

# Logistic Regression-Titanic Survivors Project

Nawal Obaid

9/6/2022

## PROBLEM STATEMENT

The sinking of the Titanic on April 15th, 1912 is one of the most tragic tragedies in history. The Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers. The numbers of survivors were low due to the lack of lifeboats for all passengers and crew. Some passengers were more likely to survive than others, such as women, children, and upper-class. This case study analyzes what sorts of people were likely to survive this tragedy. The dataset includes the following:

- 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
- Ticket: Ticket number
- Fare: Passenger fare
- Cabin: Cabin number
- Embarked: Port of Embarkation C = Cherbourg, Q = Queenstown, S = Southampton
- Target class: Survived: Survival (0 = No, 1 = Yes)

**DATA SOURCE:** <https://www.kaggle.com/c/titanic>  
(<https://www.kaggle.com/c/titanic>)

## STEP #0: LIBRARIES IMPORT

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```


## STEP #1: IMPORT DATASET

```
In [2]: # read the data using pandas dataframe
titanic = pd.read_csv('Train_Titanic.csv')
```

```
In [3]: # Show the data head!
titanic.head()
# the target class is the survived column
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Na
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C8
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	Na
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C12
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Na



```
In [4]: # Show the data tail!
titanic.tail()
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN

***We can not train a model with dataset has NaN values!***

```
In [5]: titanic.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 [6]: titanic.describe()
```

Out[6]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

## STEP #2: EXPLORE/VISUALIZE DATASET

```
In [7]: # Let's count the number of survivors and non-survivors
survived = titanic[titanic['Survived']==1]
no_survived = titanic[titanic['Survived']==0]
```

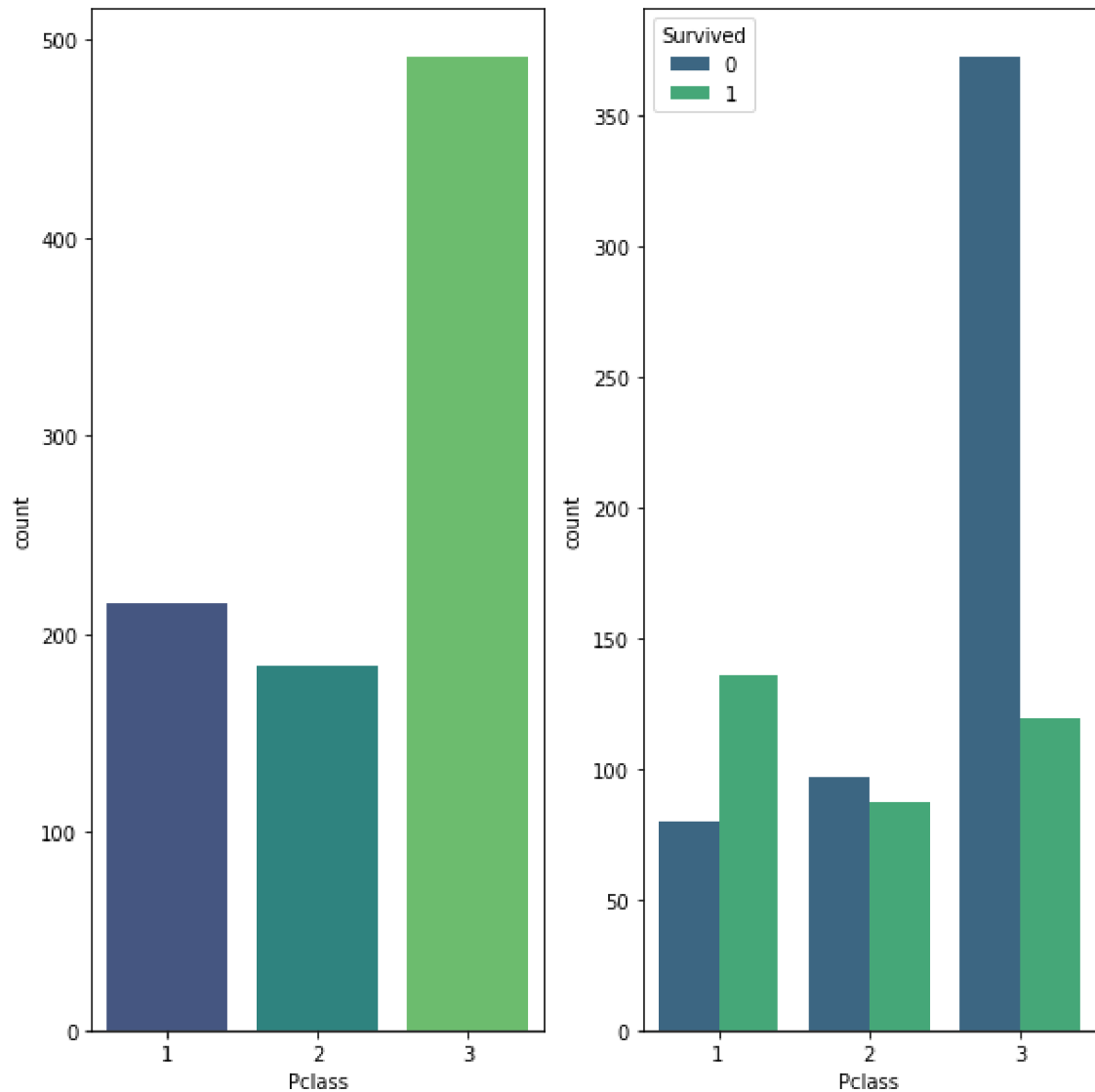
```
In [8]: # Count the survived and deceased
print("Total =", len(titanic))

print("Number of Survived passengers =", len(survived))
print("Percentage Survived =", len(survived)/len(titanic)*100.0, "%")

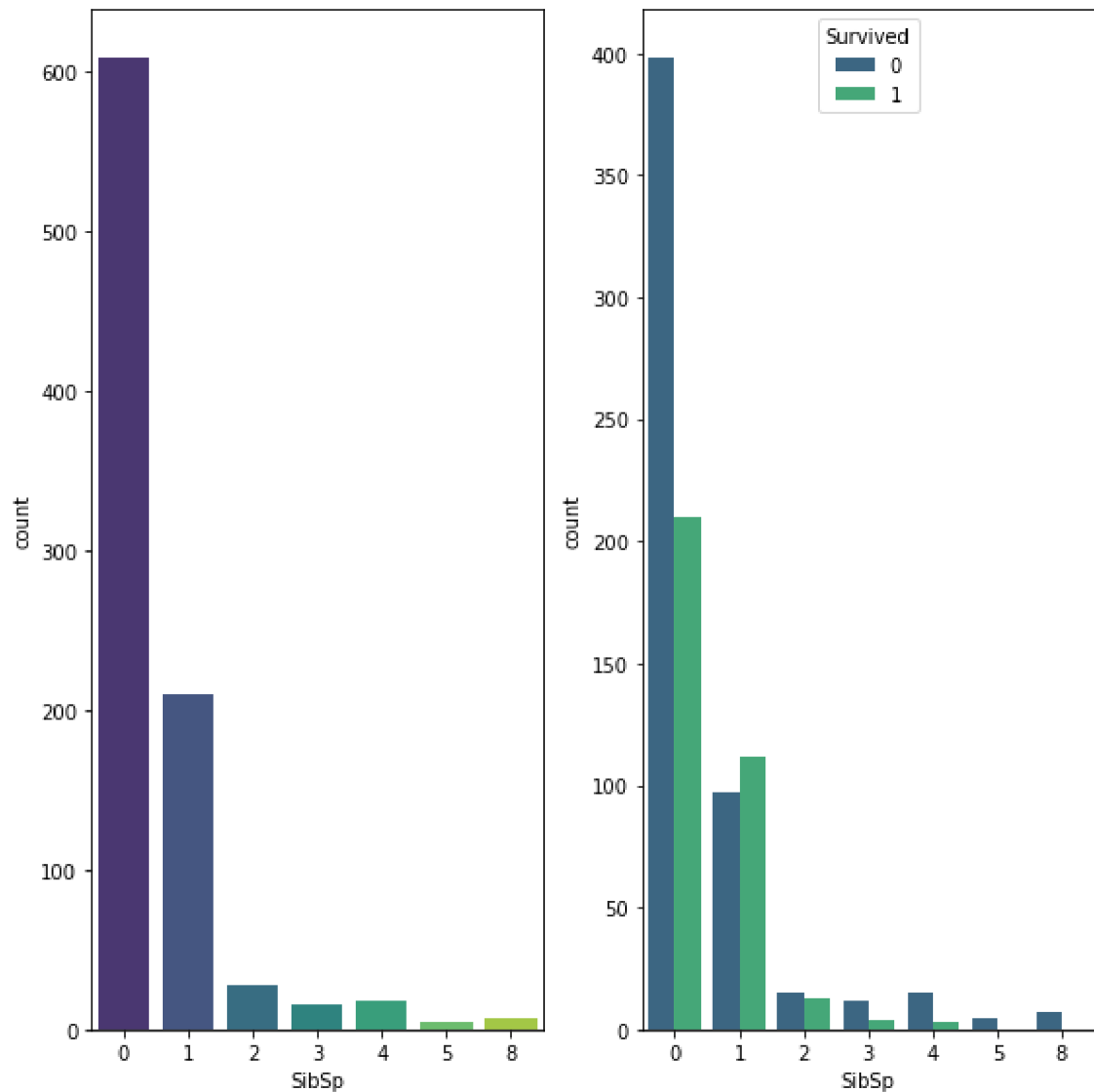
print("Did not Survive =", len(no_survived))
print("Percentage who did not survive =", len(no_survived)/len(titanic)*100.0, "%")
```

```
Total = 891
Number of Survived passengers = 342
Percentage Survived = 38.38383838383838 %
Did not Survive = 549
Percentage who did not survive = 61.61616161616161 %
```

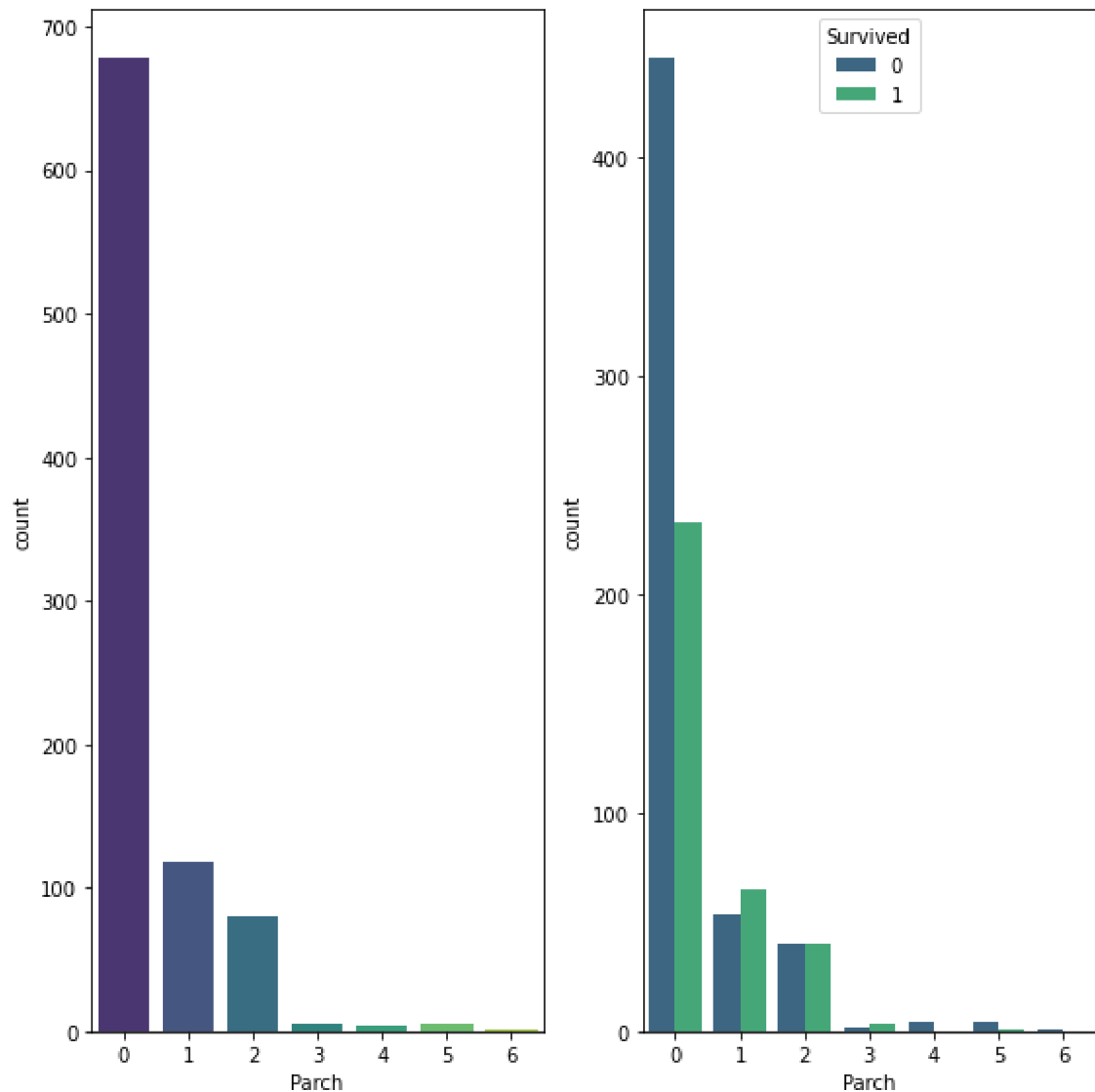
```
In [9]: # Bar Chart to indicate the number of people survived based on their class
# If you are a first class, you have a higher chance of survival
plt.figure(figsize=[8, 8])
# to view more than one plot on the grid
plt.subplot(1,2,1)
sns.countplot(x = 'Pclass', data = titanic, palette='viridis')
plt.subplot(1,2,2)
sns.countplot(x = 'Pclass', data = titanic, hue = 'Survived', palette='viridis')
# prevent overlapping
plt.tight_layout()
```



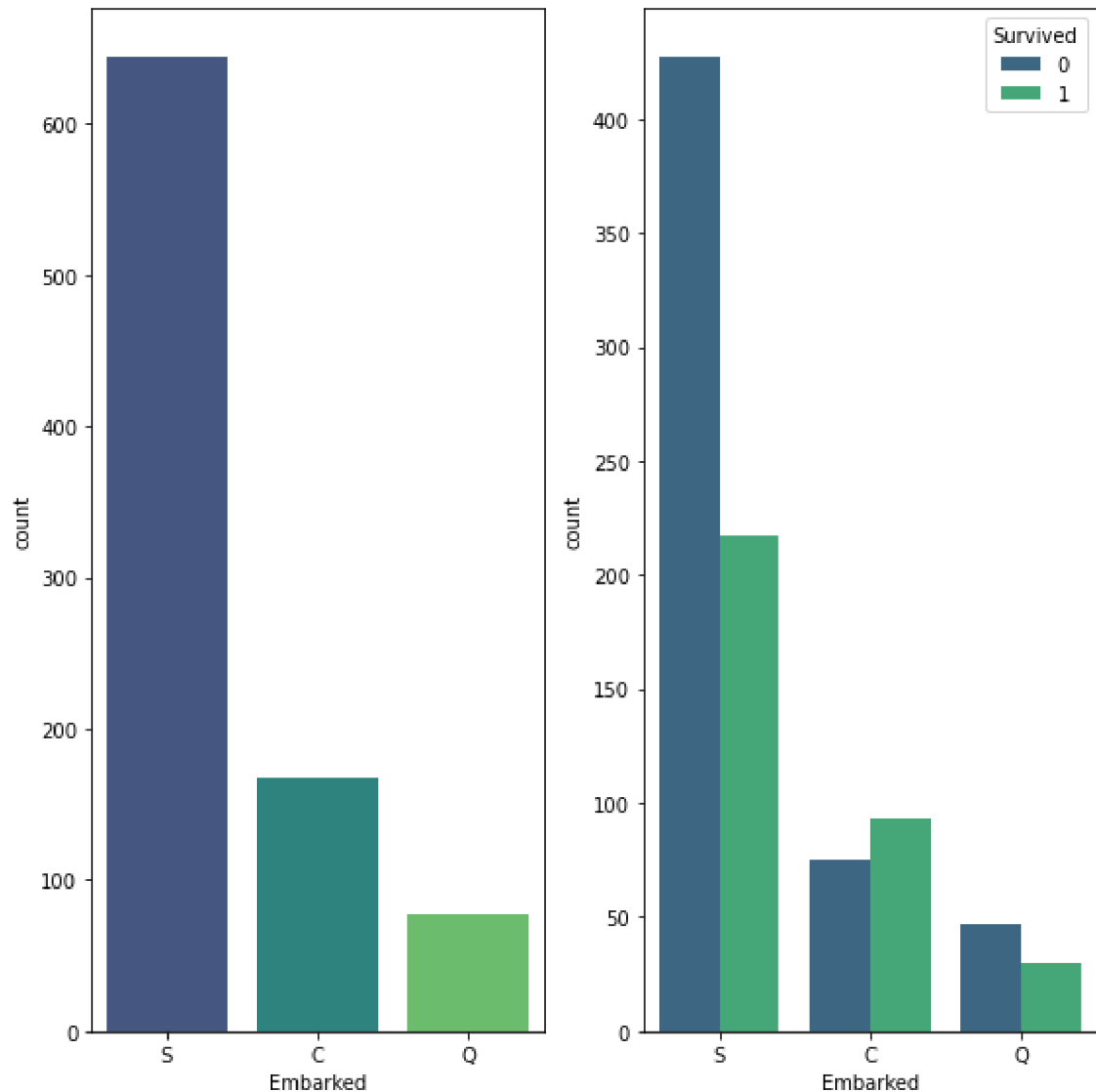
```
In [10]: # Bar Chart to indicate the number of people survived based on their siblings status
# If you have 1 siblings (SibSp = 1), you have a higher chance of survival compared to others
plt.figure(figsize=[8, 8])
# to view more than one plot on the grid
plt.subplot(1,2,1)
sns.countplot(x = 'SibSp', data = titanic, palette='viridis')
plt.subplot(1,2,2)
sns.countplot(x = 'SibSp', data = titanic, hue = 'Survived', palette='viridis')
# prevent overlapping
plt.tight_layout()
# majority of people have no siblings/spouse
```



```
In [11]: # Bar Chart to indicate the number of people survived based on their Parch status
# If you have 1, 2, or 3 family members (Parch = 1,2), you have a higher chance of survival
plt.figure(figsize=[8, 8])
# to view more than one plot on the grid
plt.subplot(1,2,1)
sns.countplot(x = 'Parch', data = titanic, palette='viridis')
plt.subplot(1,2,2)
sns.countplot(x = 'Parch', data = titanic, hue = 'Survived', palette='viridis')
# prevent overlapping
plt.tight_layout()
```

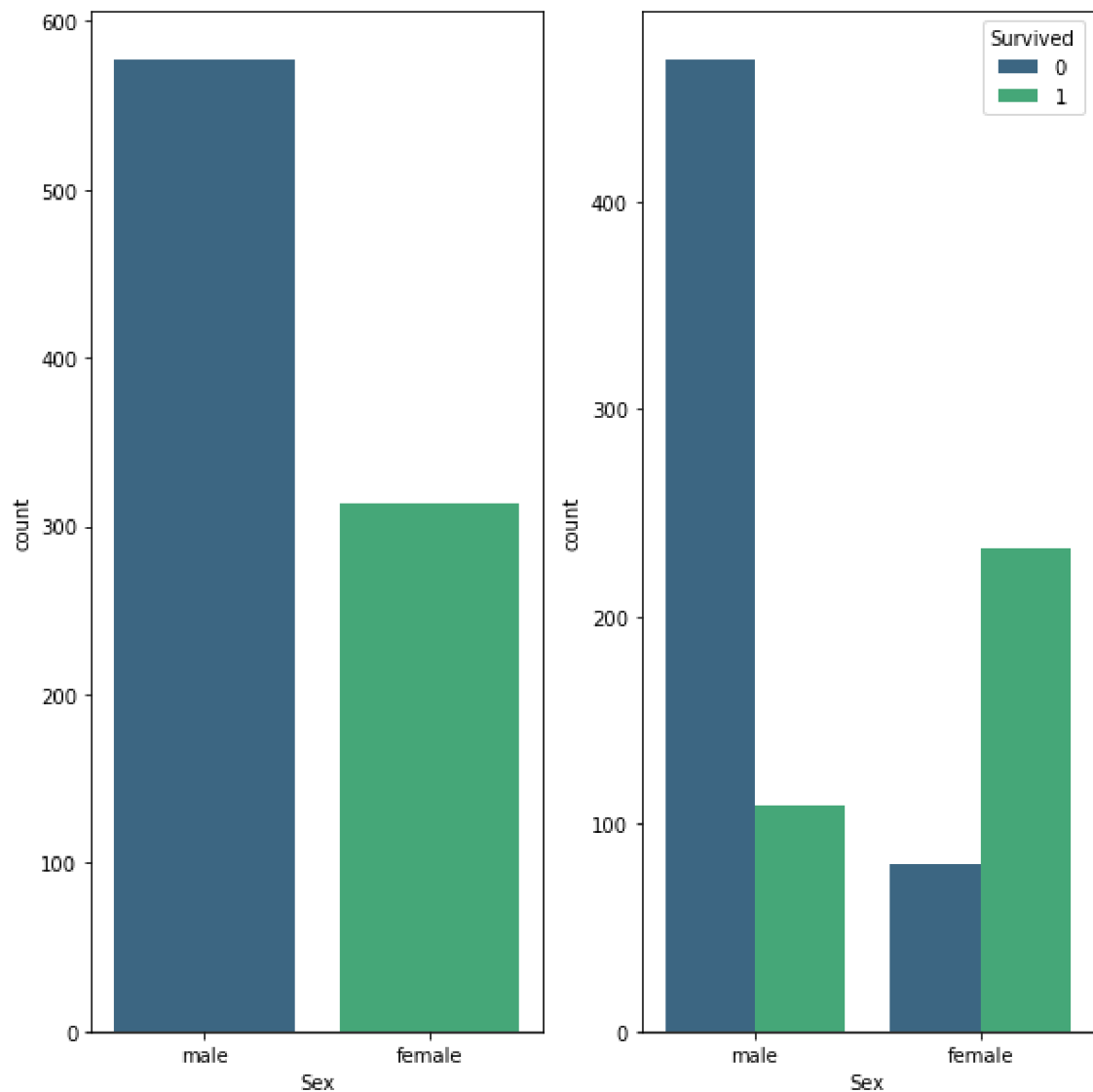


```
In [12]: # Bar Chart to indicate the number of people survived based on the port they embarked
# Port of Embarkation C = Cherbourg, Q = Queenstown, S = Southampton
# If you embarked from port "C", you have a higher chance of survival compared to
plt.figure(figsize=[8, 8])
# to view more than one plot on the grid
plt.subplot(1,2,1)
sns.countplot(x = 'Embarked', data = titanic, palette='viridis')
plt.subplot(1,2,2)
sns.countplot(x = 'Embarked', data = titanic, hue = 'Survived', palette='viridis')
# prevent overlapping
plt.tight_layout()
```



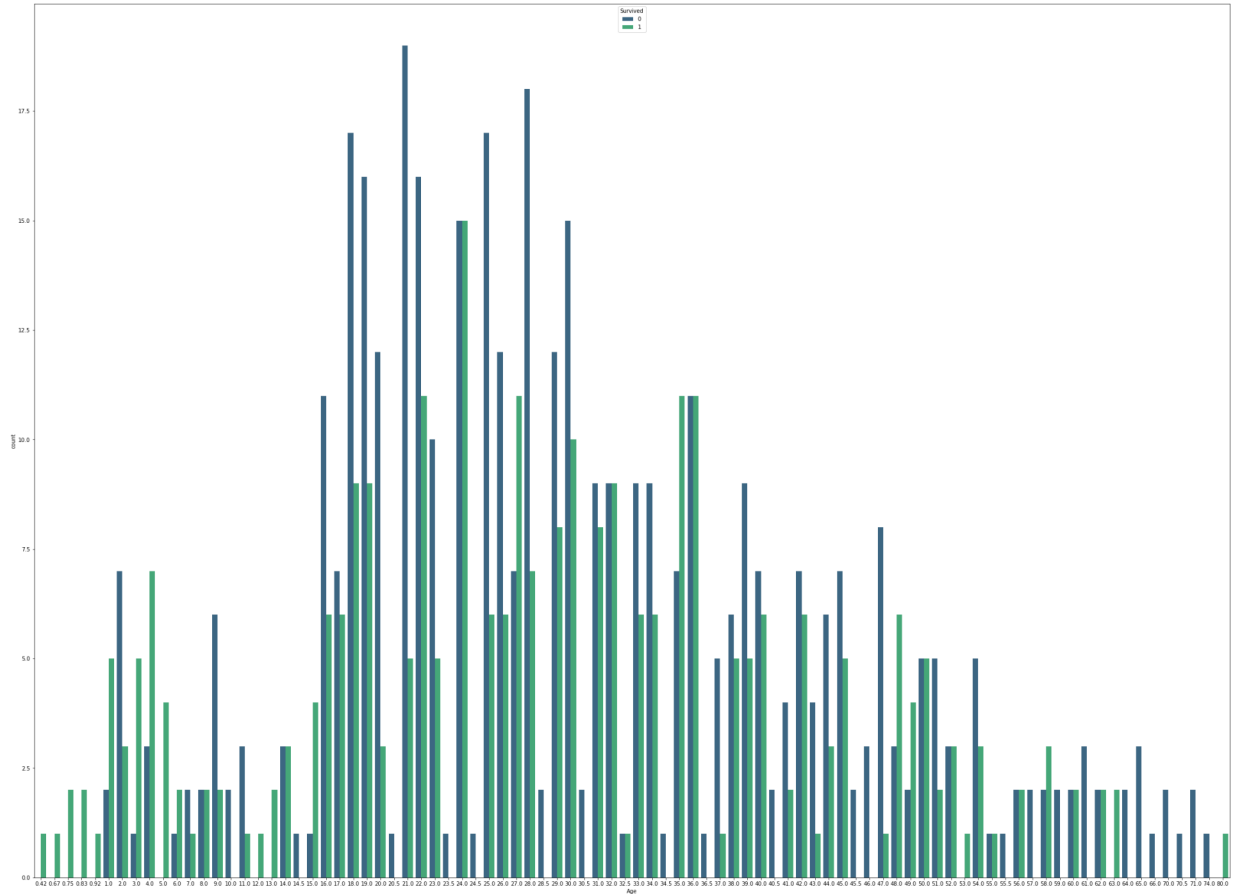


```
In [13]: # Bar Chart to indicate the number of people survived based on their sex
# If you are a female, you have a higher chance of survival compared to other por
plt.figure(figsize=(8, 8))
# to view more than one plot on the grid
plt.subplot(1,2,1)
sns.countplot(x = 'Sex', data = titanic, palette='viridis')
plt.subplot(1,2,2)
sns.countplot(x = 'Sex', data = titanic, hue = 'Survived', palette='viridis')
# prevent overlapping
plt.tight_layout()
```



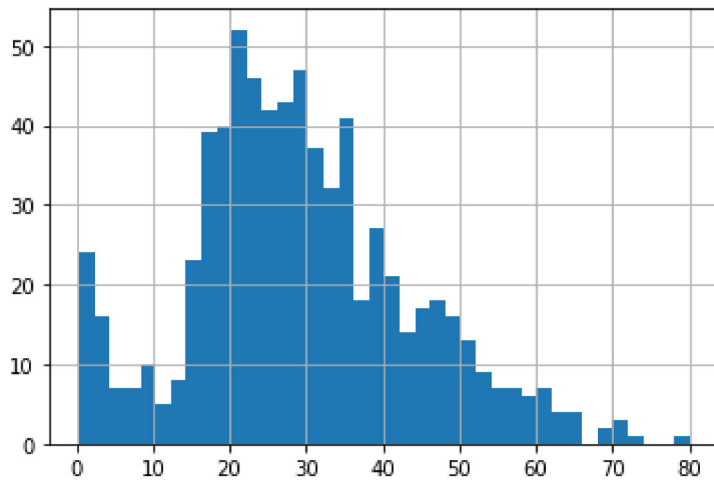
```
In [14]: # Bar Chart to indicate the number of people survived based on their age
# If you are a baby, you have a higher chance of survival
plt.figure(figsize=(40,30))
sns.countplot(x = 'Age', hue = 'Survived', data=titanic, palette='viridis')
# not a great plot! should try a different one
```

Out[14]: <AxesSubplot:xlabel='Age', ylabel='count'>



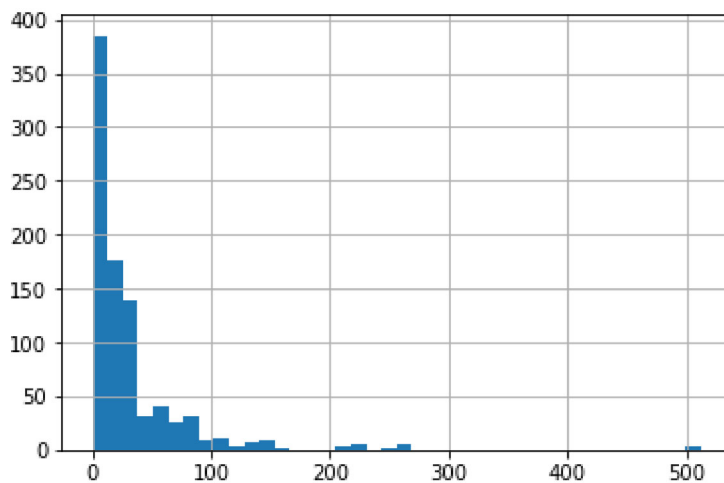
```
In [15]: # Age Histogram  
titanic['Age'].hist(bins = 40)
```

Out[15]: <AxesSubplot:>



```
In [16]: # Fare Histogram  
titanic['Fare'].hist(bins = 40)
```

Out[16]: <AxesSubplot:>



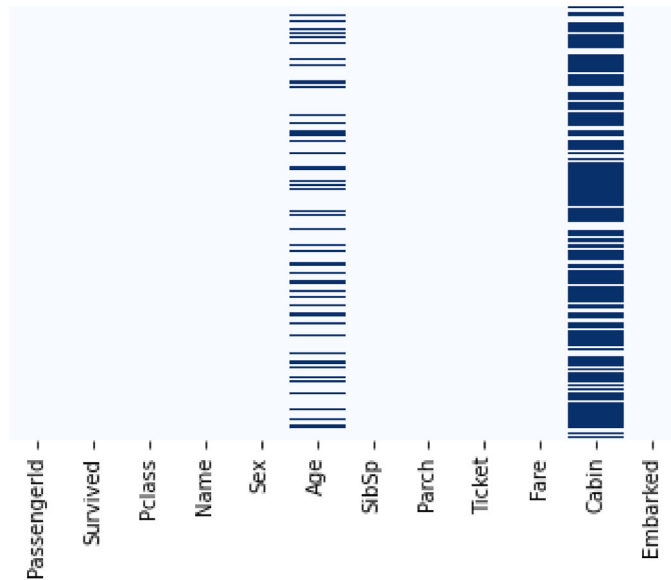
## Feature Engineering

### STEP #3: PREPARE THE DATA FOR TRAINING/ DATA CLEANING

#### Data Imputation

```
In [17]: # Let's explore which dataset is missing
sns.heatmap(titanic.isnull(), yticklabels = False, cbar = False, cmap="Blues")
```

Out[17]: <AxesSubplot:>



```
In [18]: # Let's drop the cabin coloumn and test with inplace = true
titanic.drop('Cabin',axis=1,inplace=True)
```

```
In [19]: #Let's drop some other columns/features as well
titanic.drop(['Name', 'Ticket', 'Embarked', 'PassengerId'],axis=1,inplace=True)
```

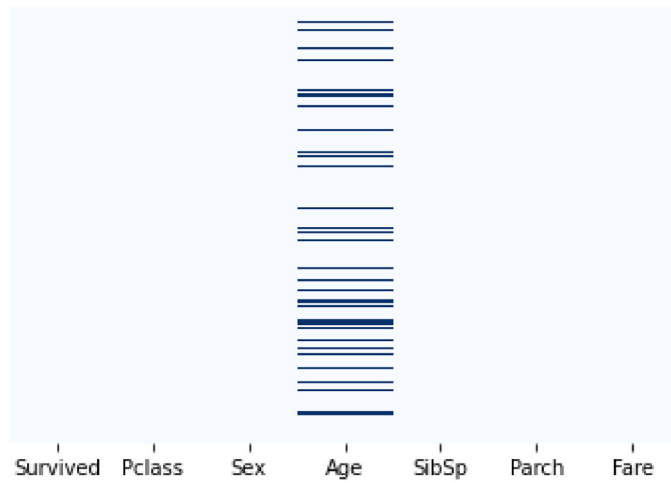
```
In [20]: titanic.head()
```

Out[20]:

	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

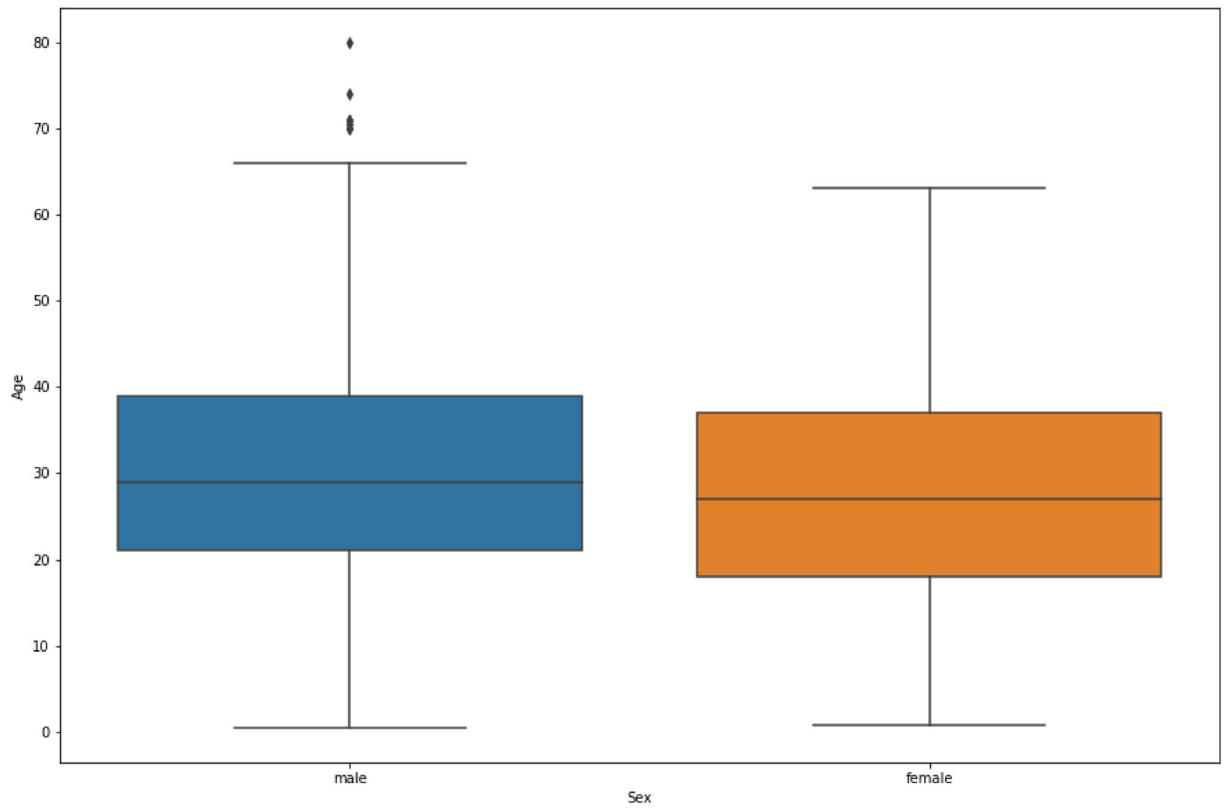
```
In [21]: # Let's view the data one more time!
sns.heatmap(titanic.isnull(), yticklabels = False, cbar = False, cmap="Blues")
# we cannot train a classifier with null values
# have to do Imputation
```

Out[21]: <AxesSubplot:>



```
In [22]: # Let's get the average age for male (~29) and female (~25)
plt.figure(figsize=(15, 10))
sns.boxplot(x='Sex', y='Age', data=titanic)
```

```
Out[22]: <AxesSubplot:xlabel='Sex', ylabel='Age'>
```



```
In [23]: # now we can fill the missing values or null values with the age mean/avg based on sex
def Fill_Age(data):
    #data index=0 is age
    #data index=1 is sex
    age = data[0]
    sex = data[1]

    if pd.isnull(age):
        if sex == 'male':
            return 29
        else:
            return 25
    else:
        return age
```

```
In [24]: titanic['Age'] = titanic[['Age', 'Sex']].apply(Fill_Age,axis=1)
```

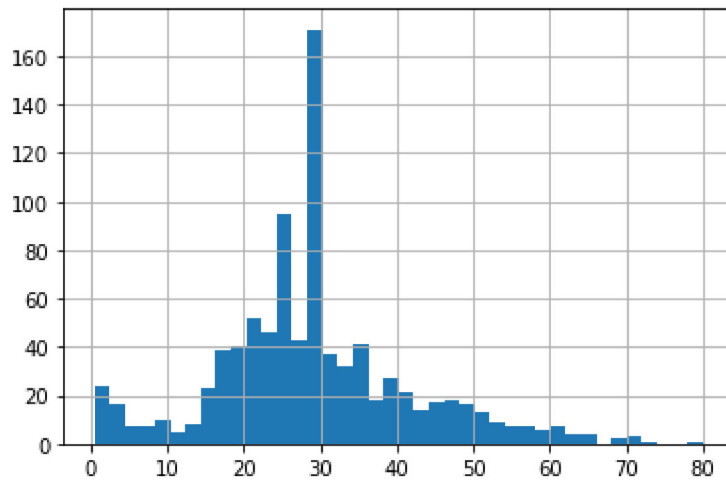
```
In [25]: # Let's view the data one more time!
sns.heatmap(titanic.isnull(), yticklabels = False, cbar = False, cmap="Blues")
```

Out[25]: <AxesSubplot:>



```
In [26]: # Age Histogram
titanic['Age'].hist(bins = 40)
```

Out[26]: <AxesSubplot:>



## Convert Categorical Columns into Dummy Variables

```
In [27]: titanic.head()
```

Out[27]:

	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



```
In [28]: pd.get_dummies(titanic['Sex'])  
# You just need one column only to represent male or female
```

Out[28]:

	female	male
0	0	1
1	1	0
2	1	0
3	1	0
4	0	1
...	...	...
886	0	1
887	1	0
888	1	0
889	0	1
890	0	1

891 rows × 2 columns

```
In [29]: male = pd.get_dummies(titanic['Sex'], drop_first = True)
```

```
In [30]: # Let's drop the sex column so we can replace it with the dummies  
titanic.drop(['Sex'], axis=1, inplace=True)
```

```
In [31]: titanic.head()
```

Out[31]:

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

```
In [32]: # Now Let's add the encoded column male again  
titanic = pd.concat([titanic, male], axis=1)
```

```
In [33]: titanic.head()
```

Out[33]:

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

## STEP#4: MODEL TRAINING

```
In [57]: #Let's drop the target coloumn before we do train test split  
X = titanic.drop('Survived',axis=1)  
y = titanic['Survived']
```

```
In [58]: from sklearn.model_selection import train_test_split
```

```
In [64]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s
```

```
In [65]: # Fitting Logistic Regression to the Training set  
from sklearn.linear_model import LogisticRegression  
classifier = LogisticRegression(random_state=0)  
classifier.fit(X_train, y_train)
```

Out[65]:

▼	LogisticRegression
	LogisticRegression(random_state=0)

## STEP#5: MODEL TESTING

```
In [66]: # the classifier now contains our trained model
```

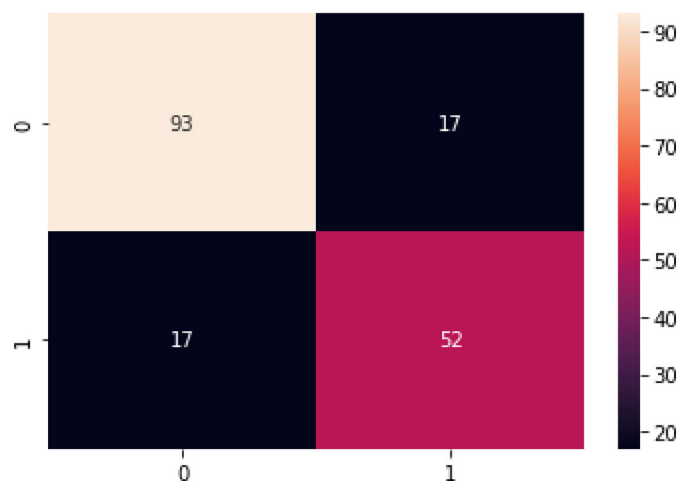
```
y_predict_test = classifier.predict(X_test)
y_predict_test
```

```
Out[66]: array([0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1,
                0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
                1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0,
                1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
                0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
                0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0,
                1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
                1, 0, 0], dtype=int64)
```

```
In [67]: from sklearn.metrics import classification_report, confusion_matrix
```

```
In [68]: cm = confusion_matrix(y_test, y_predict_test)
sns.heatmap(cm, annot=True, fmt="d")
```

```
Out[68]: <AxesSubplot:>
```



```
In [69]: print(classification_report(y_test, y_predict_test))
```

	precision	recall	f1-score	support
0	0.85	0.85	0.85	110
1	0.75	0.75	0.75	69
accuracy			0.81	179
macro avg	0.80	0.80	0.80	179
weighted avg	0.81	0.81	0.81	179