

Files

sample_data
train.csv

+ Code + Text

```
[1] 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
```

Data Collection & Processing

```
[2] # load the data from csv file to Pandas DataFrame
titanic_data = pd.read_csv('/content/train.csv')
```

```
[3] # printing the first 5 rows of the dataframe
titanic_data.head()
```

| | 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 | female | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | S |

Files

sample_data
train.csv

Disk 83.31 GB available

```
[4] titanic_data.shape

(891, 12)

[5] # getting some informations about the data
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

[6] # check the number of missing values in each column
titanic_data.isnull().sum()
```

Files

sample_data
train.csv

Disk 83.31 GB available

```
+ Code + Text
Handling the missing values

[7] # drop the "Cabin" column from the dataframe
titanic_data = titanic_data.drop(columns='Cabin', axis=1)

[8] # replacing the missing values in "Age" column with mean value
titanic_data['Age'].fillna(titanic_data['Age'].mean(), inplace=True)

[9] # finding the mode value of "Embarked" column
print(titanic_data['Embarked'].mode())

0    S
Name: Embarked, dtype: object

[10] print(titanic_data['Embarked'].mode()[0])

S

[11] # replacing the missing values in "Embarked" column with mode value
titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0], inplace=True)

▼ check the number of missing values in each column

titanic_data.isnull().sum()
```

Files

sample_data
train.csv

Disk 83.31 GB available

+ Code + Text

✓ 0s [12] # getting some statistical measures about the data
titanic_data.describe()

| | 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 |

✓ 0s [13] # finding the number of people survived and not survived
titanic_data['Survived'].value_counts()

```
0    549
1    342
Name: Survived, dtype: int64
```

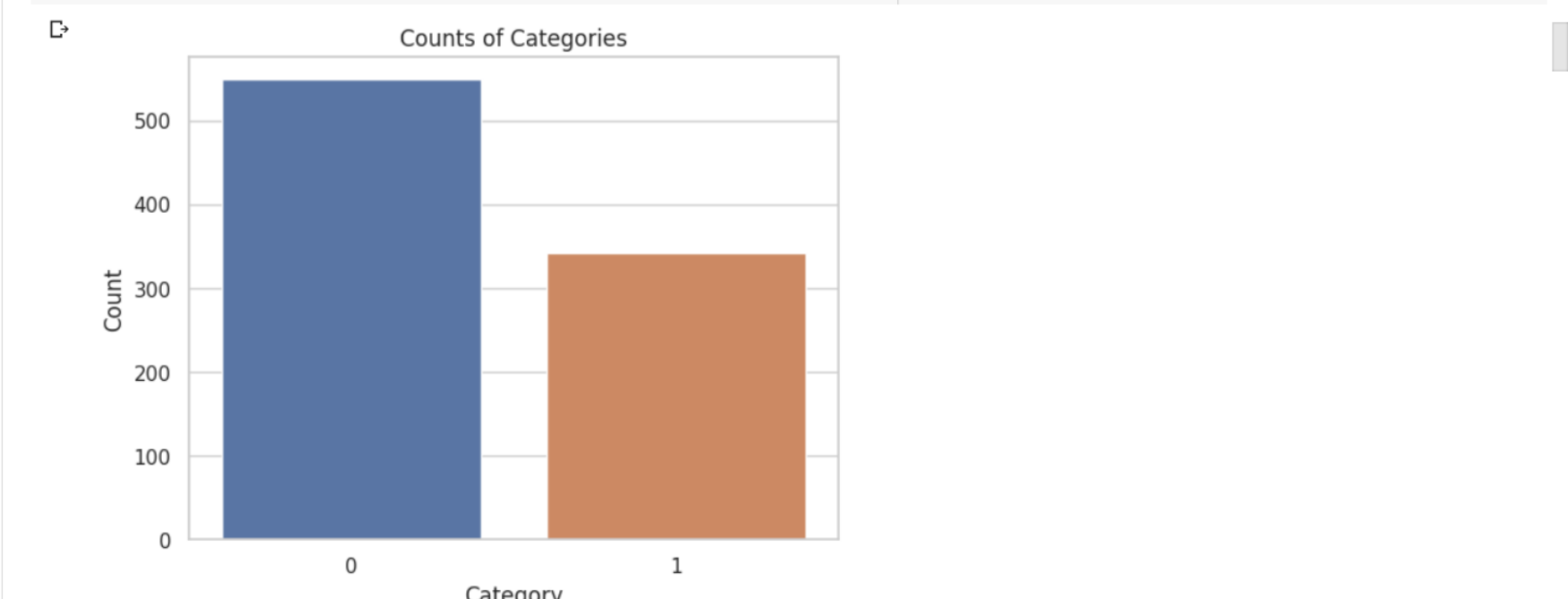
Data Visualization

Files

sample_data
train.csv

Disk 83.31 GB available

```
+ Code + Text  
0s # making a count plot for "Survived" column  
sns.barplot(x=titanic_data['Survived'].value_counts().index, y=titanic_data['Survived'].value_counts())  
plt.xlabel('Category')  
plt.ylabel('Count')  
plt.title('Counts of Categories')  
plt.show()
```



Files

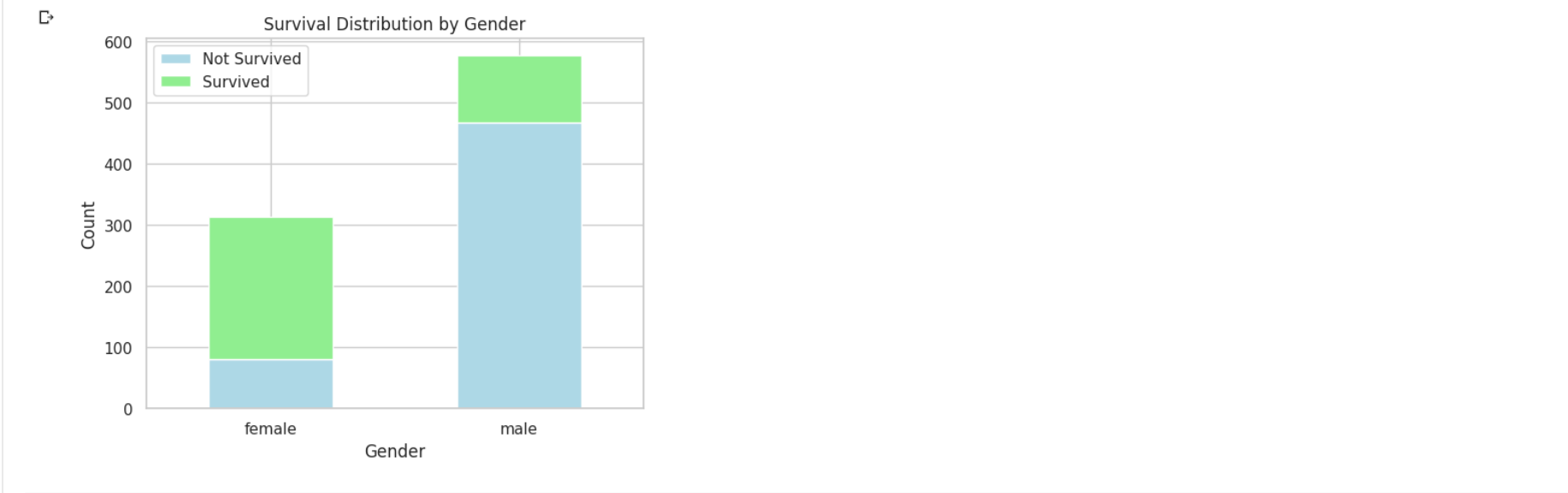
sample_data
train.csv

83.31 GB available

```
+ Code + Text
0s # Create a stacked bar plot
ax = grouped_data.plot(kind='bar', stacked=True, color=['lightblue', 'lightgreen'])

# Customize plot labels and legend
plt.title('Survival Distribution by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.legend(['Not Survived', 'Survived'])

plt.show()
```



```
0s [20] # making a count plot for "Pclass" column
sns.barplot(x=titanic_data['Pclass'].value_counts().index, y=titanic_data['Pclass'].value_counts())
plt.xlabel('Category')
plt.ylabel('Count')
```

Files

sample_data

train.csv

83.31 GB available

+ Code + Text

```
# making a count plot for "Pclass" column
sns.barplot(x=titanic_data['Pclass'].value_counts().index, y=titanic_data['Pclass'].value_counts())
plt.xlabel('Category')
plt.ylabel('Count')
plt.title('Counts of Categories')
plt.show()
```

Counts of Categories

| Category | Count |
|----------|-------|
| 1 | 220 |
| 2 | 180 |
| 3 | 490 |

```
[21] import pandas as pd
ct = pd.crosstab(titanic_data['Pclass'], titanic_data['Survived'])

# Create a stacked bar plot using Seaborn
```

0s completed at 9:48 AM

Files

sample_data
train.csv

83.31 GB available

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

```
titanic_data.head()
```

| | 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

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

```
[27] 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 |

Files

..
sample_data
train.csv

Disk 83.31 GB available

```
+ Code + Text
[28] 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

[29] X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_state=2)

[30] print(X.shape, X_train.shape, X_test.shape)
(891, 7) (712, 7) (179, 7)

Model Training

Logistic Regression

[31] model = LogisticRegression()

[32] # training the Logistic Regression model with training data
model.fit(X_train, Y_train)
```

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/ logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):

✓ 0s completed at 9:48 AM

Files

..
sample_data
train.csv

Disk 83.31 GB available

```
+ Code + Text
0s 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 0 0 1 1 0 0 1 0 1 0 1 0 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 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 1 0 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 0 0 1 1 1 0 0 1 0 0 1 0 0
 0 0 0 0 1 0 1 0 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 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 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 0 1 1 0 0
 1 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 0 1 0 0 1 0 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 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 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 0 1 0 1 0 0 0 0 1 1 0 0 0 1 1
 0 0 0 1 0 1 0 0 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0
 1 0 0 1 0 1 0 0 0 0 1 1 1 1 1 0 0 1 1 0 1 1 1 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0
 0 0 0 1 1 0 0 1 0]
```

```
[35] 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
```

```
[36] # accuracy on test data
      X_test_prediction = model.predict(X_test)
```

```
[37] print(X_test_prediction)

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

Files

sample_data
train.csv

Disk 83.31 GB available

```
+ Code + Text
1 0 0 1 0 1 0 0 0 1 1 1 1 0 0 1 1 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0
0 0 0 1 1 0 0 1 0]

[35] 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

[36] # accuracy on test data
X_test_prediction = model.predict(X_test)

[37] 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 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 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]

[38] 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

[38]
```