In [1]:

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

In [2]:

```
df = pd.read_csv("train.csv")
data = df.copy()
data.head()
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [6]:

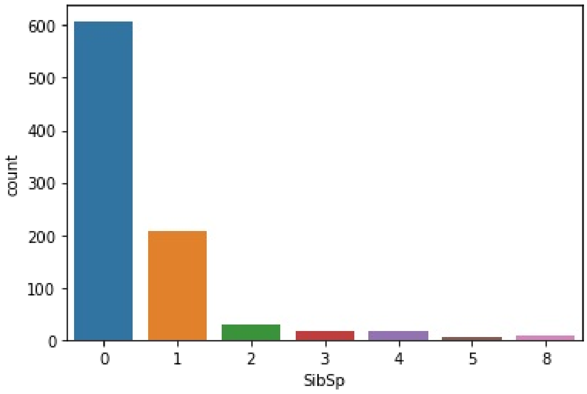
```
data.info()
```

```
<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
```

EDA

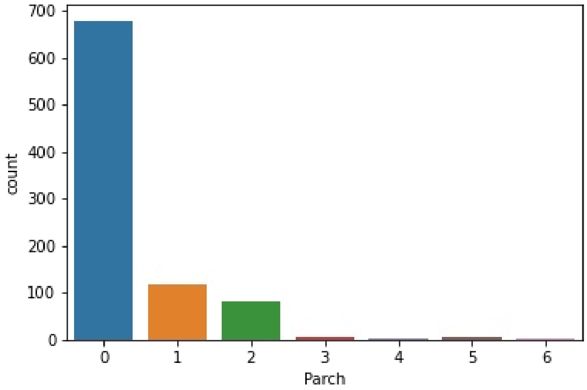
In [9]:

```
sns.countplot(x= "SibSp", data= data);
```



In [10]:

```
sns.countplot(x= "Parch", data= data);
```

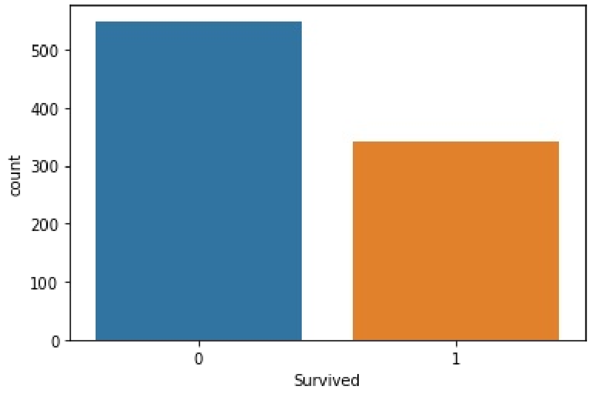


In [14]:

```
sns.countplot(data.Survived);
```

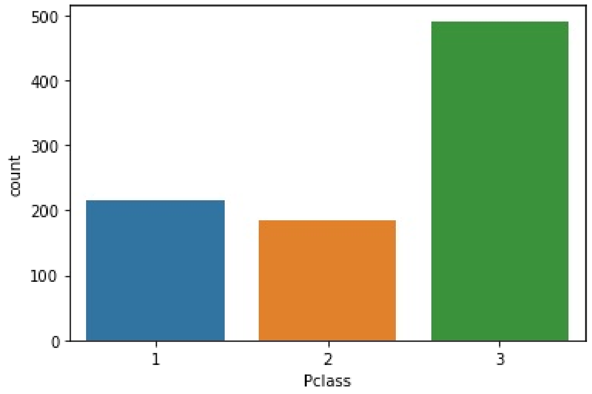
```
In [14]: sns.countplot(data.Survived);
```

```
C:\Users\Aboya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



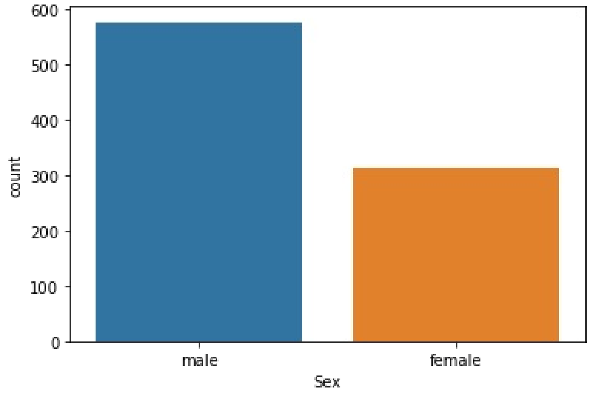
```
In [15]: sns.countplot(data.Pclass);
```

```
C:\Users\Aboya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



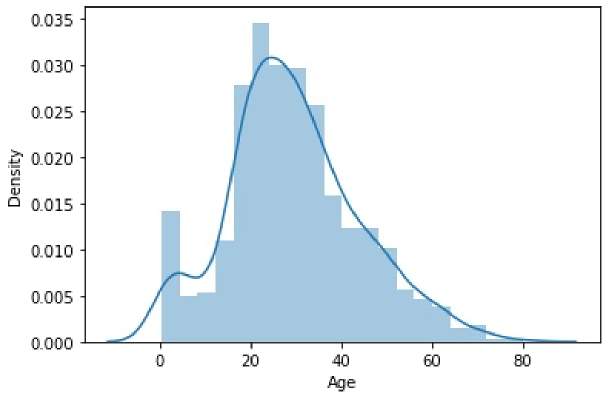
```
In [16]: sns.countplot(data.Sex);
```

```
C:\Users\Aboya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



```
In [17]: sns.distplot(data.Age);
```

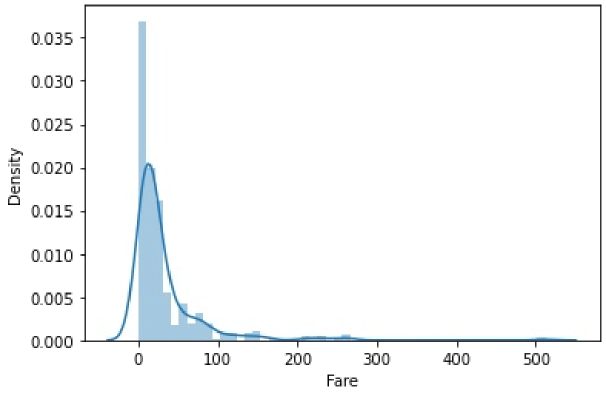
```
C:\Users\Aboya\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [18]:

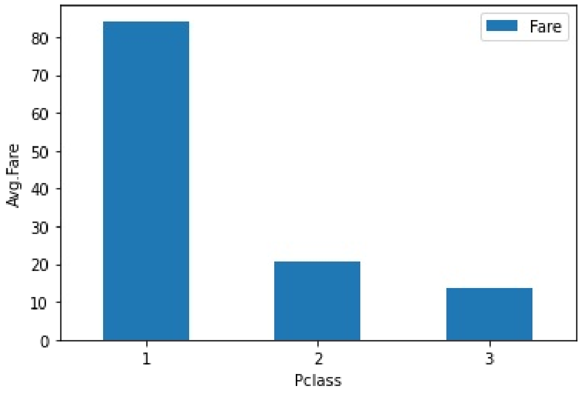
```
sns.distplot(data.Fare);
```

```
C:\Users\Aboya\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



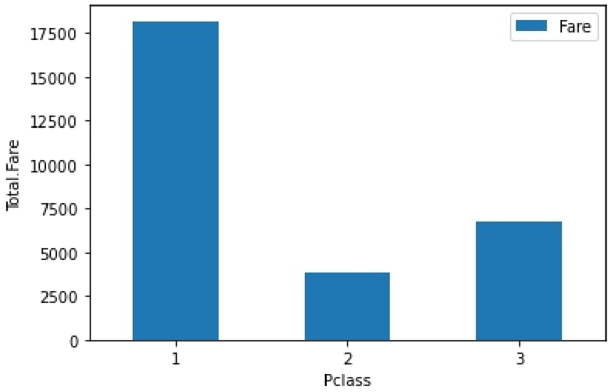
In [22]:

```
class_fare = data.pivot_table(index= "Pclass", values= "Fare")
class_fare.plot(kind= "bar")
plt.xlabel("Pclass")
plt.ylabel("Avg.Fare")
plt.xticks(rotation=0);
```



In [23]:

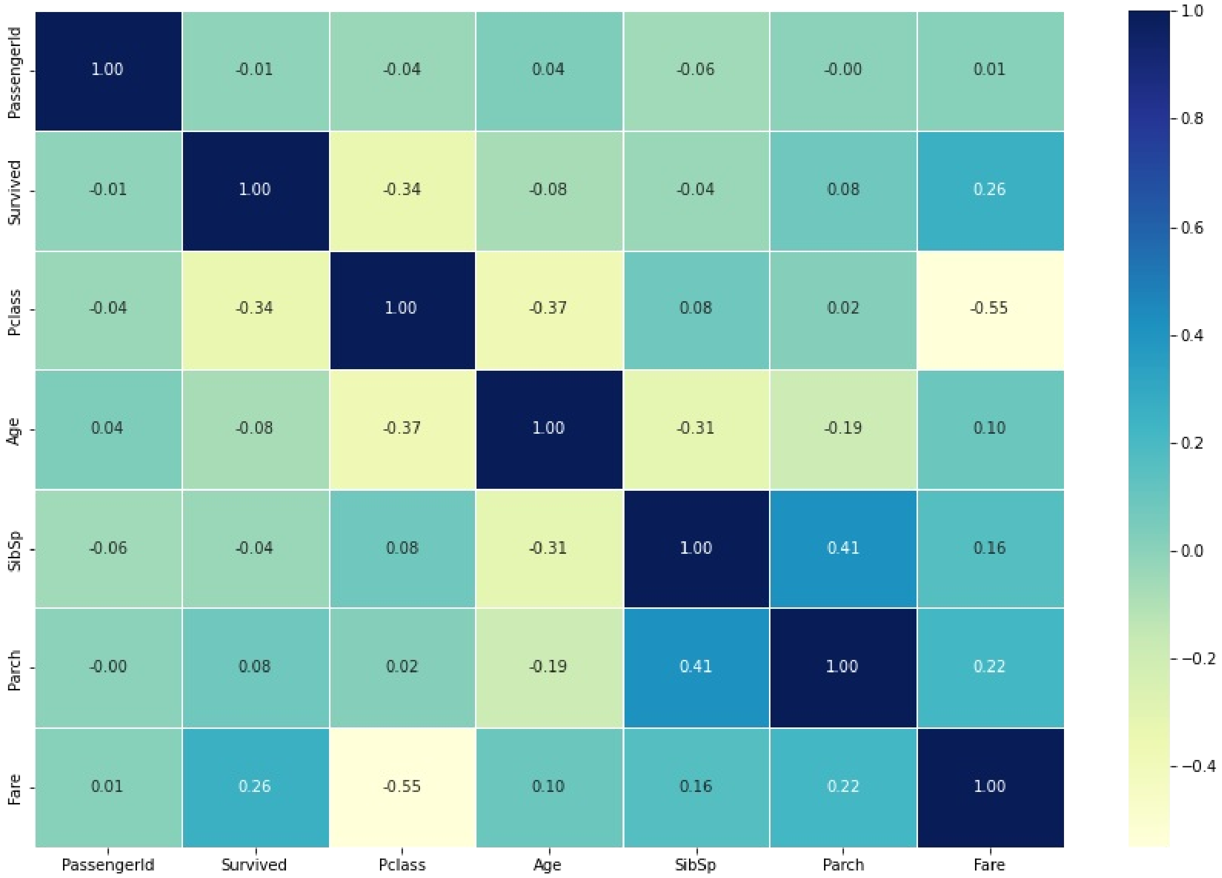
```
class_fare = data.pivot_table(index= "Pclass", values= "Fare", aggfunc= sum)
class_fare.plot(kind= "bar")
plt.xlabel("Pclass")
plt.ylabel("Total.Fare")
plt.xticks(rotation=0);
```



```
In [32]: from sklearn.preprocessing import MinMaxScaler
         minmax = MinMaxScaler()
         #minmax.fit(data.Fare)
```

```
In [33]: corr_mat = data.corr()

         ax,fig = plt.subplots(figsize=(15,10))
         ax = sns.heatmap(corr_mat,
                         annot=True,
                         linewidths=0.5,
                         fmt=".2f",
                         cmap="YlGnBu")
```



	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.00	-0.01	-0.04	0.04	-0.06	-0.00	0.01
Survived	-0.01	1.00	-0.34	-0.08	-0.04	0.08	0.26
Pclass	-0.04	-0.34	1.00	-0.37	0.08	0.02	-0.55
Age	0.04	-0.08	-0.37	1.00	-0.31	-0.19	0.10
SibSp	-0.06	-0.04	0.08	-0.31	1.00	0.41	0.16
Parch	-0.00	0.08	0.02	-0.19	0.41	1.00	0.22
Fare	0.01	0.26	-0.55	0.10	0.16	0.22	1.00

Data Processing

```
In [34]: data.isnull().sum()
```

```
Out[34]: 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 [36]: #Dealing with missing values
         data["Age"] = data.Age.fillna(data.Age.mean())
```

```
data["Embarked"] = data.Embarked.fillna(data.Embarked.mode())
```

In [43]:
```
# Drop unnecessary columns
#data = data.drop(columns=["Ticket","Name"], axis=1)
```

In [45]:
```
data.drop("Cabin", axis=1, inplace=True)
```

In [128…
```
# Split into training and test
X= data.drop(columns=["PassengerId", "Survived"], axis=1)
y= data["Survived"]
```

In [46]:
```
for label, content in data.items():
    if not pd.api.types.is_numeric_dtype(content):
        print(label)
```

```
Sex
Embarked
```

In [105…
```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
```

In [109…
```
#convert non numerical data to numerical
for label,content in data.items():
    if not pd.api.types.is_numeric_dtype(content):
        data[label] = pd.Categorical(content).codes+1
```

In [110…
```
data
```

Out[110]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	2	22.000000	1	0	7.2500	3
1	2	1	1	1	38.000000	1	0	71.2833	1
2	3	1	3	1	26.000000	0	0	7.9250	3
3	4	1	1	1	35.000000	1	0	53.1000	3
4	5	0	3	2	35.000000	0	0	8.0500	3
...	...	...	...	...	...	...	...	...	...
886	887	0	2	2	27.000000	0	0	13.0000	3
887	888	1	1	1	19.000000	0	0	30.0000	3
888	889	0	3	1	29.699118	1	2	23.4500	3
889	890	1	1	2	26.000000	0	0	30.0000	1
890	891	0	3	2	32.000000	0	0	7.7500	2

891 rows × 9 columns

In [127…
```
from sklearn. preprocessing import StandardScaler
models= StandardScaler()
models.fit_transform(data)
```

Out[127]:
```
array([[-1.73010796, -0.78927234,  0.82737724, ..., -0.47367361,
        -0.50244517,  0.58796609],
       [-1.72622007,  1.2669898 , -1.56610693, ..., -0.47367361,
         0.78684529, -1.91264387],
       [-1.72233219,  1.2669898 ,  0.82737724, ..., -0.47367361,
        -0.48885426,  0.58796609],
       ...,
       [ 1.72233219, -0.78927234,  0.82737724, ...,  2.00893337,
        -0.17626324,  0.58796609],
       [ 1.72622007,  1.2669898 , -1.56610693, ..., -0.47367361,
        -0.04438104, -1.91264387],
       [ 1.73010796, -0.78927234,  0.82737724, ..., -0.47367361,
        -0.49237783, -0.66233889]])
```

In [62]:
```
np.random.seed(42)
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import ExtraTreesClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
```

In [129…
```
from sklearn.model_selection import train_test_split

def Model_fit_score(model, X, y):
    x_train, x_test, y_train, y_test= train_test_split(X, y, test_size=0.2, random_state=42)
    model.fit(x_train, y_train)
    print("Accuracy:", model.score(x_test, y_test))
    score= cross_val_score(model, X, y, cv= 5)
    print("CV:", np.mean(score))
```

In [130…

```
model= RandomForestClassifier()
Model_fit_score(model, X, y)
```

```
Accuracy: 0.8100558659217877
CV: 0.8069926558282594
```

In [131…

```
model= KNeighborsClassifier()
Model_fit_score(model, X, y)
```

```
Accuracy: 0.7206703910614525
CV: 0.6958947963090829
```

In [132…

```
model= LogisticRegression()
Model_fit_score(model, X, y)
```

```
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:444: ConvergenceWarning: lbfgs fai
led to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:444: ConvergenceWarning: lbfgs fai
led to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Accuracy: 0.8100558659217877
CV: 0.786761659657272
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:444: ConvergenceWarning: lbfgs fai
led to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:444: ConvergenceWarning: lbfgs fai
led to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [133…

```
model= DecisionTreeClassifier()
Model_fit_score(model, X, y)
```

```
Accuracy: 0.7932960893854749
CV: 0.7699391124223214
```

In [134…

```
model= ExtraTreesClassifier()
Model_fit_score(model, X, y)
```

```
Accuracy: 0.8324022346368715
CV: 0.7991463184985248
```

In [135…

```
model= XGBClassifier()
Model_fit_score(model, X, y)
```

```
Accuracy: 0.8100558659217877
CV: 0.8126169104262131
```

In [136…

```
model= CatBoostClassifier(verbose=0)
Model_fit_score(model, X, y)
```

```
Accuracy: 0.8324022346368715
CV: 0.8215554579122466
```

In [137…

```
model= LGBMClassifier
Model_fit_score(model, X, y)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [137], in <cell line: 2>()
      1 model= LGBMClassifier
----> 2 Model_fit_score(model, X, y)

Input In [129], in Model_fit_score(model, X, y)
      3 def Model_fit_score(model, X, y):
      4     x_train, x_test, y_train, y_test= train_test_split(X, y, test_size=0.2, random_state=42)
----> 5     model.fit(x_train, y_train)
      6     print("Accuracy:", model.score(x_test, y_test))
      8     score= cross_val_score(model, X, y, cv= 5)

TypeError: fit() missing 1 required positional argument: 'y'
```

Improving The Model

In [139…

```
from sklearn.model_selection import GridSearchCV
```

In [142…

```
x_train, x_test, y_train, y_test= train_test_split(X, y, test_size=0.2, random_state=42)
```

In [145…

```
cat_grid = {'l2_leaf_reg': [1, 3, 5, 7, 9],
           'depth': [4,5,6,7,8,9, 10],
           'learning_rate' : [0.01,0.02,0.03,0.04],
            'iterations'    : [10, 20,30,40,50]}
```

In [146…

```
CBC= CatBoostClassifier()
Grid_CBC= GridSearchCV(estimator=CBC, param_grid = cat_grid, cv = 2, n_jobs=-1)

Grid_CBC.fit(x_train, y_train)
```

```
0:	learn: 0.6751097	total: 4.45ms	remaining: 218ms
1:	learn: 0.6582312	total: 11ms	remaining: 263ms
2:	learn: 0.6437254	total: 24.4ms	remaining: 383ms
3:	learn: 0.6307700	total: 27.1ms	remaining: 311ms
4:	learn: 0.6180453	total: 68.6ms	remaining: 617ms
5:	learn: 0.6067498	total: 72.1ms	remaining: 529ms
6:	learn: 0.5952668	total: 94ms	remaining: 577ms
7:	learn: 0.5845203	total: 98.1ms	remaining: 515ms
8:	learn: 0.5744704	total: 111ms	remaining: 504ms
9:	learn: 0.5652978	total: 122ms	remaining: 489ms
10:	learn: 0.5580407	total: 126ms	remaining: 446ms
11:	learn: 0.5488939	total: 138ms	remaining: 435ms
12:	learn: 0.5397212	total: 151ms	remaining: 429ms
13:	learn: 0.5320056	total: 206ms	remaining: 529ms
14:	learn: 0.5249222	total: 232ms	remaining: 542ms
15:	learn: 0.5181263	total: 283ms	remaining: 602ms
16:	learn: 0.5113397	total: 330ms	remaining: 640ms
17:	learn: 0.5048187	total: 372ms	remaining: 662ms
18:	learn: 0.5009652	total: 376ms	remaining: 613ms
19:	learn: 0.4954269	total: 429ms	remaining: 643ms
20:	learn: 0.4897464	total: 477ms	remaining: 659ms
21:	learn: 0.4846231	total: 500ms	remaining: 637ms
22:	learn: 0.4792835	total: 510ms	remaining: 598ms
23:	learn: 0.4742952	total: 554ms	remaining: 601ms
24:	learn: 0.4695238	total: 567ms	remaining: 567ms
25:	learn: 0.4648564	total: 589ms	remaining: 544ms
26:	learn: 0.4604544	total: 595ms	remaining: 507ms
27:	learn: 0.4557825	total: 651ms	remaining: 512ms
28:	learn: 0.4518822	total: 678ms	remaining: 491ms
29:	learn: 0.4481334	total: 705ms	remaining: 470ms
30:	learn: 0.4443472	total: 729ms	remaining: 447ms
31:	learn: 0.4410949	total: 771ms	remaining: 434ms
32:	learn: 0.4374166	total: 817ms	remaining: 421ms
33:	learn: 0.4342516	total: 881ms	remaining: 415ms
34:	learn: 0.4319608	total: 884ms	remaining: 379ms
35:	learn: 0.4290137	total: 932ms	remaining: 362ms
36:	learn: 0.4262768	total: 976ms	remaining: 343ms
37:	learn: 0.4233129	total: 1.02s	remaining: 322ms
38:	learn: 0.4208265	total: 1.06s	remaining: 300ms
39:	learn: 0.4183650	total: 1.11s	remaining: 279ms
40:	learn: 0.4164391	total: 1.12s	remaining: 246ms
41:	learn: 0.4141897	total: 1.13s	remaining: 216ms
42:	learn: 0.4118918	total: 1.18s	remaining: 193ms
43:	learn: 0.4100569	total: 1.19s	remaining: 162ms
44:	learn: 0.4079166	total: 1.2s	remaining: 134ms
45:	learn: 0.4070656	total: 1.21s	remaining: 105ms
46:	learn: 0.4044345	total: 1.25s	remaining: 79.6ms
47:	learn: 0.4021252	total: 1.29s	remaining: 53.7ms
48:	learn: 0.3998525	total: 1.34s	remaining: 27.3ms
49:	learn: 0.3979983	total: 1.37s	remaining: 0us
```

Out[146]:

```
▸          GridSearchCV
▸ estimator: CatBoostClassifier
      ▸ CatBoostClassifier
```

In [148…

```
Grid_CBC.best_params_
```

Out[148]:

```
{'depth': 10, 'iterations': 50, 'l2_leaf_reg': 1, 'learning_rate': 0.04}
```

In [151…

```
ideal_model= CatBoostClassifier(verbose=0, depth=9,
                               iterations=30,
                               learning_rate= 0.01)
Model_fit_score(ideal_model, X, y)

print(" Results from Grid Search " )
print("\n The best estimator across ALL searched params:\n",Grid_CBC.best_estimator_)
print("\n The best score across ALL searched params:\n",Grid_CBC.best_score_)
print("\n The best parameters across ALL searched params:\n",Grid_CBC.best_params_)
```

```
Accuracy: 0.7932960893854749
CV: 0.8058376749733224
 Results from Grid Search

 The best estimator across ALL searched params:
 <catboost.core.CatBoostClassifier object at 0x000002C9568B8F70>

 The best score across ALL searched params:
 0.8258426966292134

 The best parameters across ALL searched params:
 {'depth': 10, 'iterations': 50, 'l2_leaf_reg': 1, 'learning_rate': 0.04}
```

Complete Model Training With full Data

In [152…

```
df_test = pd.read_csv("Test.csv")
df_test.head()
```

Out[152]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

In [154…

```
df_test["Embarked"] = data.Embarked.fillna(data.Embarked.mode())
df_test["Age"] = data.Age.fillna(data.Age.mean())
```

In [155…

```
df_test = df_test.drop(columns=["Ticket","Name"], axis=1)
```

In [156…

```
for label, content in df_test.items():
    if not pd.api.types.is_numeric_dtype(content):
        print(label)
```

```
Sex
Cabin
```

In [157…

```
#convert non numerical data to numerical
for label,content in df_test.items():
    if not pd.api.types.is_numeric_dtype(content):
        df_test[label] = pd.Categorical(content).codes+1
```

In [159…

```
df_test.drop("PassengerId", axis=1, inplace= True)
```

In [161…

```
model= CatBoostClassifier(verbose=0)

model.fit(X, y)
```

Out[161]:

```
<catboost.core.CatBoostClassifier at 0x2c95636f040>
```

In [162…

```
df_test
```

Out[162]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	3	2	22.000000	0	0	7.8292	0	3
1	3	1	38.000000	1	0	7.0000	0	1
2	2	2	26.000000	0	0	9.6875	0	3
3	3	2	35.000000	0	0	8.6625	0	3
4	3	1	35.000000	1	1	12.2875	0	3
...	...	...	...	...	...	...	...	...
413	3	2	29.699118	0	0	8.0500	0	3
414	1	1	44.000000	0	0	108.9000	23	3
415	3	2	29.699118	0	0	7.2500	0	3
416	3	2	34.000000	0	0	8.0500	0	3
417	3	2	18.000000	1	1	22.3583	0	3

418 rows × 8 columns

In [163…

```
y_preds =  model.predict(df_test)
```

Test Submission

In [164…

```
final_sub =  pd.read_csv("gender_submission.csv")
```

In [166…

```
final_sub["Survived"] = y_preds
```

In [175…

```
final_sub.head(10)
```

Out[175]:

	PassengerId	Survived
0	892	0
1	893	0
2	894	0
3	895	0
4	896	1
5	897	0
6	898	0
7	899	1
8	900	1
9	901	0

In [170…

```
final_sub.to_csv("submission2.csv", index=False)
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js