Titanic ship case study

Import necessary libraries

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

1. Download the dataset:

```
# titanic.csv dataset downloaded and placed in the working directory
```

2. Load the dataset.

```
data = pd.read_csv("titanic.csv")
```

```
data.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	a
0	0	3	male	22.0	1	0	7.2500	S	Third	man	
1	1	1	female	38.0	1	0	71.2833	C	First	woman	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	
3	1	1	female	35.0	1	0	53.1000	S	First	woman	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   survived     891 non-null    int64  
 1   pclass       891 non-null    int64  
 2   sex          891 non-null    object 
 3   age          714 non-null    float64
```

```
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    object
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    object
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), float64(2), int64(4), object(7)
memory usage: 92.4+ KB
```

3. Perform Below Visualizations.

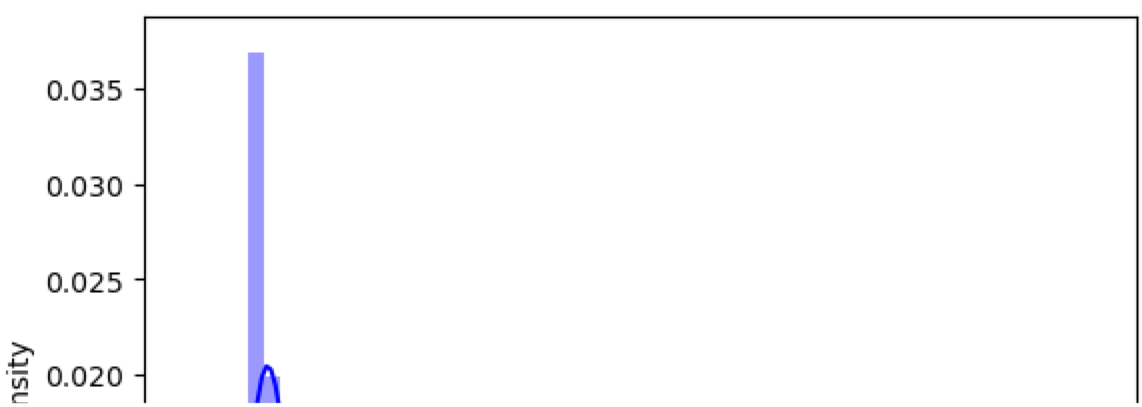
    - Univariate Analysis
    - Bi - Variate Analysis
    - Multi - Variate Analysis

Univariate Analysis

Distribution plot

```
sns.distplot(data['fare'], color = 'b')
```

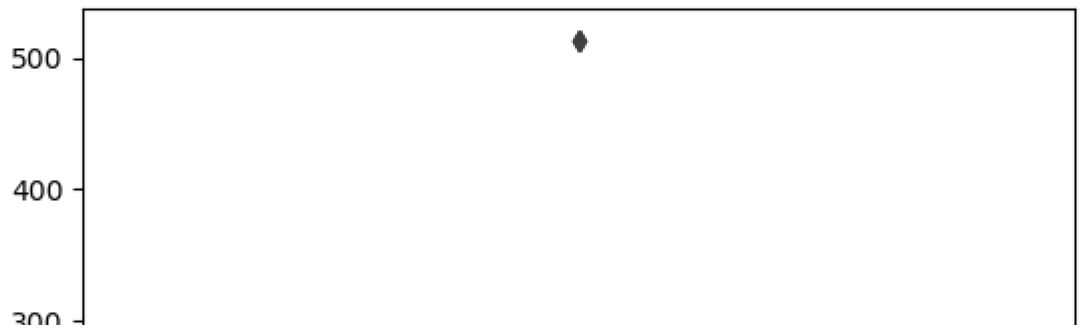
```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/1361863019.py:1

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['fare'], color = 'b')
<Axes: xlabel='fare', ylabel='Density'>
```



Box plot

```
sns.boxplot(data['fare'])
```

```
<Axes: >
```



Scatter plot

```
sns.scatterplot(data['fare'])
```

```
<Axes: ylabel='fare'>
```

Joint plot

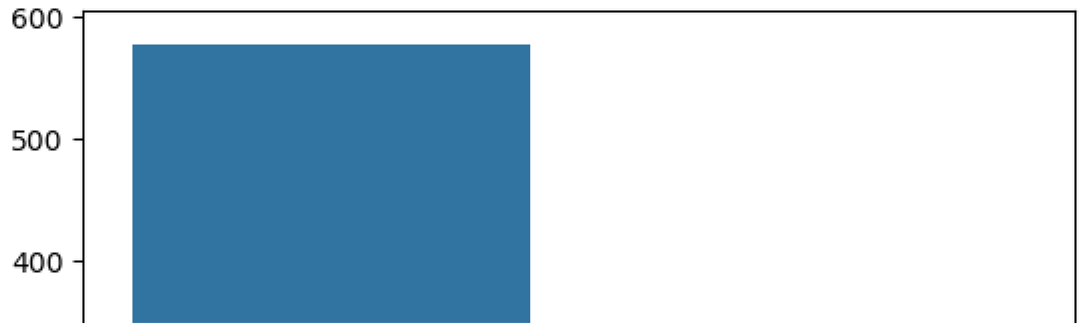
```
sns.jointplot(data['fare'])
```

```
<seaborn.axisgrid.JointGrid at 0x11cce1540>
```

Bar plot

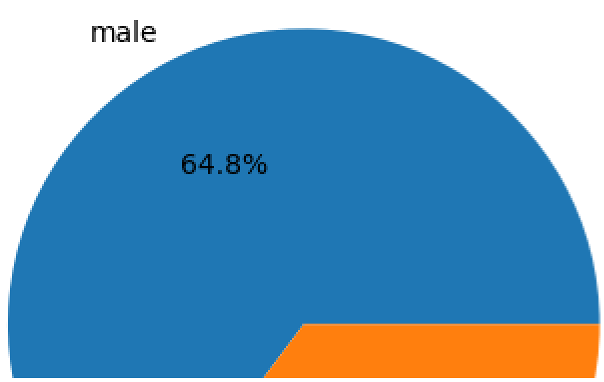
```
x = data.sex.value_counts()
sns.barplot(x=x.index, y=x.values)
```

```
<Axes: >
```



Pie plot

```
x = data['sex'].value_counts()
plt.pie(x.values,
        labels=x.index,
        autopct='%1.1f%%')
plt.show()
```



Bivariate analysis

Bar plot

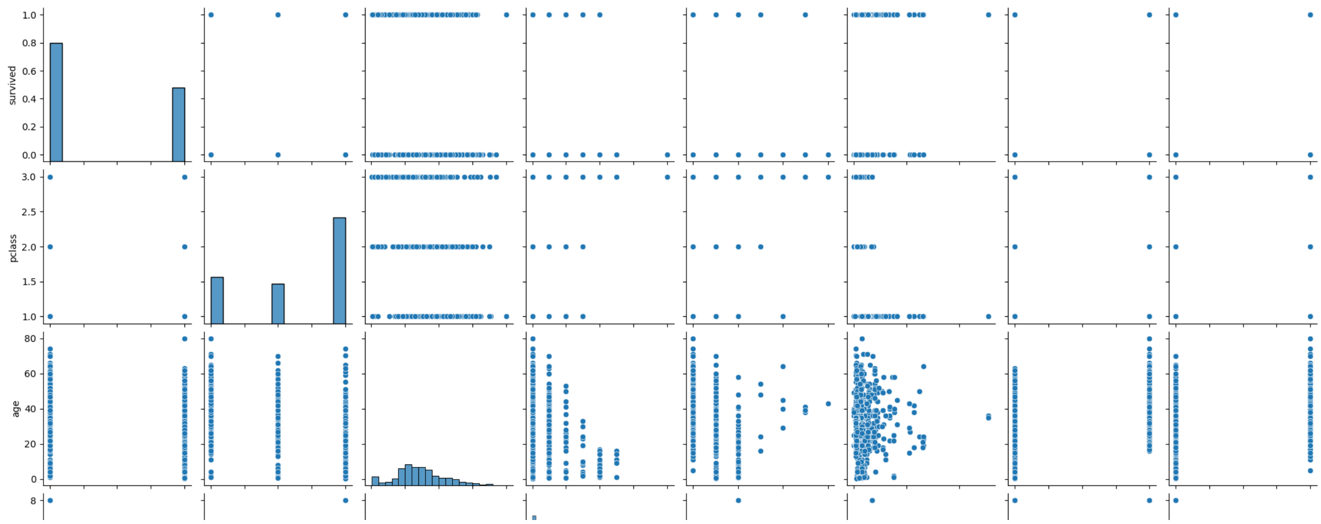
```
sns.barplot(x=data.sex, y=data.fare)
```

```
<Axes: xlabel='sex', ylabel='fare'>
```

Pair plot

```
sns.pairplot(data)
```

```
<__array_function__ internals>:180: RuntimeWarning: Converting input from bool
<__array_function__ internals>:180: RuntimeWarning: Converting input from bool
<seaborn.axisgrid.PairGrid at 0x11d19a440>
```



Multivariate Analysis

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

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/1119197534.py:1
```

4. Perform descriptive statistics on the dataset.

Measure of central tendency - Mean, Median and Mode

```
data.mean()
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/531903386.py:1:
  data.mean()
survived       0.383838
pclass         2.308642
age           29.699118
sibsp          0.523008
parch          0.381594
fare          32.204208
adult_male     0.602694
alone          0.602694
dtype: float64
```

```
data.median()
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/4184645713.py:1
  data.median()
survived       0.0000
pclass         3.0000
age           28.0000
sibsp          0.0000
parch          0.0000
fare          14.4542
adult_male     1.0000
alone          1.0000
dtype: float64
```

```
data.mode()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_m
0	0	3	male	24.0	0	0	8.05	S	Third	man	

Measure of variability

Kurtosis

```
data.kurt()
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/2907027414.py:1
  data.kurt()
survived      -1.775005
pclass        -1.280015
age            0.178274
sibsp         17.880420
parch          9.778125
fare          33.398141
adult_male    -1.827345
alone         -1.827345
dtype: float64
```

Range

```
data.max()
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/2904433368.py:1
  data.max()
survived             1
pclass               3
sex               male
age               80.0
sibsp                8
parch                6
fare          512.3292
class            Third
who              woman
adult_male        True
alive              yes
alone             True
dtype: object
```

```
data.min()
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/927168777.py:1:
  data.min()
survived           0
pclass             1
sex           female
age             0.42
sibsp              0
parch              0
fare             0.0
class          First
who            child
adult_male     False
alive             no
alone          False
dtype: object
```

```
Range = data.max()[['age', 'fare']] - data.min()[['age', 'fare']]
print(Range)
```

```
age        79.58
fare    512.3292
dtype: object
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/1058199015.py:1
  Range = data.max()[['age', 'fare']] - data.min()[['age', 'fare']]
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/1058199015.py:1
  Range = data.max()[['age', 'fare']] - data.min()[['age', 'fare']]
```

Skewness

```
data.skew()
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/1188251951.py:1
  data.skew()
survived      0.478523
pclass       -0.630548
age           0.389108
sibsp         3.695352
parch         2.749117
fare          4.787317
adult_male   -0.420431
alone        -0.420431
dtype: float64
```

Interquartile range

```
quantiles = data[['age', 'fare']].quantile(q=[0.75, 0.25])
quantiles
```

	age	fare
0.75	38.000	31.0000
0.25	20.125	7.9104

```
#Q3
quantiles.iloc[0]
```

```
age     38.0
fare    31.0
Name: 0.75, dtype: float64
```

```
#Q1
quantiles.iloc[1]
```

```
age     20.1250
```

```
fare     7.9104
Name: 0.25, dtype: float64
```

```
IQR = quantiles.iloc[0]-quantiles.iloc[1]
IQR
```

```
age     17.8750
fare    23.0896
dtype: float64
```

Upper extreme

Q3 + 1.5*IQR

```
quantiles.iloc[0] + (1.5*IQR)
```

```
age     64.8125
fare    65.6344
dtype: float64
```

Lower extreme

Q1 - 1.5*IQR

```
quantiles.iloc[1] - (1.5*IQR)
```

```
age     -6.6875
fare   -26.7240
dtype: float64
```

Standard deviation

```
data.std()
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/2723740006.py:1
  data.std()
survived       0.486592
pclass         0.836071
age           14.526497
sibsp          1.102743
parch          0.806057
fare          49.693429
adult_male     0.489615
alone          0.489615
dtype: float64
```

Variance

```
data.var()
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/445316826.py:1:
  data.var()
survived        0.236772
pclass          0.699015
age           211.019125
sibsp           1.216043
parch           0.649728
fare         2469.436846
adult_male      0.239723
alone           0.239723
dtype: float64
```

```
data.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

5. Handle the Missing values.

```
data.isnull().sum()
```

```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
```

```
alive           0
alone           0
dtype: int64
```

Handling missing value

```
data['age'].fillna(data['age'].mean(), inplace=True)
```

```
data['embarked'].fillna(data['embarked'].mode()[0], inplace=True)
```

```
data['deck'].fillna(data['deck'].mode()[0], inplace=True)
```

```
data['embark_town'].fillna(data['embark_town'].mode()[0], inplace=True)
```

```
data.isnull().sum()
```

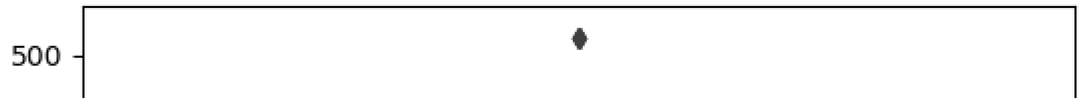
```
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
deck           0
embark_town    0
alive          0
alone          0
dtype: int64
```

6. Find the outliers and replace the outliers

Removing outliers

```
sns.boxplot(data.fare)
```

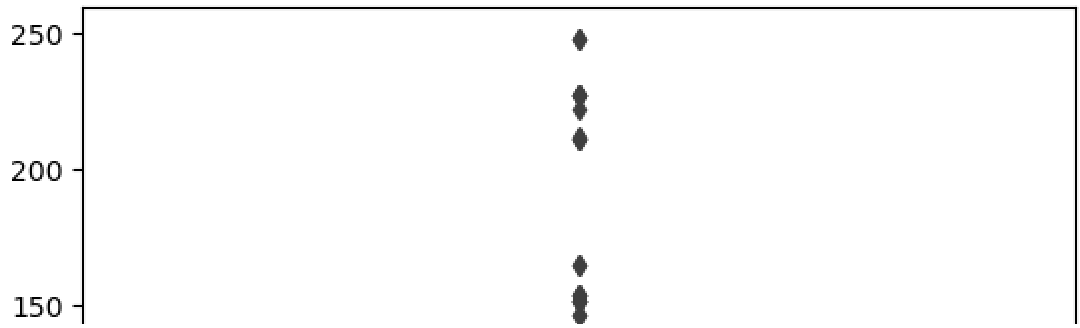
```
<Axes: >
```




```
quant99 = data.fare.quantile(0.99)
data = data[data.fare<quant99]
```

```
sns.boxplot(data.fare)
```

```
<Axes: >
```



7. Check for Categorical columns and perform encoding.

Encoding techniques

Label encoding

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
data.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	a
0	0	3	male	22.0	1	0	7.2500	S	Third	man	
1	1	1	female	38.0	1	0	71.2833	C	First	woman	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	
3	1	1	female	35.0	1	0	53.1000	S	First	woman	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 882 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     882 non-null    int64
 1   pclass       882 non-null    int64
 2   sex          882 non-null    object
 3   age          882 non-null    float64
 4   sibsp        882 non-null    int64
 5   parch        882 non-null    int64
 6   fare         882 non-null    float64
 7   embarked     882 non-null    object
 8   class        882 non-null    object
 9   who          882 non-null    object
 10  adult_male   882 non-null    bool
 11  deck         882 non-null    object
 12  embark_town  882 non-null    object
 13  alive        882 non-null    object
 14  alone        882 non-null    bool
dtypes: bool(2), float64(2), int64(4), object(7)
memory usage: 130.5+ KB
```

```
columns = ['sex', 'embarked', 'class', 'who', 'deck', 'alive']
for col in columns:
 data[col] = le.fit_transform(data[col])
```

```
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/462314644.py:3:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
  data[col] = le.fit_transform(data[col])
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/462314644.py:3:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
  data[col] = le.fit_transform(data[col])
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/462314644.py:3:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
  data[col] = le.fit_transform(data[col])
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/462314644.py:3:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
  data[col] = le.fit_transform(data[col])
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/462314644.py:3:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
  data[col] = le.fit_transform(data[col])
/var/folders/03/k1p5_v6d69bg7b999gdktlgw0000gn/T/ipykernel_3411/462314644.py:3:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
  data[col] = le.fit_transform(data[col])
```

```
data.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_
0	0	3	1	22.0	1	0	7.2500	2	2	1	
1	1	1	0	38.0	1	0	71.2833	0	0	2	
2	1	3	0	26.0	0	0	7.9250	2	2	2	
3	1	1	0	35.0	1	0	53.1000	2	0	2	
4	0	3	1	35.0	0	0	8.0500	2	2	1	

One Hot Encoding

```
data = pd.get_dummies(data, columns=['embark_town'])
```

```
data
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who
0	0	3	1	22.000000	1	0	7.2500	2	2	1
1	1	1	0	38.000000	1	0	71.2833	0	0	2

8. Split the data into dependent and independent variables.

Dependent variable

```
y = data.loc[:, 'alive':'alive']
y
```

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

882 rows × 1 columns

Independent variablea

```
X = data.drop(columns=['alive'], axis=1)
X
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who
0	0	3	1	22.000000	1	0	7.2500	2	2	1
1	1	1	0	38.000000	1	0	71.2833	0	0	2
2	1	3	0	26.000000	0	0	7.9250	2	2	2
3	1	1	0	35.000000	1	0	53.1000	2	0	2
4	0	3	1	35.000000	0	0	8.0500	2	2	1
...	...	...	...	...	...	...	...	...	...	...
886	0	2	1	27.000000	0	0	13.0000	2	1	1
887	1	1	0	19.000000	0	0	30.0000	2	0	2
888	0	3	0	29.699118	1	2	23.4500	2	2	2

9. Scale the independent variables

Scaling

StandardScaler --> mean=0 std=1

MinMaxScaler --> scale between 0 to 1

```
from sklearn.preprocessing import MinMaxScaler
scale = MinMaxScaler()
```

```
name = X.columns
X_scaled = scale.fit_transform(X)
```

```
X_scaled
```

```
array([[0., 1., 1., ..., 0., 0., 1.],
       [1., 0., 0., ..., 1., 0., 0.],
       [1., 1., 0., ..., 0., 0., 1.],
       ...,
       [0., 1., 0., ..., 0., 0., 1.],
       [1., 0., 1., ..., 1., 0., 0.],
       [0., 1., 1., ..., 0., 1., 0.]])
```

```
X = pd.DataFrame(X_scaled, columns=name)
X
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	wh
0	0.0	1.0	1.0	0.271174	0.125	0.000000	0.029290	1.0	1.0	0.
1	1.0	0.0	0.0	0.472229	0.125	0.000000	0.287989	0.0	0.0	1.
2	1.0	1.0	0.0	0.321438	0.000	0.000000	0.032018	1.0	1.0	1.
3	1.0	0.0	0.0	0.434531	0.125	0.000000	0.214527	1.0	0.0	1.
4	0.0	1.0	1.0	0.434531	0.000	0.000000	0.032523	1.0	1.0	0.
...	...	...	...	...	...	...	...	...	...	.
877	0.0	0.5	1.0	0.334004	0.000	0.000000	0.052521	1.0	0.5	0.
878	1.0	0.0	0.0	0.233476	0.000	0.000000	0.121202	1.0	0.0	1.
879	0.0	1.0	0.0	0.367921	0.125	0.333333	0.094740	1.0	1.0	1.
880	1.0	0.0	1.0	0.321438	0.000	0.000000	0.121202	0.0	0.0	0.

10. Split the data into training and testing

Train-Test Split

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
X_train
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	wh
222	1.0	0.0	1.0	0.472229	0.125	0.000000	0.363606	1.0	0.0	0.
200	0.0	1.0	1.0	0.421965	0.000	0.000000	0.026243	1.0	1.0	0.

```
y_train
```

	alive
224	1
202	0
164	0
582	0
850	0
...	...
844	0
194	1
635	1
565	0
691	1

705 rows × 1 columns

```
X_test
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	wh
150	0.0	1.0	1.0	0.692134	0.000	0.000000	0.032523	1.0	1.0	0.
406	0.0	1.0	1.0	0.367921	0.000	0.000000	0.027708	0.5	1.0	0.

```
y_test
```

	alive
152	0
411	0
519	0
103	0
590	0
...	...
367	1
372	0
267	1
324	0
472	1

177 rows × 1 columns