#MATHEMATICAL TRANSFORMATIONS

###Function Transformer
(i) Log Transformer
(ii) Reciprocal Transformer
(iii) Power Transformer (square/square root transformer)
(iv) Box-Cox transformer
(v) Yeo-Johnson Transformer

Why to do transformer?
-> Increase in model performance.
-> Data distribution (PDF) converts into Normal function/Normal Distribution

Why Normal Distribution?
-> For Statistics, claculations and problem solving gets easy.

In Machine Learning, Linear Regresion, Logistic Regression we expect normally distributed data on the other hand, Decision Tree, Random Forest don't expect normally distributed data.

Function Transformer:

```
                                   sklearn
    --------------------------------------|--------------------------------------
    |                                     |                                |
    Function                               Power                          Quantile
    Transformer:                         Transformer:                   Transformer:
    [Log Transformer                     [Box-Cox Transformer           [
     Reciprocal Transformer               Yeo-Johnson Transformer]
     Square Transformer
     Square root Transformer
     custom]
```

How to find if data is normal?

Trick 1:
sns.distplot
We can use distplot from seaborn

Trick 2:
pd.skew()
We can use skew function from pandas.
if 0 -> then it's all good
else -> skewed

Trick 3:
QQ plot

QQ Plot

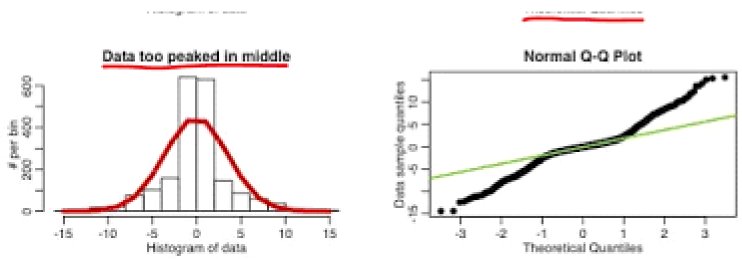
###**Normally Distributed**

x -> Theoretical Quantile
y -> Data Sample Quantile

The image on the left is a pdf (probability density function), and another is a QQ plot.

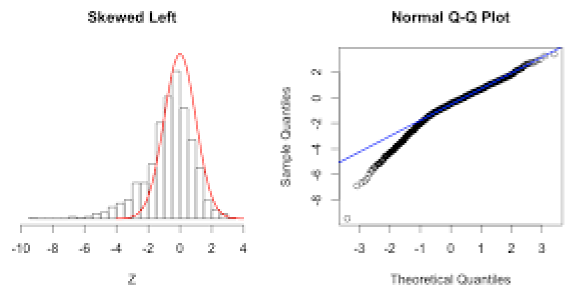
The pdf shows the dataset is normally distributed.
For normally distributed data in the QQ plot, all the points will come above the 45-degree line as you can see in the below image
In the QQ plot all data points are occured on 45 degree line.
In dist plot it's PDF shows it's normally distributed

###Too Peaked from Middle



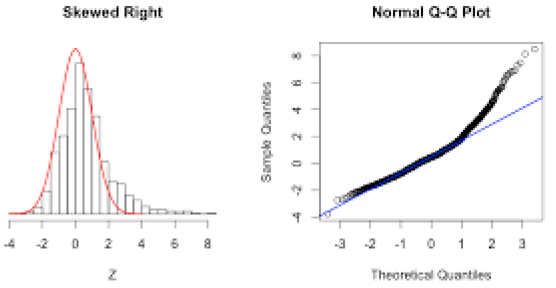
If the data is overly skewed in the middle, your QQ plot will be slightly deviated from line, and the line will deviate slightly
Here we can see in this plot, data is too peaked in the middle. In this case our QQ plot data move slightly from the line and angle of line also gets reduces.
(QQ plot data points deviates from line)

###Left Skewed



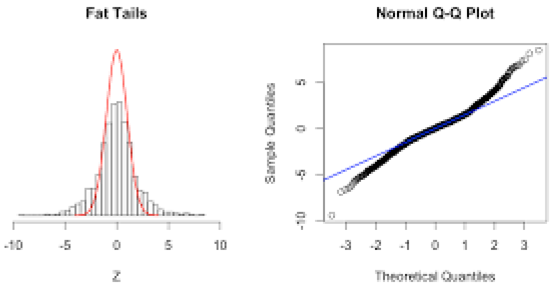
Here the data is skewed left and easily we can observe it.
In QQ plot the data points are leaving line from lower part.

###Right Skewed



Here the data is skewed right, and easily we can also observe in the plots specially in QQ plot the data are leaving from line from up.

###Fat Tails



###Thin Tails

Thin Tails Normal Q-Q Plot

The more our data lying on line, the more is the Normal Distribution, and the far the data is from line far is our Normal Distribution.

###Log Transform:
For ex: In titanic dataset if we want to apply log transform on column 'Age' we directly apply Log function on it so that it can be normally distributed.

When to use Log Transform?
--> Do not apply on -values.
--> If we have right skewed data.
--> Equivalent the scale and linearly distributed.

###Reciprocal Transform (1/x)
If large value is there it gets to smaller values and if smaller values are there, it is used to convert them to large values.
In this transformation, x will replace by the inverse of x (1/x). The reciprocal transformation will give little effect on the shape of the distribution.
This transformation can be only used for non-zero values. The skewness for the transformed data is increased.

###Square Transform (x^2)
Used for left skewed data.
It is commonly applied to counted data, especially if the values are mostly rather small. The square, x to x^2, has a moderate effect on distribution shape and it could be used to reduce left skewness. In practice, the main reason for using it is to fit a response by a quadratic function y = a + b x + c x^2.

###Square root Transform (x^1/2)
Used for count data (data that follow a Poisson distribution) or small whole numbers. This transformation also may be appropriate for percentage data where the range is between 0 and 20% or between 80 and 100%.

In [ ]:

```

```

Use all transformation techniques to get the best normalization result.

Ex: Titanic dataset
(Age, Fare, Survive)

Let's check without transform and afterwards we will use transform technique.

###**WITHOUT TRANSFORMATION**

Import necessary libraries

In [1]:
```
import pandas as pd
import numpy as np

import scipy.stats as stats

import matplotlib.pyplot as plt
import seaborn as sns

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.preprocessing import FunctionTransformer
from sklearn.compose import ColumnTransformer
```

Load Dataset

In [2]:
```
Titanic_dataset = pd.read_csv('titanic_dataset.csv', usecols=['Age','Fare','Survived'])
```

In [3]:
```
Titanic_dataset
```

Out[3]:

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

891 rows × 3 columns

In [4]:
```
Titanic_dataset.head()
```

Out[4]:

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

In [5]:
```
Titanic_dataset.isna().sum()
```

Out[5]:
```
Survived      0
Age         177
Fare          0
dtype: int64
```

So there are 177 missing values for 'Age'.
Let's fill those with it's overall mean.

In [6]:
```
Titanic_dataset = Titanic_dataset.fillna(Titanic_dataset['Age'].mean())
```

In [7]:
```
Titanic_dataset
```

Out[7]:

	Survived	Age	Fare
0	0	22.000000	7.2500
1	1	38.000000	71.2833
2	1	26.000000	7.9250
3	1	35.000000	53.1000
4	0	35.000000	8.0500
...	...	...	...
886	0	27.000000	13.0000
887	1	19.000000	30.0000
888	0	29.699118	23.4500
889	1	26.000000	30.0000
890	0	32.000000	7.7500

891 rows × 3 columns

Age and Fare will be our independent variables i.e., x.
Survived will be our dependent variable i.e., y.
Then Train Test and Split the data.

Survived -> Target/Dependent variable
Age and Fare -> Independent variables

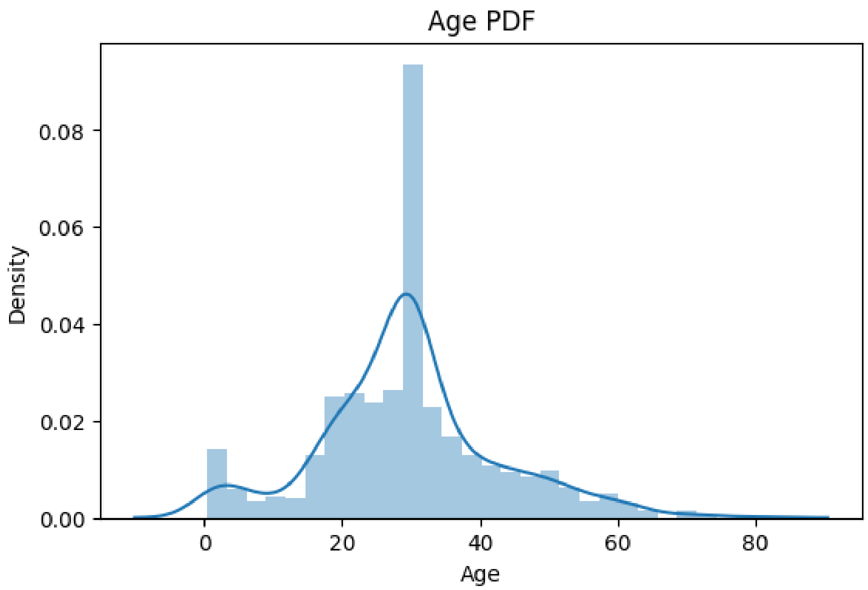
In [8]:
```
X = Titanic_dataset.iloc[:,1:3]
y = Titanic_dataset.iloc[:,0]
```

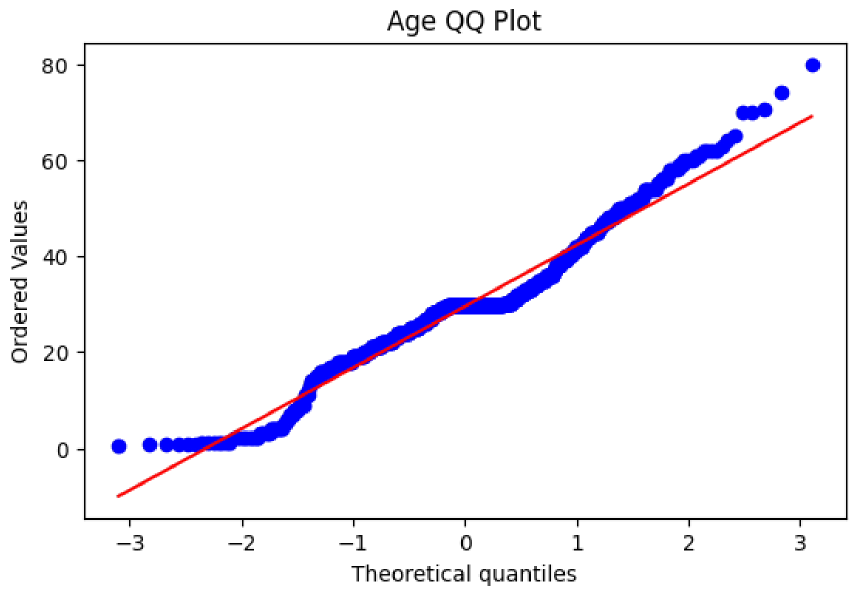
In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=42)
```

Now let's see whether Age and Fare columns are normally distrbuted or not by plotting PDF plot and QQ plot.

In [10]:
```
plt.figure(figsize=(14,4))
plt.subplot(121)
sns.distplot(X_train['Age'])
plt.title('Age PDF')

plt.subplot(122)
stats.probplot(X_train['Age'], dist="norm", plot=plt)
plt.title('Age QQ Plot')

plt.show()
```

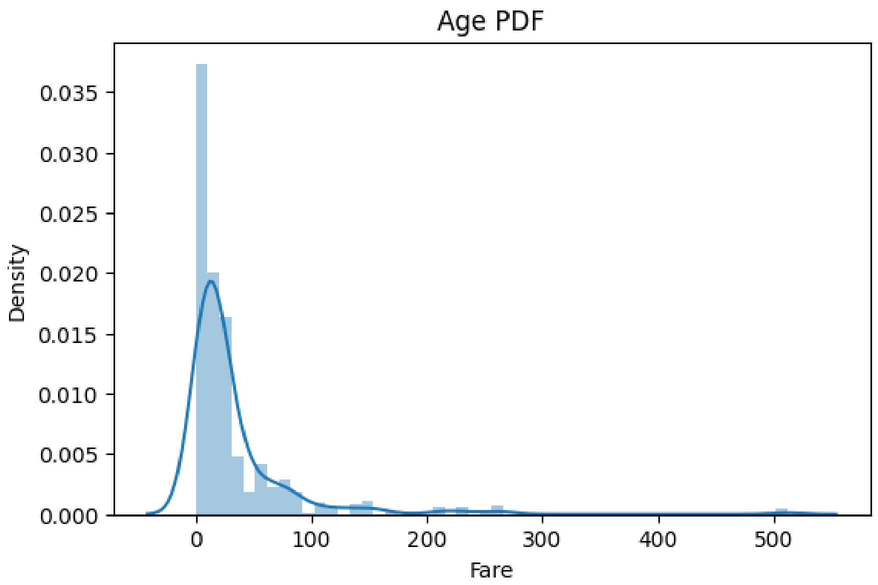
```
<ipython-input-10-1c15e8485d0e>:3: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train['Age'])
```

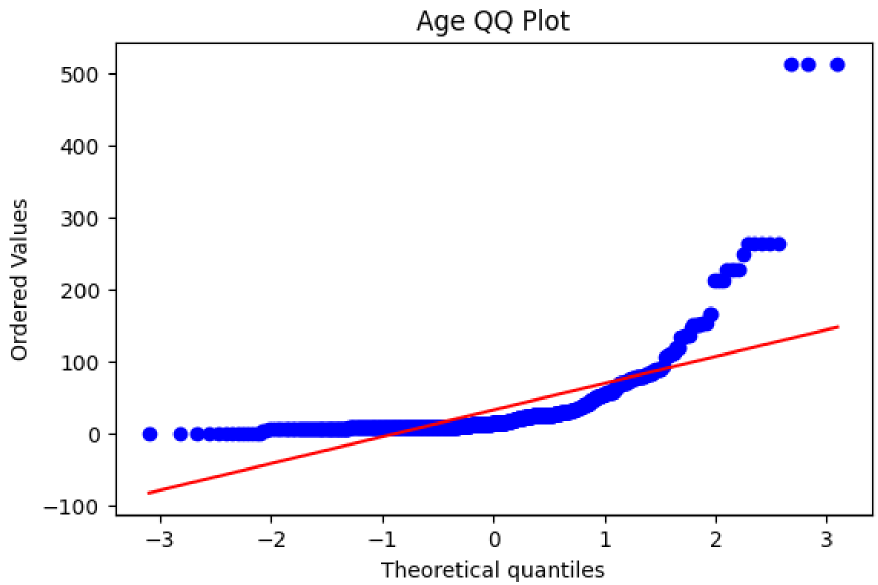

Age PDF
Density
0.08
0.06
0.04
0.02
0.00
0
20
40
60
80
Age


Age QQ Plot
Ordered Values
80
60
40
20
0
−3
−2
−1
0
1
2
3
Theoretical quantiles

In [11]:
```
plt.figure(figsize=(14,4))
plt.subplot(121)
sns.distplot(X_train['Fare'])
plt.title('Age PDF')

plt.subplot(122)
stats.probplot(X_train['Fare'], dist="norm", plot=plt)
plt.title('Age QQ Plot')

plt.show()
```

```
<ipython-input-11-921b0fa8d620>:3: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(X_train['Fare'])
```





By observing the plot we can say it is not perfectly distributed as it is deviating at some places from the line but it is closer to line mostly.

For Fare,
It is not normally distributed, well it is right skewed (So we can say, some passengers purchased tickets at high price where some purchased in less price)

As it is right skewed, we should use 'log transformer'

Firstly I will perform classifier techniques without transform method.
We will perform two techniques
(i) Logistic Regression
(ii) Decision Tree Classifier

In [12]:
```
LR = LogisticRegression()
DTC = DecisionTreeClassifier()
```

In [13]:
```
LR.fit(X_train,y_train)
DTC.fit(X_train,y_train)
```

Out[13]:
```
DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Then I fit my X_train and y_train in these models and predict X_test.

In [14]:
```
y_pred = LR.predict(X_test)
y_pred1 = DTC.predict(X_test)

print("Accuracy LR",accuracy_score(y_test,y_pred))
print("Accuracy DT",accuracy_score(y_test,y_pred1))
```

```
Accuracy LR 0.6480446927374302
Accuracy DT 0.6703910614525139
```

Then I find the accuracy for y_pred and y_test (actual).

###WITH TRANSFORM

Let's use FunctionTransformer (func=np.log1p)

In [15]:
```
FT = FunctionTransformer(func=np.log1p)
```

np.log -> x
np.log1p -> (x+1)

Fitting Transformation technique on X_train and transformation on X_test.

So that this transformation technique will be applied on Age and Fare of training data and same will be transformed on testing data.

In [16]:
```
X_train_transformed = FT.fit_transform(X_train)
X_test_transformed = FT.transform(X_test)
```

In [17]:
```
LR1 = LogisticRegression()
DTC1 = DecisionTreeClassifier()
```

In [18]:
```
LR1.fit(X_train_transformed,y_train)
DTC1.fit(X_train_transformed,y_train)
```

Out[18]:
```
DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [19]:
```
y_pred = LR1.predict(X_test_transformed)
y_pred1 = DTC1.predict(X_test_transformed)

print("Accuracy LR",accuracy_score(y_test,y_pred))
print("Accuracy DT",accuracy_score(y_test,y_pred1))
```

```
Accuracy LR 0.6815642458100558
Accuracy DT 0.6759776536312849
```

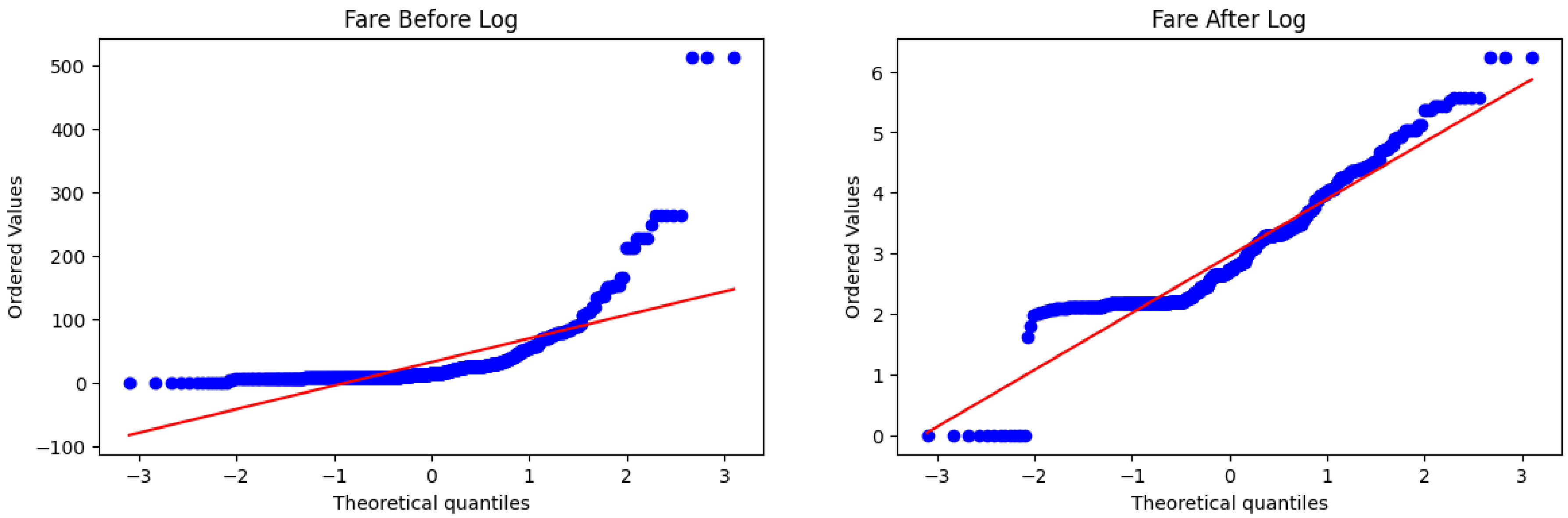
After transformation I applied classification technique on X_train, y_train.
Then predicted the results for X_test and check accuracy.

Now one thing we can observe , that is when we are predicting the result without transformation technique the Decision Tree gives us result accuracy: 0.6759776536312849 i.e., around 67.6%.
And when we use transformation technique we get the result accuracy: 0.6703910614525139 i.e., 67%
So the accracy for Decision tree is almost same i.e., with transformation and without transformation.
But when we can observe that our accuracy is increased for Logistic Regression after using transformation technique.

Logistic Regression result accuracy without transformation teechnique = 0.6480446927374302 i.e., 64.8%.
Logistic Regression result accuracy with transformation teechnique = 0.6815642458100558 i.e., 68%.

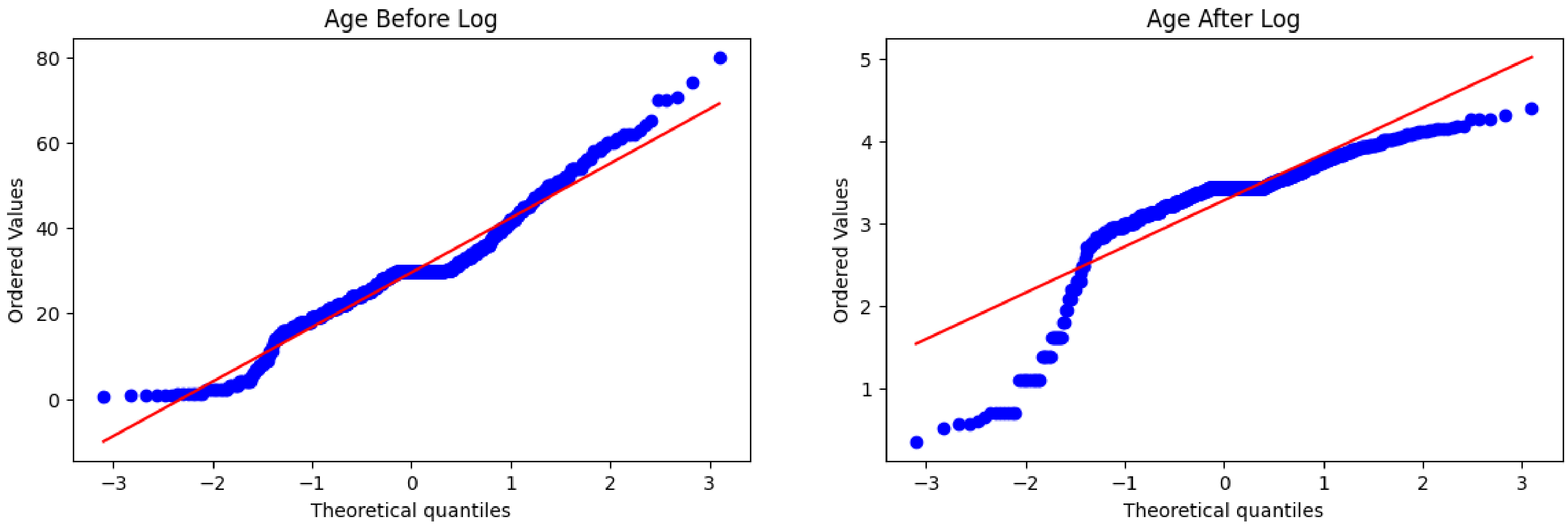
Now let's plot for Fare before Log Transformation and Fare after Log Transformation.

In [20]:
```
plt.figure(figsize=(14,4))

plt.subplot(121)
stats.probplot(X_train['Fare'], dist="norm", plot=plt)
plt.title('Fare Before Log')

plt.subplot(122)
stats.probplot(X_train_transformed['Fare'], dist="norm", plot=plt)
plt.title('Fare After Log')

plt.show()
```



Here we can observe that before transformation the data is right skewed.
So after applying Log Transformation our QQ plot shows good transformation comparitively for **'Fare'**

In [21]:
```
plt.figure(figsize=(14,4))

plt.subplot(121)
stats.probplot(X_train['Age'], dist="norm", plot=plt)
plt.title('Age Before Log')

plt.subplot(122)
stats.probplot(X_train_transformed['Age'], dist="norm", plot=plt)
plt.title('Age After Log')

plt.show()
```



Same I do for Age here, but here after Log Transformation our QQ plot shows bad normalization.
First one plot (Without using Log Transformer) looks better than second plot (using Log Transformer) So for **'Age'** Log Transform is not suitable.
Reason being was for **'Age'** the data was not right skewed, just the **'Fare'** data was right skewed and we use log transform for right skewed data.

I realise that rather than applying Log Transform on both **'Age'** and **'Fare'** just apply Log Transform on **'Fare'** (As it's data is only right skewed).

In [22]:
```
CT = ColumnTransformer([('log',FunctionTransformer(np.log1p),['Fare'])],remainder='passthrough')

X_train_transformed2 = CT.fit_transform(X_train)
X_test_transformed2 = CT.transform(X_test)
```

Again Train data and Test it and then predict the result expected.
Then we cross validate.

In [23]:
```
LR1 = LogisticRegression()
DTC1 = DecisionTreeClassifier()
```

In [24]:
```
LR1.fit(X_train_transformed2,y_train)
DTC1.fit(X_train_transformed2,y_train)
```

Out[24]:
```
DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [25]:
```
y_pred = LR1.predict(X_test_transformed2)
y_pred2 = DTC1.predict(X_test_transformed2)

print("Accuracy LR",accuracy_score(y_test,y_pred))
print("Accuracy DT",accuracy_score(y_test,y_pred2))
```

```
Accuracy LR 0.6703910614525139
Accuracy DT 0.664804469273743
```

So finally I just applied Log Transform on Fare and not Age.

In [26]:
```
X_transformed2 = CT.fit_transform(X)

LR1 = LogisticRegression()
DTC1 = DecisionTreeClassifier()

print("LR",np.mean(cross_val_score(LR1,X_transformed2,y,scoring='accuracy',cv=10)))
print("DT",np.mean(cross_val_score(DTC1,X_transformed2,y,scoring='accuracy',cv=10)))
```

```
LR 0.6712609238451936
DT 0.6532459425717853
```

So Cross validation result comes out as 67% accuracy for LogisticRegression and 66% accuracy for DecisionTreeClassifier.

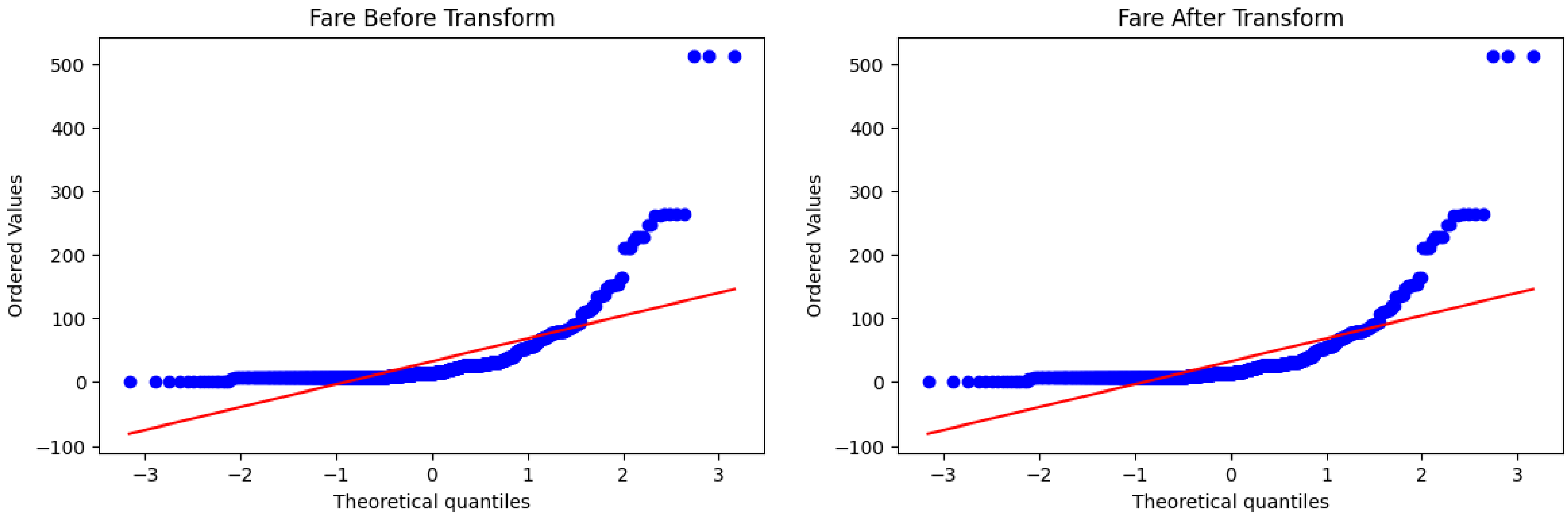
Creating Transformation:

In [27]:
```
def apply_transform(transform):
    X = Titanic_dataset.iloc[:,1:3]
    y = Titanic_dataset.iloc[:,0]

    C_T = ColumnTransformer([('log',FunctionTransformer(transform),['Fare'])],remainder='passthrough')

    X_trans = C_T.fit_transform(X)

    L_R = LogisticRegression()

    print("Accuracy",np.mean(cross_val_score(L_R,X_trans,y,scoring='accuracy',cv=10)))

    plt.figure(figsize=(14,4))

    plt.subplot(121)
    stats.probplot(X['Fare'], dist="norm", plot=plt)
    plt.title('Fare Before Transform')

    plt.subplot(122)
    stats.probplot(X_trans[:,0], dist="norm", plot=plt)
    plt.title('Fare After Transform')

    plt.show()
```

####Log Transform

In [28]:
```
apply_transform(lambda x:x) # Log Transform
```

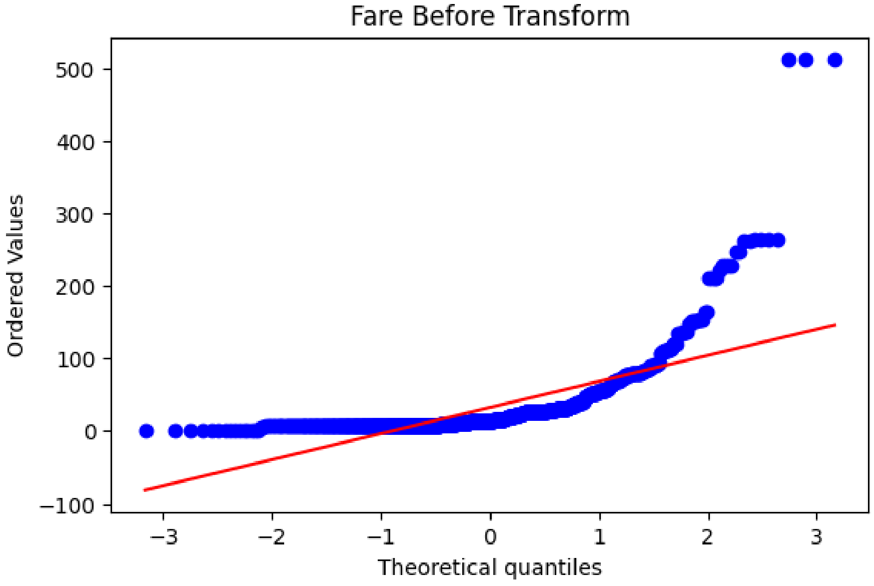
```
Accuracy 0.6589013732833957
```

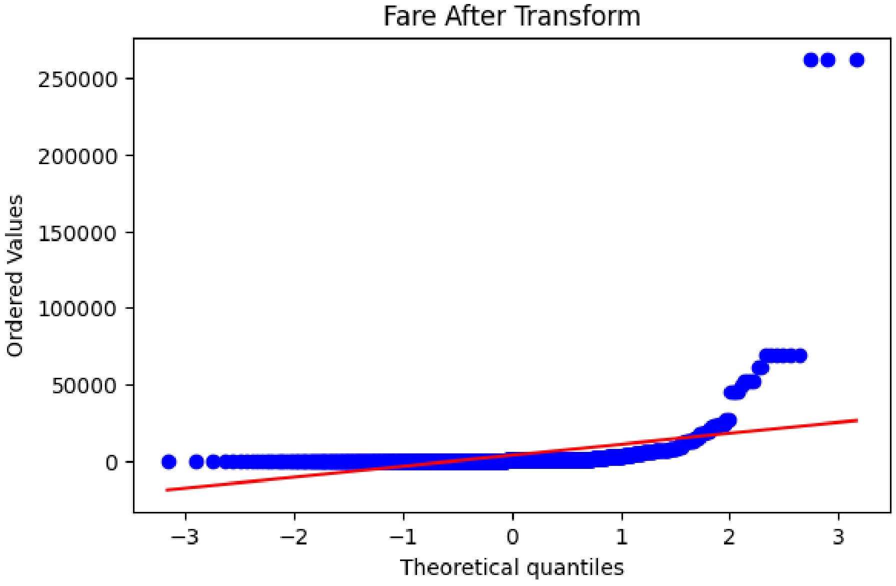


####Squared Transform

In [29]:
```
apply_transform(lambda x: x**2) # Squared Transform
```

```
Accuracy 0.6442446941323345
```

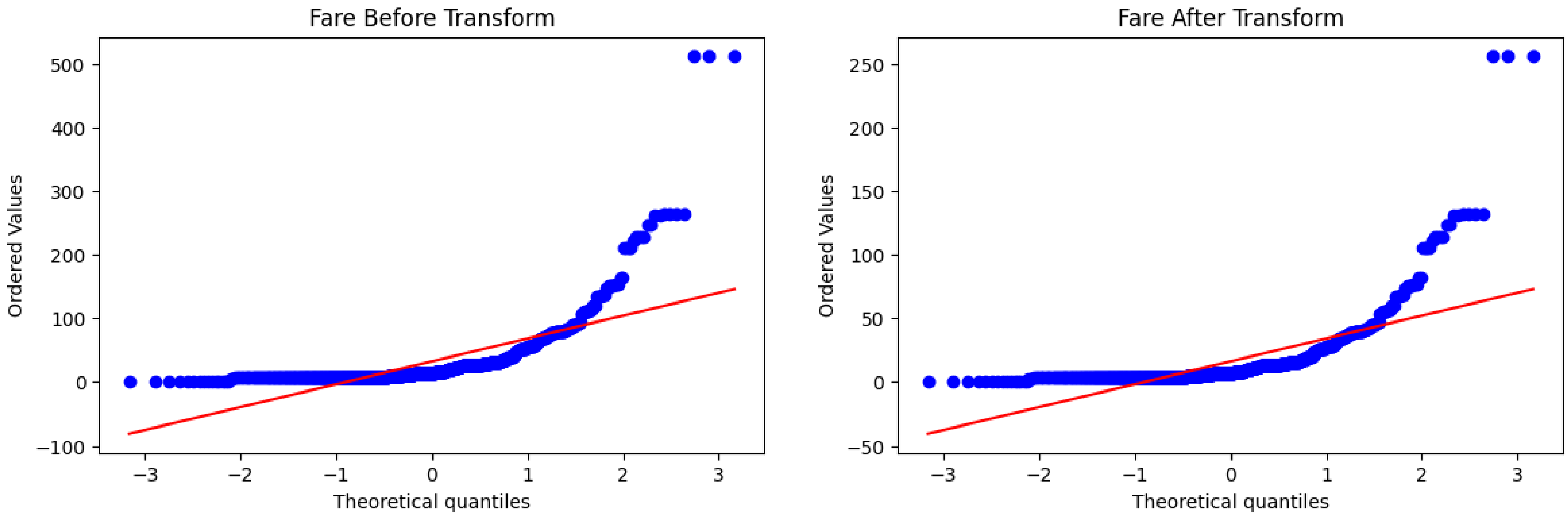




####Square Root Transform

In [30]:
```
apply_transform(lambda x: x**1/2) # Square root Transform
```

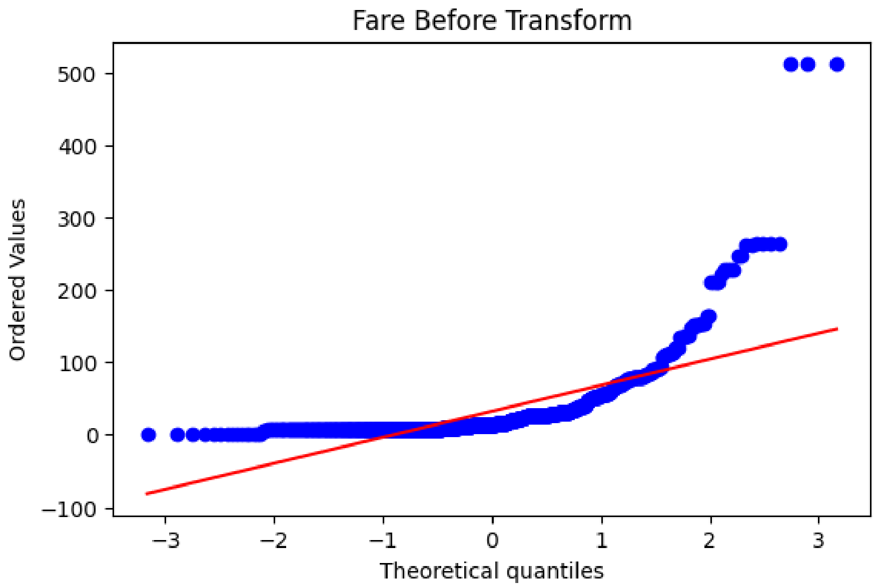
```
Accuracy 0.6589013732833957
```

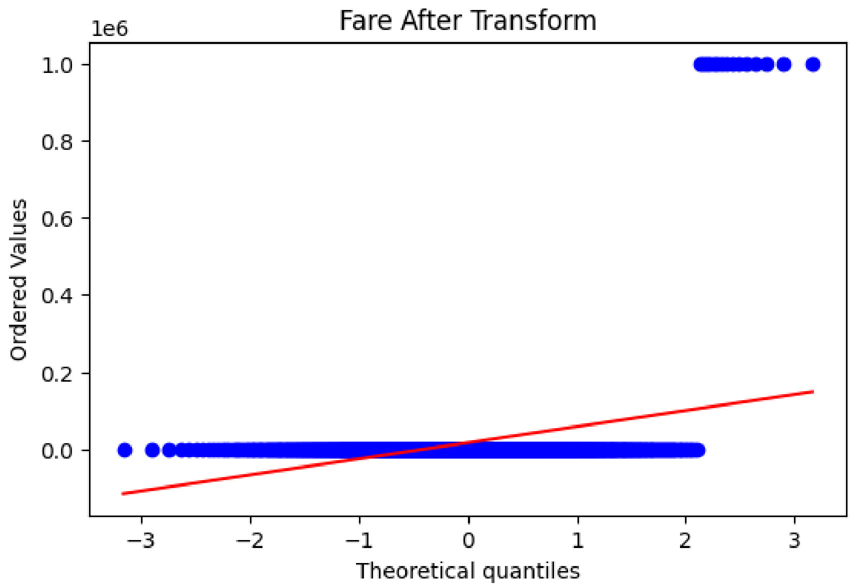


####Reciprocal Transform

In [31]:
```
apply_transform(lambda x: 1/(x + 0.000001)) # Reciprocal Transform
```

```
Accuracy 0.61729088639201
```

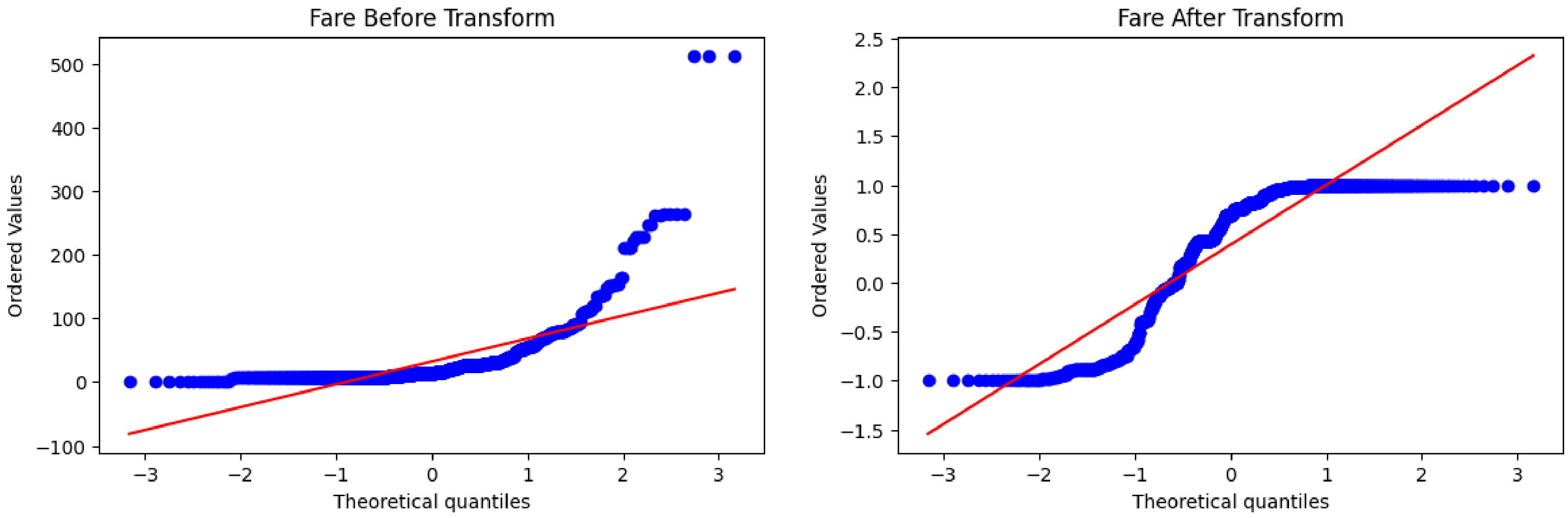




####Sinosuidal

In [32]:
```
apply_transform(np.sin)
```

```
Accuracy 0.6195131086142323
```



#POWER TRANSFORMER

Box-Cox Transform
Yeo-Johnson Transform

$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln(y_i) & \text{if } \lambda = 0, \end{cases}$$

Box–Cox transformation [edit]

The one-parameter Box–Cox transformations are defined as

$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln y_i & \text{if } \lambda = 0, \end{cases}$$

and the two-parameter Box–Cox transformations as

$$y_i^{(\lambda)} = \begin{cases} \dfrac{(y_i + \lambda_2)^{\lambda_1} - 1}{\lambda_1} & \text{if } \lambda_1 \neq 0, \\ \ln(y_i + \lambda_2) & \text{if } \lambda_1 = 0, \end{cases}$$

as described in the original article.[8][9] Moreover, the first transformations hold for $y_i > 0$, and the second for $y_i > -\lambda_2$.[8]

Exponent here is a variable called lambda that varies over the range of -5 to 5 and in process of searching, we examine all values of lambda. finally we choose the optimal value (resulting in the best approximation to a normal distribution) for our variable.

For any distribution we can convert into Normal Distribution.

Box-Cox Transformation is a transformtion of non-normal dependent variables into a Normal shape. Normality is an important assumption for many statistical techniques, if our data isn't normal, applying Box-Cox means that we are able to run a broader number of tests.

This test only works for positive data. However Box and Cox did purpose a second formula that can be use for negative y-values.

$$x(\lambda) = \frac{(x + \lambda_2)^{\lambda_1} - 1)}{\lambda_1} \; for \; \lambda_1 \neq 0$$

$$x(\lambda) = \ln(x + \lambda_2) \; for \; \lambda_1 = 0$$

```
1. Maximum likelihood   -\
                          |---> internal working
2. Bayesian statistics  -/
```

For e.g: x<=0 we ues Yeo-Johnson

$$y_i^{(\lambda)} = \begin{cases} ((y_i+1)^\lambda - 1)/\lambda & \text{if } \lambda \neq 0, y \geq 0 \\ \ln(y_i + 1) & \text{if } \lambda = 0, y \geq 0 \\ -((-y_i+1)^{(2-\lambda)} - 1)/(2-\lambda) & \text{if } \lambda \neq 2, y < 0 \\ -\ln(-y_i+1) & \text{if } \lambda = 2, y < 0 \end{cases}$$

This transformation is somewhat of an adjustment to the Box-Cox transformation, by which we can apply it to negative numbers.

Wherever we feel that the data is not normally distributed and want to normalize the data, we use both these transformation techniques. Because at the end we are going to use hyperparameter tunning which will get us to know whether to use Box-Cox or Yeo-Johnson.

Let's work on an example
Taking Concrete dataset which is used to make Roadways.

Import necessary Libraries

In [33]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
```

In [34]:
```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import PowerTransformer
```

In [35]:
```
Concrete_dataset = pd.read_csv('concrete_dataset.csv')
```

In [36]:
```
Concrete_dataset
```

Out[36]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Strength
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.89
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.27
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.05
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.30
...	...	...	...	...	...	...	...	...	...
1025	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28	44.28
1026	322.2	0.0	115.6	196.0	10.4	817.9	813.4	28	31.18
1027	148.5	139.4	108.6	192.7	6.1	892.4	780.0	28	23.70
1028	159.1	186.7	0.0	175.6	11.3	989.6	788.9	28	32.77
1029	260.9	100.5	78.3	200.6	8.6	864.5	761.5	28	32.40

1030 rows × 9 columns

In [37]:
```
Concrete_dataset.isna().sum()
```

Out[37]:
```
Cement                0
Blast Furnace Slag    0
Fly Ash               0
Water                 0
Superplasticizer      0
Coarse Aggregate      0
Fine Aggregate        0
Age                   0
Strength              0
dtype: int64
```

As there are no missing values.

Are there many value 0's ?
We use describe() function which shows us min, max values.

In [38]:
```
Concrete_dataset.describe()
```

Out[38]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Strength
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000
mean	281.167864	73.895825	54.188350	181.567282	6.204660	972.918932	773.580485	45.662136	35.817961
std	104.506364	86.279342	63.997004	21.354219	5.973841	77.753954	80.175980	63.169912	16.705742
min	102.000000	0.000000	0.000000	121.800000	0.000000	801.000000	594.000000	1.000000	2.330000
25%	192.375000	0.000000	0.000000	164.900000	0.000000	932.000000	730.950000	7.000000	23.710000
50%	272.900000	22.000000	0.000000	185.000000	6.400000	968.000000	779.500000	28.000000	34.445000
75%	350.000000	142.950000	118.300000	192.000000	10.200000	1029.400000	824.000000	56.000000	46.135000
max	540.000000	359.400000	200.100000	247.000000	32.200000	1145.000000	992.600000	365.000000	82.600000

x -> all columns except 'Strength', which is our target variable
y -> target variable 'Strength'.

In [39]:
```
X = Concrete_dataset.drop(columns=['Strength'])
y = Concrete_dataset.iloc[:,-1]
```

In [40]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=42)
```

In [41]:
```
LR_ = LinearRegression()

LR_.fit(X_train,y_train)

y_pred = LR_.predict(X_test)

r2_score(y_test,y_pred)
```

Out[41]: 0.627553179231485

In [42]:
```
LR_ = LinearRegression()
np.mean(cross_val_score(LR_,X,y,scoring='r2'))
```

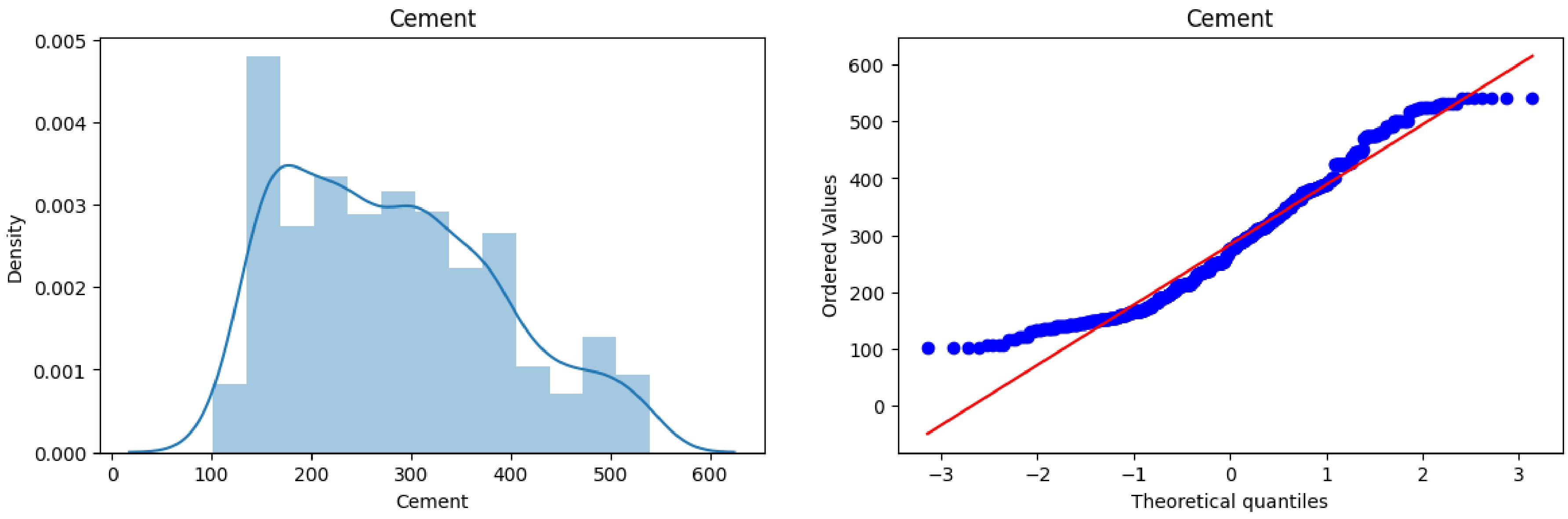
Out[42]: 0.46099404916628606

We just simply train test and split the data, fit LinearRegression() on train data, train predicted the result for test data. Then I calculated r2 score, accuracy.
Then cross validate the data and observed result is getting more bad.

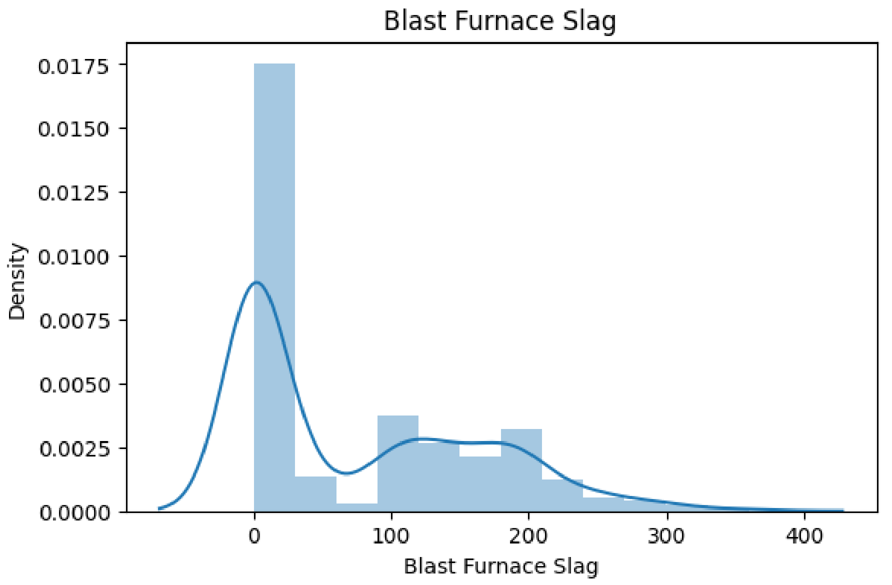
Let's plot distplot and QQ plot on X_ttrain each column data.
Plot the distplots without any transformation.

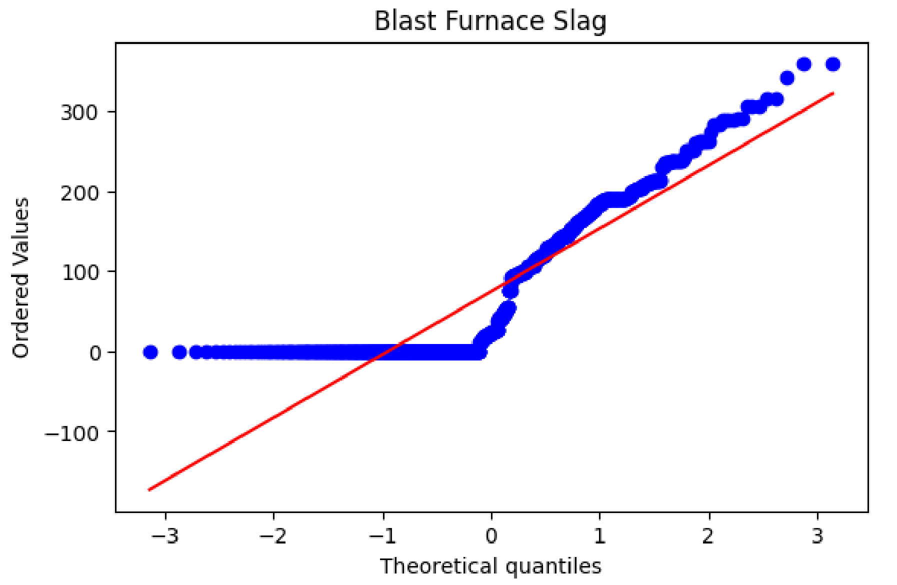
In [43]:
```
for col in X_train.columns:
    plt.figure(figsize=(14,4))
    plt.subplot(121)
    sns.distplot(X_train[col])
    plt.title(col)

    plt.subplot(122)
    stats.probplot(X_train[col], dist="norm", plot=plt)
    plt.title(col)

    plt.show()
```

```
<ipython-input-43-672c00931c94>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
```

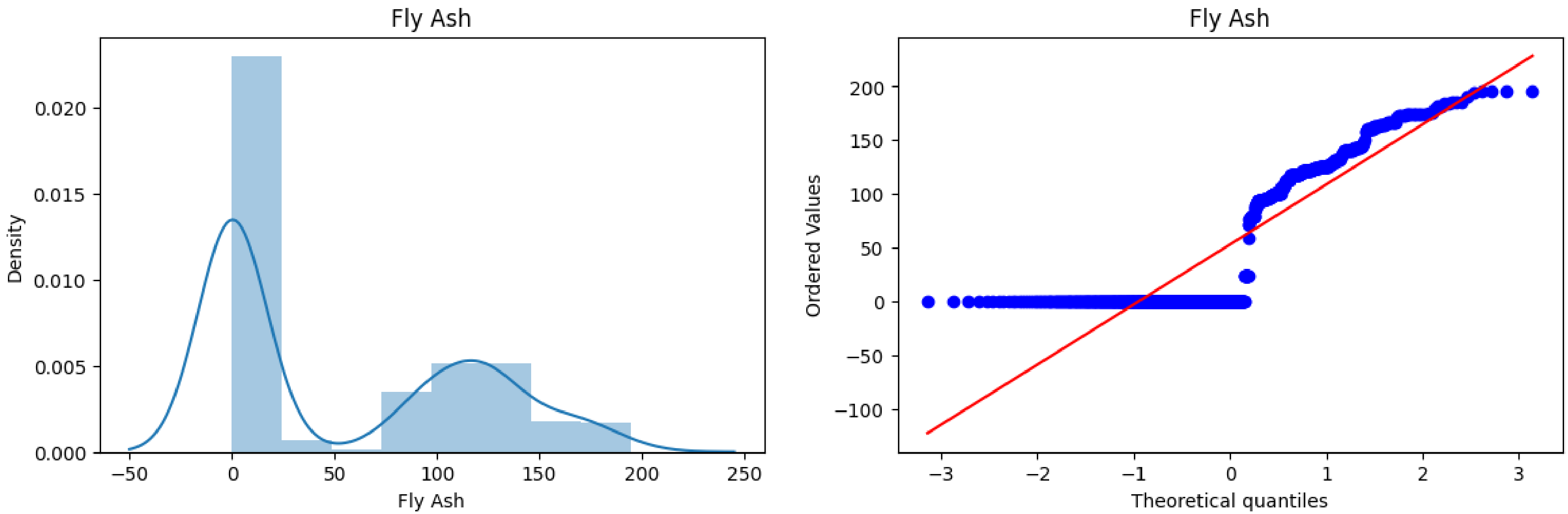



```
<ipython-input-43-672c00931c94>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
```

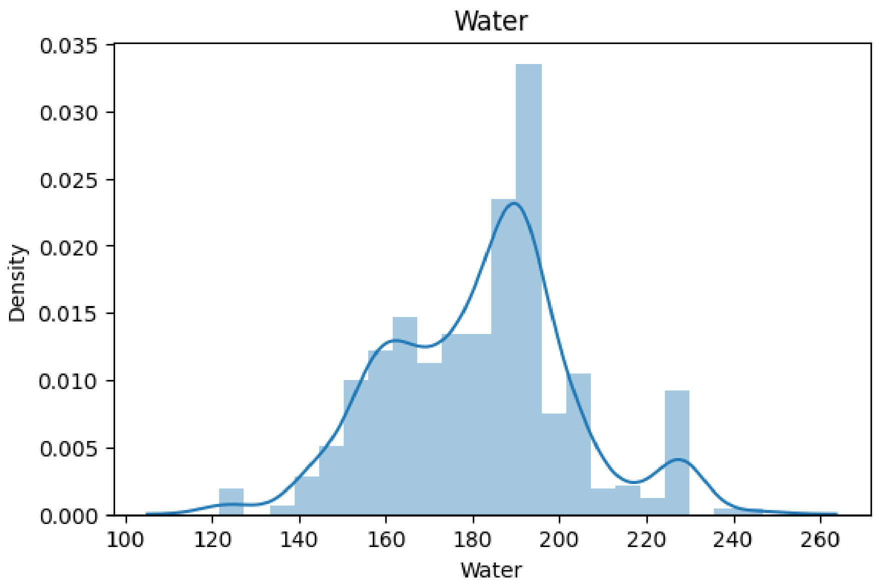


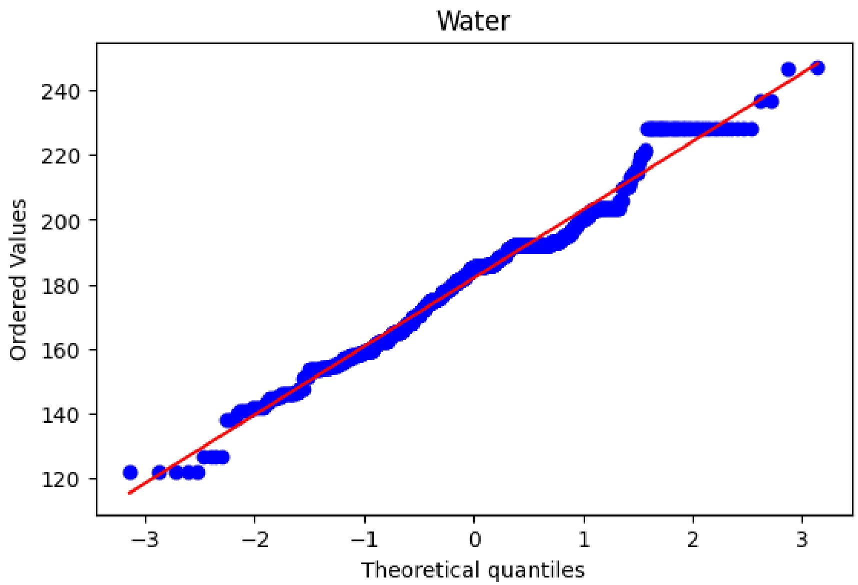



```
<ipython-input-43-672c00931c94>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
```

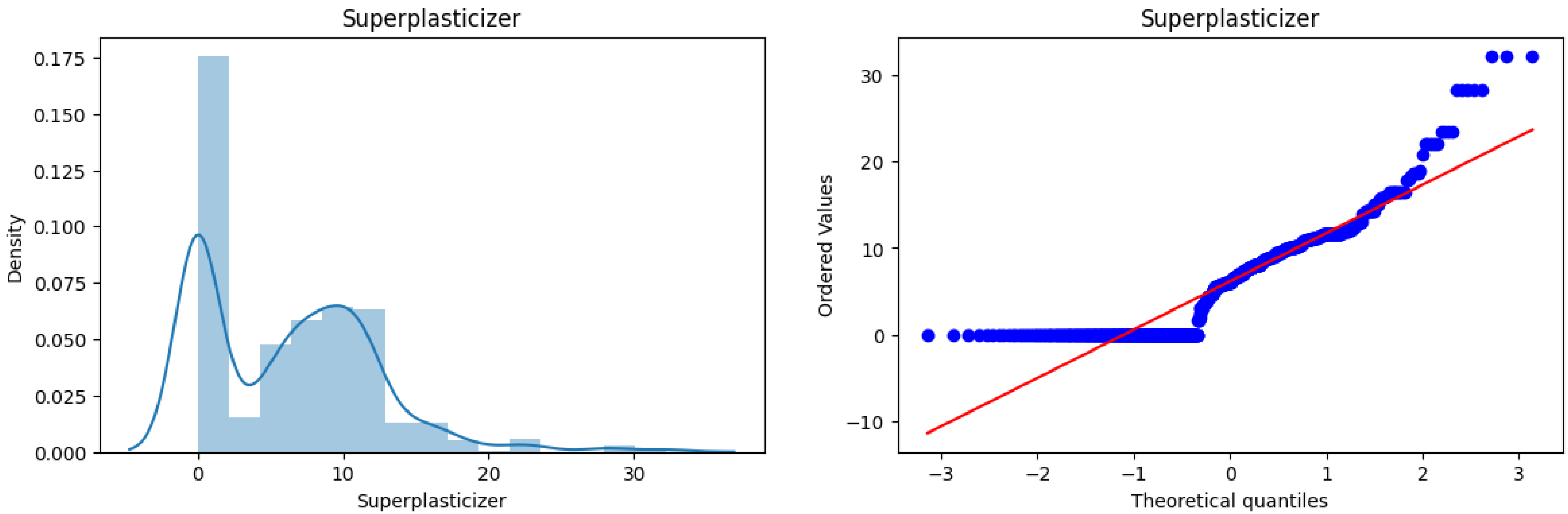



```
<ipython-input-43-672c00931c94>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
```

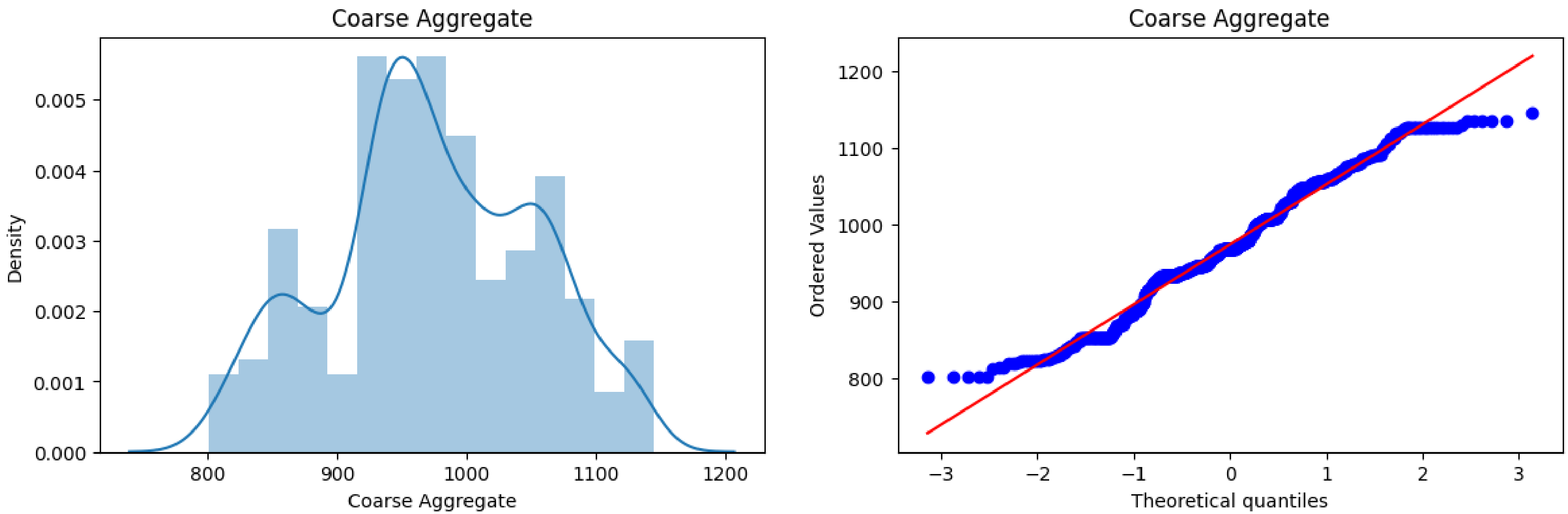




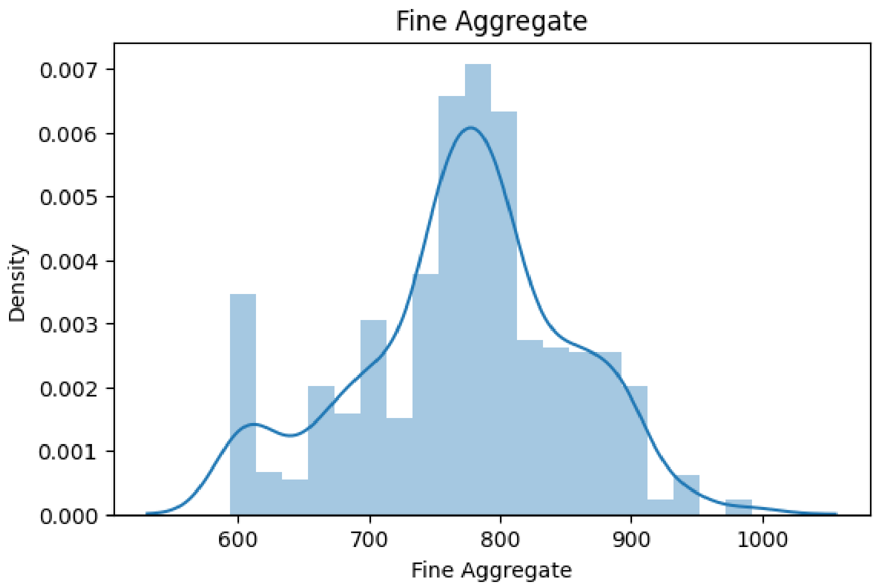

```
<ipython-input-43-672c00931c94>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
```

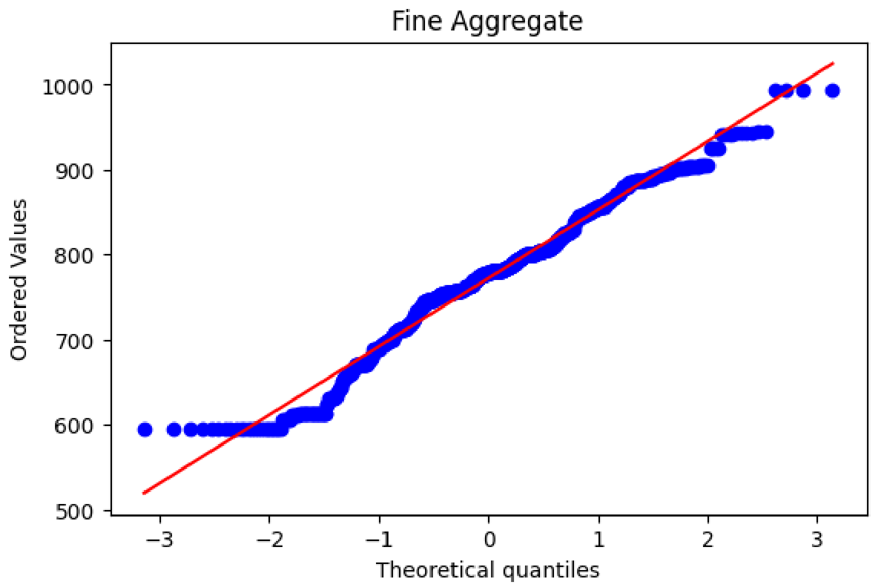



```
<ipython-input-43-672c00931c94>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
```

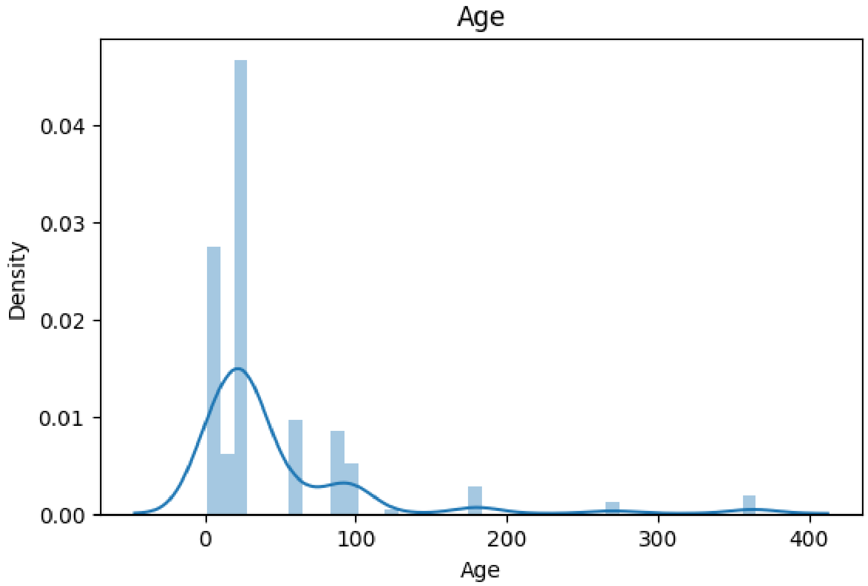


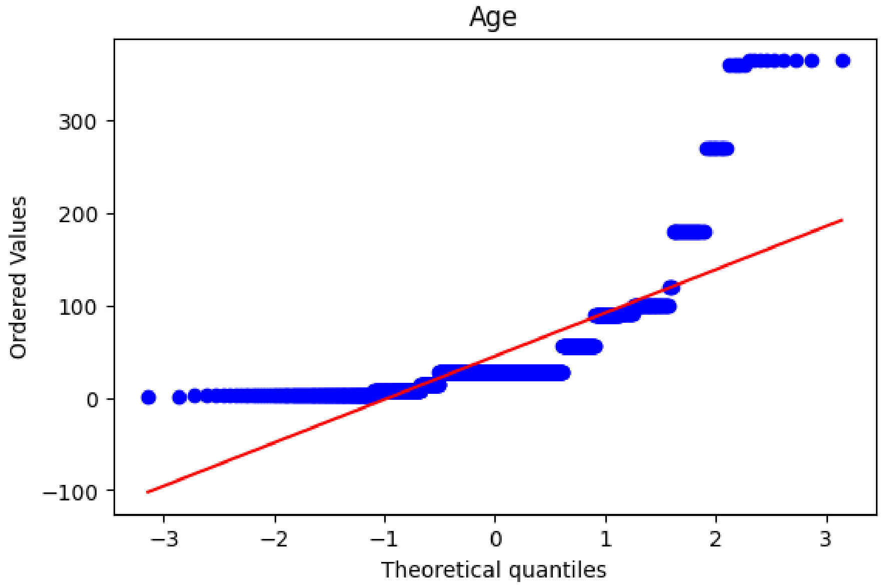

```
<ipython-input-43-672c00931c94>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
```






```
<ipython-input-43-672c00931c94>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
```





Cement -> data points are lying near on and around the line just at some places it is deflecting.
Blast Furnace Slag -> It is right skewed.
Fly Ash -> Bi model.
Water -> nearly mostly lying near around and on the line.
Superplasticizer -> Right skewed bi-model.
Coarse Aggregate -> Good enough.
Fine Aggregate -> Somewhat Good.
Age -> Worst.

We didn't applied Scaling here,
We are going to scale without transformation and with transformation. So scaling impact will be equal/same.
If we would have implemented Scaling, our result would has definitely improved.

Now we will be applying Box-Cox transformer.

In [44]:
```
PT = PowerTransformer(method='box-cox')
```

In PowerTransformer, by default the data is Standardized so we need not to scale, internally StandardScaler is applied)

Box-Cox < Yeo-Johnson

Apply Box-Cox transformation on X_train and X_test

#BOX-COX

In [45]:
```
X_train_transformed = PT.fit_transform(X_train+0.000001)
X_test_transformed = PT.transform(X_test+0.000001)
```

In [46]:
```
pd.DataFrame({'cols':X_train.columns,'box_cox_lambdas':PT.lambdas_})
```

Out[46]:

	cols	box_cox_lambdas
0	Cement	0.177025
1	Blast Furnace Slag	0.025093
2	Fly Ash	-0.038970
3	Water	0.772682
4	Superplasticizer	0.098811
5	Coarse Aggregate	1.129813
6	Fine Aggregate	1.782019
7	Age	0.066631

When we apply transformation on independent columns, we calculate lamda value for each column.
So after transnformation **.lambdas_** gives us all lamda values for each column.

In [47]:
```
LR = LinearRegression()
LR.fit(X_train_transformed,y_train)
```

Out[47]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [48]:
```
y_pred2 = LR.predict(X_test_transformed)
```

In [49]:
```
r2_score(y_test,y_pred2)
```

Out[49]:
```
0.8047825006181188
```

The result we can see is very good than before which was around 63% and now its 80.5%

Fitted LinearRegression of X_train_transformed data and predicted results for X_test_transformed data.
Then I calculated r2 score which quietly seems better than before.

Then I fit transformation (Box-Cox) on X data then applied Linear Regression on it and cross validate it, whose accuracy comes out to 67%.
(Again not so good but improved from before)

In [50]:
```
X
```

Out[50]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360
...	...	...	...	...	...	...	...	...
1025	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28
1026	322.2	0.0	115.6	196.0	10.4	817.9	813.4	28
1027	148.5	139.4	108.6	192.7	6.1	892.4	780.0	28
1028	159.1	186.7	0.0	175.6	11.3	989.6	788.9	28
1029	260.9	100.5	78.3	200.6	8.6	864.5	761.5	28

1030 rows × 8 columns

In [51]:
```
P_T = PowerTransformer(method='box-cox')
X_transformed = P_T.fit_transform(X+0.0000001)
```

Here I applied Box-Cox Power Transformation method on X data and stored the result in X_transformed. (+ 0.0000001 for in case if value is 0 so Box-Cox won't be giving result for 0's so for that I added 0.0000001 which won't effect our result because it is very minimal) So transformation will be for X+0.0000001

Let's just cross-validate it...

In [52]:
```
Linear_Regression = LinearRegression()
np.mean(cross_val_score(Linear_Regression, X_transformed, y, scoring = 'r2'))
```

Out[52]:
```
0.6658537942219863
```

So after cross-validation our value comes out to around 66.6% = 67%

Let's see the difference before and after Box-Cox Transformation

In [53]:
```
X_train_transformed = pd.DataFrame(X_train_transformed,columns=X_train.columns)
```

Firstly I will be converting my array to dataframe

In [54]:
```
X_train_transformed
```

Out[54]:

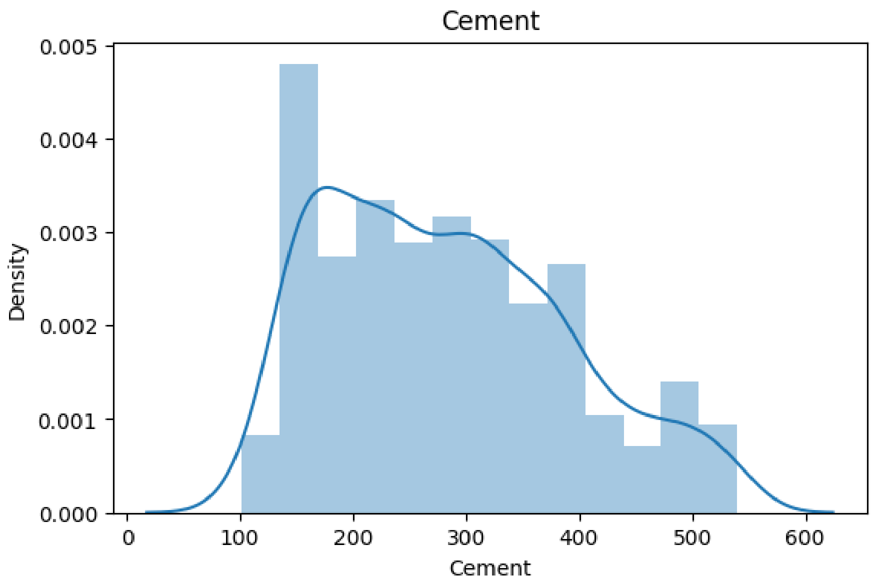
	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age
0	-1.279751	0.956151	1.128045	-0.301920	0.898506	-0.262500	-0.677568	0.106010
1	1.244706	0.701994	1.137577	-0.165055	0.749235	-1.912728	-0.314062	0.106010
2	0.079842	-1.085667	1.131407	-1.047291	0.788486	1.018407	0.025957	-1.675970
3	-0.145641	0.898125	1.096514	0.581224	0.733792	-1.750779	0.583253	0.106010
4	-1.131044	0.787203	1.133149	-1.105297	0.811152	1.368575	0.262623	-1.675970
...	...	...	...	...	...	...	...	...
819	0.183601	0.997079	-0.887212	-1.769923	0.820684	0.390563	0.358776	-1.675970
820	-0.198733	-1.085667	1.133622	-1.839129	0.838861	1.445973	0.322797	-0.475625
821	-0.844517	-1.085667	1.133681	-0.706089	0.788486	1.345367	0.297157	1.246733
822	1.565624	0.925496	-0.887212	-0.019311	0.761011	-1.537255	0.076211	0.106010
823	0.426556	-1.085667	1.126106	-0.543871	0.791094	-0.620834	0.095108	0.106010

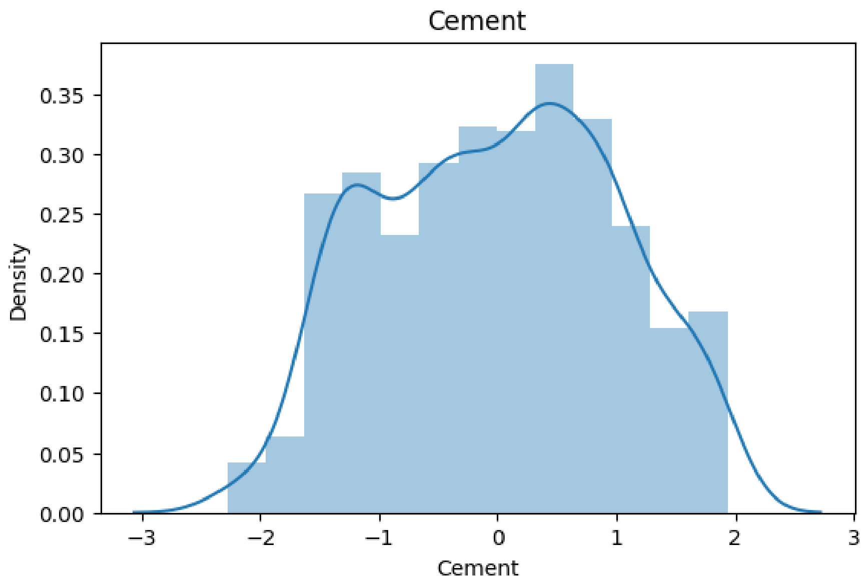
824 rows × 8 columns

Now lets plot and see the difference.

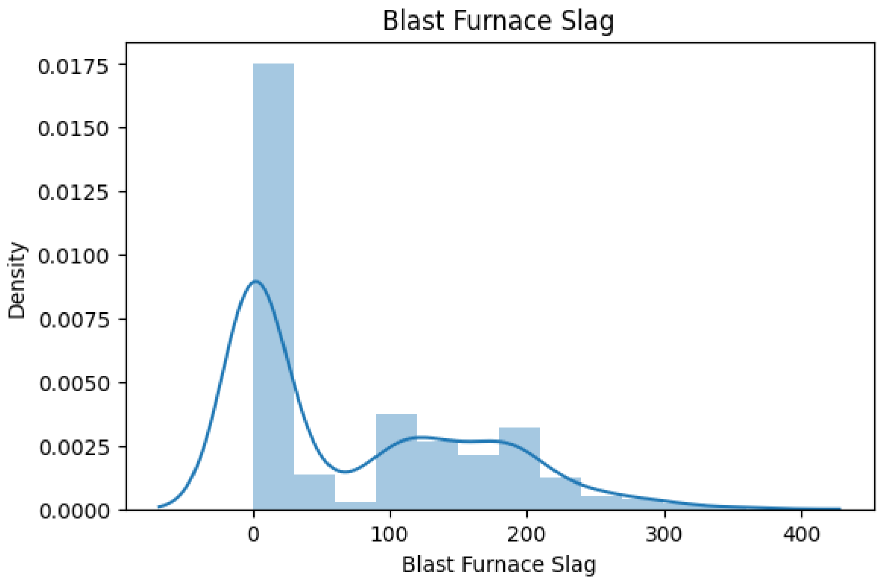
In [55]:
```
for col in X_train_transformed.columns:
    plt.figure(figsize=(14,4))
    plt.subplot(121)
    sns.distplot(X_train[col])
    plt.title(col)

    plt.subplot(122)
    sns.distplot(X_train_transformed[col])
    plt.title(col)

    plt.show()
```

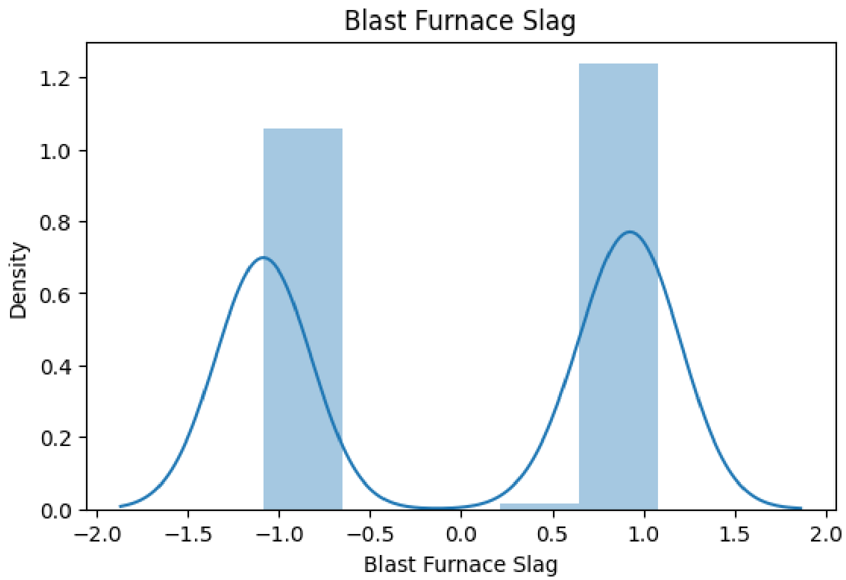
```
<ipython-input-55-5f5798d6ff69>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-55-5f5798d6ff69>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed[col])
```


Cement
0.005
0.004
0.003
0.002
0.001
0.000
Density
0
100
200
300
400
500
600
Cement

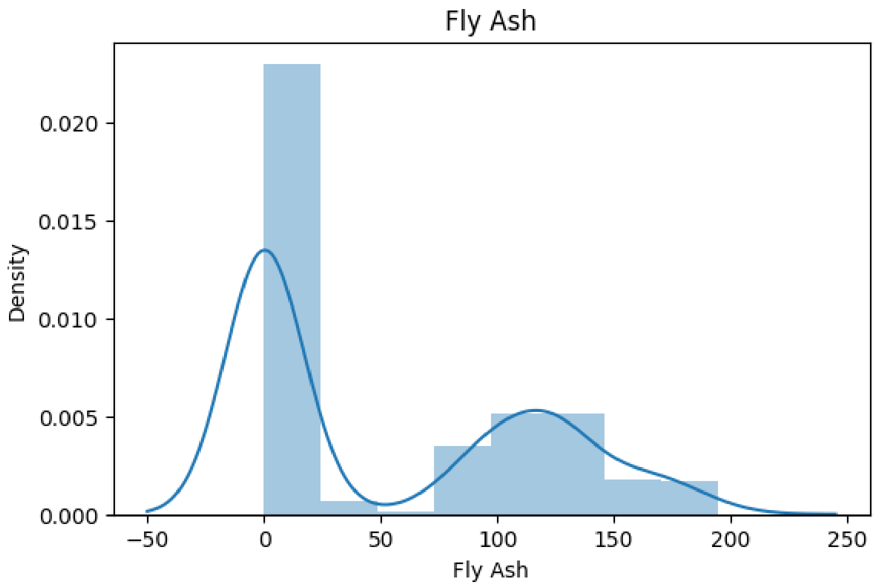

Cement
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Density
−3
−2
−1
0
1
2
3
Cement

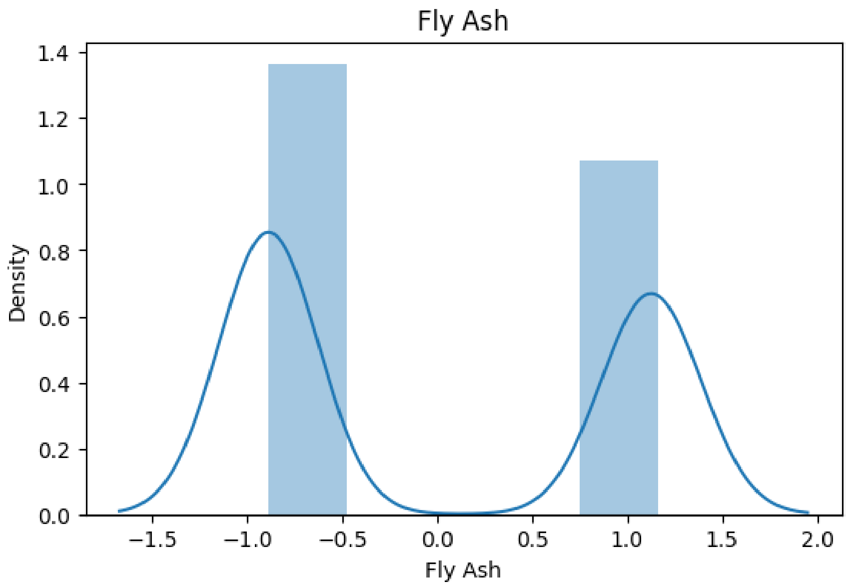
```
<ipython-input-55-5f5798d6ff69>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-55-5f5798d6ff69>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed[col])
```


Blast Furnace Slag
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
Density
0
100
200
300
400
Blast Furnace Slag

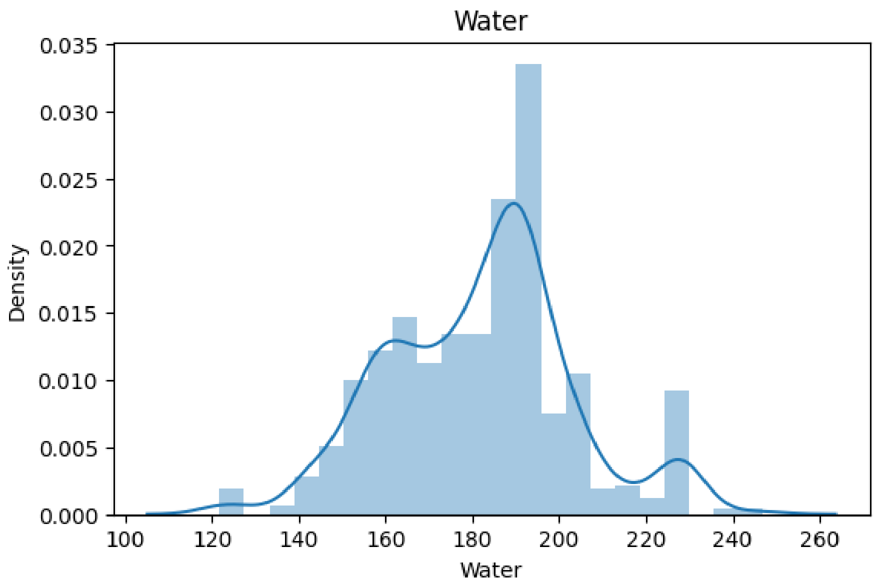

Blast Furnace Slag
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Density
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Blast Furnace Slag

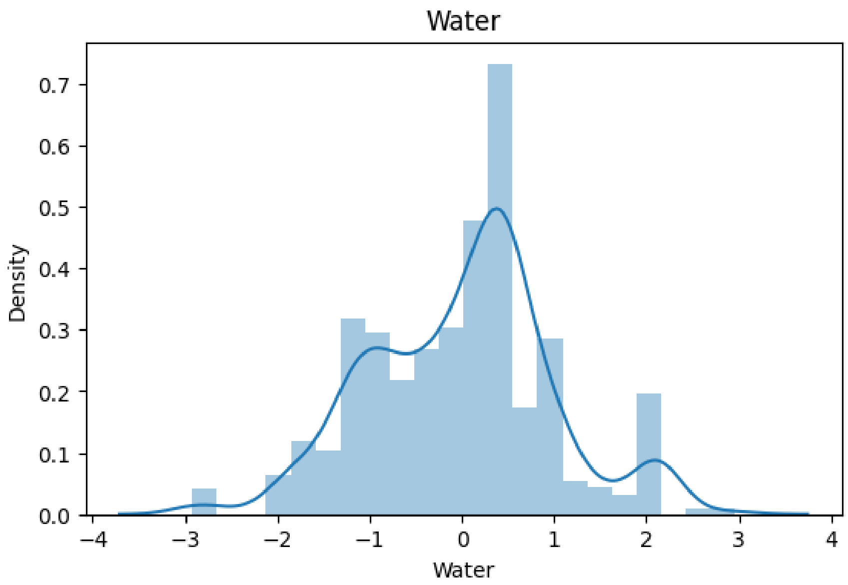
```
<ipython-input-55-5f5798d6ff69>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-55-5f5798d6ff69>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed[col])
```


Fly Ash
Density
Fly Ash

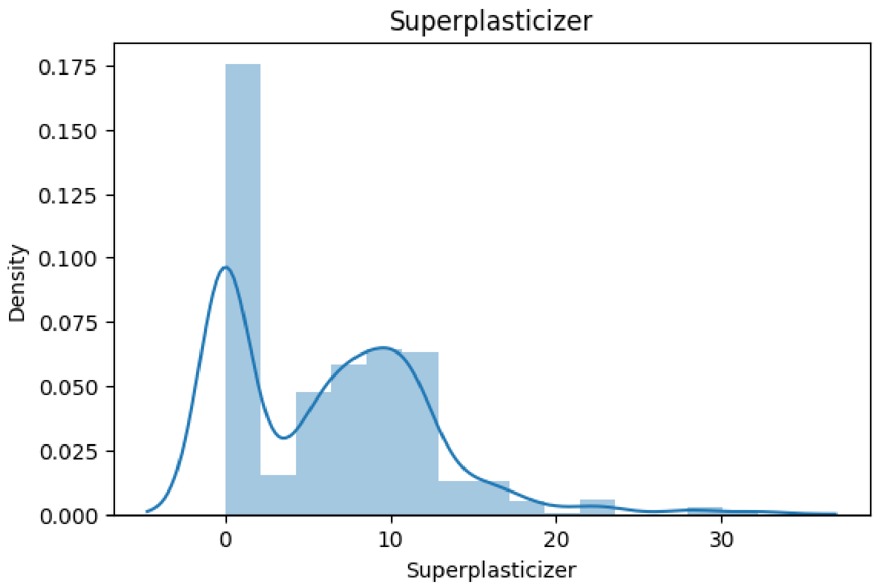

Fly Ash
Density
Fly Ash

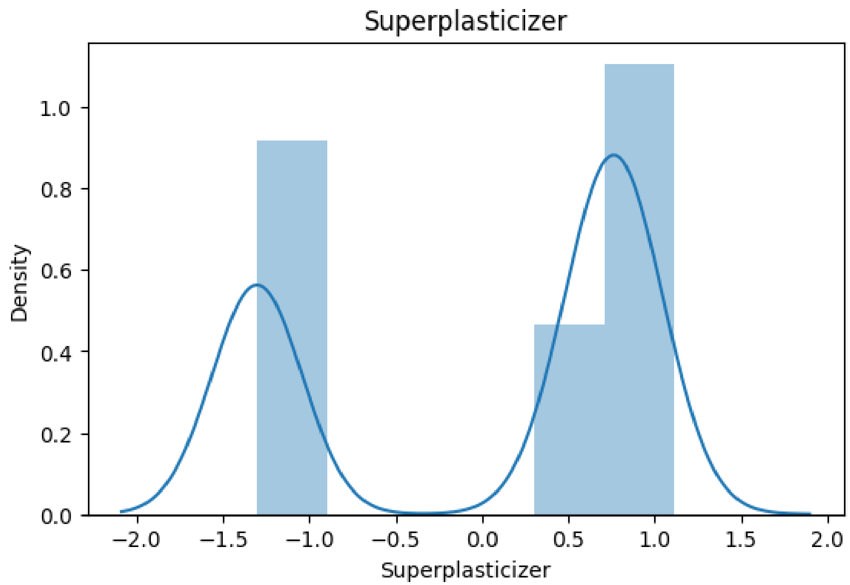
```
<ipython-input-55-5f5798d6ff69>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-55-5f5798d6ff69>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed[col])
```


Water
Density
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
100
120
140
160
180
200
220
240
260
Water

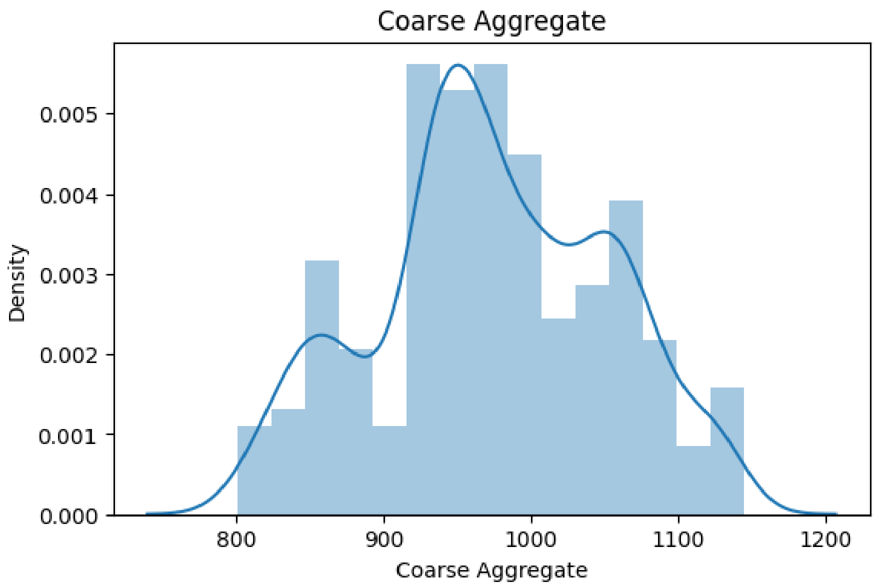

Water
Density
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−4
−3
−2
−1
0
1
2
3
4
Water

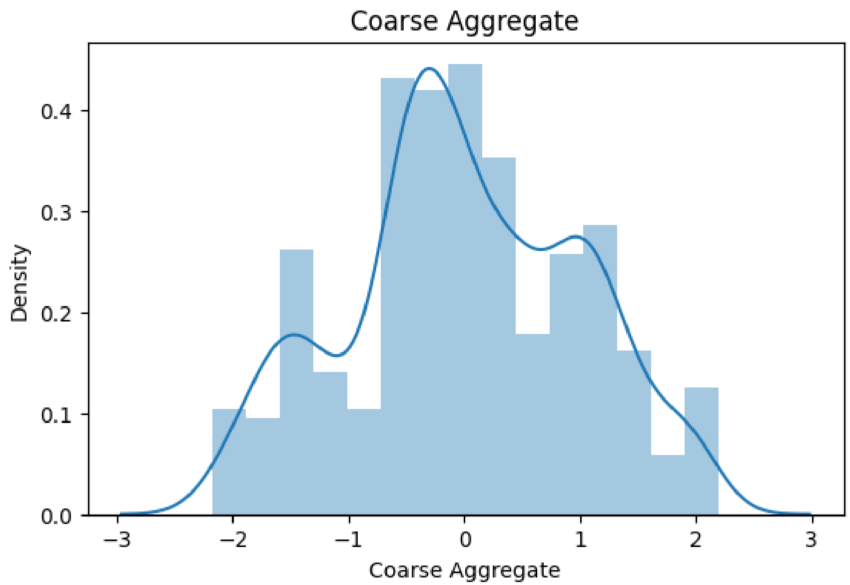
```
<ipython-input-55-5f5798d6ff69>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-55-5f5798d6ff69>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed[col])
```


Superplasticizer
Density
Superplasticizer
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0
10
20
30

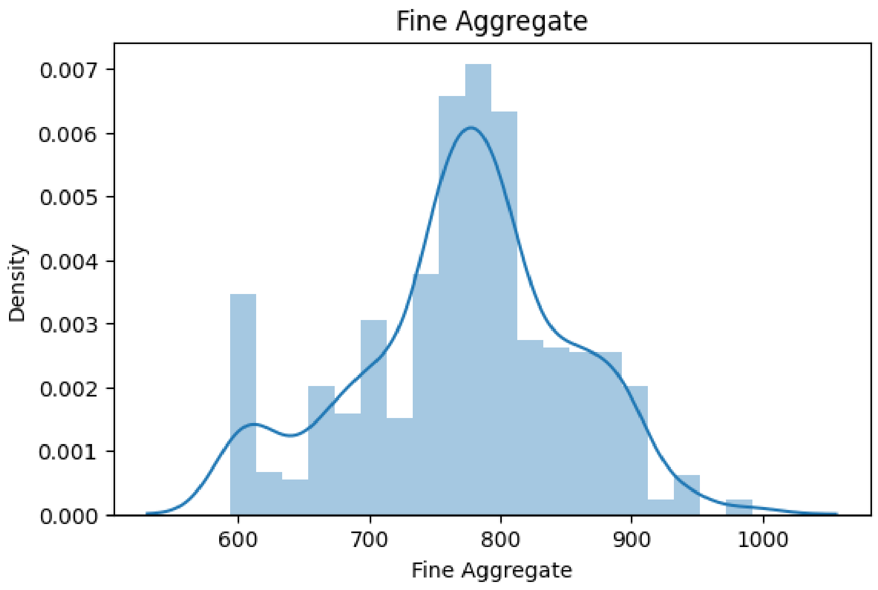

Superplasticizer
Density
Superplasticizer
0.0
0.2
0.4
0.6
0.8
1.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

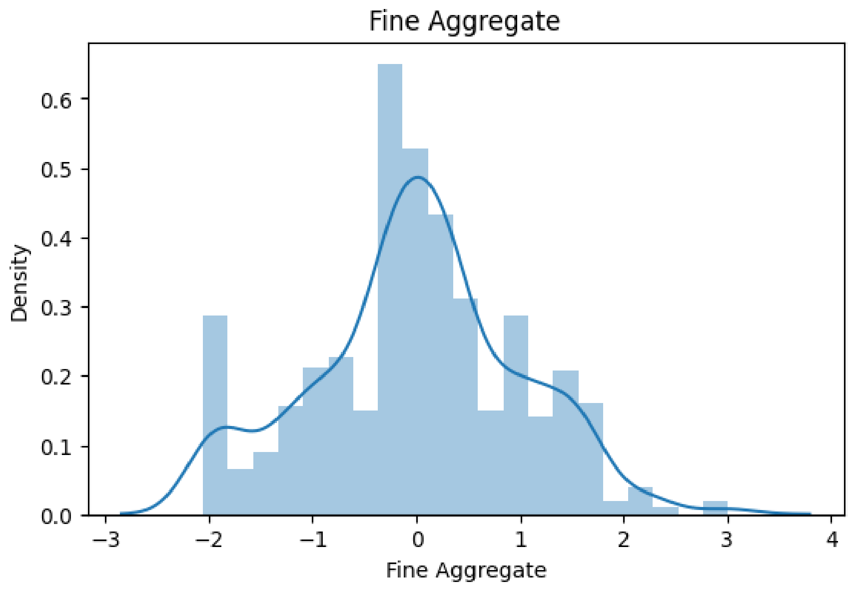
```
<ipython-input-55-5f5798d6ff69>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-55-5f5798d6ff69>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed[col])
```


Coarse Aggregate
Density
0.000
0.001
0.002
0.003
0.004
0.005
800
900
1000
1100
1200
Coarse Aggregate

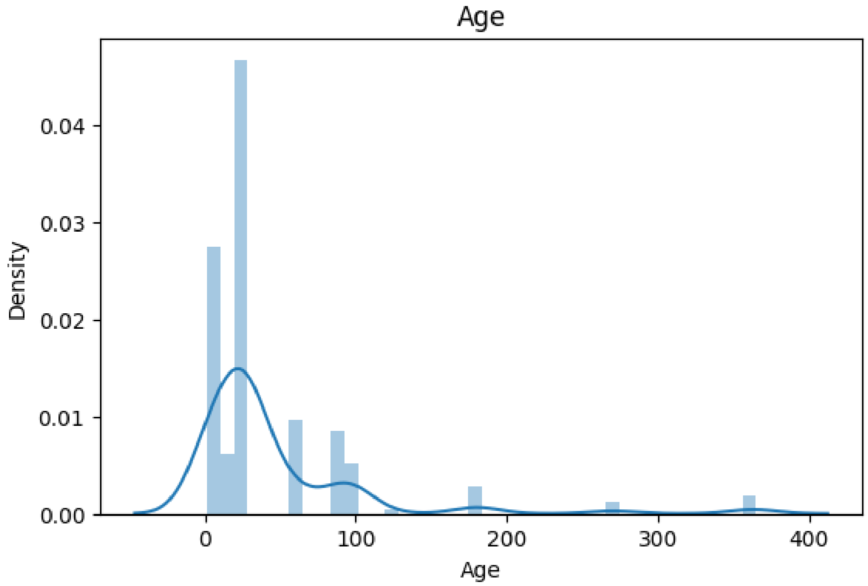

Coarse Aggregate
Density
0.0
0.1
0.2
0.3
0.4
−3
−2
−1
0
1
2
3
Coarse Aggregate

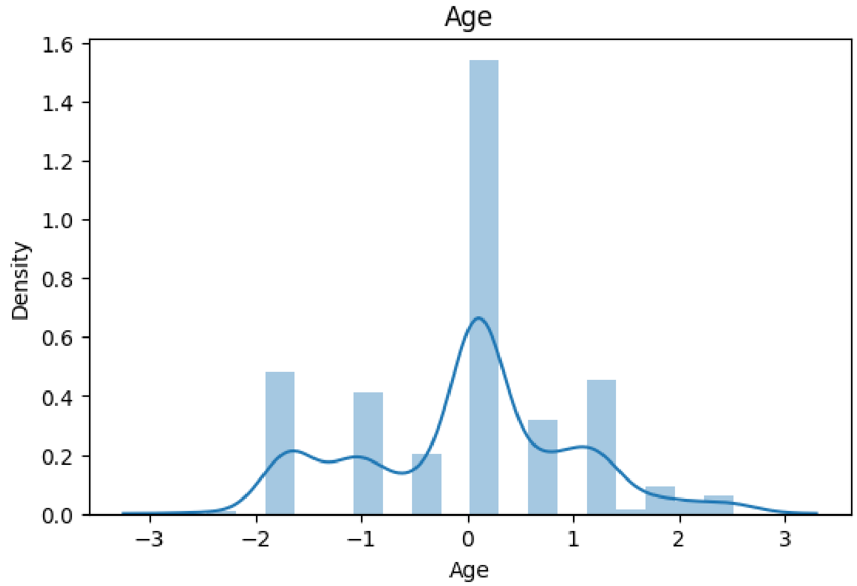
```
<ipython-input-55-5f5798d6ff69>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-55-5f5798d6ff69>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed[col])
```


Fine Aggregate
Density
0.000
0.001
0.002
0.003
0.004
0.005
0.006
0.007
600
700
800
900
1000
Fine Aggregate


Fine Aggregate
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
−3
−2
−1
0
1
2
3
4
Fine Aggregate

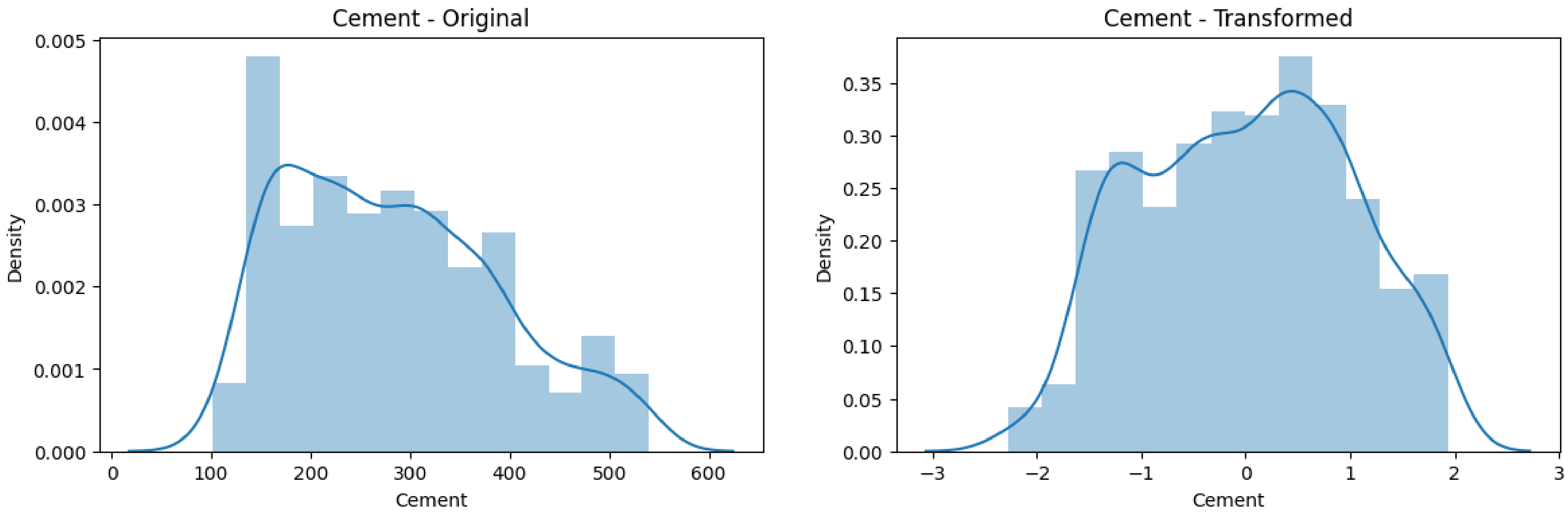
```
<ipython-input-55-5f5798d6ff69>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-55-5f5798d6ff69>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed[col])
```


Age
Density
0.00
0.01
0.02
0.03
0.04
0
100
200
300
400
Age

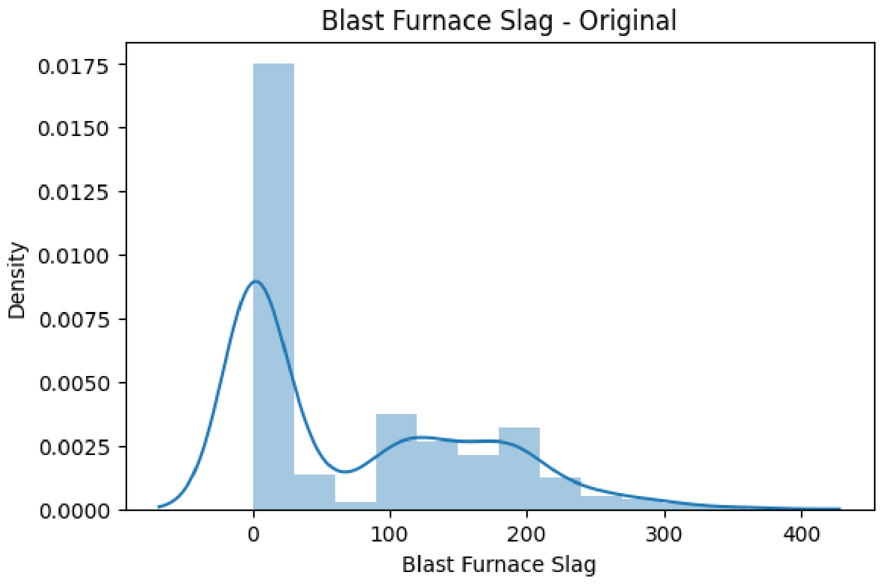

Age
Density
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
−3
−2
−1
0
1
2
3
Age

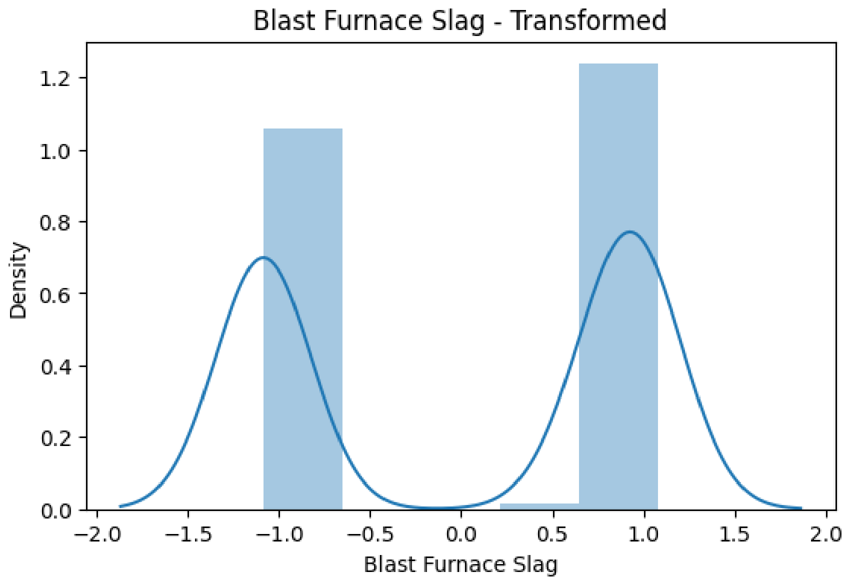
In [56]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

for col in X_train_transformed.columns:
    plt.figure(figsize=(14, 4))
    plt.subplot(121)
    sns.distplot(X_train[col])
    plt.title(f"{col} - Original")

    plt.subplot(122)
    sns.distplot(X_train_transformed[col])
    plt.title(f"{col} - Transformed")

    plt.show()
```

```
<ipython-input-56-79c5de2bae39>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-56-79c5de2bae39>:12: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed[col])
```

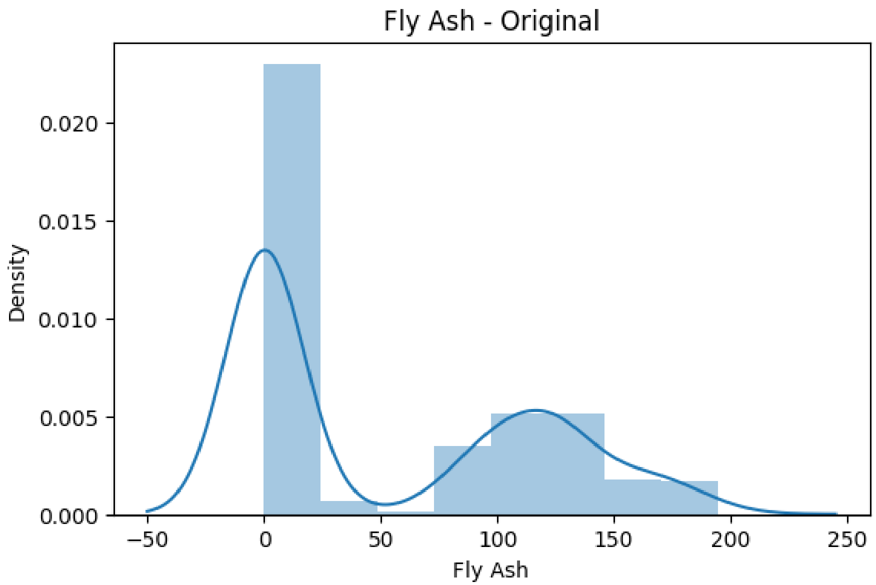

Cement - Original
Density
0.005
0.004
0.003
0.002
0.001
0.000
0
100
200
300
400
500
600
Cement
Cement - Transformed
Density
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−3
−2
−1
0
1
2
3
Cement

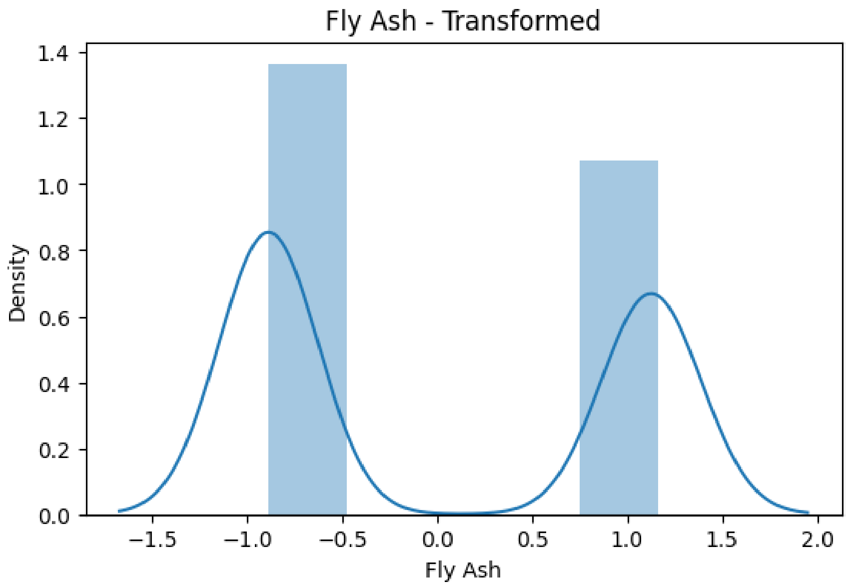
```
<ipython-input-56-79c5de2bae39>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-56-79c5de2bae39>:12: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed[col])
```


Blast Furnace Slag - Original
Density
Blast Furnace Slag
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0
100
200
300
400

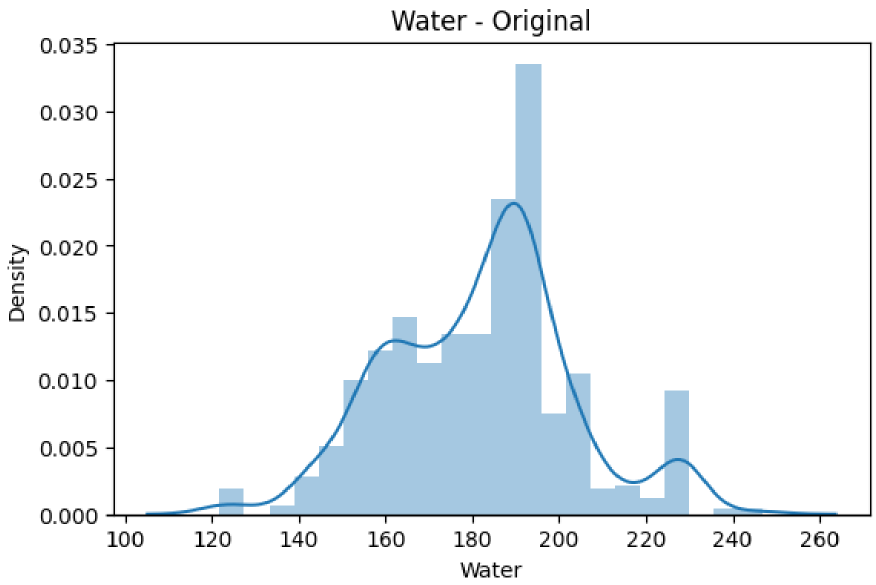

Blast Furnace Slag - Transformed
Density
Blast Furnace Slag
0.0
0.2
0.4
0.6
0.8
1.0
1.2
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

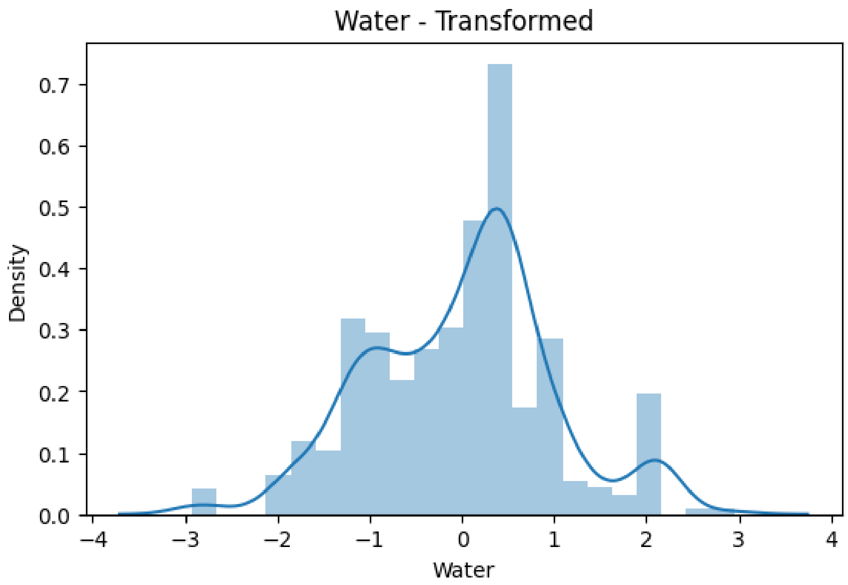
```
<ipython-input-56-79c5de2bae39>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-56-79c5de2bae39>:12: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed[col])
```


Fly Ash - Original
Density
0.020
0.015
0.010
0.005
0.000
−50
0
50
100
150
200
250
Fly Ash

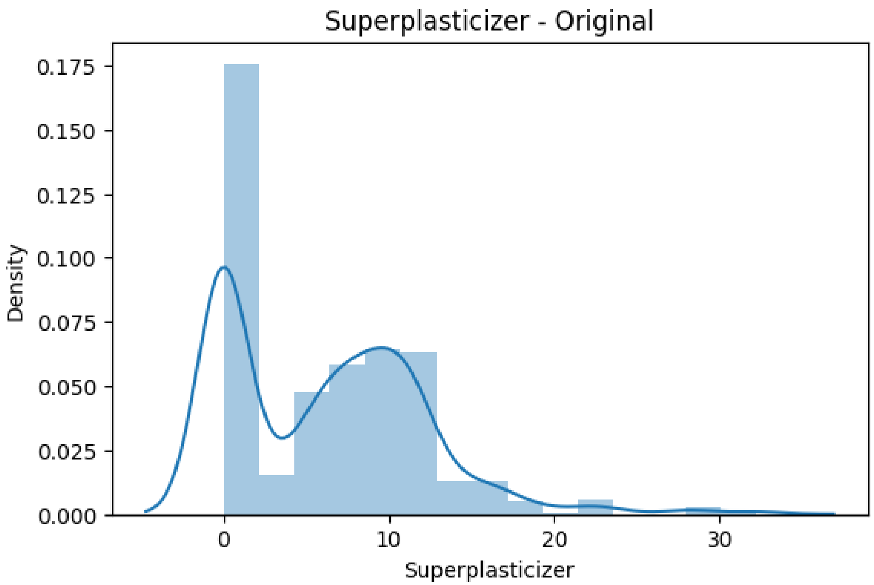

Fly Ash - Transformed
Density
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Fly Ash

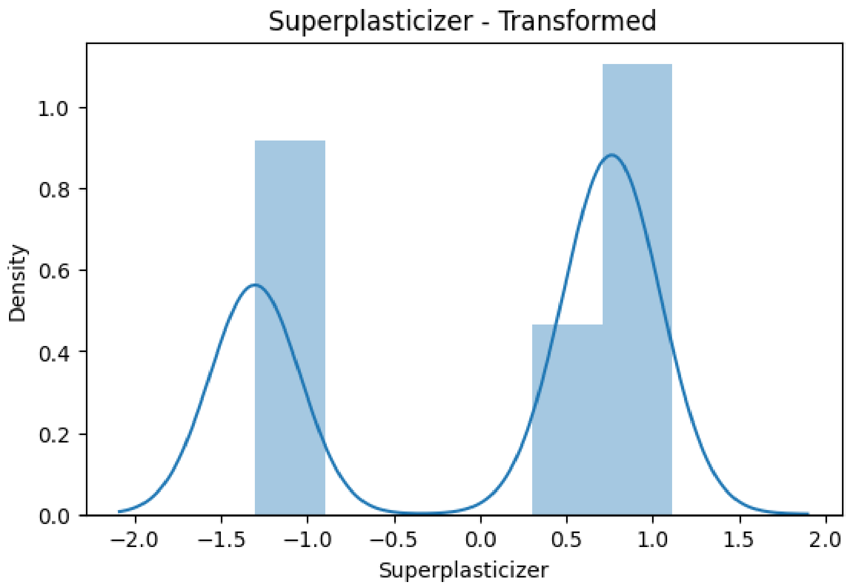
```
<ipython-input-56-79c5de2bae39>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-56-79c5de2bae39>:12: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed[col])
```


Water - Original
Density
Water
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
100
120
140
160
180
200
220
240
260

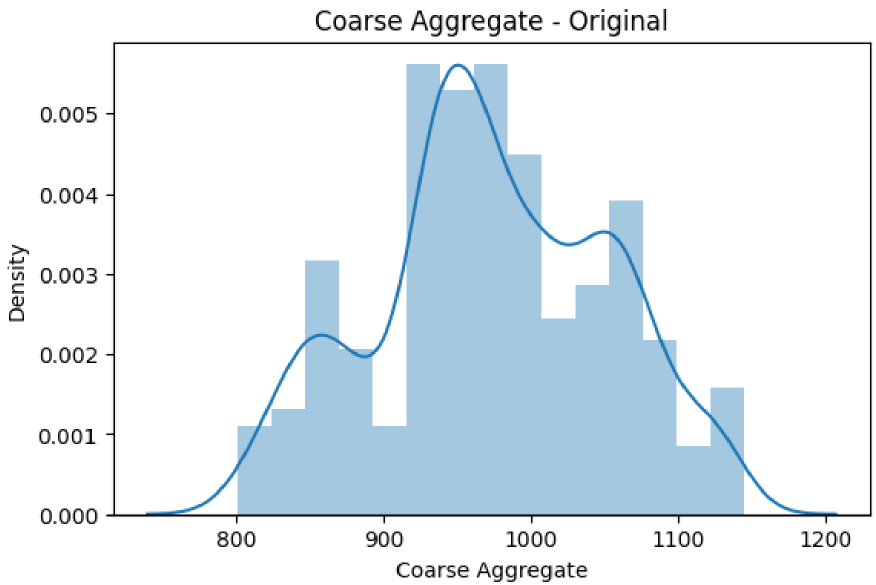

Water - Transformed
Density
Water
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−4
−3
−2
−1
0
1
2
3
4

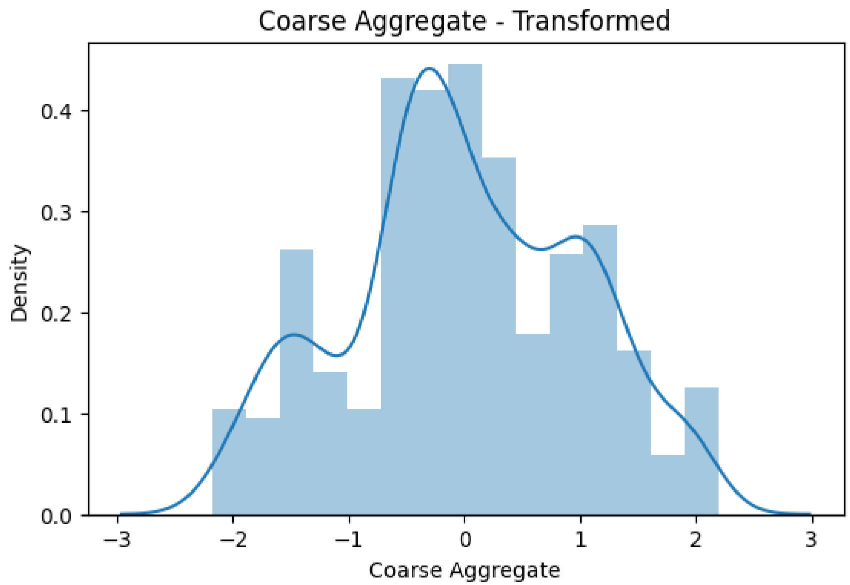
```
<ipython-input-56-79c5de2bae39>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-56-79c5de2bae39>:12: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed[col])
```


Superplasticizer - Original
Density
Superplasticizer
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0
10
20
30

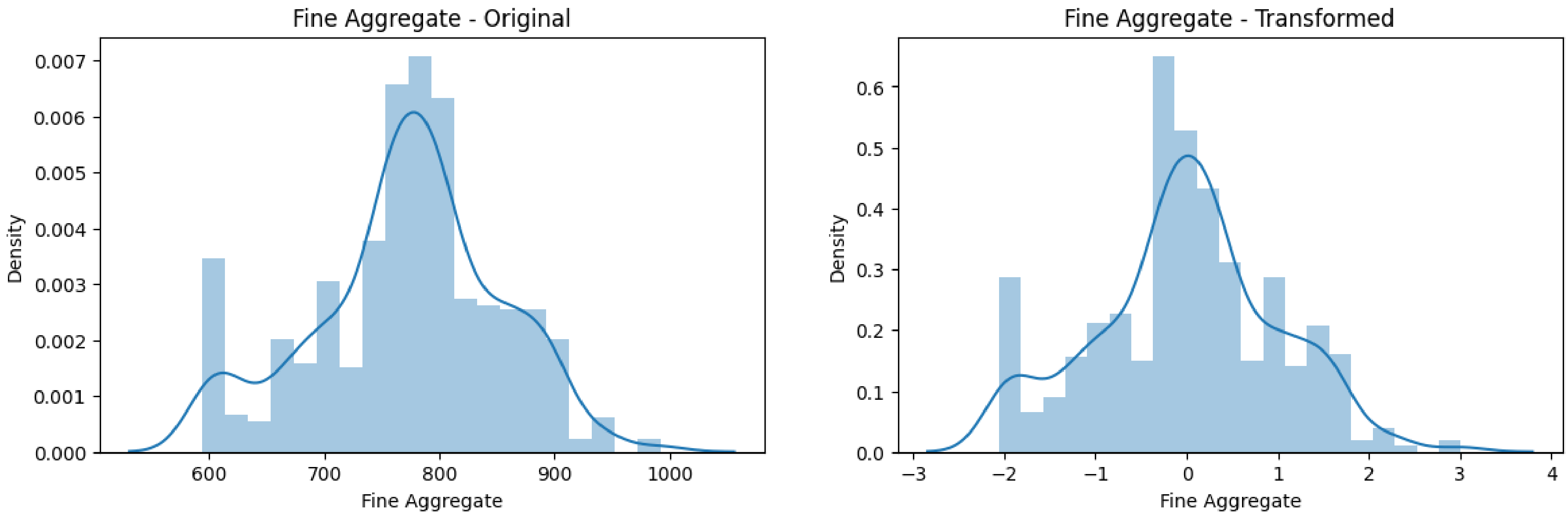

Superplasticizer - Transformed
Density
Superplasticizer
0.0
0.2
0.4
0.6
0.8
1.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

```
<ipython-input-56-79c5de2bae39>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-56-79c5de2bae39>:12: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(X_train_transformed[col])
```

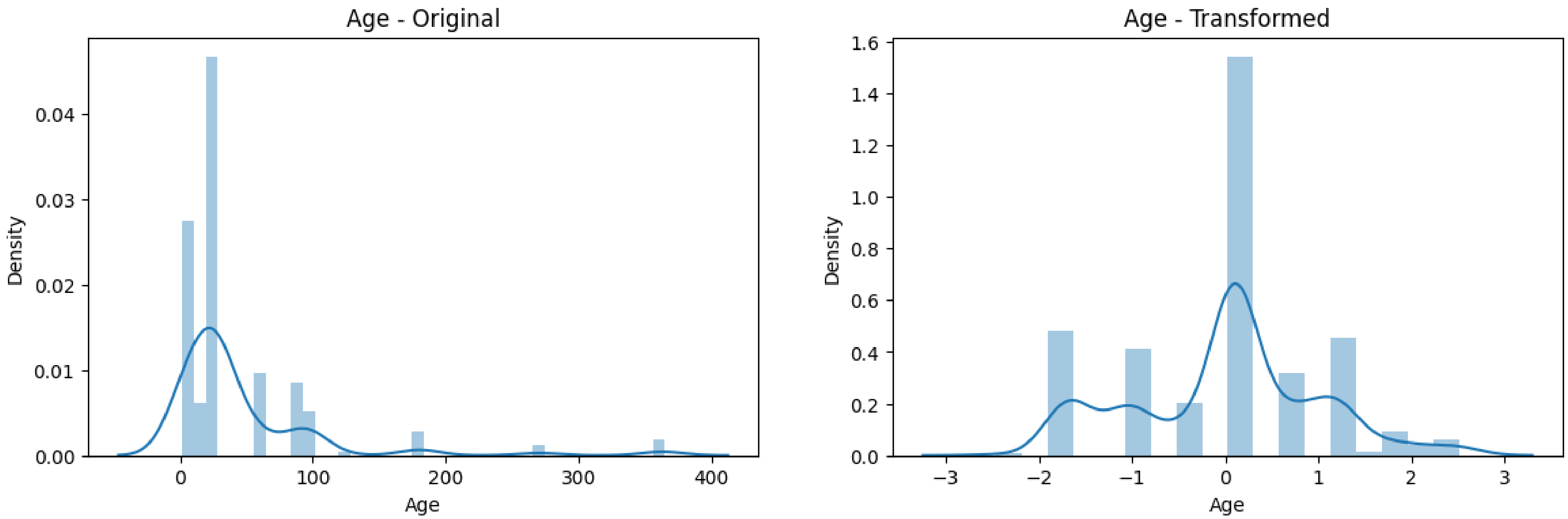

Coarse Aggregate - Original
Density
0.000
0.001
0.002
0.003
0.004
0.005
800
900
1000
1100
1200
Coarse Aggregate


Coarse Aggregate - Transformed
Density
0.0
0.1
0.2
0.3
0.4
−3
−2
−1
0
1
2
3
Coarse Aggregate

```
<ipython-input-56-79c5de2bae39>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-56-79c5de2bae39>:12: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(X_train_transformed[col])
```


Fine Aggregate - Original
Density
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
600
700
800
900
1000
Fine Aggregate
Fine Aggregate - Transformed
Density
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−3
−2
−1
0
1
2
3
4
Fine Aggregate

```
<ipython-input-56-79c5de2bae39>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-56-79c5de2bae39>:12: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed[col])
```



Cement -> slight transformation, the data which was right skewed transformed to normal slightly which is good.
Blast Furnance -> Data was right skewed, after transformation it didn't get properly Normally distributed but the bi-models which we can see are normally distributed.
Same goes with **Fly Ash** and **Superplasticizer**.
Water, Coarse Aggregate and **Fine Aggregate** remains unchanged.
Age -> Right skewed data to standardize and results to normally distributed.

#YEO-JOHNSON

In [57]:
```
Power_Transformer = PowerTransformer()
```

I didn't provide any value in PowerTransformer(), so by default value is Yeo-Johnson.

In [58]:
```
X_train_transformed_2 = Power_Transformer.fit_transform(X_train)
X_test_transformed_2 = Power_Transformer.transform(X_test)
```

Transformed X_train data and stored in X_train_transformed_2.

In [59]:
```
Linear_Regression_ = LinearRegression()
Linear_Regression_.fit(X_train_transformed_2,y_train)
```

Out[59]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Applied LinearRegression model on X_train_transfomed_2 and y_train

In [60]:
```
y_pred_3 = Linear_Regression_.predict(X_test_transformed_2)
```

Predicted X_train_transformed_2.

In [61]:
```
print(r2_score(y_test,y_pred_3))
```

```
0.8161906512004999
```

In [62]:
```
pd.DataFrame({'cols':X_train.columns,'Yeo_Johnson_lambdas':Power_Transformer.lambdas_})
```

Out[62]:

	cols	Yeo_Johnson_lambdas
0	Cement	0.174348
1	Blast Furnace Slag	0.015715
2	Fly Ash	-0.161447
3	Water	0.771307
4	Superplasticizer	0.253935
5	Coarse Aggregate	1.130050
6	Fine Aggregate	1.783100
7	Age	0.019885

Then I calculated the accuracy of Actual and Predicted values, converted it into DataFrame and calculated exponent through **.lambdas_**

We can compare the .lambdas values for Box-Cox and Yeo-Johnson, there is slight difference.

Let's cross validate

In [63]:
```
P_T_2 = PowerTransformer()
X_transformed_2 = P_T_2.fit_transform(X)
```

In [64]:
```
L_R_2 = LinearRegression()
np.mean(cross_val_score(L_R_2,X_transformed_2,y,scoring='r2'))
```

Out[64]:
```
0.6834625141500866
```

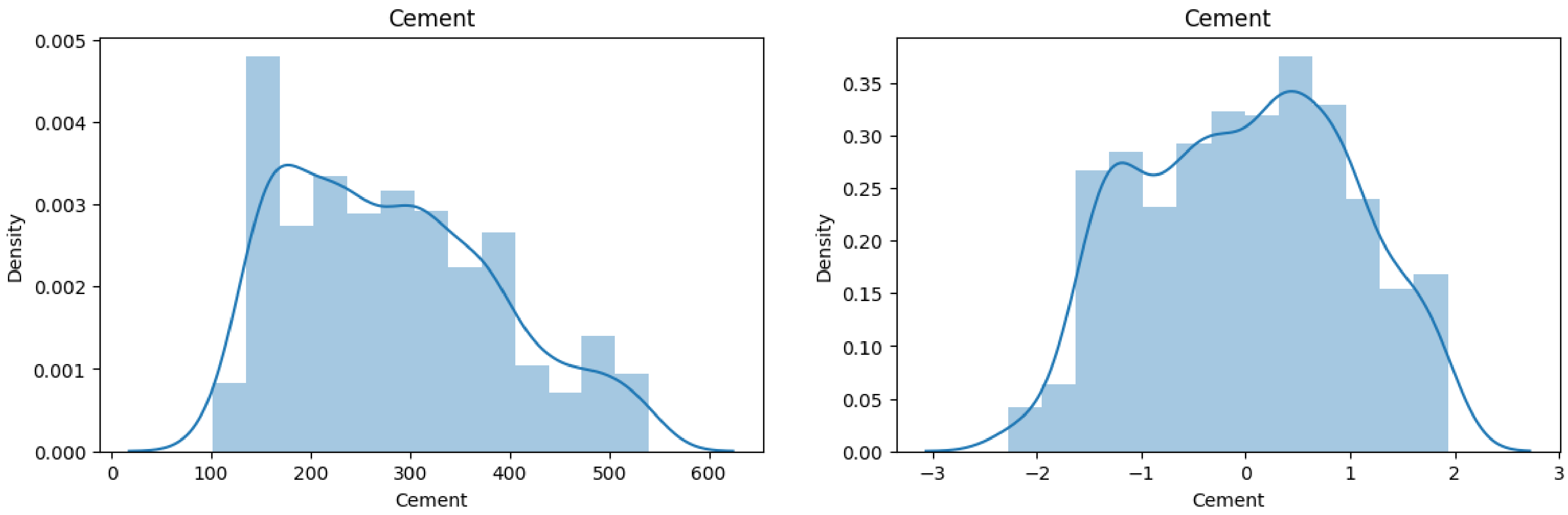
Cross validate score which comes out to 68%
Normal prediction -> 46%
Box-Cox -> 66%
Yeo-Johnson -> 68%

In [65]:
```
X_train_transformed_2 = pd.DataFrame(X_train_transformed_2,columns=X_train.columns)
```

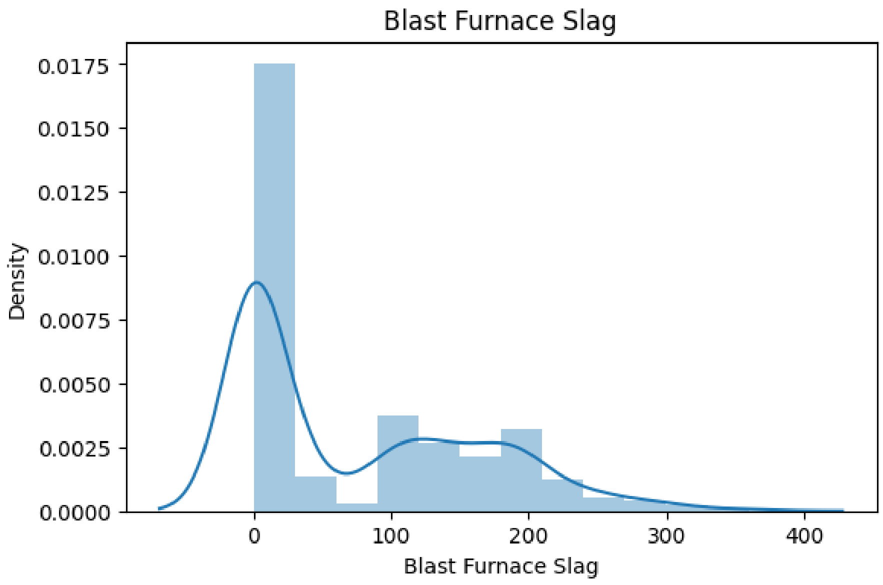
Then Converted in dataframe.

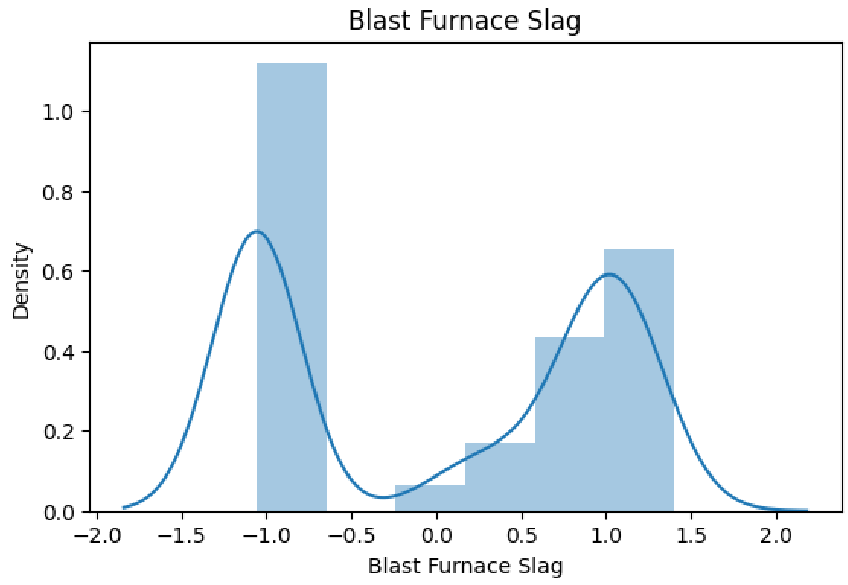
In [66]:
```
for col in X_train_transformed_2.columns:
    plt.figure(figsize=(14,4))
    plt.subplot(121)
    sns.distplot(X_train[col])
    plt.title(col)

    plt.subplot(122)
    sns.distplot(X_train_transformed_2[col])
    plt.title(col)

    plt.show()
```

```
<ipython-input-66-a65d510a5ac1>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-66-a65d510a5ac1>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed_2[col])
```

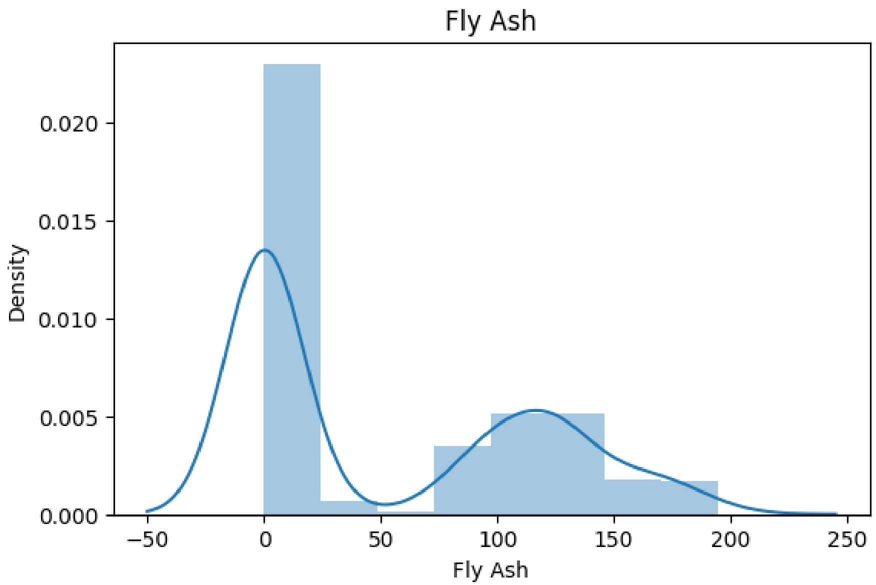

Cement
0.005
0.004
0.003
0.002
0.001
0.000
Density
0
100
200
300
400
500
600
Cement
Cement
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Density
−3
−2
−1
0
1
2
3
Cement

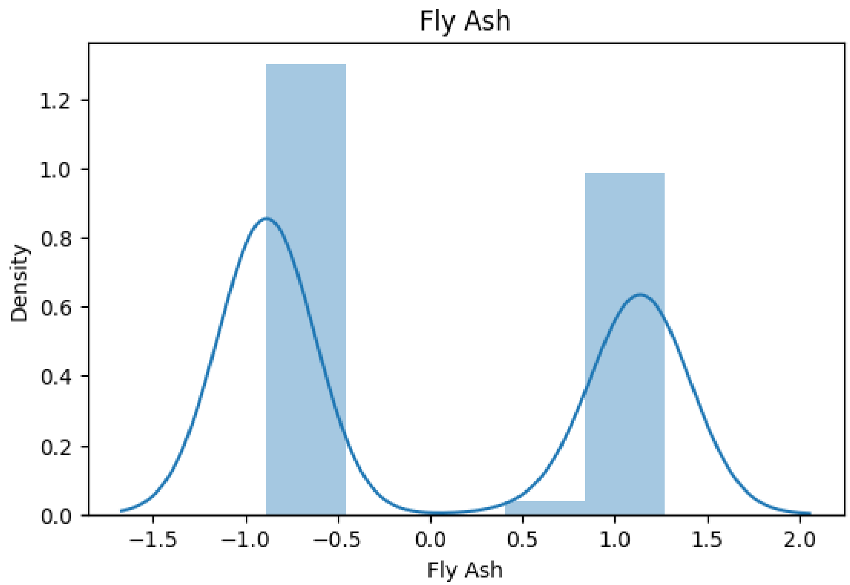
```
<ipython-input-66-a65d510a5ac1>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-66-a65d510a5ac1>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed_2[col])
```


Blast Furnace Slag
Density
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0
100
200
300
400
Blast Furnace Slag

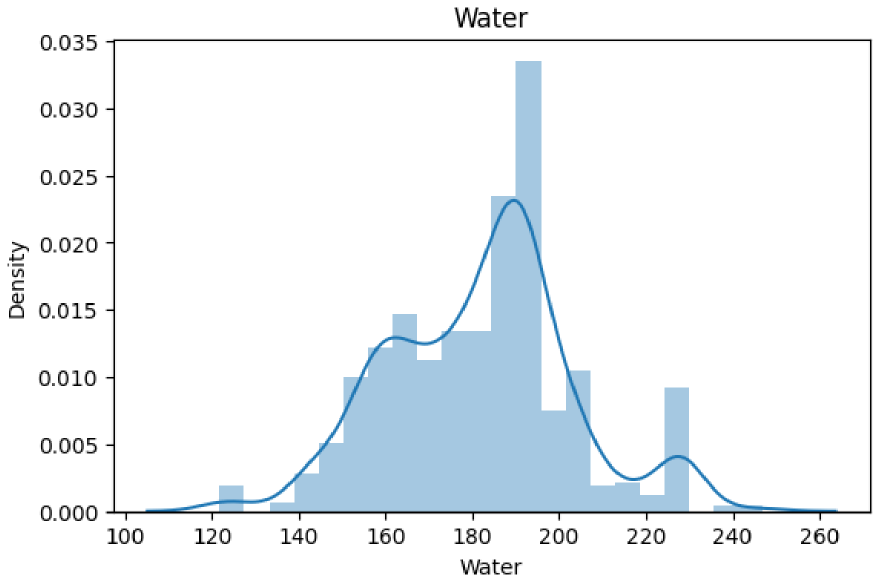

Blast Furnace Slag
Density
1.0
0.8
0.6
0.4
0.2
0.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Blast Furnace Slag

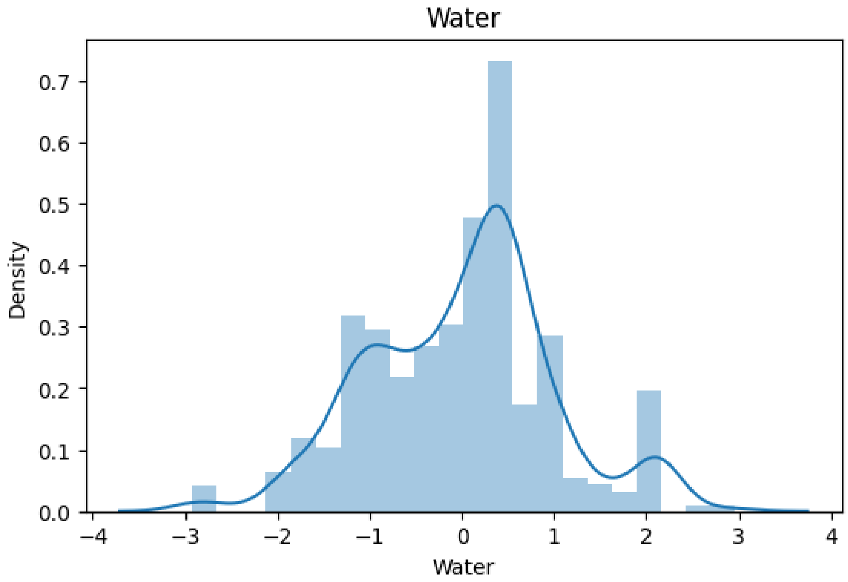
```
<ipython-input-66-a65d510a5ac1>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-66-a65d510a5ac1>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed_2[col])
```


Fly Ash
Density
0.020
0.015
0.010
0.005
0.000
−50
0
50
100
150
200
250
Fly Ash

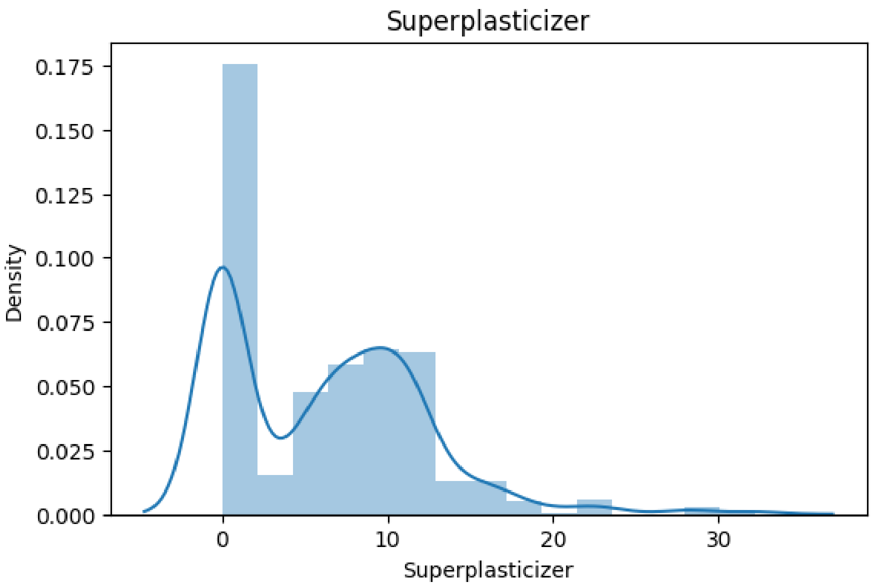

Fly Ash
Density
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Fly Ash

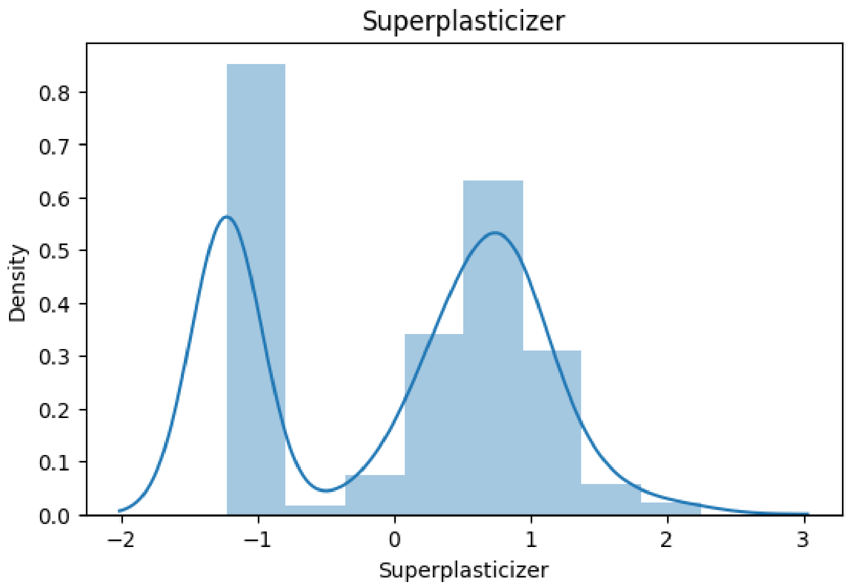
```
<ipython-input-66-a65d510a5ac1>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-66-a65d510a5ac1>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed_2[col])
```


Water
Density
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
100
120
140
160
180
200
220
240
260
Water

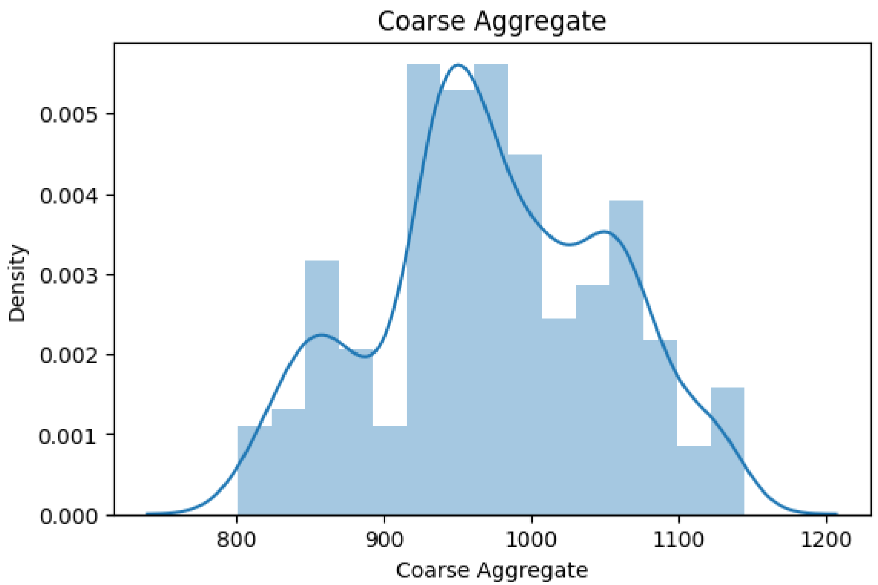

Water
Density
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−4
−3
−2
−1
0
1
2
3
4
Water

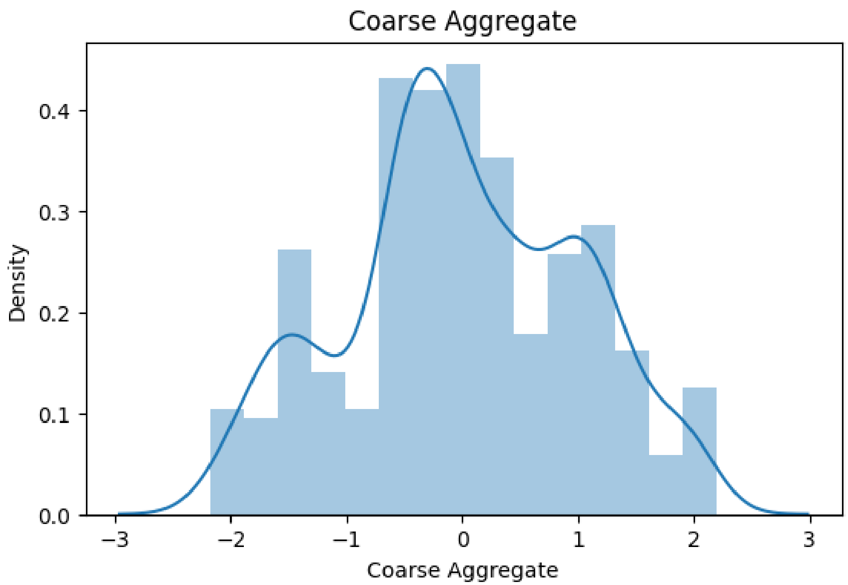
```
<ipython-input-66-a65d510a5ac1>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-66-a65d510a5ac1>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed_2[col])
```


Superplasticizer
Density
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
10
20
30
Superplasticizer

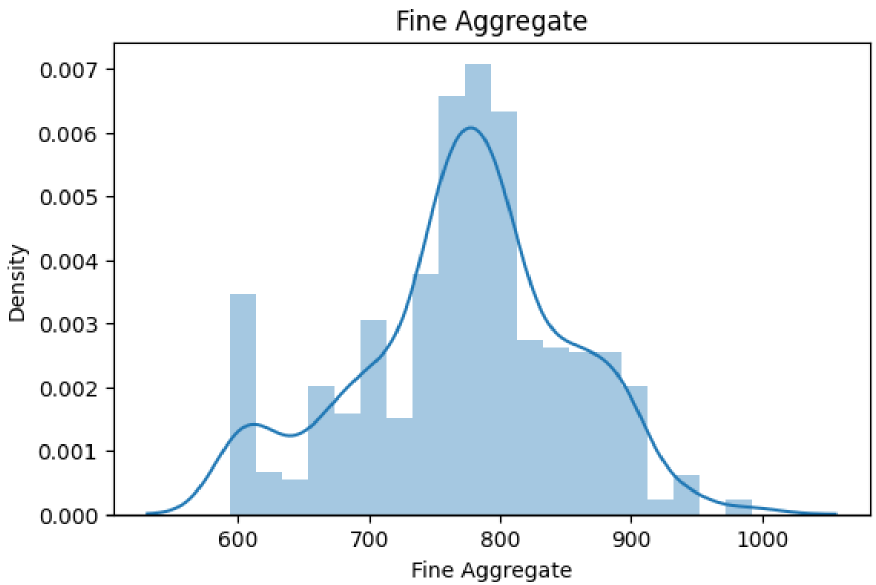

Superplasticizer
Density
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−2
−1
0
1
2
3
Superplasticizer

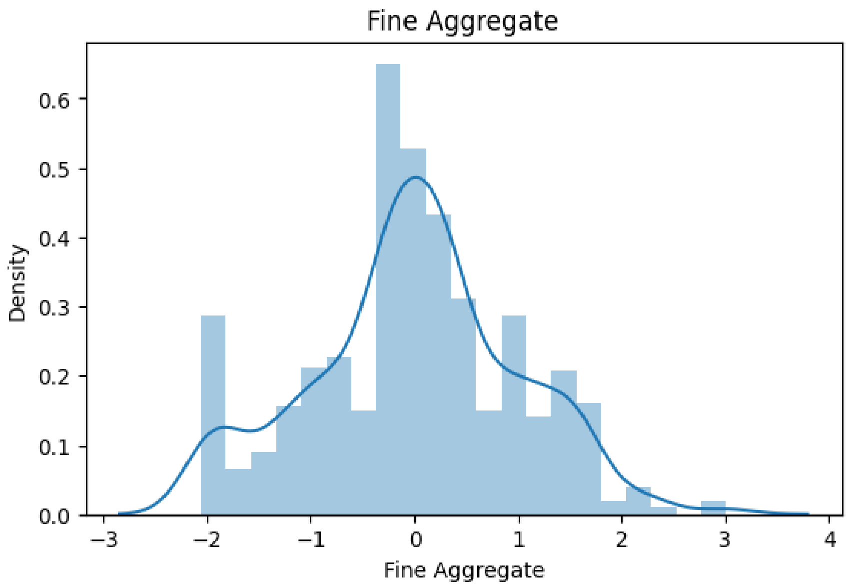
```
<ipython-input-66-a65d510a5ac1>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-66-a65d510a5ac1>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed_2[col])
```


Coarse Aggregate
Density
0.000
0.001
0.002
0.003
0.004
0.005
800
900
1000
1100
1200
Coarse Aggregate

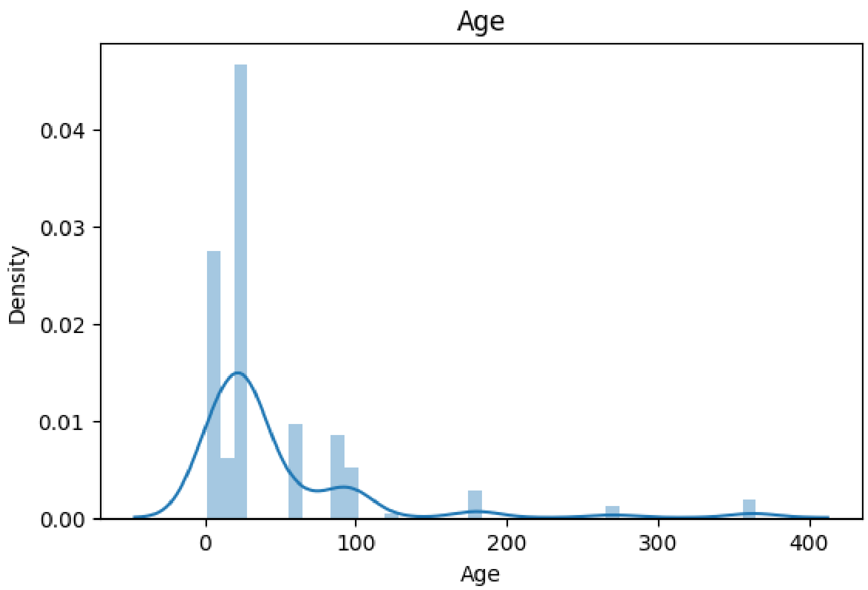

Coarse Aggregate
Density
0.0
0.1
0.2
0.3
0.4
−3
−2
−1
0
1
2
3
Coarse Aggregate

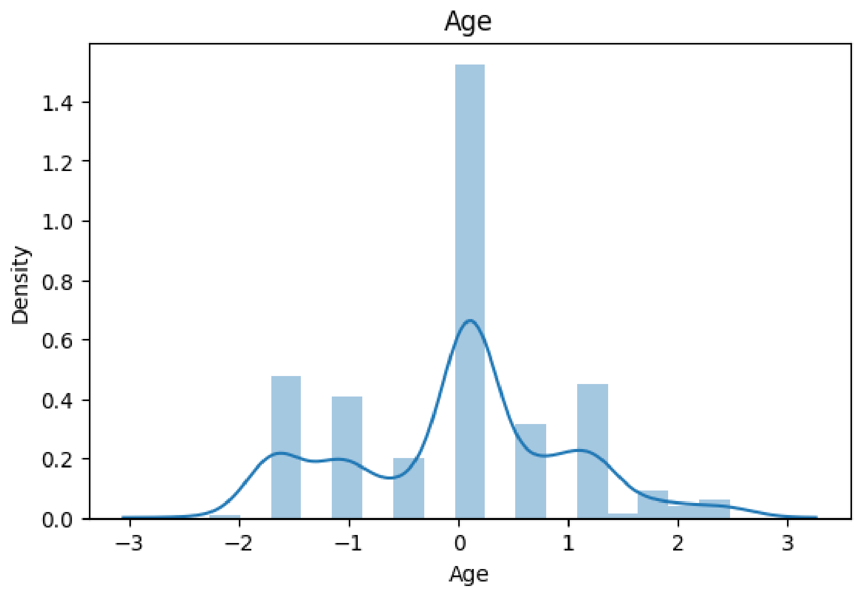
```
<ipython-input-66-a65d510a5ac1>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-66-a65d510a5ac1>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6
372750bbe5751)

  sns.distplot(X_train_transformed_2[col])
```


Fine Aggregate
Density
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
600
700
800
900
1000
Fine Aggregate


Fine Aggregate
Density
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−3
−2
−1
0
1
2
3
4
Fine Aggregate

```
<ipython-input-66-a65d510a5ac1>:4: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(X_train[col])
<ipython-input-66-a65d510a5ac1>:8: UserWarning:

`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 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(X_train_transformed_2[col])
```





Plotted before transformation and after transformation
And we can observe the difference of Box-Cox and Yeo-Johnson is very minute.

In [67]:
```
pd.DataFrame({'cols':X_train.columns,'box_cox_lambdas':P_T_2.lambdas_,'Yeo_Johnson_lambdas':Power_Transformer.lambdas_
```

Out[67]:

	cols	box_cox_lambdas	Yeo_Johnson_lambdas
0	Cement	0.169544	0.174348
1	Blast Furnace Slag	0.016633	0.015715
2	Fly Ash	-0.136480	-0.161447
3	Water	0.808438	0.771307
4	Superplasticizer	0.264160	0.253935
5	Coarse Aggregate	1.129395	1.130050
6	Fine Aggregate	1.830763	1.783100
7	Age	0.001771	0.019885

Here is the comparison of Box-Cox and Yeo-Johnson exponent values.