import imp library

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

datasets(titanic)

```
df = pd.read_csv('/content/titanic_datasets.csv')
```

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

Next steps: Generate code with df | New interactive sheet

```
df = pd.read_csv('/content/titanic_datasets.csv' , usecols=['Age' , 'Survived' , 'Fare'])
```

```
df.head()
```

	Survived	Age	Fare
0	0	34.5	7.8292
1	1	47.0	7.0000
2	0	62.0	9.6875
3	0	27.0	8.6625
4	1	22.0	12.2875

Next steps: Generate code with df | New interactive sheet

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

	0
Survived	0
Age	86
Fare	1

dtype: int64

```
df['Age'].fillna(df['Age'].mean(),inplace=True)
df['Fare'].fillna(df['Fare'].mean(),inplace=True)
```

```
ning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

e, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


rue)
ning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

e, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


e=True)
```

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

	0
Survived	0
Age	0
Fare	0

dtype: int64

```
df.head()
```

	Survived	Age	Fare
0	0	34.5	7.8292
1	1	47.0	7.0000
2	0	62.0	9.6875
3	0	27.0	8.6625
4	1	22.0	12.2875

Next steps: Generate code with df | New interactive sheet

```
x =df.iloc[:,1:]
y = df.iloc[:,:1]
```

```
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=42)
```

```
x_train
```

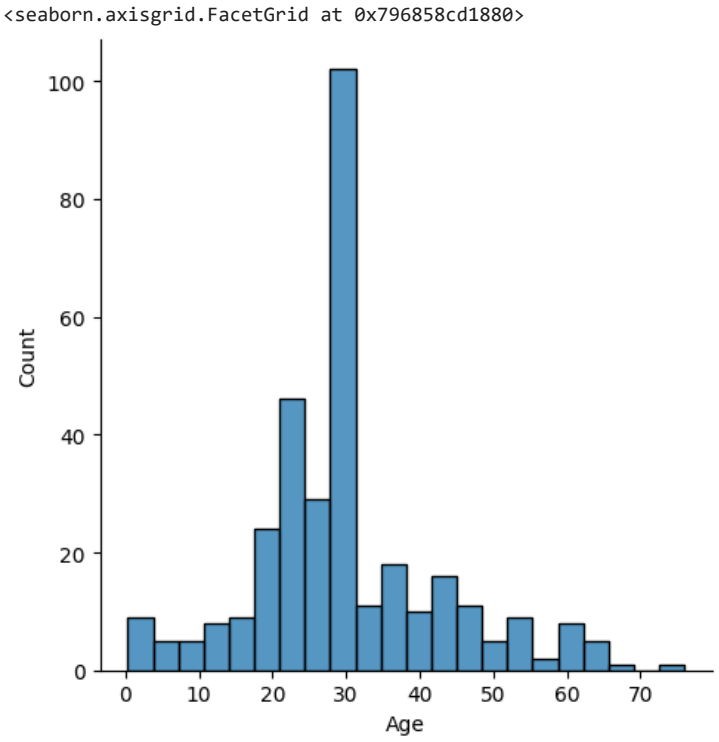
	Age	Fare
336	32.00000	13.0000
31	24.00000	31.5000
84	30.27259	10.7083
287	24.00000	82.2667
317	19.00000	10.5000
...	...	...
71	21.00000	7.8958
106	21.00000	7.8208
270	46.00000	75.2417
348	24.00000	13.5000
102	30.27259	7.7500

334 rows × 2 columns

Next steps: Generate code with `x_train` New interactive sheet

```
sns.displot(x_train['Age'])
```

```
<seaborn.axisgrid.FacetGrid at 0x796858cd1880>
```



Function Transformer

```
from sklearn.preprocessing import FunctionTransformer
```

Log Transformer

```
log_transformer = FunctionTransformer(func=np.log1p)
```

```
log_transformer.fit_transform(x_train)
log_transformer.transform(x_test)
```

	Age	Fare
321	3.258097	2.107689
324	3.688879	5.358177
388	3.091042	2.169054
56	3.583519	2.185579
153	3.610918	2.578951
...	...	...
57	3.258097	2.157559
126	3.135494	2.174274
24	3.891820	5.573579
17	3.091042	2.107178
66	2.944439	2.183711

84 rows × 2 columns

sqr_transformer

```
sqr_transformer = FunctionTransformer(lambda x : x**2)
```

```
sqr_transformer.fit_transform(x_train)
sqr_transformer.transform(x_test)
```

	Age	Fare
321	625.0	52.261333
324	1521.0	44663.538906
388	441.0	60.062500
56	1225.0	62.343658
153	1296.0	148.432799
...	...	...
57	625.0	58.522500
126	484.0	60.774498
24	2304.0	68840.640625
17	441.0	52.200625
66	324.0	62.081793

84 rows × 2 columns

sqr_root_transformer

```
sqr_root_transformer = FunctionTransformer(np.sqrt)
```

```
sqr_root_transformer.fit_transform(x_train)
sqr_root_transformer.transform(x_test)
```

	Age	Fare
321	5.000000	2.688717
324	6.244998	14.537452
388	4.582576	2.783882
56	5.916080	2.809947
153	6.000000	3.490458
...	...	...
57	5.000000	2.765863
126	4.690416	2.792096
24	6.928203	16.197994
17	4.582576	2.687936
66	4.242641	2.806991

84 rows × 2 columns

POWER TRANSFORMER

Box-Cox

```
from sklearn.preprocessing import PowerTransformer
```

```
box_cox = PowerTransformer(method='box-cox')
```

```
box_cox.fit_transform(x_train+0.000000001)
box_cox.transform(x_test+0.000000001)
```

```
array([[-0.34727758, -0.81842492],
       [ 0.70391823,  2.40618683],
       [-0.66942134, -0.76923558],
       [ 0.41350755, -0.75595537],
       [ 0.48675844, -0.4344662 ],
       [ 1.47151638,  0.18886839],
       [-0.03621302, -0.75332007],
       [ 1.40337964,  0.18886839],
       [-0.83543191, -0.38426061],
       [ 0.06074297, -0.74212879],
       [-0.66942134,  0.12450767],
       [ 1.53935708,  0.56710721],
       [-1.09184768, -0.70157492],
       [ 0.70391823,  1.61569954],
       [ 0.06074297, -0.74212879],
       [ 0.06074297,  0.91786511],
       [-0.1130516 , -0.76694387],
       [ 1.80789625,  0.79652565],
       [-1.64009545,  0.28945694],
       [-0.50682284, -0.54741215],
       [ 1.94055959, -0.38426061],
       [ 0.84666548,  0.83258873],
       [-2.39278613, -0.33897427],
       [ 0.04005079, -0.38426061],
       [ 0.06074297, -0.25852995],
       [-0.87751744, -0.81318132],
       [-0.34727758,  0.89898675],
       [ 0.06074297,  0.20727162],
       [ 0.70391823, -0.81842492],
       [ 0.04005079, -0.38426061],
       [ 0.19094547,  0.06451972],
```

```
       [-0.58771971,  0.57396864],
       [ 0.26561804,  1.99835958],
       [ 0.06074297, -0.74212879],
       [-0.58771971, -0.54741215],
       [-0.34727758, -0.54741215],
       [-0.42668831, -0.76923558],
       [ 0.06074297, -0.26908137],
       [-0.03621302,  0.18886839],
       [ 0.22834428,  2.40715697],
       [-0.42668831, -0.76923558],
       [ 0.06074297,  0.09908886],
       [ 0.84666548, -0.26908137],
       [ 0.06074297,  0.57188474],
       [-0.07455894,  0.24542092],
       [-0.66942134, -0.7597237 ],
       [ 0.06074297,  1.13037963],
       [-0.42668831,  0.24542092],
       [ 1.80789625,  0.17869423],
       [ 0.48675844, -0.81640507],
       [ 0.37671169, -0.76199627],
       [ 1.12771959, -0.81883335],
       [ 0.06074297, -0.74212879],
       [ 1.26618822, -0.54741215],
       [-0.03621302, -0.33423301],
       [-0.75197468,  0.18886839],
       [ 0.06074297, -0.74212879],
```

Yeo-Johnson

```
from sklearn.preprocessing import PowerTransformer
```

```
yeo_jonson = PowerTransformer()
```

```
yeo_jonson.fit_transform(x_train)
yeo_jonson.transform(x_test)
```

```
       [-0.34873796,  1.06426324],
       [ 0.0604976 ,  0.41731628],
       [ 0.7051842 , -0.97766898],
       [ 0.03974779, -0.31131584],
       [ 0.19105097,  0.25833336],
       [-0.58994857,  0.78358946],
       [ 0.26591517,  1.78705262],
       [ 0.0604976 , -0.85174925],
       [-0.58994857, -0.54759084],
       [-0.34873796, -0.54759084],
       [-0.42840014, -0.89604948],
       [ 0.0604976 , -0.15427929],
       [-0.03673254,  0.39737025],
       [ 0.22854666,  1.99022729],
       [-0.42840014, -0.89604948],
       [ 0.0604976 ,  0.29774461],
       [ 0.84818551, -0.15427929],
       [ 0.0604976 ,  0.781665  ],
       [-0.07518968,  0.45816086],
       [-0.67191343, -0.88044951],
       [ 0.0604976 ,  1.24267087],
       [-0.42840014,  0.45816086],
       [ 1.81028646,  0.38627464],
       [ 0.48757892, -0.97428624],
       [ 0.37728021, -0.88417125],
       [ 1.12964688, -0.97835333],
       [ 0.0604976 , -0.85174925],
       [ 1.26827002, -0.54759084],
       [-0.03673254, -0.24213862],
       [-0.75473108,  0.39737025],
       [ 0.0604976 , -0.85174925],
       [-0.26977034, -0.31131584],
       [-2.77512929,  0.16849407],
       [-0.42840014, -0.97428624],
       [ 1.47377065, -0.19361231],
       [-0.67191343, -0.44599719],
       [ 0.0604976 ,  0.45816086],
       [ 0.77688336,  0.39737025],
       [-1.18358562,  0.76832708],
       [-0.34873796, -0.54759084],
       [ 0.84818551, -0.88107384],
       [-0.42840014, -0.9266548 ],
       [ 0.7051842 ,  0.52267036],
       [ 0.98964451,  0.45816086],
       [ 1.94295348,  1.76553593],
       [ 0.03974779,  0.18892021],
       [ 0.19105097, -0.92213077],
       [ 0.0604976 ,  0.58228424],
       [ 1.12964688,  0.52267036],
       [-0.58994857, -0.76676615],
       [ 0.03974779,  0.39737025],
       [-0.58994857, -0.23890597],
       [-0.26977034,  0.23621193],
       [-0.34873796, -0.91124125],
       [-0.58994857, -0.88916415],
       [ 1.33708659,  2.11297168],
       [-0.67191343, -0.97835333],
       [-0.92313102, -0.87674093]])
```