```
In [9]:

Removing all variables...
```

```
In [9]: import        as
   ...: import
   ...:
   ...: # a. Getting the data
   ...:      = "C:/Users/ujjwa/OneDrive - Centennial College/Documents/Semester_3/
Artificial_Intelligence/Assignments/Logistic_Regression/"
   ...:           = "titanic.csv"
   ...:
   ...:           =   .    .
   ...:
   ...:                =   .

In [10]:
    ...: print                 .     3
    ...: print                 .
    ...: print                 .
   PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0            1         0       3  ...   7.2500   NaN         S
1            2         1       1  ...  71.2833   C85         C
2            3         1       3  ...   7.9250   NaN         S

[3 rows x 12 columns]
(891, 12)
<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
None

In [11]: print                    'Sex' .
    ...: print                    'Pclass' .
['male' 'female']
```

```
[3 1 2]

In [12]: import                   as

In [13]:       =   .                        .                     .
    ...:      .          .    1 .      float       =0 .        ='bar'         =True
    ...:    .      '(Ujjwal) Passenger Class vs Passenger Survival'
    ...:    .       'Passenger Class (1st, 2nd, and 3rd)'
    ...:    .       'Passenger Survival (1 = survived)'
Out[13]: Text(0, 0.5, 'Passenger Survival (1 = survived)')

In [14]:       =   .                        .                  .
    ...:      .          .    1 .      float       =0 .        ='bar'         =True
    ...:    .      '(Ujjwal) Passenger Gender vs Passenger Survival'
    ...:    .       'Passenger Gender (Male and Female)'
    ...:    .       'Passenger Survival (1 = survived)'
Out[14]: Text(0, 0.5, 'Passenger Survival (1 = survived)')

In [15]:                  =  'Survived'  'Sex'  'Pclass'  'Fare'  'SibSp'  'Parch'
    ...:            =
    ...:   .        .                                =0.2         = 12  12
        ='hist'
    ...:    .

In [16]:                 =  'PassengerId'  'Name'  'Ticket'  'Cabin'
    ...: for   in
    ...:                   .            =            =True
    ...: print
     Survived  Pclass     Sex   Age  SibSp  Parch     Fare Embarked
0           0       3    male  22.0      1      0   7.2500        S
1           1       1  female  38.0      1      0  71.2833        C
2           1       3  female  26.0      0      0   7.9250        S
3           1       1  female  35.0      1      0  53.1000        S
4           0       3    male  35.0      0      0   8.0500        S
..        ...     ...     ...   ...    ...    ...      ...      ...
886         0       2    male  27.0      0      0  13.0000        S
887         1       1  female  19.0      0      0  30.0000        S
888         0       3  female   NaN      1      2  23.4500        S
889         1       1    male  26.0      0      0  30.0000        C
890         0       3    male  32.0      0      0   7.7500        Q

[891 rows x 8 columns]

In [17]:                    =  'Sex'  'Embarked'
    ...: for       in
    ...:                    = 'value' + '_' +
    ...:     print
    ...:                    =   .                                          =
    ...:                      =                .
    ...:                    =
    ...:
    ...: for   in
    ...:                   .            =            =True
    ...:
    ...: print
```

```
value_Sex
value_Embarked
     Survived  Pclass   Age  ...  Embarked_C  Embarked_Q  Embarked_S
0           0       3  22.0  ...           0           0           1
1           1       1  38.0  ...           1           0           0
2           1       3  26.0  ...           0           0           1
3           1       1  35.0  ...           0           0           1
4           0       3  35.0  ...           0           0           1
..        ...     ...   ...  ...         ...         ...         ...
886         0       2  27.0  ...           0           0           1
887         1       1  19.0  ...           0           0           1
888         0       3   NaN  ...           0           0           1
889         1       1  26.0  ...           1           0           0
890         0       3  32.0  ...           0           1           0

[891 rows x 11 columns]

In [18]:          =                'Age' .
    ...:                    'Age'  =                'Age' .

In [19]:                =               .       float

In [20]:               .
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Survived    891 non-null    float64
 1   Pclass      891 non-null    float64
 2   Age         891 non-null    float64
 3   SibSp       891 non-null    float64
 4   Parch       891 non-null    float64
 5   Fare        891 non-null    float64
 6   Sex_female  891 non-null    float64
 7   Sex_male    891 non-null    float64
 8   Embarked_C  891 non-null    float64
 9   Embarked_Q  891 non-null    float64
 10  Embarked_S  891 non-null    float64
dtypes: float64(11)
memory usage: 76.7 KB

In [21]: def normalize_dataframe
    ...:         =             -           .       /            .      -           .
    ...:     return

In [22]:                      =

In [23]: print                      .     2
   Survived  Pclass       Age  ...  Embarked_C  Embarked_Q  Embarked_S
0       0.0     1.0  0.271174  ...         0.0         0.0         1.0
1       1.0     0.0  0.472229  ...         1.0         0.0         0.0

[2 rows x 11 columns]
```

```
In [24]:                     .            = 9  10
    ...:    .

In [25]: from                         import

In [26]:                =

In [27]:          =                     .            = 'Survived'

In [28]:          =                      'Survived'

In [29]:      = 84

In [30]:                                                              =
                                              =0.3              =

In [31]: from                           import
    ...: from                      import
    ...: from                         import
    ...: import       as

In [32]:              =

In [33]:             .
Out[33]: LogisticRegression()

In [34]:             =   .          zip                      .                       .

In [35]: print
            0                        1
0      Pclass    [-2.0024965886472677]
1         Age     [-1.671489410236257]
2       SibSp    [-1.3543887932123984]
3       Parch    [-0.8021837011776475]
4        Fare     [0.4348431336707537]
5  Sex_female     [1.4779698424175023]
6    Sex_male    [-1.4775097824861738]
7  Embarked_C    [0.19737518913601654]
8  Embarked_Q  [-0.006712720850837989]
9  Embarked_S    [-0.3465841109942421]

In [36]:               = None
    ...:                   = 0

In [37]:            =   .       0.10  0.50  0.05

In [38]: for           in
    ...:     # Splitting the data into training and test sets based on the current
test_size
    ...:                                           =
         =                       =
    ...:
    ...:     # Creating and fitting the logistic regression model
    ...:                 =
    ...:                .
    ...:
    ...:     # 10-fold cross-validation and collect accuracy scores
    ...:            =                                                   =10
       ='accuracy'
    ...:
    ...:     # Calculating the minimum, mean, and maximum accuracy scores
    ...:                  =   .
    ...:                   =   .
    ...:                  =   .
    ...:
    ...:     # Printing the results for the current test size
    ...:     print f"Test Size: {         :.0%}"
    ...:     print f"Minimum Accuracy: {            :.4f}"
    ...:     print f"Mean Accuracy: {             :.4f}"
    ...:     print f"Maximum Accuracy: {            :.4f}"
    ...:     print
    ...:
    ...:     # Checking if the current test size has a better mean accuracy
    ...:     if                >
    ...:                           =
    ...:                       =
Test Size: 10%
Minimum Accuracy: 0.7125
Mean Accuracy: 0.7991
Maximum Accuracy: 0.9000

Test Size: 15%
Minimum Accuracy: 0.7237
Mean Accuracy: 0.8085
Maximum Accuracy: 0.8947

Test Size: 20%
Minimum Accuracy: 0.7361
Mean Accuracy: 0.8119
Maximum Accuracy: 0.8732

Test Size: 25%
Minimum Accuracy: 0.6866
Mean Accuracy: 0.8159
Maximum Accuracy: 0.8955

Test Size: 30%
Minimum Accuracy: 0.6935
Mean Accuracy: 0.8138
Maximum Accuracy: 0.9194

Test Size: 35%
Minimum Accuracy: 0.7241
Mean Accuracy: 0.8168
Maximum Accuracy: 0.8966

Test Size: 40%
Minimum Accuracy: 0.7037
Mean Accuracy: 0.8204
Maximum Accuracy: 0.8868
```

```
Test Size: 45%
Minimum Accuracy: 0.6531
Mean Accuracy: 0.8122
Maximum Accuracy: 0.8980


In [39]: print f"Recommended Best Test Size: {              :.0%}"
Recommended Best Test Size: 40%

In [40]:                                                              =
                                              =0.3              =

In [41]:                             =             .                                1

In [42]:                    =                             > 0.5

In [43]: from                 import


In [44]:                 =
    ...: print
0.7388059701492538

In [45]:                         =
    ...: print
[[131  31]
 [ 39  67]]

In [46]:                              =

    ...: print
              precision    recall  f1-score   support

         0.0       0.77      0.81      0.79       162
         1.0       0.68      0.63      0.66       106

    accuracy                           0.74       268
   macro avg       0.73      0.72      0.72       268
weighted avg       0.74      0.74      0.74       268


In [47]:                          =                             > 0.75

In [48]:                       =
    ...: print
0.7574626865671642

In [49]: from                 import

In [50]:
    ...:                               =

    ...: print
[[158   4]
```

```
 [ 61  45]]

In [51]:                =
           .
    ...: print f"Accuracy on Training Data: {              :.4f}"
Accuracy on Training Data: 0.8218

In [52]:
    ...:
    ...:               =
    ...: print f"Accuracy on Test Data: {             :.4f}"
Accuracy on Test Data: 0.7388

In [53]:
    ...: if                  >
    ...:     print "Accuracy on Test Data is higher than Training Data."
    ...: else
    ...:     print "Accuracy on Training Data is higher than Test Data."
Accuracy on Training Data is higher than Test Data.

In [54]:
    ...:                  =
    ...: print f"Precision at Threshold 0.5: {                :.4f}"
    ...:
    ...:              =
    ...: print f"Recall at Threshold 0.5: {             :.4f}"
    ...:
    ...: # Calcualting precision score and recall score at threshold 0.75
    ...:                        =
    ...: print f"Precision at Threshold 0.5: {                      :.4f}"
    ...:
    ...:                     =
    ...: print f"Recall at Threshold 0.5: {                   :.4f}"
Precision at Threshold 0.5: 0.6837
Recall at Threshold 0.5: 0.6321
Precision at Threshold 0.5: 0.9184
Recall at Threshold 0.5: 0.4245

In [55]:
    ...: if                    >
    ...:     print "Accuracy at Threshold 0.5 is higher than at Threshold 0.75."
    ...: elif                    <
    ...:     print "Accuracy at Threshold 0.75 is higher than at Threshold 0.5."
    ...: else
    ...:     print "Accuracy is the same at both thresholds."
Accuracy at Threshold 0.75 is higher than at Threshold 0.5.

In [56]:
    ...:
    ...: if                     >
    ...:     print "Precision at Threshold 0.5 is higher than at Threshold 0.75."
    ...: elif                     <
    ...:     print "Precision at Threshold 0.75 is higher than at Threshold 0.5."
    ...: else
    ...:     print "Precision is the same at both thresholds."
    ...:
    ...: # Compairing the recall
    ...: if                     >
    ...:     print "Recall at Threshold 0.5 is higher than at Threshold 0.75."
    ...: elif                     <
    ...:     print "Recall at Threshold 0.75 is higher than at Threshold 0.5."
    ...: else
    ...:     print "Recall is the same at both thresholds."
Precision at Threshold 0.75 is higher than at Threshold 0.5.
Recall is the same at both thresholds.

In [57]:
```