MyEduSolve

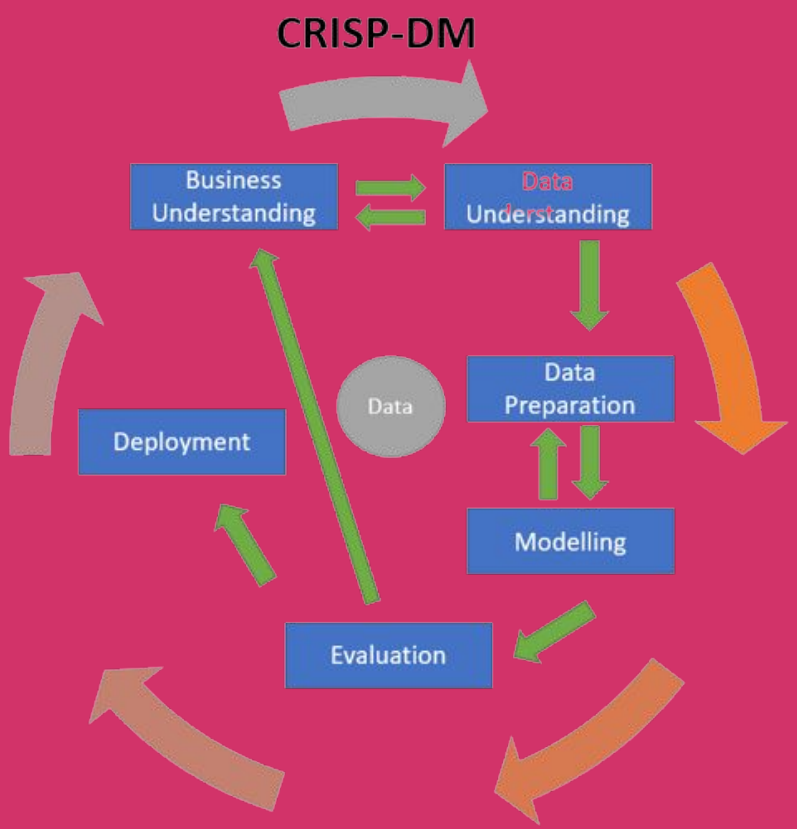
Stages of Data Preparation

created by Sintaks Group

Sintaks Group

This project is created by **Sintaks Group**, with the following members:

- Hasballah Askar
- Galang Setia Nugroho
- Khalishah Fiddina
- Tifani Amalina
- Muhammad Ilham Hakiqi



Data Preparation is one of six stages contained in the crisp-dm methodology.

Data Preparation is the process of gathering, combining, structuring, and organizing data so it can be used in business intelligence (BI), analytics, and data visualization applications. The components of data preparation include **data preprocessing, profiling, cleansing, validation, and transformation.**

Data Cleansing

Data Cleansing is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset.

1. Import the required libraries

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

Note:

- **pandas**, for data manipulation and analysis
- **numpy**, used for working with arrays
- **matplotlib** and **seaborn** for visualization

2. Create a data frame then load the dataset. Here, the dataset used is data Titanic.csv

If you're using Google Colab, use this code below to upload the data file:

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

```
Choose Files  Titanic.csv
• Titanic.csv(text/csv) - 61194 bytes, last modified: 4/16/2022 - 100% done
Saving Titanic.csv to Titanic.csv
{'Titanic.csv': b'PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,P
```

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

3. Show the top 10 rows of the data set that has been loaded

```
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

4. Check data condition

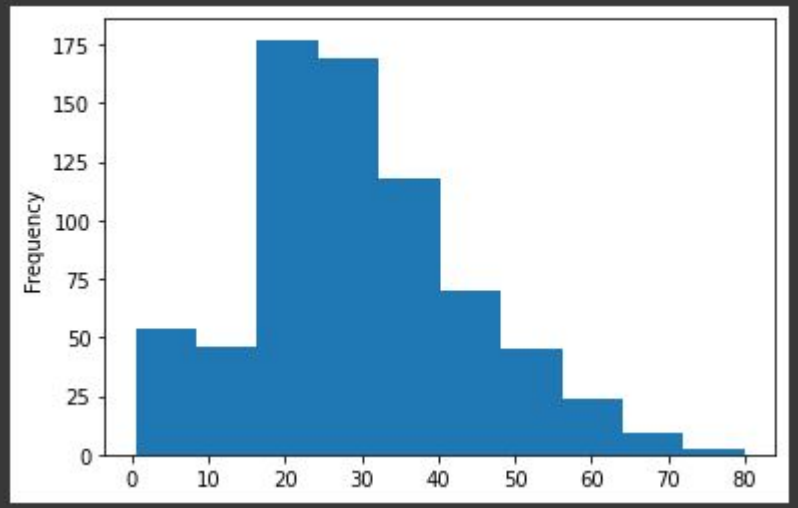
```
df.info()
```

```
<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
```

Column Age

1. Show visualization of the data in column 'Age' in histogram

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



The total data entry is 891 rows, meanwhile column 'Age' only has 714 rows.
This means column 'Age' has null values, so imputation needs to be done on this column.

2. Because the data distribution of column 'Age' is skewed, imputation is done using median.

The code below is used to fill the null-value with the median.

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

Show the dataset information to see whether column 'Age' has been imputed or not. Turns out the number of rows in ‘Age’ column has changed.

```
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
```

Column Cabin

The total number of data entries is 891, meanwhile ‘Cabin’ column is 204.
This means that ‘Cabin’ column has null values.

1. Show the data proportion of 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 ‘Cabin’ column value has too many unique data and also information in ‘Cabin’ column is not very informative to find out about the Survived data.

2. Therefore, will permanently delete the Cabin column

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

Display the dataset information to see whether column 'Cabin' 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
```

It turns out that the ‘Cabin’ column doesn't exist anymore

Column Embarked

The total number of data entries is 891, meanwhile ‘Embarked’ column is 889. This means that ‘Embarked’ column has null values.

1. Check the location or index of the null values

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

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

2. Show the data proportions of column 'Embarked'

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

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

Turns out that ‘Embarked’ column data is in the form of categorical data.

If we want to do imputation on ‘Embarked’, we need to check the data type of column 'Embarked' first.

3. Because the data is categorical, imputation is done using mode (Modus)

As you can see, 'S' is the data that appears most often, then 'S' is the mode of this column.

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

After we do the imputation, we can see that the proportion has changed

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

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

Column SibSp dan Column Parch

1. Here, we will do data manipulation to make the data easier to read by the machine.

   Column 'SibSp' (Sibling Spouse), states the number of siblings or spouses brought by the passenger.
   Column 'Parch' (Parent Children), states the number of parents or children brought by the passenger.
   We will create a new column called 'Alone' which states whether the passenger is alone or with family.

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

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

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

2. Show only the top rows of the latest dataset

```
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

Relations Between Column Sex and Column Survived

Check the data proportion of column 'Sex' that survived

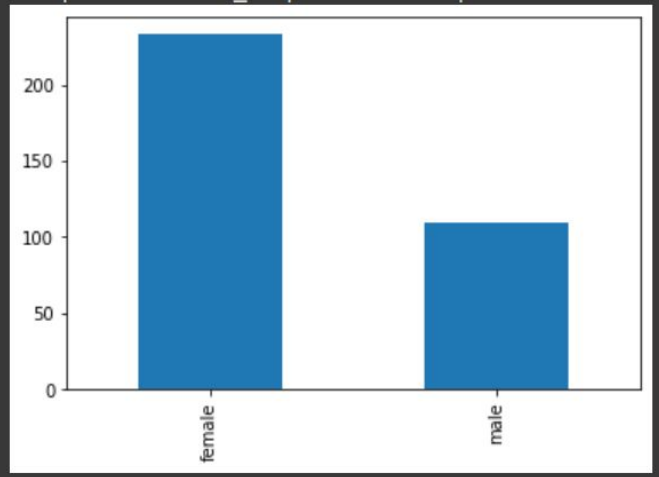
```
df['Sex'][df['Survived']==1].value_counts()
```

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

Display the data visualization of column 'Sex' that survived

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd80a39edd0>
```



Check the data proportion of column 'Sex' that is not survived

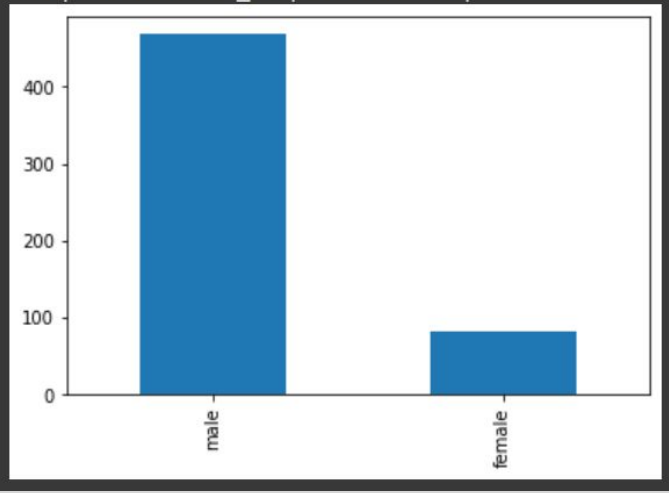
```
df['Sex'][df['Survived']==0].value_counts()
```

```
male      468
female     81
Name: Sex, dtype: int64
```

Display the data visualization of column 'Sex' that is not survived

```
df['Sex'][df['Survived']==0].value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd80a1092d0>
```



Exploratory Data Analysis

Exploratory Data Analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.

- Have to import the following library first

```
[23] #import packages
     import pandas as pd
```

Upload files

```
[7] from google.colab import files
    files.upload()
```

Choose Files Titanic.csv

- **Titanic.csv**(text/csv) - 61194 bytes, last modified: 4/12/2022 - 100% done

```
Saving Titanic.csv to Titanic.csv
{'Titanic.csv': b'PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked\r\n1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,
```

Remove the default python index starting from 0. change the index to the passenger id

before:

```
[26] df = pd.read_csv("Titanic.csv")
```

```
[27] 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

After:

```
[28] df = pd.read_csv("Titanic.csv",index_col=0)
```

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

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

find info from data by using df.info() to find out which column, data type and non-null data

```
[30] 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
```

by using the syntax df.isnull().sum() we see the number of null values of each dataset variable

```
[13] 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
```

by using the syntax df.describe() we see mean,std, min, etc. of each dataset variable

```
[14] 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 value in that column for example column sex

```
[16] df.Sex.unique()

     array(['male', 'female'], dtype=object)
```

Display a number of unique value in that column

```
[31] df.Sex.nunique()

     2
```

Show the propotion of unique value in that column

```
[20] df.Sex.value_counts()

     male      577
     female    314
     Name: Sex, dtype: int64
```

display the number of rows and the number of columns of the dataset

```
[21] df.shape

     (891, 11)
```

Show presence of duplicate data

```
[32] df.duplicated()

     PassengerId
     1      False
     2      False
     3      False
     4      False
     5      False
            ...
     887    False
     888    False
     889    False
     890    False
     891    False
     Length: 891, dtype: bool
```

Drop duplicate data from dataset

```
[33] 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

Column Embarked

The total number of data entries is 891, while the `Embarked` column is 889. This means that there is `null` data in the `Embarked` column. Check the location where the null data is

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

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

Show the proportion of `Embarked` column data. It turns out that the data column `Embarked` is in the form of categoric data

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

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
PassengerId											
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 the propotion of the embarked column to see if any data categorical

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

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

If we are going to imputation on column `Embarked`, So it is necessary to check the data type column `Embarked` first. The data column `Embarked` is in the form of categoric data, then imputation uses mode From the proportion of `Embarked` column, 'S' is the power that occurs most often, then S is the mode

```
[37] 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 `Embarked` is still an Object data type, to facilitate the analysis process, then it will be converted to Numeric data type

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

```
[40] 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
```

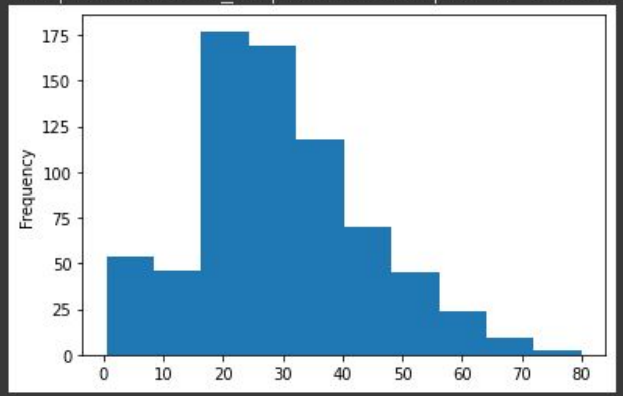
display dataset info to see if the Embarked column has changed its data type, it turns out that the column `Embarked` has now changed its data to Numeric

Column Age

The total number of data entries is 891, while the `Age` column is 714. This means that there is `null` data in the `Age` column. An imputation will be carried out on the `Age` column. To determine what methods we will use in the inputation column `Age`, then we display the histogram column `Age`

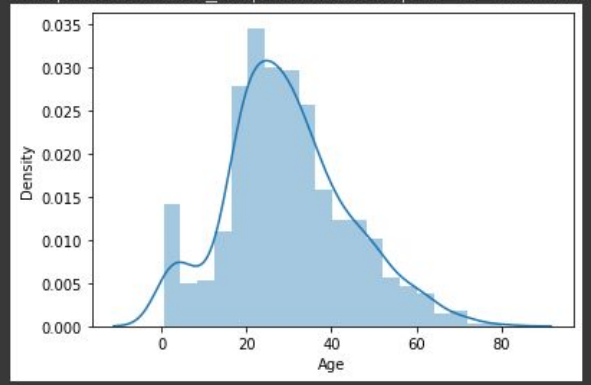
```
[42] df.Age.plot(kind='hist')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f034aa50a10>
```




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

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f033b943dd0>
```



Because column `Age` has Skewness distribution, it will imputation on column `Age` using Median

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

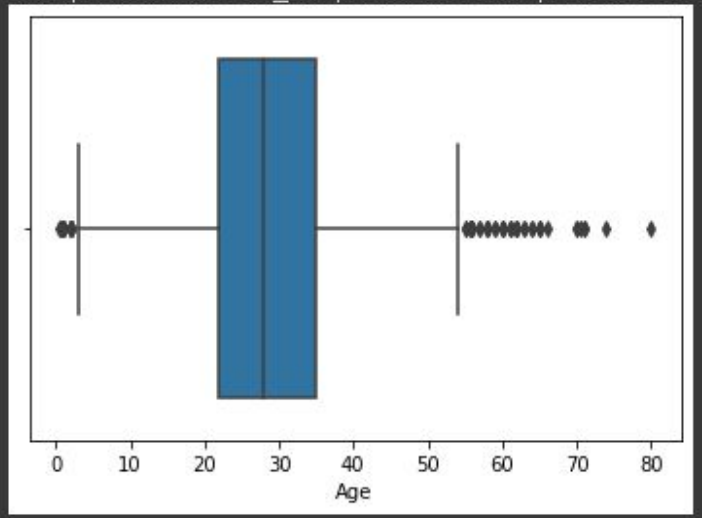
```
[47] df.info()

     <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       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  0 non-null      float64
     dtypes: float64(3), int64(4), object(4)
     memory usage: 83.5+ KB
```

Show dataset info to see if column `Age` has been imputed. It turns out that the column 'Age' has now changed in number

to see visualization of outliers in column `Age` using boxplot

```
[48] sns.boxplot(df['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:4
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f0339869990>
```



Column Cabin

- The total number of data entries is 891, while the `Cabin` column is 204. This means that there is `null` data in the `Cabin` column.

- Displays the proportion of the 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
```

- In addition, there are many uniqueness in the cabin column and the data is less informative to use. thing we can do is remove the cabin column.

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

```
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       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   Embarked  891 non-null    int64
dtypes: float64(2), int64(5), object(3)
memory usage: 76.6+ KB
```

Column Name and Ticket

- If you find a lot of unique in a column in a data, and the data is less informative to use. thing we can do is to delete the column.

For Column Ticket:

```
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       714 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
```

Column Sex

- To facilitate the process of analyzing data, you can change the object's data type to a numeric data type.
- Can use encoder instead of map

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

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

```
0    577
1    314
Name: Sex, dtype: int64
```

Data Visualization

for example, we will create a data visualization of "survived"

- To perform visualization, import the required packages

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

    import seaborn as sns
```

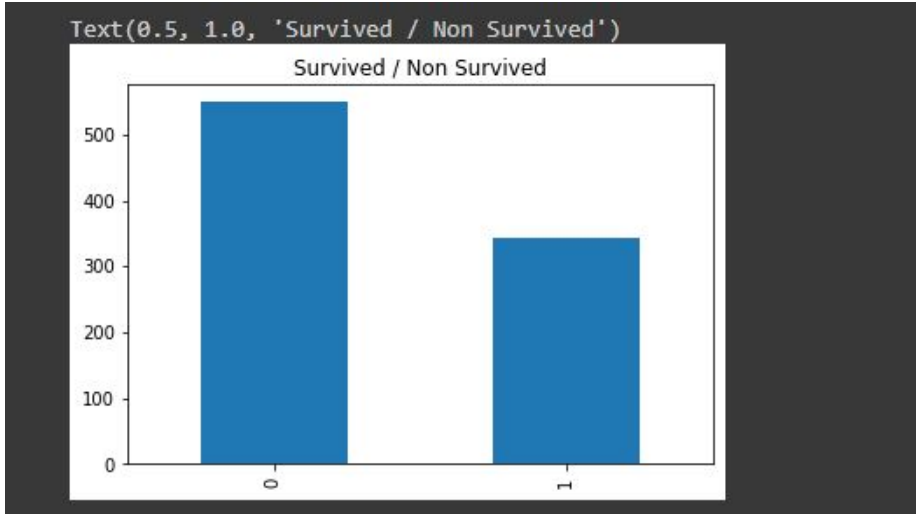
- First show the proportion of Survived data

```
df.Survived.value_counts()

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

Data Visualization

- The following is a syntax that can be used to display the results of the visualization

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



Data Visualization

- Create a data frame from column Survived

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

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

```
df_survived
```

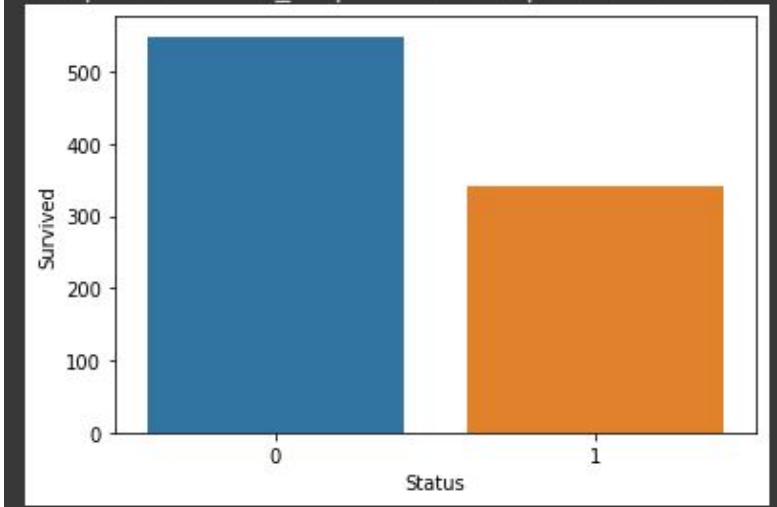
	Survived	Status
0	549	0
1	342	1

Data Visualization

- Displaying visualization

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f74b9c09ed0>
```



Data Visualization

- So that the data can be easily read, we can also replace the information contained in a column. For example, we will replace the information contained in the status column. which was originally 0 and 1, we change to life and dead. 0 for passengers who dead and 1 for passengers who alive.

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

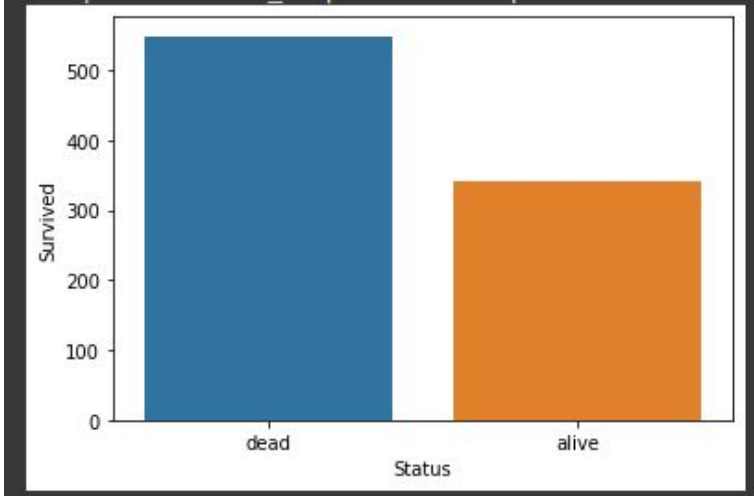
	Survived	Status
0	549	dead
1	342	alive

Data Visualization

- Displaying visualization

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f74b9702550>
```



Thanks!