What is Feature, Scaling?

Scale: 0-8.5 0-170 0-41

Male Hight In Feet	Female Hight In CM	Life Span In Year
8.0	150	30
8.5	165	40
7.9	170	36
8.2	140	41

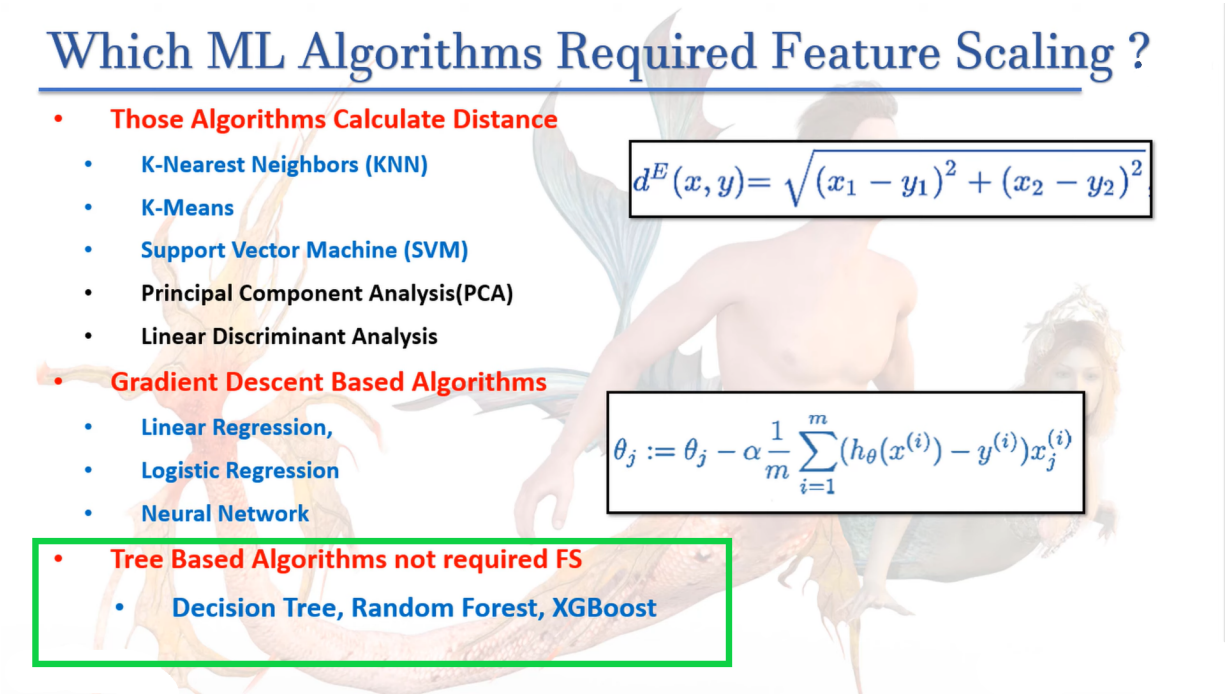
What is Feature Scaling?

- **Feature Scaling is a method to scale numeric features in the same scale or range *(like:-1 to 1, 0 to 1)*.**
- **This last step involved in Data Preprocessing and before ML model training.**
- **It is also called as data normalization.**
- **We apply Feature Scaling on independent variables.**
- **We fit feature scaling with train data and transform on train and test data.**

Why Feature Scaling?

- **The scale of raw features is different according to its units.**
- **Machine Learning algorithms can't understand features units, understand only numbers.**
- ***Ex: If hight 140cm and 8.2feet***
- ***ML Algorithms understand numbers then 140 > 8.2***

Male Hight In Feet	Female Hight In CM	Life Span In Year
8.0	150	30
8.5	165	40
7.9	170	36
8.2	140	41



In [2]:
```
1 import pandas as pd
2 import seaborn as sns
```

In [3]:
```
1 df=sns.load_dataset("titanic")
```

In [4]:
```
1 df1=df[['survived','pclass','age','parch']]
```

In [5]:
```
1 df1
```

Out[5]:

	survived	pclass	age	parch
0	0	3	22.0	0
1	1	1	38.0	0
2	1	3	26.0	0
3	1	1	35.0	0
4	0	3	35.0	0
...	...	...	...	...
886	0	2	27.0	0
887	1	1	19.0	0
888	0	3	NaN	2
889	1	1	26.0	0
890	0	3	32.0	0

891 rows × 4 columns

In [6]:
```
1 df1.isnull().sum()
```

Out[6]:
```
survived      0
pclass        0
age         177
parch         0
dtype: int64
```

In [7]:
```
1 df2=df1.fillna(df1.mean())
```

In [8]:
```
1 df2
```

Out[8]:

	survived	pclass	age	parch
0	0	3	22.000000	0
1	1	1	38.000000	0
2	1	3	26.000000	0
3	1	1	35.000000	0
4	0	3	35.000000	0
...	...	...	...	...
886	0	2	27.000000	0
887	1	1	19.000000	0
888	0	3	29.699118	2
889	1	1	26.000000	0
890	0	3	32.000000	0

891 rows × 4 columns

In [9]:
```
1 x=df2.drop(['survived'],axis=1)
```

In [10]:
```
1 x
```

Out[10]:

	pclass	age	parch
0	3	22.000000	0
1	1	38.000000	0
2	3	26.000000	0
3	1	35.000000	0
4	3	35.000000	0
...	...	...	...
886	2	27.000000	0
887	1	19.000000	0
888	3	29.699118	2
889	1	26.000000	0
890	3	32.000000	0

891 rows × 3 columns

In [11]:
```
1 y=df2.survived
```

In [12]:
```
1 y
```

Out[12]:
```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: survived, Length: 891, dtype: int64
```

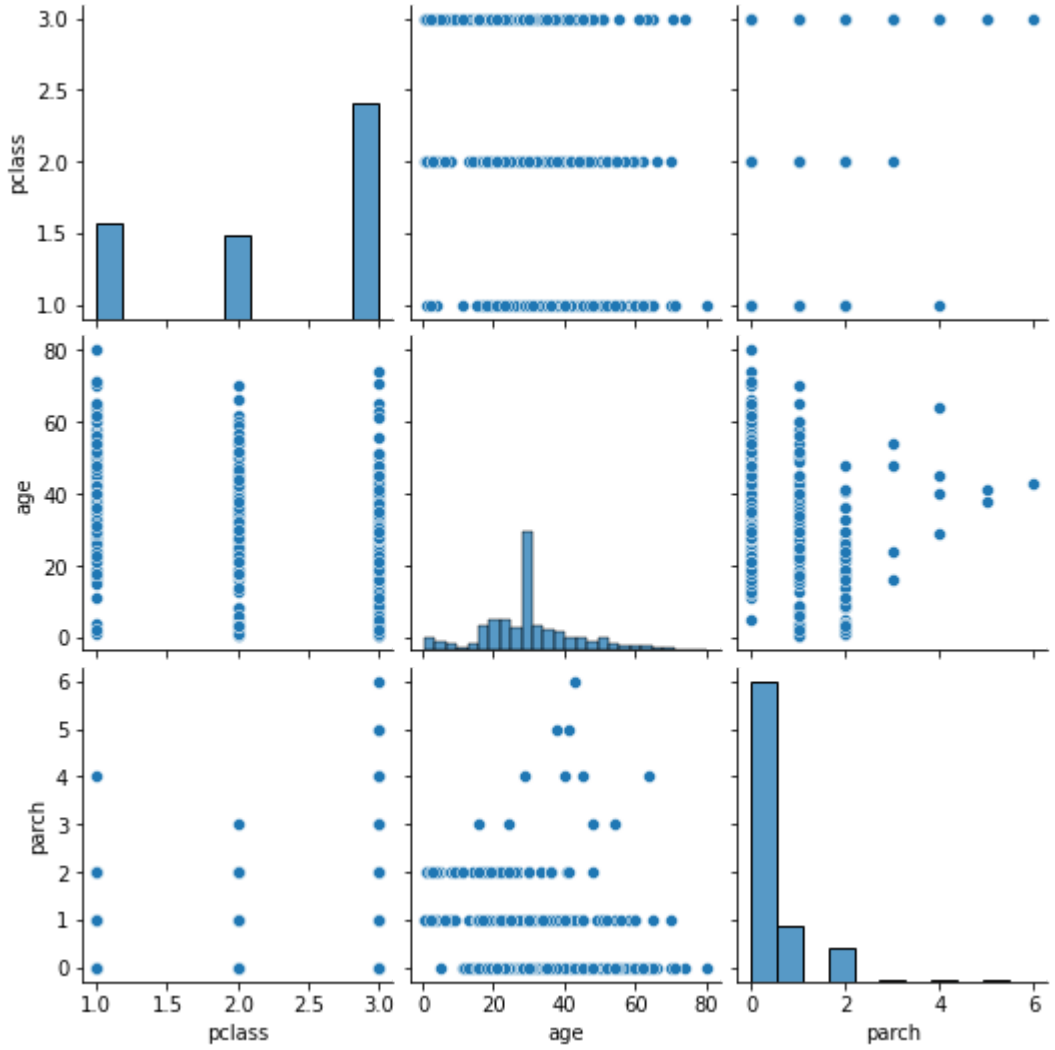
Data Splitting

In [13]:
```
1 from sklearn.model_selection import train_test_split
```

In [14]:
```
1 xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=.2 ,random_state=1)
```

In [15]:
```
1 sns.pairplot(xtrain)
```

Out[15]: <seaborn.axisgrid.PairGrid at 0x239749f0130>



Difference between

Standardization vs Normalization?

- There is no any thumb rule to use Standardization or Normalization for special ML algo.
- But mostly Standardization use for clustering analyses, Principal Component Analysis(PCA).
- Normalization prefer for Image processing because pixel intensity between 0 to 255, neural network algorithm require data in scale 0-1, K-Nearest Neighbors.

Standardization

What is Standardization?

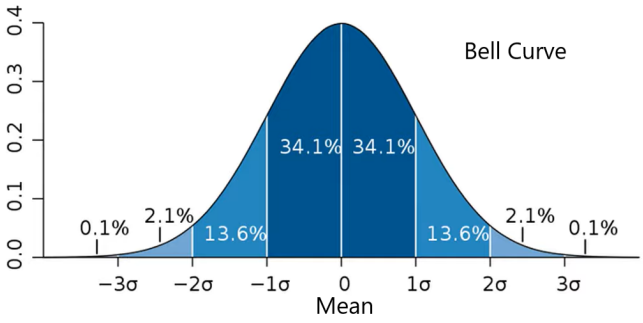
- Standardisation rescale the feature such as mean(μ) = 0 and standard deviation (σ) = 1.
- The result of standardisation is Z called as Z-score normalization.
- If data follow normal distribution (gaussian distribution).
- If the original distribution is normal, then the standardised distribution will be normal.
- If the original distribution is skewed, then the standardised distribution of the variable will also be skewed.

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

What is Standardization?

- **Standardisation rescale the feature such mean(μ) = 0 and standard deviation (σ) = 1.**

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



StandardScaler()

StandardScaler(*, copy=True, with_mean=True, with_std=True)

In [16]:
```
1 from sklearn.preprocessing import StandardScaler
```

In [17]:
```
1 sc=StandardScaler()
2 sc.fit(xtrain)
```

Out[17]:
```
StandardScaler()
```

In [18]:
```
1 sc.mean_  # mean of xtrain
```

Out[18]:
```
array([ 2.3005618 , 30.07175975,  0.37078652])
```

In [19]:
```
1 sc.scale_  # standard deviation of xtrain
```

Out[19]:
```
array([ 0.83605529, 13.2589318 ,  0.77542337])
```

In [20]:
```
1 xtrain.describe()
```

Out[20]:

	pclass	age	parch
count	712.000000	712.000000	712.000000
mean	2.300562	30.071760	0.370787
std	0.836643	13.268253	0.775968
min	1.000000	0.420000	0.000000
25%	2.000000	22.000000	0.000000
50%	3.000000	29.699118	0.000000
75%	3.000000	36.000000	0.000000
max	3.000000	80.000000	6.000000

In [21]:
```
1 xtrain_sc=sc.transform(xtrain)
2 xtest_sc=sc.transform(xtest)
3 #xtrain_sc=sc.transform(xtrain).round(2)
4 #xtest_sc=sc.transform(xtest).round(2)
```

In [22]:
```
1 xtrain_sc
```

Out[22]:
```
array([[ 0.83659324, -0.02810499, -0.478173  ],
       [-1.55559305, -0.00541218, -0.478173  ],
       [-0.35949991,  0.29627125, -0.478173  ],
       ...,
       [-0.35949991, -0.6841999 , -0.478173  ],
       [ 0.83659324, -0.02810499, -0.478173  ],
       [ 0.83659324, -0.6841999 , -0.478173  ]])
```

In [23]:
```
1 xtest_sc
```

```
[-1.55559305e+00,  1.42758410e+00, -4.78172997e-01],
[ 8.36593238e-01, -6.08779038e-01, -4.78172997e-01],

[-3.59499905e-01,  1.35216325e+00, -4.78172997e-01],
[ 8.36593238e-01, -3.82516467e-01, -4.78172997e-01],
[-3.59499905e-01, -5.33358181e-01, -4.78172997e-01],
[ 8.36593238e-01,  2.20850389e-01, -4.78172997e-01],
[-3.59499905e-01, -8.08330388e-02, -4.78172997e-01],
[-1.55559305e+00,  7.00086753e-02, -4.78172997e-01],
[-3.59499905e-01,  4.47112961e-01, -4.78172997e-01],
[ 8.36593238e-01,  7.00086753e-02, -4.78172997e-01],
[-3.59499905e-01, -1.56253896e-01, -4.78172997e-01],
[-1.55559305e+00,  1.65384667e+00,  8.11445085e-01],
[ 8.36593238e-01, -2.81049867e-02, -4.78172997e-01],
[-3.59499905e-01, -9.10462467e-01, -4.78172997e-01],
[-1.55559305e+00,  7.00086753e-02,  2.10106317e+00],
[-1.55559305e+00, -8.35041610e-01, -4.78172997e-01],
[-3.59499905e-01, -1.56253896e-01, -4.78172997e-01],
[ 8.36593238e-01, -6.08779038e-01, -4.78172997e-01],
[ 8.36593238e-01, -2.81049867e-02,  2.10106317e+00],
```

In [24]:
```
1 xtrain_sc_df=pd.DataFrame(xtrain_sc,columns=['pclass','age','parch'])
```

In [25]:
```
1 xtest_sc_df=pd.DataFrame(xtest_sc,columns=['pclass','age','parch'])
```

In [26]:
```
1 xtrain_sc_df.describe().round(2)
```

Out[26]:

	pclass	age	parch
count	712.00	712.00	712.00
mean	0.00	0.00	0.00
std	1.00	1.00	1.00
min	-1.56	-2.24	-0.48
25%	-0.36	-0.61	-0.48
50%	0.84	-0.03	-0.48
75%	0.84	0.45	-0.48
max	0.84	3.77	7.26

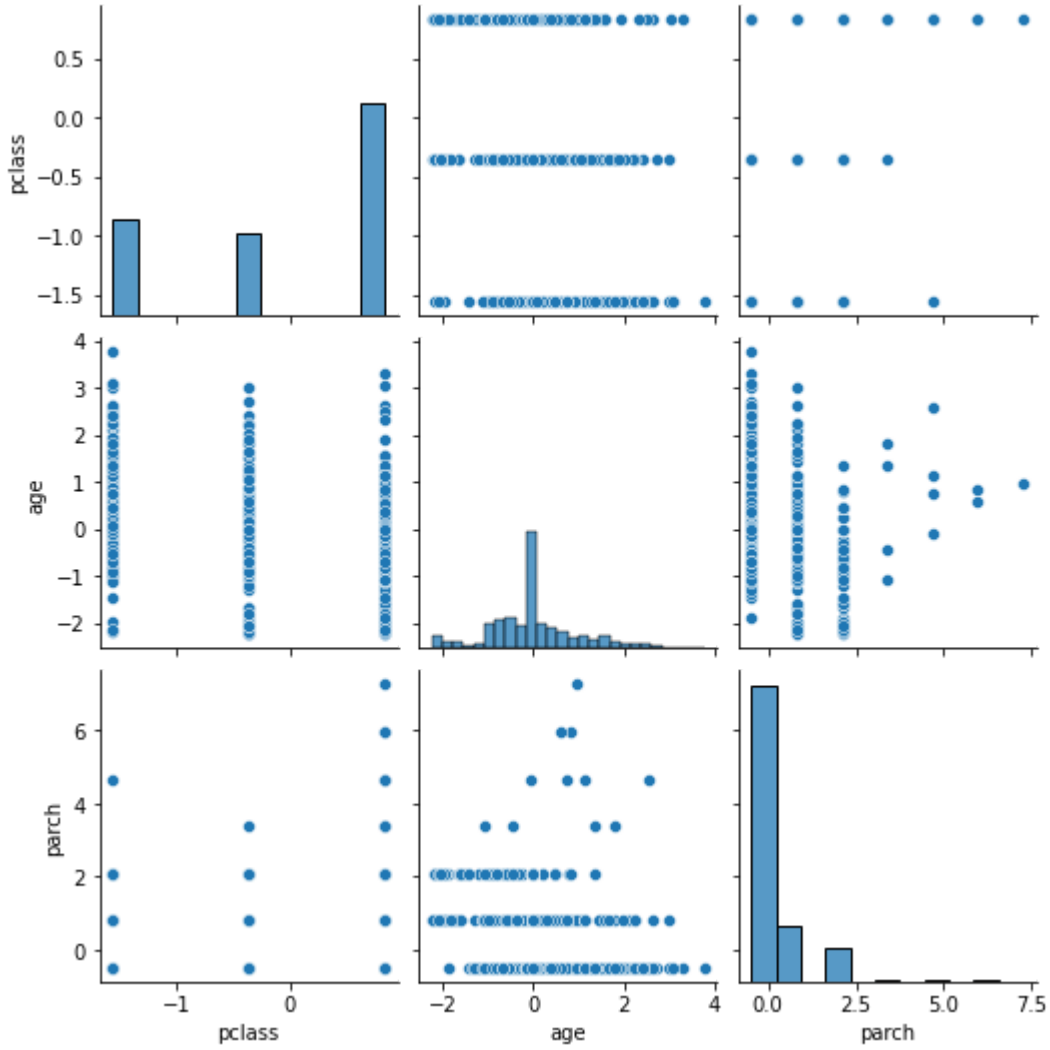
In [27]:
```
1 xtest_sc_df.describe().round(2)
```

Out[27]:

	pclass	age	parch
count	179.00	179.00	179.00
mean	0.05	-0.14	0.07
std	1.00	0.89	1.18
min	-1.56	-2.21	-0.48
25%	-0.36	-0.61	-0.48
50%	0.84	-0.08	-0.48
75%	0.84	0.15	-0.48
max	0.84	2.33	5.97

In [28]:
```
1 sns.pairplot(xtrain_sc_df)
```

Out[28]: <seaborn.axisgrid.PairGrid at 0x239748c61f0>

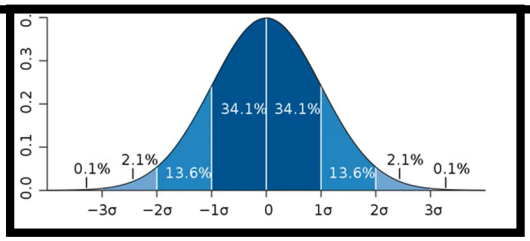


Normalization

What is Normalization?

- **Normalization rescale the feature in fixed range between 0 to 1.**
- **Normalization also called as Min-Max Scaling.**
- **If data doesn't follow normal distribution (Gaussian distribution).** When Data is not support or not found as a BELL curve we generally use NORMALIZATION.

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



MinMaxScaler()

MinMaxScaler(feature_range=(0, 1), *, copy=True, clip=False)

In [29]:
```
1 from sklearn.preprocessing import MinMaxScaler
```

In [30]:
```
1 mms=MinMaxScaler()
```

In [31]:
```
1 mms.fit(xtrain) #Learned min max o xtrain
```

Out[31]:
```
MinMaxScaler()
```

In [32]:
```
1 xtrain_mms=mms.transform(xtrain)
2 xtest_mms=mms.transform(xtest)
```

In [33]:
```
1 xtrain_mms_df=pd.DataFrame(xtrain_mms,columns=['pclass','age','parch'])
```

In [34]:
```
1 xtest_mms_df=pd.DataFrame(xtest_mms,columns=['pclass','age','parch'])
```

In [35]:
```
1 xtrain_mms_df.describe()
```

Out[35]:

	pclass	age	parch
count	712.000000	712.000000	712.000000
mean	0.650281	0.372603	0.061798
std	0.418322	0.166728	0.129328
min	0.000000	0.000000	0.000000
25%	0.500000	0.271174	0.000000
50%	1.000000	0.367921	0.000000
75%	1.000000	0.447097	0.000000
max	1.000000	1.000000	1.000000

In [36]:
```
1 xtrain.describe()
```

Out[36]:

	pclass	age	parch
count	712.000000	712.000000	712.000000
mean	2.300562	30.071760	0.370787
std	0.836643	13.268253	0.775968
min	1.000000	0.420000	0.000000
25%	2.000000	22.000000	0.000000
50%	3.000000	29.699118	0.000000
75%	3.000000	36.000000	0.000000
max	3.000000	80.000000	6.000000

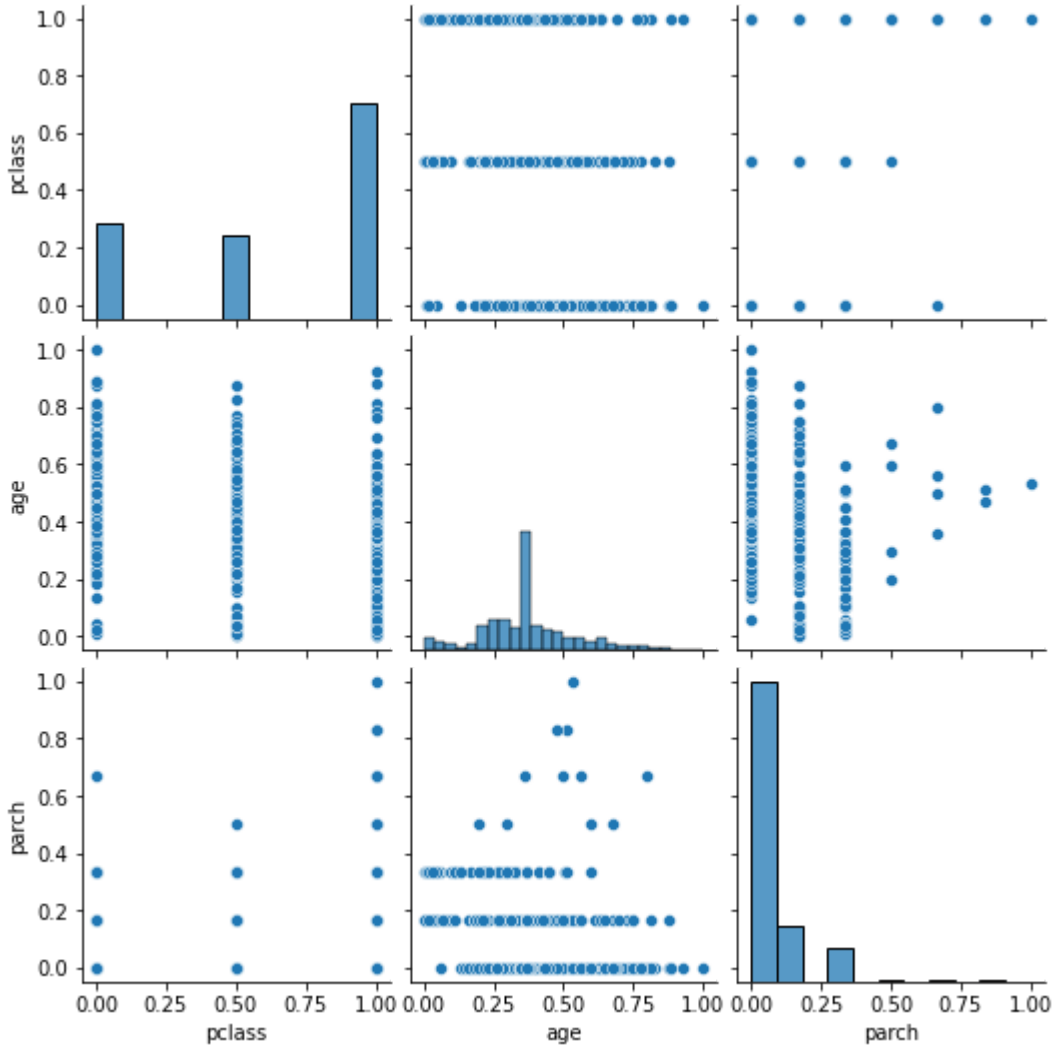
In [37]:
```
1 xtest_mms_df.describe()
```

Out[37]:

	pclass	age	parch
count	179.000000	179.000000	179.000000
mean	0.670391	0.349295	0.070764
std	0.417679	0.148310	0.152861
min	0.000000	0.004147	0.000000
25%	0.500000	0.271174	0.000000
50%	1.000000	0.359135	0.000000
75%	1.000000	0.396833	0.000000
max	1.000000	0.761247	0.833333

In [38]:
```
1 sns.pairplot(xtrain_mms_df)
```

Out[38]: <seaborn.axisgrid.PairGrid at 0x23975864910>

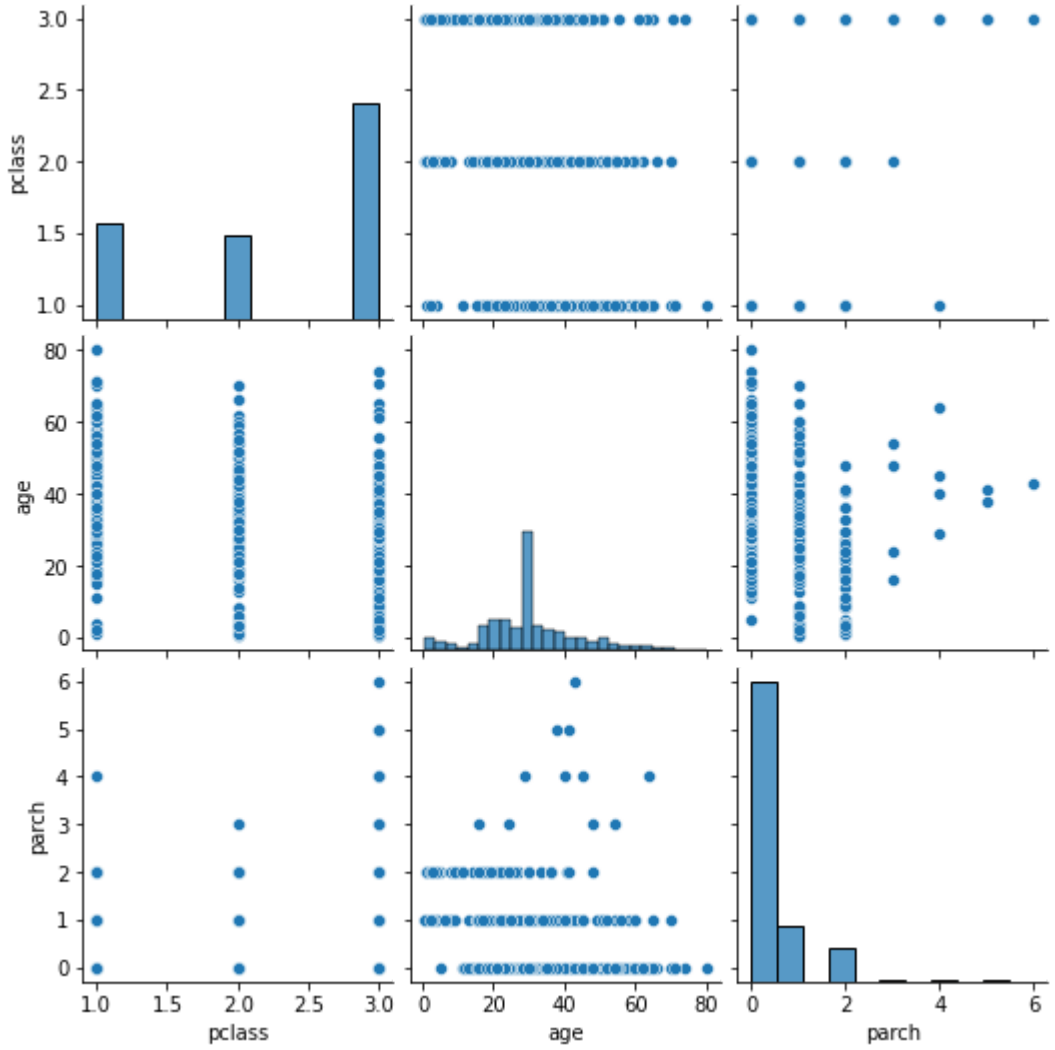


In [ ]:
```
1
```

Comparing with Pair Plot

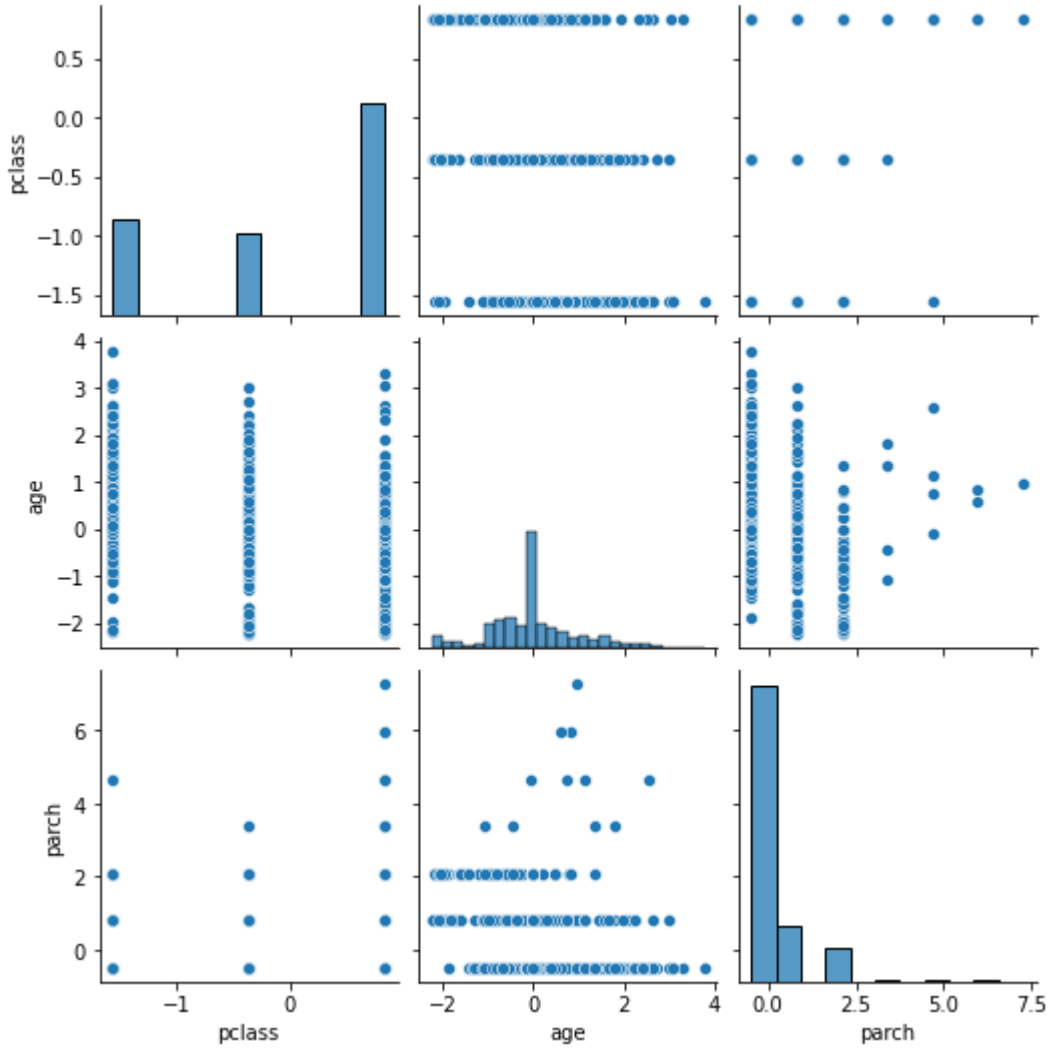
In [39]:
```
1 sns.pairplot(xtrain)
```

Out[39]: <seaborn.axisgrid.PairGrid at 0x23975e791c0>



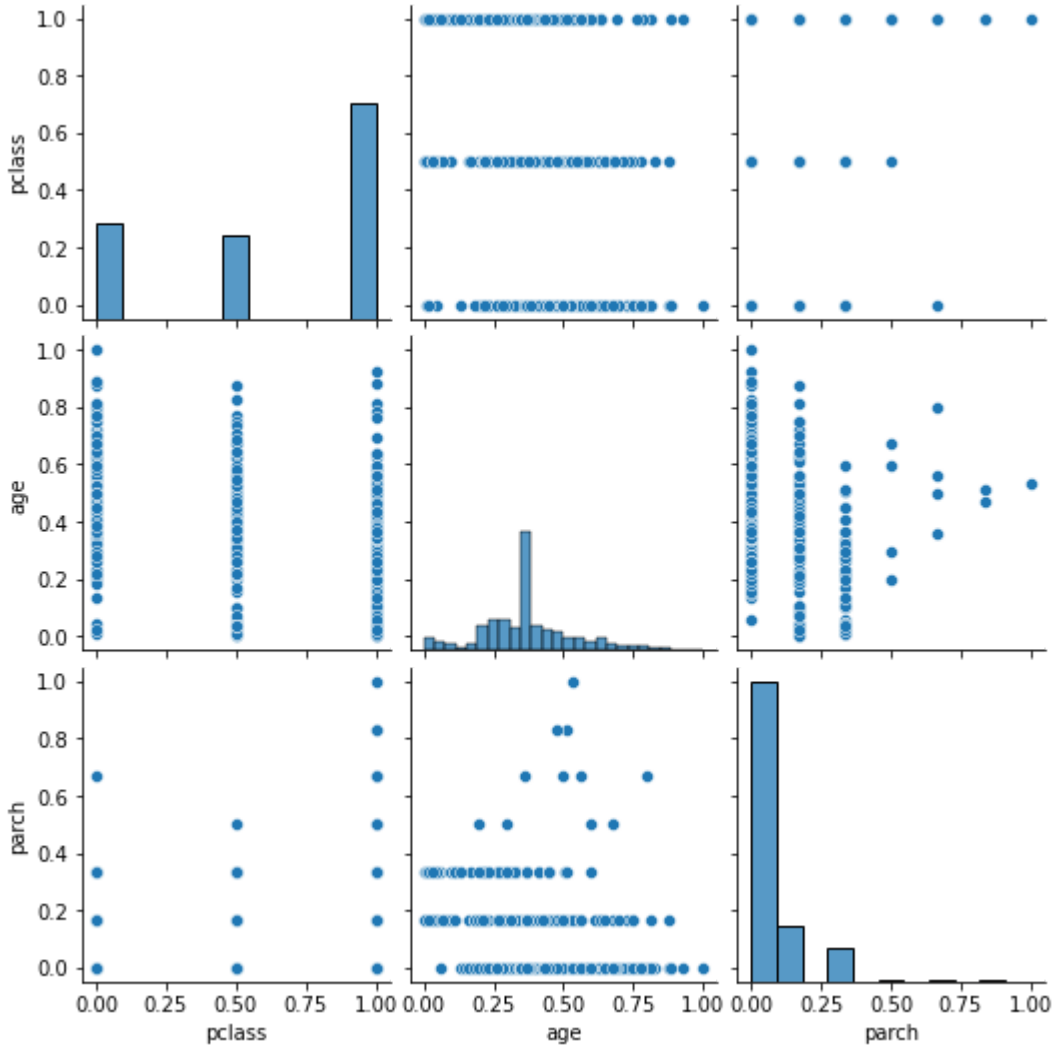
In [40]:
```
1 sns.pairplot(xtrain_sc_df)
```

Out[40]: <seaborn.axisgrid.PairGrid at 0x23975e922b0>

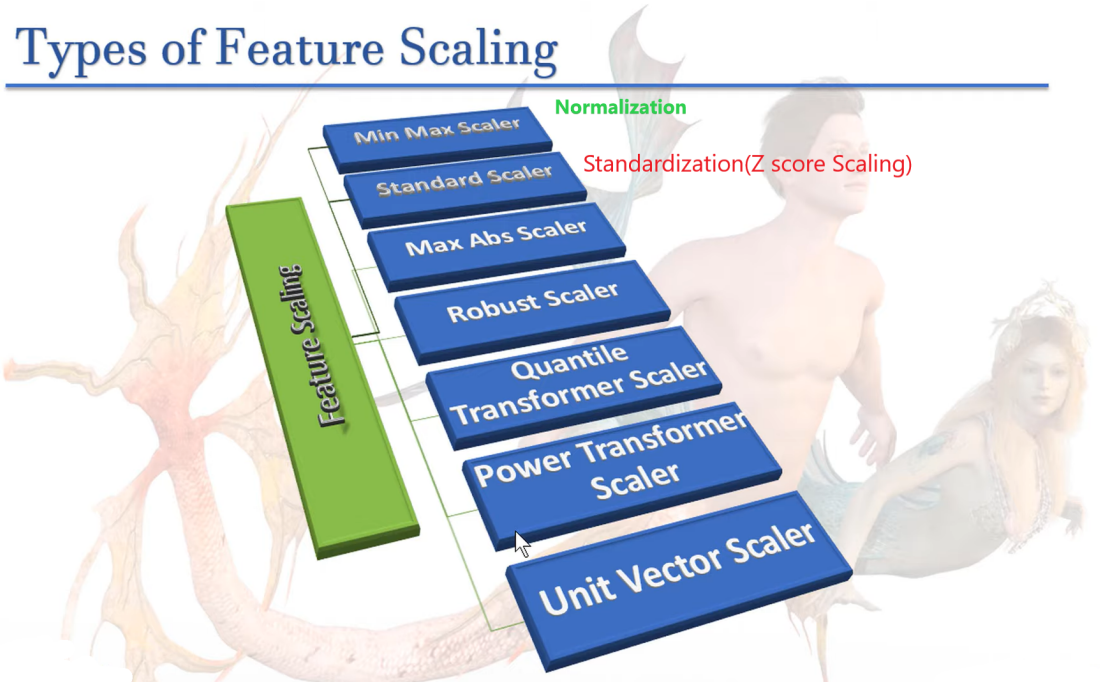


In [41]:
```
1 sns.pairplot(xtrain_mms_df)
```

Out[41]: <seaborn.axisgrid.PairGrid at 0x23977a94700>



Others Feature Engineering


Types of Feature Scaling
Normalization
Standardization(Z score Scaling)
Feature Scaling
Min Max Scaler
Standard Scaler
Max Abs Scaler
Robust Scaler
Quantile Transformer Scaler
Power Transformer Scaler
Unit Vector Scaler

In [ ]:
```
1
```