Bharat Data Science Internship

Mayuri Arun Pathak, Data Science Intern

Task 2 : Titanic Classification

- Objective: Make a system which tells whether person will be save from sinking. What factors were most likely lead to success-socio-economic status, age,gender and more.

- dataset link: https://www.kaggle.com/c/titanic/data
- Dataset Introduction:
- This data set consists of information about the passengers of the RMS Titanic ship And also have info about is that particular passenger has survived that disaster or not. For every individual person, we have information about —
  - Passenger Id- Id number of passengers in the data set
  - Name- Name of the passenger
  - Survival- Person survived or not (0 for No & 1 for Yes)
  - P-class- With what class of ticket that passenger was traveling.
  - sex- Male Or Female
  - Age- Age of the person in Years
  - Sibsp- Number of siblings/spouses on the Titanic
  - parch- Number of parents/children on the Titanic
  - Ticket- Ticket number
  - Fare- Amount of money that person paid to travel
  - Cabin- Cabin Number of Passenger
  - Embarked- Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)

- Steps we follow:
  - 1> Data Collection & Processing
  - 2> Data Analysis
  - 3> Data Visualization
  - 4> Encoding the Categorical Columns
  - 5> Model Training
  - 6> Model Evaluation

- Importing Libraries:

In [53]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Supress warnings
import warnings
warnings.filterwarnings("ignore")
```

Data Collection & Processing

In [54]:
```
import os
os.chdir("H:\\Data Science\\Internship\\Bharat Data Science Internship")
```

In [55]:
```
titanic_data = pd.read_csv("titanic_data.csv")
```

- Printing first five rows

In [56]:
```
titanic_data.head()
```

Out[56]:

	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

- Observations:
- As we can see above the data set has 12 columns or features. we already discussed what these features are all about earlier.
- Here the Survived column is the target variable or class label. The target variable is the feature that needs to be predicted by our models.
- We have numerical, categorical Types of features

- number of rows and columns

In [57]:
```
titanic_data.shape
```

Out[57]:
```
(891, 12)
```

- getting some informations about the data

In [58]:
```
 titanic_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
```

In [59]:
```
titanic_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
```


- Observation:
- The shape of data is (891,12) means in our data set we have 891 rows and 12 columns. Each row has info about a passenger so totally we have data of 891 passengers.
- In the above output, column consists of the name of the column, Non-null Count means How many non-null values we have in that column, Dtype means What type of value that column consists of (int64 means int value, float64 means float value, object means string value)
- In the age column we can see, Out of 891 values, we have 714 non-null values. It implies that we have 177 Null values. (891–714 = 177)
- Same in the Cabin feature Out of 891 values we have only 204 non-null values. it implies that we have 687 Null values. But this is Huge. we have only 23% of values present in the data set and 77% of values are missing so we can drop this feature while making our model.
- Except for age and cabin, any features do not have any null values.

- check the number of missing values in each column

In [60]:
```
titanic_data.isnull().sum()
```

Out[60]:
```
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
```

- Handling the Missing values

- drop the "Cabin" column from the dataframe

In [61]:
```
titanic_data = titanic_data.drop(columns='Cabin', axis=1)
```

- replacing the missing values in "Age" column with mean value

In [62]:
```
titanic_data['Age'].fillna(titanic_data['Age'].mean(), inplace=True)
```

- finding the mode value of "Embarked" column

In [63]:
```
print(titanic_data['Embarked'].mode())
```

```
0    S
Name: Embarked, dtype: object
```

In [64]:
```
print(titanic_data['Embarked'].mode()[0])
```

```
S
```

- replacing the missing values in "Embarked" column with mode value

In [65]:
```
titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0], inplace=True)
```

- check the number of missing values in each column

In [66]:
```
titanic_data.isnull().sum()
```

Out[66]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

Data Analysis

- getting some statistical measures about the data

In [67]:
```
titanic_data.describe()
```

Out[67]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	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	13.002015	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	22.000000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

- Observation:
- In the Age feature we can see the count is 714 it means for 714 persons we have age value, else are missing. We have seen this above right.??. We have a Mean of 29.69, It means The mean age of all the passengers is 29.69 Years. Std (stander deviation) as 14.52 It means most of the people have age in the range (29.69-14.52) to (29.69+14.52) because in a Continuous Random Variable most of the values can be found in the range of (mean-std) to (mean+std).
- The min age is 0.42 in the above image so It means out of all passengers we have 0.4 years old as the minimum age of any passenger. And We have a 25% value(25th Percentile value) as 20.12 years It means 25% of passengers have Age less than 20.12 years. same we have 50%value (50th percentile value) as 28.00 It Means 50% of the passengers have age less than 28.00 years. same as 75% value (75th percentile value)
- we have as 38.00 It means 75% people have Age less than 38.00 years. Max value we can see as 80.00 years It Means out of all passengers the Highest aged person has age 80.00 Years

- finding the number of people survived and not survived

In [68]:
```
titanic_data['Survived'].value_counts()
```

Out[68]:
```
0    549
1    342
Name: Survived, dtype: int64
```

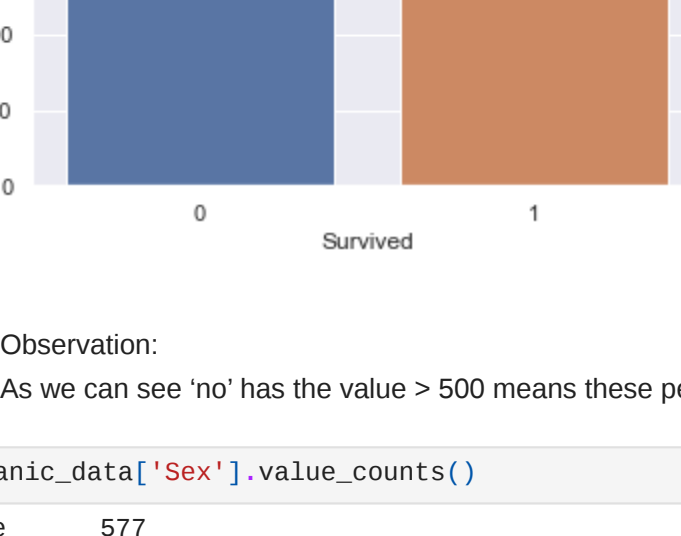
Data Visualization

In [69]:
```
sns.set()
```

- making a count plot for "Survived" column

In [70]:
```
sns.countplot('Survived', data=titanic_data)
```

Out[70]:
```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```



- Observation:
- As we can see 'no' has the value > 500 means these people can't survive in the disaster & approx 350 people survived. It is telling the same story as we have seen above.

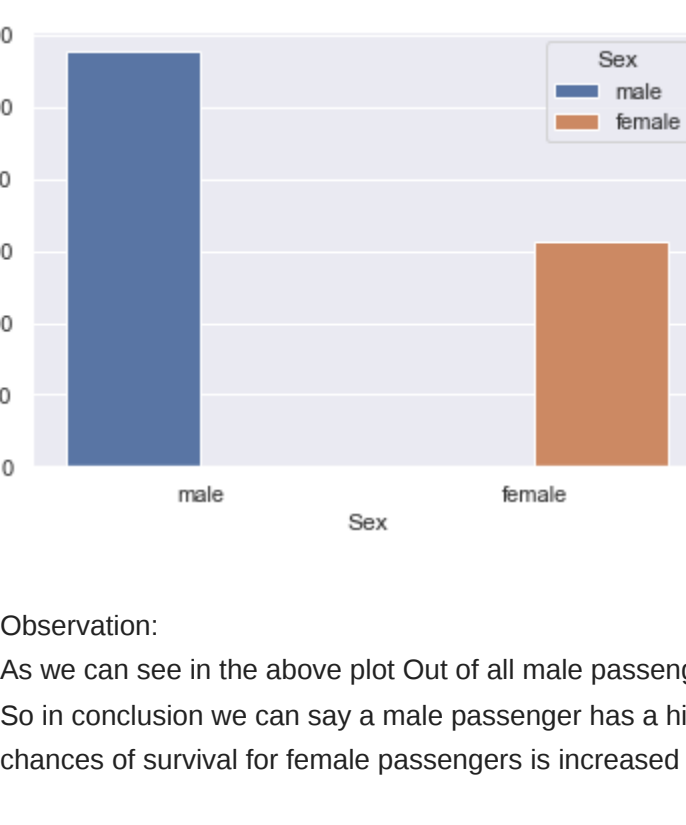
In [71]:
```
titanic_data['Sex'].value_counts()
```

Out[71]:
```
male      577
female    314
Name: Sex, dtype: int64
```

- making a count plot for "Sex" column

In [72]:
```
sns.countplot('Sex',hue = titanic_data["Sex"], data=titanic_data)
```

Out[72]:
```
<AxesSubplot:xlabel='Sex', ylabel='count'>
```

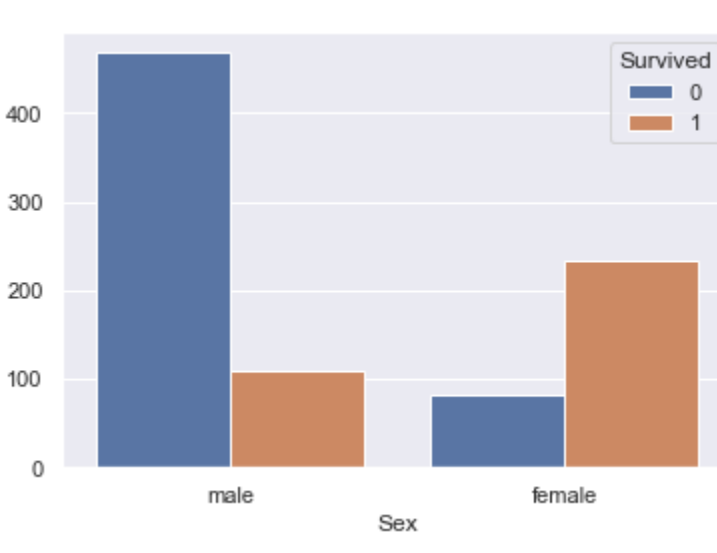


- Observation:
- As we can see in the above plot Out of all male passengers most of them died. And out of all Female passengers, most of them survived.
- So in conclusion we can say a male passenger has a higher chance to die and female passengers have a higher chance to survive. Or Because of the Ladies First policy, priorities were given to female passengers so the chances of survival for female passengers is increased than male passengers.

- number of survivors Gender wise

In [73]:
```
sns.countplot('Sex', hue='Survived', data=titanic_data)
```

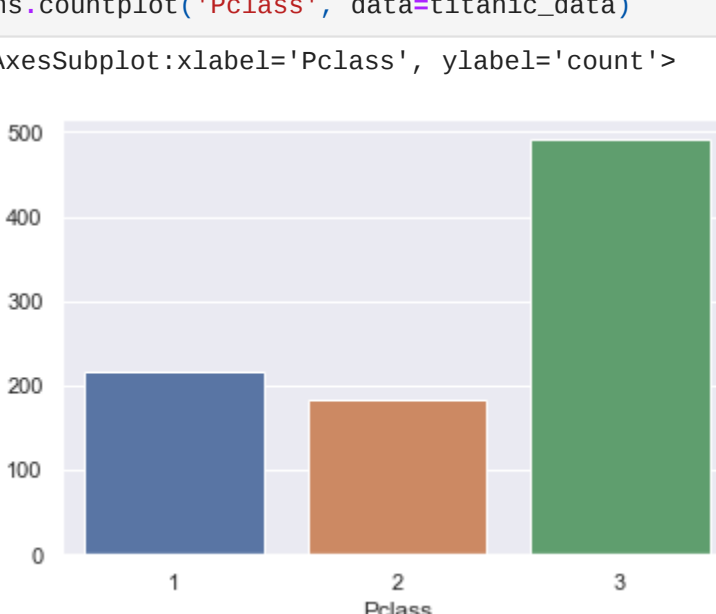
Out[73]:
```
<AxesSubplot:xlabel='Sex', ylabel='count'>
```



- making a count plot for "Pclass" column

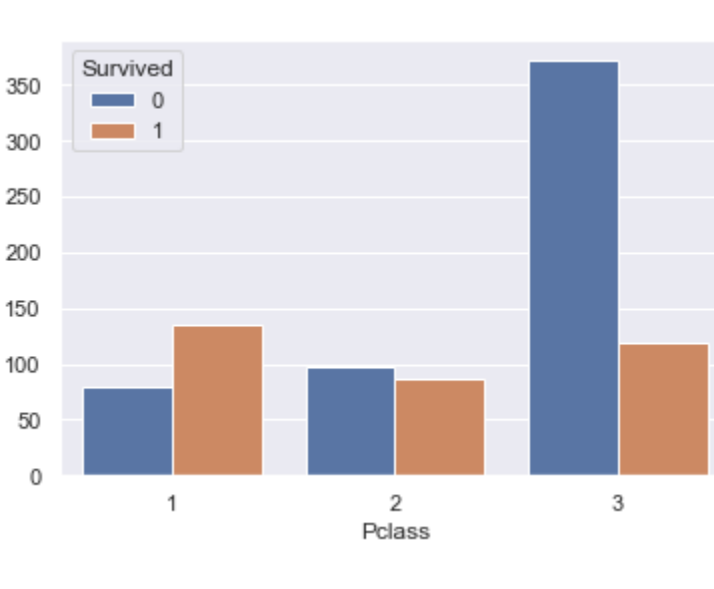
In [74]:
```
sns.countplot('Pclass', data=titanic_data)
```

Out[74]:
```
<AxesSubplot:xlabel='Pclass', ylabel='count'>
```



In [75]:
```
sns.countplot('Pclass', hue='Survived', data=titanic_data)
```

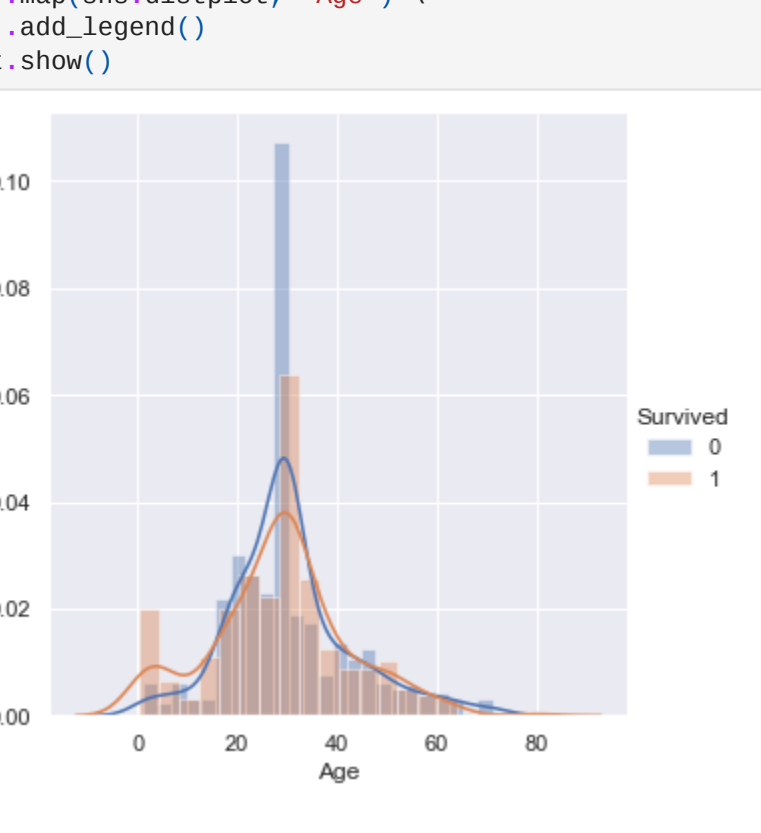
Out[75]:
```
<AxesSubplot:xlabel='Pclass', ylabel='count'>
```



- Observations:
- I have heard or read somewhere there is not any value in poor people's lives. The same concept is applying here. In the above output plot, we can see Persons who were traveling in 3rd class, most of them died or can not survived than other class passengers.
- Persons who were traveling in 2nd class, out of them almost equal number of people died and survived.
- Persons who were traveling in 1st class, out of a large number of people survived and also a fair number of persons died.
- So as the conclusion we can say that person who was traveling in lower class like 3rd has higher chances to die. & persons who were traveling in higher class like 1st has higher chance to survive.

- Age-Feature

In [76]:
```
#Filling null Values
data = titanic_data.fillna(28)
# plotting Pdfs
sns.FacetGrid(titanic_data, hue="Survived", size=5) \
   .map(sns.distplot, "Age") \
   .add_legend()
plt.show()
```

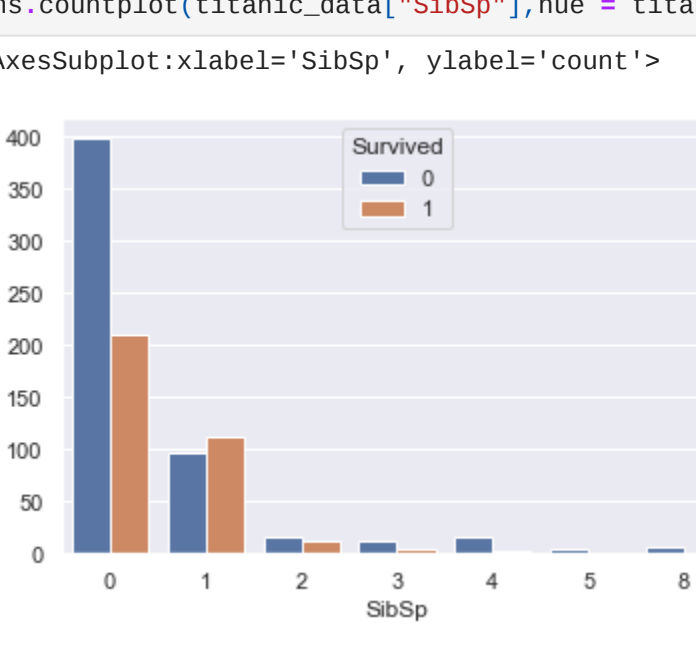


- Observations:
- Pdfs are almost overlapping on each other. so we can not conclude anything big clearly. but some of the conclusions are listed below.
- Passengers who have age <20 years so child's or teenagers, out of the numbers of passengers who survived is greater than numbers of people who died. This means while Rescuing priorities were given to children or teenagers.
- Persons who have age >20 years and <45 years. Out of the numbers of passengers who died is greater than numbers of passengers who survived.
- The rest of the pdfs are overlapping much so we will not struggle here much to get some info. But people with an age greater than 65 almost all of them died. it could be because of various reasons like people leave them on their luck or they died because of cold water of the Atlantic Ocean of that time.. etc...

- Sibsp Feature

In [77]:
```
sns.countplot(titanic_data["SibSp"],hue = titanic_data["Survived"],data = titanic_data)
```

Out[77]:
```
<AxesSubplot:xlabel='SibSp', ylabel='count'>
```

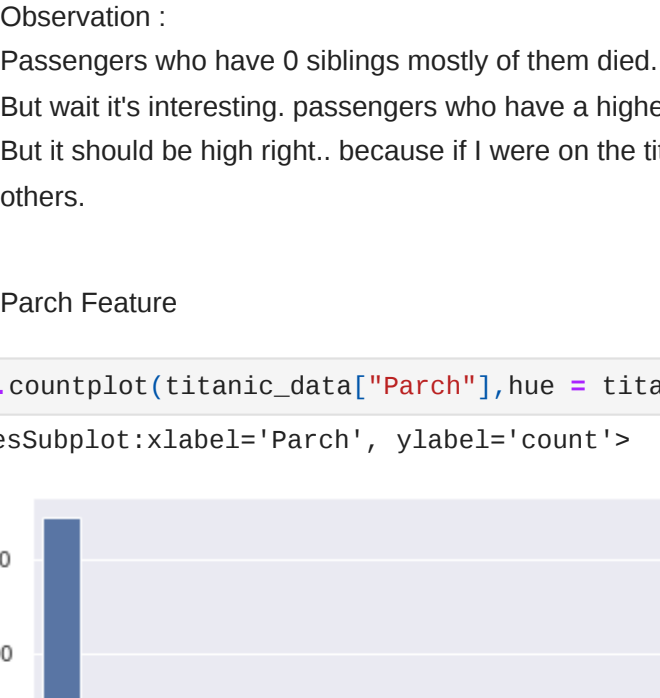


- Observation :
- Passengers who have 0 siblings mostly of them died. Passengers with 1 sibling have an equal chance to both die and survive.
- But wait it's interesting. passengers who have a higher number of siblings like 3, 4, 5, 8 have a very low chance to survive or almost 0% chance to survive.
- But it should be high right.. because if I were on the titanic and have 4, 5, or 8 siblings on titanic then the chances of my survival should be high right. But sadly in hard times, people think only for themselves not for others.

- Parch Feature

In [78]:
```
sns.countplot(titanic_data["Parch"],hue = titanic_data["Survived"],data = titanic_data)
```

Out[78]:
```
<AxesSubplot:xlabel='Parch', ylabel='count'>
```



- Observation:
- like sibs, almost the Same theory is applying here

- Bivariate Analysis
- Pair plots are the most widely used plots to perform visualization bi-variate analysis. But the problem with the pair plots is if we have n features in the dataset then it returns n*n plots. in the current data set, we have 12 features now so it will return 12*12= 144 plots which is again not easy to analyze. so we will take only a subset of features and analyze them.

In [79]:
```
sns.set_style("whitegrid");
sns.pairplot(titanic_data[["Survived","Pclass","Fare","Age"]], hue="Survived", size=3);
plt.show()
```

- Observations:
- From the above grid of plots, we can see that passengers who paid higher fares or traveling in the upper class have a higher chance to survive.
- Age is also giving some info like young persons have a higher chance to survive than old peoples.
- The picture is not very clear with pair plots so let's go towards uni-variate analysis to understand what exactly is happening.

Encoding the Categorical Columns

In [80]:
```
titanic_data['Sex'].value_counts()
```

Out[80]:
```
male      577
female    314
Name: Sex, dtype: int64
```

In [81]:
```
titanic_data['Embarked'].value_counts()
```

Out[81]:
```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

- converting categorical Columns

In [82]:
```
titanic_data.replace({'Sex':{'male':0,'female':1}, 'Embarked':{'S':0,'C':1,'Q':2}}, inplace=True)
```

In [83]:
```
titanic_data.head()
```

Out[83]:

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

- Separating features & Target

In [84]:
```
X = titanic_data.drop(columns = ['PassengerId','Name','Ticket','Survived'],axis=1)
Y = titanic_data['Survived']
```

In [85]:
```
print(X)
```

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

[891 rows x 7 columns]
```

In [86]:
```
print(Y)
```

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

- Splitting the data into training data & Test data

In [87]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_state=2)
```

In [88]:
```
print(X.shape, X_train.shape, X_test.shape)
```

```
(891, 7) (712, 7) (179, 7)
```

Model Training

Logistic Regression

In [89]:
```
model = LogisticRegression()
```

- training the Logistic Regression model with training data

In [90]:
```
model.fit(X_train, Y_train)
```

Out[90]:
```
LogisticRegression()
```

Model Evaluation

- Accuracy Score

- accuracy on training data

In [91]:
```
X_train_prediction = model.predict(X_train)
```

In [92]:
```
print(X_train_prediction)
```

```
[0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 1 1 0 0 1 0 1
 0 0 0 0 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 0 0 1 1 0 1 0 0 1
 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 1 1 1 0 1 0 0 0 0 0 1 0 0 0
 1 1 0 0 1 0 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 0 0 1 1 1 0 0 1 0 0
 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 0 1 1 1
 0 0 0 1 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 1 1 0 1 1 1 1 0 0 0 0 0 0 0
 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0
 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 0
 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 1 1
 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 0 1 1 1 0 0
 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 1 1 0 1 1 0 0 0
 0 1 0 1 0 0 1 1 0 0 0 0 1 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 1 0 0
 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 1 0 1 0 0 1 0 0 0 1 1 0 1 0
 0 0 0 0 1 0 0 1 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 1 1 0 1 0
 0 1 0 0 1 0 0 1 0 0 0 0 1 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0
 0 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0 0 1 1
 0 0 0 1 0 1 0 0 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0
 1 0 0 1 0 1 0 0 0 1 1 1 1 1 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 0 1 0 0 0 0 0 0
 0 0 0 1 1 0 0 1 0]
```

In [93]:
```
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
print('Accuracy score of training data : ', training_data_accuracy)
```

```
Accuracy score of training data :  0.8075842696629213
```

- accuracy on test data

In [94]:
```
X_test_prediction = model.predict(X_test)
```

In [95]:
```
print(X_test_prediction)
```

```
[0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1
 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 1 0
 1 0 0 0 1 0 1 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0
 0 0 0 1 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 1 0 1 0 0
 0 1 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 0 1 0 0 1 1 1 0 0 0 0 0]
```

In [96]:
```
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
print('Accuracy score of test data : ', test_data_accuracy)
```

```
Accuracy score of test data :  0.7821229050279329
```

Suggestations are always Welcome!

Thank You!