MyEduSolve

Kampus Merdeka INDONESIA JAYA

Using Python in:

Step by Step Data Cleansing and Exploratory Data Analysis of Titanic DataSet

Comment Team: Deni Riswana, Muhammad Fadhil Yusuf, Ahmad Fauzan Azhim, Andi Muhammad Yusuf, Shinta Nuriyah Arief, and Tiara Angela Hanami

Github Page: https://github.com/deniriswana/Data-Cleansing-and-Exploratory-Data-Analysis-of-Titanic-DataSet

Outline

The Summary Of Material

- Use Case Summary
    - Objective Statements
    - Challenges
    - Methodology
    - Expected Outcome
- Data Understanding
- Data Preparation
- Data Cleansing
- Exploratory Data Analysis
- Data Visualization

Use Case Summary

The sinking of the Titanic is one of the most infamous shipwrecks in history. In this project the function of Data Cleansing and Exploratory Data Analysis (EDA) is used to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.

Objective Statement

- Get the insight about how many people survived and died
- Knowing the age range and gender of the passengers
- Identify patterns by visualizing data in graphs
- Discover errors, outliers, and missing values in the data

Challenges

- Large size of data, can not maintain by excel spreadsheet
- The data have a lot missing values

Methodology

- Descriptive analysis
- Graph/chart analysis

Expected Outcome

- Get the insight about how many people survived and died
- Knowing the age range and gender of the passengers

DataSet Understanding

The first step is to understand the dataset we are going to work with. In this Titanic DataSet there where a total of 12 columns.

Download the Titanic DataSet: https://bit.ly/TitanicCsvDataSet

PassengerID	The ID of passengers
survived	Survival (0 = No; 1 = Yes)
pclass	Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
name	Name
sex	Sex
age	Age
sibsp	Number of Siblings/Spouses Aboard
parch	Number of Parents/Children Aboard
ticket	Ticket Number
fare	Passenger Fare
cabin	Cabin
embarked	Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

Data Preparation

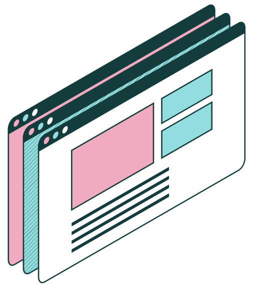
Data Preparation is the stage where we prepare tools and materials before carrying out data cleansing and exploratory data analysis.

Code Used

Python Vers. 3.10.2

Packages

Pandas, Numpy, Matplotlib, Seaborn



IDE for Python

Google Colab

Data Cleansing

Data Cleansing

Data cleaning and preparation is an integral part of data science. Oftentimes, raw data comes in a form that isn't ready for analysis or modeling due to structural characteristics or even the quality of the data.

- Importing Libraries

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

- Input Titanic Dataset

```
from google.colab import files
files.upload()
```

```
df = pd.read_csv("Titanic.csv")
```

- Show first 5 rows of as preview of Titanic Dataset

```
df.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 (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

```
df.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
```

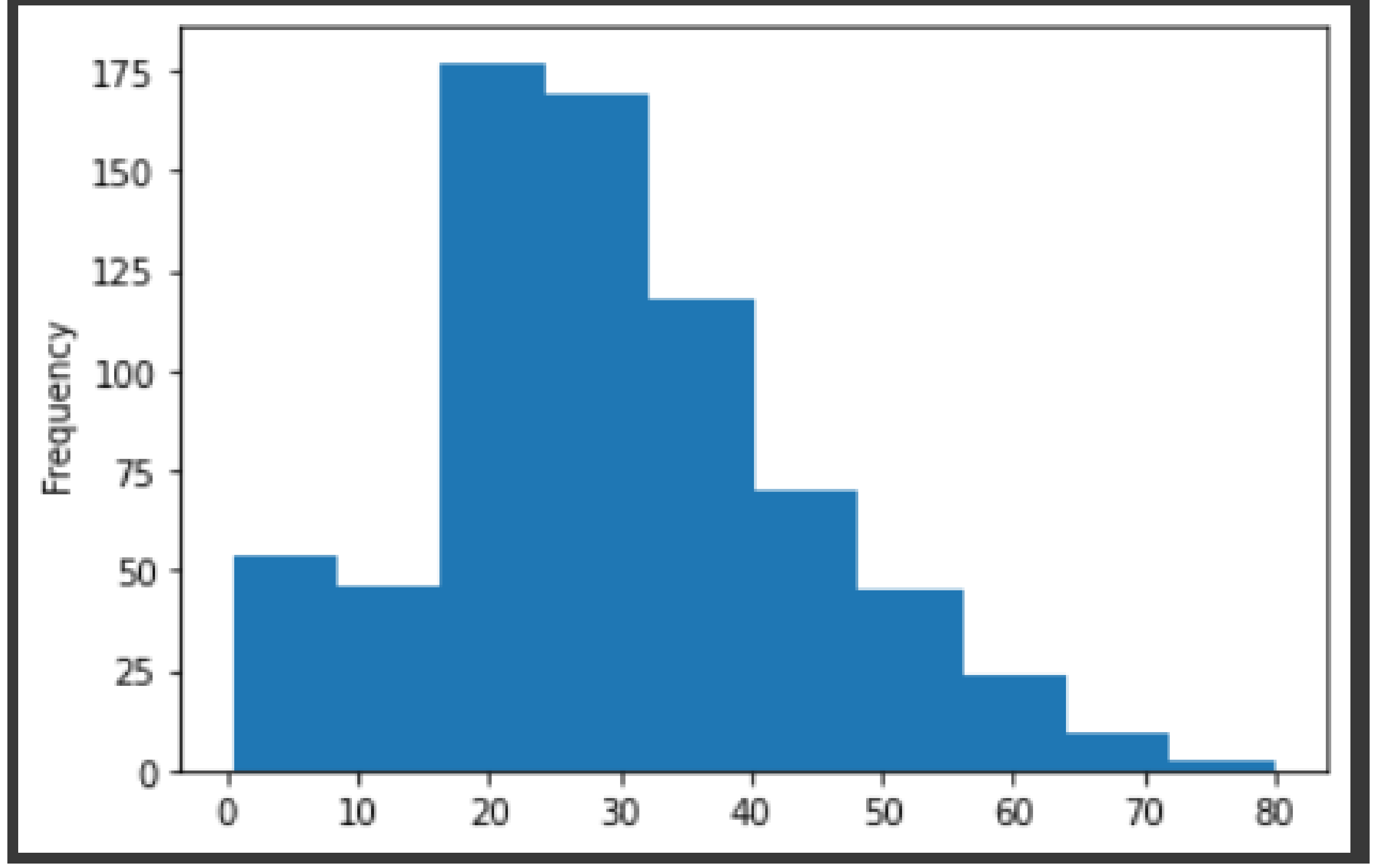
We see that, at the top of the displayed output of the info method, we have RangeIndex: 891 entries. If we look at the Non-null Count column in our displayed output, we see that the columns with 891 non-null values have zero missing, while columns with less than 891 non-null values have some missing. For example, the Age column has 714 non-null values, which means that it contains 177 missing values.

Handling Missing Value

- **Age Column**

To see the distribution of data from the age column, a histogram is used.

```
df.Age.plot(kind='hist');
```



Because column Age has a skewness distribution then we will imputation on column Age using median.

```
val = df.Age.median()
df['Age'] = df.Age.fillna(val)
```

Show the dataset info to see if the Age column has been imputed, it turns out that the Age column has now changed in number

```
df.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          891 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
```

Cabin Column

The total number of data entries is 891, while the Cabin column is 204. it means that there is null data in the Cabin column show proportion of data column Cabin

```
[ ]  df.Cabin.value_counts()

B96 B98        4
G6             4
C23 C25 C27    4
C22 C26        3
F33            3
              ..
E34            1
C7             1
C54            1
E36            1
C148           1
Name: Cabin, Length: 147, dtype: int64
```

It can be seen that the value column of Cabi has too many unique data, right? and also the Cabin info column is not very informative to find out Survived data then we will delete the Cabin column

```
[ ]  df.drop('Cabin', axis=1, inplace = True)
```

display dataset info to see if the Cabin column has been deleted or not

```
[ ]  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 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          891 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  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
```

Embarked Column

the total number of data entry is 891 while the embarked column has 889
data it means that there is null data in the embarked column, we check where the null data is

```
[ ]  df['Embarked'].value_counts()

S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

```
[ ]  df.Embarked[df.Embarked.isnull()]

61     NaN
829    NaN
Name: Embarked, dtype: object
```

show proportion of data column Embarked it turns out that
the Embarked column data is in the form of categorical data

```
[ ]  df.Embarked.value_counts()

     S    644
     C    168
     Q     77
     Name: Embarked, dtype: int64
```

when we are going to imputation on the Embarked column then we check the data type of the Embarked column first data column Embarked in the form of categoric data then the imputation uses its mode from the Embarked column proportion, S is the data that appears most often, then S is the mode

```
[ ]  val = df.Embarked.mode().values[0]
     df['Embarked'] = df.Embarked.fillna(val)
```

After the imputation, it can be seen that the proportion has changed

```
[ ]  df.Embarked.value_counts()

     S    646
     C    168
     Q     77
     Name: Embarked, dtype: int64
```

Column SibSp and Parch

After we do imputation Data now we would like to Manipulated it

The manipulation here is not to change the data value therapy to make this data easier to read by the machine . Column SibSp(sibling Spouse) means a column that states the number of siblings or the number of partners brought by the Passenger column Parch (Parent Child) means a column that states the number of parents or the number of children brought by the Passanger. We will create a new column that displays whether he is alone or with his family

```
[ ]  df['Alone']=df['SibSp']+df['Parch']
```

```
[ ]  df['Alone'][df['Alone']>0]='With Family'
     df['Alone'][df['Alone']==0]='Without Family'
```

Now display the latest data view

```
[ ] df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Alone
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S	With Family
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C	With Family
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S	Without Family
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S	With Family
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S	Without Family

After we manipulated the data we can see in the column Alone, the passanger bring their child or sibling categorize as a passanger with family and the passanger that not come with their child or sibling categorize as a passanger without family

The Relationship Between Column Sex and Column Survived

Now we want to see the relationship between column sex and column survived to see the proportion of Survived Sex Column

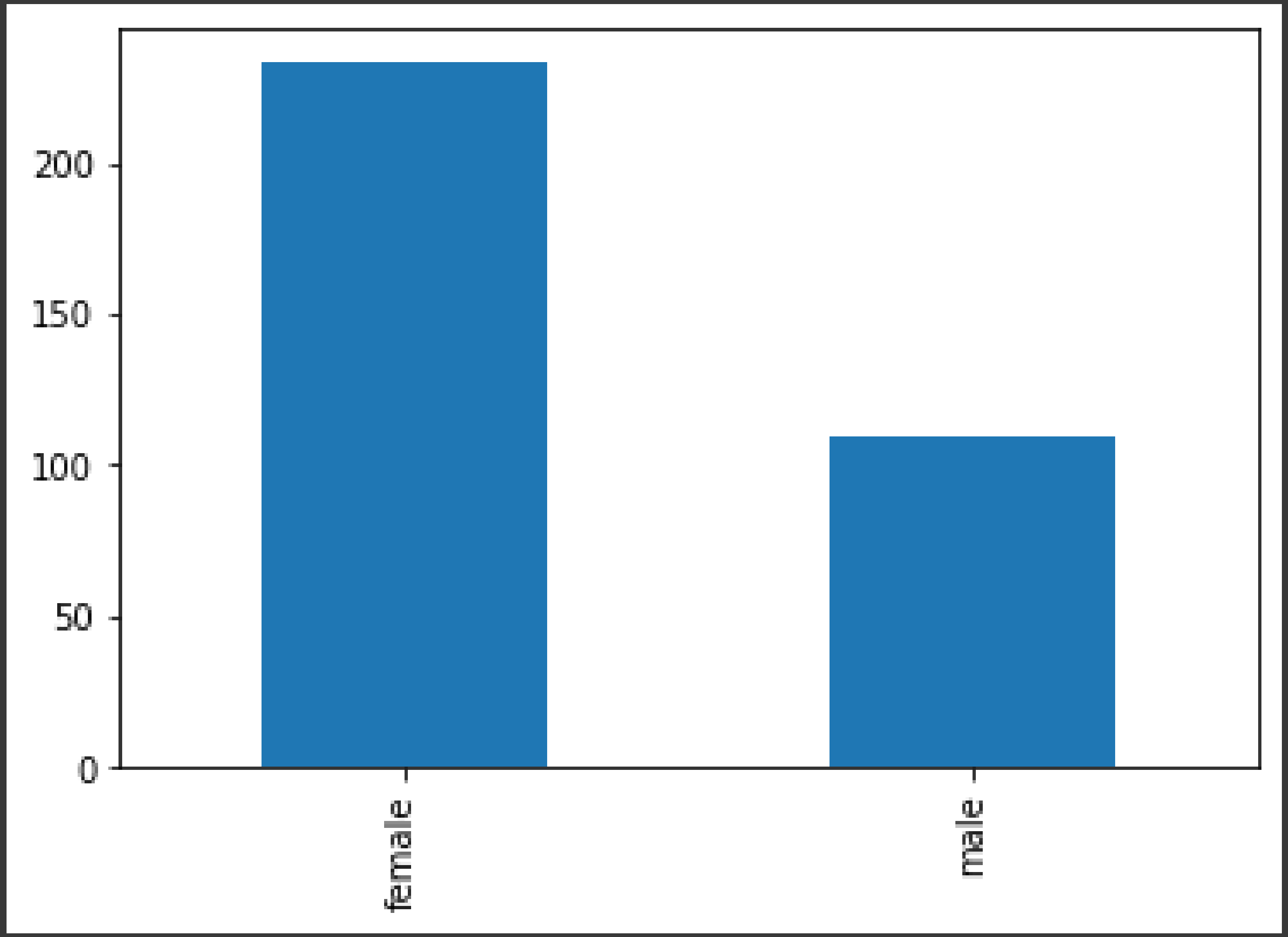
Following Code:

```
[ ]  df.Sex[df['Survived']==1].value_counts()

     female    233
     male      109
     Name: Sex, dtype: int64
```

Lastly visualize the data to see the comparison of proportion that survived passanger

```
[ ]  df.Sex[df['Survived']==1].value_counts().plot(kind='bar');
```



Exploratory Data Analysis

Exploratory Data Analysis

import pandas package

```
import pandas as pd
```

Load the data set

```
from google.colab import files
files.upload()
```

```
Choose Files  No file chosen        Upload widget is only available when the cell has been executed ir
Saving Titanic.csv to Titanic.csv
{'Titanic.csv': b'PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabi
```

Pandas default index starts from zero while the dataset index of the passangerId column starts from one then we will use the dataset index from the passangerId column

```
[ ] #Pandas default index starts from zero
    #while the dataset index of the passangerId column starts from one
    #then we will use the dataset index from the passangerId column

    df = pd.read_csv('Titanic.csv', index_col=0)
```

```
[ ] #Show the loaded dataset
    df.head()
```

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

- check data condition
- Int 64 index displays the total number of data entered
- non null count is the number of data entered that are not null
- O type is the data type of the column

```
[ ] #check data condition
    #Int 64 index displays the total number of data entered
    #non null count is the number of data entered that are not null
    #O type is the data type of the column

    df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Name      891 non-null    object
 3   Sex       891 non-null    object
 4   Age       714 non-null    float64
 5   SibSp     891 non-null    int64
 6   Parch     891 non-null    int64
 7   Ticket    891 non-null    object
 8   Fare      891 non-null    float64
 9   Cabin     204 non-null    object
 10  Embarked  889 non-null    object
dtypes: float64(2), int64(4), object(5
memory usage: 83.5+ KB
```

Display Nan number of dataset

```
[ ] #Display Nan number of dataset

    df.isnull().sum()
```

```
Survived      0
Pclass        0
Name          0
Sex           0
Age         177
SibSp         0
Parch         0
Ticket        0
Fare          0
Cabin       687
Embarked      2
dtype: int64
```

Display calculations from column datasets of type integer or float

```
[ ] #Display calculations from column datasets of type integer or float

    df.describe()
```

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

- Show any unique values in that column
- For example from column sex

```
[ ] #Show any unique values in that column
    #For example from column sex

    df.Pclass.unique()

    array([3, 1, 2])
```

- Show the number of unique values in the column

```
[ ] #Show the number of unique values in the column

    df.Pclass.nunique()

    3
```

- Show the number of rows and the number of columns of the data set

```
[ ] #Show the number of rows and the number of columns of the data set
    df.shape
```

```
(891, 11)
```

- duplicate data results

```
[ ] df[df.duplicated()]
```

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
PassengerId											

- display a data table from a duplicate

```
[ ] df.drop_duplicates()
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 11 columns

Embarked Column

- **Displays the missing value in the embarked column**

The total number of data entries is 891, while the Embarked column is 889. It means there is 2 null data in the Embarked column. This show that Passengerid 62 and 830 has missing value on embarked column.

```
PassengerId
62      NaN
830     NaN
Name: Embarked, dtype: object
```

This code shows which passengers id has Null value on embarked column

```
df[df.Embarked.isnull()]
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
62	1	1	Icard, Miss. Amelie	female	38.0	0	0	113572	80.0	B28	NaN
830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62.0	0	0	113572	80.0	B28	NaN

- ## Show and impute the proportions of the Embarked column

It turns out that the data from the Embarked column is in the form of categorical data. If we are going to imput the Embarked column, Then we check the data type of the Embarked column first

```
df.Embarked.value_counts()
```

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

Data column Embarked in the form of categoric data, the imputation uses the mode. From the Eembarked column proportion, S is the data that appears most often, then S is the mode

```
val = df.Embarked.mode().values[0]
df['Embarked'] = df. Embarked .fillna(val)
```

After imputation completed, it can be seen that the proportion has been changed

```
df.Embarked.value_counts()
```

```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

- **Convert embarked column from an object data type to numeric type**

Embarked column is still an object data type, to facilitate the analysis process, it must be converted from an object type to a numeric type

```
df.Embarked = df.Embarked.map ({'S':0,'C':1, 'Q':2})
```

Then show dataset info to see if the Embarked column has changed its data type. It turns out that the Embarked column has now changed to numeric

```
df.info()
```

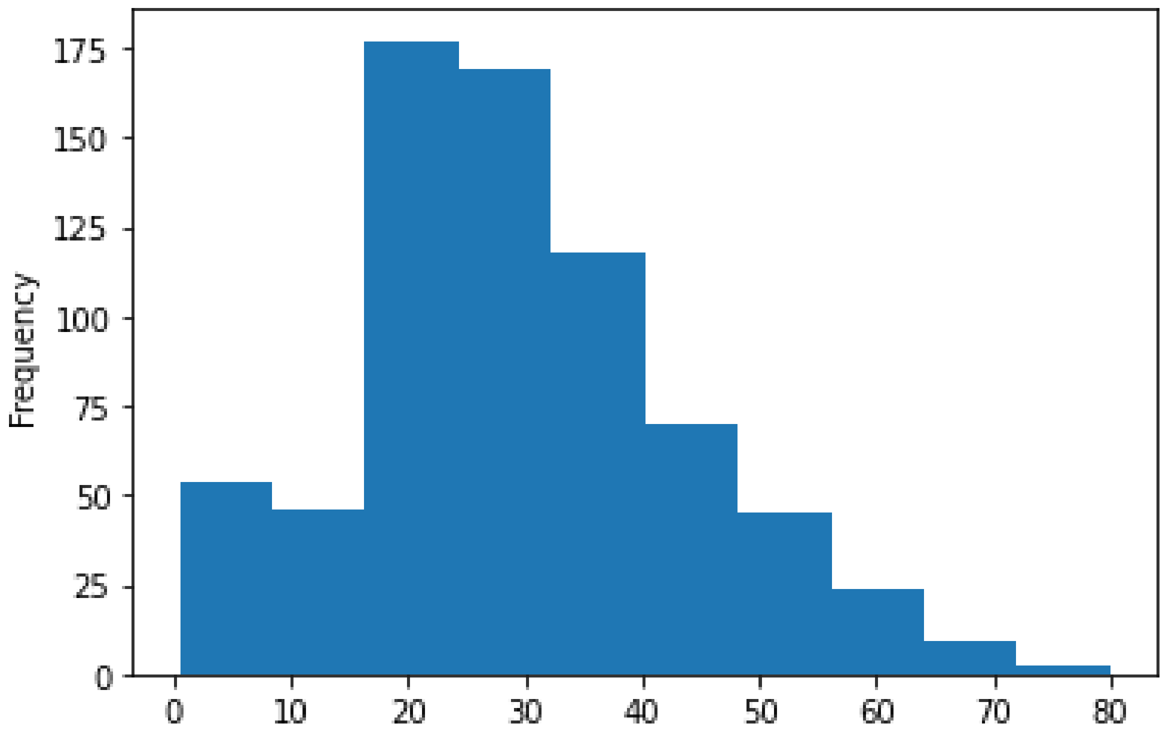
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Name      891 non-null    object
 3   Sex       891 non-null    object
 4   Age       714 non-null    float64
 5   SibSp     891 non-null    int64
 6   Parch     891 non-null    int64
 7   Ticket    891 non-null    object
 8   Fare      891 non-null    float64
 9   Cabin     204 non-null    object
 10  Embarked  891 non-null    int64
dtypes: float64(2), int64(5), object(4)
memory usage: 83.5+ KB
```

Age Column

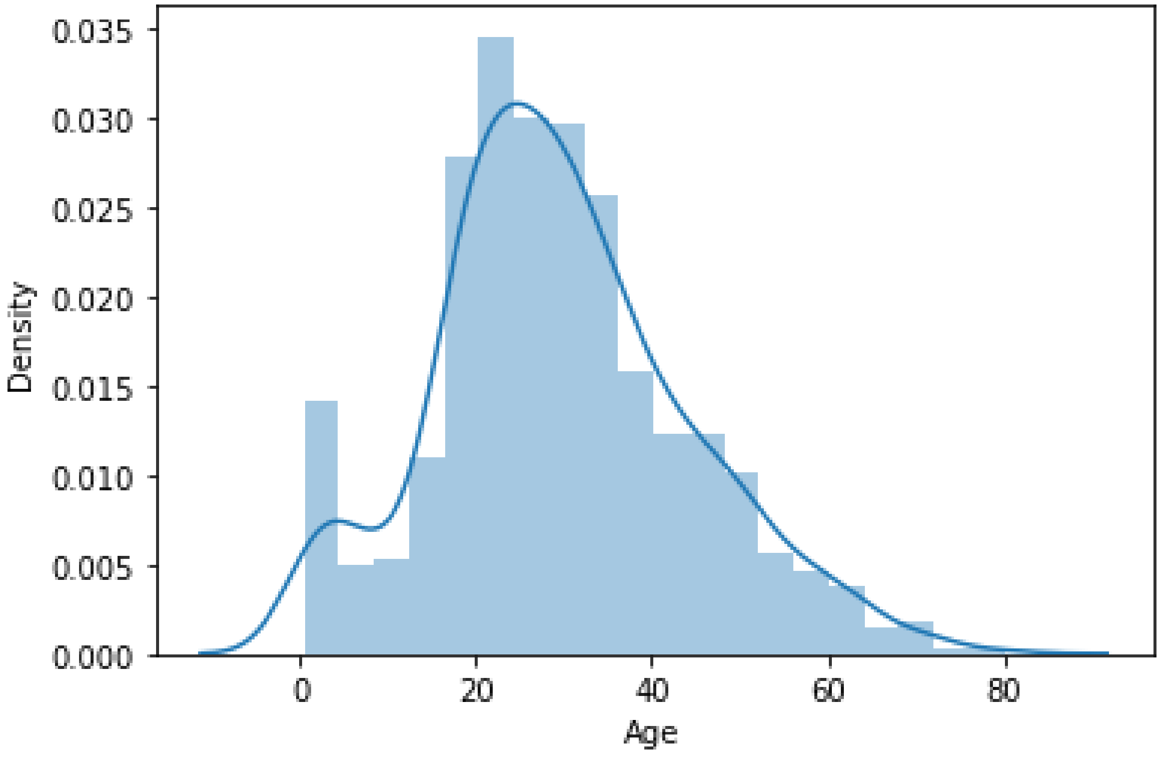
- **Displays the missing value in the Age column**

The total number of data entries is 891, while the column age is 714. Means there is null data in column Age, cause of that we will do an imputation on column Age. To determine what methods will be used, so we have to display the histogram column Age.

```python
df.Age.plot(kind='hist');
```




```python
import seaborn as sns
sns.distplot(df['Age'])
```



Note: The histograms show the column age has a kurtosis

Because of the age column has a skewness distribution, then the imputation in the age column will use the median.

```
val = df.Age.median()
df['Age'] = df.Age.fillna(val)
```

Display df.info() to check if the Age column has been imputed. It turns out that the Age column has now changed in number and has no null value.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Name      891 non-null    object
 3   Sex       891 non-null    object
 4   Age       891 non-null    float64
 5   SibSp     891 non-null    int64
 6   Parch     891 non-null    int64
 7   Ticket    891 non-null    object
 8   Fare      891 non-null    float64
 9   Cabin     204 non-null    object
 10  Embarked  891 non-null    int64
dtypes: float64(2), int64(5), object(4)
memory usage: 83.5+ KB
```

Cabin Column

- **Show nulls data in the Cabin column**

The total number of data entries is 891, while the Cabin column is 204, means there is null data in the Cabin column. This how to show Cabin column. This how to show proportion data of Cabin column

```
df.Cabin.value_counts()

B96 B98        4
G6             4
C23 C25 C27    4
C22 C26        3
F33            3
              ..
E34            1
C7             1
C54            1
E36            1
C148           1
Name: Cabin, Length: 147, dtype: int64
```

See that the Cabin column has too much Unique data and also the column cabin info is not very informative to find out Survived data

```
df.drop('Cabin', axis=1, inplace =True)
```

Because of that we have to delete the Cabin column

Them show dataset info to see if the cabin column will be deleted.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Pclass    891 non-null    int64  
 2   Name      891 non-null    object 
 3   Sex       891 non-null    object 
 4   Age       891 non-null    float64
 5   SibSp     891 non-null    int64  
 6   Parch     891 non-null    int64  
 7   Ticket    891 non-null    object 
 8   Fare      891 non-null    float64
 9   Embarked  891 non-null    int64  
dtypes: float64(2), int64(5), object(3)
memory usage: 76.6+ KB
```

Name Column

- **Delete Name column due to uninformative reason**

Since the Name column has too many unique data, and also the Name info column is not very informative to find out Survived data, we will delete the Name column.

use inplace = True for permanent result.

Show dataset info tocheck if the cabin column will be deleted. An the result it turns out that the column cabin now has been removed.

```
df.drop('Name', axis=1, inplace =True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Sex       891 non-null    object
 3   Age       891 non-null    float64
 4   SibSp     891 non-null    int64
 5   Parch     891 non-null    int64
 6   Ticket    891 non-null    object
 7   Fare      891 non-null    float64
 8   Embarked  891 non-null    int64
dtypes: float64(2), int64(5), object(2)
memory usage: 69.6+ KB
```

Sex Column

- **Convert Sex column data type from object to numeric**

Column Age is still an Object data type, to facilitate the analysis process, it must be converted the object type to a numeric type.

```
df.Sex = df.Sex.map({'male':0, 'female':1})
```

Show dataset info to see if the Age column has changed its data type. An the result it turns out that the Age column has now changed its data type to numeric.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Sex       891 non-null    int64
 3   Age       891 non-null    float64
 4   SibSp     891 non-null    int64
 5   Parch     891 non-null    int64
 6   Ticket    891 non-null    object
 7   Fare      891 non-null    float64
 8   Embarked  891 non-null    int64
dtypes: float64(2), int64(6), object(1)
memory usage: 69.6+ KB
```

Ticket Column

- **Delete Ticket column due to uninformative reason**

Since the Ticket column has too many unique data, and also the Name info column is not very informative to find out Survived data, we will delete the Name column.

use inplace = True for permanent result.

Show dataset info tocheck if the cabin column will be deleted. An the result it turns out that the column cabin now has been removed.

```
df.drop('Ticket', axis=1, inplace = True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Pclass    891 non-null    int64  
 2   Sex       891 non-null    int64  
 3   Age       891 non-null    float64
 4   SibSp     891 non-null    int64  
 5   Parch     891 non-null    int64  
 6   Fare      891 non-null    float64
 7   Embarked  891 non-null    int64  
dtypes: float64(2), int64(6)
memory usage: 62.6 KB
```

Data Visualization for Survived Passengers

Survived Passengers Visualization

- **To do visualization, import required packages**

Import matplotlib and seaborn packages to do some visualization in Python

```
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
```

Now we will visualize the survivor data and firstly we have to display the proportion of the data survived.

```
df.Survived.value_counts()

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

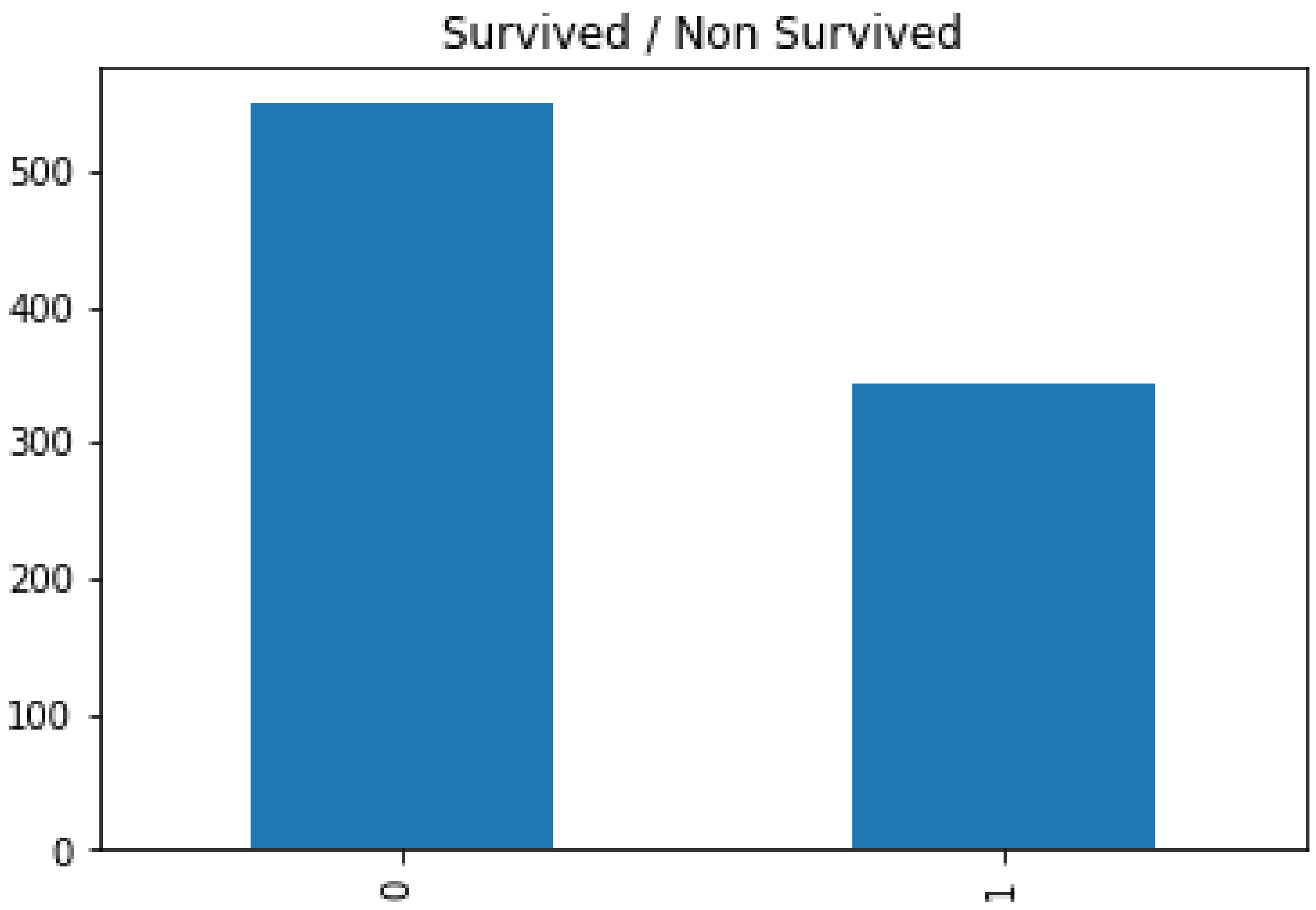
And the following is a visualization of the surviving passengers taken from the Survived column proportion data.

```
df.Survived.value_counts().plot(kind='bar')
plt.title('Survived / Non Survived');
```

Survived Passengers Visualization

- Create Visualization

And the following is a visualization of the surviving passengers taken from the Survived column proportion data.



Display a data frame from the Survived column

```
df_survived = pd.DataFrame(df.Survived.value_counts())
```

```
df_survived['Status']=[0,1]
```

```
df_survived
```

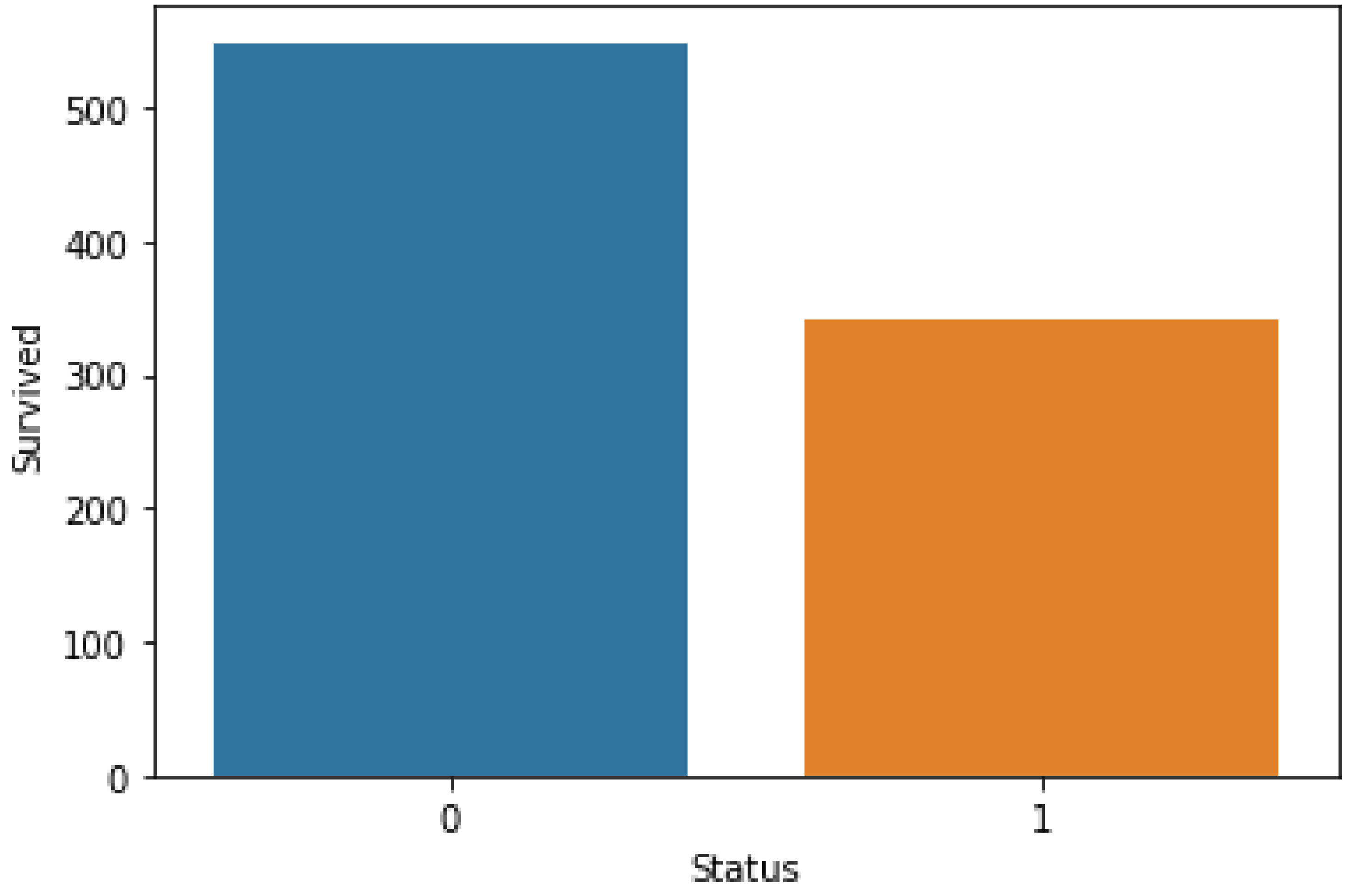
	Survived	Status
0	549	0
1	342	1

Survived Passengers Visualization

- **Changing data labels on charts**

then we recreate the visualization from the dataframe

```
sns.barplot(x='Status',y='Survived' , data=df_survived);
```



We can also change the status, the example is changed to died and alive

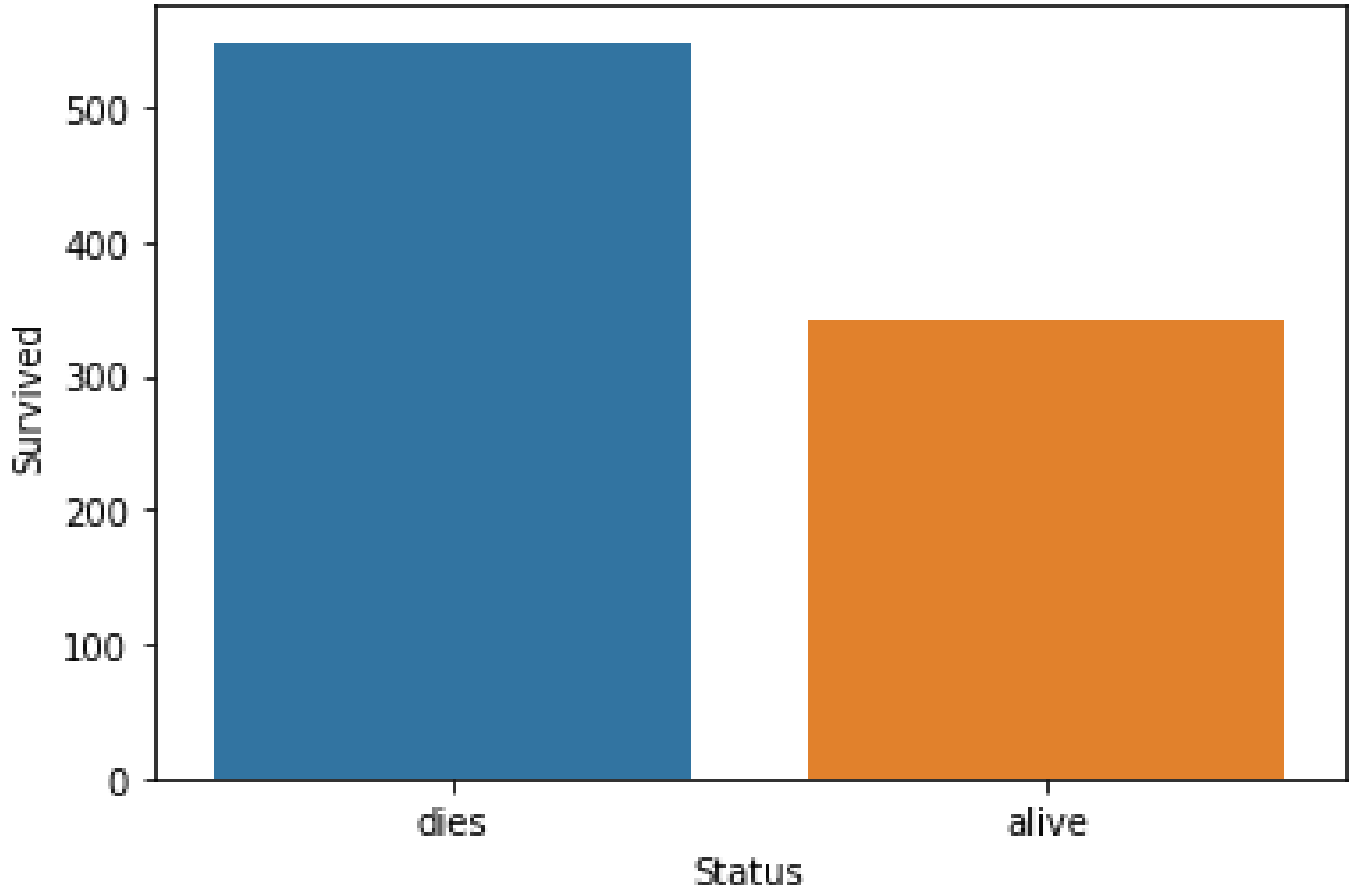
```
df_survived2 = pd.DataFrame(df.Survived.value_counts())
df_survived2['Status']=['dies','alive']
df_survived2
```

	Survived	Status
0	549	dies
1	342	alive

Survived Passengers Visualization

And apply the modified data label on the chart

```
sns.barplot(x='Status',y='Survived' , data=df_survived2);
```



Conclusion: From the results of exploratory data analysis and data visualization that has been carried out, it shows that there are 342 passengers who survived and 549 passengers who died.

Thank You

Github Page: https://github.com/deniriswana/Data-Cleansing-and-Exploratory-Data-Analysis-of-Titanic-DataSet