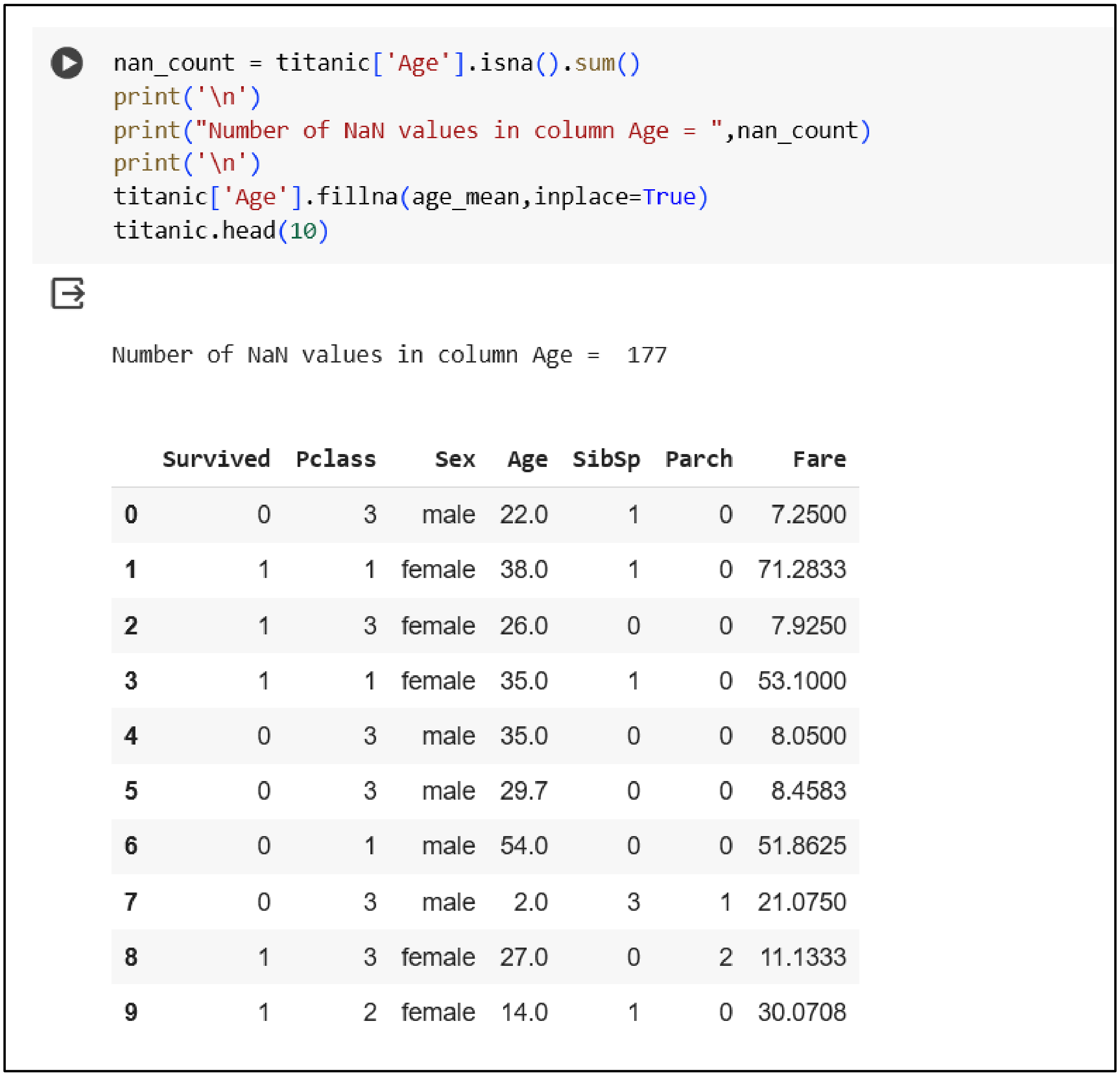
	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	0	3	male	22.0	1	0	7.2500
1	1	1	female	38.0	1	0	71.2833
2	1	3	female	26.0	0	0	7.9250
3	1	1	female	35.0	1	0	53.1000
4	0	3	male	35.0	0	0	8.0500
5	0	3	male	NaN	0	0	8.4583
6	0	1	male	54.0	0	0	51.8625
7	0	3	male	2.0	3	1	21.0750
8	1	3	female	27.0	0	2	11.1333
9	1	2	female	14.0	1	0	30.0708

(ii) How many ‘NaN’ entries in ’Age’ column? Replace all ‘NaN’ values in the ‘Age’ columns with mean value of the ‘Age’ column vector. Please round off the mean value to two decimals.



(ii) How many ‘NaN’ entries in ‘Age’ column? Replace all ‘NaN’ values in the ‘Age’ column with mean value of the ‘Age’ column vector. Please round off the mean value to two decimals.

```
[ ]  age_mean = titanic['Age'].mean()
     age_mean = round(age_mean,2)
     print(age_mean)

     29.7
```

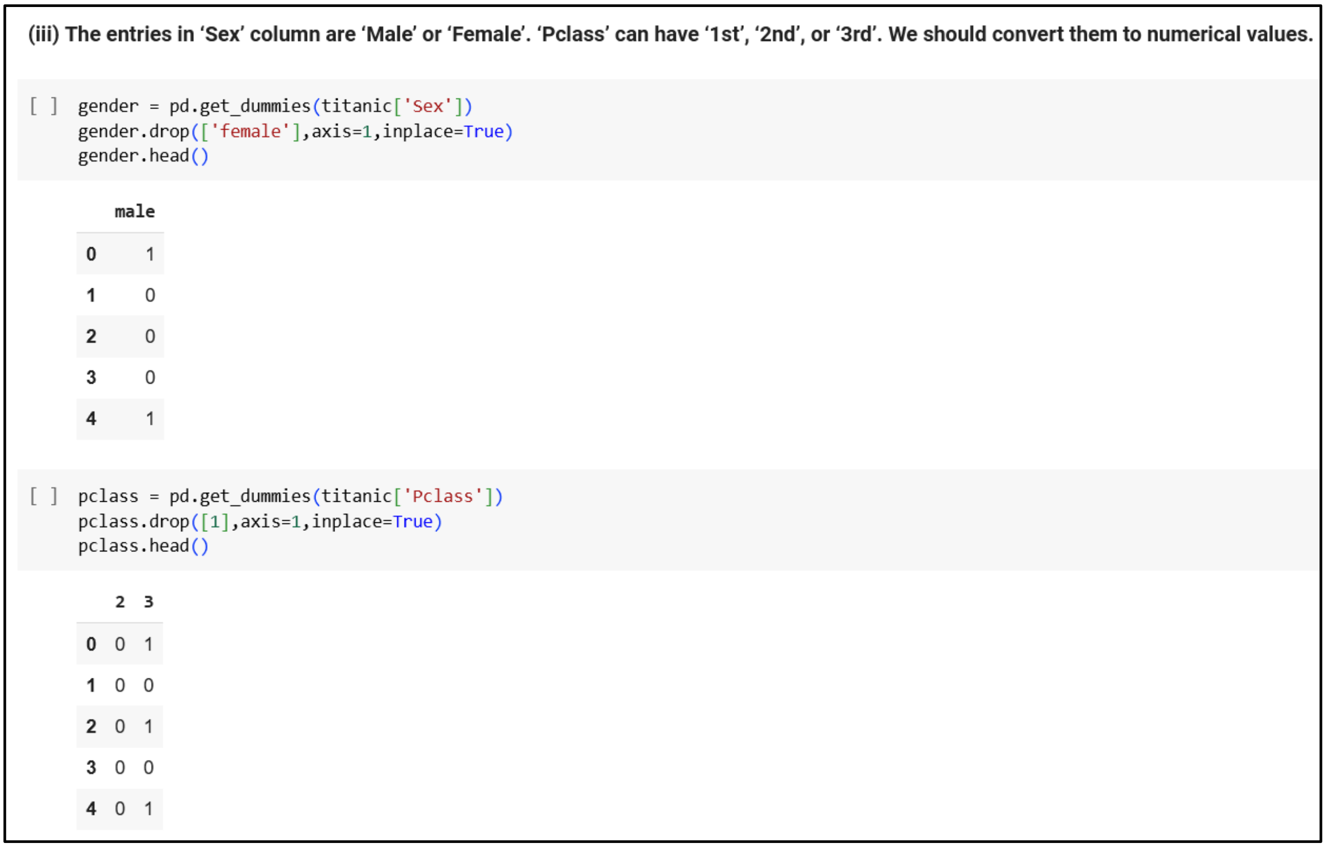


```python
nan_count = titanic['Age'].isna().sum()
print('\n')
print("Number of NaN values in column Age = ",nan_count)
print('\n')
titanic['Age'].fillna(age_mean,inplace=True)
titanic.head(10)
```

```
Number of NaN values in column Age =  177
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	0	3	male	22.0	1	0	7.2500
1	1	1	female	38.0	1	0	71.2833
2	1	3	female	26.0	0	0	7.9250
3	1	1	female	35.0	1	0	53.1000
4	0	3	male	35.0	0	0	8.0500
5	0	3	male	29.7	0	0	8.4583
6	0	1	male	54.0	0	0	51.8625
7	0	3	male	2.0	3	1	21.0750
8	1	3	female	27.0	0	2	11.1333
9	1	2	female	14.0	1	0	30.0708

(iii) The entries in the ‘Sex’ column are ‘Male’ or ‘Female’. ‘Pclass’ can have 1st,2nd,3rd. We should convert them to numerical values.



(iii) The entries in ‘Sex’ column are ‘Male’ or ‘Female’. ‘Pclass’ can have ‘1st’, ‘2nd’, or ‘3rd’. We should convert them to numerical values.

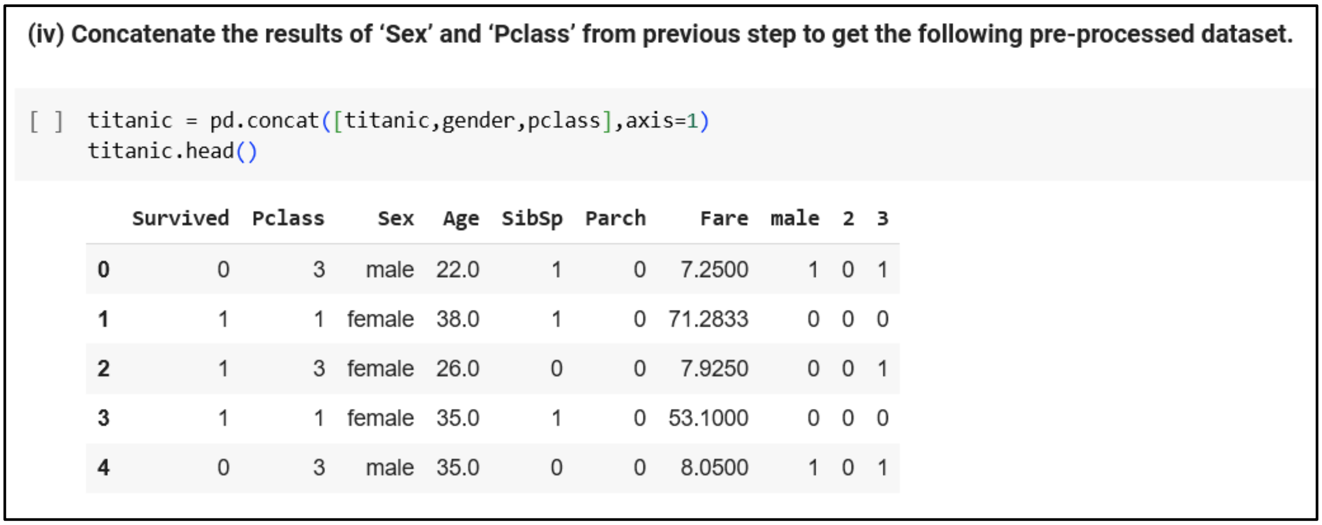
```
[ ]  gender = pd.get_dummies(titanic['Sex'])
     gender.drop(['female'],axis=1,inplace=True)
     gender.head()
```

	male
0	1
1	0
2	0
3	0
4	1

```
[ ]  pclass = pd.get_dummies(titanic['Pclass'])
     pclass.drop([1],axis=1,inplace=True)
     pclass.head()
```

	2	3
0	0	1
1	0	0
2	0	1
3	0	0
4	0	1

(iv) Concatenate the results of ‘Sex’ and ‘Pclass’ from previous step to get the following pre-processed dataset.

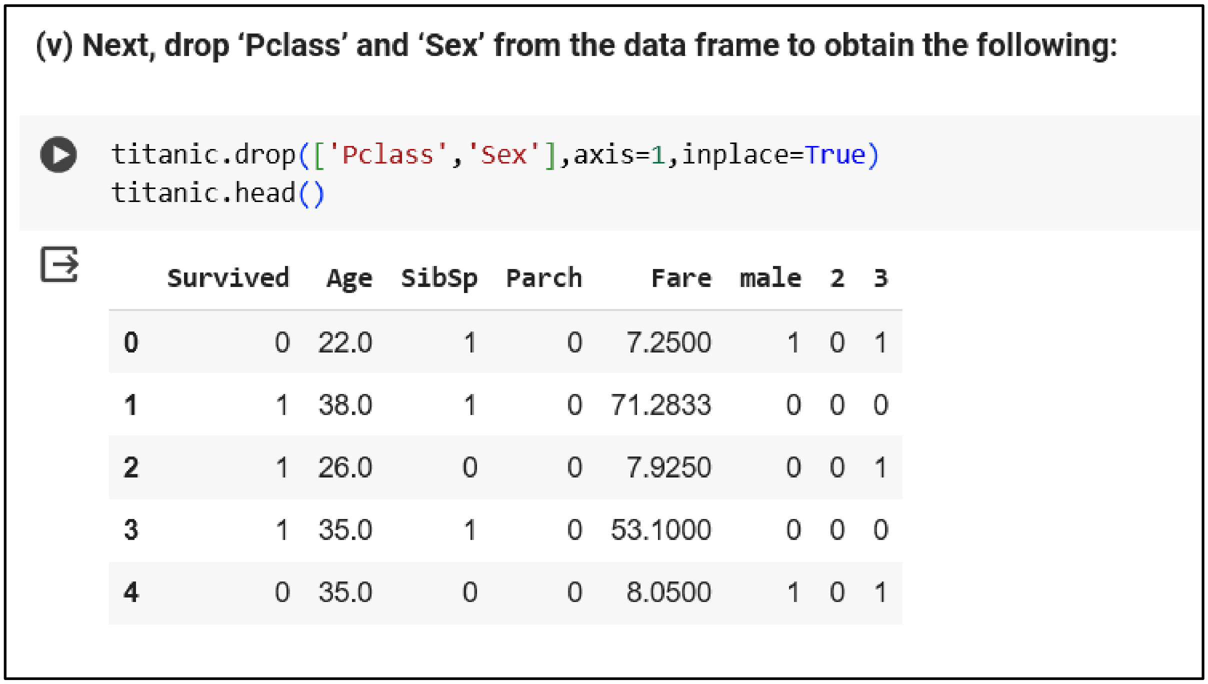


(iv) Concatenate the results of ‘Sex’ and ‘Pclass’ from previous step to get the following pre-processed dataset.

```
[ ]  titanic = pd.concat([titanic,gender,pclass],axis=1)
     titanic.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	male	2	3
0	0	3	male	22.0	1	0	7.2500	1	0	1
1	1	1	female	38.0	1	0	71.2833	0	0	0
2	1	3	female	26.0	0	0	7.9250	0	0	1
3	1	1	female	35.0	1	0	53.1000	0	0	0
4	0	3	male	35.0	0	0	8.0500	1	0	1

(v) Next, drop ‘Pclass’ and ‘Sex’ from the data frame to obtain the following:

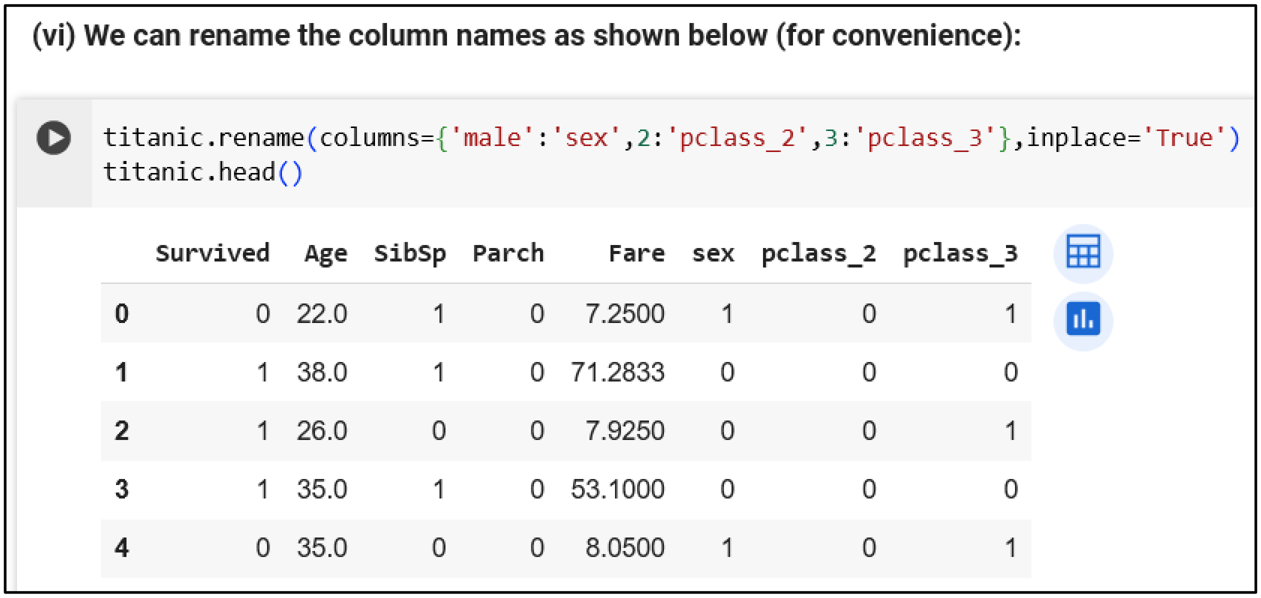


(v) Next, drop ‘Pclass’ and ‘Sex’ from the data frame to obtain the following:

```
titanic.drop(['Pclass','Sex'],axis=1,inplace=True)
titanic.head()
```

	Survived	Age	SibSp	Parch	Fare	male	2	3
0	0	22.0	1	0	7.2500	1	0	1
1	1	38.0	1	0	71.2833	0	0	0
2	1	26.0	0	0	7.9250	0	0	1
3	1	35.0	1	0	53.1000	0	0	0
4	0	35.0	0	0	8.0500	1	0	1

(vi) We can rename the column names as shown below (for convenience):

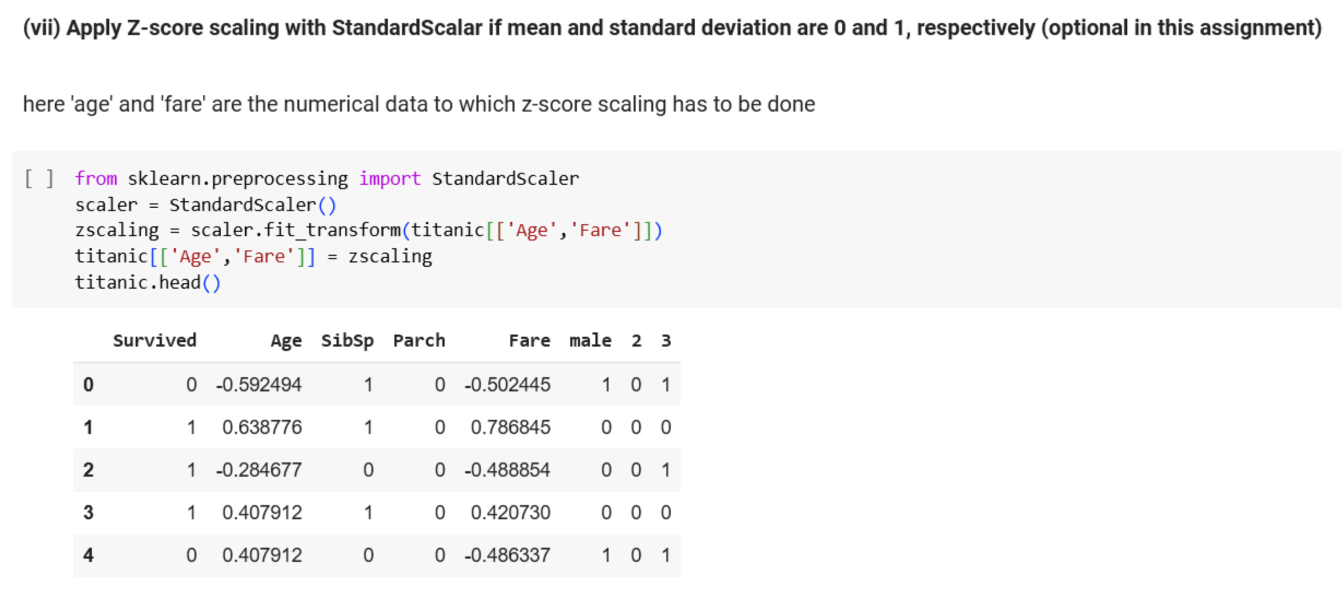


(vi) We can rename the column names as shown below (for convenience):

```
titanic.rename(columns={'male':'sex',2:'pclass_2',3:'pclass_3'},inplace='True')
titanic.head()
```

	Survived	Age	SibSp	Parch	Fare	sex	pclass_2	pclass_3
0	0	22.0	1	0	7.2500	1	0	1
1	1	38.0	1	0	71.2833	0	0	0
2	1	26.0	0	0	7.9250	0	0	1
3	1	35.0	1	0	53.1000	0	0	0
4	0	35.0	0	0	8.0500	1	0	1

(vii) Apply Z-score scaling with StandardScalar if mean and standard deviation are 0 and 1, respectively (optional in this assignment)



(vii) Apply Z-score scaling with StandardScalar if mean and standard deviation are 0 and 1, respectively (optional in this assignment)

here 'age' and 'fare' are the numerical data to which z-score scaling has to be done

```
[ ]  from sklearn.preprocessing import StandardScaler
     scaler = StandardScaler()
     zscaling = scaler.fit_transform(titanic[['Age','Fare']])
     titanic[['Age','Fare']] = zscaling
     titanic.head()
```

	Survived	Age	SibSp	Parch	Fare	male	2	3
0	0	-0.592494	1	0	-0.502445	1	0	1
1	1	0.638776	1	0	0.786845	0	0	0
2	1	-0.284677	0	0	-0.488854	0	0	1
3	1	0.407912	1	0	0.420730	0	0	0
4	0	0.407912	0	0	-0.486337	1	0	1

IV. Training and Testing

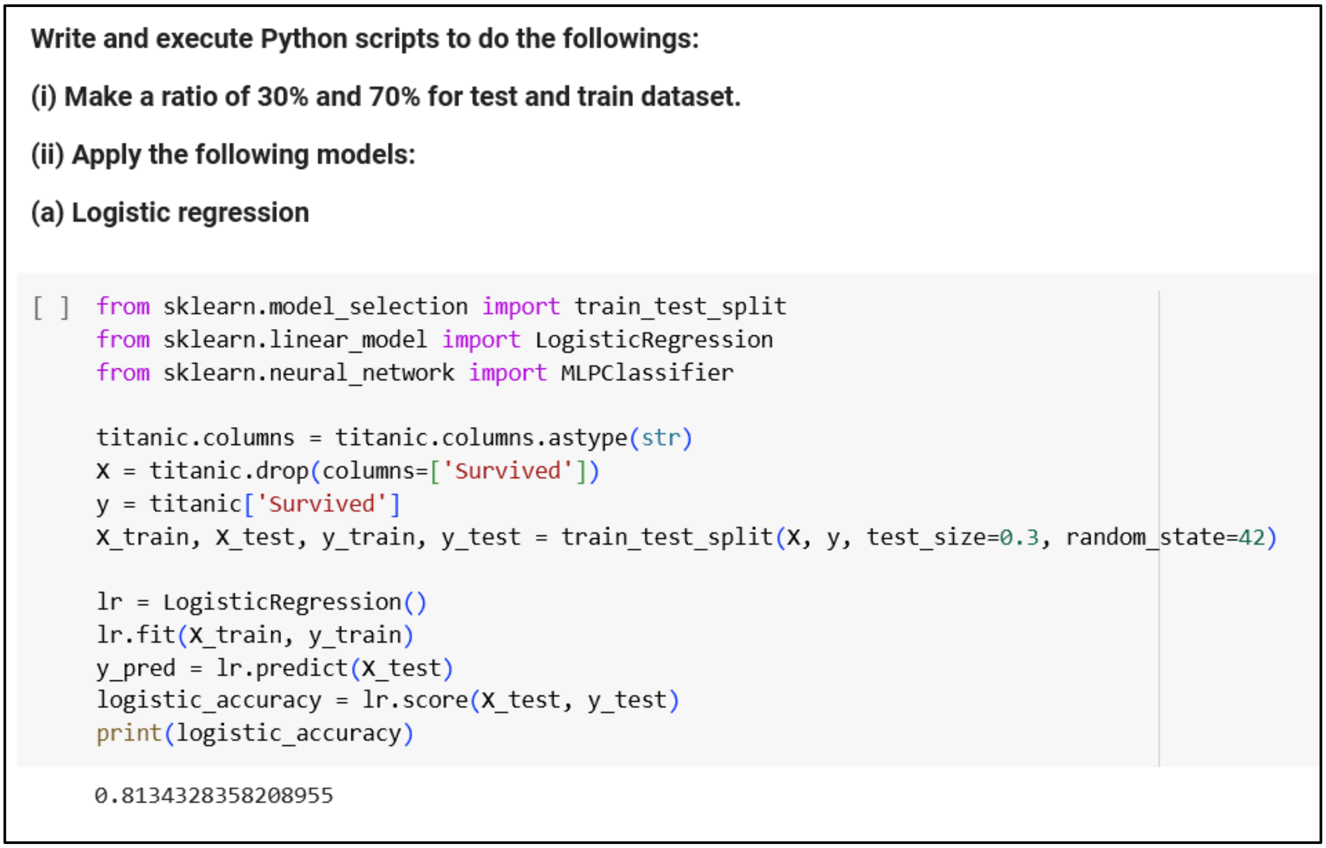
1) Write and execute Python scripts to do the followings:

(i) Make a ratio of 30% and 70% for test and train dataset.

(ii) Apply the following models:

(a) Logistic regression

(b) Neural Networks classifier



Write and execute Python scripts to do the followings:

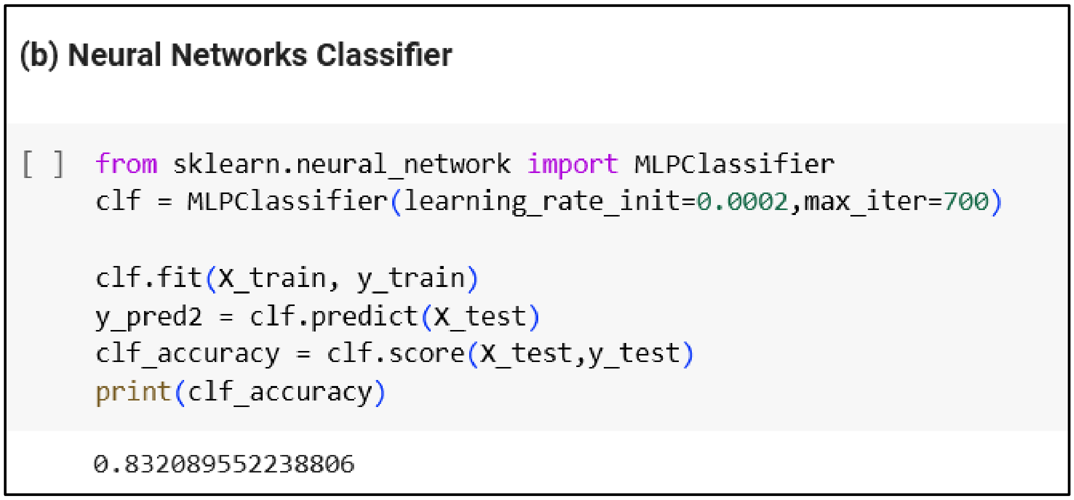
(i) Make a ratio of 30% and 70% for test and train dataset.

(ii) Apply the following models:

(a) Logistic regression

```
[ ]  from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LogisticRegression
     from sklearn.neural_network import MLPClassifier

     titanic.columns = titanic.columns.astype(str)
     X = titanic.drop(columns=['Survived'])
     y = titanic['Survived']
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

     lr = LogisticRegression()
     lr.fit(X_train, y_train)
     y_pred = lr.predict(X_test)
     logistic_accuracy = lr.score(X_test, y_test)
     print(logistic_accuracy)
```

```
0.8134328358208955
```



(b) Neural Networks Classifier

```
[ ]  from sklearn.neural_network import MLPClassifier
     clf = MLPClassifier(learning_rate_init=0.0002,max_iter=700)

     clf.fit(X_train, y_train)
     y_pred2 = clf.predict(X_test)
     clf_accuracy = clf.score(X_test,y_test)
     print(clf_accuracy)
```

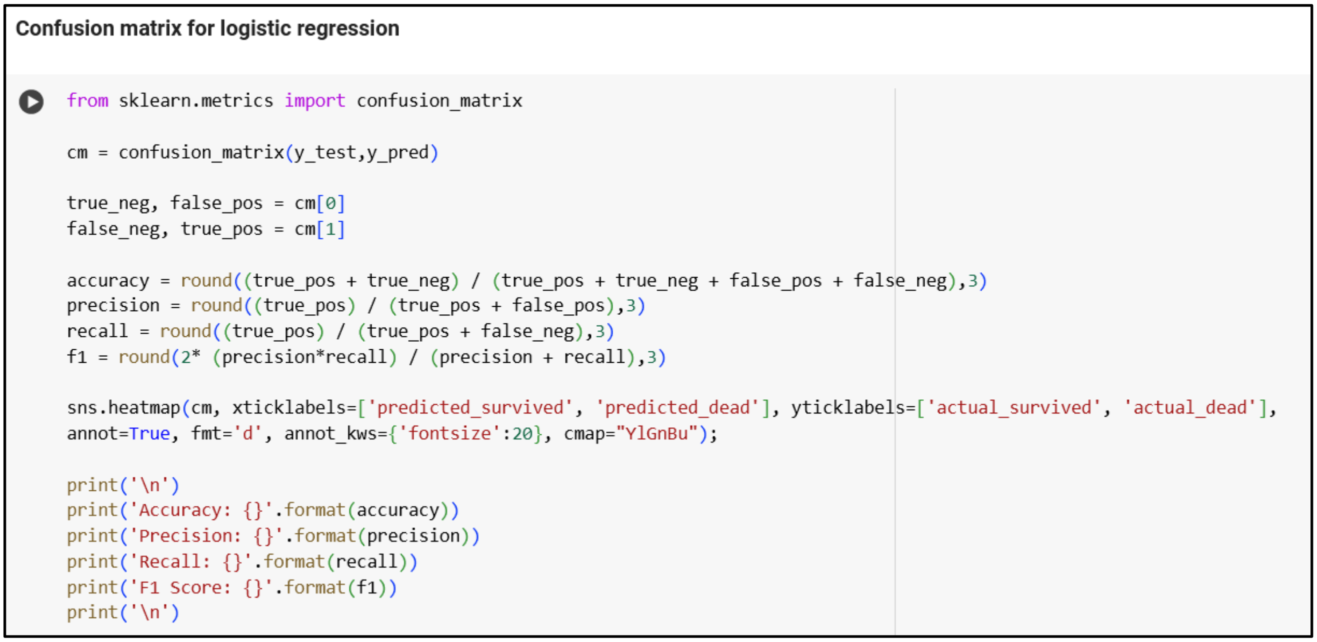
```
0.832089552238806
```

V. Performance Study

1) Write and execute Python scripts to do the followings:

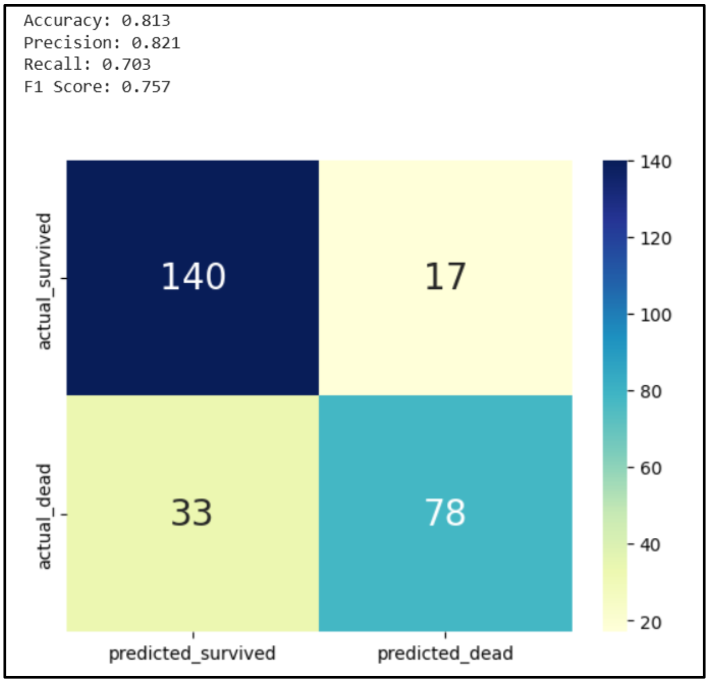
(i) Plot confusion matrix.

(ii) Find Precision, Recall, F1score, and Accuracy.

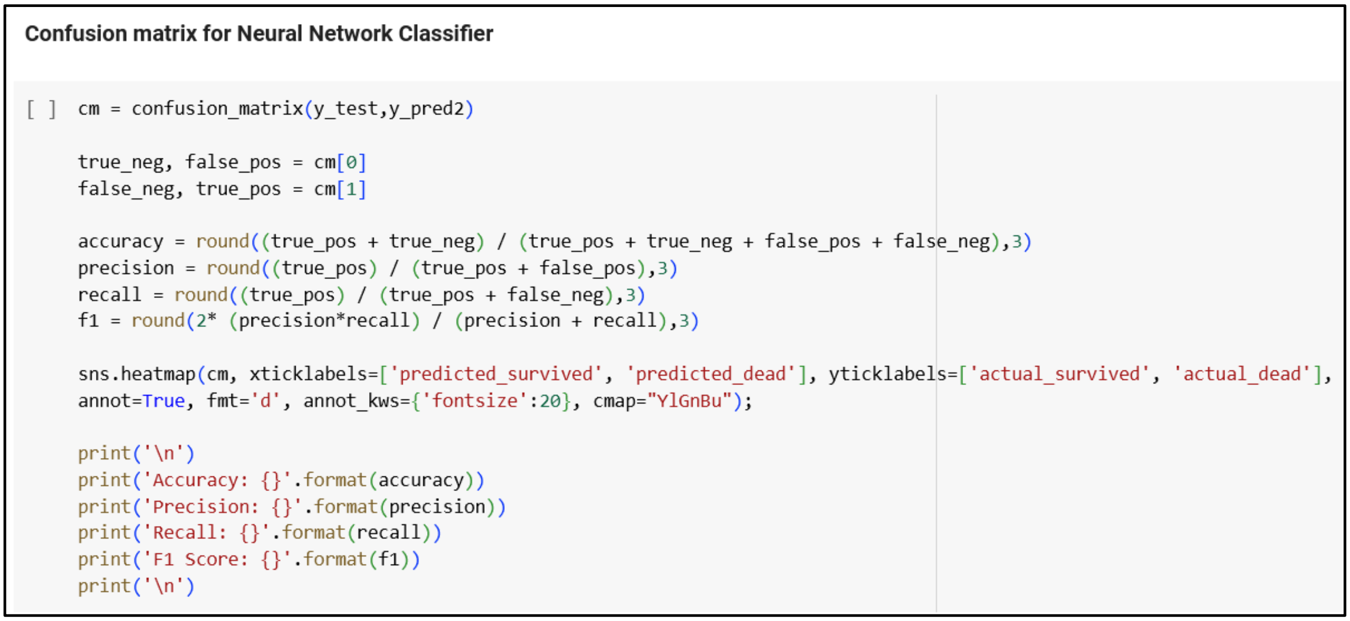


Confusion matrix for logistic regression

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test,y_pred)

true_neg, false_pos = cm[0]
false_neg, true_pos = cm[1]

accuracy = round((true_pos + true_neg) / (true_pos + true_neg + false_pos + false_neg),3)
precision = round((true_pos) / (true_pos + false_pos),3)
recall = round((true_pos) / (true_pos + false_neg),3)
f1 = round(2* (precision*recall) / (precision + recall),3)

sns.heatmap(cm, xticklabels=['predicted_survived', 'predicted_dead'], yticklabels=['actual_survived', 'actual_dead'],
annot=True, fmt='d', annot_kws={'fontsize':20}, cmap="YlGnBu");

print('\n')
print('Accuracy: {}'.format(accuracy))
print('Precision: {}'.format(precision))
print('Recall: {}'.format(recall))
print('F1 Score: {}'.format(f1))
print('\n')
```

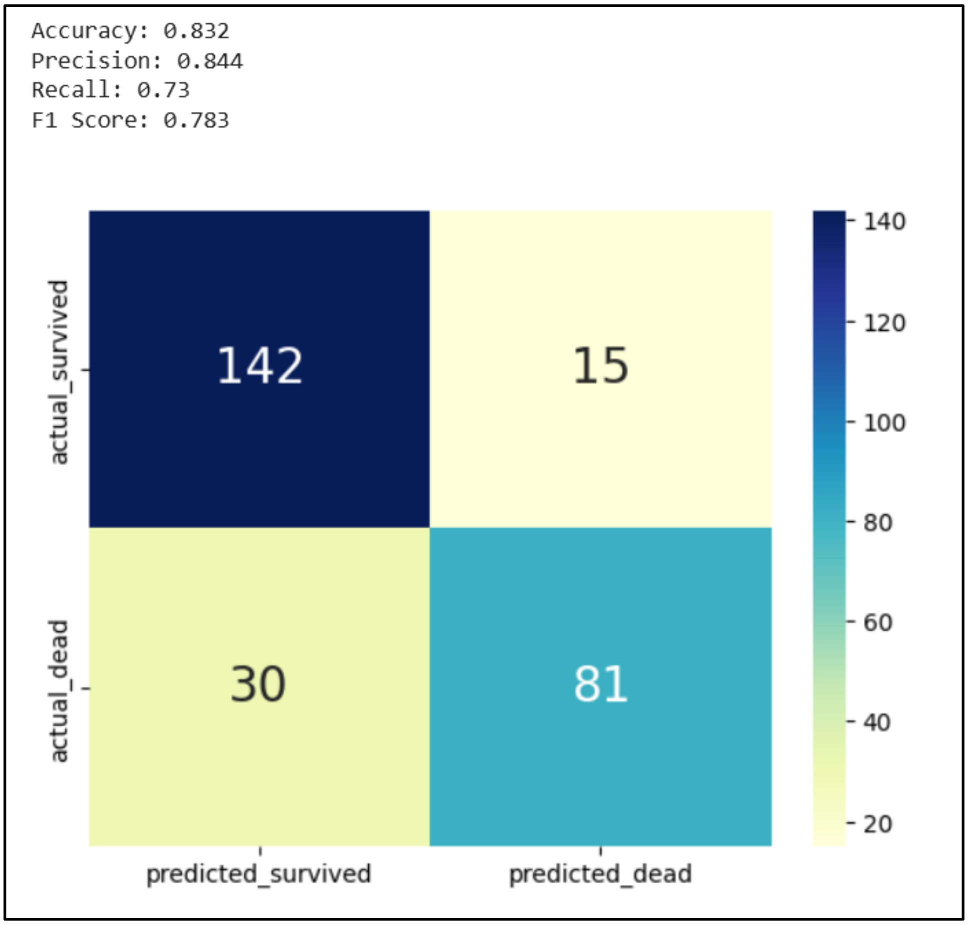


```
Accuracy: 0.813
Precision: 0.821
Recall: 0.703
F1 Score: 0.757
```



Confusion matrix for Neural Network Classifier

```
cm = confusion_matrix(y_test,y_pred2)

true_neg, false_pos = cm[0]
false_neg, true_pos = cm[1]

accuracy = round((true_pos + true_neg) / (true_pos + true_neg + false_pos + false_neg),3)
precision = round((true_pos) / (true_pos + false_pos),3)
recall = round((true_pos) / (true_pos + false_neg),3)
f1 = round(2* (precision*recall) / (precision + recall),3)

sns.heatmap(cm, xticklabels=['predicted_survived', 'predicted_dead'], yticklabels=['actual_survived', 'actual_dead'],
annot=True, fmt='d', annot_kws={'fontsize':20}, cmap="YlGnBu");

print('\n')
print('Accuracy: {}'.format(accuracy))
print('Precision: {}'.format(precision))
print('Recall: {}'.format(recall))
print('F1 Score: {}'.format(f1))
print('\n')
```



```
Accuracy: 0.832
Precision: 0.844
Recall: 0.73
F1 Score: 0.783
```

-- The End --