Problem: Preprocessing of Titanic Dataset

Possible Model can be created out of given data to predict if a passenger survived or not.

Type of Model Required: Classification

In [3]:

```
#importing libraries

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

In [36]:

```
#load Titanic.csv

titanic_dataset = pd.read_csv('Datasets/Titanic.csv')
```

In [6]:

```
titanic_dataset.info()
```

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

In [7]:

```
titanic_dataset.head()
```

Out[7]:

	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	8.0500

In [8]:

```
#Let's check the correlation of the given columns in the dataset to see which on
e we need to drop
```

In [9]:

```
titanic_dataset.corr()
```

Out[9]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

In [11]:

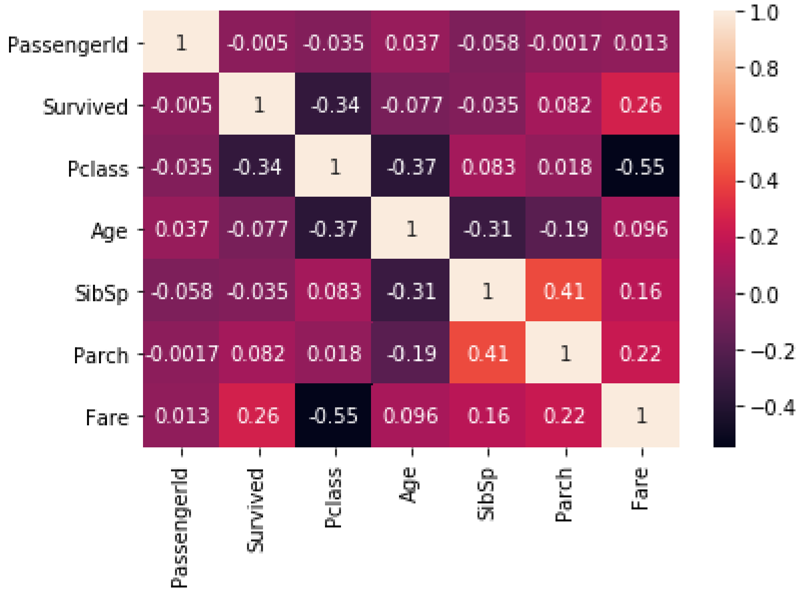
```
#Let's create a heatmap to understand better

import seaborn as sns

sns.heatmap(titanic_dataset.corr(), annot = True)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1dd8b210>
```



In [37]:

```
#Here, we can note, 'PassengerId' have very low correlation with 'survival' and
 all the other variables. We can safely drop it.
#However there is a confusion in considering 'Age', 'SibSp' and 'Parch' to be on
the safer side will be keep them.
#We can also drop 'Ticket', 'Name' as it will not matter. And drop 'Embarked' as
place of orgin will not matter. 'Cabin' number will also not matter as, we alrea
dy took 'Pclass' it will tell the cabin type of passenager.


titanic_dataset = titanic_dataset.drop(columns=['PassengerId','Name','Ticket','C
abin','Embarked'], axis=1)
```

In [38]:

```
titanic_dataset
```

Out[38]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	0	3	male	22.0	1	0	7.2500
1	1	1	female	38.0	1	0	71.2833
2	1	3	female	26.0	0	0	7.9250
3	1	1	female	35.0	1	0	53.1000
4	0	3	male	35.0	0	0	8.0500
...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000
887	1	1	female	19.0	0	0	30.0000
888	0	3	female	NaN	1	2	23.4500
889	1	1	male	26.0	0	0	30.0000
890	0	3	male	32.0	0	0	7.7500

891 rows × 7 columns

In [32]:

```
#Handling Missing Data
```

In [39]:

```
titanic_dataset.isnull().any()
```

Out[39]:

```
Survived    False
Pclass      False
Sex         False
Age          True
SibSp       False
Parch       False
Fare        False
dtype: bool
```

In [40]:

```
#Here we age have missing data, we can use the mean of the 'age' column to fill
 in

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

Out[40]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	0	3	male	22.000000	1	0	7.2500
1	1	1	female	38.000000	1	0	71.2833
2	1	3	female	26.000000	0	0	7.9250
3	1	1	female	35.000000	1	0	53.1000
4	0	3	male	35.000000	0	0	8.0500
...	...	...	...	...	...	...	...
886	0	2	male	27.000000	0	0	13.0000
887	1	1	female	19.000000	0	0	30.0000
888	0	3	female	29.699118	1	2	23.4500
889	1	1	male	26.000000	0	0	30.0000
890	0	3	male	32.000000	0	0	7.7500

891 rows × 7 columns

In [41]:

```
#Lets confirm if the missing value is taken care of

titanic_dataset.isnull().any()
```

Out[41]:

```
Survived    False
Pclass      False
Sex         False
Age         False
SibSp       False
Parch       False
Fare        False
dtype: bool
```

In [44]:

```
#Finding columns with categorical values

titanic_dataset.columns[ titanic_dataset.dtypes =='object']
```

Out[44]:

```
Index(['Sex'], dtype='object')
```

In [48]:

```
titanic_dataset
```

Out[48]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	0	3	male	22.000000	1	0	7.2500
1	1	1	female	38.000000	1	0	71.2833
2	1	3	female	26.000000	0	0	7.9250
3	1	1	female	35.000000	1	0	53.1000
4	0	3	male	35.000000	0	0	8.0500
...	...	...	...	...	...	...	...
886	0	2	male	27.000000	0	0	13.0000
887	1	1	female	19.000000	0	0	30.0000
888	0	3	female	29.699118	1	2	23.4500
889	1	1	male	26.000000	0	0	30.0000
890	0	3	male	32.000000	0	0	7.7500

891 rows × 7 columns

In [63]:

```
#Let's separate the independent and dependent variable and convert the DataFrame
into numpy array

x = titanic_dataset.iloc[:,1:].values
x
```

Out[63]:

```
array([[3, 'male', 22.0, 1, 0, 7.25],
       [1, 'female', 38.0, 1, 0, 71.2833],
       [3, 'female', 26.0, 0, 0, 7.925],
       ...,
       [3, 'female', 29.69911764705882, 1, 2, 23.45],
       [1, 'male', 26.0, 0, 0, 30.0],
       [3, 'male', 32.0, 0, 0, 7.75]], dtype=object)
```

In [51]:

```
x.shape
```

Out[51]:

```
(891, 6)
```

In [52]:

```
y = titanic_dataset.iloc[:,0].values
y
```

Out[52]:

```
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,
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, 0, 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, 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, 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])
```

In [84]:

```
y.shape
```

Out[84]:

```
(891,)
```

In [53]:

```
#Here, we see 'Sex' is a categorical value column
#Let's see how many categories are there?

titanic_dataset['Sex'].value_counts()
```

Out[53]:

```
male      577
female    314
Name: Sex, dtype: int64
```

In [65]:

```
#We have only two types of categories, so let's use Label Encoder to Encode the
 column

from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import LabelEncoder

lb = LabelEncoder()

x[:,1] = lb.fit_transform(x[:,1])
x
```

Out[65]:

```
array([[3, 1, 22.0, 1, 0, 7.25],
       [1, 0, 38.0, 1, 0, 71.2833],
       [3, 0, 26.0, 0, 0, 7.925],
       ...,
       [3, 0, 29.69911764705882, 1, 2, 23.45],
       [1, 1, 26.0, 0, 0, 30.0],
       [3, 1, 32.0, 0, 0, 7.75]], dtype=object)
```

In [75]:

```
#Train Test and Split

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y, test_size = 0.2, random_st
ate = 0)
```

In [76]:

```
x_train.shape
```

Out[76]:

```
(712, 6)
```

In [77]:

```
x_test.shape
```

Out[77]:

```
(179, 6)
```

In [78]:

```
y_train.shape
```

Out[78]:

```
(712,)
```

In [79]:

```
y_test.shape
```

Out[79]:

```
(179,)
```

In [ ]:

```

```