DATA PREPARATION

TEAM MEMBERS

ABHIJITH
ANN MARYA
ANSIL
ILLIYAS
MUHAMMED ANSHEER
SHERIN
YADU

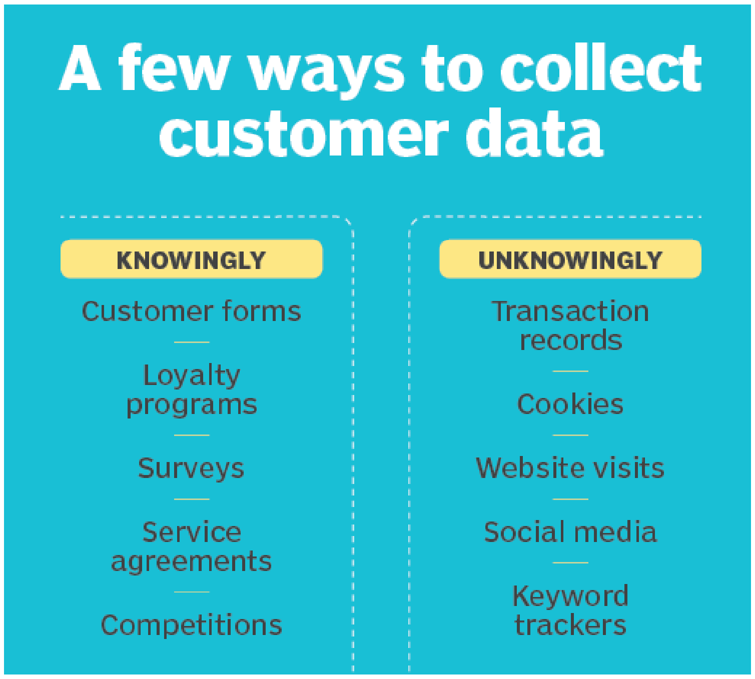
DATA PREPARATION

One of the primary purposes of data preparation is to **ensure that raw data being readied for processing and analysis is accurate and consistent** so the predictions will be accurate.

Data preparation is done in a series of steps. There's some variation in the data preparation steps listed by different data professionals and software vendors, but the process typically involves the following tasks:

1) Data collection

During this step relevant **data is gathered from operational systems, data warehouses, data lakes and other data sources**. The methods used to collect data vary based on the type of application. Some involve the use of technology, while others are manual procedures. For example, to analyze sales and the effectiveness of its marketing campaigns, a retailer might collect customer data from transaction records, website visits, mobile applications, its loyalty program and an online survey.



an example of customer data collection methods

There are different ways in which data can be collected,

- **Automated data collection functions** built into business applications, websites, Transactional Data and mobile apps.

  Analytics platforms like Google Analytics or Mixpanel, customer relationship management (CRM) tools like Salesforce, and e-commerce platforms like Shopify.

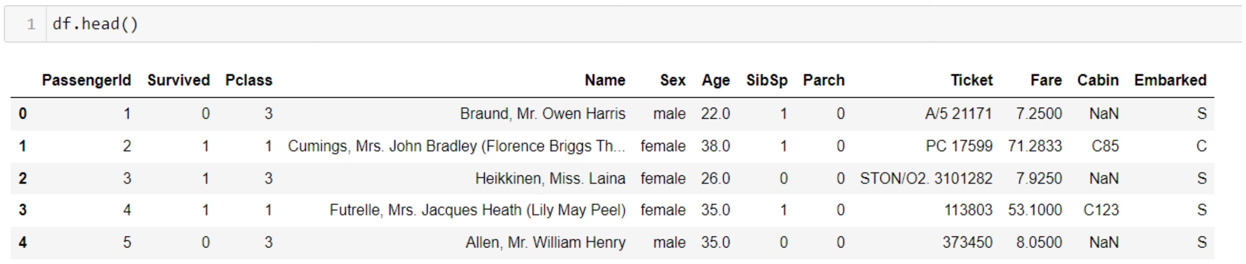
- **Sensors that collect operational data** from industrial equipment, vehicles and other machinery;
- **Collection of data from information services providers and other external data sources;**

  APIs, web scraping tools, and data integration platforms.
- **Tracking social media, discussion forums, reviews sites, blogs and other online channels;**

  Social media monitoring tools and web scraping tools.
- **Surveys, questionnaires and forms done online, in person or by phone, email or regular mail, polls;**

  Google forms, survey monkey etc.
- **Focus groups, Delphi method and one-on-one interviews;**
- **Direct observation of participants in a research study, Public Datasets**

Let's start with a data set Titanic collected from Kaggle



```
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

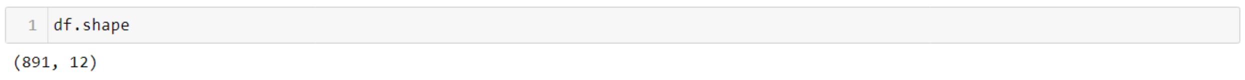
2) Data discovery and profiling

The next step is to **explore the collected data** to better understand what it contains and what needs to be done to prepare it for the intended uses. To help with that, data profiling **identifies patterns, relationships and other attributes in the data, as well as inconsistencies, anomalies, missing values and other issues** so they can be addressed.

The basic operations include:

2.1 Data Structure Understanding

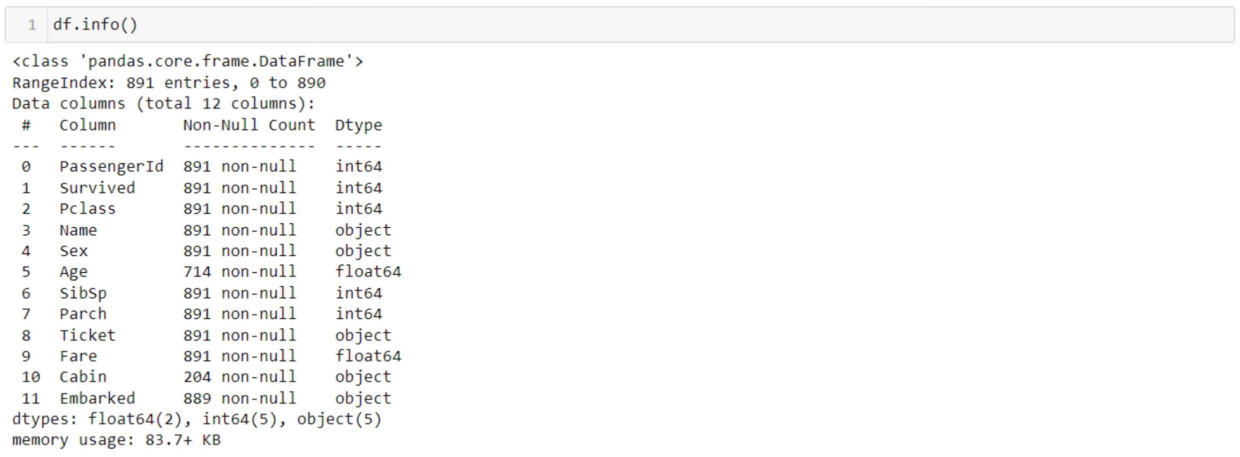
To understand the structure of the dataset, including the number of columns, data types, and relationships between different attributes. This understanding is essential for data preprocessing and data modeling tasks.



```
df.shape
```

```
(891, 12)
```

This will give you the number of columns and rows associated with your dataset, to get the details about the features (columns) we can use



```
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
```

2.2 Data Distribution Analysis

Helps to analyze the distribution of values within each column. This includes calculating summary statistics like mean, median, mode, minimum, maximum, standard deviation, and percentiles. Understanding the data distribution is vital for making data-driven decisions and identifying patterns or trends.

```python
df.describe()
```

```
       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
```

2.3 Cardinality and Uniqueness

Provides insights into the number of distinct values and the uniqueness of each attribute. This is useful for determining the level of variability in the data and identifying potential primary keys in relational databases.

```python
card = pd.DataFrame(columns=['total','unique'],index=df.columns)
card.total = df.count()
card.unique = [len(df[i].value_counts()) for i in df.columns ]
card
```

```
             total  unique
PassengerId    891     891
   Survived    891       2
     Pclass    891       3
       Name    891     891
        Sex    891       2
        Age    714      88
      SibSp    891       7
      Parch    891       7
     Ticket    891     681
       Fare    891     248
      Cabin    204     147
   Embarked    889       3
```

2.4 Data Completeness

Helps to assess the completeness of the data by determining the number of missing values in each column. Identifying missing data is important for imputation strategies or deciding whether a particular attribute should be included in the analysis.

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

```
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
```

2.5 Cross-column Analysis

Data profiling enables analysts to explore relationships and correlations between different columns. This includes identifying co-occurrences and dependencies between attributes.

2.5.1 Covariance

covariance explains how one or more variables are related to each other.

```
df.cov()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	66231.000000	-0.626966	-7.561798	138.696504	-16.325843	-0.342697	161.883369
Survived	-0.626966	0.236772	-0.137703	-0.551296	-0.018954	0.032017	6.221787
Pclass	-7.561798	-0.137703	0.699015	-4.496004	0.076599	0.012429	-22.830196
Age	138.696504	-0.551296	-4.496004	211.019125	-4.163334	-2.344191	73.849030
SibSp	-16.325843	-0.018954	0.076599	-4.163334	1.216043	0.368739	8.748734
Parch	-0.342697	0.032017	0.012429	-2.344191	0.368739	0.649728	8.661052
Fare	161.883369	6.221787	-22.830196	73.849030	8.748734	8.661052	2469.436846

2.5.2 Correlation

It is same as covariance, but gives the measure of that relationship

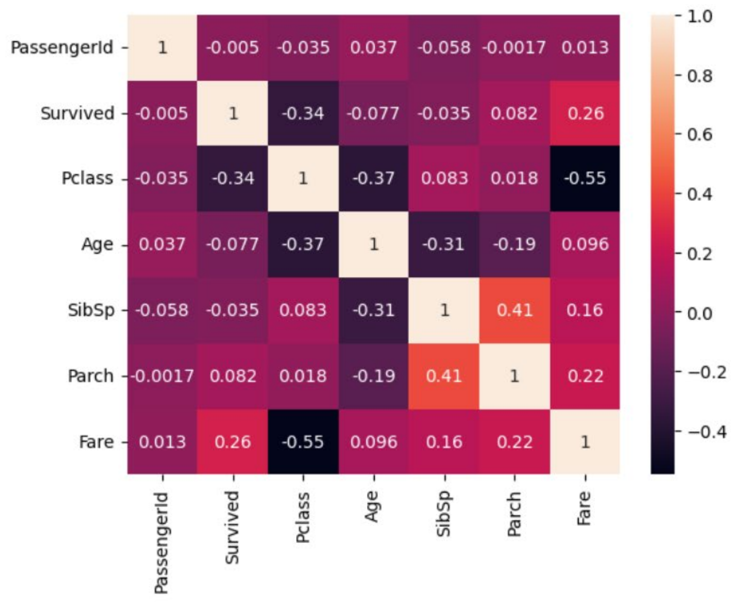
```
df.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

We can also use graphical representation for analyzing this

```
sns.heatmap(df.corr(), annot=True)
```

```
<Axes: >
```



	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1	-0.005	-0.035	0.037	-0.058	-0.0017	0.013
Survived	-0.005	1	-0.34	-0.077	-0.035	0.082	0.26
Pclass	-0.035	-0.34	1	-0.37	0.083	0.018	-0.55
Age	0.037	-0.077	-0.37	1	-0.31	-0.19	0.096
SibSp	-0.058	-0.035	0.083	-0.31	1	0.41	0.16
Parch	-0.0017	0.082	0.018	-0.19	0.41	1	0.22
Fare	0.013	0.26	-0.55	0.096	0.16	0.22	1

2.6 Crosstabulation (Cross Tab)

Crosstabulation is used to examine the relationship between two categorical variables. It creates a contingency table that shows the frequency distribution of the data for each combination of the two variables. They are useful for comparing the distribution of different categories across multiple attributes and identifying co-occurrences or dependencies.

```
pd.crosstab(df['Embarked'],df['Survived'], margins=True, margins_name="total")
```

Survived	0	1	total
Embarked			
C	75	93	168
Q	47	30	77
S	427	217	644
total	549	340	889

Here we can see the number of passengers survived based on the destination they are travelling

Automating Data discovery and profiling

We can automate the above discussed things by using the library pandas-profiling

```
pip install ydata-profiling
```

```
from ydata_profiling import ProfileReport
```

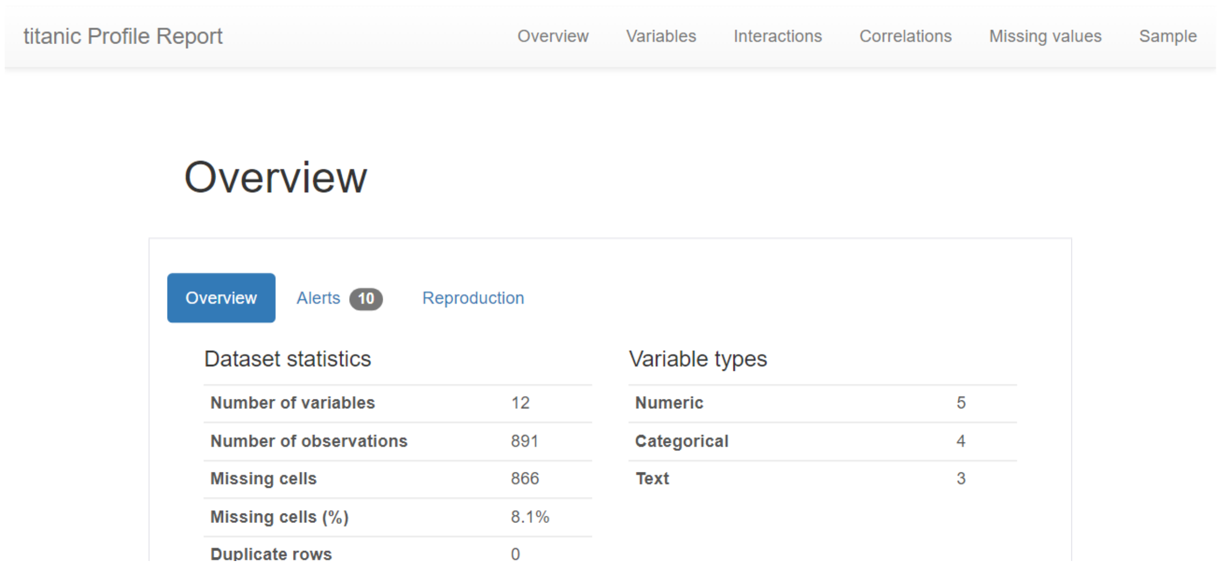
Run the following code to get an overview of our dataset

```
profile = ProfileReport(df,title="titanic Profile Report")
profile
```

```
Summarize dataset: 100%          47/47 [00:04<00:00, 5.12it/s, Completed]

Generate report structure: 100%          1/1 [00:04<00:00, 4.36s/it]

Render HTML: 100%          1/1 [00:01<00:00, 1.40s/it]
```



Dataset statistics

Number of variables	12
Number of observations	891
Missing cells	866
Missing cells (%)	8.1%
Duplicate rows	0

Variable types

Numeric	5
Categorical	4
Text	3

We can also save the report

```
profile.to_file("report.html")
```

```
Export report to file: 100%          1/1 [00:00<00:00, 3.03it/s]
```

Some tools used for data discovery and profiling

Open source data profiling tools

1. Quadient DataCleaner
2. Aggregate Profiler
3. Talend Open Studio

Commercial data profiling tools

1. Data Profiling in Informatica
2. Oracle Enterprise Data Quality
3. SAS DataFlux

3) Data cleansing

Next, the identified data errors and issues are corrected to create complete and accurate data sets. For example, as part of cleansing data sets, faulty data is removed or fixed, missing values are filled in and inconsistent entries are harmonized. The types of issues that are commonly fixed as part of data cleansing projects include the following:

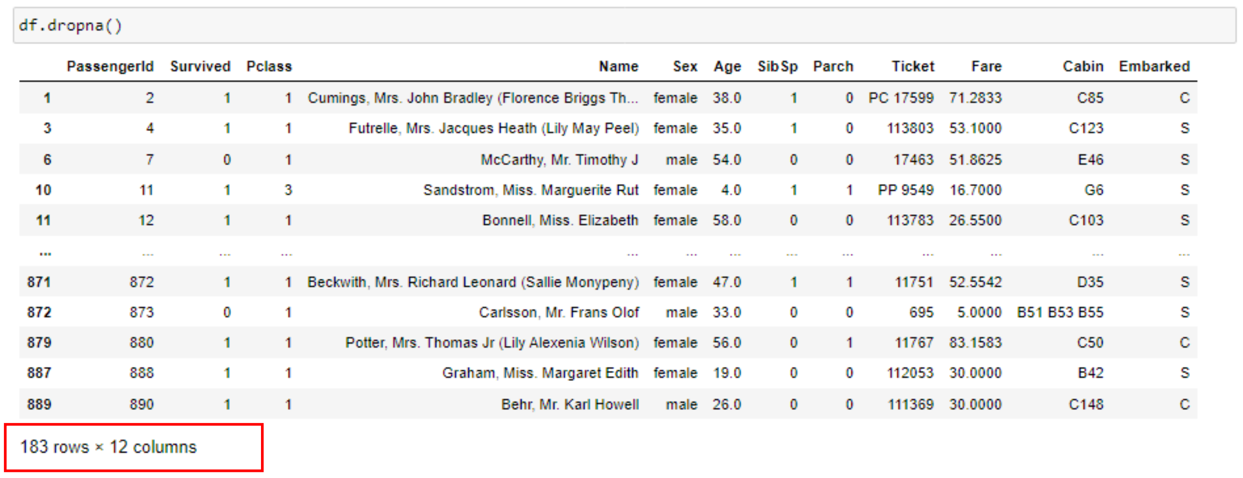
3.1 Missing data

In this step we deal with the missing data in our dataset notated as NaN or NULL, we deal this in several ways,

3.1.1 Deletion of row or column

Rows or Columns are getting deleted if the number of NULL values on each cross over a certain value, which will decrease the performance of our model.

3.1.1.1 Dropping rows

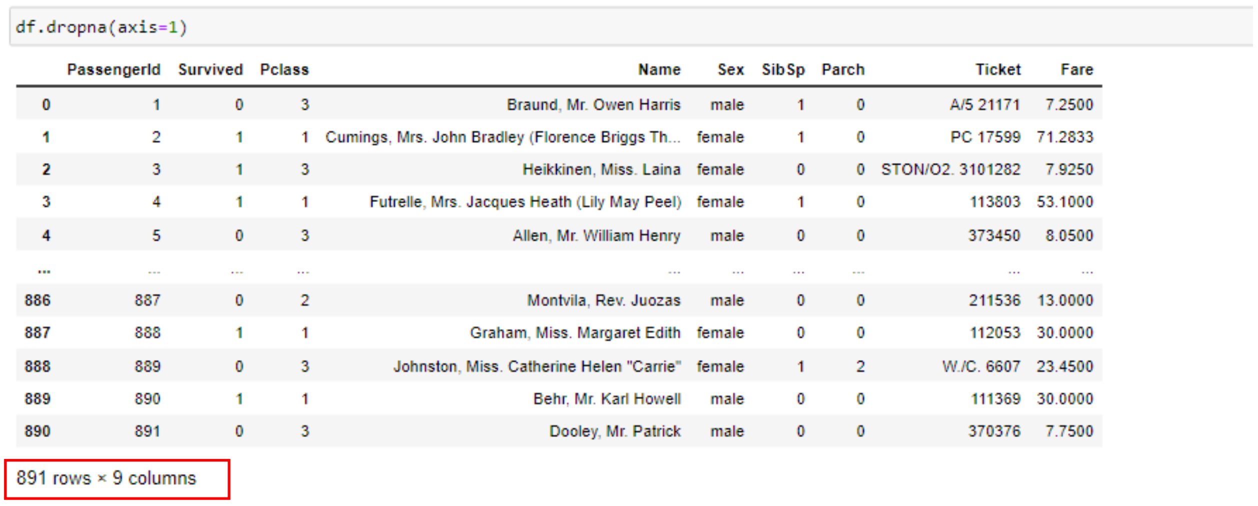


```
df.dropna()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
...	...	...	...	...	...	...	...	...	...	...	...	...
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.5542	D35	S
872	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	695	5.0000	B51 B53 B55	S
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583	C50	C
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C

183 rows × 12 columns

3.1.1.2 Dropping Columns



```
df.dropna(axis=1)
```

	PassengerId	Survived	Pclass	Name	Sex	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	0	0	373450	8.0500
...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	0	0	211536	13.0000
887	888	1	1	Graham, Miss. Margaret Edith	female	0	0	112053	30.0000
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	1	2	W./C. 6607	23.4500
889	890	1	1	Behr, Mr. Karl Howell	male	0	0	111369	30.0000
890	891	0	3	Dooley, Mr. Patrick	male	0	0	370376	7.7500

891 rows × 9 columns

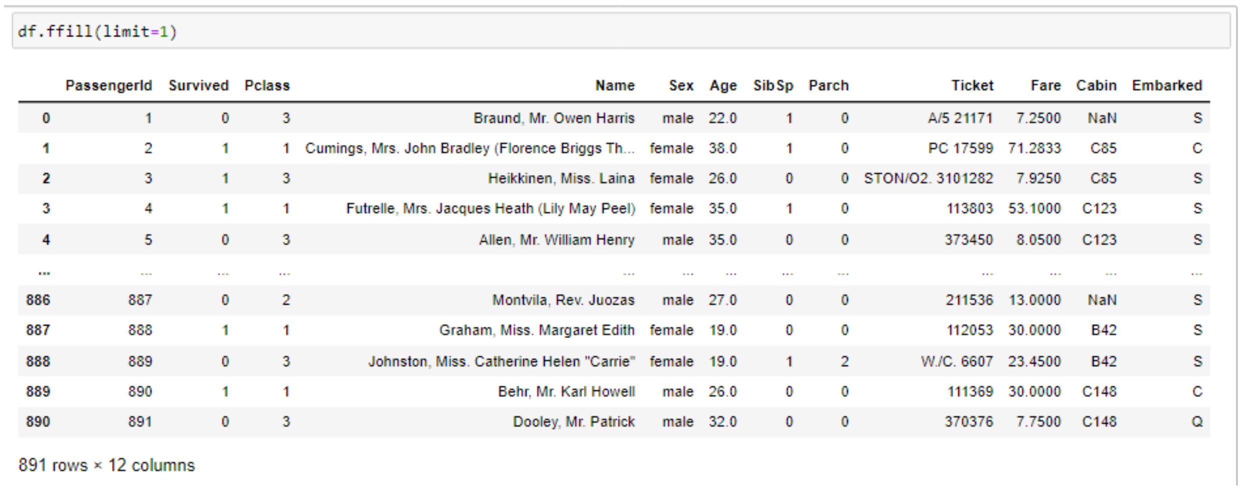
3.1.2 Imputation

Imputation involves replacing missing values with estimated values based on existing data. This allows you to retain all the data, and it can reduce the bias introduced by removing incomplete cases.

3.1.2.1 Forward Fill (ffill) and Backward Fill (bfill)

To propagate the last valid observation forward:

It will paste the previous value to the next cell if it is NULL



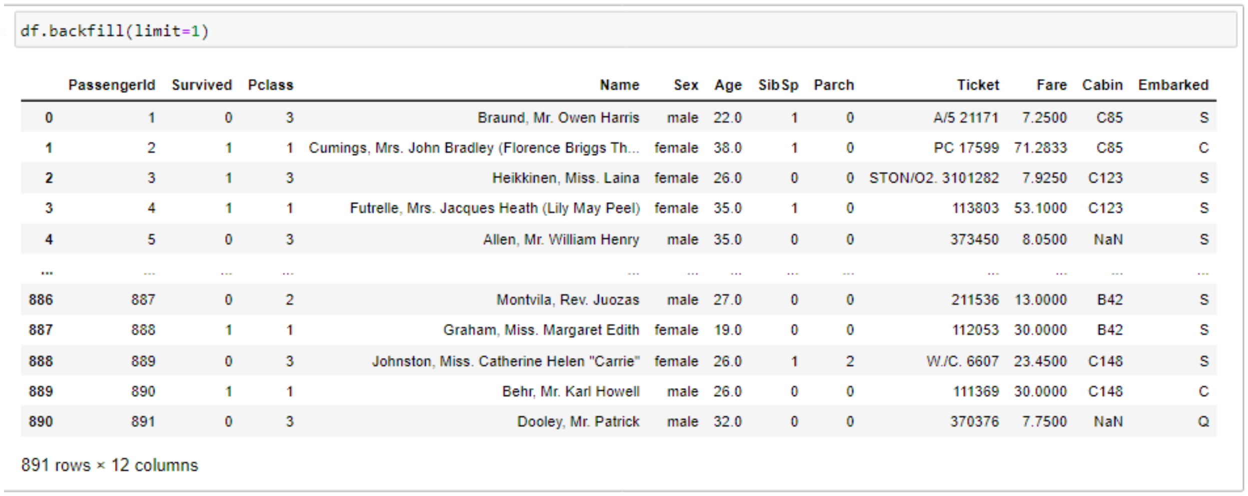
```
df.ffill(limit=1)
```

	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	C85	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	C123	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	19.0	1	2	W./C. 6607	23.4500	B42	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	C148	Q

891 rows × 12 columns

To propagate the next valid observation backward:

It will paste the next value to the current cell if it is NULL



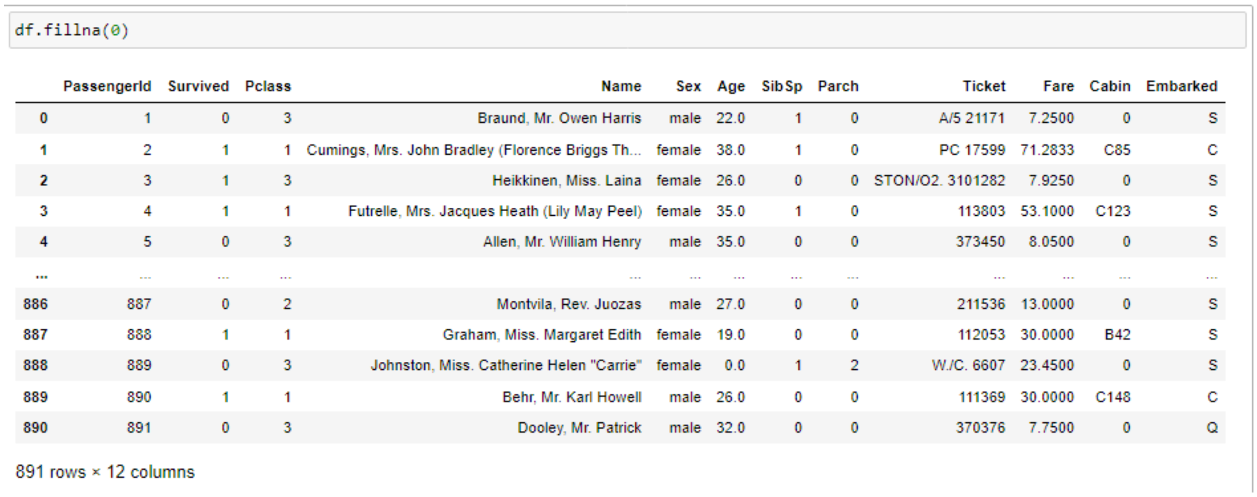
```
df.backfill(limit=1)
```

	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	C85	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	C123	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
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	B42	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	26.0	1	2	W./C. 6607	23.4500	C148	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

3.1.2.2 Simple imputation

Fill the NULL values with the specified values



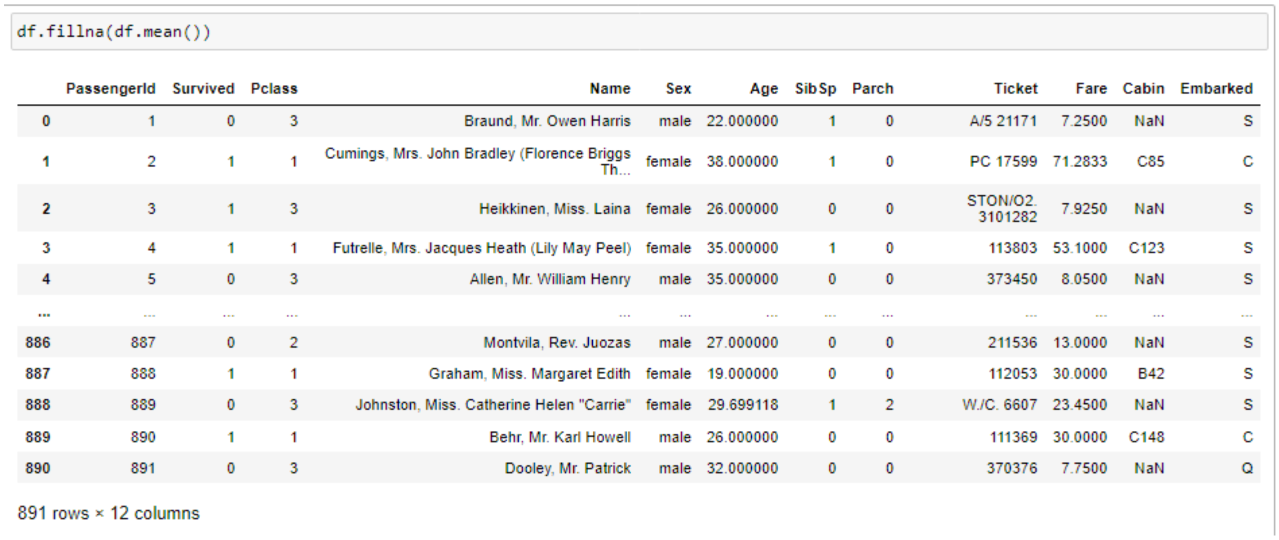
```
df.fillna(0)
```

	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	0	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	0	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	0	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	0	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	0.0	1	2	W./C. 6607	23.4500	0	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	0	Q

891 rows × 12 columns

3.1.2.3 Mean

Filling the null values with Mean value



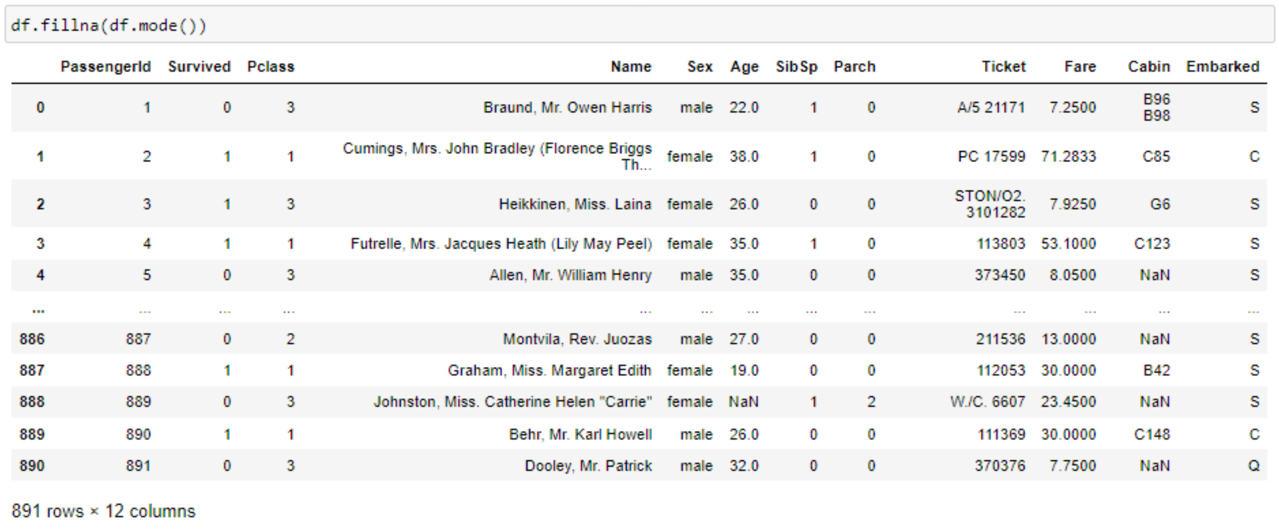
```
df.fillna(df.mean())
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

3.1.2.4 Mode

Fill the NULL values with Mode



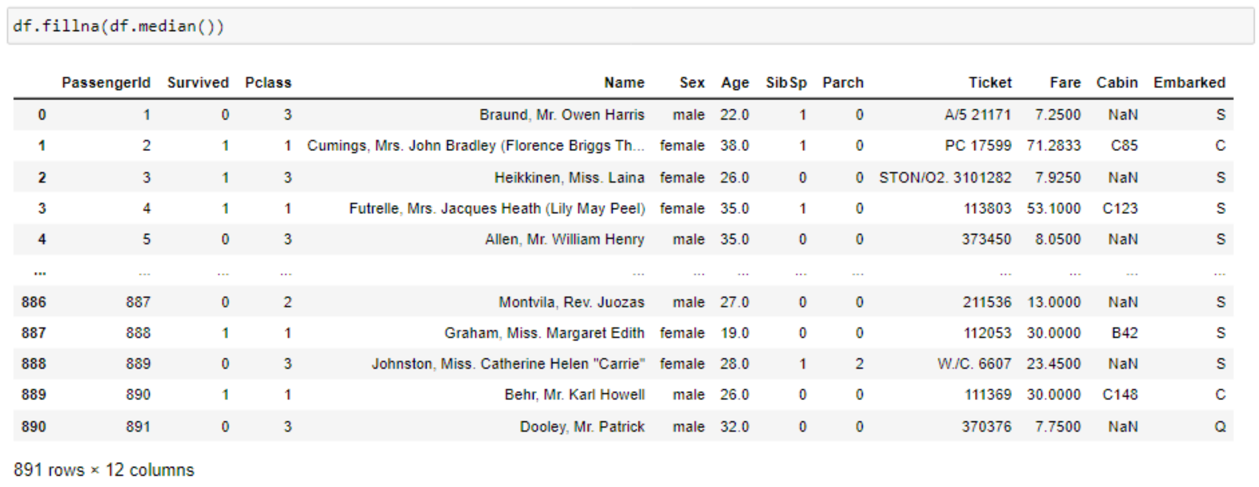
```
df.fillna(df.mode())
```

	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	B96 B98	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	G6	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
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

3.1.2.5 Median

Fill the NULL values with Median



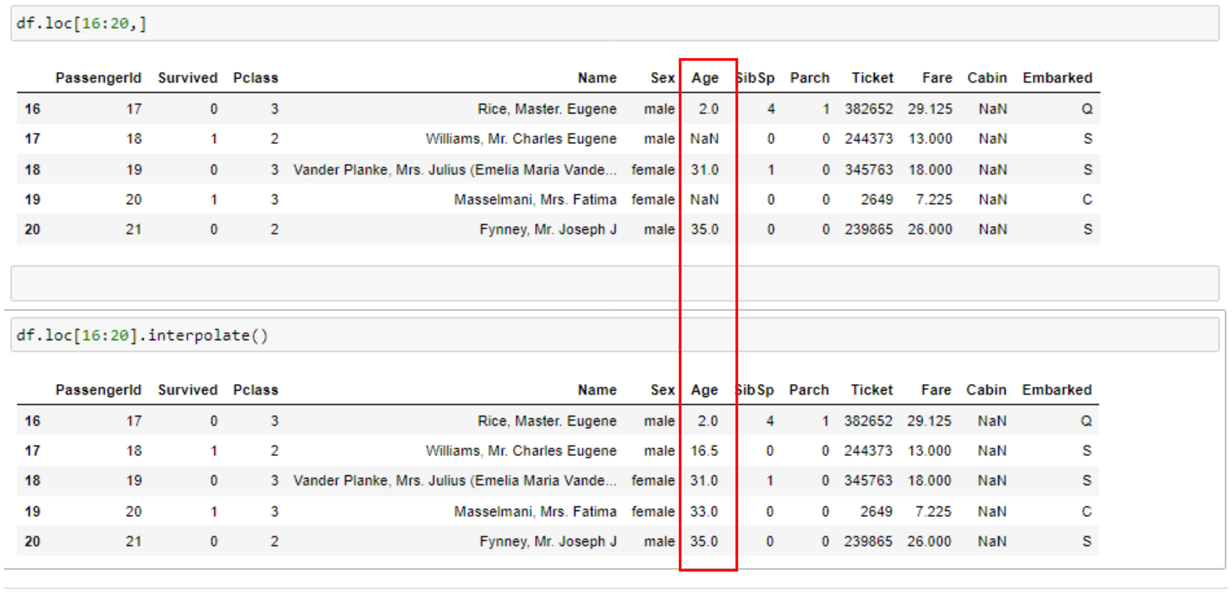
```
df.fillna(df.median())
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	28.0	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

3.1.2.6 Interpolate

Taking average of the previous and next value to fill the NULL value



```
df.loc[16:20,]
```

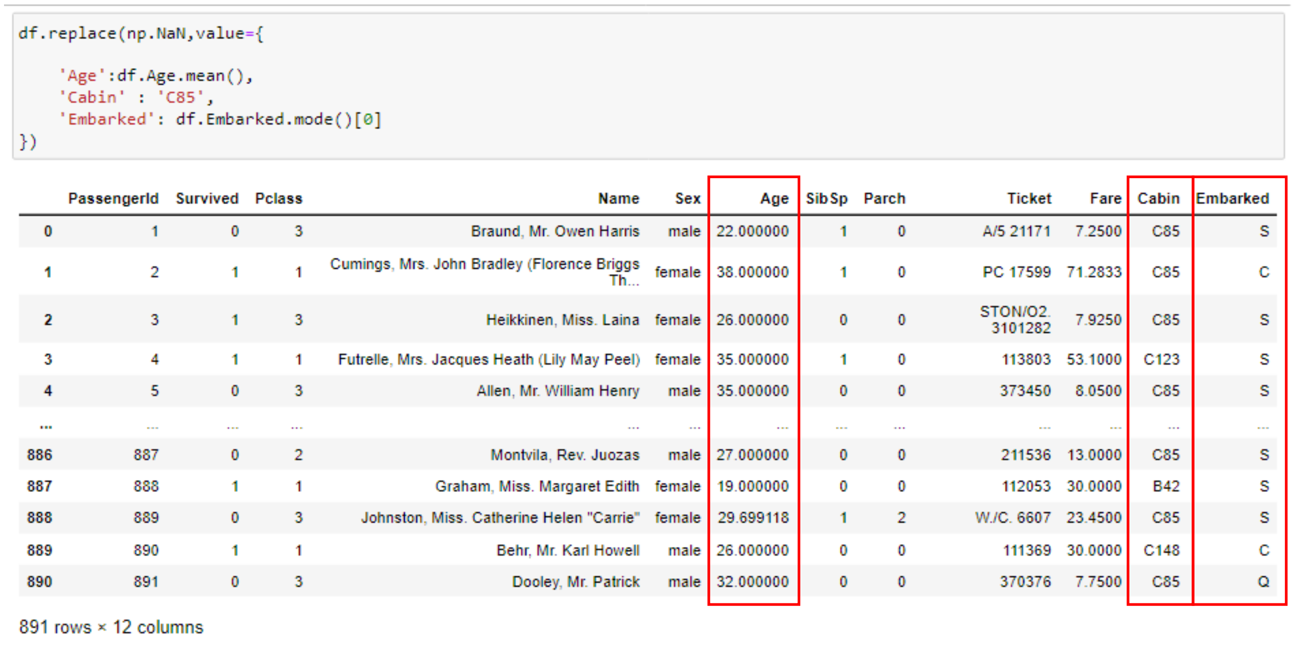
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
16	17	0	3	Rice, Master. Eugene	male	2.0	4	1	382652	29.125	NaN	Q
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.000	NaN	S
18	19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vande...	female	31.0	1	0	345763	18.000	NaN	S
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.225	NaN	C
20	21	0	2	Fynney, Mr. Joseph J	male	35.0	0	0	239865	26.000	NaN	S

```
df.loc[16:20].interpolate()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
16	17	0	3	Rice, Master. Eugene	male	2.0	4	1	382652	29.125	NaN	Q
17	18	1	2	Williams, Mr. Charles Eugene	male	16.5	0	0	244373	13.000	NaN	S
18	19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vande...	female	31.0	1	0	345763	18.000	NaN	S
19	20	1	3	Masselmani, Mrs. Fatima	female	33.0	0	0	2649	7.225	NaN	C
20	21	0	2	Fynney, Mr. Joseph J	male	35.0	0	0	239865	26.000	NaN	S

3.1.2.7 Replace

Replace the NULL values with the values given in ‘*values’* parameter



```
df.replace(np.NaN,value={

    'Age':df.Age.mean(),
    'Cabin' : 'C85',
    'Embarked': df.Embarked.mode()[0]
})
```

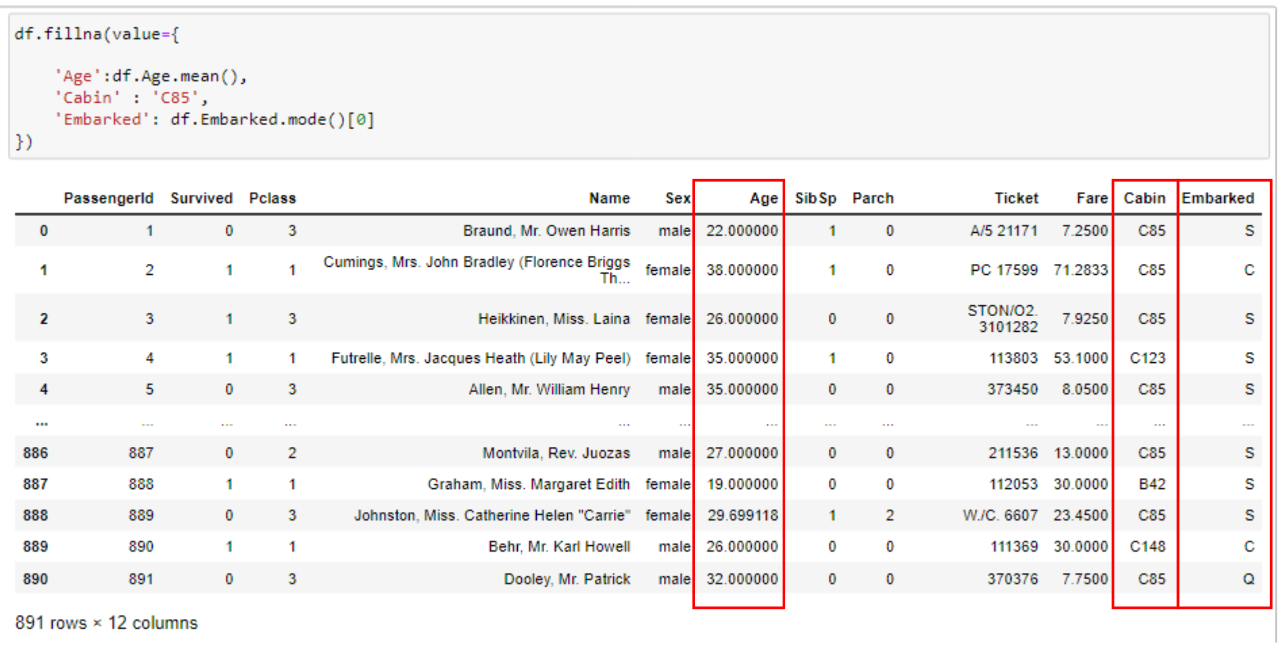
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	C85	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	C85	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450	8.0500	C85	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	211536	13.0000	C85	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./C. 6607	23.4500	C85	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	370376	7.7500	C85	Q

891 rows × 12 columns

3.1.2.8 fillna

Fill the NULL values with the values given in ‘*values’* parameter

.



```
df.fillna(value={

    'Age':df.Age.mean(),
    'Cabin' : 'C85',
    'Embarked': df.Embarked.mode()[0]
})
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	C85	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	C85	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450	8.0500	C85	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	211536	13.0000	C85	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./C. 6607	23.4500	C85	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	370376	7.7500	C85	Q

891 rows × 12 columns

3.1.2 Inconsistent data

Attributes are often formatted differently from system to system. For example, one data set might include a customer's middle initial, while another doesn't. Data elements such as terms and identifiers may also vary.

The common basic data inconsistent problems can be solved by

- Fix capitalization inconsistencies
- Lowercase
- Fix whitespace
- Remove whitespace

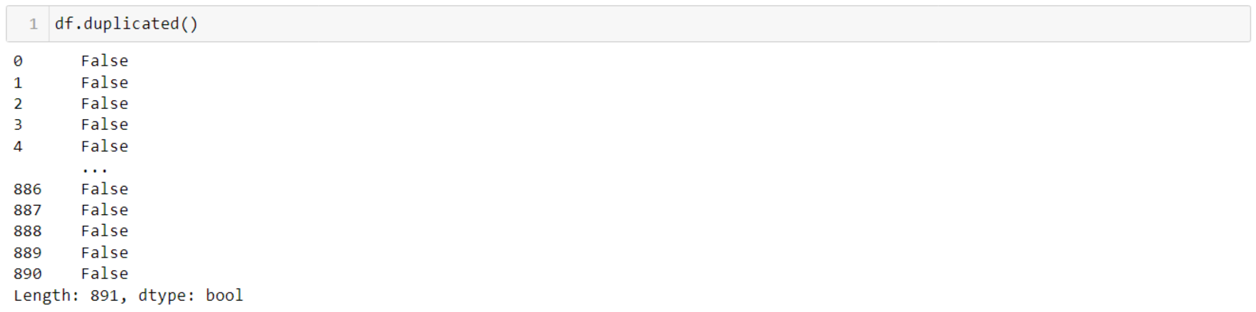
Luckily there are libraries to deal with the inconsistencies, one of them are Fuzzy. This automatically finds the text strings that are very similar to the target string which is very helpful in dealing with the inconsistencies.

3.1.3 Duplicate data

Data cleansing identifies duplicate records in data sets and either removes or merges them through the use of deduplication measures. For example, when data from two systems is combined, duplicate data entries can be reconciled to create single records.

We can simply use pandas method to deal with duplicate values

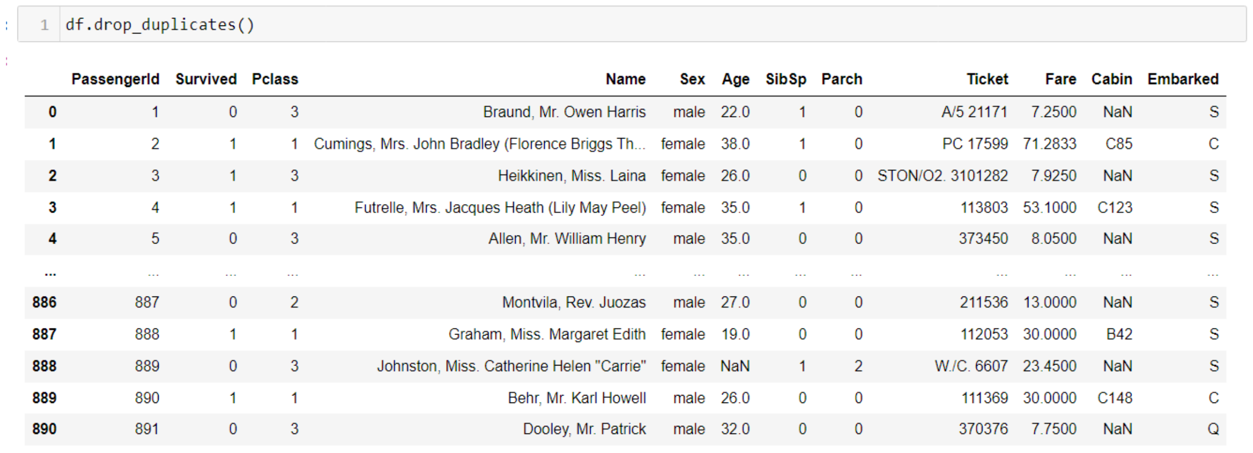
To find duplicate values exists or not



```
df.duplicated()
```

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

To return duplicates removed data



```
df.drop_duplicates()
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

Some other Tools that we can use to deal with the duplicate data are

- Dedupe
- Data Ladder

3.1.4 Irrelevant data

Some data -- outliers or out-of-date entries, for example -- may not be relevant to analytics applications and could skew their results. Data cleansing removes redundant data from data sets, which streamlines data preparation and reduces the required amount of data processing and storage resources.

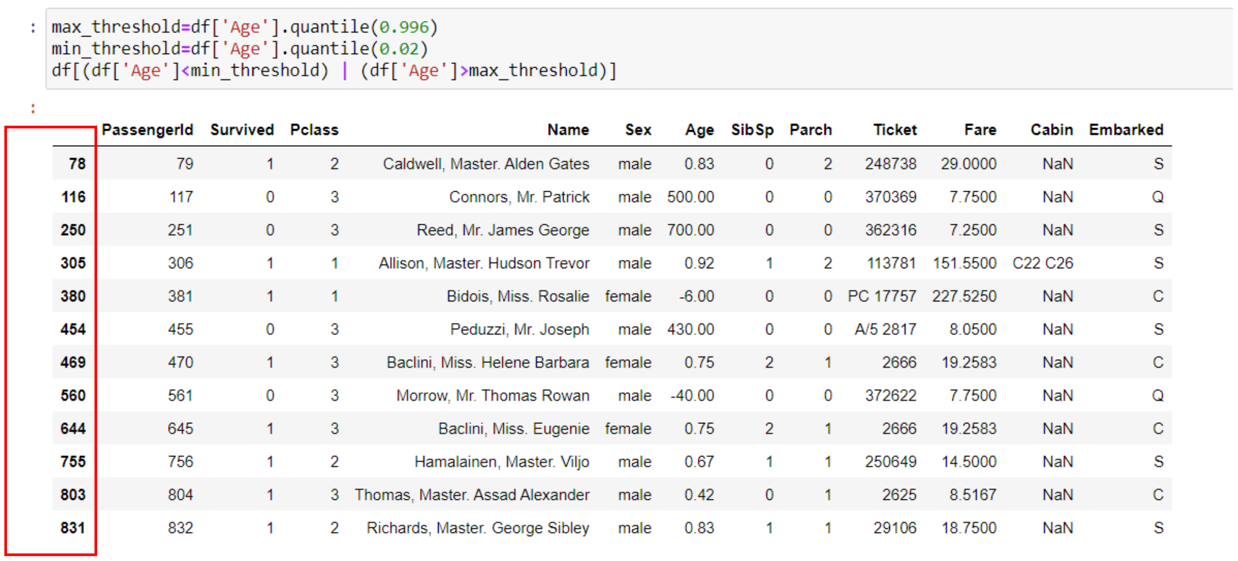
3.1.4.1 Outliers

An outlier is an object that deviates significantly from the rest of the objects. They can be caused by measurement or execution errors. Let's find it for the column 'Age' in different methods

3.1.4.1.1 Using IQR Method (Interquartile range)

Interquartile range or IQR is a quantity that measures the difference between the first and the third quartiles in a given dataset.

- Arrange the data in the increasing order
- Calculate the first and third quantile (q1 and q3 respectively)
- Find the IQR (q3-q1)
- Find lower bound and upper bound

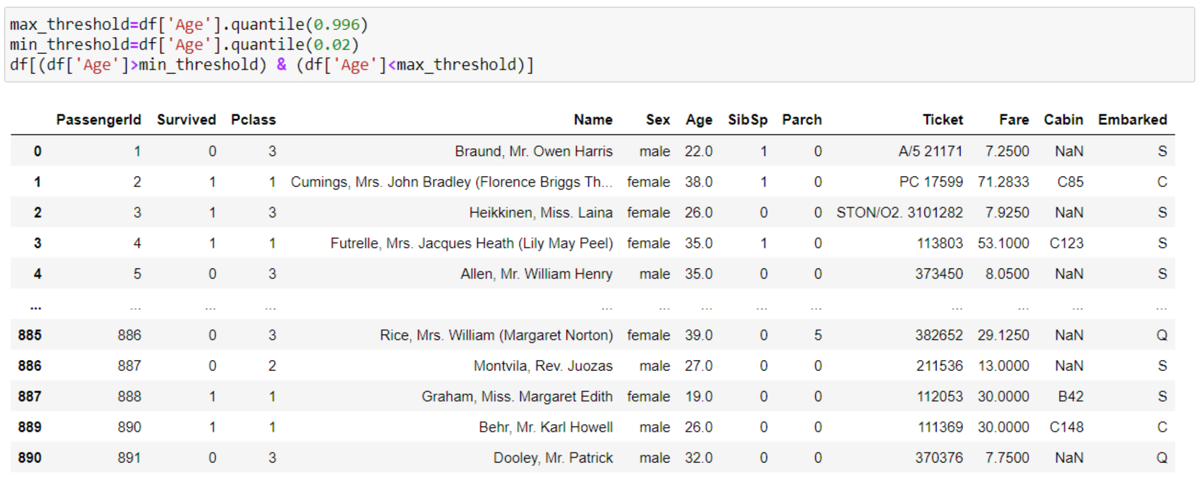


```
: max_threshold=df['Age'].quantile(0.996)
  min_threshold=df['Age'].quantile(0.02)
  df[(df['Age']<min_threshold) | (df['Age']>max_threshold)]
```

:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
78	79	1	2	Caldwell, Master. Alden Gates	male	0.83	0	2	248738	29.0000	NaN	S
116	117	0	3	Connors, Mr. Patrick	male	500.00	0	0	370369	7.7500	NaN	Q
250	251	0	3	Reed, Mr. James George	male	700.00	0	0	362316	7.2500	NaN	S
305	306	1	1	Allison, Master. Hudson Trevor	male	0.92	1	2	113781	151.5500	C22 C26	S
380	381	1	1	Bidois, Miss. Rosalie	female	-6.00	0	0	PC 17757	227.5250	NaN	C
454	455	0	3	Peduzzi, Mr. Joseph	male	430.00	0	0	A/5 2817	8.0500	NaN	S
469	470	1	3	Baclini, Miss. Helene Barbara	female	0.75	2	1	2666	19.2583	NaN	C
560	561	0	3	Morrow, Mr. Thomas Rowan	male	-40.00	0	0	372622	7.7500	NaN	Q
644	645	1	3	Baclini, Miss. Eugenie	female	0.75	2	1	2666	19.2583	NaN	C
755	756	1	2	Hamalainen, Master. Viljo	male	0.67	1	1	250649	14.5000	NaN	S
803	804	1	3	Thomas, Master. Assad Alexander	male	0.42	0	1	2625	8.5167	NaN	C
831	832	1	2	Richards, Master. George Sibley	male	0.83	1	1	29106	18.7500	NaN	S

Removing outliers



```
max_threshold=df['Age'].quantile(0.996)
min_threshold=df['Age'].quantile(0.02)
df[(df['Age']>min_threshold) & (df['Age']<max_threshold)]
```

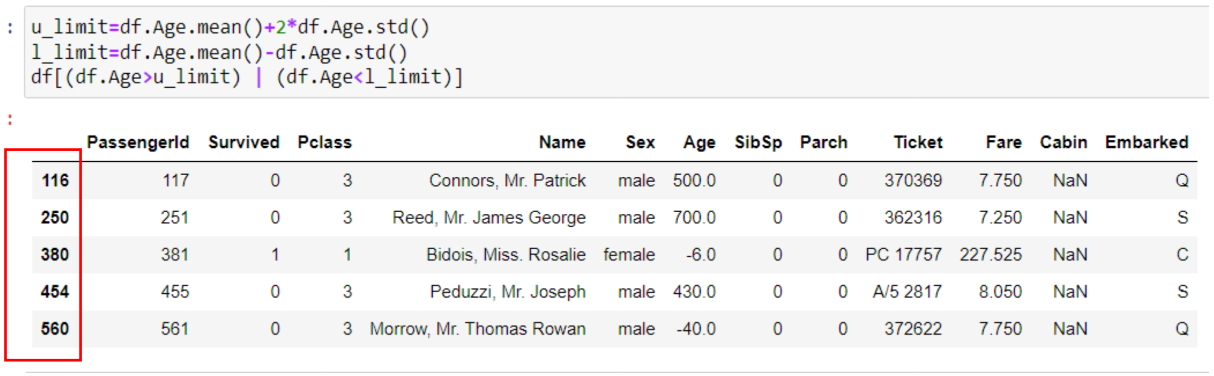
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 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
...	...	...	...	...	...	...	...	...	...	...	...	...
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	NaN	Q
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

3.1.4.1.2 Using Standard Deviation

Finding Outliers

- Arrange the data in increasing order.
- Find the threshold value as below:
- $UB = mean + 2 * SD$
- $LB = mean - 2 * SD$

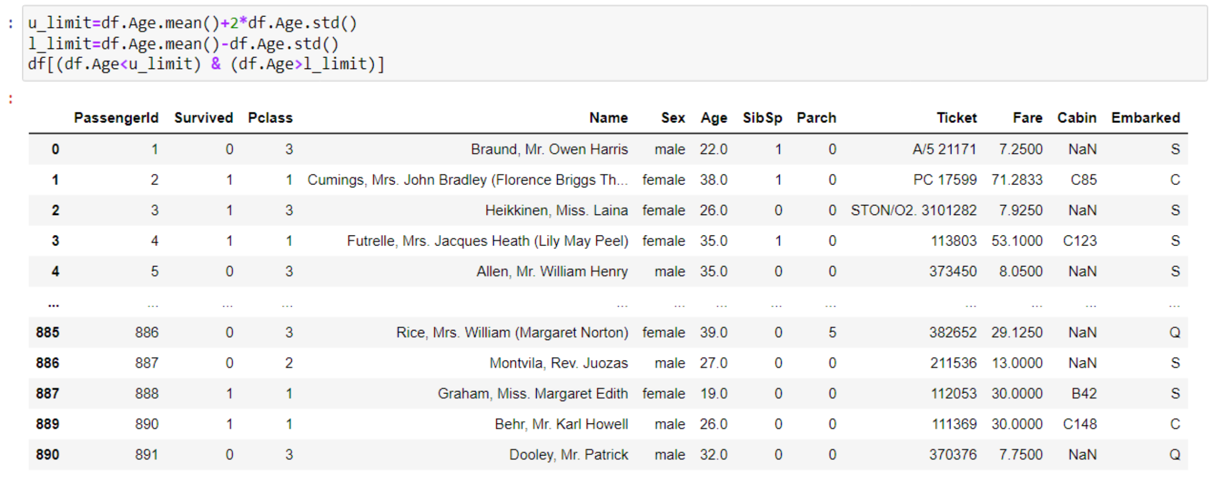
Note: 2 is tunable parameter



```
u_limit=df.Age.mean()+2*df.Age.std()
l_limit=df.Age.mean()-df.Age.std()
df[(df.Age>u_limit) | (df.Age<l_limit)]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
116	117	0	3	Connors, Mr. Patrick	male	500.0	0	0	370369	7.750	NaN	Q
250	251	0	3	Reed, Mr. James George	male	700.0	0	0	362316	7.250	NaN	S
380	381	1	1	Bidois, Miss. Rosalie	female	-6.0	0	0	PC 17757	227.525	NaN	C
454	455	0	3	Peduzzi, Mr. Joseph	male	430.0	0	0	A/5 2817	8.050	NaN	S
560	561	0	3	Morrow, Mr. Thomas Rowan	male	-40.0	0	0	372622	7.750	NaN	Q

Removing Outliers



```
u_limit=df.Age.mean()+2*df.Age.std()
l_limit=df.Age.mean()-df.Age.std()
df[(df.Age<u_limit) & (df.Age>l_limit)]
```

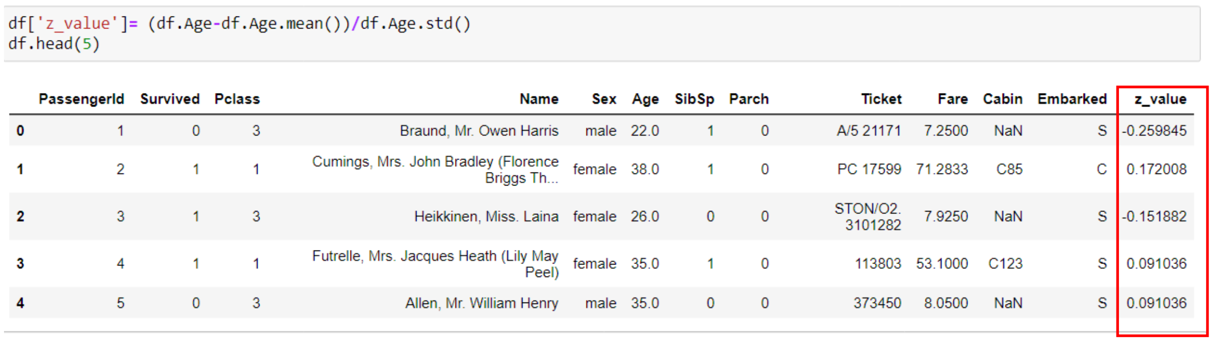
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 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
...	...	...	...	...	...	...	...	...	...	...	...	...
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	NaN	Q
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

3.1.4.1.3 Using Z-value

- Arrange the data in the increasing order
- Calculate the Z-Score using the formula
- $ZScore = \frac{X-mean}{SD}$
- Find the threshold value as below:
- $UB \; = \; Zscore > 3$
- $LB \; = \; Zscore < -3$

Note: 3 is tunable parameter

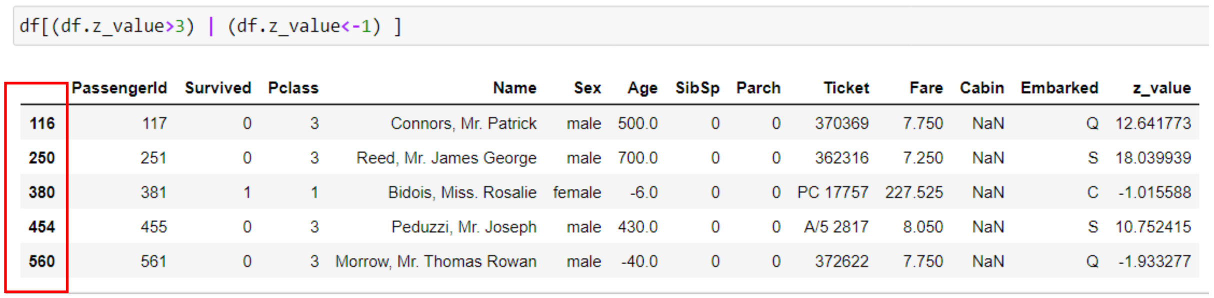
Creating column of z-values



```
df['z_value']= (df.Age-df.Age.mean())/df.Age.std()
df.head(5)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	z_value
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	-0.259845
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	0.172008
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	-0.151882
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	0.091036
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0.091036

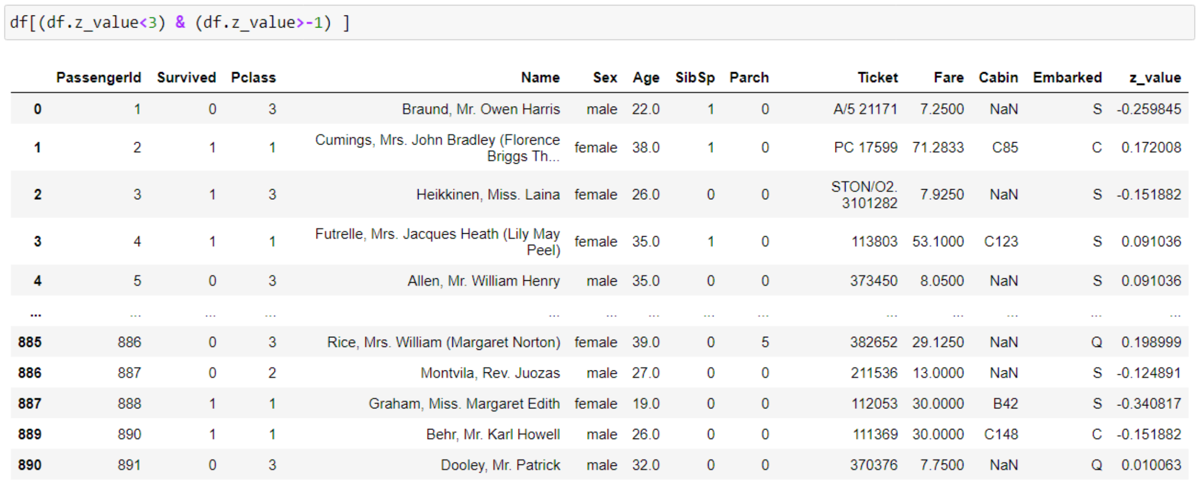
Finding Outliers



```
df[(df.z_value>3) | (df.z_value<-1) ]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	z_value
116	117	0	3	Connors, Mr. Patrick	male	500.0	0	0	370369	7.750	NaN	Q	12.641773
250	251	0	3	Reed, Mr. James George	male	700.0	0	0	362316	7.250	NaN	S	18.039939
380	381	1	1	Bidois, Miss. Rosalie	female	-6.0	0	0	PC 17757	227.525	NaN	C	-1.015588
454	455	0	3	Peduzzi, Mr. Joseph	male	430.0	0	0	A/5 2817	8.050	NaN	S	10.752415
560	561	0	3	Morrow, Mr. Thomas Rowan	male	-40.0	0	0	372622	7.750	NaN	Q	-1.933277

Removing Outliers



```
df[(df.z_value<3) & (df.z_value>-1) ]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	z_value
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	-0.259845
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	0.172008
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	-0.151882
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	0.091036
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0.091036
...	...	...	...	...	...	...	...	...	...	...	...	...	...
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	NaN	Q	0.198999
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S	-0.124891
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S	-0.340817
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C	-0.151882
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q	0.010063

3.1.4.2 Noise

Handling Noise:

Noise refers to random errors or fluctuations in the data that are not representative of the true underlying patterns. It can be introduced during data collection, transmission, or storage.

Techniques to handle noise include:

- Smoothing: Applying techniques like moving averages or kernel smoothing to reduce random fluctuations.
- Aggregation: Combining data points into groups or intervals to reduce the effect of noise.
- Filtering: Using digital signal processing techniques like low-pass filters to remove high-frequency noise.

Tools that help cleanse data are available in a variety of products and platforms, including the following:

- specialized data cleaning tools from vendors such as Data Ladder and WinPure;
- data quality software from vendors such as Datactics, Experian, Innovative Systems, Melissa, Microsoft and Precisely;
- data preparation tools from vendors such as Altair, DataRobot, Tableau, Tibco Software and Trifacta;
- data management platforms from vendors such as Alteryx, Ataccama, IBM, Informatica, SAP, SAS, Syniti and Talend;
- customer and contact data management software from vendors such as Redpoint Global, RingLead, Synthio and Tye;
- tools for cleansing data in Salesforce systems from vendors such as Cloudingo and Plauti; and
- open source tools, such as DataCleaner and OpenRefine

4) Data structuring

At this point, the data needs to be modeled and organized to meet the analytics requirements. For example, data stored in comma-separated values (CSV) files or other file formats has to be converted into tables to make it accessible to BI and analytics tools.

Data Structuring Methods:

- Data Tables: Organizing data in rows and columns, similar to spreadsheets.
- Hierarchical Data Structure: Representing data in a tree-like structure with parent-child relationships.
- Network Data Structure: Representing data as nodes connected by edges in a graph.
- Relational Database Structure: Using tables with defined relationships to organize data efficiently.
- Time Series Data Structure: Storing data indexed by time for time-based analysis.

5) Data transformation and enrichment

In addition to being structured, the data typically must be transformed into a unified and usable format. Data transformation may involve creating new fields or columns that aggregate values from existing ones. Data enrichment further enhances and optimizes data sets as needed, through measures such as augmenting and adding data.

There are many different types of data transformation, depending on what kind of data you have and what you want to do with it. Some common types include:

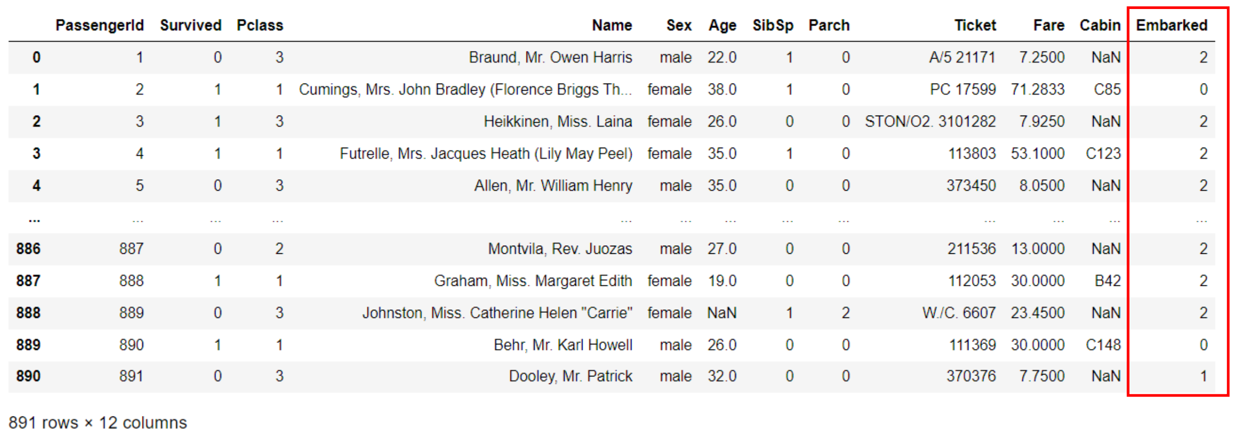
5.1 Encoding

5.1.1 Label Encoding

used to convert categorical columns into numerical

```python
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()

df['Embarked'] = label_encoder.fit_transform(df['Embarked'])
df
```



	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	2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	2
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	2
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	2
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	2
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	2
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	1

891 rows × 12 columns

5.1.2 One hot encoding

One-Hot Encoding is another popular technique for treating categorical variables. It simply creates additional features based on the number of unique values in the categorical feature. Every unique value in the category will be added as a feature.

First, we will create some dummy variable

```python
ports = pd.get_dummies(df.Embarked , prefix='Embarked')
ports.head()
```

	Embarked_C	Embarked_Q	Embarked_S
0	0	0	1
1	1	0	0
2	0	0	1
3	0	0	1
4	0	0	1

Then we apply the One hot encoder

```python
from sklearn.preprocessing import OneHotEncoder
```

```python
prepared_df = pd.concat([df, pd.get_dummies(df['Embarked'],drop_first=True)], axis=1)
prepared_df.drop(['Embarked'], axis=1, inplace=True)
```

```python
prepared_df.head()
```

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

5.2 Feature Scaling

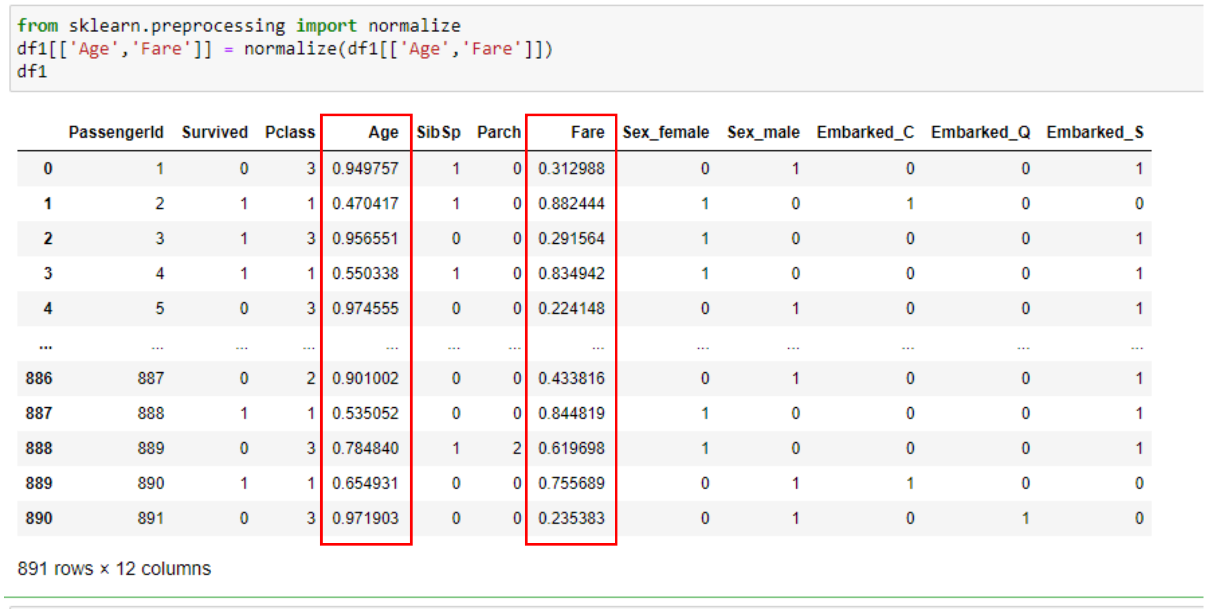
It is the process of transforming the features so that they have a similar scale

5.2.1 Normalization

Normalization scales the variable so that it has a range of values between 0 and 1.

$$X_{norm} = \frac{X_i - X_{min}}{X_{max} - X_{min}}$$

- **x_{norm}**: The ith normalized value in the dataset
- **x_i**: The ith value in the dataset
- **x_{max}**: The minimum value in the dataset
- **x_{min}**: The maximum value in the dataset



```
from sklearn.preprocessing import normalize
df1[['Age','Fare']] = normalize(df1[['Age','Fare']])
df1
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
0	1	0	3	0.949757	1	0	0.312988	0	1	0	0	1
1	2	1	1	0.470417	1	0	0.882444	1	0	1	0	0
2	3	1	3	0.956551	0	0	0.291564	1	0	0	0	1
3	4	1	1	0.550338	1	0	0.834942	1	0	0	0	1
4	5	0	3	0.974555	0	0	0.224148	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	0.901002	0	0	0.433816	0	1	0	0	1
887	888	1	1	0.535052	0	0	0.844819	1	0	0	0	1
888	889	0	3	0.784840	1	2	0.619698	1	0	0	0	1
889	890	1	1	0.654931	0	0	0.755689	0	1	1	0	0
890	891	0	3	0.971903	0	0	0.235383	0	1	0	1	0

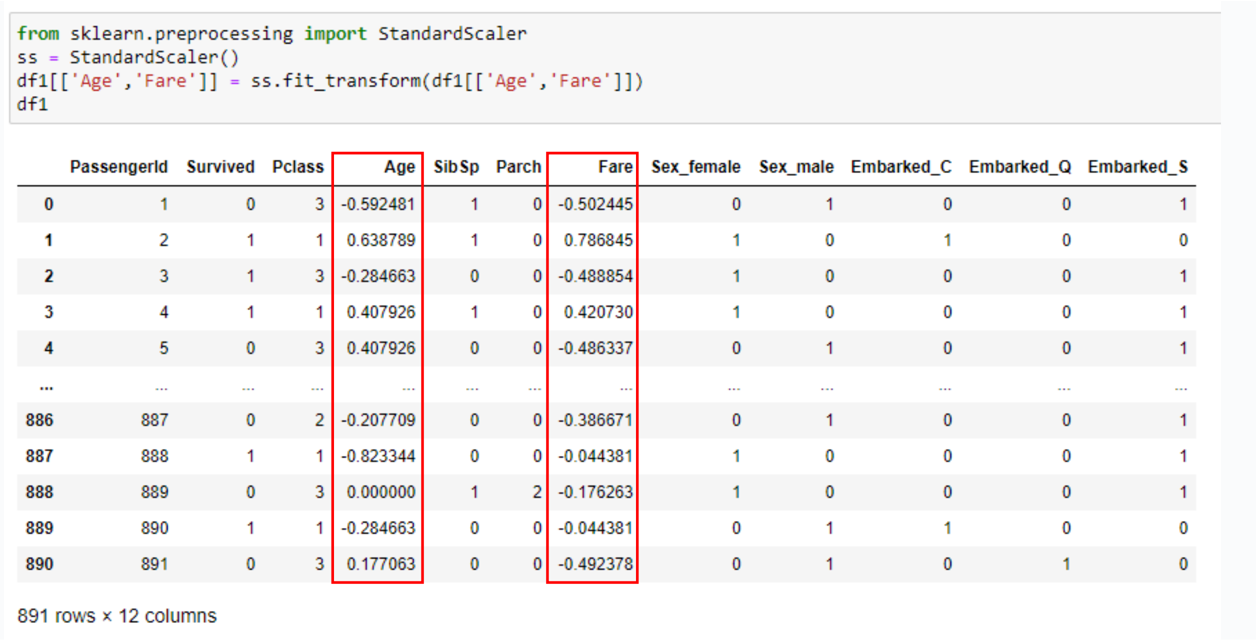
891 rows × 12 columns

5.2.2 Standardization

Standardization scales the variable so that it has zero mean and unit variance.

$$z = \frac{x - \mu}{\sigma}$$

- **x** is the original feature value.
- **μ** is the mean of the feature values.
- **σ** is the standard deviation of the feature values.
- **z** is the standardized value of



```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
df1[['Age','Fare']] = ss.fit_transform(df1[['Age','Fare']])
df1
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
0	1	0	3	-0.592481	1	0	-0.502445	0	1	0	0	1
1	2	1	1	0.638789	1	0	0.786845	1	0	1	0	0
2	3	1	3	-0.284663	0	0	-0.488854	1	0	0	0	1
3	4	1	1	0.407926	1	0	0.420730	1	0	0	0	1
4	5	0	3	0.407926	0	0	-0.486337	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	-0.207709	0	0	-0.386671	0	1	0	0	1
887	888	1	1	-0.823344	0	0	-0.044381	1	0	0	0	1
888	889	0	3	0.000000	1	2	-0.176263	1	0	0	0	1
889	890	1	1	-0.284663	0	0	-0.044381	0	1	1	0	0
890	891	0	3	0.177063	0	0	-0.492378	0	1	0	1	0

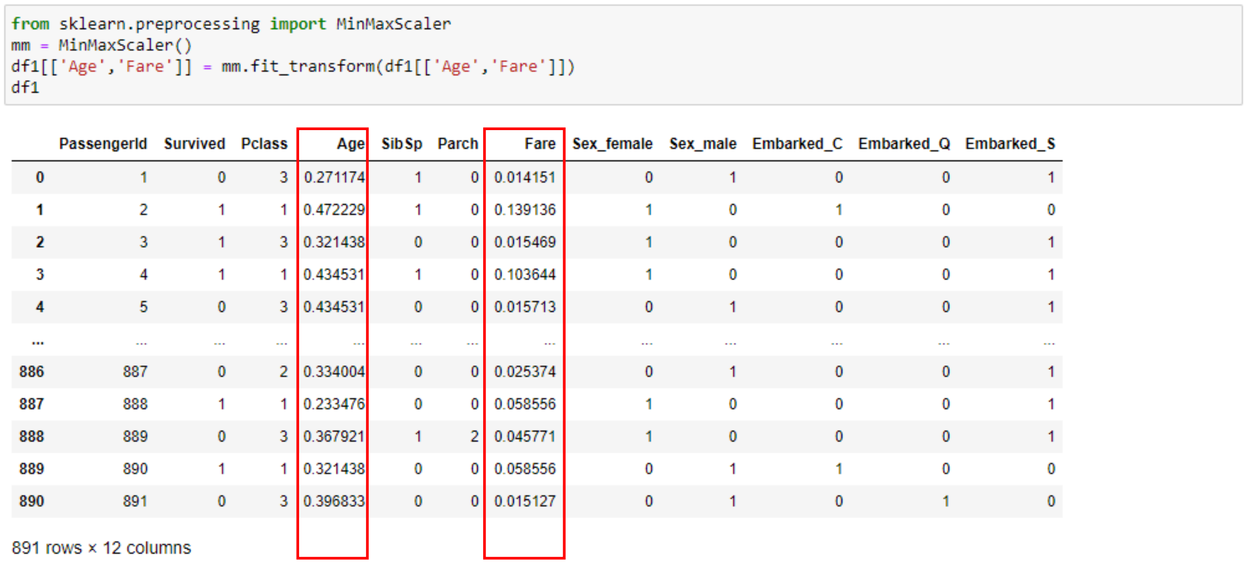
891 rows × 12 columns

5.2.3 Min Max scaling

Rescaling the features to a specific range, such as between 0 and 1, by subtracting the minimum value and dividing by the range.

$$X_{scaled} = \frac{(X - X_{min})}{(X_{\max} - X_{min})}$$

- X is the original feature value.
- X_min is the minimum value of the feature in the data.
- X_max is the maximum value of the feature in the data.
- X_scaled is the scaled/normalized value of the feature.



```
from sklearn.preprocessing import MinMaxScaler
mm = MinMaxScaler()
df1[['Age','Fare']] = mm.fit_transform(df1[['Age','Fare']])
df1
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
0	1	0	3	0.271174	1	0	0.014151	0	1	0	0	1
1	2	1	1	0.472229	1	0	0.139136	1	0	1	0	0
2	3	1	3	0.321438	0	0	0.015469	1	0	0	0	1
3	4	1	1	0.434531	1	0	0.103644	1	0	0	0	1
4	5	0	3	0.434531	0	0	0.015713	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	0.334004	0	0	0.025374	0	1	0	0	1
887	888	1	1	0.233476	0	0	0.058556	1	0	0	0	1
888	889	0	3	0.367921	1	2	0.045771	1	0	0	0	1
889	890	1	1	0.321438	0	0	0.058556	0	1	1	0	0
890	891	0	3	0.396833	0	0	0.015127	0	1	0	1	0

891 rows × 12 columns

5.3 Dimensionality reduction

is the process of reducing the number of features (or dimensions) in a dataset while retaining as much information as possible. It can be done by

5.3.1 Feature extraction

Feature Extraction is the process of creating new features from existing ones

5.3.1.1 PCA

It is used to reduce the dimensionality of a dataset while preserving the most important patterns or relationships between the variables without any prior knowledge of the target variables.

```python
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca_out = pca.fit_transform(df2)
pca_out = pd.DataFrame(pca_out,
                       columns=['feature{}'.
                       format(i+1)
                        for i in range(2)])
pca_out
```

```
       feature1    feature2
0    445.062022  -23.831859
1    443.899433   40.222677
2    443.060237  -23.159503
3    441.945564   22.031915
4    441.059758  -23.045630
...         ...         ...
886 -440.950042  -20.325702
887 -441.993146   -3.314120
888 -442.976037   -9.876632
889 -443.993259   -3.319091
890 -444.936510  -25.593893

891 rows × 2 columns
```

We can see here earlier we has so many columns, now reduced to 2.

5.3.1.2 SVD

Similar to PCS, SVD can be thought of as a projection method where data with m-columns (features) is projected into a subspace with m or fewer columns, whilst retaining the essence of the original data.

```python
from sklearn.decomposition import TruncatedSVD
svd = TruncatedSVD(n_components=2)
svd_out = svd.fit_transform(df2)
svd_out = pd.DataFrame(svd_out,
                       columns=['feature{}'.
                       format(i+1)
                        for i in range(2)])
svd_out
```

```
       feature1    feature2
0      1.413899    7.183485
1      5.943252   71.072868
2      3.447422    7.747826
3      6.936015   52.805596
4      5.451144    7.756100
...         ...         ...
886  886.363905  -36.054680
887  888.298288  -19.128725
888  888.943862  -25.717298
889  890.294839  -19.240002
890  890.070322  -41.520339

891 rows × 2 columns
```

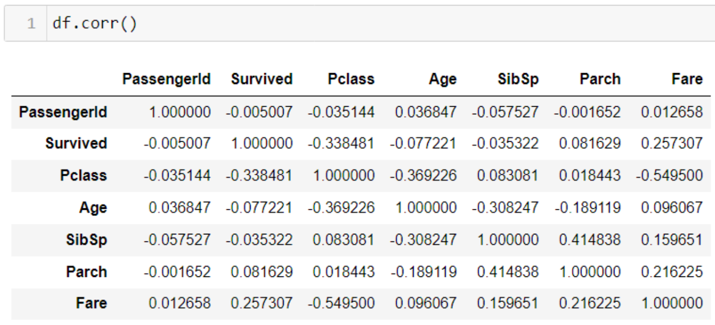
5.3.2 Feature selection

Feature Selection is the process of selecting a subset of relevant features from the dataset to be used in a machine-learning model.

5.3.2.1 Correlation

Correlation explains how one or more variables are related to each other. These variables can be input data features which have been used to forecast our target variable.

It can be found out by



```
df.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

5.3.2.2 VIF

Variance Inflation Factor (VIF) is used to detect the presence of multicollinearity. Variance inflation factors (VIF) measure how much the variance of the estimated regression coefficients is inflated as compared to when the predictor variables are not linearly related.

It is obtained by regressing each independent variable, say X on the remaining independent variables (say Y and Z) and checking how much of it (of X) is explained by these variables.

$$VIF = \frac{1}{1 - R^2}$$

Hence, From the formula, it is clear that higher the VIF, higher the R2 which means the variable X is collinear with Y and Z variables. If all the variables are completely orthogonal, R^2 will be 0 resulting in VIF of 1.

```
X = df[['Sex', 'Age', 'Fare','Survived','Pclass']]
```

```
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns

# calculating VIF for each feature
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(len(X.columns))]
vif_data
```

	feature	VIF
0	Sex	2.224014
1	Age	3.797733
2	Fare	1.615223
3	Survived	2.320361
4	Pclass	3.279303

5.3.2.3 Dependent & independent

Splitting our dataset into dependent(Y) and independent(X) sets for training our model

```
#Independent Variable(Features)
X = df[['Age','Sex','Fare','Pclass','SibSp','Parch']]
X
```

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

891 rows × 6 columns

```
# Dependent Variable(Label)
y = df[['Survived']]
y
```

	Survived
0	0
1	1
2	1
3	1
4	0
...	...
886	0
887	1
888	0
889	1
890	0

891 rows × 1 columns

5.4 Feature engineering

Feature Engineering is the process of creating new features or transforming existing features to improve the performance of a machine-learning model

5.4.1 Polynomial Features

Polynomial engineering is a technique used in feature engineering to capture nonlinear relationships between features and the target variable in machine learning models. It involves creating higher-order polynomial features by raising the existing numerical features to various powers (e.g., square, cube) and combining them through multiplication.

The role of polynomial engineering in feature engineering can be summarized as follows:
Modeling Nonlinear Relationships: Linear models, such as linear regression, can only capture linear relationships between features and the target variable. However, many real-world relationships are not linear, and this is where polynomial engineering becomes useful. By introducing polynomial features, the model gains the ability to approximate more complex, nonlinear patterns in the data.

When applying polynomial engineering, **it's crucial to carefully select the degree of the polynomial** (the highest power used in creating the features) and to evaluate its impact on model performance. In some cases, a quadratic relationship (degree = 2) may be sufficient, while in others, higher degrees may be necessary to adequately capture the complexity of the data. Experimentation and cross-validation are essential to find the appropriate balance between model complexity and generalization.

5.4.2 Interaction

The role of interaction in feature engineering, specifically feature interaction, is to **capture and represent the combined effect or synergy between different features in a machine learning model**. When two or more features interact, their joint effect on the target variable is not simply the sum of their individual effects. Instead, their combination creates a new relationship that can be crucial for accurately predicting the target variable.

There are various ways to incorporate feature interaction in feature engineering:

- **Product Features:** Creating new features by taking the product of two or more relevant features.
- **Grouping or Binning Features**: Combining categorical or continuous features to represent joint categories or intervals.

- **Cross-Features:** Generating new features by combining different categorical variables (e.g., creating a new feature that represents the combination of "gender" and "occupation").
- **Embeddings:** In natural language processing and recommender systems, embeddings can be used to capture interactions between words or items.

5.4.3 Domain specific transformation

Domain-specific transformation in feature engineering refers to the process of applying data transformations and feature engineering techniques that are specific to the particular domain or problem at hand. Different domains have unique characteristics and requirements, and domain-specific transformations aim to exploit this domain knowledge to create more informative and relevant features for machine learning models.

Examples of domain-specific transformations in various domains include:

- **Finance:**
  Logarithmic transformation of financial data to stabilize variance and normalize distributions.
- **Healthcare:**
  Extracting relevant features from medical images, such as the number of tumors or the presence of certain structures. Transforming time series data from patient records, such as calculating moving averages or detecting critical events.
- **Natural Language Processing (NLP):**
  Tokenization and stemming to break text into individual words and reduce inflected words to their base form. Building word embeddings or word vectors to represent words as dense numerical vectors.
- **Retail and Marketing:**
  Creating customer-specific features, such as customer lifetime value or purchase frequency, to predict customer behavior.
  Time-based features, such as day of the week or holiday indicators, to capture seasonal effects in sales data.
- **Environmental Science:**
  Transforming weather data, such as calculating averages or aggregating hourly readings into daily values.
  Incorporating geographic features, such as latitude and longitude, to capture spatial relationships.

6) Data validation and publishing.

In this last step, automated routines are run against the data to validate its consistency, completeness and accuracy. The prepared data is then stored in a data warehouse, a data lake or another repository and either used directly by whoever prepared it or made available for other users to access.

The most common types of data validation include the following:

- **Data type validation** is common and confirms that the data in each field, column, list, range or file matches a specified data type and format.
- **Constraint validation** checks to see if a given data field input fits a specified requirement within certain ranges. For example, it verifies that a data field has a minimum or maximum number of characters.
- **Structured validation** ensures that data is compliant with a specified data format, structure or schema.
- **Consistency validation** makes sure data styles are consistent. For example, it confirms that all values are listed to two decimal points.
- **Code validation** is similar to a consistency check and confirms that codes used for different data inputs are correct. For example, it checks a country code or North American Industry Classification System (NAICS) codes.

How to perform data validation

Among the most basic and common ways that data is used is within a spreadsheet program such as Microsoft Excel or Google Sheets. In both Excel and Sheets, the data validation process is a straightforward, integrated feature. Excel and Sheets both have a menu item listed as *Data* > *Data Validation*. By selecting the *Data Validation* menu, a user can choose the specific data type or constraint validation required for a given file or data range.

To explore more

Data Validation in Machine Learning is imperative, not optional (analyticsvidhya.com)

Data Validation — Overview, Types, How To Perform | Built In