Assignment1

September 29, 2024

[124]:

```
import numpy as np
import matplotlib.pyplot as plt
import statistics
```

[125]:

```
prices = [8730,8781,9449,10224,10575,11070,11485,11845,11580,11960,12565,13645,
          14575,14610,14450,13970,14490,16395,17820,18160,24000,24110,24820,
          25980,27400,32720]
prices_max = [14840,16535,18328,19571,20295,20325,19695,19435,19785,24335,25010,
              25450,26015,26795,26670,24425,24350,25805,25800,26070,38565,38675,
              39035,39730,40945,55620]
years = [1992,1993,1994,1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,
         2006,2007,2008,2009,2010,2011,2019,2020,2021,2022,2023,2024]
years_train =␣
 ↪[1992,1993,1994,1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,
               2007,2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019,
               2020,2021,2022,2023,2024,2025]
```

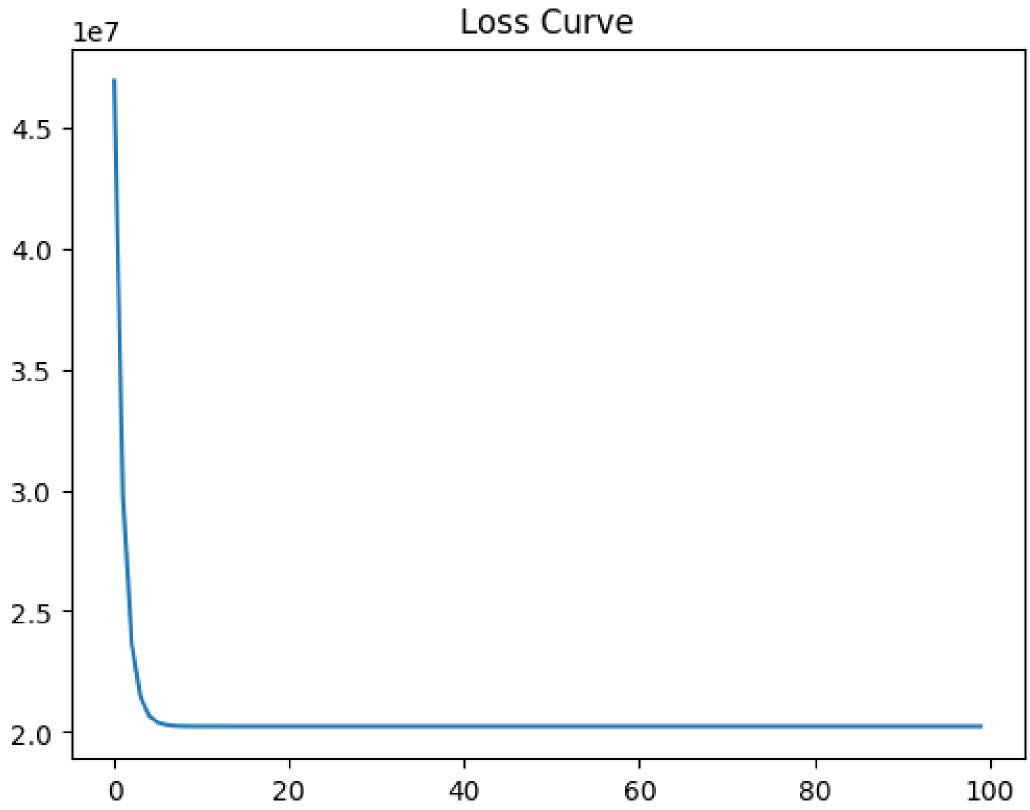
[126]:

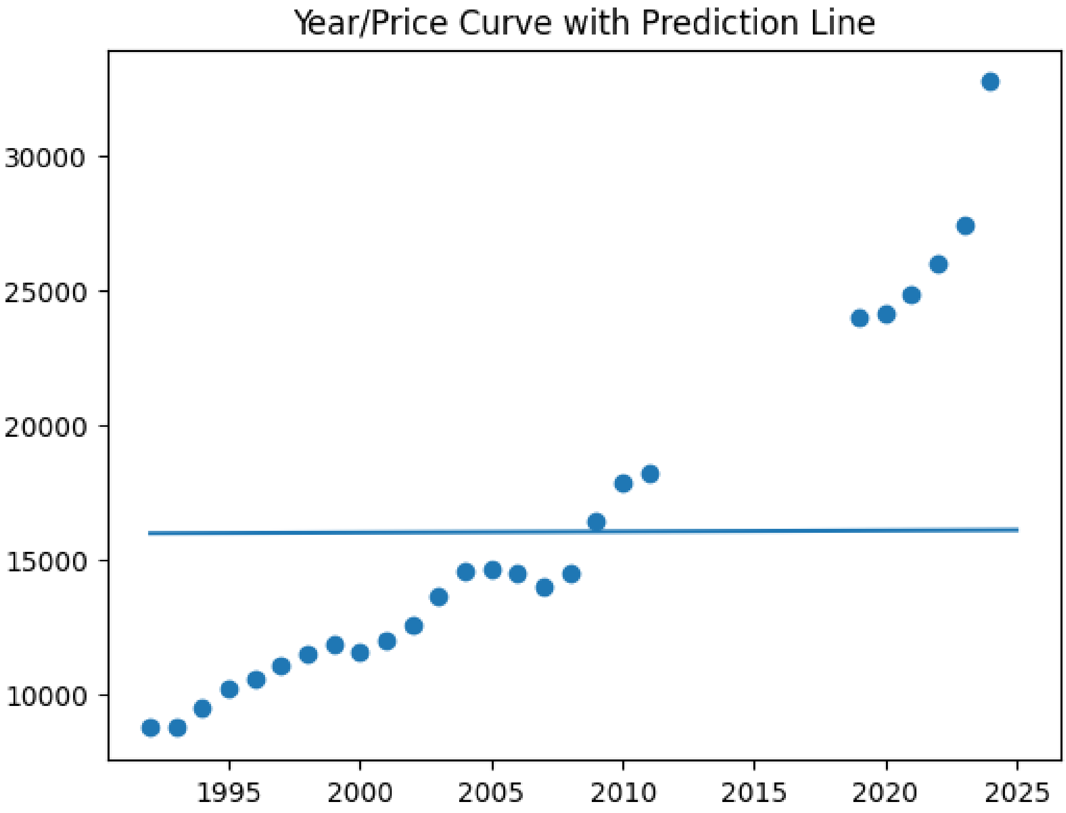
```
# model function
def f_theta(input_x):
  global theta_0
  global theta_1
  return theta_0 + input_x*theta_1

# cost function
def cost(theta_0, theta_1, inputs, outputs):
    error_sum = 0
    for idx, x in enumerate(inputs):
        error_sum += (f_theta(inputs[idx])-outputs[idx])**2
    return error_sum/(2*len(inputs))

# partial derivative calculations
def theta_0_partial(inputs, outputs):
    error_sum = 0
    for idx, x in enumerate(inputs):
      error_sum += (f_theta(inputs[idx])-outputs[idx])
    return error_sum/len(inputs)
def theta_1_partial(inputs, outputs):
    error_sum = 0
    for idx, x in enumerate(inputs):
      error_sum += (f_theta(inputs[idx])-outputs[idx]) * inputs[idx]
    return error_sum/len(inputs)

# update rule
def update_weights(inputs, outputs):
    global theta_0
    #print("Theta_0 partial: ", theta_0_partial())
    theta_0 = theta_0 - l_rate*theta_0_partial(inputs, outputs)
    global theta_1
    #print("Theta_1 partial: ", theta_1_partial())
    theta_1 = theta_1 - l_rate*theta_1_partial(inputs, outputs)

# plot curves and report final weights/predictions
def report(loss_in, theta_0_in, theta_1_in, prices_pd_in, prices_in):
    plt.figure()
    plt.title("Loss Curve")
    plt.plot(loss_in)
    print("Final weights: theta_0: ",theta_0_in, " theta_1: ", theta_1_in)
    print("Prediction: 2012-$", prices_pd_in[20]," 2013-$", prices_pd_in[21],"␣
 ↪2014-$", prices_pd_in[22]," 2015-$", prices_pd_in[23])
    print(" 2016-$", prices_pd_in[24]," 2017-$", prices_pd_in[25]," 2018-$",␣
 ↪prices_pd_in[26], " 2025-$", prices_pd_in[33])
    plt.figure()
    plt.title("Year/Price Curve with Prediction Line")
    plt.scatter(years, prices_in)
    plt.plot(years_train, prices_pd_in)
```

[127]:

```
# no feature scaling/dynamic learning rate
l_rate = .0000001
theta_0 = 8700
theta_1 = 0

# minimum price

# training
loss = []
ind = 0
while ind < 100:
  loss.append(cost(theta_0,theta_1,years,prices))
  update_weights(years, prices)
  ind += 1

# generating predictions
prices_pd = []
for x in years_train:
    prices_pd.append(f_theta(x))

print("No FS/DLR: Minimum Price")
report(loss, theta_0, theta_1, prices_pd, prices)
```

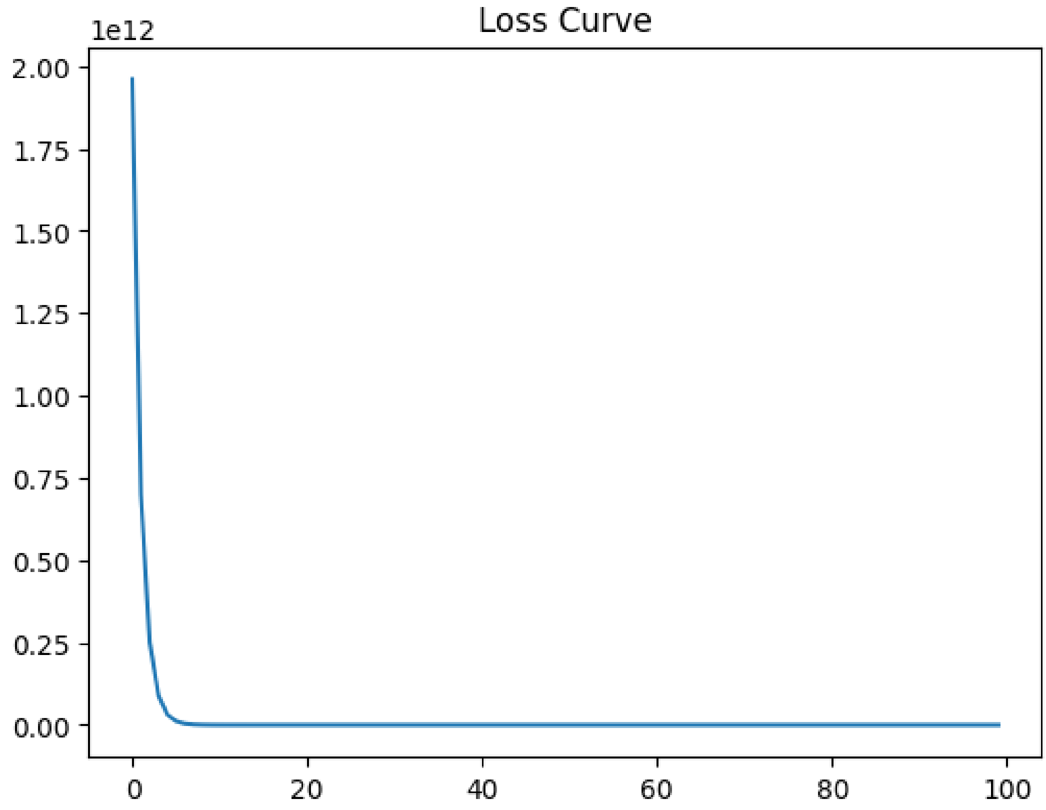
```
No FS/DLR: Minimum Price
Final weights: theta_0:  8700.001512297235  theta_1:  3.6426677425105387
Prediction: 2012-$ 16029.049010228438  2013-$ 16032.69167797095  2014-$
16036.33434571346  2015-$ 16039.97701345597
 2016-$ 16043.619681198481  2017-$ 16047.26234894099  2018-$ 16050.9050166835
2025-$ 16076.403690881076
```

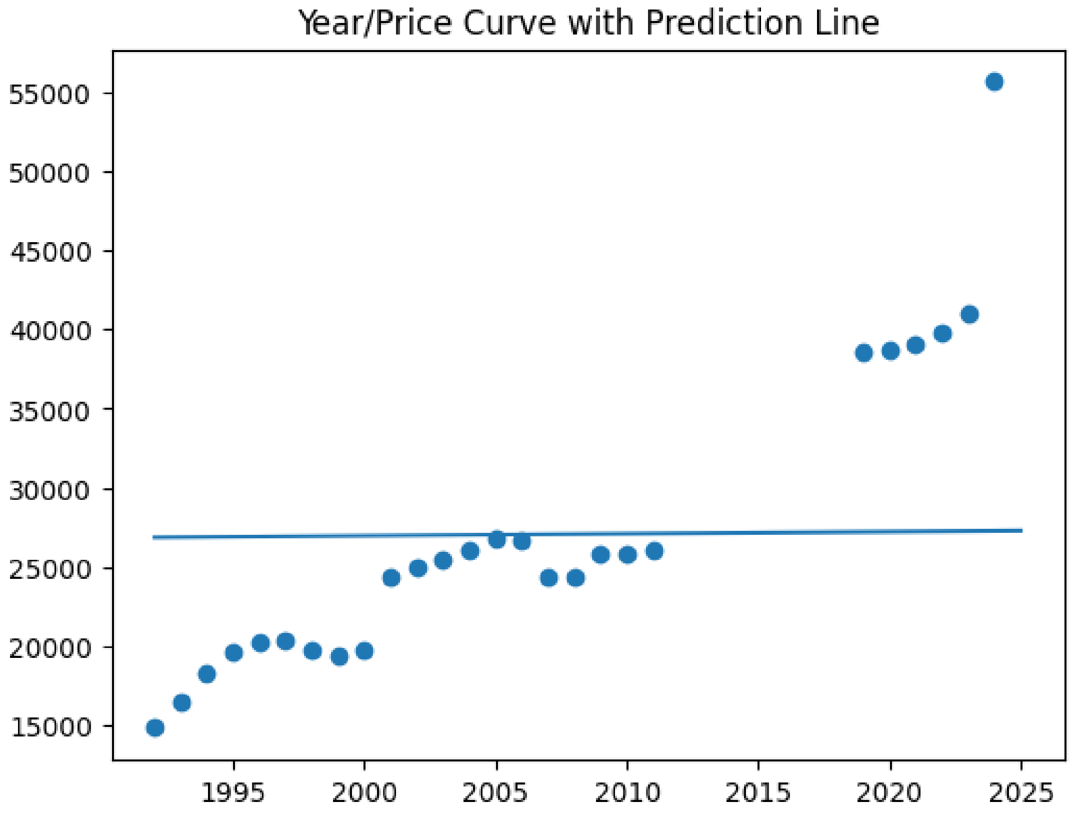




[128]:
```
# maximum price

l_rate = .0000001
theta_0 = 1000
theta_1 = 1000

# training
loss = []
ind = 0
while ind < 100:
  loss.append(cost(theta_0,theta_1,years,prices_max))
  update_weights(years, prices_max)
  ind += 1

# generating predictions
prices_pd = []
for x in years_train:
    prices_pd.append(f_theta(x))

print("No FS/DLR: Maximum Price")
report(loss, theta_0, theta_1, prices_pd, prices_max)
```

```
No FS/DLR: Maximum Price
Final weights: theta_0:  999.5075865549154  theta_1:  12.983620883919087
Prediction: 2012-$ 27122.55280500012  2013-$ 27135.536425884038  2014-$
27148.520046767957  2015-$ 27161.503667651876
 2016-$ 27174.487288535795  2017-$ 27187.470909419713  2018-$ 27200.454530303632
2025-$ 27291.339876491067
```



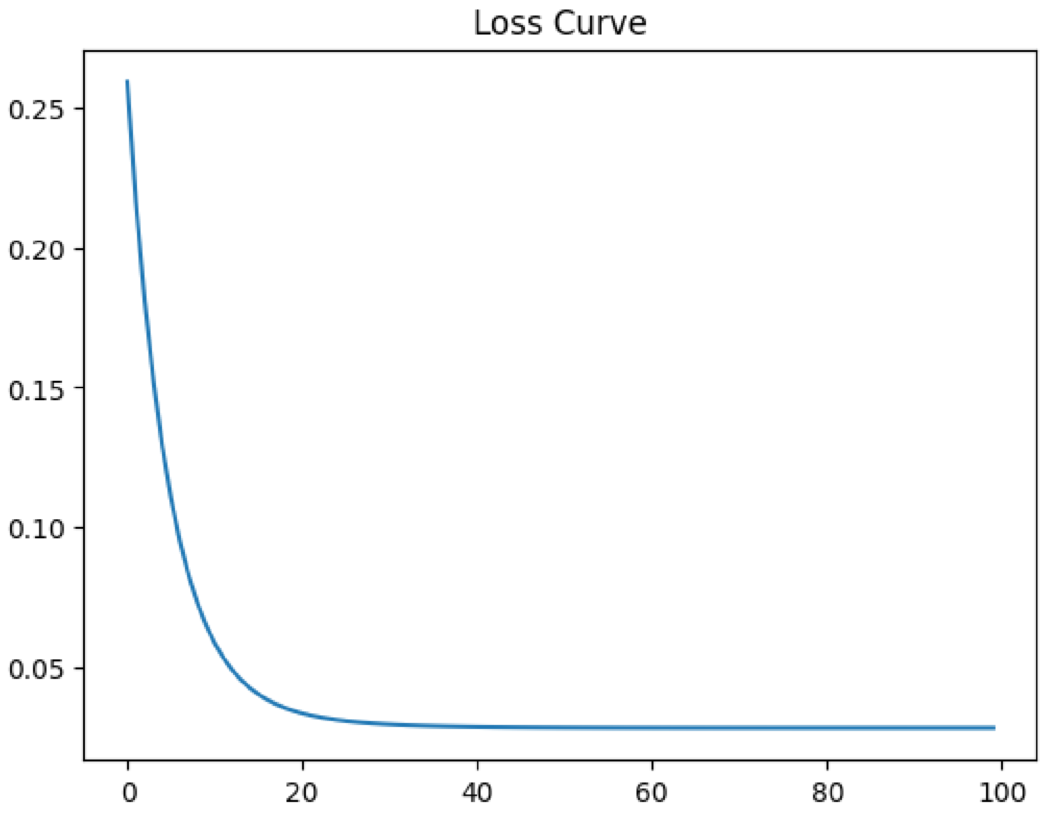


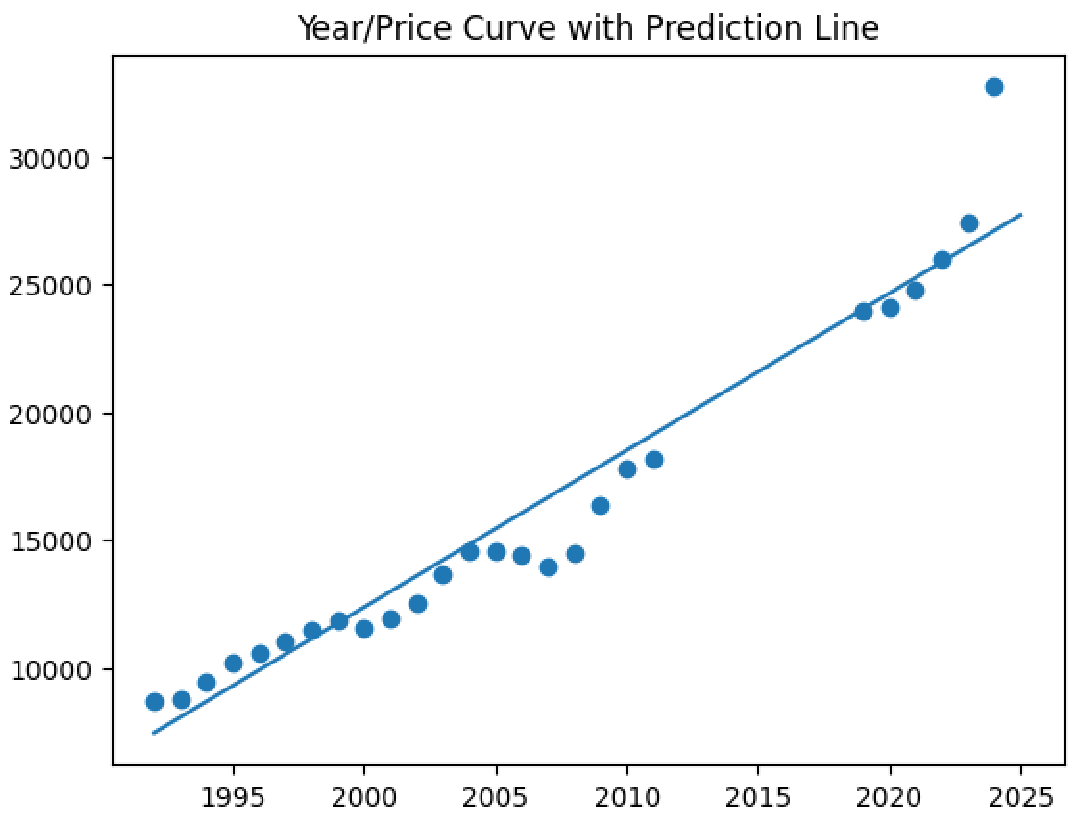
[129]:

```
# with feature scaling
theta_0 = .5
theta_1 = .5
l_rate = .1

# normalizing years
years_sc = []
years_mean = statistics.mean(years)
years_stdev = statistics.stdev(years)
for x in years:
    years_sc.append((x-years_mean)/years_stdev)

# normalizing minimum prices
prices_sc = []
for x in prices:
    prices_sc.append((x-statistics.mean(prices))/statistics.stdev(prices))

# training
loss2 = []
c = 500
it = 0
```

```
while it < 100:
    loss2.append(cost(theta_0, theta_1, years_sc, prices_sc))
    update_weights(years_sc, prices_sc)
    l_rate = (l_rate*c)/(c+it) # implementation of dynamic learning rate
    it += 1

# generating predictions
# f_theta(normalized_year) = x, x is normalized price prediction
# prices_pd[x] = stdev(prices)*prices_sc[x] + mean(prices)
prices_pd = []
for x in years_train:
    price_pd_norm = f_theta((x-years_mean)/years_stdev)
    prices_pd.append(statistics.stdev(prices)*price_pd_norm + statistics.
 ↪mean(prices))

print("With FS/DLR: Minimum Price")
report(loss2, theta_0, theta_1, prices_pd, prices)
```

```
With FS/DLR: Minimum Price
Final weights: theta_0:  0.023741437770232677  theta_1:  0.9455559879859002
Prediction: 2012-$ 19740.46752451299  2013-$ 20353.652427750774  2014-$
20966.837330988557  2015-$ 21580.022234226337
 2016-$ 22193.207137464124  2017-$ 22806.392040701903  2018-$ 23419.576943939686
2025-$ 27711.87126660416
```

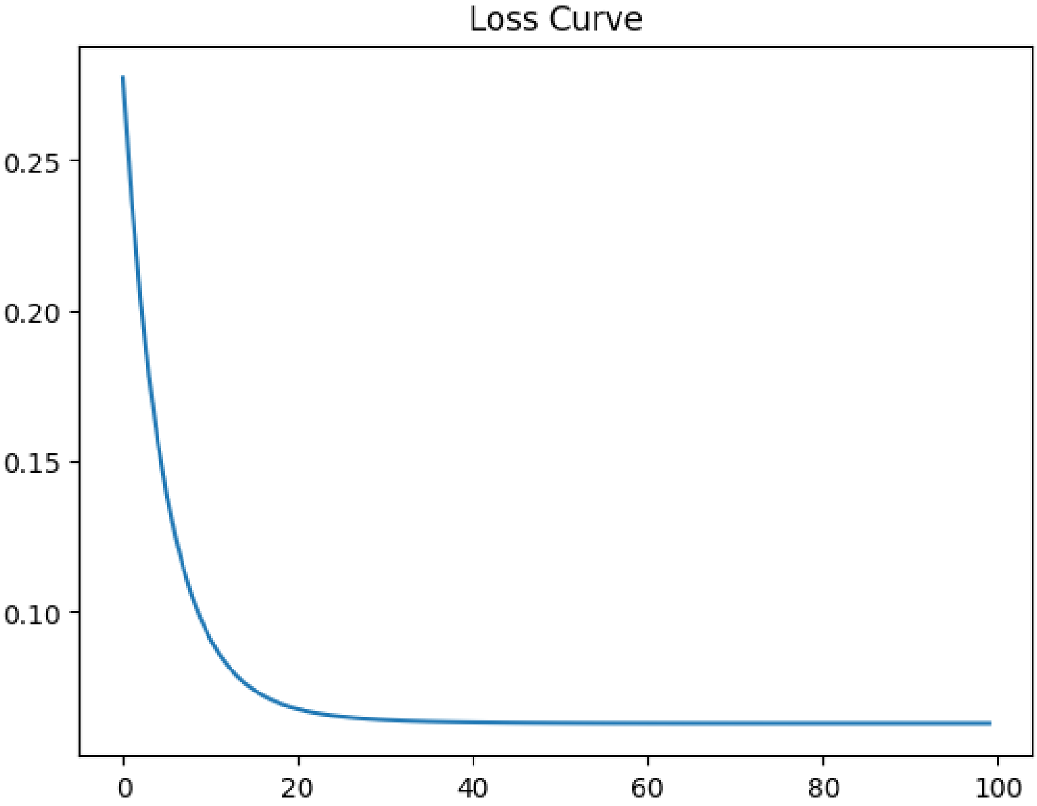

Loss Curve
0.25
0.20
0.15
0.10
0.05
0
20
40
60
80
100

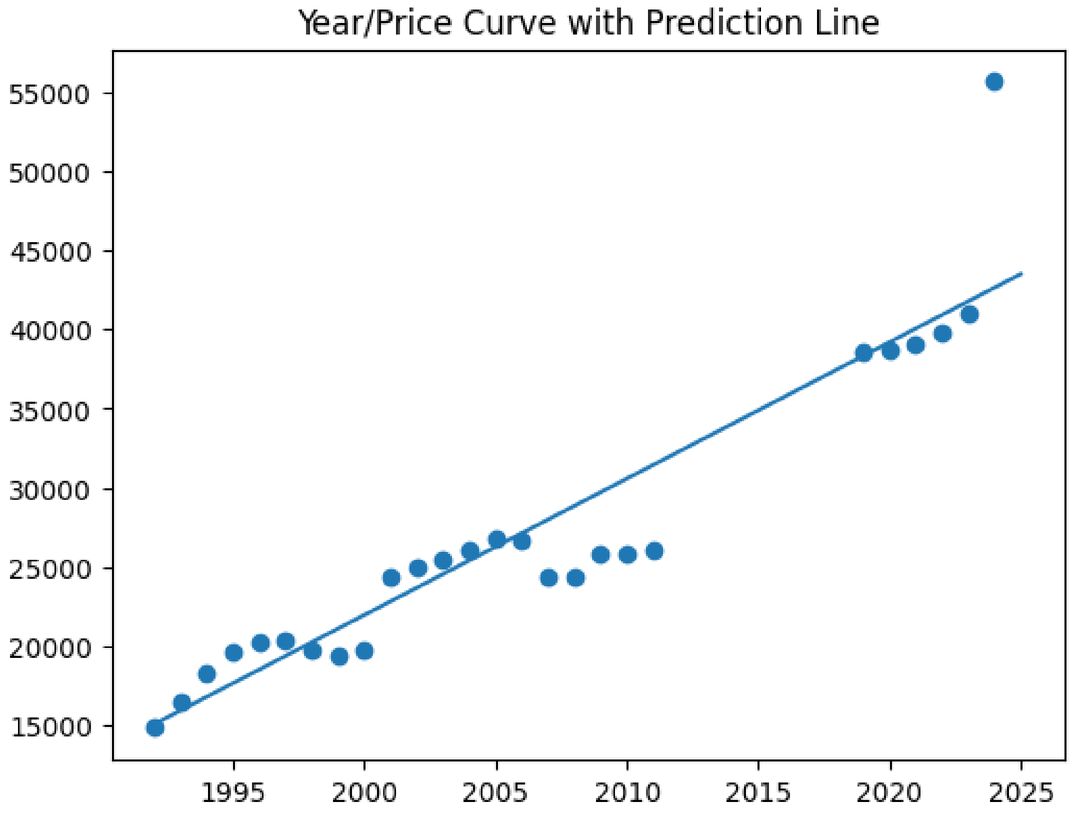


[130]:

```
# maximum price
l_rate = .1
theta_0 = .5
theta_1 = .5

prices_m_sc = []
for x in prices_max:
    prices_m_sc.append((x-statistics.mean(prices_max))/statistics.
 ↪stdev(prices_max))

# training
loss3 = []
c = 500
it = 0
while it < 100:
    loss3.append(cost(theta_0, theta_1, years_sc, prices_m_sc))
    update_weights(years_sc, prices_m_sc)
    l_rate = (l_rate*c)/(c+it) #implementation of dynamic learning rate
    it += 1

# generating predictions
# f_theta(normalized_year) = x, x is normalized price prediction
# prices_pd[x] = stdev(prices)*prices_sc[x] + mean(prices)
prices_m_pd = []
for x in years_train:
    price_pd_norm = f_theta((x-years_mean)/years_stdev)
    prices_m_pd.append(statistics.stdev(prices_max)*price_pd_norm + statistics.
  ↪mean(prices_max))

print("With FS/DLR: Maximum Price")
report(loss3, theta_0, theta_1, prices_m_pd, prices_max)
```

```
With FS/DLR: Maximum Price
Final weights: theta_0:  0.023741437770232 93  theta_1:  0.9096663286815427
Prediction: 2012-$ 32293.255912580826  2013-$ 33153.70362509567  2014-$
34014.15133761052  2015-$ 34874.59905012537
 2016-$ 35735.04676264022  2017-$ 36595.494475155065  2018-$ 37455.94218766991
2025-$ 43479.07617527385
```





Summary

My model has found a predicted price range of \$27711.87-\$43479.08 for the 2025 Ranger. The final weight values have been stated above for all four models. I believe that my last two models (with FS/DLR) accurately model the data, as evidenced by their year-price plots and my model's prediction line. The models with FS/DLR produced much more accurate results than the models without FS/DLR. My model could be improved by selecting different starting weights, a different normalization method, or a different value of the constant c for DLR. I would not buy a Ford Ranger at the predicted price range.