In [2]:
```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import pydot
%matplotlib inline
#this is for Back End of the matplotlib for better visualization
```

In [3]:
```
titanic_train = pd.read_csv("titanic_train.csv")
titanic_test = pd.read_csv("titanic_test.csv")
gen_df = pd.read_csv("gender_baseline.csv")
```

In [4]:
```
titanic_train.head()
```

Out[4]:

	passenger_id	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarke
0	1216	3	Smyth, Miss. Julia	female	NaN	0	0	335432	7.7333	NaN	
1	699	3	Cacic, Mr. Luka	male	38.0	0	0	315089	8.6625	NaN	
2	1267	3	Van Impe, Mrs. Jean Baptiste (Rosalie Paula Go...	female	30.0	1	1	345773	24.1500	NaN	
3	449	2	Hocking, Mrs. Elizabeth (Eliza Needs)	female	54.0	1	3	29105	23.0000	NaN	
4	576	2	Veal, Mr. James	male	40.0	0	0	28221	13.0000	NaN	

In [5]:
```
titanic_test
```

Out[5]:

	passenger_id	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	e
0	295	1	Thayer, Mr. John Borland Jr	male	17.0	0	2	17421	110.8833	C70	
1	1150	3	Risien, Mr. Samuel Beard	male	NaN	0	0	364498	14.5000	NaN	
2	89	1	Davidson, Mr. Thornton	male	31.0	1	0	F.C. 12750	52.0000	B71	
3	1063	3	Nirva, Mr. Iisakki Antino Aijo	male	41.0	0	0	SOTON/O2 3101272	7.1250	NaN	
4	1020	3	Minkoff, Mr. Lazar	male	21.0	0	0	349211	7.8958	NaN	
...	...	...	...	...	...	...	...	...	...	...	...
454	1194	3	Sdycoff, Mr. Todor	male	NaN	0	0	349222	7.8958	NaN	
455	403	2	Eitemiller, Mr. George Floyd	male	23.0	0	0	29751	13.0000	NaN	
456	108	1	Fleming, Miss. Margaret	female	NaN	0	0	17421	110.8833	NaN	
457	510	2	Mudd, Mr. Thomas Charles	male	16.0	0	0	S.O./P.P. 3	10.5000	NaN	
458	1265	3	Van Impe, Miss. Catharina	female	10.0	0	2	345773	24.1500	NaN	

459 rows × 14 columns

In [6]:
```
#printing their shapes

print("The shape for train is {}.\n The shape for test is {}.".format(titanic_
train.shape, titanic_test.shape))
```

```
The shape for train is (850, 15).
 The shape for test is (459, 14).
```

In [7]:
```
titanic_train.shape
```

Out[7]:
```
(850, 15)
```

In [8]:
```
gen_df
```

Out[8]:

	passenger_id	survived
0	295	0
1	1150	0
2	89	0
3	1063	0
4	1020	0
...	...	...
454	1194	0
455	403	0
456	108	1
457	510	0
458	1265	1

459 rows × 2 columns

In [9]:
```
titanic_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 850 entries, 0 to 849
Data columns (total 15 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   passenger_id  850 non-null    int64  
 1   pclass        850 non-null    int64  
 2   name          850 non-null    object 
 3   sex           850 non-null    object 
 4   age           676 non-null    float64
 5   sibsp         850 non-null    int64  
 6   parch         850 non-null    int64  
 7   ticket        850 non-null    object 
 8   fare          849 non-null    float64
 9   cabin         191 non-null    object 
 10  embarked      849 non-null    object 
 11  boat          308 non-null    object 
 12  body          73 non-null     float64
 13  home.dest     464 non-null    object 
 14  survived      850 non-null    int64  
dtypes: float64(3), int64(5), object(7)
memory usage: 99.7+ KB
```

In [10]:
```
titanic_train_copy = titanic_train.copy()
```

In [11]:
```
#del titanic_train['passenger_id']
```

In [12]:
```
titanic_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 850 entries, 0 to 849
Data columns (total 15 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   passenger_id  850 non-null    int64  
 1   pclass        850 non-null    int64  
 2   name          850 non-null    object 
 3   sex           850 non-null    object 
 4   age           676 non-null    float64
 5   sibsp         850 non-null    int64  
 6   parch         850 non-null    int64  
 7   ticket        850 non-null    object 
 8   fare          849 non-null    float64
 9   cabin         191 non-null    object 
 10  embarked      849 non-null    object 
 11  boat          308 non-null    object 
 12  body          73 non-null     float64
 13  home.dest     464 non-null    object 
 14  survived      850 non-null    int64  
dtypes: float64(3), int64(5), object(7)
memory usage: 99.7+ KB
```

In [13]:
```
df_drop = titanic_train.drop(columns=["name", "fare", "passenger_id", "body",
"home.dest", "ticket"], axis = 1)
```

In [14]:
```
df_drop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 850 entries, 0 to 849
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   pclass    850 non-null    int64  
 1   sex       850 non-null    object 
 2   age       676 non-null    float64
 3   sibsp     850 non-null    int64  
 4   parch     850 non-null    int64  
 5   cabin     191 non-null    object 
 6   embarked  849 non-null    object 
 7   boat      308 non-null    object 
 8   survived  850 non-null    int64  
dtypes: float64(1), int64(4), object(4)
memory usage: 59.9+ KB
```

In [15]:
```
df_drop = df_drop.fillna(-99999)
```

In [16]:
```
df_drop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 850 entries, 0 to 849
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   pclass    850 non-null    int64  
 1   sex       850 non-null    object 
 2   age       850 non-null    float64
 3   sibsp     850 non-null    int64  
 4   parch     850 non-null    int64  
 5   cabin     850 non-null    object 
 6   embarked  850 non-null    object 
 7   boat      850 non-null    object 
 8   survived  850 non-null    int64  
dtypes: float64(1), int64(4), object(4)
memory usage: 59.9+ KB
```

- The above dataframe tell us the submission format we need to submit in a competition

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

In [18]:
```
#setting X for our predictors and y for what we want to predict.

# X=titanic_train.drop(['survived'],axis=1)
```

In [19]:
```
# y = titanic_train['survived']
```

In [20]:
```
# #splitting with train_test_split imported earlier

# X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=42,test_size
=0.25)
```

In [21]:
```
df_drop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 850 entries, 0 to 849
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   pclass    850 non-null    int64  
 1   sex       850 non-null    object 
 2   age       850 non-null    float64
 3   sibsp     850 non-null    int64  
 4   parch     850 non-null    int64  
 5   cabin     850 non-null    object 
 6   embarked  850 non-null    object 
 7   boat      850 non-null    object 
 8   survived  850 non-null    int64  
dtypes: float64(1), int64(4), object(4)
memory usage: 59.9+ KB
```

In [22]:
```
from sklearn.preprocessing import LabelEncoder
```

In [23]:
```
lb = LabelEncoder()
```

In [24]:
```
df_drop['sex'] =lb.fit_transform(df_drop["sex"])
```

In [25]:
```
df_drop['boat'] = df_drop['boat'].replace('D', -99999)
```

In [26]:
```
df_drop['boat'] = df_drop['boat'].replace('B', -999999)
```

In [27]:
```
df_drop['boat'] = df_drop['boat'].replace('C D', -999999)
```

In [28]:
```
df_drop['boat'] = df_drop['boat'].replace('13 15 B', -999999)
```

In [29]:
```
df_drop['boat'] = df_drop['boat'].replace('A', -999999)
```

In [30]:
```
df_drop['boat'] = df_drop['boat'].replace('C', -999999)
```

In [31]:
```
df_drop['boat'] = df_drop['boat'].replace('5 7', -999999)
```

In [32]:
```
df_drop['boat'] = df_drop['boat'].replace('13 15', -999999)
```

In [33]:
```
df_drop['boat'] = df_drop['boat'].replace('5 9', -999999)
```

In [34]:
```
df_drop['boat'] = df_drop['boat'].replace('15 16', -999999)
```

In [35]:
```
df_drop["boat"] = lb.fit_transform(df_drop['boat'].astype(str))
```

In [36]:
```
df_drop['boat'].unique()
```

Out[36]:
```
array([ 6,  0, 12,  3,  1, 16, 15,  9, 14,  8, 17,  7, 11,  4,  5, 10, 13,
        2])
```

In [37]:
```
#setting X for our predictors and y for what we want to predict.

X=df_drop.drop(['survived'],axis=1)
```

In [38]:
```
X
```

Out[38]:

	pclass	sex	age	sibsp	parch	cabin	embarked	boat
0	3	0	-99999.0	0	0	-99999	Q	6
1	3	1	38.0	0	0	-99999	S	0
2	3	0	30.0	1	1	-99999	S	0
3	2	0	54.0	1	3	-99999	S	12
4	2	1	40.0	0	0	-99999	S	0
...	...	...	...	...	...	...	...	...
845	1	1	55.0	0	0	C39	S	0
846	1	1	58.0	0	0	B37	C	0
847	2	0	24.0	1	0	-99999	S	5
848	3	0	3.0	1	1	-99999	S	0
849	2	1	52.0	0	0	-99999	S	0

850 rows × 8 columns

In [39]:
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 850 entries, 0 to 849
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   pclass    850 non-null    int64
 1   sex       850 non-null    int32
 2   age       850 non-null    float64
 3   sibsp     850 non-null    int64
 4   parch     850 non-null    int64
 5   cabin     850 non-null    object
 6   embarked  850 non-null    object
 7   boat      850 non-null    int32
dtypes: float64(1), int32(2), int64(3), object(2)
memory usage: 46.6+ KB
```

In [40]:
```
df_drop['embarked'] = lb.fit_transform(df_drop['embarked'].astype(str))
```

In [41]:
```
df_drop['cabin'] = lb.fit_transform(df_drop['cabin'].astype(str))
```

In [42]:
```
df_drop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 850 entries, 0 to 849
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   pclass    850 non-null    int64  
 1   sex       850 non-null    int32  
 2   age       850 non-null    float64
 3   sibsp     850 non-null    int64  
 4   parch     850 non-null    int64  
 5   cabin     850 non-null    int32  
 6   embarked  850 non-null    int32  
 7   boat      850 non-null    int32  
 8   survived  850 non-null    int64  
dtypes: float64(1), int32(4), int64(4)
memory usage: 46.6 KB
```

In [43]:
```
X= df_drop.drop(columns='survived')
```

In [44]:
```
y = df_drop['survived']
```

In [45]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, rand
om_state = 42)
```

In [46]:
```
X_train
```

Out[46]:

	pclass	sex	age	sibsp	parch	cabin	embarked	boat
332	3	0	18.0	0	0	0	3	0
383	1	0	58.0	0	0	39	1	0
281	3	0	29.0	0	4	0	3	0
2	3	0	30.0	1	1	0	3	0
231	1	1	47.0	1	0	71	1	0
...	...	...	...	...	...	...	...	...
71	2	1	39.0	0	0	0	3	0
106	2	1	36.0	0	0	0	3	0
270	3	1	35.0	0	0	0	3	0
435	3	1	28.0	0	0	0	3	0
102	3	1	-99999.0	0	0	0	3	0

680 rows × 8 columns

In [47]:
```
#Let's start by using Logistic Regression model since the problem is about cla
ssification
from sklearn.linear_model import LogisticRegression
#instanciating the model
lr = LogisticRegression(max_iter=100000)
```

In [48]:
```
#fitting our model
lr.fit(X_train, y_train)
```

Out[48]:
```
LogisticRegression(max_iter=100000)
```

In [49]:
```
lr_pred = lr.predict(X_test)
```

In [50]:
```
lr_pred
```

Out[50]:
```
array([0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0,
       0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1], dtype=int64)
```

In [51]:
```
from sklearn.metrics import accuracy_score
```

In [52]:
```
acc = accuracy_score(y_test, lr_pred)
```

In [53]:
```
acc
```

Out[53]:
```
0.9176470588235294
```

In [54]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [55]:
```
knn = KNeighborsClassifier()
```

In [56]:
```
knn.fit(X_train, y_train)
```

Out[56]:
```
KNeighborsClassifier()
```

In [57]:
```
knn_pred = knn.predict(X_test)
```

In [58]:
```
from sklearn.metrics import accuracy_score
```

In [59]:
```
knn_acc = accuracy_score(y_test, knn_pred)
```

In [60]:
```
knn_acc
```

Out[60]:
```
0.9058823529411765
```

In [61]:
```
from sklearn.svm import SVC
```

In [62]:
```
svm = SVC(kernel='linear')
```

In [63]:
```
svm.fit(X_train, y_train)
```

Out[63]:
```
SVC(kernel='linear')
```

In [65]:
```
svm_pred = svm.predict(X_test)
```

In [66]:
```
acc_svm = accuracy_score(y_test, svm_pred)
```

In [67]:
```
acc_svm
```

Out[67]:
```
0.9529411764705882
```

In [68]:
```
from sklearn.metrics import auc
```

In [72]:
```
from sklearn.metrics import precision_recall_curve
```

In [74]:
```
pre_rec_cur = precision_recall_curve(y_test, svm_pred )
```

In [75]:
```
pre_rec_cur
```

Out[75]:
```
(array([0.4       , 0.98387097, 1.        ]),
 array([1.        , 0.89705882, 0.        ]),
 array([0, 1], dtype=int64))
```

In [ ]: