Load Dataset

```
from models.utils import loadData

x, y = loadData()

print("Input features: ", x)
```

```
Input features:       9.1
0    8.0
1    9.1
2    8.4
3    6.9
4    7.7
..   ...
94   7.8
95  10.2
96   6.1
97   7.3
98   7.3

[99 rows x 1 columns]
```

```
print("Output Labels: ", y)
```

```
Output Labels:      0.99523
0   0.99007
1   0.99769
2   0.99386
3   0.99508
4   0.99630
..      ...
94  0.99620
95  0.99760
96  0.99464
97  0.99830
98  0.99670

[99 rows x 1 columns]
```

Normalisation

```
from models.utils import MinMaxScaler

scaler = MinMaxScaler()
xNorm = scaler.fitTransform(x)
yNorm = scaler.fitTransform(y)
```

```
print(xNorm)
print(yNorm)
```

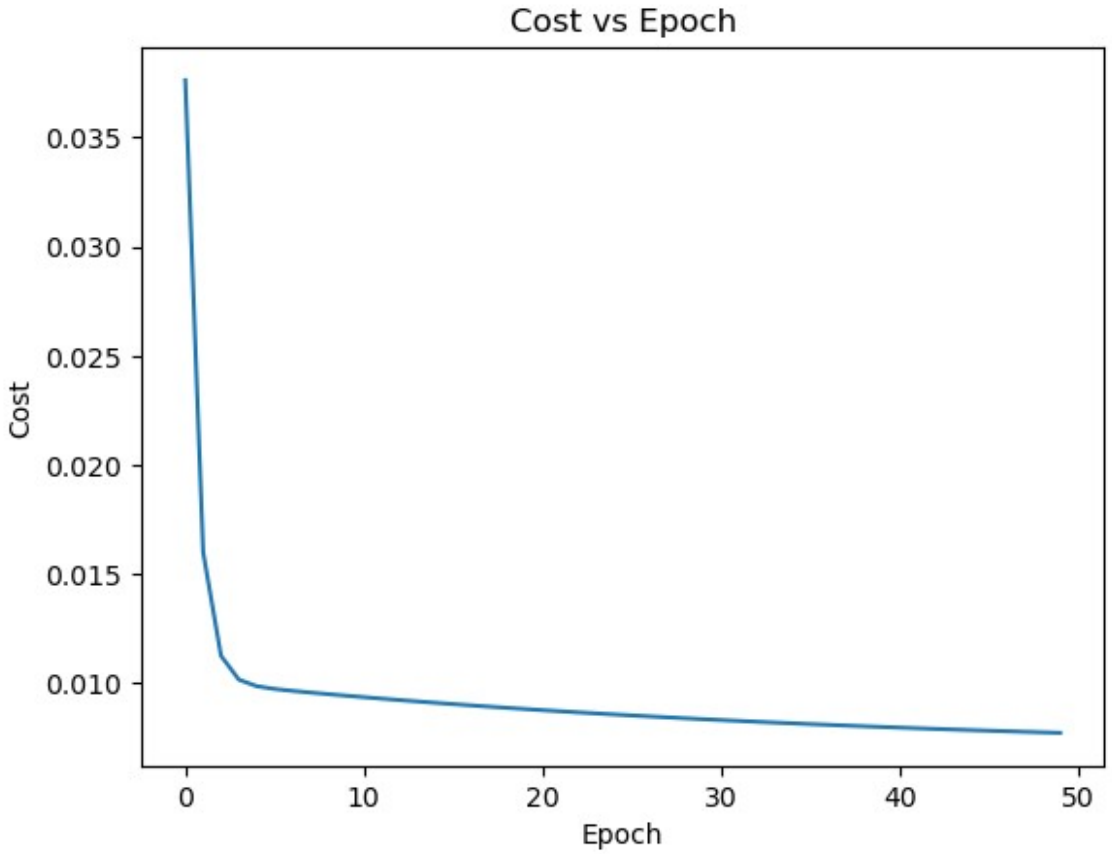
```
         9.1
0   0.254902
1   0.362745
2   0.294118
3   0.147059
4   0.225490
..       ...
94  0.235294
95  0.470588
96  0.068627
97  0.186275
98  0.186275

[99 rows x 1 columns]
     0.99523
0   0.000000
1   0.580350
2   0.288652
3   0.381569
4   0.474486
..       ...
94  0.466870
95  0.573496
96  0.348058
97  0.626809
98  0.504950

[99 rows x 1 columns]
```

Batch Gradient Descent

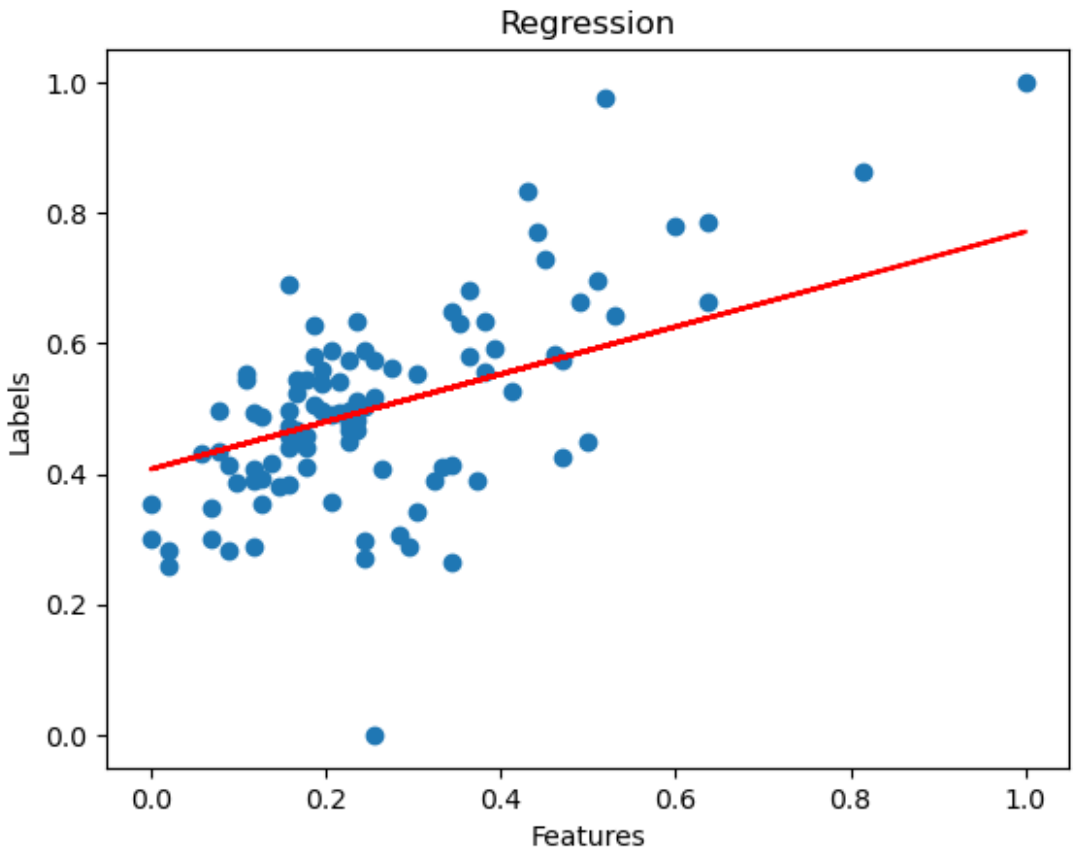
```
from models.utils import LinearRegression

model = LinearRegression()
alpha = 0.5
model.fit(xNorm, yNorm, epochs=50, a=alpha)
yHat = model.predict(10)
print("Prediction: ", yHat)
print(f"Cost Function: {model.costFunc(xNorm.to_numpy(),
yNorm.to_numpy())}")
print(f"Parameters:\t w:{model.w}, b: {model.b}")
```

```
Prediction:  [4.04513539]
Cost Function: [0.00773091]
Parameters:      w:[0.36380035], b: [0.40713192]
```

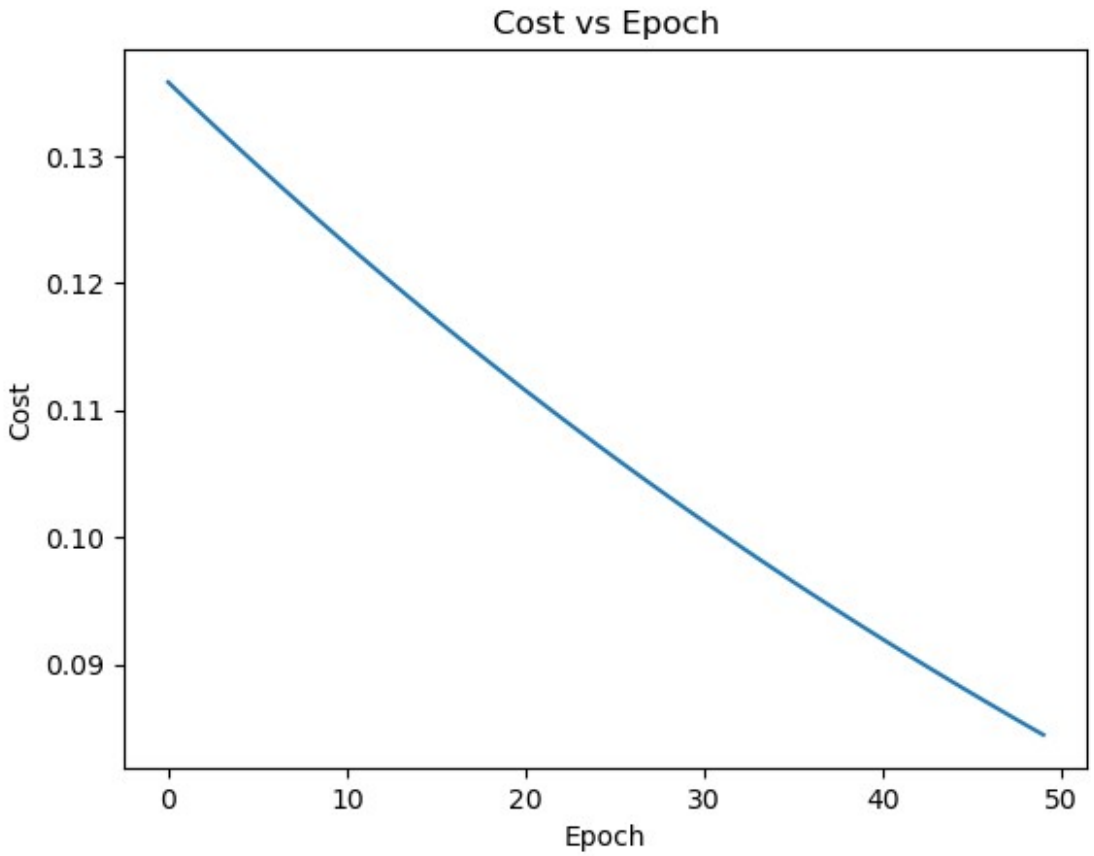
```
model.plotCost(epochs=50)
```



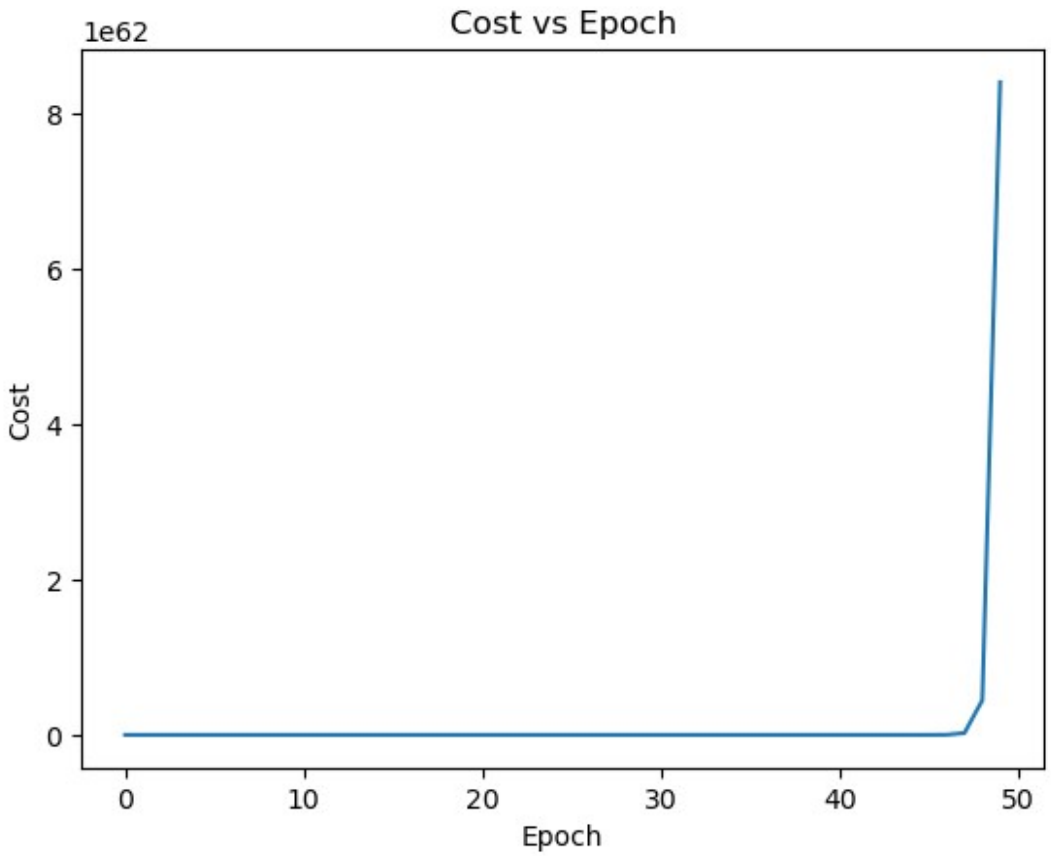

```
model.scatterPlot(xNorm, yNorm)
```




```
alpha = 0.005
model = LinearRegression()
model.fit(xNorm, yNorm, epochs=50, a=alpha)
yHat = model.predict(10)
print("Prediction: ", yHat)

Prediction:  [0.43793493]

model.plotCost(epochs=50)
```




```
alpha=5
model = LinearRegression()
model.fit(xNorm, yNorm, epochs=50, a=alpha)
yHat = model.predict(10)
print("Prediction: ", yHat)

Prediction:  [-1.40626562e+32]

model.plotCost(epochs=50)
```

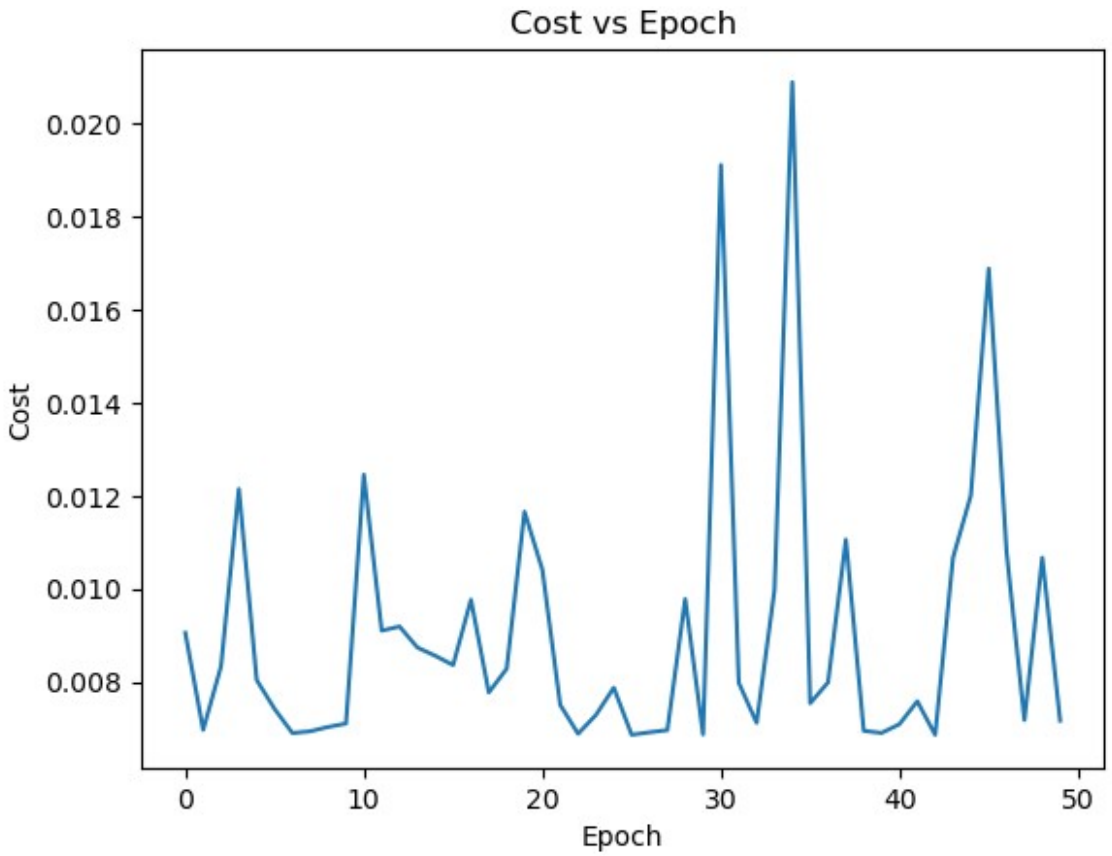


Stochastic Gradient Descent

```
alpha = 0.5
model = LinearRegression()
model.sgdFit(xNorm, yNorm, epochs=50, a=alpha)
yHat = model.predict(10)
print("Prediction: ", yHat)
```

```
Prediction:  [6.64958808]
```

```
model.plotCost(epochs=50)
```

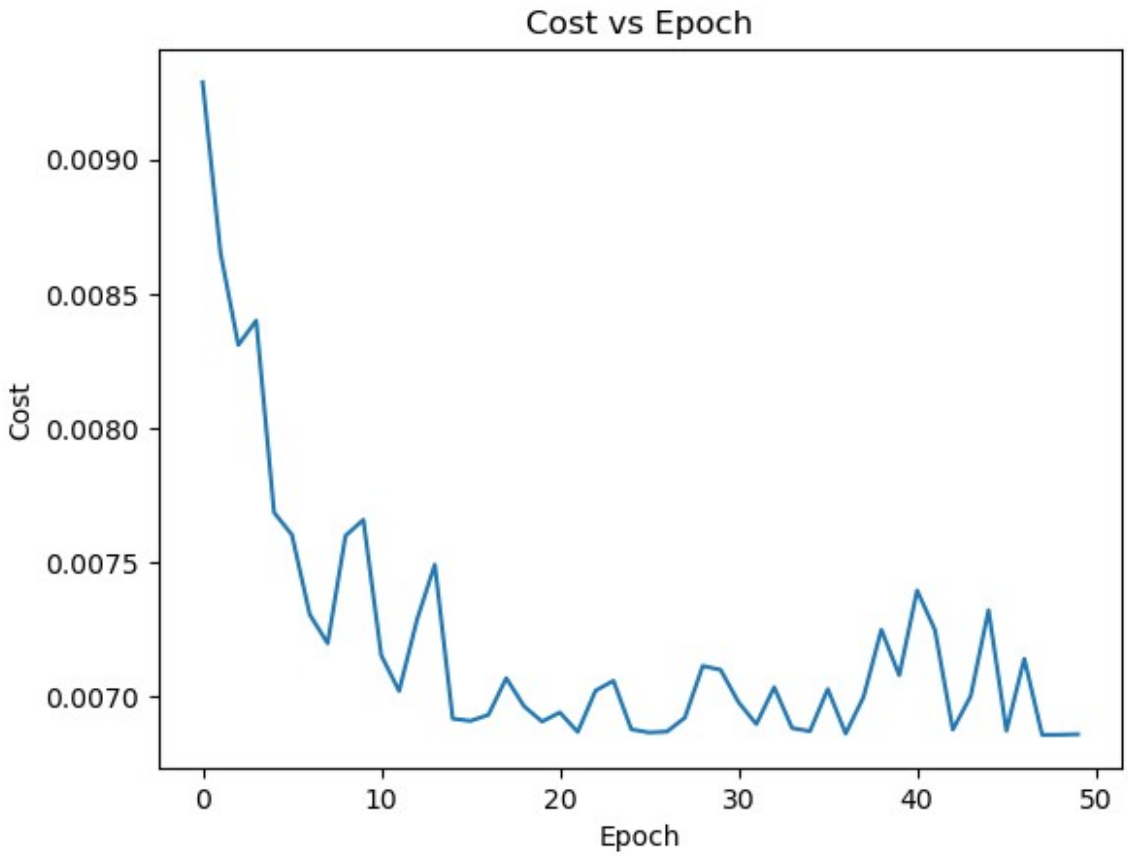


Mini-Batch Gradient Descent

```
model = LinearRegression()
model.mbgdFit(xNorm, yNorm, batchSize=10, epochs=50, a=alpha)
yHat = model.predict(10)
print("Prediction: ", yHat)
```

```
Prediction:  [6.44835099]
```

```
model.plotCost(epochs=50)
```


Cost vs Epoch
Cost
0.0090
0.0085
0.0080
0.0075
0.0070
0
10
20
30
40
50
Epoch