Titanic Survival Predictor

```
import pandas as pd
import numpy as np
df=pd.read_csv("https://sololearn.com/uploads/files/titanic.csv")

print(df)
print(df.describe())
```

```
     Survived  Pclass     Sex   Age  Siblings/Spouses
Parents/Children  \
0           0       3    male  22.0                 1
0
1           1       1  female  38.0                 1
0
2           1       3  female  26.0                 0
0
3           1       1  female  35.0                 1
0
4           0       3    male  35.0                 0
0
..        ...     ...     ...   ...               ...               ..
.
882         0       2    male  27.0                 0
0
883         1       1  female  19.0                 0
0
884         0       3  female   7.0                 1
2
885         1       1    male  26.0                 0
0
886         0       3    male  32.0                 0
0

        Fare
0     7.2500
1    71.2833
2     7.9250
3    53.1000
4     8.0500
..       ...
882  13.0000
883  30.0000
884  23.4500
885  30.0000
886   7.7500

[887 rows x 7 columns]
         Survived      Pclass         Age  Siblings/Spouses
```

```
Parents/Children  \
count  887.000000  887.000000  887.000000        887.000000
887.000000
mean     0.385569    2.305524   29.471443          0.525366
0.383315
std      0.487004    0.836662   14.121908          1.104669
0.807466
min      0.000000    1.000000    0.420000          0.000000
0.000000
25%      0.000000    2.000000   20.250000          0.000000
0.000000
50%      0.000000    3.000000   28.000000          0.000000
0.000000
75%      1.000000    3.000000   38.000000          1.000000
0.000000
max      1.000000    3.000000   80.000000          8.000000
6.000000

            Fare
count  887.00000
mean    32.30542
std     49.78204
min      0.00000
25%      7.92500
50%     14.45420
75%     31.13750
max    512.32920

print(df.head())

   Survived  Pclass     Sex   Age  Siblings/Spouses  Parents/Children
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

print(df.head(n=10))

   Survived  Pclass     Sex   Age  Siblings/Spouses  Parents/Children
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  27.0                 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
```

```
df["male"]=df["Sex"]=="male"
```

```
df.head()
```

```
   Survived  Pclass     Sex   Age  Siblings/Spouses  Parents/Children
\
0         0       3    male  22.0                 1                 0

1         1       1  female  38.0                 1                 0

2         1       3  female  26.0                 0                 0

3         1       1  female  35.0                 1                 0

4         0       3    male  35.0                 0                 0


      Fare   male
0   7.2500   True
1  71.2833  False
2   7.9250  False
3  53.1000  False
4   8.0500   True
```

```
col=df["Fare"]
col
```

```
0       7.2500
1      71.2833
2       7.9250
3      53.1000
4       8.0500
        ...
882    13.0000
```

```
883    30.0000
884    23.4500
885    30.0000
886     7.7500
Name: Fare, Length: 887, dtype: float64
```

```
col=df["Fare"]
col
print(np.mean(col))
```

```
32.30542018038328
```

```
import numpy as np
print(np.mean(col))
```

```
32.30542018038328
```

```
col1=df[["Age","Fare"]]
```

```
col1
```

```
      Age     Fare
0    22.0   7.2500
1    38.0  71.2833
2    26.0   7.9250
3    35.0  53.1000
4    35.0   8.0500
..    ...      ...
882  27.0  13.0000
883  19.0  30.0000
884   7.0  23.4500
885  26.0  30.0000
886  32.0   7.7500

[887 rows x 2 columns]
```

```
print(np.mean(col1))
```

```
Age     29.471443
Fare    32.305420
dtype: float64
```

```
print(np.percentile(col,25))
```

```
7.925
```

```
df
```

```
     Survived  Pclass     Sex   Age  Siblings/Spouses
Parents/Children  \
0           0       3    male  22.0                 1
0
1           1       1  female  38.0                 1
```

```
0
2           1       3  female  26.0                 0
0
3           1       1  female  35.0                 1
0
4           0       3    male  35.0                 0
0
..        ...     ...     ...   ...               ...               ..
.
882         0       2    male  27.0                 0
0
883         1       1  female  19.0                 0
0
884         0       3  female   7.0                 1
2
885         1       1    male  26.0                 0
0
886         0       3    male  32.0                 0
0

        Fare   male
0     7.2500   True
1    71.2833  False
2     7.9250  False
3    53.1000  False
4     8.0500   True
..       ...    ...
882  13.0000   True
883  30.0000  False
884  23.4500  False
885  30.0000   True
886   7.7500   True

[887 rows x 8 columns]

print(df["Fare"].values[:11])

[ 7.25   71.2833  7.925  53.1     8.05    8.4583 51.8625 21.075
11.1333
 30.0708 16.7   ]

print(df.shape)

(887, 8)

df

     Survived  Pclass     Sex   Age  Siblings/Spouses
Parents/Children  \
0           0       3    male  22.0                 1
0
```

```
1           1       1  female  38.0                1
0
2           1       3  female  26.0                0
0
3           1       1  female  35.0                1
0
4           0       3    male  35.0                0
0
..        ...     ...     ...   ...               ...               ..
.
882         0       2    male  27.0                0
0
883         1       1  female  19.0                0
0
884         0       3  female   7.0                1
2
885         1       1    male  26.0                0
0
886         0       3    male  32.0                0
0

        Fare   male
0     7.2500   True
1    71.2833  False
2     7.9250  False
3    53.1000  False
4     8.0500   True
..       ...    ...
882  13.0000   True
883  30.0000  False
884  23.4500  False
885  30.0000   True
886   7.7500   True

[887 rows x 8 columns]
```

```
df["male"]=df["Sex"]=="male"
```

```
del(df["male"])
df
```

```
     Survived  Pclass     Sex   Age  Siblings/Spouses
Parents/Children  \
0           0       3    male  22.0                1
0
1           1       1  female  38.0                1
0
2           1       3  female  26.0                0
0
3           1       1  female  35.0                1
0
```

```
4           0       3    male  35.0                0
0
..        ...     ...     ...   ...               ...               ..
.
882         0       2    male  27.0                0
0
883         1       1  female  19.0                0
0
884         0       3  female   7.0                1
2
885         1       1    male  26.0                0
0
886         0       3    male  32.0                0
0

        Fare
0     7.2500
1    71.2833
2     7.9250
3    53.1000
4     8.0500
..       ...
882  13.0000
883  30.0000
884  23.4500
885  30.0000
886   7.7500

[887 rows x 7 columns]
```

```
arr=df[["Pclass","Fare","Age"]].values
arr
```

```
array([[ 3.    ,  7.25  , 22.    ],
       [ 1.    , 71.2833, 38.    ],
       [ 3.    ,  7.925 , 26.    ],
       ...,
       [ 3.    , 23.45  ,  7.    ],
       [ 1.    , 30.    , 26.    ],
       [ 3.    ,  7.75  , 32.    ]])
```

```
arr
```

```
array([[ 3.    ,  7.25  , 22.    ],
       [ 1.    , 71.2833, 38.    ],
       [ 3.    ,  7.925 , 26.    ],
       ...,
       [ 3.    , 23.45  ,  7.    ],
       [ 1.    , 30.    , 26.    ],
       [ 3.    ,  7.75  , 32.    ]])
```

```
print(arr[0,1])
```

```
7.25
```

```
print(arr[0])
```

```
[ 3.    7.25 22.  ]
```

```
print(arr[:,2])
```

```
[22.   38.   26.   35.   35.   27.   54.    2.   27.   14.    4.   58.
 20.   39.   14.   55.    2.   23.   31.   22.   35.   34.   15.   28.
  8.   38.   26.   19.   24.   23.   40.   48.   18.   66.   28.   42.
 18.   21.   18.   14.   40.   27.    3.   19.   30.   20.   27.   16.
 18.    7.   21.   49.   29.   65.   46.   21.   28.5   5.   11.   22.
 38.   45.    4.   64.    7.   29.   19.   17.   26.   32.   16.   21.
 26.   32.   25.   23.   28.    0.83 30.   22.   29.   31.   28.   17.
 33.   16.   20.   23.   24.   29.   20.   46.   26.   59.   22.   71.
 23.   34.   34.   28.   29.   21.   33.   37.   28.   21.   29.   38.
 28.   47.   14.5  22.   20.   17.   21.   70.5  29.   24.    2.   21.
 19.   32.5  32.5  54.   12.   19.   24.    2.   45.   33.   20.   47.
 29.   25.   23.   19.   37.   16.   24.   40.   22.   24.   19.   18.
 19.   27.    9.   36.5  42.   51.   22.   55.5  40.5  27.   51.   16.
 30.   37.    5.   44.   40.   26.   17.    1.    9.   48.   45.   60.
 28.   61.    4.    1.   21.   56.   18.    5.   50.   30.   36.    8.
 39.    9.    1.    4.   39.   26.   45.   40.   36.   32.   19.   19.
  3.   44.   58.   28.   42.   21.   24.   28.   17.   34.   45.5  18.
  2.   32.   26.   16.   40.   24.   35.   22.   30.   22.   31.   27.
 42.   32.   30.   16.   27.   51.   22.   38.   22.   19.   20.5  18.
 12.   35.   29.   59.    5.   24.   21.   44.    8.   19.   33.   19.
 18.   29.   22.   30.   44.   25.   24.   37.   54.   18.   29.   62.
 30.   41.   29.   38.   30.   35.   50.    3.   52.   40.   21.   36.
 16.   25.   58.   35.   28.   25.   41.   37.   33.   63.   45.   21.
  7.   35.   65.   28.   16.   19.   57.   33.   30.   22.   42.   22.
 26.   19.   36.   24.   24.   30.   23.5   2.   47.   50.   20.   24.
 19.   46.   28.    0.92 42.   17.   30.   30.   24.   18.   26.   28.
 43.   26.   24.   54.   31.   40.   22.   27.   30.   22.   20.   36.
 61.   36.   31.   16.   28.   45.5  38.   16.   42.   30.   29.   41.
 45.   45.    2.   24.   28.   25.   36.   24.   40.   34.    3.   42.
 23.   43.   15.   25.   23.   28.   22.   38.   22.   23.   40.   29.
 45.   35.   27.   30.   60.   35.   22.   24.   25.   18.   19.   22.
  3.   25.   22.   27.   20.   19.   42.    1.   32.   35.   27.   18.
  1.   36.   19.   17.   36.   21.   28.   23.   24.   22.   31.   46.
 23.   28.   39.   26.   21.   28.   20.   34.   51.    3.   21.    3.
 42.   27.   33.   22.   44.   32.   34.   18.   30.   10.   21.   29.
 28.   18.   54.   28.   19.   28.   32.   28.   33.   42.   17.   50.
 14.   21.   24.   64.   31.   45.   20.   25.   28.   29.    4.   13.
 34.    5.   52.   36.   28.   30.   49.   24.   29.   65.   41.   50.
 17.   48.   34.   47.   48.   34.   38.   21.   56.   22.    0.75 39.
 38.   33.   23.   22.   40.   34.   29.   22.    2.    9.   37.   50.
 63.   25.    8.   35.   58.   30.    9.   19.   21.   55.   71.   21.
```

```
 26.   54.   55.   25.   24.   17.   21.   21.   37.   16.   18.   33.
 37.   28.   26.   29.   66.   36.   54.   24.   47.   34.   30.   36.
 32.   30.   22.   35.   44.   18.   40.5  50.   49.   39.   23.    2.
 17.   17.   24.   30.    7.   45.   30.   69.   22.   36.    9.   11.
 32.   50.   64.   19.   27.   33.    8.   17.   27.   21.   22.   22.
 62.   48.   45.   39.   36.   30.   40.   28.   40.   62.   24.   19.
 29.   28.   32.   62.   53.   36.   22.   16.   19.   34.   39.   18.
 32.   25.   39.   54.   36.   16.   18.   47.   60.   22.   22.   35.
 52.   47.   40.   37.   36.   31.   49.   18.   49.   24.   42.   37.
 44.   35.   36.   30.   27.   22.   40.   39.   21.   18.   22.   35.
 24.   34.   26.    4.   26.   27.   42.   20.   21.   21.   61.   57.
 21.   26.   18.   80.   51.   32.   30.    9.   28.   32.   31.   41.
 37.   20.   24.    2.   32.    0.75 48.   19.   56.   21.   23.   23.
 18.   21.   16.   18.   24.   27.   32.   23.   58.   50.   40.   47.
 36.   20.   32.   25.   49.   43.   48.   40.   31.   70.   31.   19.
 18.   24.5  18.   43.   36.   28.   27.   20.   14.   60.   25.   14.
 19.   18.   15.   31.    4.   37.   25.   60.   52.   44.   19.   49.
 42.   18.   35.   18.   25.   26.   39.   45.   42.   22.    4.   24.
 41.   48.   29.   52.   19.   38.   27.   33.    6.   17.   34.   50.
 27.   20.   30.   28.   25.   25.   29.   11.   41.   23.   23.
28.5
 48.   35.   20.   32.   45.   36.   21.   24.   31.   70.   16.   30.
 19.   31.    4.    6.   33.   23.   48.    0.67 28.   18.   34.   33.
 23.   41.   20.   36.   16.   51.   46.   30.5  28.   32.   24.   48.
 57.   29.   54.   18.   20.    5.   22.   43.   13.   17.   29.   35.
 25.   25.   18.    8.    1.   46.   20.   16.   21.   43.   25.   39.
 49.   31.   30.   30.   34.   31.   11.    0.42 27.   31.   39.   18.
 39.   33.   26.   39.   35.    6.   30.5  39.   23.   31.   43.   10.
 52.   27.   38.   27.    2.   36.   23.    1.   19.   62.   15.
0.83
 30.   23.   18.   39.   21.   20.   32.   29.   20.   16.   30.
34.5
 17.   42.   18.   35.   28.   40.    4.   74.    9.   16.   44.   18.
 45.   51.   24.   30.   41.   21.   48.   14.   24.   42.   27.   31.
 23.    4.   26.   47.   33.   47.   28.   15.   20.   19.   23.   56.
 25.   33.   22.   28.   25.   39.   27.   19.    7.   26.   32.  ]
```

```
arr
```

```
array([[ 3.    ,  7.25  , 22.    ],
       [ 1.    , 71.2833, 38.    ],
       [ 3.    ,  7.925 , 26.    ],
       ...,
       [ 3.    , 23.45  ,  7.    ],
       [ 1.    , 30.    , 26.    ],
       [ 3.    ,  7.75  , 32.    ]])
```

```
arr[:,0]
```

```
array([3., 1., 3., 1., 3., 3., 1., 3., 3., 2., 3., 1., 3., 3., 3., 2.,
3.,
```

2., 3., 3., 2., 2., 3., 1., 3., 3., 3., 1., 3., 3., 1., 1., 3.,
2.,
1., 1., 3., 3., 3., 3., 3., 2., 2., 3., 3., 3., 3., 3., 3., 3.,
3.,
1., 2., 1., 1., 2., 3., 2., 3., 3., 1., 1., 3., 1., 3., 2., 3.,
3.,
3., 2., 3., 2., 3., 3., 3., 3., 3., 2., 3., 3., 3., 3., 1., 2.,
3.,
3., 3., 1., 3., 3., 3., 1., 3., 3., 3., 1., 1., 2., 2., 3., 3.,
1.,
3., 3., 3., 3., 3., 3., 3., 1., 3., 3., 3., 3., 3., 3., 2., 1.,
3.,
2., 3., 2., 2., 1., 3., 3., 3., 3., 3., 3., 3., 3., 2., 2., 2.,
1.,
1., 3., 1., 3., 3., 3., 3., 2., 2., 3., 3., 2., 2., 2., 1., 3.,
3.,
3., 1., 3., 3., 3., 3., 3., 2., 3., 3., 3., 3., 1., 3., 1., 3.,
1.,
3., 3., 3., 1., 3., 3., 1., 2., 3., 3., 2., 3., 2., 3., 1., 3.,
1.,
3., 3., 2., 2., 3., 2., 1., 1., 3., 3., 3., 2., 3., 3., 3., 3.,
3.,
3., 3., 3., 3., 1., 3., 2., 3., 2., 3., 1., 3., 2., 1., 2., 3.,
2.,
3., 3., 1., 3., 2., 3., 2., 3., 1., 3., 2., 3., 2., 3., 2., 2.,
2.,
2., 3., 3., 2., 3., 3., 1., 3., 2., 1., 2., 3., 3., 1., 3., 3.,
3.,
1., 1., 1., 2., 3., 1., 1., 3., 2., 3., 3., 1., 1., 1., 3., 2.,
1.,
3., 1., 3., 2., 3., 3., 3., 3., 3., 3., 1., 3., 3., 3., 2., 3.,
1.,
1., 2., 3., 3., 1., 3., 1., 1., 1., 3., 3., 3., 2., 3., 1., 1.,
1.,
2., 1., 1., 1., 2., 3., 2., 3., 2., 2., 1., 1., 3., 3., 2., 2.,
3.,
1., 3., 2., 3., 1., 3., 1., 1., 3., 1., 3., 1., 1., 3., 1., 2.,
1.,
2., 2., 2., 2., 2., 3., 3., 3., 3., 1., 3., 3., 3., 3., 1., 2.,
3.,
3., 3., 2., 3., 3., 3., 3., 1., 3., 3., 1., 1., 3., 3., 1., 3.,
1.,
3., 1., 3., 3., 1., 3., 3., 1., 3., 2., 3., 2., 3., 2., 1., 3.,
3.,
1., 3., 3., 3., 2., 2., 2., 3., 3., 3., 3., 3., 2., 3., 2., 3.,
3.,
3., 3., 1., 2., 3., 3., 2., 2., 2., 3., 3., 3., 3., 3., 3., 2.,
2.,
3., 3., 1., 3., 2., 3., 1., 1., 3., 2., 1., 2., 2., 3., 3., 2.,
3.,

1., 2., 1., 3., 1., 2., 3., 1., 1., 3., 3., 1., 1., 2., 3., 1.,
3.,
1., 2., 3., 3., 2., 1., 3., 3., 3., 3., 2., 2., 3., 1., 2., 3.,
3.,
3., 3., 2., 3., 3., 1., 3., 1., 1., 3., 3., 3., 3., 1., 1., 3.,
3.,
1., 3., 1., 3., 3., 3., 3., 3., 1., 1., 2., 1., 3., 3., 3., 3.,
1.,
1., 3., 1., 2., 3., 2., 3., 1., 3., 3., 1., 3., 3., 2., 1., 3.,
2.,
2., 3., 3., 3., 3., 2., 1., 1., 3., 1., 1., 3., 3., 2., 1., 1.,
2.,
2., 3., 2., 1., 2., 3., 3., 3., 1., 1., 1., 1., 3., 3., 3., 2.,
3.,
3., 3., 3., 3., 3., 3., 2., 1., 1., 3., 3., 3., 2., 1., 3., 3.,
2.,
1., 2., 1., 3., 1., 2., 1., 3., 3., 3., 1., 3., 3., 2., 3., 2.,
3.,
3., 1., 2., 3., 1., 3., 1., 3., 3., 1., 2., 1., 3., 3., 3., 3.,
3.,
2., 3., 3., 2., 2., 3., 1., 3., 3., 3., 1., 2., 1., 3., 3., 1.,
3.,
1., 1., 3., 2., 3., 2., 3., 3., 3., 1., 3., 3., 3., 1., 3., 1.,
3.,
3., 3., 2., 3., 3., 3., 2., 3., 3., 2., 1., 1., 3., 1., 3., 3.,
2.,
2., 3., 3., 1., 2., 1., 2., 2., 2., 3., 3., 3., 3., 1., 3., 1.,
3.,
3., 2., 2., 3., 3., 3., 1., 1., 3., 3., 3., 1., 2., 3., 3., 1.,
3.,
1., 1., 3., 3., 3., 2., 2., 1., 1., 3., 1., 1., 1., 3., 2., 3.,
1.,
2., 3., 2., 3., 2., 2., 1., 3., 2., 3., 2., 3., 1., 3., 2., 2.,
2.,
3., 3., 1., 3., 3., 1., 1., 1., 3., 3., 1., 3., 2., 1., 3., 2.,
3.,
3., 3., 2., 2., 3., 2., 3., 1., 3., 3., 3., 1., 3., 1., 1., 3.,
3.,
3., 3., 3., 2., 3., 2., 3., 3., 3., 3., 1., 3., 1., 1., 3., 3.,
3.,
3., 3., 3., 1., 3., 2., 3., 1., 3., 2., 1., 3., 3., 3., 2., 2.,
1.,
3., 3., 3., 1., 3., 2., 1., 3., 3., 2., 3., 3., 1., 3., 2., 3.,
3.,
1., 3., 1., 3., 3., 3., 3., 2., 3., 1., 3., 2., 3., 3., 3., 1.,
3.,
3., 3., 1., 3., 2., 1., 3., 3., 3., 3., 3., 2., 1., 3., 3., 3.,
1.,
2., 3., 1., 1., 3., 3., 3., 2., 1., 3., 2., 2., 2., 1., 3., 3.,
3.,

```
       1., 1., 3., 2., 3., 3., 3., 3., 1., 2., 3., 3., 2., 3., 3., 2.,
1.,
       3., 1., 3.])
```

```
mask=arr[:,0]<18
print(mask)
```

```
[ True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
```

```
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
```

```
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True
True
```

```
  True  True  True  True  True  True  True  True  True  True  True
True
  True  True  True  True  True  True  True  True  True  True  True]
```

mask

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
```

```
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
True,  True,  True,  True,  True,  True,  True,  True,  True,
```

```
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True])
```

```
arr[mask]
```

```
array([[ 3.    ,  7.25  , 22.    ],
       [ 1.    , 71.2833, 38.    ],
       [ 3.    ,  7.925 , 26.    ],
       ...,
       [ 3.    , 23.45  ,  7.    ],
       [ 1.    , 30.    , 26.    ],
       [ 3.    ,  7.75  , 32.    ]])
```

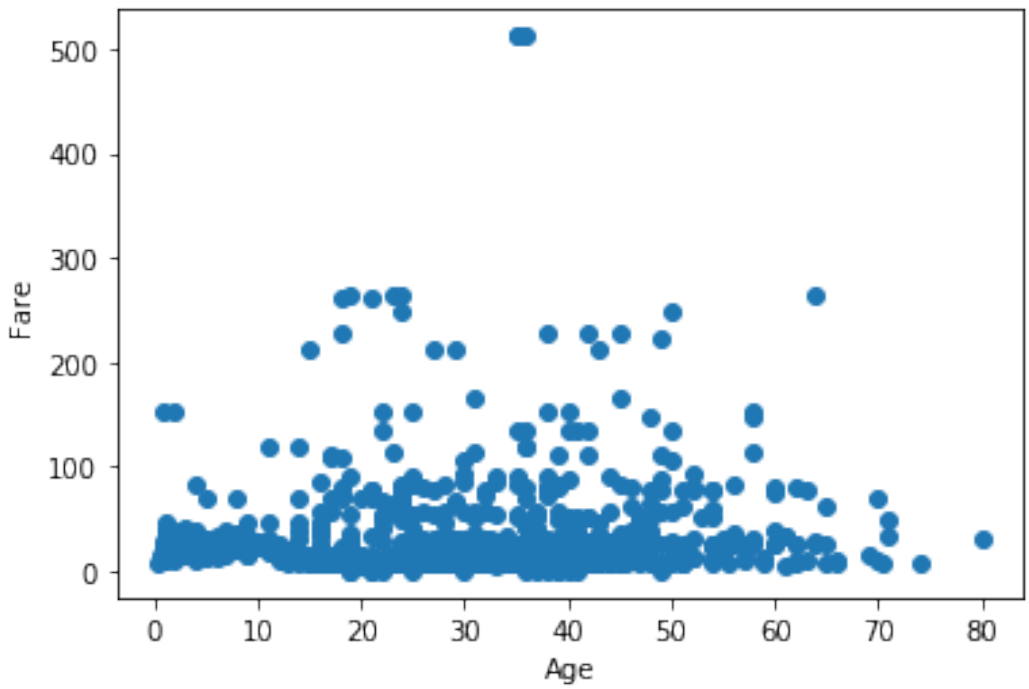
```
final=arr[arr[:,0]<18].sum()
```

```
final
```

```
56841.0777
```

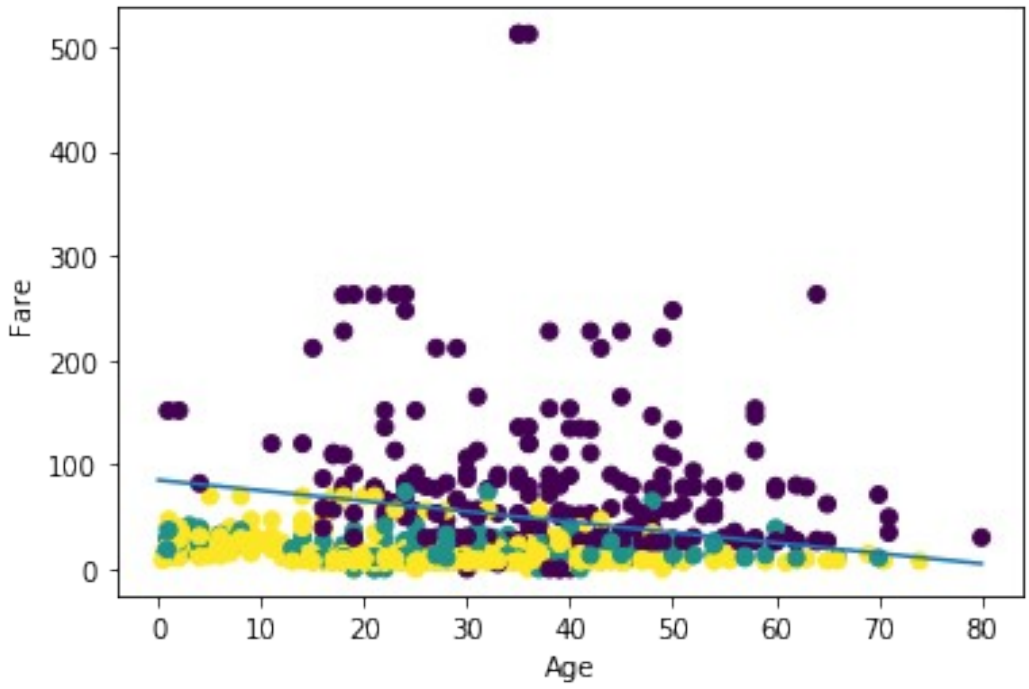
```
#SCATTER PLOT
import matplotlib.pyplot as plt
c=plt.scatter(df["Age"],df["Fare"])
plt.xlabel("Age")
plt.ylabel("Fare")
```

```
Text(0, 0.5, 'Fare')
```



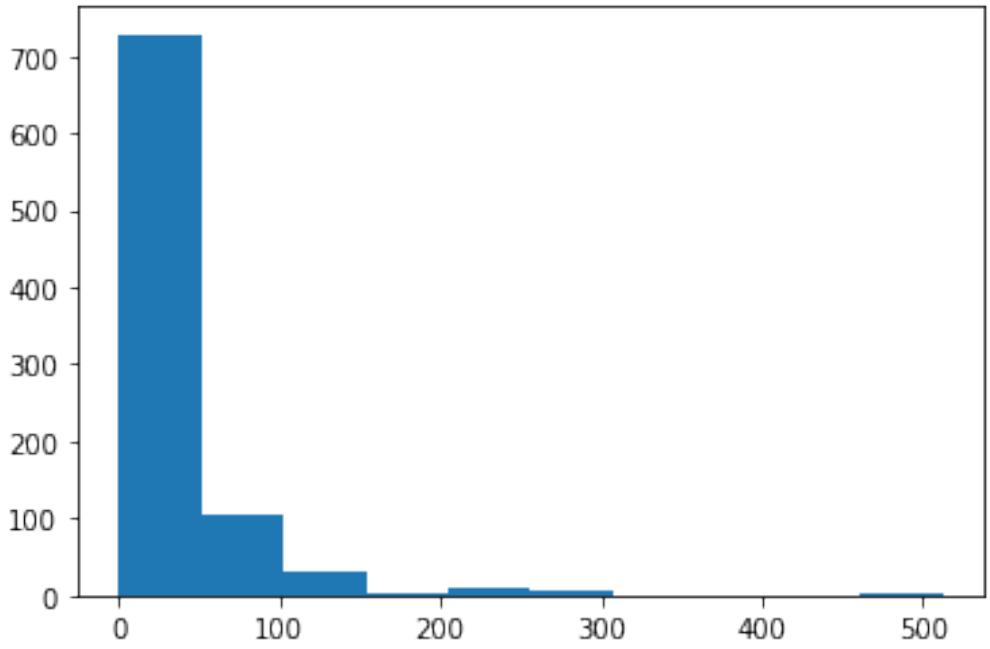
```
#SCATTER PLOT
import matplotlib.pyplot as plt
c=plt.scatter(df["Age"],df["Fare"],c=df["Pclass"])
plt.xlabel("Age")
plt.ylabel("Fare")
plt.plot([0,80],[85,5])
c
```

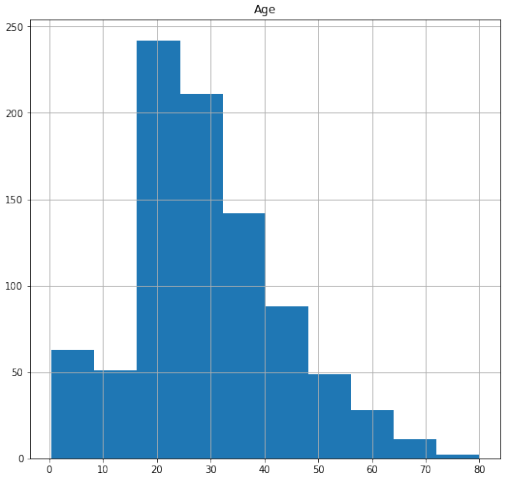
```
<matplotlib.collections.PathCollection at 0x8b2a0f0>
```

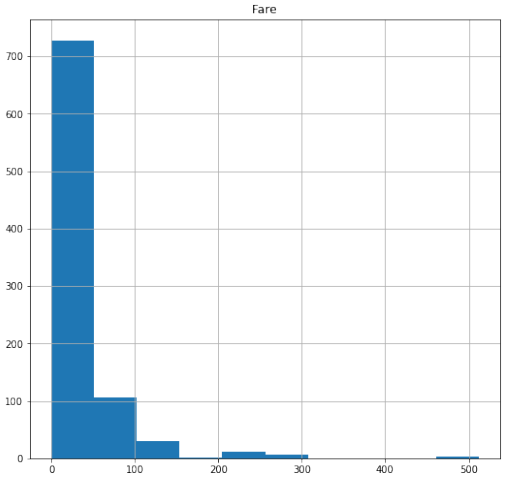


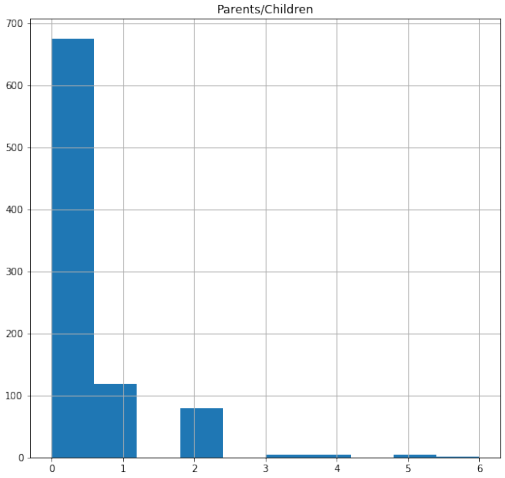

```
#HISTOGRAM
import matplotlib.pyplot as plt
d=plt.hist(df["Fare"])
e=df.hist(figsize=(20,30))
print(d)
print(e)
```

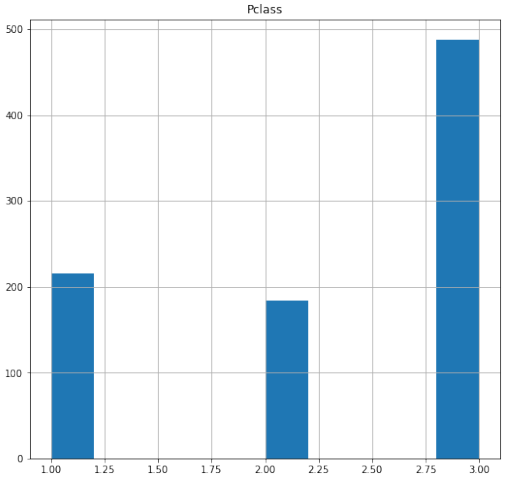
```
(array([728., 106.,  31.,   2.,  11.,   6.,   0.,   0.,   0.,   3.]),
array([  0.     ,  51.23292, 102.46584, 153.69876, 204.93168, 256.1646
,
       307.39752, 358.63044, 409.86336, 461.09628, 512.3292 ]), <a
list of 10 Patch objects>)
[[<matplotlib.axes._subplots.AxesSubplot object at 0x08BB8B70>
  <matplotlib.axes._subplots.AxesSubplot object at 0x08BD7BF0>]
 [<matplotlib.axes._subplots.AxesSubplot object at 0x08BF7CD0>
  <matplotlib.axes._subplots.AxesSubplot object at 0x08C18DB0>]
 [<matplotlib.axes._subplots.AxesSubplot object at 0x08C3AE90>
  <matplotlib.axes._subplots.AxesSubplot object at 0x08C5AF70>]]
```

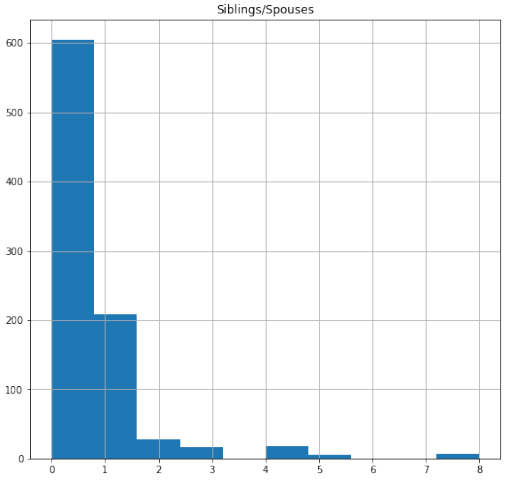

700
600
500
400
300
200
100
0
0
100
200
300
400
500

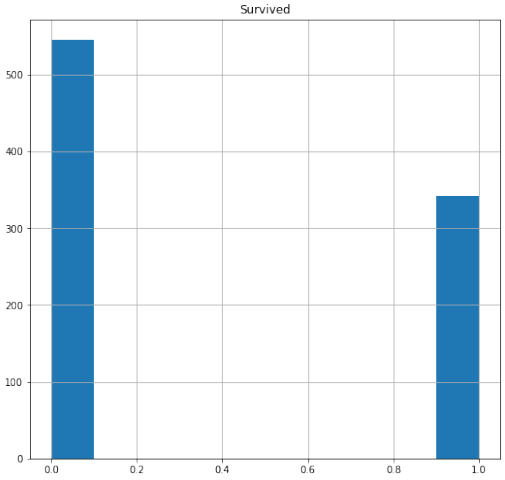








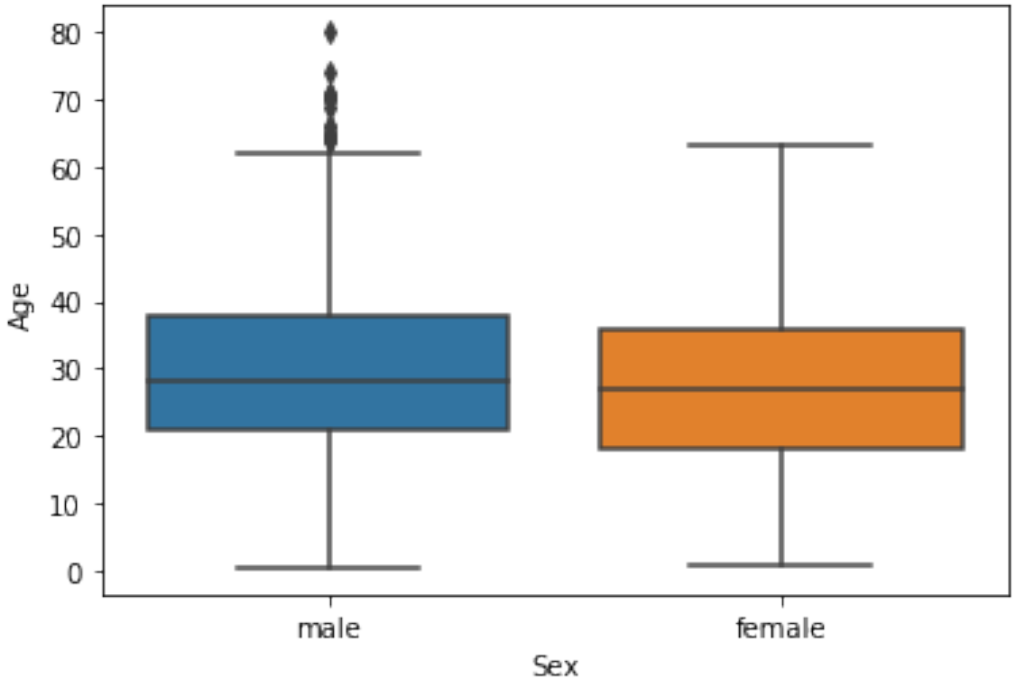





```
#BOX PLOT
import seaborn as sns
```

```
g=sns.boxplot(x="Sex",y="Age", data=df)
g
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xba6e690>
```




```
#SUMMARISING DATA
import pandas as pd
h=pd.crosstab(df["Fare"],df["Sex"])
h
```

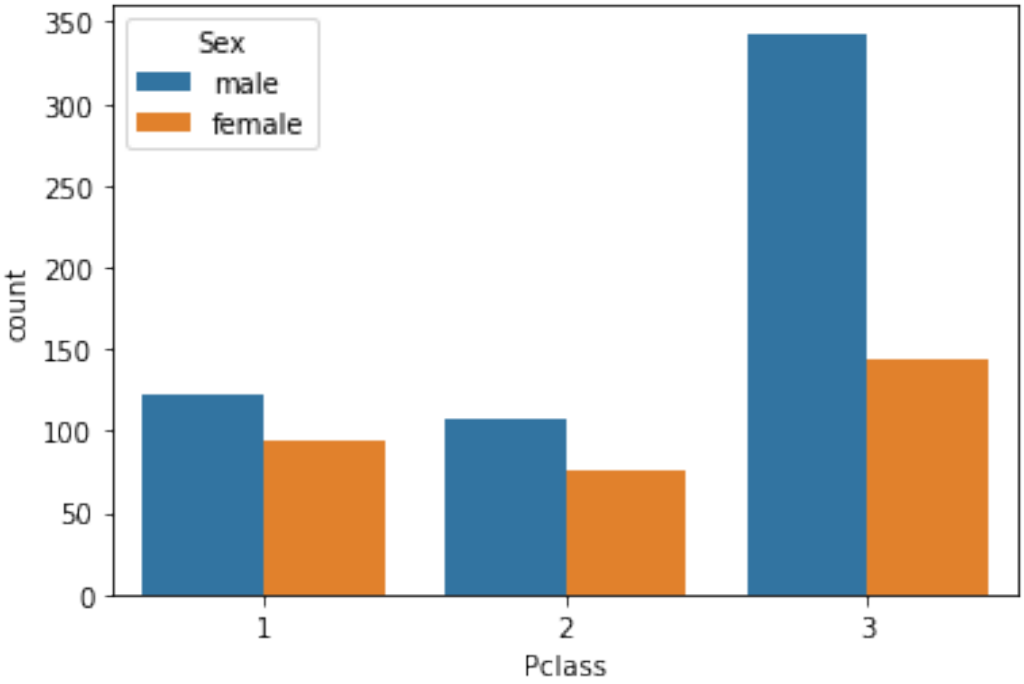
```
Sex       female  male
Fare
0.0000         0    15
4.0125         0     1
5.0000         0     1
6.2375         0     1
6.4375         0     1
...          ...   ...
227.5250       3     1
247.5208       1     1
262.3750       2     0
263.0000       2     2
512.3292       1     2

[248 rows x 2 columns]
```

```
#COUNT PLOT
import seaborn as sns
```

```
i=sns.countplot(x="Pclass",hue="Sex", data=df)
i
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xbac80d0>
```




```
# PAIR PLOT
```

```
import seaborn as sns
j=sns.pairplot(df)
j
```

```
<seaborn.axisgrid.PairGrid at 0xbb29990>
```


```
print(df.head())
```

```
   Survived  Pclass     Sex   Age  Siblings/Spouses  Parents/Children
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
```

```
import pandas as pd
df["male"]=df["Sex"]=="male"
df.head()
```

```
   Survived  Pclass     Sex   Age  Siblings/Spouses  Parents/Children
\
0         0       3    male  22.0                 1                 0

1         1       1  female  38.0                 1                 0

2         1       3  female  26.0                 0                 0

3         1       1  female  35.0                 1                 0

4         0       3    male  35.0                 0                 0


      Fare   male
0   7.2500   True
1  71.2833  False
2   7.9250  False
3  53.1000  False
4   8.0500   True
```

```
k=df[["Pclass","Sex","Age","Siblings/Spouses","Parents/
Children","Fare","male"]].values
k
```

```
array([[3, 'male', 22.0, ..., 0, 7.25, True],
       [1, 'female', 38.0, ..., 0, 71.2833, False],
       [3, 'female', 26.0, ..., 0, 7.925, False],
       ...,
       [3, 'female', 7.0, ..., 2, 23.45, False],
       [1, 'male', 26.0, ..., 0, 30.0, True],
       [3, 'male', 32.0, ..., 0, 7.75, True]], dtype=object)
```

```
y=df["Survived"].values
y
```

```
array([0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0,
1,
       1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
1,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1,
1,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0,
1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0,
0,
```

1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
0,
0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0,
0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
1,
0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0,
1,
0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
0,
0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0,
0,
0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
0,
1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0,
1,
1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
0,
0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0,
0,
0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0,
1,
1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0,
0,
0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0,
0,
0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0,
0,
1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
0,
0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,
1,
0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
0,
1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
1,
0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
0,
1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
0,
0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
1,
0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0,
1,
0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0,
0,
1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
0,
0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0,
0,

```
       0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0,
1,
       1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0,
0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1,
1,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0,
       1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
0,
       1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1,
1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
0,
       1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0,
0,
       0, 0, 0, 1, 0, 1, 0], dtype=int64)
```

```
#SKLEARN MODULE
#BUILD LOGICAL REGRESSION MODEL WITH SKLEARN

from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
x=df[["Fare","Age"]].values
y=df["Survived"].values
model.fit(x,y)
print(model.coef_,model.intercept_)
```

```
[[ 0.01611624 -0.01585256]] [-0.49616551]

E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22.
Specify a solver to silence this warning.
  FutureWarning)
```

```
#MAKE PREDICTIONS WITH THE MODEL

from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
x=df[["Fare","Age","Pclass"]].values
y=df["Survived"].values
model.fit(x,y)
output=model.predict([[200.0,80.0,2]])
print(output)
print(model.predict(x[:5]))
print(y[:5])
```

```
[0]
[0 1 0 1 0]
[0 1 1 1 0]
```

```
E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22.
Specify a solver to silence this warning.
  FutureWarning)
```

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
x=df[["Age","Fare","male","Pclass","Siblings/Spouses","Parents/Childre
n"]].values
y=df["Survived"].values
model.fit(x,y)
print(model.predict(x))
```

```
print(y[:6])
```

```
[0 1 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 0 1 0 1 0 0 1 1 0 0 1 1 0 0
0 0
 0 1 1 0 1 1 1 0 0 1 0 1 0 0 1 1 0 0 1 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0
0 0
 0 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 1 0 0 1
0 1
 0 1 1 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0
1 0
 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
1 0
 1 0 0 0 1 0 1 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0
0 0
 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 1 1 0 0 0 1 0 0 0 1 1 1
1 1
 0 0 0 1 0 0 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0
0 1
 0 1 1 0 0 1 0 1 1 1 0 1 1 1 1 0 0 1 1 0 1 1 0 0 1 1 0 1 0 1 1 1 0 0 1
0 1
 0 0 1 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1
0 0
 0 1 1 1 1 1 0 0 1 1 0 1 0 0 0 1 0 1 0 0 0 1 1 0 1 0 0 1 0 0 1 0 1 0 0
0 0
 1 0 0 1 0 0 1 1 1 0 1 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 1 0 0 1 0 0 1 0
1 1
 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 0 0 0 1 0 0
0 0
 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 1 0 1
0 1
 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 0 1 0 1 1 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0
1 1
 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 1 1 0 1 1 0 0 0 1 0 0 0 0 0 1 0
0 0
```

```
 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 1 0 0
0 0
 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0
 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0
0 0
 1 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0
1 0
 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 1 0 1 0 0 1 0
1 1
 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 0 0 0 0
1 0
 0 0 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1
1 1
 1 1 0 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 1 1 0 0 0 1 1 0 1 0 0 0 0 1 1 0
0]

E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22.
Specify a solver to silence this warning.
  FutureWarning)


[0 1 1 1 0 0]
```

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
x=df[["Age","Fare","male","Pclass","Siblings/Spouses","Parents/Childre
n"]].values
y=df["Survived"].values
model.fit(x,y)
y_pred=model.predict(x)
print((y_pred).sum())
print((y==y_pred).sum())
print(y.sum())
#print((y==y_pred).sum()/y.shape[0])
#print(model.score(x,y))
#print(y.shape[0])
```

```
E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22.
Specify a solver to silence this warning.
  FutureWarning)

298
713
342
```

Breast cancer Predictor

from sklearn.datasets import load_breast_cancer cancer_data=load_breast_cancer() print(cancer_data.keys()) print(cancer_data["DESCR"])

```
import pandas as pd
from sklearn.datasets import load_breast_cancer
cancer_data=load_breast_cancer()
df=pd.DataFrame(cancer_data["data"],columns=cancer_data["feature_names
"])
print(df.head())
```

```
   mean radius  mean texture  mean perimeter  mean area  mean
smoothness  \
0        17.99         10.38          122.80     1001.0
0.11840
1        20.57         17.77          132.90     1326.0
0.08474
2        19.69         21.25          130.00     1203.0
0.10960
3        11.42         20.38           77.58      386.1
0.14250
4        20.29         14.34          135.10     1297.0
0.10030

   mean compactness  mean concavity  mean concave points  mean
symmetry  \
0           0.27760          0.3001              0.14710
0.2419
1           0.07864          0.0869              0.07017
0.1812
2           0.15990          0.1974              0.12790
0.2069
3           0.28390          0.2414              0.10520
0.2597
4           0.13280          0.1980              0.10430
0.1809

   mean fractal dimension  ...  worst radius  worst texture  worst
perimeter  \
0                 0.07871  ...         25.38          17.33
184.60
1                 0.05667  ...         24.99          23.41
158.80
2                 0.05999  ...         23.57          25.53
152.50
3                 0.09744  ...         14.91          26.50
98.87
4                 0.05883  ...         22.54          16.67
152.20
```

```
   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   worst concave points  worst symmetry  worst fractal dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 30 columns]
```

```
import pandas as pd
from sklearn.datasets import load_breast_cancer
cancer_data=load_breast_cancer()
df=pd.DataFrame(cancer_data["data"],columns=cancer_data["feature_names
"])
print(df.head())
```

```
   mean radius  mean texture  mean perimeter  mean area  mean
smoothness  \
0        17.99         10.38          122.80     1001.0
0.11840
1        20.57         17.77          132.90     1326.0
0.08474
2        19.69         21.25          130.00     1203.0
0.10960
3        11.42         20.38           77.58      386.1
0.14250
4        20.29         14.34          135.10     1297.0
0.10030

   mean compactness  mean concavity  mean concave points  mean
symmetry  \
0           0.27760          0.3001              0.14710
0.2419
1           0.07864          0.0869              0.07017
0.1812
2           0.15990          0.1974              0.12790
0.2069
3           0.28390          0.2414              0.10520
0.2597
4           0.13280          0.1980              0.10430
0.1809
```

```
   mean fractal dimension  ...  worst radius  worst texture  worst
perimeter  \
0                 0.07871  ...         25.38          17.33
184.60
1                 0.05667  ...         24.99          23.41
158.80
2                 0.05999  ...         23.57          25.53
152.50
3                 0.09744  ...         14.91          26.50
98.87
4                 0.05883  ...         22.54          16.67
152.20

   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   worst concave points  worst symmetry  worst fractal dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 30 columns]
```

```
import pandas as pd
from sklearn.datasets import load_breast_cancer
cancer_data=load_breast_cancer()
print(cancer_data["target_names"])
df["result"]=cancer_data["target"]
print(df.tail())
```

```
['malignant' 'benign']
     mean radius  mean texture  mean perimeter  mean area  mean
smoothness  \
564        21.56         22.39          142.00     1479.0
0.11100
565        20.13         28.25          131.20     1261.0
0.09780
566        16.60         28.08          108.30      858.1
0.08455
567        20.60         29.33          140.10     1265.0
0.11780
568         7.76         24.54           47.92      181.0
0.05263
```

```
     mean compactness  mean concavity  mean concave points  mean
symmetry  \
564           0.11590         0.24390              0.13890
0.1726
565           0.10340         0.14400              0.09791
0.1752
566           0.10230         0.09251              0.05302
0.1590
567           0.27700         0.35140              0.15200
0.2397
568           0.04362         0.00000              0.00000
0.1587

     mean fractal dimension  ...  worst texture  worst perimeter
worst area  \
564                 0.05623  ...          26.40           166.10
2027.0
565                 0.05533  ...          38.25           155.00
1731.0
566                 0.05648  ...          34.12           126.70
1124.0
567                 0.07016  ...          39.42           184.60
1821.0
568                 0.05884  ...          30.37            59.16
268.6

     worst smoothness  worst compactness  worst concavity  \
564           0.14100            0.21130           0.4107
565           0.11660            0.19220           0.3215
566           0.11390            0.30940           0.3403
567           0.16500            0.86810           0.9387
568           0.08996            0.06444           0.0000

     worst concave points  worst symmetry  worst fractal dimension
result
564                0.2216          0.2060                  0.07115
0
565                0.1628          0.2572                  0.06637
0
566                0.1418          0.2218                  0.07820
0
567                0.2650          0.4087                  0.12400
0
568                0.0000          0.2871                  0.07039
1

[5 rows x 31 columns]
```

```
import pandas as pd
from sklearn.datasets import load_breast_cancer
```

```
from sklearn.linear_model import LogisticRegression
cancer_data=load_breast_cancer()
df=pd.DataFrame(cancer_data["data"],columns=cancer_data["feature_names
"])
df["target"]=cancer_data["target"]
model=LogisticRegression()
x=df[cancer_data["feature_names"]].values
y=df["target"].values
model.fit(x,y)
print(model.predict([x[0]]))
print(y[0])
print(model.score(x,y))
```

```
[0]
0
0.9578207381370826
```

```
E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22.
Specify a solver to silence this warning.
  FutureWarning)
```

```
import pandas as pd
from sklearn.metrics import
accuracy_score,precision_score,recall_score,f1_score
from sklearn.linear_model import LogisticRegression
model=LogisticRegression(solver="liblinear")
df=pd.read_csv("https://sololearn.com/uploads/files/titanic.csv")
df["male"]=df["Sex"]=="male"
x=df[["Pclass","male","Age","Siblings/Spouses","Parents/Children","Far
e"]].values
y=df["Survived"].values
model.fit(x,y)
y_pred=model.predict(x)
print("Accuracy:",accuracy_score(y,y_pred))
print("Precision:",precision_score(y,y_pred))
print("Recall:",recall_score(y,y_pred))
print("F1:",f1_score(y,y_pred))
```

```
Accuracy: 0.8038331454340474
Precision: 0.7818791946308725
Recall: 0.6812865497076024
F1: 0.728125
```

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y,y_pred))
print(confusion_matrix(y_pred,y))
```

```
[[480  65]
 [109 233]]
[[480 109]
 [ 65 233]]
```

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
df=pd.read_csv("https://sololearn.com/uploads/files/titanic.csv")
df["male"]=df["Sex"]=="male"
x=df[["Pclass","male","Age","Siblings/Spouses","Parents/Children","Far
e"]].values
y=df["Survived"].values
x_train,x_test,y_train,y_test=train_test_split(x,y)
print("whole dataset:",x.shape,y.shape)
print("training dataset:",x_train.shape,y_train.shape)
print("test dataset:",x_test.shape,y_test.shape)
```

```
whole dataset: (887, 6) (887,)
training dataset: (665, 6) (665,)
test dataset: (222, 6) (222,)
```

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import
accuracy_score,precision_score,recall_score,f1_score
df=pd.read_csv("https://sololearn.com/uploads/files/titanic.csv")
df["male"]=df["Sex"]=="male"
x=df[["Pclass","male","Age","Siblings/Spouses","Parents/Children","Far
e"]].values
y=df["Survived"].values
x_train,x_test,y_train,y_test=train_test_split(x,y)
model=LogisticRegression(solver="liblinear")
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
print("Accuracy:",accuracy_score(y_test,y_pred))
print("Precision:",precision_score(y_test,y_pred))
print("Recall:",recall_score(y_test,y_pred))
print("F1:",f1_score(y_test,y_pred))
```

```
Accuracy: 0.7837837837837838
Precision: 0.8028169014084507
Recall: 0.6263736263736264
F1: 0.7037037037037038
```

```
#USING A RANDOM STATE
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import
```

```
accuracy_score,precision_score,recall_score,f1_score
df=pd.read_csv("https://sololearn.com/uploads/files/titanic.csv")
df["male"]=df["Sex"]=="male"
x=df[["Pclass","male","Age","Siblings/Spouses","Parents/Children","Far
e"]].values
y=df["Survived"].values
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=27)
model=LogisticRegression(solver="liblinear")
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
print("Accuracy:",accuracy_score(y_test,y_pred))
print("Precision:",precision_score(y_test,y_pred))
print("Recall:",recall_score(y_test,y_pred))
print("F1:",f1_score(y_test,y_pred))
```

```
Accuracy: 0.7882882882882883
Precision: 0.7368421052631579
Recall: 0.6746987951807228
F1: 0.7044025157232704
```

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_score,recall_score
from sklearn.model_selection import train_test_split
df=pd.read_csv("https://sololearn.com/uploads/files/titanic.csv")
df["male"]=df["Sex"]=="male"
x=df[["Pclass","male","Age","Siblings/Spouses","Parents/Children","Far
e"]].values
y=df["Survived"].values
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=27)
model=LogisticRegression(solver="liblinear")
model.fit(x_train,y_train)
ypred=model.predict_proba(x_test)[:,1]

print("precision:",precision_score(y_test,y_pred))
print("Recall:",recall_score(y_test,y_pred))
print("F1:",f1_score(y_test,y_pred))
```

```
precision: 0.7368421052631579
Recall: 0.6746987951807228
F1: 0.7044025157232704
```

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score,
precision_recall_fscore_support

sensitivity_score = recall_score
```

```
def specificity_score(y_true, y_pred):
    p, r, f, s = precision_recall_fscore_support(y_true, y_pred)
    return r[0]

df = pd.read_csv('https://sololearn.com/uploads/files/titanic.csv')
df['male'] = df['Sex'] == 'male'
X = df[['Pclass', 'male', 'Age', 'Siblings/Spouses',
'Parents/Children', 'Fare']].values
y = df['Survived'].values

X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=5)

model = LogisticRegression(solver="liblinear")
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("sensitivity:", sensitivity_score(y_test, y_pred))
print("specificity:", specificity_score(y_test, y_pred))
```

```
sensitivity: 0.6097560975609756
specificity: 0.9285714285714286
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier()
df = pd.read_csv('https://sololearn.com/uploads/files/titanic.csv')
df['male'] = df['Sex'] == 'male'
X = df[['Pclass', 'male', 'Age', 'Siblings/Spouses',
'Parents/Children', 'Fare']].values
y = df['Survived'].values
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=5)
model.fit(x_train,y_train)
print(model.predict([[3,True,22,1,0,7.25]]))
```

```
[0]
```