Loading Dataset

```
from models.vectorUtils 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.vectorUtils import MinMaxScaler

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

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

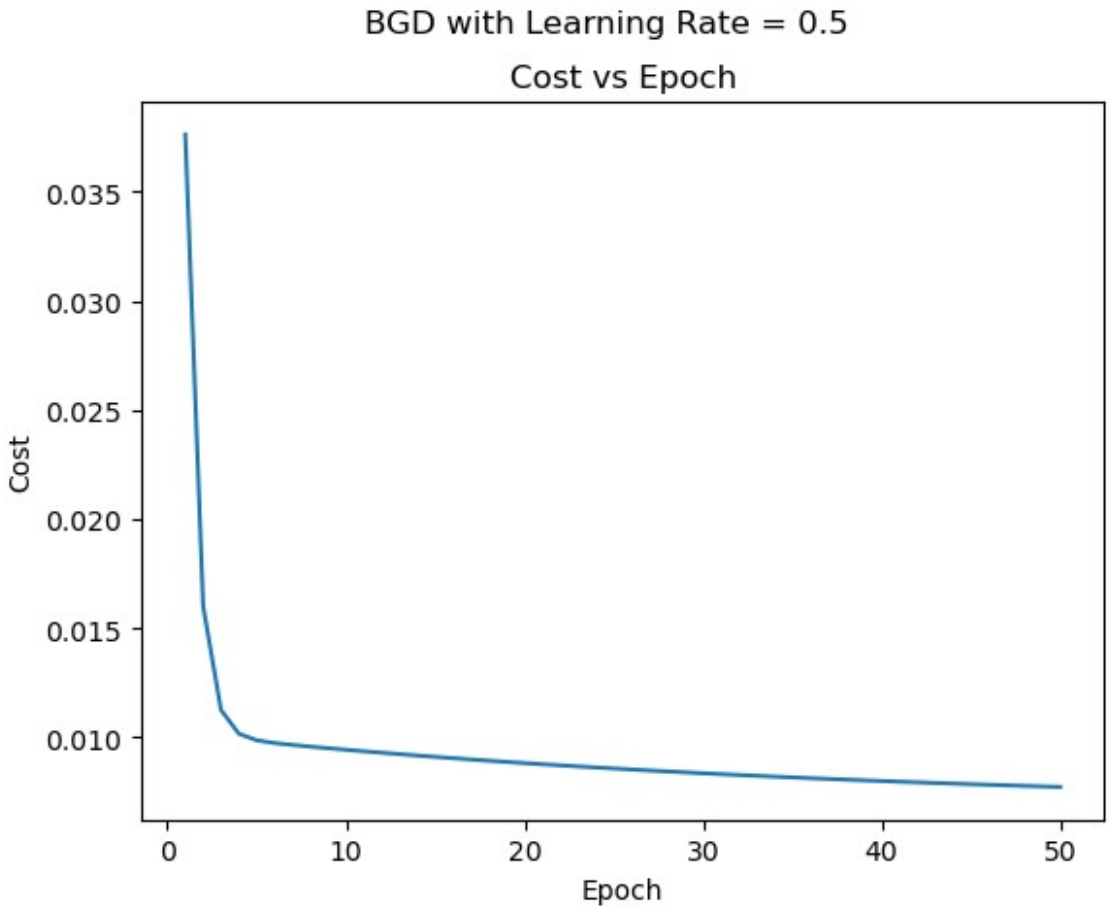
```
         9.1
0   0.254902
1   0.362745
2   0.294118
3   0.147059
4   0.225490
..       ...
94  0.235294
95  0.470588
96  0.068627
97  0.186275
98  0.186275

[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

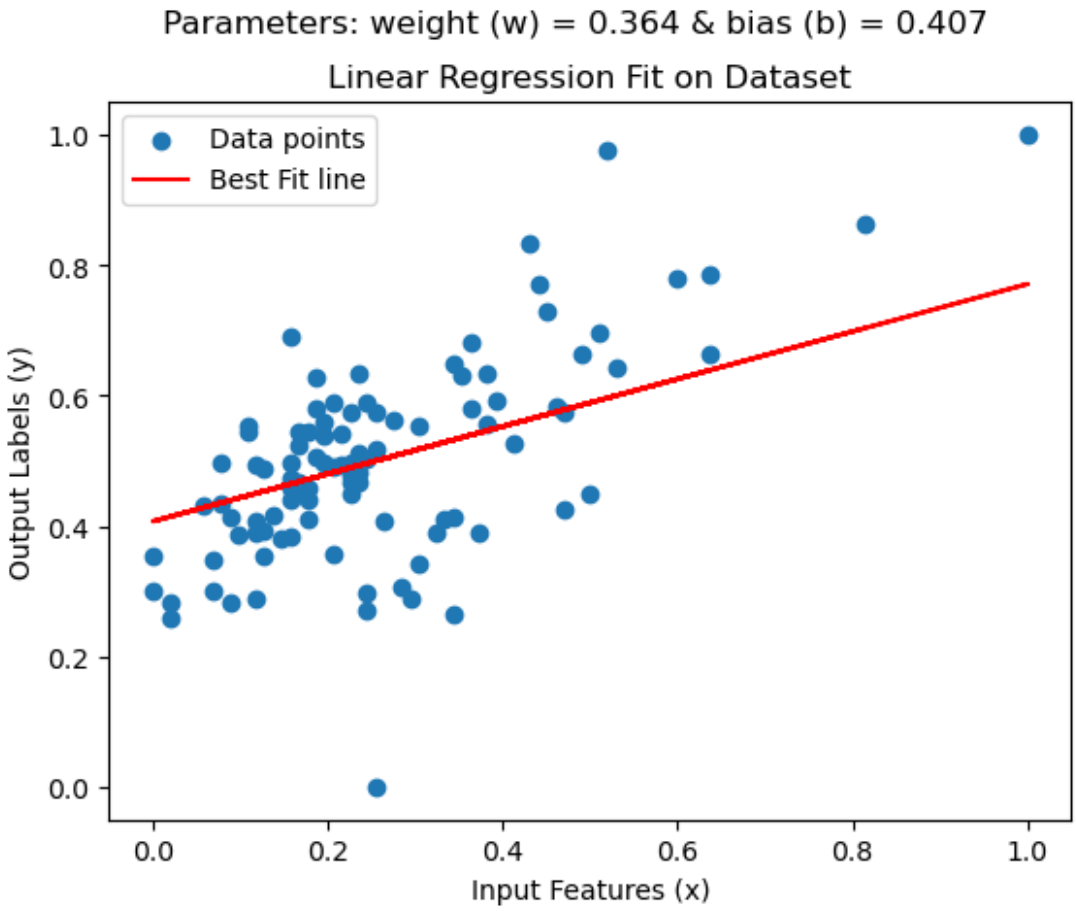
```
from models.vectorUtils import LinearRegression

model = LinearRegression()
lr = 0.5
model.fit(xNorm, yNorm, epochs=50, learning_rate=lr)
yHat = model.predict(10)
print("Prediction for input (10) =", yHat)
print(f"Final cost = {model.costFunc(xNorm.to_numpy(),
yNorm.to_numpy())}")
print(f"Parameters:\tw = {model.w}, b = {model.b}")
```

```
Prediction for input (10) = 4.045135392012695
Final cost = 0.007730905347767666
Parameters:     w = 0.3638003476515592, b = 0.4071319154971029
```

```
model.plotCost(epochs=50, subtitle=f"BGD with Learning Rate = {lr}")
```



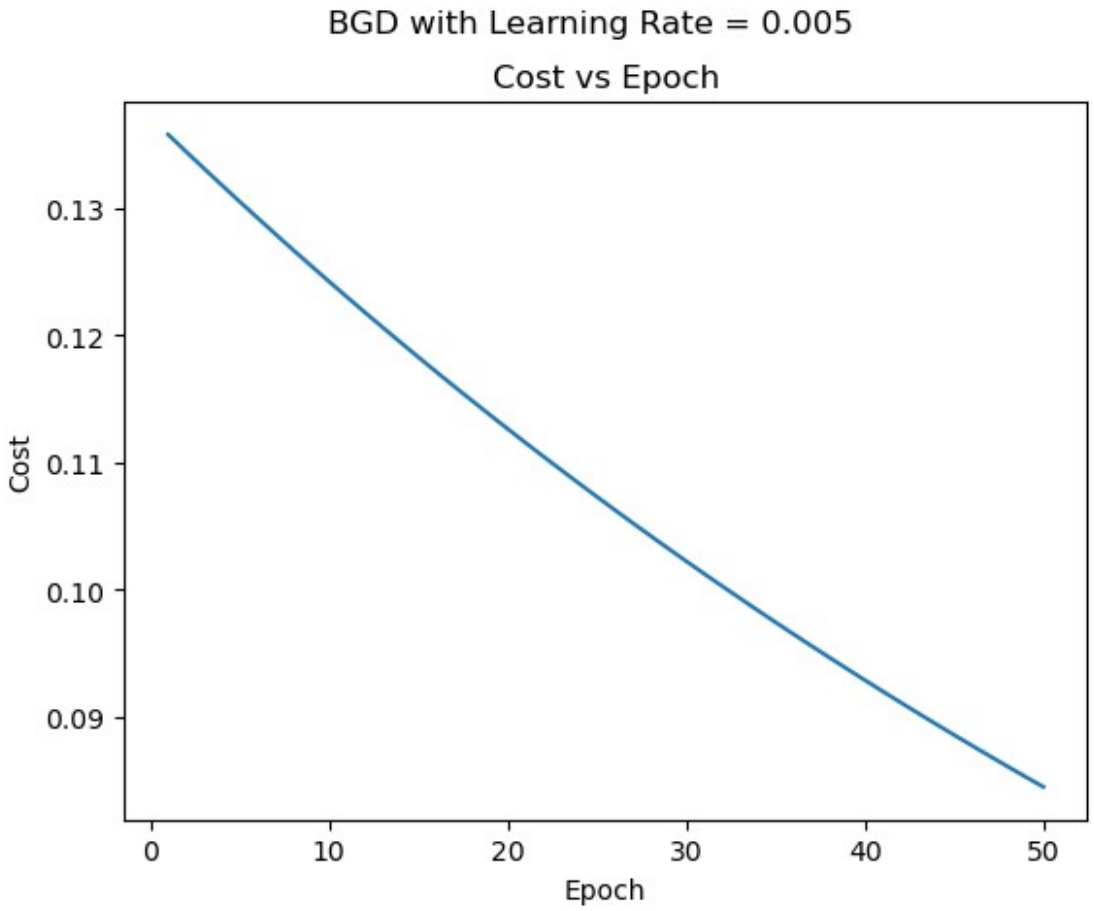

```
model.scatterPlot(xNorm, yNorm, subtitle=f"Parameters: weight (w) =
{model.w:.3f} & bias (b) = {model.b:.3f}")
```




```
lr = 0.005
model = LinearRegression()
model.fit(xNorm, yNorm, epochs=50, learning_rate=lr)
yHat = model.predict(10)
print("Prediction for input (10) =", yHat)
```

```
Prediction for input (10) = 0.43793493346817175
```

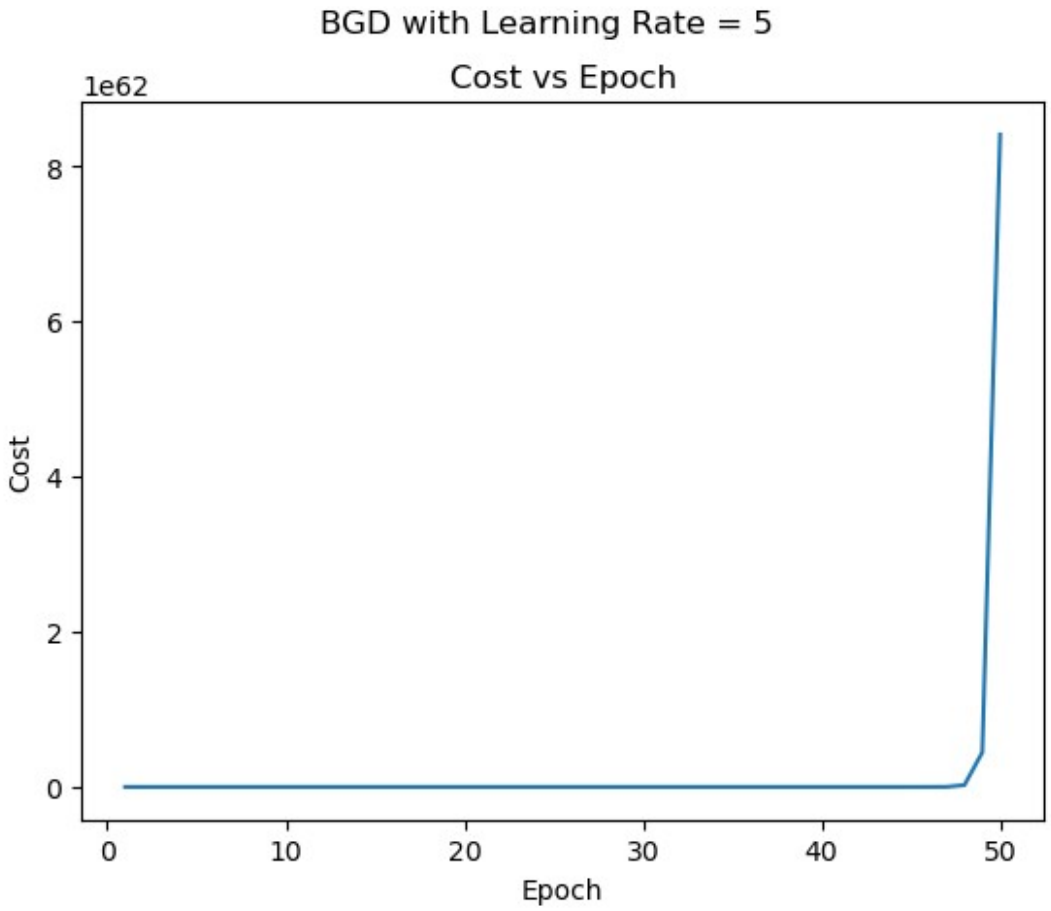
```
model.plotCost(epochs=50, subtitle=f"BGD with Learning Rate = {lr}")
```




```
lr = 5
model = LinearRegression()
model.fit(xNorm, yNorm, epochs=50, learning_rate=lr)
yHat = model.predict(10)
print("Prediction for input (10) =", yHat)
```

```
Prediction for input (10) = -1.4062656248784365e+32
```

```
model.plotCost(epochs=50, subtitle=f"BGD with Learning Rate = {lr}")
```



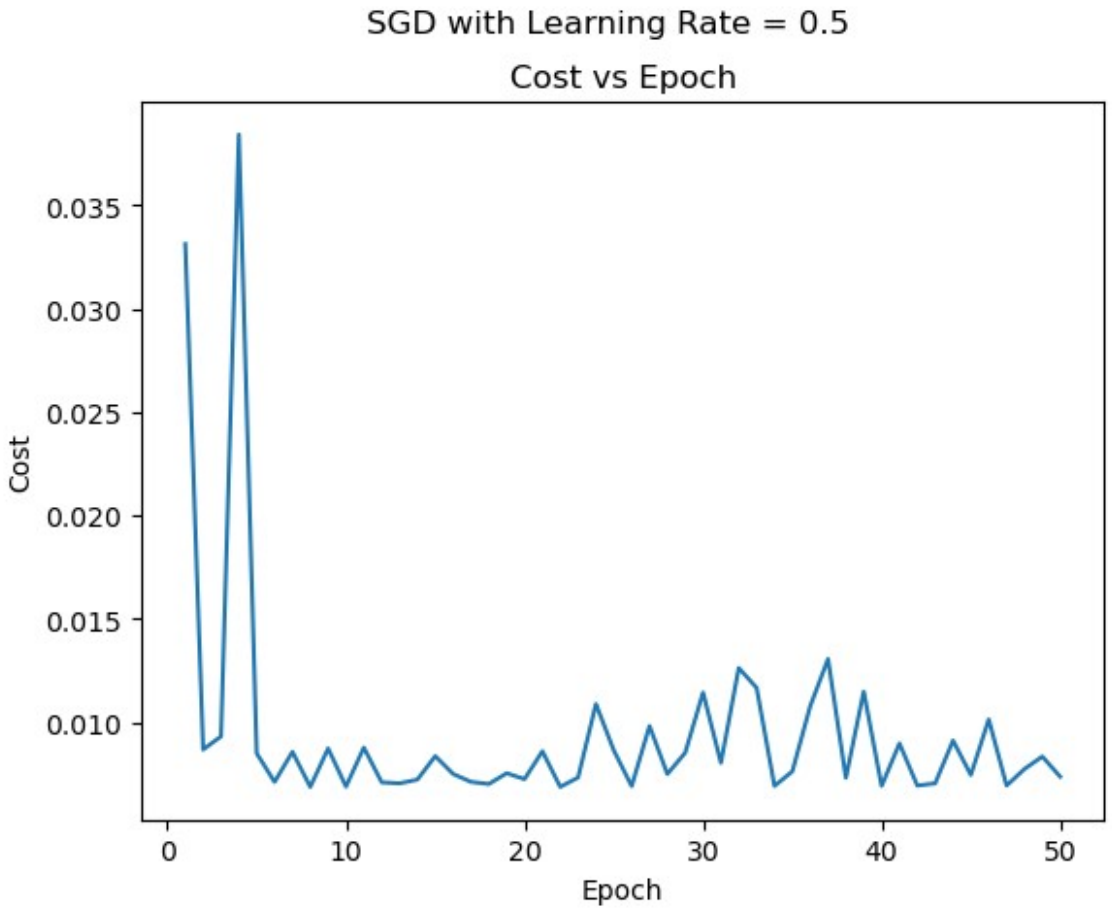
Stochastic Gradient Descent

Same learning rate as BGD

```
lr = 0.5
model = LinearRegression()
model.sgdFit(xNorm, yNorm, epochs=50, learning_rate=lr)
yHat = model.predict(10)
print("Prediction for input (10) =", yHat)
```

```
Prediction for input (10) = 5.7983831088447895
```

```
model.plotCost(epochs=50, subtitle=f"SGD with Learning Rate = {lr}")
```

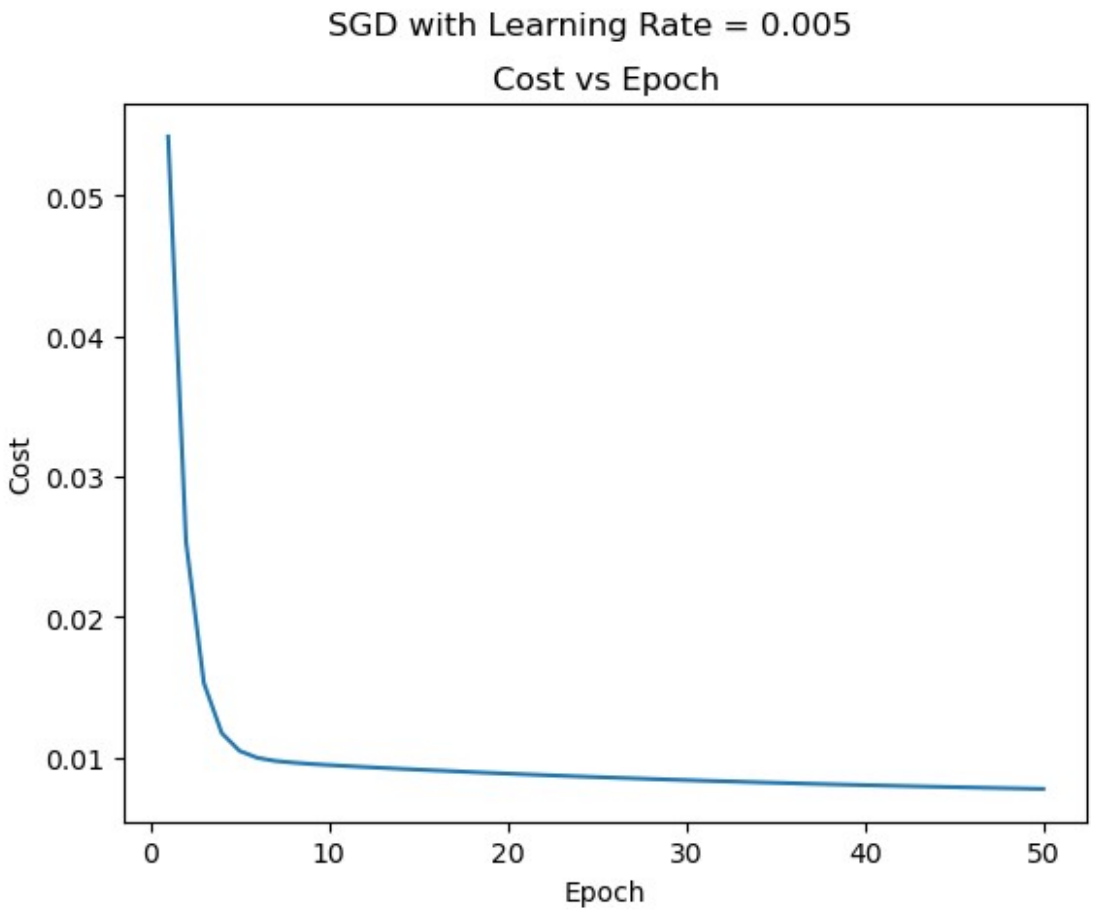


Optimal learning rate

```
lr = 0.005
model = LinearRegression()
model.sgdFit(xNorm, yNorm, epochs=50, learning_rate=lr)
yHat = model.predict(10)
print("Prediction for input (10) =", yHat)
```

```
Prediction for input (10) = 4.019389221470472
```

```
model.plotCost(epochs=50, subtitle=f"SGD with Learning Rate = {lr}")
```



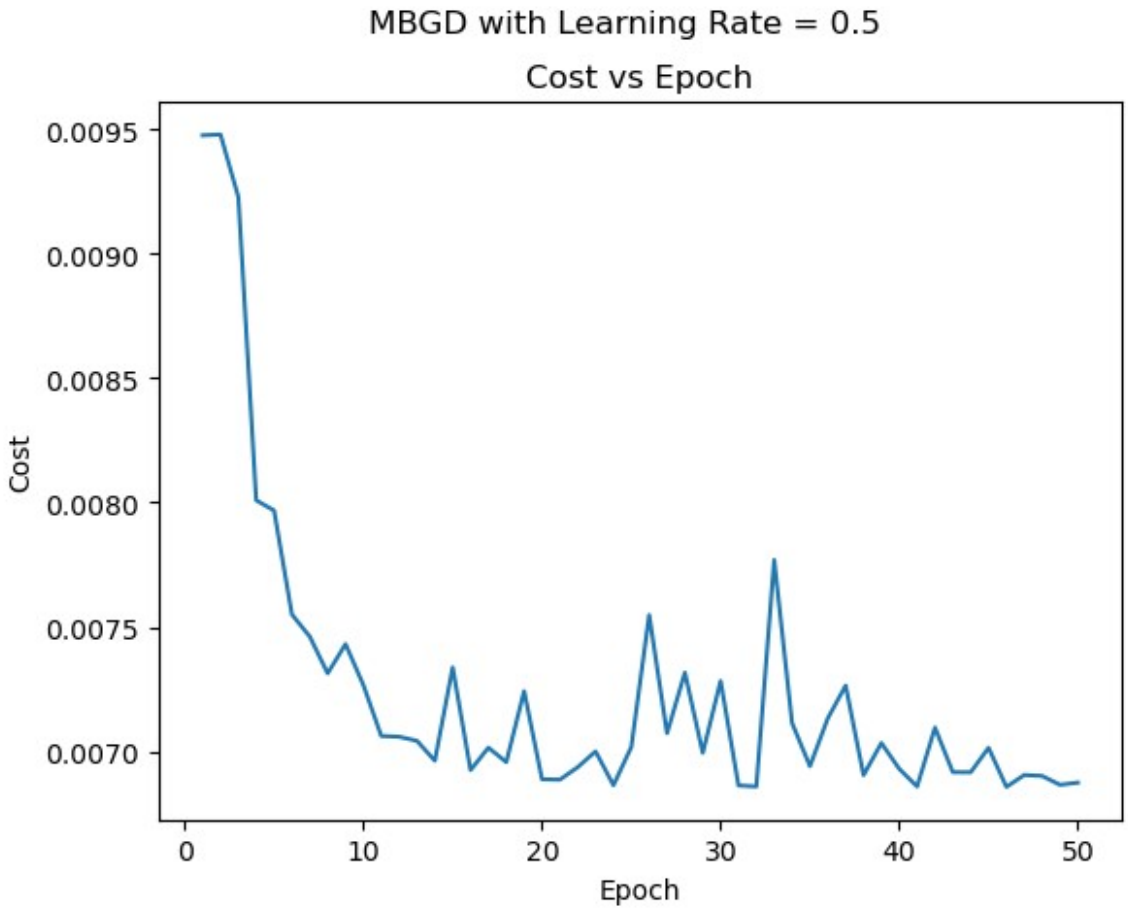
Mini-Batch Gradient Descent

Same Learning Rate as BGD

```
lr = 0.5
model = LinearRegression()
model.mbgdFit(xNorm, yNorm, epochs=50, batch_size=10,
learning_rate=lr)
yHat = model.predict(10)
print("Prediction for input (10) =", yHat)
```

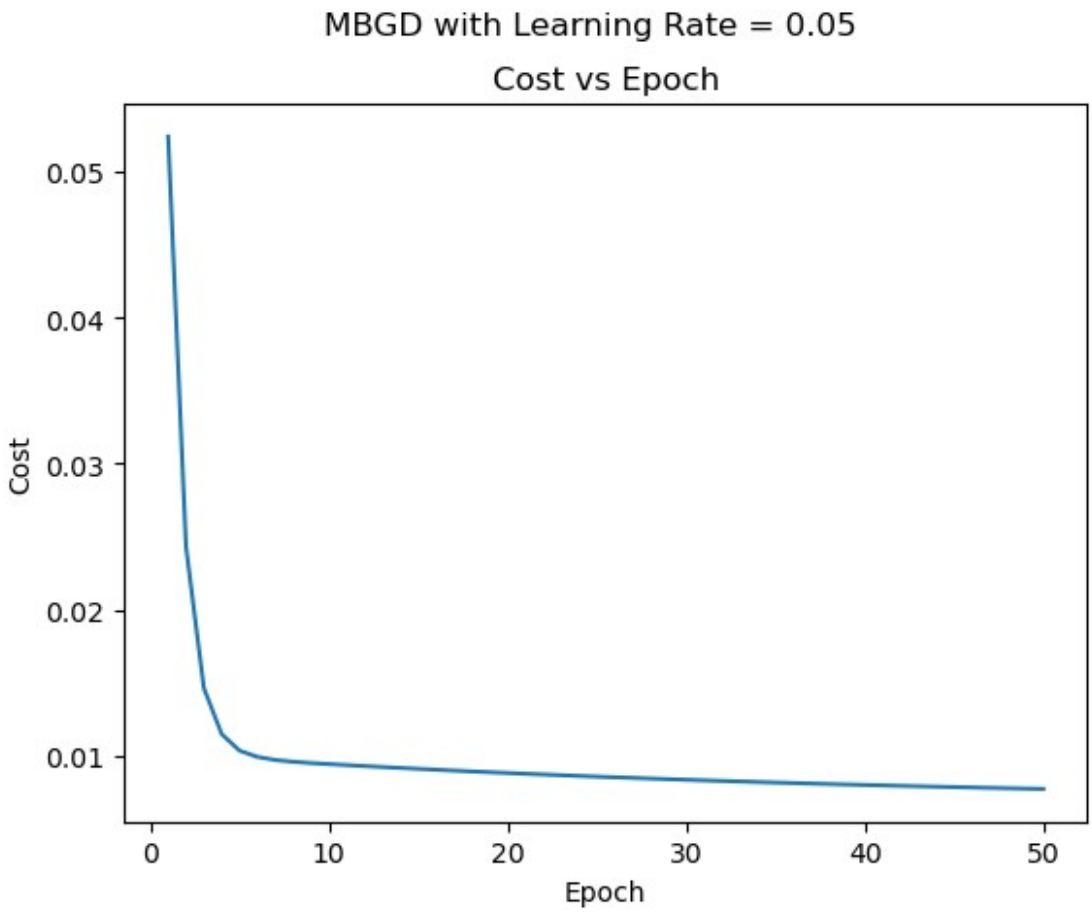
```
Prediction for input (10) = 6.369295609598814
```

```
model.plotCost(epochs=50, subtitle=f"MBGD with Learning Rate = {lr}")
```



Optimal learning rate

```
lr = 0.05
model = LinearRegression()
model.mbgdFit(xNorm, yNorm, epochs=50, batch_size=10,
learning_rate=lr)
yHat = model.predict(10)
print("Prediction for input (10) =", yHat)

Prediction for input (10) = 4.043960040429436

model.plotCost(epochs=50, subtitle=f"MBGD with Learning Rate = {lr}")
```


MBGD with Learning Rate = 0.05
Cost vs Epoch
Cost
0.05
0.04
0.03
0.02
0.01
0
10
20
30
40
50
Epoch