```
# Стандартный импорт для работы с данными
import numpy as np
import pandas as pd

# Импорт модулей для стандартизации и нормализации
from sklearn.preprocessing import MinMaxScaler,StandardScaler

# Подружаем файл при условии, что он в области видимости
#в рабочей папке
ds = pd.read_csv('dataset.csv')
ds.head()
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age
SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0
1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
1
2                             Heikkinen, Miss. Laina  female  26.0
0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
1
4                           Allen, Mr. William Henry    male  35.0
0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

```
# смотрим  имена столбцов
ds.columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age',
'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

```
#Если нужно оставить несколько столбцов, вносим их далее через запятую
df = ds[['Survived', 'Sex', 'Age', 'SibSp',
       'Parch','Fare']]
df
```

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

[891 rows x 6 columns]
```

```
# Удаляем строки с проущенными значениями
df = df.dropna()
df
```

```
     Survived     Sex   Age  SibSp  Parch     Fare
0           0    male  22.0      1      0   7.2500
1           1  female  38.0      1      0  71.2833
2           1  female  26.0      0      0   7.9250
3           1  female  35.0      1      0  53.1000
4           0    male  35.0      0      0   8.0500
..        ...     ...   ...    ...    ...      ...
885         0  female  39.0      0      5  29.1250
886         0    male  27.0      0      0  13.0000
887         1  female  19.0      0      0  30.0000
889         1    male  26.0      0      0  30.0000
890         0    male  32.0      0      0   7.7500

[714 rows x 6 columns]
```

```
from sklearn.preprocessing import LabelEncoder

# Используем LabelEncoder для кодировки строкового столбца
ncoder = LabelEncoder()
df['Sex']= ncoder.fit_transform(df['Sex'])

df.head()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
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/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
```

```
  This is separate from the ipykernel package so we can avoid doing
imports until
```

```
   Survived  Sex   Age  SibSp  Parch     Fare
0         0    1  22.0      1      0   7.2500
1         1    0  38.0      1      0  71.2833
2         1    0  26.0      0      0   7.9250
3         1    0  35.0      1      0  53.1000
4         0    1  35.0      0      0   8.0500
```

```
#Сохраним названия столбцов перед нормализацией, чтобы восстановить в
дальнейшем
columns = df.columns
columns
```

```
Index(['Survived', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare'],
dtype='object')
```

```
# minmas scaler
mms= MinMaxScaler()
# обучение нормализатора
df = mms.fit_transform(np.array(df))
```

```
# выыведем 1 строку для оценки результата
df[:1]
```

```
array([[0.        , 1.        , 0.27117366, 0.2       , 0.        ,
        0.01415106]])
```

```
# преобразуем обратно в DataFrame
df = pd.DataFrame(data = df, columns = columns)
df.head()
```

```
   Survived  Sex       Age  SibSp  Parch      Fare
0       0.0  1.0  0.271174    0.2    0.0  0.014151
1       1.0  0.0  0.472229    0.2    0.0  0.139136
2       1.0  0.0  0.321438    0.0    0.0  0.015469
3       1.0  0.0  0.434531    0.2    0.0  0.103644
4       0.0  1.0  0.434531    0.0    0.0  0.015713
```

```
# описательная статистика
df.describe()
```

```
         Survived         Sex         Age       SibSp       Parch
Fare
count  714.000000  714.000000  714.000000  714.000000  714.000000
714.000000
mean     0.406162    0.634454    0.367921    0.102521    0.071895
0.067719
std      0.491460    0.481921    0.182540    0.185957    0.142215
0.103291
min      0.000000    0.000000    0.000000    0.000000    0.000000
```

```
0.000000
25%      0.000000    0.000000    0.247612    0.000000    0.000000
0.015713
50%      0.000000    1.000000    0.346569    0.000000    0.000000
0.030726
75%      1.000000    1.000000    0.472229    0.200000    0.166667
0.065144
max      1.000000    1.000000    1.000000    1.000000    1.000000
1.000000
```

```
#Сумма элементов
df.values.sum()
```

```
1178.580104011869
```

```
# Округляем до ближайшего целого
round(df.values.sum())
```

```
1179
```

```
# Стандартизация
std = StandardScaler()
df_s = std.fit_transform(np.array(df))
```

```
df_s[:1]
```

```
array([[-0.82702011,  0.75905134, -0.53037664,  0.52457013, -
0.50589515,
        -0.51897787]])
```