PlayTennis DataSet

```
In [ ]: import pandas as pd
        import numpy as np
```

```
In [2]: df=pd.read_csv('play_tennis.csv')
```

```
In [3]: df
```

Out[3]:

	day	outlook	temp	humidity	wind	play
0	D1	Sunny	Hot	High	Weak	No
1	D2	Sunny	Hot	High	Strong	No
2	D3	Overcast	Hot	High	Weak	Yes
3	D4	Rain	Mild	High	Weak	Yes
4	D5	Rain	Cool	Normal	Weak	Yes
5	D6	Rain	Cool	Normal	Strong	No
6	D7	Overcast	Cool	Normal	Strong	Yes
7	D8	Sunny	Mild	High	Weak	No
8	D9	Sunny	Cool	Normal	Weak	Yes
9	D10	Rain	Mild	Normal	Weak	Yes
10	D11	Sunny	Mild	Normal	Strong	Yes
11	D12	Overcast	Mild	High	Strong	Yes
12	D13	Overcast	Hot	Normal	Weak	Yes
13	D14	Rain	Mild	High	Strong	No

```
In [4]: df.shape

Out[4]: (14, 6)
```

```
In [5]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   day       14 non-null     object
 1   outlook   14 non-null     object
 2   temp      14 non-null     object
 3   humidity  14 non-null     object
 4   wind      14 non-null     object
 5   play      14 non-null     object
dtypes: object(6)
memory usage: 804.0+ bytes
```

```
In [6]: df.describe()
```

Out[6]:

	day	outlook	temp	humidity	wind	play
count	14	14	14	14	14	14
unique	14	3	3	2	2	2
top	D1	Sunny	Mild	High	Weak	Yes
freq	1	5	6	7	8	9

```
In [7]: df.isnull()
```

Out[7]:

	day	outlook	temp	humidity	wind	play
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
5	False	False	False	False	False	False
6	False	False	False	False	False	False
7	False	False	False	False	False	False
8	False	False	False	False	False	False
9	False	False	False	False	False	False
10	False	False	False	False	False	False
11	False	False	False	False	False	False
12	False	False	False	False	False	False
13	False	False	False	False	False	False

In [8]:
```
from sklearn.preprocessing import LabelEncoder
```

In [9]:
```
encoder = LabelEncoder()
```

In [11]:
```
new_play_tennis=pd.DataFrame()
new_play_tennis
```

Out[11]: —

In [12]:
```
new_play_tennis['outlook'] = encoder.fit_transform(df['outlook']) # converting car name to numeric data
new_play_tennis['temp'] = encoder.fit_transform(df['temp'])
new_play_tennis['humidity'] = encoder.fit_transform(df['humidity'])
new_play_tennis['wind'] = encoder.fit_transform(df['wind'])
new_play_tennis['play']=df['play']
```

In [13]:
```
new_play_tennis
```

Out[13]:

	outlook	temp	humidity	wind	play
0	2	1	0	1	No
1	2	1	0	0	No
2	0	1	0	1	Yes
3	1	2	0	1	Yes
4	1	0	1	1	Yes
5	1	0	1	0	No
6	0	0	1	0	Yes
7	2	2	0	1	No
8	2	0	1	1	Yes
9	1	2	1	1	Yes
10	2	2	1	0	Yes
11	0	2	0	0	Yes
12	0	1	1	1	Yes
13	1	2	0	0	No

In [15]:
```
X = pd.DataFrame(new_play_tennis.iloc[:, 0:4].values)
X
```

Out[15]:

	0	1	2	3
0	2	1	0	1
1	2	1	0	0
2	0	1	0	1
3	1	2	0	1
4	1	0	1	1
5	1	0	1	0
6	0	0	1	0
7	2	2	0	1
8	2	0	1	1
9	1	2	1	1
10	2	2	1	0
11	0	2	0	0
12	0	1	1	1
13	1	2	0	0

In [16]:
```
Y = pd.DataFrame(new_play_tennis.iloc[:,-1].values)
Y
```

Out[16]:

	0
0	No
1	No
2	Yes
3	Yes
4	Yes
5	No
6	Yes
7	No
8	Yes
9	Yes
10	Yes
11	Yes
12	Yes
13	No

In [17]:
```
from sklearn.model_selection import train_test_split
```

In [36]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, random_state=10)
```

In [37]:
```
from sklearn.naive_bayes import GaussianNB
```

In [38]:
```
clf=GaussianNB()
```

In [39]:
```
clf.fit(X_train,Y_train)
```

```
C:\Users\Personal\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A co
lumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for e
xample using ravel().
  y = column_or_1d(y, warn=True)
```

Out[39]:
```
▾ GaussianNB
GaussianNB()
```

```
In [40]: Y_pred=clf.predict(X_test)
```

```
In [41]: Y_pred
```

```
Out[41]: array(['Yes', 'No', 'Yes', 'Yes', 'Yes'], dtype='<U3')
```

```
In [42]: from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score,confusion_matrix
```

```
In [43]: accuracy_score(Y_test,Y_pred)
```

```
Out[43]: 1.0
```

```
In [44]: precision_score(Y_test, Y_pred, average='micro')
```

```
Out[44]: 1.0
```

```
In [45]: recall_score(Y_test, Y_pred, average='micro')
```

```
Out[45]: 1.0
```

```
In [46]: f1_score(Y_test, Y_pred, average='micro')
```

```
Out[46]: 1.0
```

```
In [69]: confusion_matrix(Y_test, Y_pred)
```

```
Out[69]: array([[1, 0],
                [0, 4]], dtype=int64)
```

```
In [72]: def outcome():
             print("---------MENU----------")
             print("For Outlook Column:- 0-Overcast, 1-Rain, 2-Sunny")
             x1=int(input("Outlook: "))
             print("For Temp Column:- 0-Cool, 1-Hot, 2-Mild")
             x2=int(input("Temp: "))
             print("For Humidity Column:- 0-High, 1-Normal")
             x3=int(input("Humidity: "))
             print("For Wind Column:- 0-Strong, 1-Weak")
             x4=int(input("Wind: "))

             list_x=[[x1,x2,x3,x4]]
             a = pd.DataFrame(list_x)

             y_pred_single = clf.predict(a)

             print("Prediction for given input is : ", y_pred_single[0])
```

```
In [73]: outcome()
```

```
---------MENU----------
For Outlook Column:- 0-Overcast, 1-Rain, 2-Sunny
Outlook: 1
For Temp Column:- 0-Cool, 1-Hot, 2-Mild
Temp: 0
For Humidity Column:- 0-High, 1-Normal
Humidity: 1
For Wind Column:- 0-Strong, 1-Weak
Wind: 0
Prediction for given input is :  No
```

Titanic DataSet

```
In [7]: import pandas as pd
        import numpy as np
```

```
In [8]: df=pd.read_csv('Titanic-Dataset.csv')
        df
```

Out[8]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Survived
0	1	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	1
2	3	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	1
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	1
4	5	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S	0
887	888	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S	1
888	889	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S	0
889	890	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C	1
890	891	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q	0

891 rows × 12 columns

In [9]:
```
df.shape
```
Out[9]:
```
(891, 12)
```

In [10]:
```
df.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   Pclass       891 non-null    int64  
 2   Name         891 non-null    object 
 3   Sex          891 non-null    object 
 4   Age          714 non-null    float64
 5   SibSp        891 non-null    int64  
 6   Parch        891 non-null    int64  
 7   Ticket       891 non-null    object 
 8   Fare         891 non-null    float64
 9   Cabin        204 non-null    object 
 10  Embarked     889 non-null    object 
 11  Survived     891 non-null    int64  
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [11]:
```
df.describe
```

Out[11]:
```
<bound method NDFrame.describe of      PassengerId  Pclass                                               Na
me  \
0              1       3                            Braund, Mr. Owen Harris
1              2       1  Cumings, Mrs. John Bradley (Florence Briggs Th...
2              3       3                             Heikkinen, Miss. Laina
3              4       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)
4              5       3                           Allen, Mr. William Henry
..           ...     ...                                                ...
886          887       2                              Montvila, Rev. Juozas
887          888       1                       Graham, Miss. Margaret Edith
888          889       3           Johnston, Miss. Catherine Helen "Carrie"
889          890       1                              Behr, Mr. Karl Howell
890          891       3                                Dooley, Mr. Patrick

        Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked  \
0      male  22.0      1      0         A/5 21171   7.2500   NaN        S
1    female  38.0      1      0          PC 17599  71.2833   C85        C
2    female  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S
3    female  35.0      1      0            113803  53.1000  C123        S
4      male  35.0      0      0            373450   8.0500   NaN        S
..      ...   ...    ...    ...               ...      ...   ...      ...
886    male  27.0      0      0            211536  13.0000   NaN        S
887  female  19.0      0      0            112053  30.0000   B42        S
888  female   NaN      1      2        W./C. 6607  23.4500   NaN        S
889    male  26.0      0      0            111369  30.0000  C148        C
890    male  32.0      0      0            370376   7.7500   NaN        Q

     Survived
0           0
1           1
2           1
3           1
4           0
..        ...
886         0
887         1
888         0
889         1
890         0

[891 rows x 12 columns]>
```

In [12]:
```
df.isnull()
```

Out[12]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Survived
0	False	False	False	False	False	False	False	False	False	True	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	True	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	True	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	True	False	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	True	False	False	False	False	True	False	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	True	False	False

891 rows × 12 columns

In [13]:
```
from sklearn.preprocessing import LabelEncoder
```

In [14]:
```
encoder = LabelEncoder()
```

In [15]:
```
new_data=pd.DataFrame()
new_data
```

Out[15]: —

In [16]:
```
new_data['Pclass'] = df['Pclass']
new_data['Name'] = encoder.fit_transform(df['Name'])
new_data['Sex'] = encoder.fit_transform(df['Sex'])
new_data['Age'] = df['Age']
new_data['SibSp'] = df['SibSp']
new_data['Parch'] = df['Parch']
new_data['Ticket'] = encoder.fit_transform(df['Ticket'])
new_data['Fare'] = df['Fare']
new_data['Cabin'] = encoder.fit_transform(df['Cabin'])
new_data['Embarked'] = encoder.fit_transform(df['Embarked'])
new_data['Survived'] = df['Survived']

new_data
```

Out[16]:

	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Survived
0	3	108	1	22.0	1	0	523	7.2500	147	2	0
1	1	190	0	38.0	1	0	596	71.2833	81	0	1
2	3	353	0	26.0	0	0	669	7.9250	147	2	1
3	1	272	0	35.0	1	0	49	53.1000	55	2	1
4	3	15	1	35.0	0	0	472	8.0500	147	2	0
...	...	...	...	...	...	...	...	...	...	...	...
886	2	548	1	27.0	0	0	101	13.0000	147	2	0
887	1	303	0	19.0	0	0	14	30.0000	30	2	1
888	3	413	0	NaN	1	2	675	23.4500	147	2	0
889	1	81	1	26.0	0	0	8	30.0000	60	0	1
890	3	220	1	32.0	0	0	466	7.7500	147	1	0

891 rows × 11 columns

In [17]:
```
from sklearn.impute import SimpleImputer
import numpy as np
```

In [18]:
```
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
```

In [19]:
```
data = imputer.fit_transform(new_data)
```

In [20]:
```
data
```

Out[20]:
```
array([[  3., 108.,   1., ..., 147.,   2.,   0.],
       [  1., 190.,   0., ...,  81.,   0.,   1.],
       [  3., 353.,   0., ..., 147.,   2.,   1.],
       ...,
       [  3., 413.,   0., ..., 147.,   2.,   0.],
       [  1.,  81.,   1., ...,  60.,   0.,   1.],
       [  3., 220.,   1., ..., 147.,   1.,   0.]])
```

In [22]:
```
data=pd.DataFrame(data)
data
```

Out[22]:

	0	1	2	3	4	5	6	7	8	9	10
0	3.0	108.0	1.0	22.000000	1.0	0.0	523.0	7.2500	147.0	2.0	0.0
1	1.0	190.0	0.0	38.000000	1.0	0.0	596.0	71.2833	81.0	0.0	1.0
2	3.0	353.0	0.0	26.000000	0.0	0.0	669.0	7.9250	147.0	2.0	1.0
3	1.0	272.0	0.0	35.000000	1.0	0.0	49.0	53.1000	55.0	2.0	1.0
4	3.0	15.0	1.0	35.000000	0.0	0.0	472.0	8.0500	147.0	2.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
886	2.0	548.0	1.0	27.000000	0.0	0.0	101.0	13.0000	147.0	2.0	0.0
887	1.0	303.0	0.0	19.000000	0.0	0.0	14.0	30.0000	30.0	2.0	1.0
888	3.0	413.0	0.0	29.699118	1.0	2.0	675.0	23.4500	147.0	2.0	0.0
889	1.0	81.0	1.0	26.000000	0.0	0.0	8.0	30.0000	60.0	0.0	1.0
890	3.0	220.0	1.0	32.000000	0.0	0.0	466.0	7.7500	147.0	1.0	0.0

891 rows × 11 columns

In [23]:

```
data.isnull()
```

Out[23]:

	0	1	2	3	4	5	6	7	8	9	10
0	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	False
887	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	False	False	False	False	False	False
889	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	False

891 rows × 11 columns

In [25]:

```
X = pd.DataFrame(data.iloc[:, 0:10].values)
X
```

Out[25]:

	0	1	2	3	4	5	6	7	8	9
0	3.0	108.0	1.0	22.000000	1.0	0.0	523.0	7.2500	147.0	2.0
1	1.0	190.0	0.0	38.000000	1.0	0.0	596.0	71.2833	81.0	0.0
2	3.0	353.0	0.0	26.000000	0.0	0.0	669.0	7.9250	147.0	2.0
3	1.0	272.0	0.0	35.000000	1.0	0.0	49.0	53.1000	55.0	2.0
4	3.0	15.0	1.0	35.000000	0.0	0.0	472.0	8.0500	147.0	2.0
...	...	...	...	...	...	...	...	...	...	...
886	2.0	548.0	1.0	27.000000	0.0	0.0	101.0	13.0000	147.0	2.0
887	1.0	303.0	0.0	19.000000	0.0	0.0	14.0	30.0000	30.0	2.0
888	3.0	413.0	0.0	29.699118	1.0	2.0	675.0	23.4500	147.0	2.0
889	1.0	81.0	1.0	26.000000	0.0	0.0	8.0	30.0000	60.0	0.0
890	3.0	220.0	1.0	32.000000	0.0	0.0	466.0	7.7500	147.0	1.0

891 rows × 10 columns

In [26]:

```
Y = pd.DataFrame(data.iloc[:, -1].values)
Y
```

Out[26]:

	0
0	0.0
1	1.0
2	1.0
3	1.0
4	0.0
...	...
886	0.0
887	1.0
888	0.0
889	1.0
890	0.0

891 rows × 1 columns

In [27]:
```python
from sklearn.model_selection import train_test_split
```

In [28]:
```python
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, random_state=42)
```

In [29]:
```python
from sklearn.naive_bayes import GaussianNB
```

In [30]:
```python
clf=GaussianNB()
```

In [31]:
```python
clf.fit(X_train,Y_train)
```

```
C:\Users\Personal\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A co
lumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for e
xample using ravel().
  y = column_or_1d(y, warn=True)
```

Out[31]:
```
▾ GaussianNB
GaussianNB()
```

In [32]:
```python
Y_pred=clf.predict(X_test)
```

In [33]:
```python
Y_pred
```

Out[33]:
```
array([0., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1., 0., 0., 0., 0., 1., 1.,
       1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
       0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 1., 1.,
       1., 0., 1., 0., 1., 0., 1., 1., 1., 0., 1., 1., 0., 0., 1., 0., 0.,
       0., 1., 1., 1., 1., 1., 0., 0., 0., 1., 1., 0., 0., 1., 1., 0., 1.,
       1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0.,
       1., 0., 0., 0., 1., 0., 0., 1., 1., 0., 1., 0., 1., 0., 1., 1., 1.,
       0., 0., 1., 1., 0., 0., 1., 1., 1., 1., 0., 1., 0., 0., 1., 1., 1.,
       1., 0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0.,
       0., 0., 0., 0., 1., 1., 1., 0., 0., 0., 1., 0., 1., 0., 1., 0., 0.,
       1., 1., 0., 1., 0., 0., 0., 1., 1., 1., 0., 0., 0., 0., 1., 1., 0.,
       0., 1., 1., 0., 0., 0., 0., 1., 1., 1., 0., 1., 0., 0., 0., 1., 0.,
       0., 1., 0., 1., 0., 0., 1., 0., 1., 0., 0., 0., 1., 1., 1., 0., 0.,
       1., 0., 1., 0., 1., 0., 1., 1., 0., 0., 1., 0., 1., 0., 0., 1., 0.,
       1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 1.,
       0., 0., 1., 0., 1., 1., 1., 1., 0., 0., 0., 0., 1., 1., 0., 1., 0.,
       0., 1., 1., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0.,
       0., 0., 0., 1., 1., 0.])
```

In [34]:
```python
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score,confusion_matrix
```

In [35]:
```python
accuracy_score(Y_test,Y_pred)
```

Out[35]:
```
0.8067796610169492
```

In [36]:
```python
precision_score(Y_test, Y_pred, average='micro')
```

Out[36]:
```
0.8067796610169492
```

In [37]:
```
recall_score(Y_test, Y_pred, average='micro')
```
Out[37]:
```
0.8067796610169492
```
In [38]:
```
f1_score(Y_test, Y_pred, average='micro')
```
Out[38]:
```
0.8067796610169492
```
In [39]:
```
confusion_matrix(Y_test, Y_pred)
```
Out[39]:
```
array([[147,  28],
       [ 29,  91]], dtype=int64)
```
In [41]:
```
data.describe()
```
Out[41]:

	0	1	2	3	4	5	6	7	8	9
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	2.308642	445.000000	0.647587	29.699118	0.523008	0.381594	338.528620	32.204208	130.744108	1.538721
std	0.836071	257.353842	0.477990	13.002015	1.102743	0.806057	200.850657	49.693429	36.024237	0.794231
min	1.000000	0.000000	0.000000	0.420000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2.000000	222.500000	0.000000	22.000000	0.000000	0.000000	158.500000	7.910400	147.000000	1.000000
50%	3.000000	445.000000	1.000000	29.699118	0.000000	0.000000	337.000000	14.454200	147.000000	2.000000
75%	3.000000	667.500000	1.000000	35.000000	1.000000	0.000000	519.500000	31.000000	147.000000	2.000000
max	3.000000	890.000000	1.000000	80.000000	8.000000	6.000000	680.000000	512.329200	147.000000	3.000000

In [44]:
```
def result():
    print("---------MENU----------")
    print("For Pclass range:1.0 - 3.00")
    x1=int(input("Pclass: "))
    print("For Name range:0.0 - 890.00")
    x2=int(input("Name: "))
    print("For Sex range:0.0 - 1.0")
    x3=int(input("Sex: "))
    print("For Age range:0.42 - 80.00")
    x4=int(input("Age: "))
    print("For SibSp range:0.00 - 8.00")
    x5=int(input("SibSp: "))
    print("For Parch range:0.00 - 8.00")
    x6=int(input("Parch: "))
    print("For Ticket range:0.00 - 680.00")
    x7=int(input("Ticket: "))
    print("For Fare range:0.00 - 512.3292")
    x8=int(input("Fare: "))
    print("For Cabin range:0.00 - 147.00")
    x9=int(input("Cabin: "))
    print("For Embarked range:0.00 - 3.00")
    x10=int(input("Embarked: "))


    list_x=[[x1,x2,x3,x4,x5,x6,x7,x8,x9,x10]]
    a = pd.DataFrame(list_x)

    y_pred_single = clf.predict(a)

    print("Prediction for given input is : ", y_pred_single[0])
```
In [46]:
```
result()
```

```
---------MENU----------
For Pclass range:1.0 - 3.00
Pclass: 2
For Name range:0.0 - 890.00
Name: 878
For Sex range:0.0 - 1.0
Sex: 1
For Age range:0.42 - 80.00
Age: 78
For SibSp range:0.00 - 8.00
SibSp: 6
For Parch range:0.00 - 8.00
Parch: 4
For Ticket range:0.00 - 680.00
Ticket: 543
For Fare range:0.00 - 512.3292
Fare: 342
For Cabin range:0.00 - 147.00
Cabin: 67
For Embarked range:0.00 - 3.00
Embarked: 0
Prediction for given input is :  1.0
```