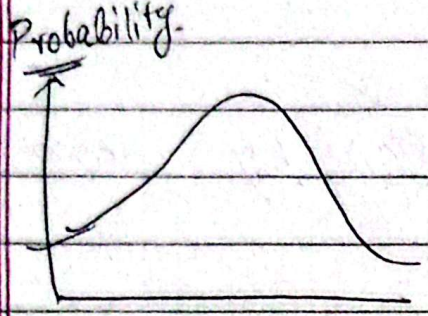
Function transformer.

Page No.
Date : / /



↳ normal distribution

- we won't get the data normally distributed
- The main objective of the function transformer is to distribute data normally.
- It makes problem solving more easier.
- Statistics algorithms like linear regression, logistic regression needs to be normally distributed data.
  However, DecisionTree, RandomForest ~~do~~ they doesn't need ~~@~~ normally distributed data.

Page No.
Date: / /

Sklearn

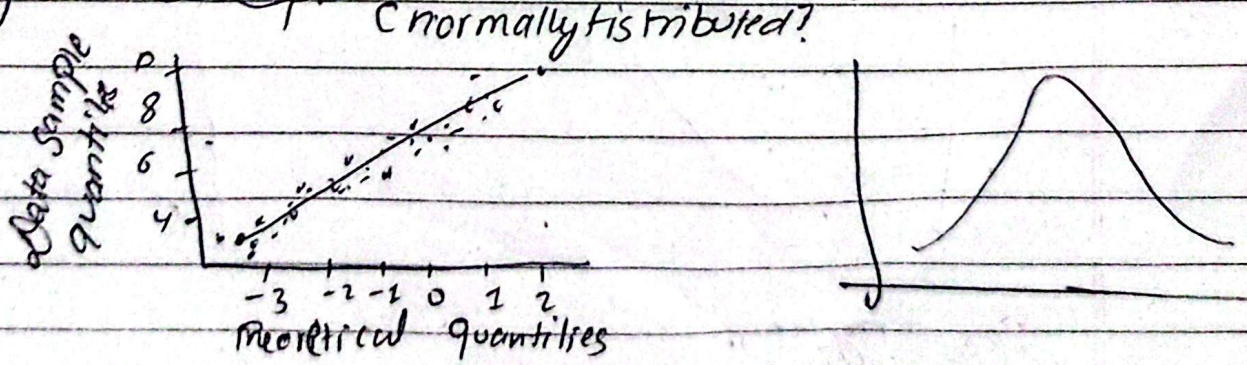
- Function Transformer
  - log transform
  - Reciprocal
  - square / sq.root
  - Custom
- Power Transformer
  - Box-Cox
  - Yeo-Johnson.
- Quantile Transformer.

(Q) How to find if data is normal?

Way 1: Seaborn displot:

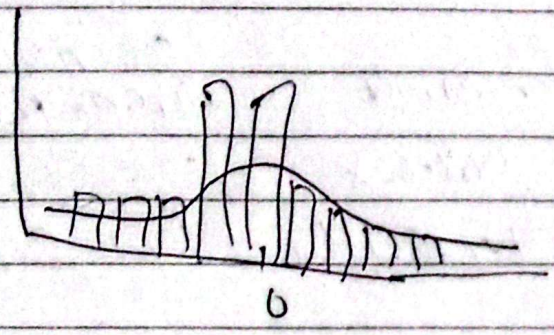
Way 2: Pandas .skew()
it should be 0.

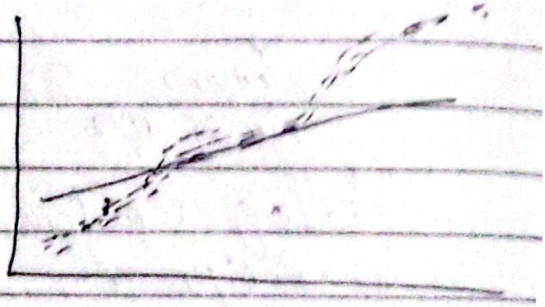
way 3: QQ plot: (most reliable).



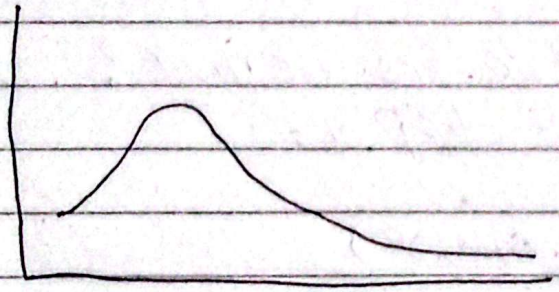
Page No.
Date: / /

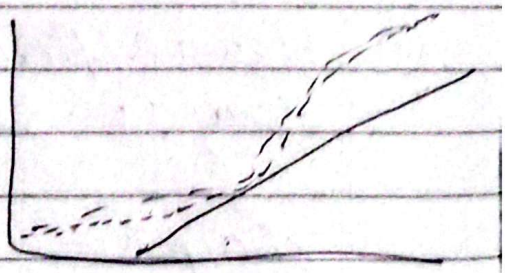
two peak in middle
~~Low Peak~~ :



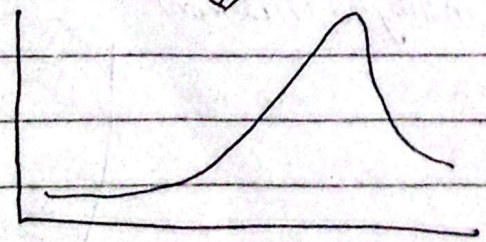


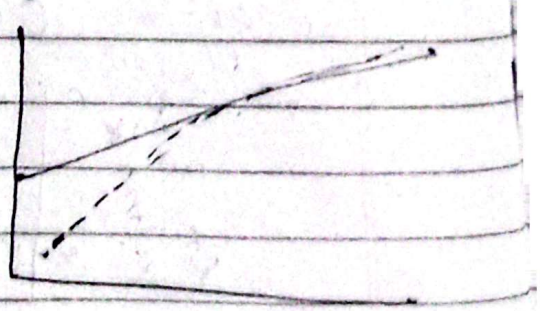
- Skewed data:



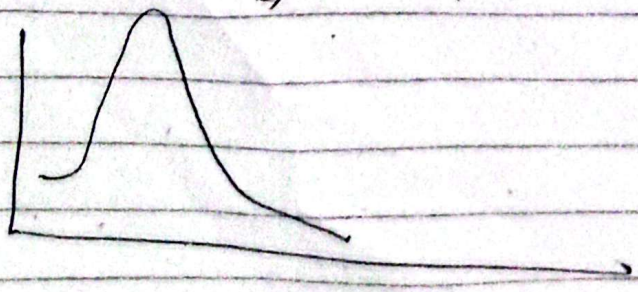


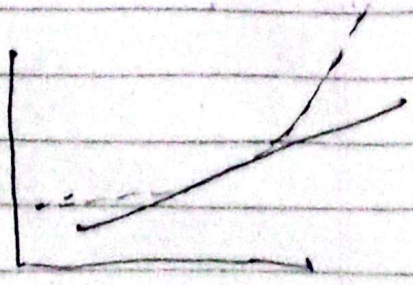
(Skew left)





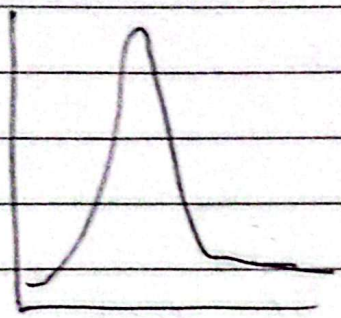
(Skew right)



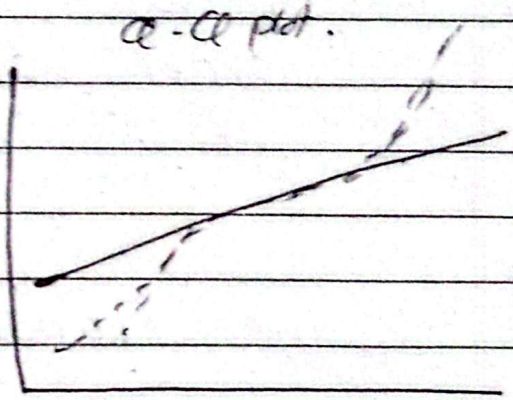


Page No.
Date: / /

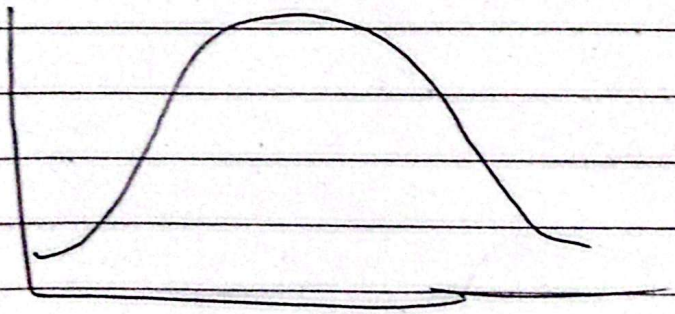
Fat-tails

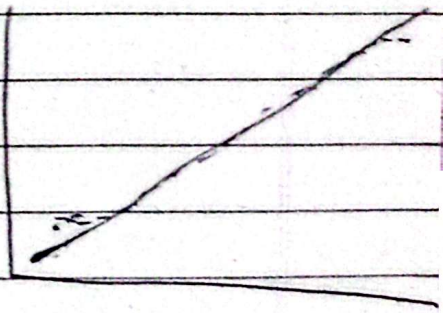


Q-Q plot.



Thin-tails





Log Transform.

- Take log of that value
- Then, data will be more normally distributed, ~~than~~ then that of right now.
- Better to use in right skewed data.

Don't do in . -ve values

.

Page No.
Date : / /

Reciprocal Transforms: ($1/x$)
larger values changes to smaller values and vice versa.

Square transforms (x^2)
→ use for left skewed data,

Square root transforms ($\sqrt{x}$)

Code

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

import scipy.stats as stats

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import FunctionTransformer
```

[2]:
```
df = pd.read_csv('titanic_dataset.csv', usecols=['Age','Fare','Survived'] )
df.sample(5)
```

[2]:

	Survived	Age	Fare
267	0	NaN	7.5500
284	1	2.0	20.2125
8	1	18.0	7.2292
110	0	41.0	15.0458
239	1	48.0	106.4250

Feb 28 23:01

```
[9]: X_train.shape

[9]: (334, 2)

[10]: X_test.shape

[10]: (84, 2)

[11]: print(X_train['Fare'])
      print(X_train['Fare'].dtype)
      print(X_train['Fare'].head())

155     24.00000
287     24.00000
300     32.00000
394     29.00000
261     21.00000
          ...
211     30.27259
67      47.00000
25      50.00000
196      6.00000
175     15.00000
Name: Fare, Length: 334, dtype: float64
float64
155    24.0
287    24.0
300    32.0
394    29.0
261    21.0
Name: Fare, dtype: float64
```

Code

```
Name: Fare, dtype: float64
```

[14]:
```
stats.probplot(X_train['Fare'], dist = 'norm',plot=plt)
plt.show()
```