Gaussian Distribution

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import scipy.stats as stats
```

```
/Users/abhilashavadhanula/Downloads/anaconda3/lib/python3.6/importlib/_bootstrap.py:219: Runt
imeWarning: numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from
C header, got 216 from PyObject
  return f(*args, **kwds)
/Users/abhilashavadhanula/Downloads/anaconda3/lib/python3.6/importlib/_bootstrap.py:219: Runt
imeWarning: numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from
C header, got 216 from PyObject
  return f(*args, **kwds)
/Users/abhilashavadhanula/Downloads/anaconda3/lib/python3.6/importlib/_bootstrap.py:219: Runt
imeWarning: numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from
C header, got 216 from PyObject
  return f(*args, **kwds)
/Users/abhilashavadhanula/Downloads/anaconda3/lib/python3.6/importlib/_bootstrap.py:219: Runt
imeWarning: numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from
C header, got 216 from PyObject
  return f(*args, **kwds)
```

In [17]:
```
data=pd.read_csv('/Users/abhilashavadhanula/downloads/titanic/train.csv',usecols=['Age','Far
e','Survived'])
```

In [18]:
```
data.head()
```

Out[18]:

	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 [19]:
```
data.isnull().sum()
```

Out[19]:
```
Survived      0
Age         177
Fare          0
dtype: int64
```

In [24]:
```
def impute_na(data, variable):
    # function to fill na with a random sample
    df = data.copy()

    # random sampling
    df[variable+'_random'] = df[variable]

    # extract the random sample to fill the na
    random_sample = df[variable].dropna().sample(df[variable].isnull().sum(), random_state=0
)

    # pandas needs to have the same index in order to merge datasets
    random_sample.index = df[df[variable].isnull()].index
    df.loc[df[variable].isnull(), variable+'_random'] = random_sample

    return df[variable+'_random']
```

In [25]:
```
data['Age']=impute_na(data,'Age')
```

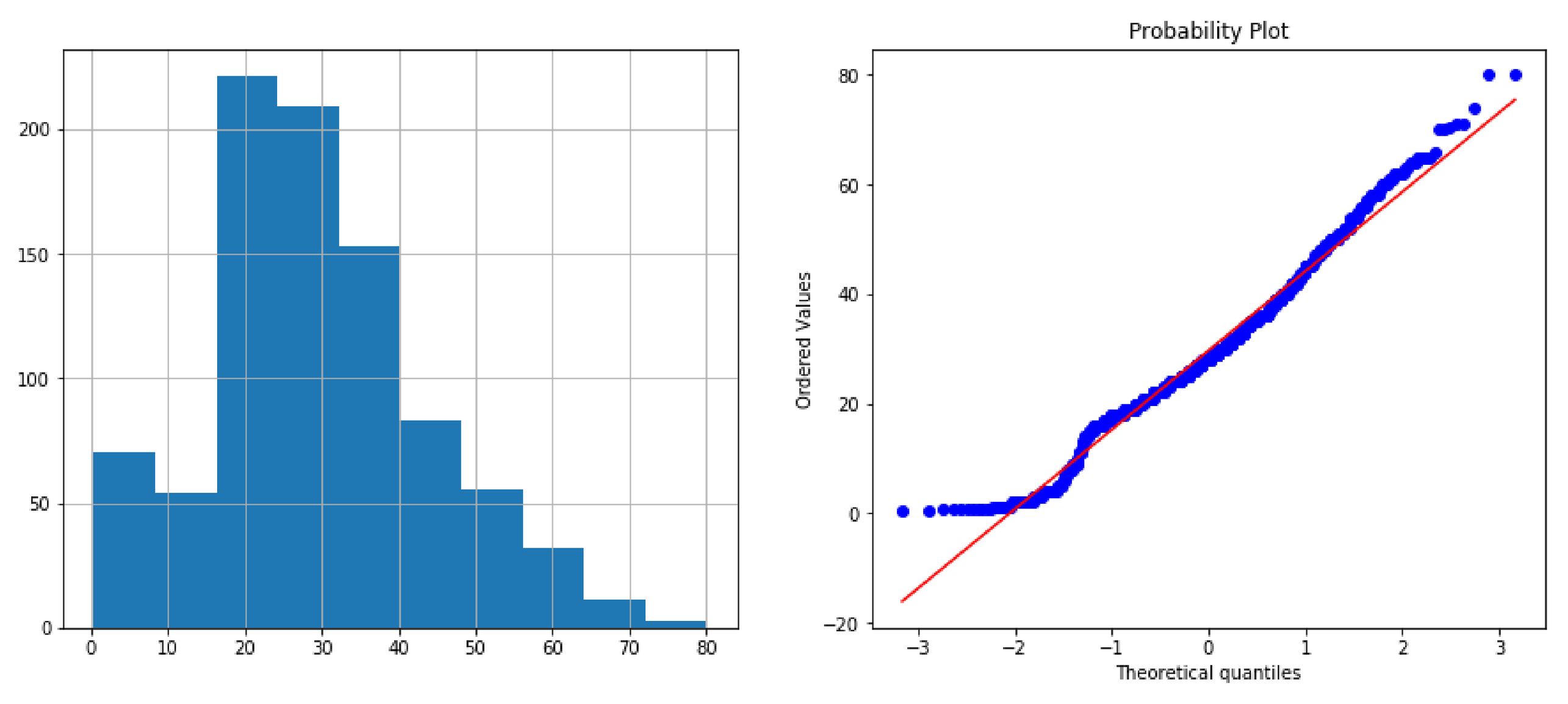
In [26]:
```
data.isnull().sum()
```

Out[26]:
```
Survived    0
Age         0
Fare        0
dtype: int64
```

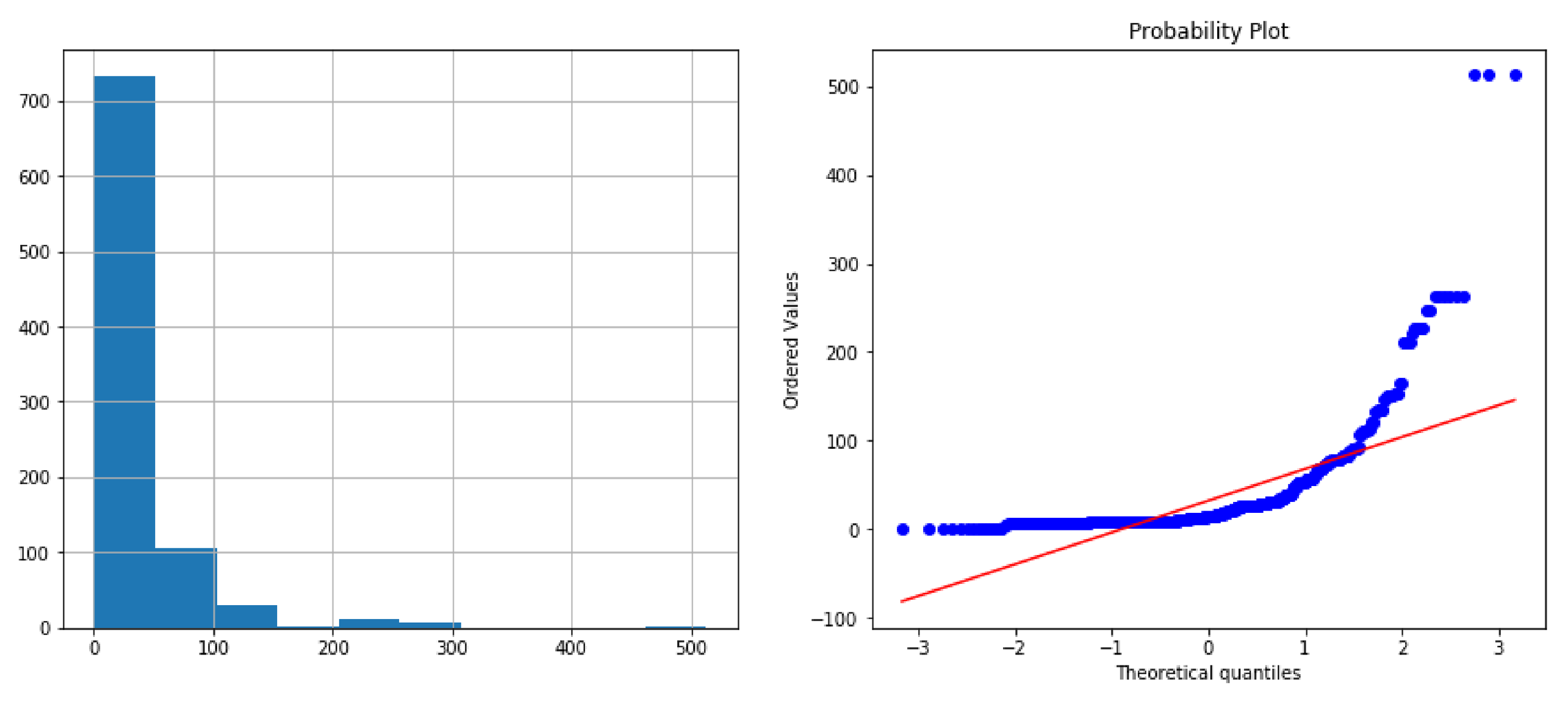
QQ-Plot

In [27]:
```
def diagnostic_plots(df, variable):
    # function to plot a histogram and a Q-Q plot
    # side by side, for a certain variable

    plt.figure(figsize=(15,6))
    plt.subplot(1, 2, 1)
    df[variable].hist()

    plt.subplot(1, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)

    plt.show()
```

In [28]:
```
diagnostic_plots(data, 'Age')
```

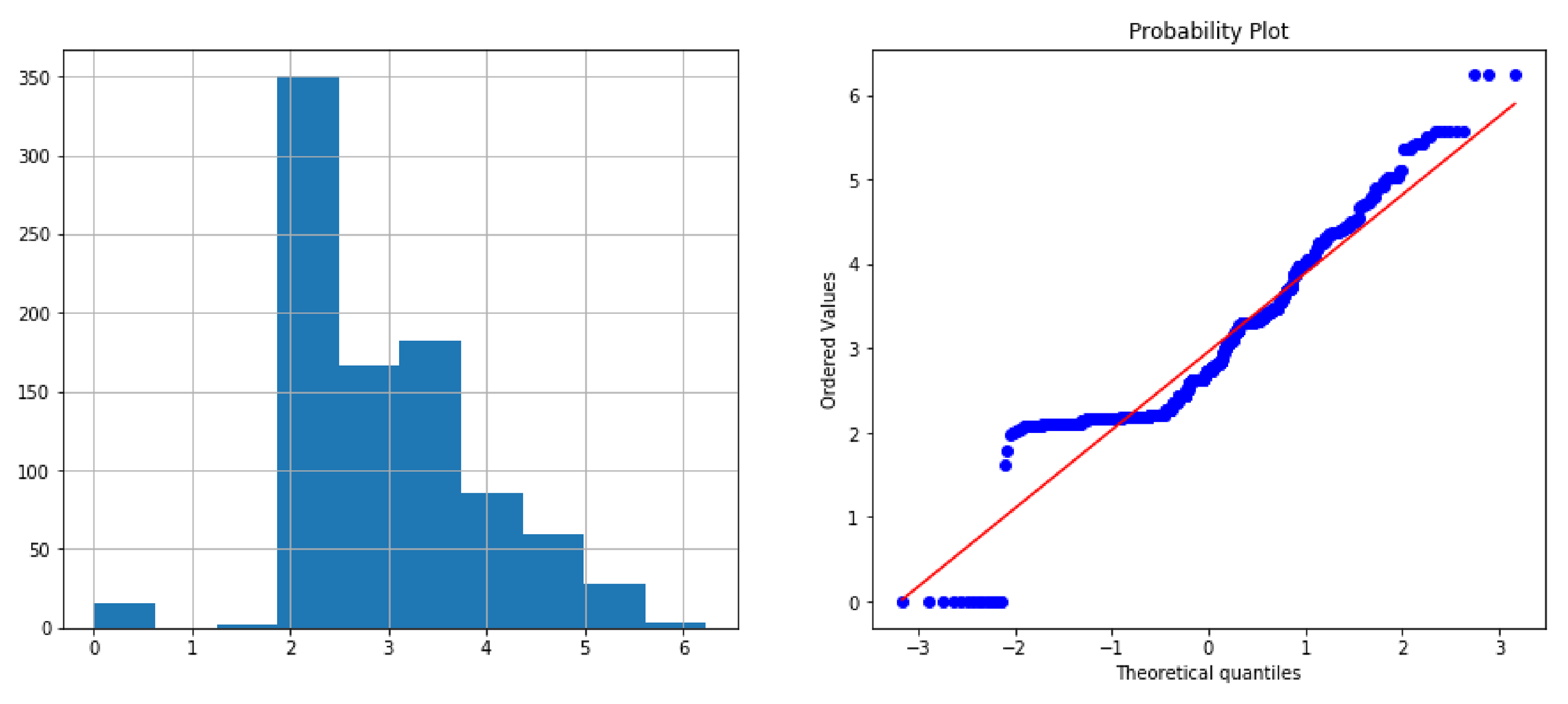


In [30]:
```
diagnostic_plots(data, 'Fare')
```



Logarithmic Transformation

In [31]:
```
data['Log_Fare']=np.log(data['Fare']+1)
diagnostic_plots(data,'Log_Fare')
```



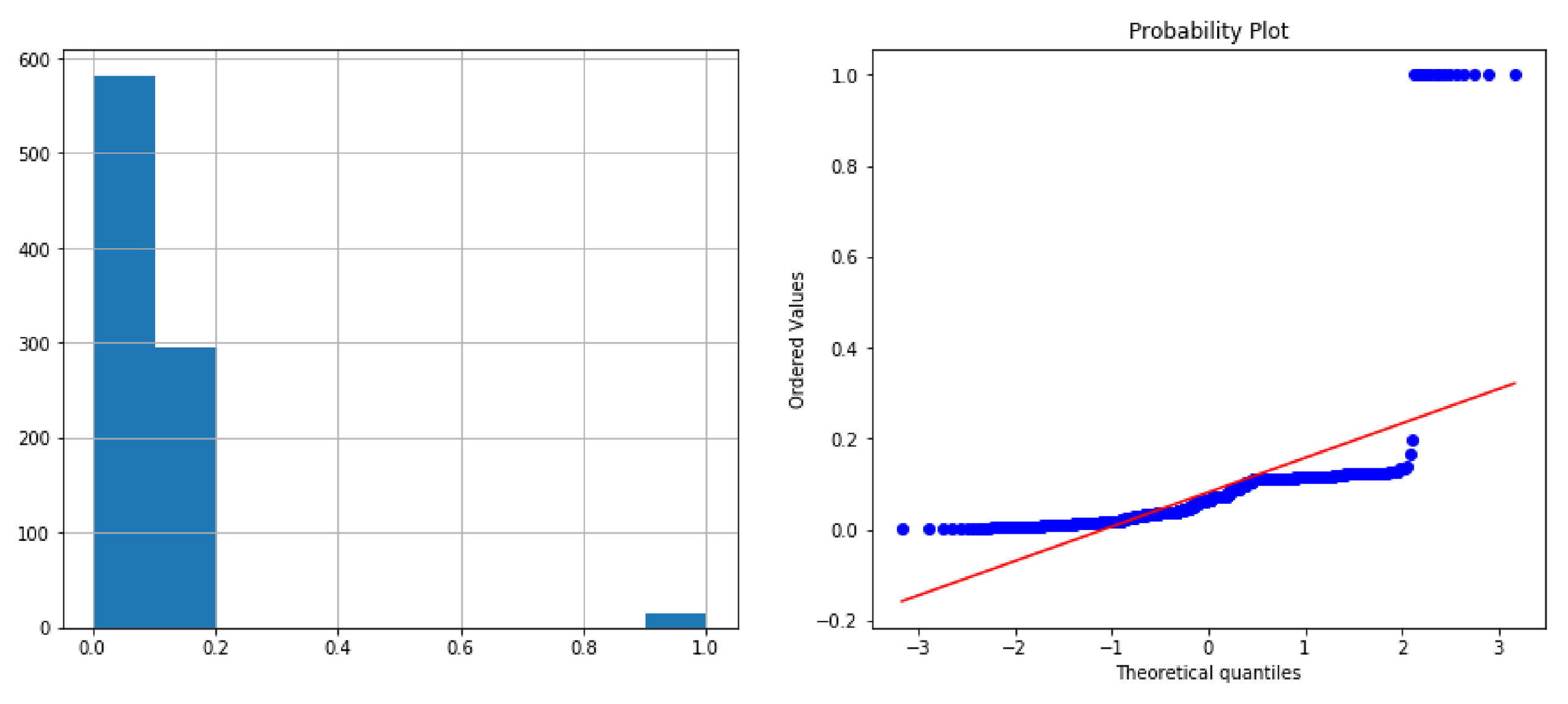
The logarithmic distribution does a good job in making Fare variable look Gaussian Distributed

Reciprocal transformation

In [ ]:
```
```

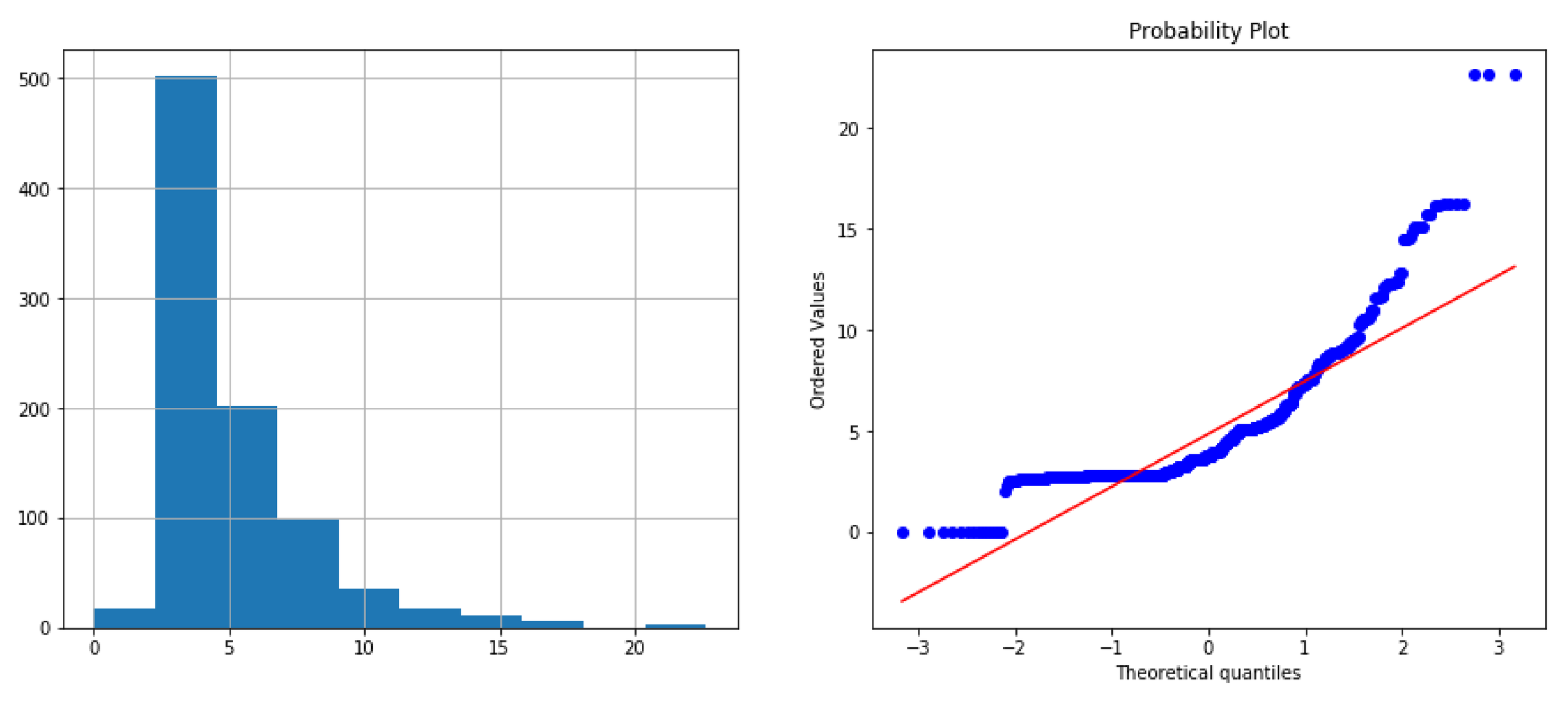
In [ ]:
```
```

In [32]:
```
data['Rec_Fare']=1/(data['Fare']+1)
diagnostic_plots(data,'Rec_Fare')
```



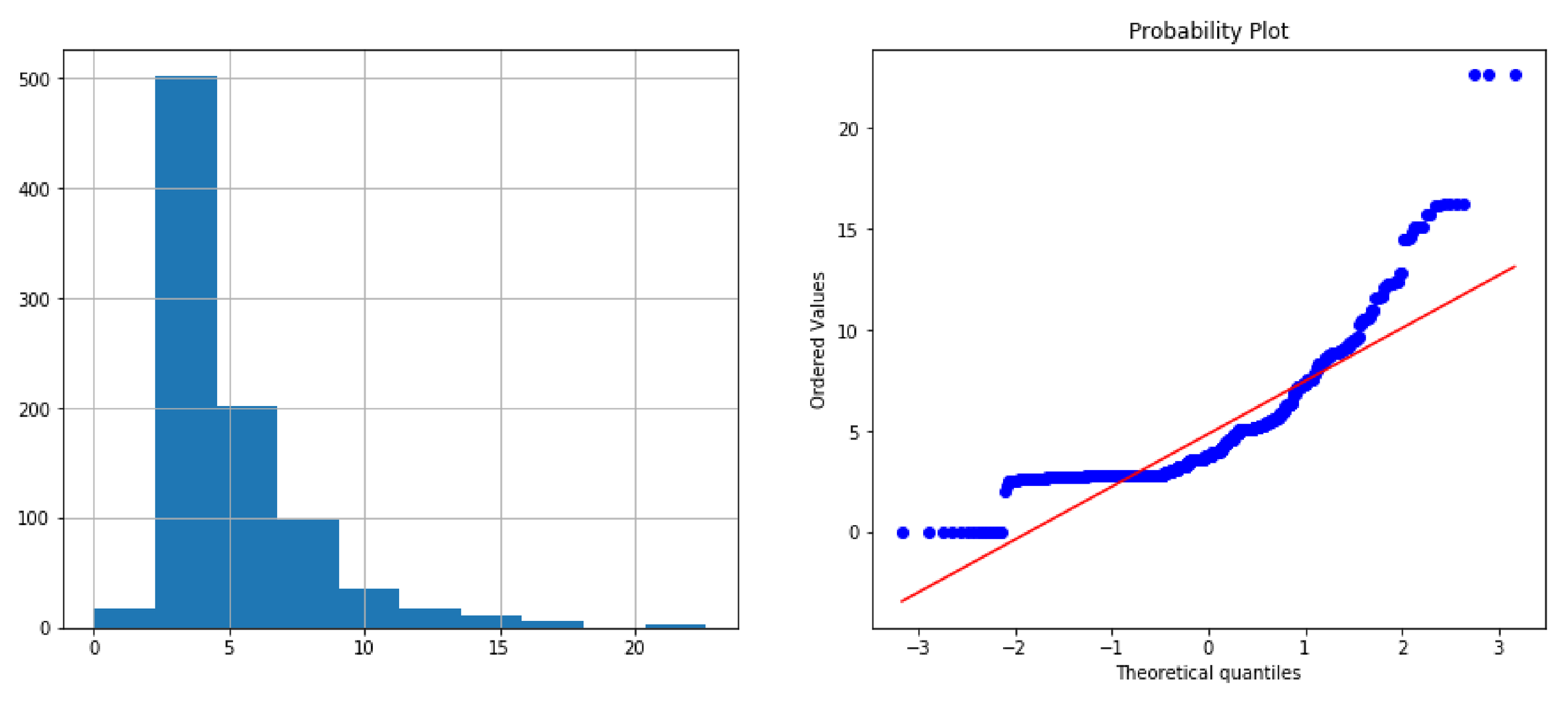
Sqroot Transformation

In [33]:
```
data['sqr_Fare']=data['Fare']**(1/2)
diagnostic_plots(data,'sqr_Fare')
```



Exponential Transformation

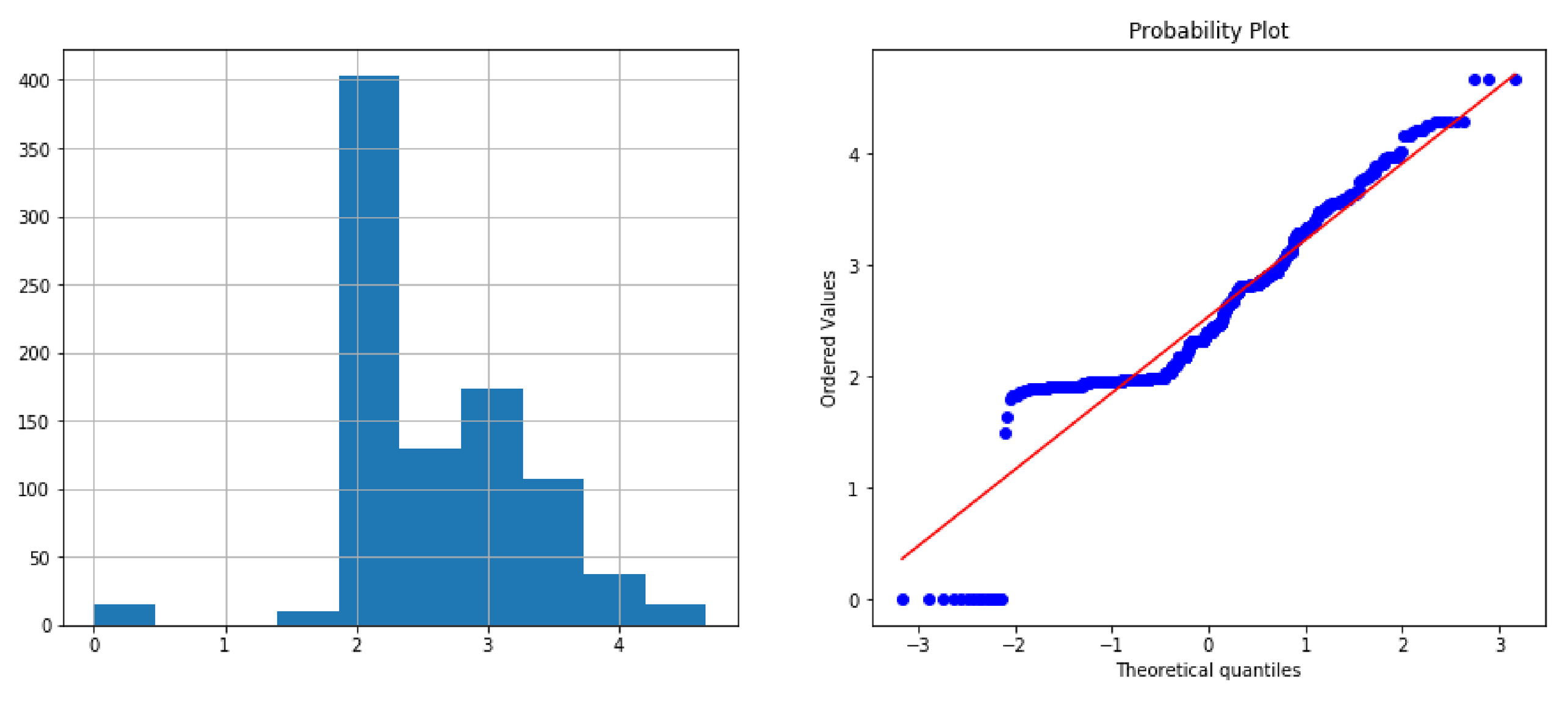
In [35]:
```
data['Exp_Fare']=data['Fare']**(1/5)
diagnostic_plots(data,'sqr_Fare')
```



BoxCox Transformation

In [36]:
```
data['Fare_boxcox'], param = stats.boxcox(data.Fare+1) # you can vary the exponent as needed

print('Optimal lambda: ', param)

diagnostic_plots(data, 'Fare_boxcox')
```

```
Optimal lambda:  -0.09778702818680361
```



In [ ]:
```
```