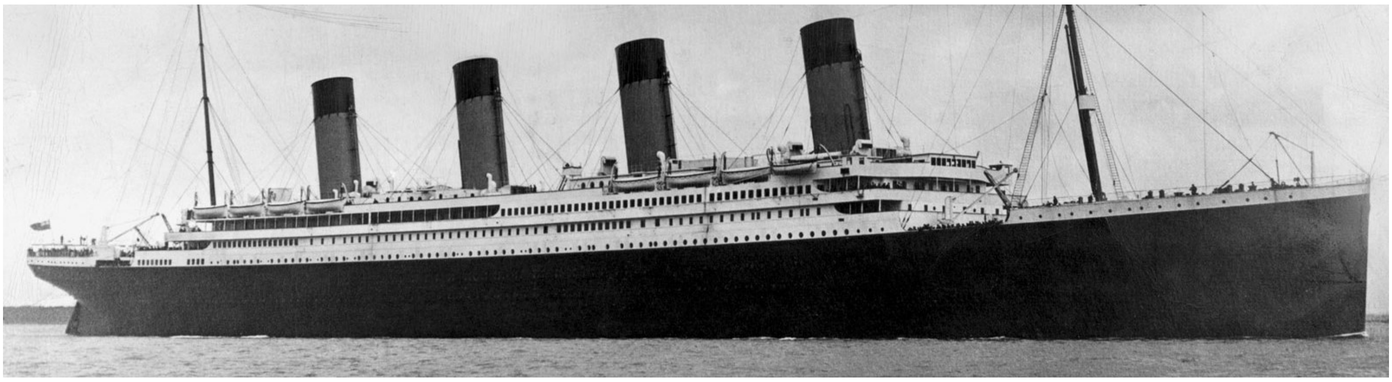
Binary Class Classification using Logistic Regression on Titanic Dataset ¶

Titanic: Machine Learning from Disaster



Data Description

Overview

The data has been split into two groups:

- training set (train.csv)
- test set (test.csv)

The **training set** should be used to build your machine learning models. For the training set, we provide the outcome (also known as the “ground truth”) for each passenger. Your model will be based on “features” like passengers’ gender and class. You can also use feature engineering to create new features.

The **test set** should be used to see how well your model performs on unseen data. For the test set, we do not provide the ground truth for each passenger. It is your job to predict these outcomes. For each passenger in the test set, use the model you trained to predict whether or not they survived the sinking of the Titanic.

We also include gender_submission.csv, a set of predictions that assume all and only female passengers survive, as an example of what a submission file should look like.

Data Dictionary

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	

Variable	Definition	Key
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Variable Notes

pclass: A proxy for socio-economic status (SES)

- 1st = Upper
- 2nd = Middle
- 3rd = Lower

age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

sibsp: The dataset defines family relations in this way...

- Sibling = brother, sister, stepbrother, stepsister
- Spouse = husband, wife (mistresses and fiancés were ignored)

parch: The dataset defines family relations in this way...

- Parent = mother, father
- Child = daughter, son, stepdaughter, stepson
- Some children travelled only with a nanny, therefore parch=0 for them.

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

In [2]:

```
df = pd.read_csv('datasets/Titanic.csv')
df.head()
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0050

In [3]:

```
df.shape
```

Out[3]:

```
(891, 12)
```

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

In [5]:

```
df.isnull().sum()
```

Out[5]:

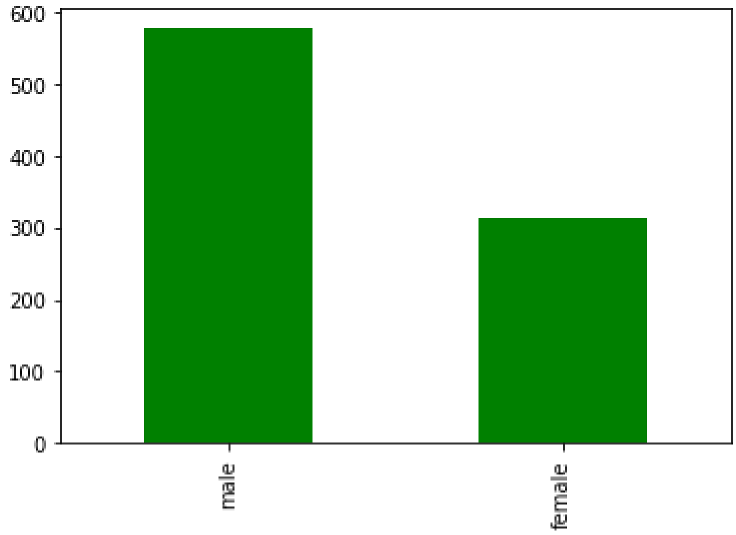
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [6]:

```
df['Sex'].value_counts().plot(kind = 'bar',color ='g')
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x284daf22208>
```



In [7]:

```
df['Embarked'].value_counts()
```

Out[7]:

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [8]:

```
df['Embarked'].fillna('S',inplace = True)
```

In [9]:

```
df['Embarked'].value_counts()
```

Out[9]:

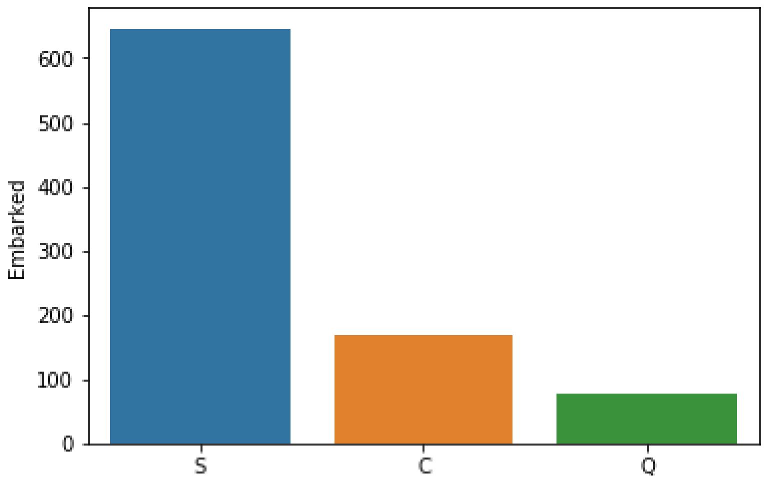
```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

In [10]:

```
sns.barplot(df['Embarked'].value_counts().index,df['Embarked'].value_counts())
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x284db6633c8>
```



In [11]:

```
df.columns
```

Out[11]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [12]:

```
df['Sex'] = pd.get_dummies(df['Sex'],drop_first=True)
```

In [13]:

```
df.head()
```

Out[13]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cal
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	N
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.9250	N
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	113803	53.1000	C1
4	5	0	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.0500	N

In [14]:

```python
from sklearn.preprocessing import LabelEncoder

lb = LabelEncoder()

lb.fit_transform(df['Sex'])
```

Out[14]:

```
array([1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1,
       0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0,
       1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1,
       0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0,
       1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0,
       1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0,
       1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1,
       1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
       1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0,
       1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0,
       0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1], dtype=int64)
```

In [15]:

```
df['Embarked'] = lb.fit_transform(df['Embarked'])
df.head()
```

Out[15]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.0500

In [16]:

```
df.columns
```

Out[16]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [17]:

```
req = df[['Survived', 'Pclass','Sex', 'Age', 'SibSp',
       'Parch', 'Fare', 'Embarked']]

cor = req.corr()
```

In [18]:

```
cor
```

Out[18]:

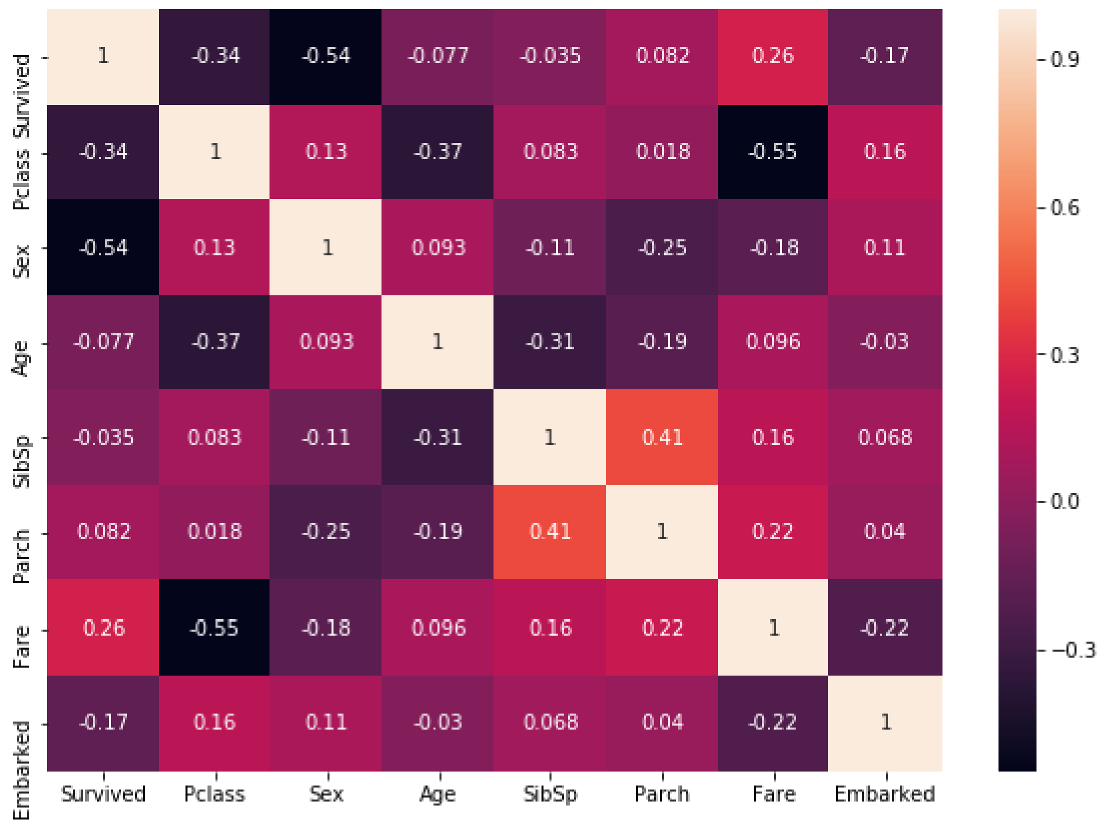
	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarke
Survived	1.000000	-0.338481	-0.543351	-0.077221	-0.035322	0.081629	0.257307	-0.16767
Pclass	-0.338481	1.000000	0.131900	-0.369226	0.083081	0.018443	-0.549500	0.16209
Sex	-0.543351	0.131900	1.000000	0.093254	-0.114631	-0.245489	-0.182333	0.10826
Age	-0.077221	-0.369226	0.093254	1.000000	-0.308247	-0.189119	0.096067	-0.03039
SibSp	-0.035322	0.083081	-0.114631	-0.308247	1.000000	0.414838	0.159651	0.06823
Parch	0.081629	0.018443	-0.245489	-0.189119	0.414838	1.000000	0.216225	0.03979
Fare	0.257307	-0.549500	-0.182333	0.096067	0.159651	0.216225	1.000000	-0.22471
Embarked	-0.167675	0.162098	0.108262	-0.030394	0.068230	0.039798	-0.224719	1.00000

In [19]:

```
plt.figure(figsize=(10,7))
sns.heatmap(cor,annot=True)
```

Out[19]:

<matplotlib.axes._subplots.AxesSubplot at 0x284db6cc780>



By observing the above corelation plot we can clearly state that there is no corelation between **Age** and other columns so we will fill the age with mean value.

In [20]:

```
req['Age'].fillna(req['Age'].mean(),inplace = True)
```

```
C:\Users\Jesus\Anaconda3\lib\site-packages\pandas\core\generic.py:6130: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  self._update_inplace(new_data)
```

In [21]:

```
req.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
Survived    891 non-null int64
Pclass      891 non-null int64
Sex         891 non-null uint8
Age         891 non-null float64
SibSp       891 non-null int64
Parch       891 non-null int64
Fare        891 non-null float64
Embarked    891 non-null int32
dtypes: float64(2), int32(1), int64(4), uint8(1)
memory usage: 46.2 KB
```

In [22]:

```
df.duplicated().sum()
```

Out[22]:

```
0
```

In [23]:

```
req[req.duplicated()]
```

Out[23]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
47	1	3	0	29.699118	0	0	7.7500	1
76	0	3	1	29.699118	0	0	7.8958	2
77	0	3	1	29.699118	0	0	8.0500	2
87	0	3	1	29.699118	0	0	8.0500	2
95	0	3	1	29.699118	0	0	8.0500	2
101	0	3	1	29.699118	0	0	7.8958	2
121	0	3	1	29.699118	0	0	8.0500	2
133	1	2	0	29.000000	1	0	26.0000	2
173	0	3	1	21.000000	0	0	7.9250	2
196	0	3	1	29.699118	0	0	7.7500	1
198	1	3	0	29.699118	0	0	7.7500	1
201	0	3	1	29.699118	8	2	69.5500	2
213	0	2	1	30.000000	0	0	13.0000	2
223	0	3	1	29.699118	0	0	7.8958	2
241	1	3	0	29.699118	1	0	15.5000	1
260	0	3	1	29.699118	0	0	7.7500	1
274	1	3	0	29.699118	0	0	7.7500	1
295	0	1	1	29.699118	0	0	27.7208	0
300	1	3	0	29.699118	0	0	7.7500	1
304	0	3	1	29.699118	0	0	8.0500	2
313	0	3	1	28.000000	0	0	7.8958	2
320	0	3	1	22.000000	0	0	7.2500	2
324	0	3	1	29.699118	8	2	69.5500	2
335	0	3	1	29.699118	0	0	7.8958	2
343	0	2	1	25.000000	0	0	13.0000	2
354	0	3	1	29.699118	0	0	7.2250	0
355	0	3	1	28.000000	0	0	9.5000	2
358	1	3	0	29.699118	0	0	7.8792	1
359	1	3	0	29.699118	0	0	7.8792	1
364	0	3	1	29.699118	1	0	15.5000	1
...	...	...	...	...	...	...	...	...
692	1	3	1	29.699118	0	0	56.4958	2
696	0	3	1	44.000000	0	0	8.0500	2
709	1	3	1	29.699118	1	1	15.2458	0

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
732	0	2	1	29.699118	0	0	0.0000	2
733	0	2	1	23.000000	0	0	13.0000	2
734	0	2	1	23.000000	0	0	13.0000	2
738	0	3	1	29.699118	0	0	7.8958	2
739	0	3	1	29.699118	0	0	7.8958	2
757	0	2	1	18.000000	0	0	11.5000	2
758	0	3	1	34.000000	0	0	8.0500	2
760	0	3	1	29.699118	0	0	14.5000	2
773	0	3	1	29.699118	0	0	7.2250	0
776	0	3	1	29.699118	0	0	7.7500	1
790	0	3	1	29.699118	0	0	7.7500	1
792	0	3	0	29.699118	8	2	69.5500	2
800	0	2	1	34.000000	0	0	13.0000	2
808	0	2	1	39.000000	0	0	13.0000	2
815	0	1	1	29.699118	0	0	0.0000	2
832	0	3	1	29.699118	0	0	7.2292	0
837	0	3	1	29.699118	0	0	8.0500	2
838	1	3	1	32.000000	0	0	56.4958	2
844	0	3	1	17.000000	0	0	8.6625	2
846	0	3	1	29.699118	8	2	69.5500	2
859	0	3	1	29.699118	0	0	7.2292	0
863	0	3	0	29.699118	8	2	69.5500	2
870	0	3	1	26.000000	0	0	7.8958	2
877	0	3	1	19.000000	0	0	7.8958	2
878	0	3	1	29.699118	0	0	7.8958	2
884	0	3	1	25.000000	0	0	7.0500	2
886	0	2	1	27.000000	0	0	13.0000	2

111 rows × 8 columns

In [24]:

```
Y = req['Survived']
X = req.drop('Survived',axis = 1)
```

In [25]:

```
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.3, random_state = 4
```

In [26]:

```
X.head()
```

Out[26]:

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

In [27]:

```
X_train.head()
```

Out[27]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
445	1	1	4.000000	0	2	81.8583	2
650	3	1	29.699118	0	0	7.8958	2
172	3	0	1.000000	1	1	11.1333	2
450	2	1	36.000000	1	2	27.7500	2
314	2	1	43.000000	1	1	26.2500	2

In [28]:

```
X_train.head()
```

Out[28]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
445	1	1	4.000000	0	2	81.8583	2
650	3	1	29.699118	0	0	7.8958	2
172	3	0	1.000000	1	1	11.1333	2
450	2	1	36.000000	1	2	27.7500	2
314	2	1	43.000000	1	1	26.2500	2

In [29]:

```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()

logreg.fit(X,Y)
```

```
C:\Users\Jesus\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:
433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Speci
fy a solver to silence this warning.
  FutureWarning)
```

Out[29]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [30]:

```
logreg.predict([[1,1,4.000000,0,2,81.8583,2]])
```

Out[30]:

```
array([1], dtype=int64)
```

In [31]:

```
Y_train.head(1)
```

Out[31]:

```
445    1
Name: Survived, dtype: int64
```

In [32]:

```
logreg.predict_proba([[1,1,4.000000,0,2,81.8583,2]])
```

Out[32]:

```
array([[0.3874428, 0.6125572]])
```

In [33]:

```
Y_pred = logreg.predict(X_test)
```

In [34]:

```
Y_pred
```

Out[34]:

```
array([0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 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, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0], dtype=int64)
```

In [35]:

```
from sklearn.metrics import confusion_matrix
con = confusion_matrix(Y_pred,Y_test)
con
```

Out[35]:

```
array([[138,  31],
       [ 19,  80]], dtype=int64)
```

In [36]:

```
Y_test.shape
```

Out[36]:

```
(268,)
```

In [37]:

```
con.sum()
```

Out[37]:

```
268
```

In [38]:

```
from sklearn.metrics import accuracy_score

accuracy_score(Y_pred,Y_test)
```

Out[38]:

```
0.8134328358208955
```

In [39]:

```
from sklearn.metrics import classification_report

cp = classification_report(Y_pred,Y_test)

print(cp)
```

```
              precision    recall  f1-score   support

           0       0.88      0.82      0.85       169
           1       0.72      0.81      0.76        99

   micro avg       0.81      0.81      0.81       268
   macro avg       0.80      0.81      0.80       268
weighted avg       0.82      0.81      0.82       268
```

Binary Class Classification using Logistic Regression on Breast Cancer Dataset

In [40]:

```
from sklearn.datasets import load_breast_cancer

data = load_breast_cancer()

df.keys()
```

Out[40]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [41]:

```
data.target_names
```

Out[41]:

```
array(['malignant', 'benign'], dtype='<U9')
```

In [42]:

```
data.feature_names
```

Out[42]:

```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

In [44]:

```
print(data.DESCR)
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the three
        largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 3 is Mean Radius, fie
ld
        13 is Radius SE, field 23 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign

    :Summary Statistics:

    ===================================== ====== ======
                                           Min    Max
    ===================================== ====== ======
    radius (mean):                        6.981  28.11
    texture (mean):                       9.71   39.28
    perimeter (mean):                     43.79  188.5
    area (mean):                          143.5  2501.0
    smoothness (mean):                    0.053  0.163
    compactness (mean):                   0.019  0.345
    concavity (mean):                     0.0    0.427
    concave points (mean):                0.0    0.201
    symmetry (mean):                      0.106  0.304
    fractal dimension (mean):             0.05   0.097
    radius (standard error):              0.112  2.873
    texture (standard error):             0.36   4.885
    perimeter (standard error):           0.757  21.98
    area (standard error):                6.802  542.2
    smoothness (standard error):          0.002  0.031
    compactness (standard error):         0.002  0.135
    concavity (standard error):           0.0    0.396
    concave points (standard error):      0.0    0.053
```

```
    symmetry (standard error):            0.008  0.079
    fractal dimension (standard error):   0.001  0.03
    radius (worst):                       7.93   36.04
    texture (worst):                      12.02  49.54
    perimeter (worst):                    50.41  251.2
    area (worst):                         185.2  4254.0
    smoothness (worst):                   0.071  0.223
    compactness (worst):                  0.027  1.058
    concavity (worst):                    0.0    1.252
    concave points (worst):               0.0    0.291
    symmetry (worst):                     0.156  0.664
    fractal dimension (worst):            0.055  0.208
    ===================================== ====== ======

    :Missing Attribute Values: None

    :Class Distribution: 212 - Malignant, 357 - Benign

    :Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

    :Donor: Nick Street

    :Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2 (https://goo.gl/U2Uwz2)

Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References

   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extract
ion
     for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
     Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
     San Jose, CA, 1993.
   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis
and
     prognosis via linear programming. Operations Research, 43(4), pages 570
```

```
-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techn
iques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77
(1994)
     163-171.
```

In [45]:

```python
data.data
```

Out[45]:

```
array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
        1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
        8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
        8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
        7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
        1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
        7.039e-02]])
```

In [46]:

```python
import pandas as pd

canc = pd.DataFrame(data.data,columns=data.feature_names)

canc.head()
```

Out[46]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mear symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.241
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.181
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.206
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.259
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.180

5 rows × 30 columns

In [47]:

```python
canc['output'] = data.target
```

In [48]:

```python
canc.head()
```

Out[48]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

5 rows × 31 columns

In [49]:

```python
X = canc.drop('output',axis = 1)
Y = canc['output']
```

In [50]:

```python
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.3, random_state = 4
```

In [51]:

```python
from sklearn.linear_model import LogisticRegression

lgc = LogisticRegression()

lgc.fit(X_train,Y_train)

Y_pred = lgc.predict(X_test)
```

```
C:\Users\Jesus\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:
433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Speci
fy a solver to silence this warning.
  FutureWarning)
```

In [52]:

```
from sklearn.metrics import accuracy_score,confusion_matrix

print('Confusion Matrix:\n', confusion_matrix(Y_test,Y_pred))

print('accuracy_score:',accuracy_score(Y_test,Y_pred))
```

```
Confusion Matrix:
 [[ 59   4]
 [  2 106]]
accuracy_score: 0.9649122807017544
```