Feature Engineering 101

Topic - 4

Function Transformer

Sample Code

```python
from sklearn.base import BaseEstimator, TransformerMixin

class MultiplyTransformer(BaseEstimator, TransformerMixin):
    def __init__(self, factor=1.0):
        self.factor = factor

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        return X * self.factor
```

Function Transformer

In [1]:
```python
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
```

In [2]:
```python
df = pd.read_csv('train.csv',usecols=['Age','Fare','Survived'])
```

In [3]:
```python
df
```

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

	Survived	Age	Fare
888	0	NaN	23.4500
889	1	26.0	30.0000
890	0	32.0	7.7500

891 rows × 3 columns

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
Survived      0
Age         177
Fare          0
dtype: int64
```

In [5]:
```
df['Age'].fillna(df['Age'].mean(),inplace=True)
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
Survived    0
Age         0
Fare        0
dtype: int64
```

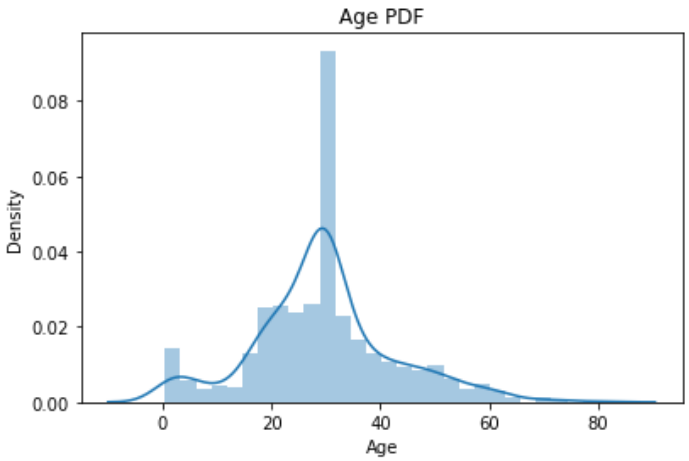
In [7]:
```
X = df.iloc[:,1:3]
y = df.iloc[:,0]
```

In [8]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=42)
```

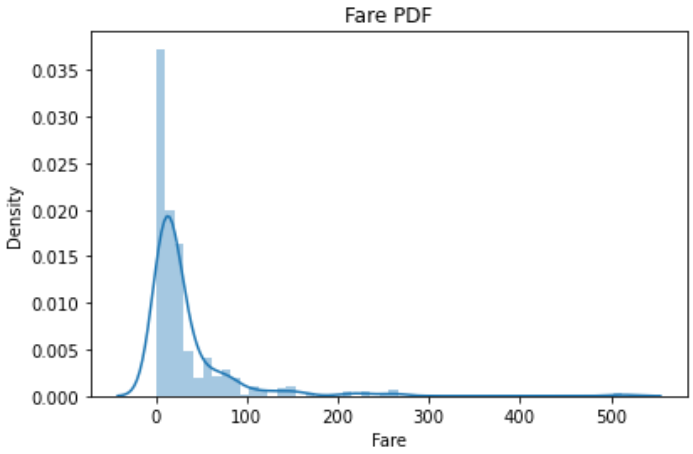
In [9]:
```
#Age>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
plt.figure(figsize=(14,4))
plt.subplot(121)
sns.distplot(X_train['Age'])
plt.title('Age PDF')

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

plt.show()

#Fare >>>>>>>>>>>>>>>>>>>>>>>>>>>
plt.figure(figsize=(14,4))
plt.subplot(121)
sns.distplot(X_train['Fare'])
plt.title('Fare PDF')

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

plt.show()
```

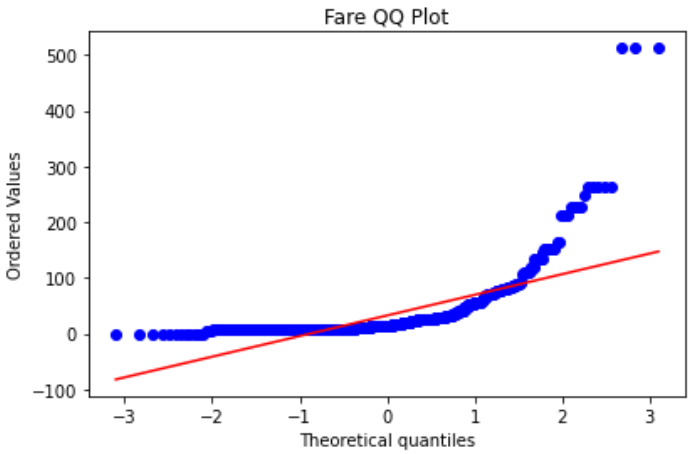
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
```

```
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```





In [10]:
```
#Now the classifire time

clf = LogisticRegression()
clf2 = DecisionTreeClassifier()
```

In [11]:
```
#Predict nad Fit

clf.fit(X_train,y_train)
clf2.fit(X_train,y_train)

y_pred = clf.predict(X_test)
y_pred1 = clf2.predict(X_test)
```

In [12]:
```
#Acc. Score

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

```
Accuracy LR 0.6480446927374302
Accuracy DT 0.664804469273743
```

Now use Log1 TransforMer

In [13]:
```
trf = FunctionTransformer(func=np.log1p)
```

In [14]:
```
X_train_transformed = trf.fit_transform(X_train)
X_test_transformed = trf.transform(X_test)
```

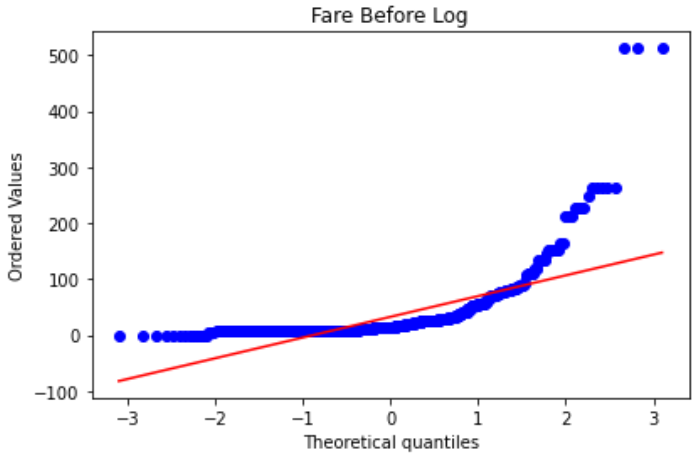
In [15]:
```
clf = LogisticRegression()
clf2 = DecisionTreeClassifier()

clf.fit(X_train_transformed,y_train)
clf2.fit(X_train_transformed,y_train)

y_pred = clf.predict(X_test_transformed)
y_pred1 = clf2.predict(X_test_transformed)

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

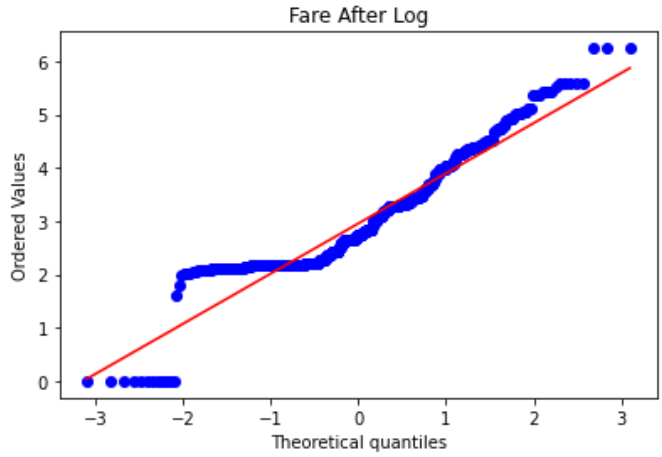
```
Accuracy LR 0.6815642458100558
Accuracy DT 0.6815642458100558
```

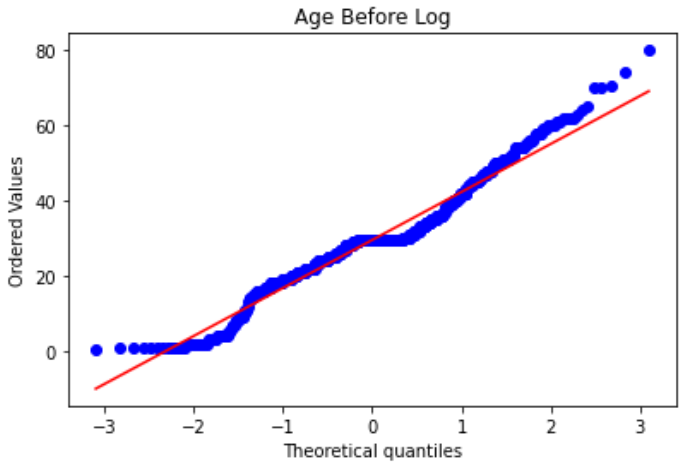
In [16]:
```
#With cross val score

X_transformed = trf.fit_transform(X)

clf = LogisticRegression()
clf2 = DecisionTreeClassifier()

print("LR",np.mean(cross_val_score(clf,X_transformed,y,scoring='accuracy',cv=10)))
print("DT",np.mean(cross_val_score(clf2,X_transformed,y,scoring='accuracy',cv=10)))
```

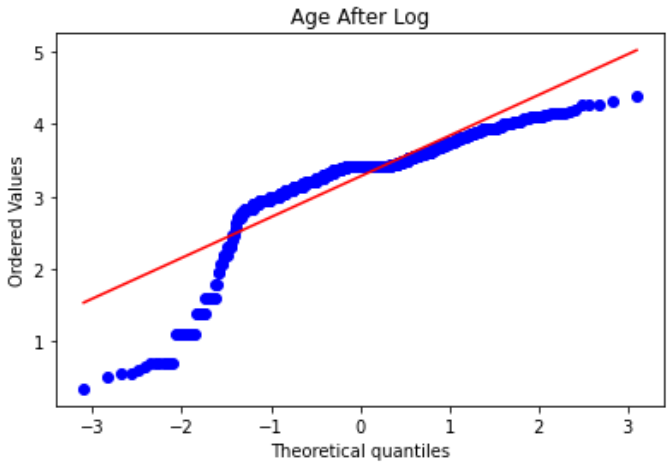
```
LR 0.678027465667915
DT 0.661123595505618
```

In [17]:
```
#Fare >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
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()

#Age >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
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()
```









Now use Log TransforMer

In [18]:
```
trf2 = ColumnTransformer([('log',FunctionTransformer(np.log1p),['Fare'])],remainder='passt

X_train_transformed2 = trf2.fit_transform(X_train)
X_test_transformed2 = trf2.transform(X_test)
```

In [19]:
```
clf = LogisticRegression()
clf2 = DecisionTreeClassifier()

clf.fit(X_train_transformed2,y_train)
clf2.fit(X_train_transformed2,y_train)

y_pred = clf.predict(X_test_transformed2)
y_pred2 = clf2.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 0.6871508379888268
```

In [20]:
```
X_transformed2 = trf2.fit_transform(X)

clf = LogisticRegression()
clf2 = DecisionTreeClassifier()

print("LR",np.mean(cross_val_score(clf,X_transformed2,y,scoring='accuracy',cv=10)))
print("DT",np.mean(cross_val_score(clf2,X_transformed2,y,scoring='accuracy',cv=10)))
```

```
LR 0.6712609238451936
DT 0.6610736579275904
```

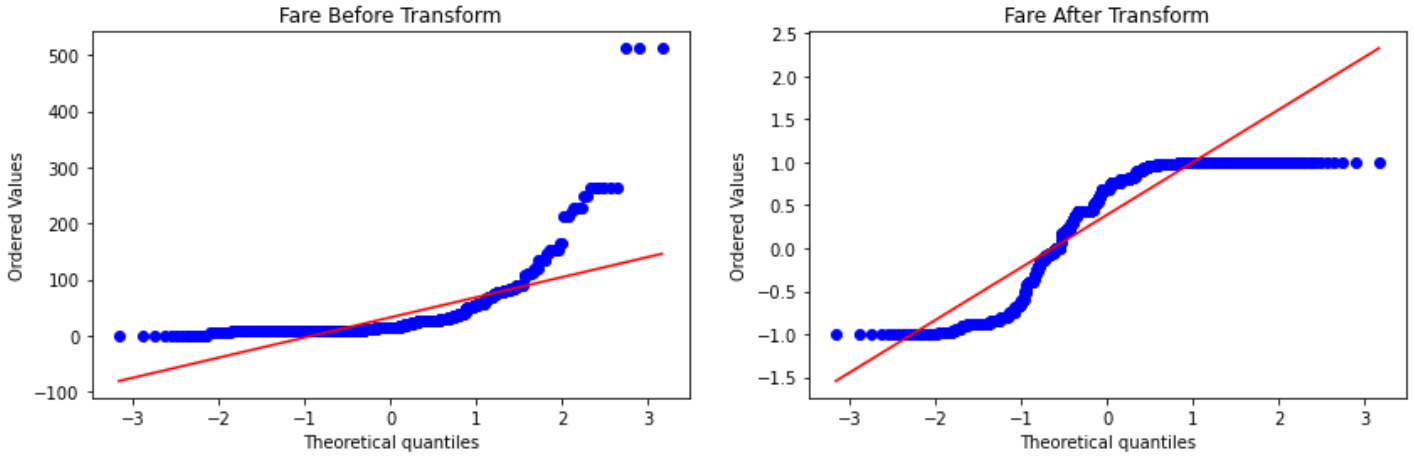
In [21]:

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

    trf = ColumnTransformer([('log',FunctionTransformer(transform),['Fare'])],remainder='p

    X_trans = trf.fit_transform(X)

    clf = LogisticRegression()

    print("Accuracy",np.mean(cross_val_score(clf,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()
```

In [22]:

```
apply_transform(np.sin)
```

```
Accuracy 0.6195131086142323
```



In [ ]: