In [1]:
```
import pandas as pd
import numpy as np
```

```
C:\Users\yashm\anaconda3\lib\importlib\_bootstrap.py:219: RuntimeWarning: numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from C header, got 216 from PyObject
  return f(*args, **kwds)
```

In [3]:
```
df = pd.read_csv('Titanic.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	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

Detecting Missing Values

In [6]:
```
df.isna().sum() # Find out the number of missing values for each column
```

Out[6]:
```
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 [7]:
```
100*df.isna().sum()/df.shape[0] #Percentage of missing values in each column
```

Out[7]:
```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
dtype: float64
```

1. If we have more than 30% of values in a column that are missing then we drop it
2. If we have 20-30% of values missing in any column then we have to take a call according to domain knowledge i.e whether to drop the column or impute the missing values
3. If we have less than 20% of values as missing values, as a general rule of thumb if it is an important attribute we will fill the NA values

```
### Delete the rows with all missing values
```

In [10]:
```
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
```

In [11]:
```
df.dropna().info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 183 entries, 1 to 889
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  183 non-null    int64  
 1   Survived     183 non-null    int64  
 2   Pclass       183 non-null    int64  
 3   Name         183 non-null    object 
 4   Sex          183 non-null    object 
 5   Age          183 non-null    float64
 6   SibSp        183 non-null    int64  
 7   Parch        183 non-null    int64  
 8   Ticket       183 non-null    object 
 9   Fare         183 non-null    float64
 10  Cabin        183 non-null    object 
 11  Embarked     183 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 18.6+ KB
```

Pros vs Cons

Pros:

1. Model trained without any values imputed is a more robust model(Because no missing values were filled with user bias)

Cons:

1. Loss of a lot of data/information
2. Works very very poorly if one column has a high percentage of missing values(>50%)

Imputing missing values with Mean/Median

In [12]:
```
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
```

In [13]:
```
df.describe()
```

Out[13]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	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	14.526497	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	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [15]:
```
df['Age'].fillna(value = df['Age'].mean(),inplace = True)
```

Pros vs Cons

Pros:

1. Prevents data loss and loss of information
2. Works well when the dataset is small and easy to model

Cons:

1. Works only with Numerical Variables
2. Can cause data leakage

```
### Imputing missing values for Categorical variables
```

In [18]:
```
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
```

In [20]:
```
df[df.Embarked.isna()]
```

Out[20]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
61	62	1	1	Icard, Miss. Amelie	female	38.0	0	0	113572	80.0	B28	NaN
829	830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62.0	0	0	113572	80.0	B28	NaN

In [21]:
```
df.Embarked.value_counts()
```

Out[21]:
```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [24]:
```
df.Embarked.fillna(value = 'S')
```

Out[24]:
```
0      S
1      C
2      S
3      S
4      S
      ..
886    S
887    S
888    S
889    C
890    Q
Name: Embarked, Length: 891, dtype: object
```

Using Machine learning to Predict the missing values

In [61]:
```
df = pd.read_csv('Titanic.csv')
df.head()
```

Out[61]:

	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 [62]:
```
data = df.drop(['Name','Ticket','Cabin','Embarked','Sex','Survived','PassengerId'],1)
```

In [63]:
```
data.isna().sum()
```

Out[63]:
```
Pclass      0
Age       177
SibSp       0
Parch       0
Fare        0
dtype: int64
```

In [64]:
```
test_data = data[data['Age'].isna()]
train_data = data.dropna()
```

In [65]:
```
from sklearn.linear_model import LinearRegression
```

In [66]:
```
model = LinearRegression()

model.fit(train_data.drop('Age',1),train_data.Age)
```

Out[66]:
```
LinearRegression()
```

In [67]:
```
y_pred = model.predict(test_data.drop('Age',1))
```

In [68]:
```
len(y_pred)
```

Out[68]:
```
177
```

In [69]:
```
df['Age'].isna().sum()
```

Out[69]:
```
177
```

Scaling in ML

In [70]:
```
df.head()
```

Out[70]:

	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 [72]:
```
df = df.select_dtypes(include = np.number)
```

In [73]:
```
df
```

Out[73]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
0	1	0	3	22.0	1	0	7.2500
1	2	1	1	38.0	1	0	71.2833
2	3	1	3	26.0	0	0	7.9250
3	4	1	1	35.0	1	0	53.1000
4	5	0	3	35.0	0	0	8.0500
...	...	...	...	...	...	...	...
886	887	0	2	27.0	0	0	13.0000
887	888	1	1	19.0	0	0	30.0000
888	889	0	3	NaN	1	2	23.4500
889	890	1	1	26.0	0	0	30.0000
890	891	0	3	32.0	0	0	7.7500

891 rows × 7 columns

In [93]:
```
from sklearn.preprocessing import MinMaxScaler,StandardScaler
```

In [94]:
```
minmax = MinMaxScaler()
```

In [98]:
```
scaled_data = minmax.fit_transform(df)
```

In [96]:
```
scaled_data = pd.DataFrame(scaled_data,columns = df.columns)
```

In [97]:
```
scaled_data.describe()
```

Out[97]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.500000	0.383838	0.654321	0.367921	0.065376	0.063599	0.062858
std	0.289162	0.486592	0.418036	0.182540	0.137843	0.134343	0.096995
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.250000	0.000000	0.500000	0.247612	0.000000	0.000000	0.015440
50%	0.500000	0.000000	1.000000	0.346569	0.000000	0.000000	0.028213
75%	0.750000	1.000000	1.000000	0.472229	0.125000	0.000000	0.060508
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

In [100]:
```
unscaled_data = minmax.inverse_transform(scaled_data)
```

In [103]:
```
pd.DataFrame(unscaled_data,columns = df.columns)
```

Out[103]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
0	1.0	0.0	3.0	22.0	1.0	0.0	7.2500
1	2.0	1.0	1.0	38.0	1.0	0.0	71.2833
2	3.0	1.0	3.0	26.0	0.0	0.0	7.9250
3	4.0	1.0	1.0	35.0	1.0	0.0	53.1000
4	5.0	0.0	3.0	35.0	0.0	0.0	8.0500
...	...	...	...	...	...	...	...
886	887.0	0.0	2.0	27.0	0.0	0.0	13.0000
887	888.0	1.0	1.0	19.0	0.0	0.0	30.0000
888	889.0	0.0	3.0	NaN	1.0	2.0	23.4500
889	890.0	1.0	1.0	26.0	0.0	0.0	30.0000
890	891.0	0.0	3.0	32.0	0.0	0.0	7.7500

891 rows × 7 columns