Assignment I - Rick Kabuto

1. Get the Data (Price of a Ford Ranger: 1992–2025)

- Source: Cars.com - Ford Ranger Research

2. Linear Regression – Model Training (10 points: 5 points per plot)

- Train 2 separate linear regression models:
  - One for minimum price
  - One for maximum price
- Plot loss curves for each model over 100 iterations.
- Question: What are your final model weights after training?

3. Price Prediction for Missing Years (2012–2018) (10 points: 5 points per plot)

- Since prices are missing from 2012–2018, predict them using your models as if Ford had produced the Ranger during those years.
- Plot price-vs-year curves:
  - One for min. price
  - One for max. price

4.. Feature Scaling + Dynamic Learning Rate (20 points: 5 points per plot)

- Repeat steps 2 & 3 but with:
  - Feature scaling
  - Dynamic learning rate
- Question: Did these adjustments improve the results? Why or why not?

5. Summary Paragraph Address all of the following in your PDF:

- What is the predicted price of a 2026 Ford Ranger? Would you consider buying it?
- Reflect on each model you trained in steps 2 & 4:
  - Your thoughts on their performance
  - What could be done to improve them?

Step 1) 1992-2025 Ford Ranger Data For Prices Of Car

1992 Ranger - $8,730–$14,840 ***1993 Ranger*** - $8,781–$16,535 ***1994 Ranger*** - $9,449–$18,328
1995 Ranger - $10,224–$19,571 ***1996 Ranger*** - $10,575–$20,295 ***1997 Ranger*** - $11,070–$20,325
1998 Ranger - $11,485–$19,695 ***1999 Ranger*** - $11,845–$19,435 ***2000 Ranger*** - $11,580–$19,785
2001 Ranger - $11,960–$24,335 ***2002 Ranger*** - $12,565–$25,010 ***2003 Ranger*** - $13,645–$25,450
2004 Ranger -$14,575–$26,015 ***2005 Ranger*** - $14,610–$26,795 ***2006 Ranger*** - $14,450–$26,670
2007 Ranger - $13,970–$24,425 ***2008 Ranger*** - $14,490–$24,350 ***2009 Ranger*** - $16,395–$25,805
2010 Ranger - $17,820–$25,800 ***2011 Ranger*** - $18,160–$26,070 ***2012 Ranger* - N/A**
2013 Ranger* - N/A *2014 Ranger* - N/A *2015 Ranger - N/A ***2016 Ranger* - N/A**
2017 Ranger* - N/A 2018 Ranger - N/A** ***2019 Ranger - $24,000–$38,565
2020 Ranger - $24,110–$38,675 **2021 Ranger** - $24,820–$39,035 ***2022 Ranger*** - $25,980–$39,730
2023 Ranger - $27,400–$40,945 ***2024 Ranger*** - $32,820–$55,720 ***2025 Ranger*** - $33,330–$56,070

Step 2 - Linear Regression – Model Training

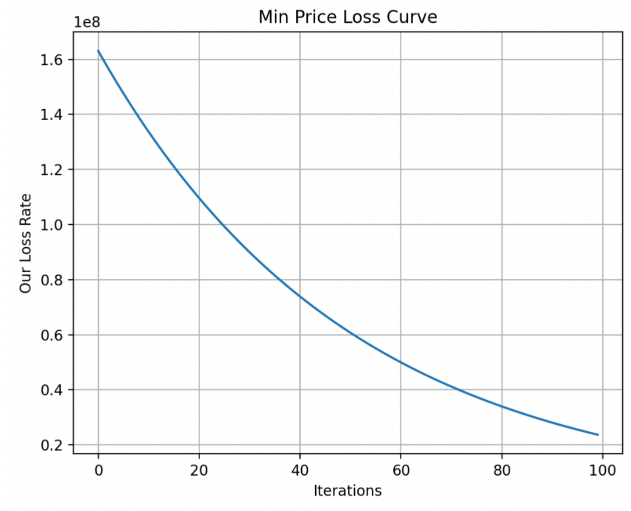
Part2-LinearRegression.py

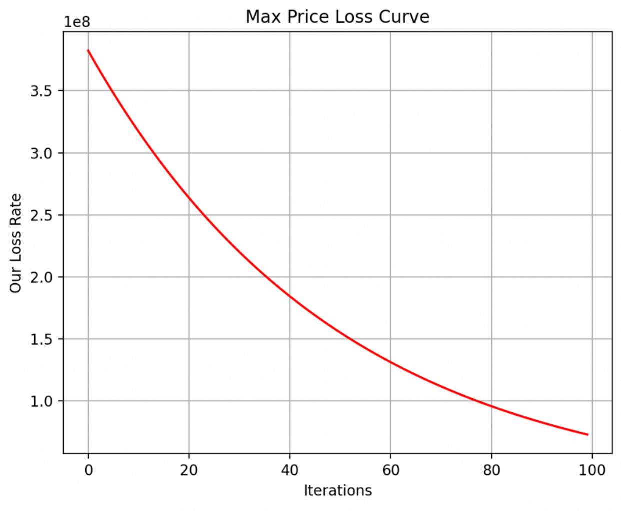
```
import numpy as np
import matplotlib.pyplot as plt

FordYears = np.array([
  1992,1993,1994,1995,1996,1997,1998,1999,2000,
  2001,2002,2003,2004,2005,2006,2007,2008,2009,
  2010,2011,2019,2020,2021,2022,2023,2024,2025
])

Lowestprice = np.array([
  8730,8781,9449,10224,10575,11070,11485,11845,11580,
  11960,12565,13645,14572,14160,14450,13970,14490,16395,
  17820,18160,24000,24110,24820,25980,27400,32820,33330
])

Highestprice = np.array([
  14840,16535,18328,19571,20295,20325,19695,19435,19785,
  24,335,25010,25450,26015,26795,26670,24425,24350,25805,
  25800,26070,38565,38675,39035,39730,40945,55720,56070
])

def LinearRegression(a, b, learning=0.01, iterations = 100):
  c = len(a)
  theta = [0.0 for _ in range(len(a[0]))]
  history = []
  d = 0
  while d < iterations:
     predictions = []
     for i in range(c):
        pred = 0.0
        for j in range(len(theta)):
           pred += theta[j] * a[i][j]
        predictions.append(pred)
     errorcounter = [predictions[i] - b[i] for i in range(c)]
     miss = sum(error ** 2 for error in errorcounter) / (2*c)
     history.append(miss)
     #-------------------------------------------------------
     #Train 2 separate linear regression models:
     #One for minimum price
     #One for maximum price
     #-------------------------------------------------------
```

```
        grad = [0.0 for _ in range(len(theta))]
        for j in range(len(theta)):
            for i in range(c):
                grad[j] += errorcounter[i] * a[i][j]
            grad[j] /= c
        for j in range(len(theta)):
            theta[j] -= learning * grad[j]
        d += 1
    return np.array(theta), history
    #End of Linear Regression Function

#Plot Loss Curves over 100 iterations
#Our Function To Graph Our Results
def plot_losses(TrueLoss, TrueMax):
    fig, axes = plt.subplots(1, 2, figsize=(12,5))
    axes[0].plot(TrueLoss, label = 'The Model For Min Price')
    axes[0].set_title('Min Price Loss Curve')
    axes[0].set_xlabel('Iterations')
    axes[0].set_ylabel('Our Loss Rate')
    axes[0].grid(True)
    axes[1].plot(TrueMax, label = 'The Model for max Price', color='red')
    axes[1].set_title('Max Price Loss Curve')
    axes[1].set_xlabel('Iterations')
    axes[1].set_ylabel('Our Loss Rate')
    axes[1].grid(True)
    fig.tight_layout()
    plt.show()
```

Main.py

```
import numpy as np
from DataAndFunctions import FordYears, Lowestprice, Highestprice, LinearRegression, plot_losses
#------------------------------------------------------
#Question 2 - Linear Regression – Model Training
x = FordYears.reshape(-1, 1)
Normalization = (x - x.mean()) / x.std()
y = np.c_[np.ones(x.shape[0]), Normalization]

MinimumTheta, TrueLoss = LinearRegression(y, Lowestprice)
MaximumTheta, TrueMax = LinearRegression(y, Highestprice)
plot_losses(TrueLoss, TrueMax)
print(f"Our final model weights after training")
print(f"Min Price Intercept: {MinimumTheta[0]:.2f}, slope = {MinimumTheta[1]:.2f}")
print(f"Max Price Intercept: {MaximumTheta[0]:.2f}, slope = {MaximumTheta[1]:.2f}")
#------------------------------------------------------
```





Question: What are your final model weights after training?

Min Price Intercept: 10538.87, slope = 4346.51

Max Price Intercept: 17804.49, slope = 6287.02

Part 3 - Price Prediction for Missing Years (2012–2018) (10 points: 5 points per plot)

Note - This code is extended from Part 2, but I am only showing the code relevant to the question

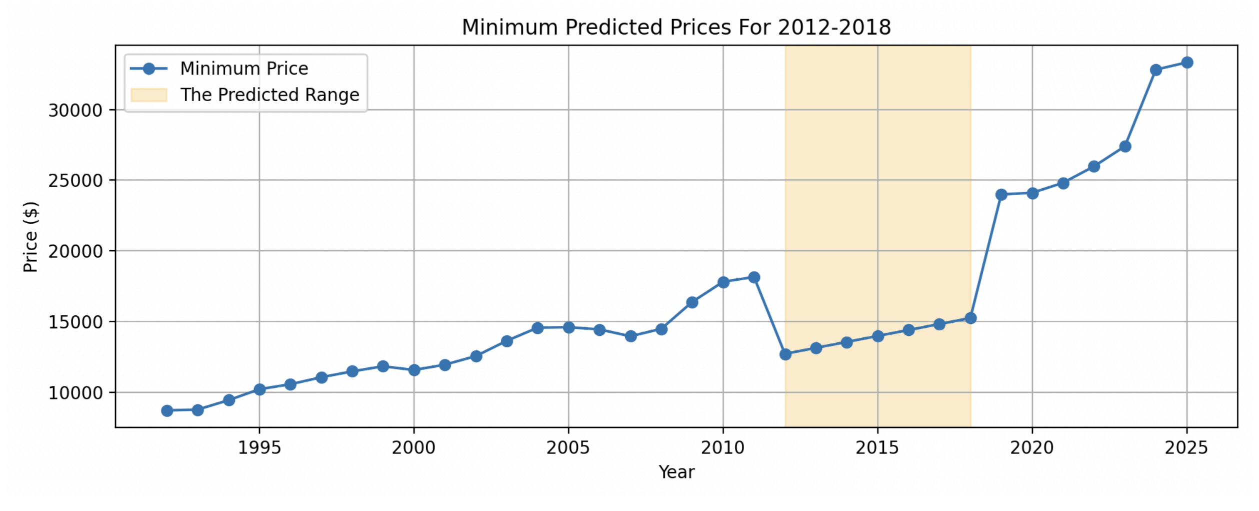
Part3_PricePrediction.py

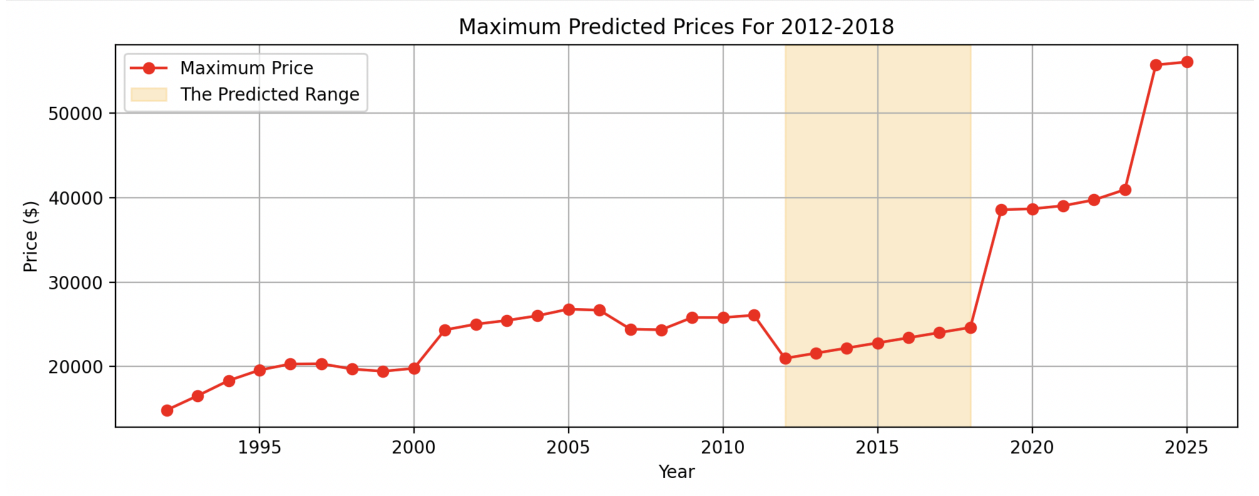
```
#-------------------------------------------------------------------------------------------------------------------
#Price Prediction for Missing Years (2012–2018) (10 points: 5 points per plot)
#Since prices are missing from 2012–2018, predict them using your models as if Ford had produced the
Ranger during those years.
#Plot price-vs-year curves:
#One for min. price
#One for max. price
#-------------------------------------------------------------------------------------------------------------------
import numpy as np
import matplotlib.pyplot as plt

def predict_missing(FordYears, Lowestprice, Highestprice, MinimumTheta, MaximumTheta, x):
  EmptyYears = np.array([2012,2013,2014,2015,2016,2017,2018,])
  Average = x.mean()
  Standard = x.std()

  EmptyNormalization = []
  for year in EmptyYears:
     EmptyNormalization.append([(year - Average) / Standard])
  MissingYs = []
  for row in EmptyNormalization:
    MissingYs.append([1.0] + row)
MissingYs = np.array(MissingYs)

LowestPrediction = []
HighestPrediction = []
for row in MissingYs:
    low_sum = 0
    high_sum = 0
    for i in range(len(row)):
        low_sum += row[i] * MinimumTheta[i]
        high_sum += row[i] * MaximumTheta[i]
    LowestPrediction.append(low_sum)
    HighestPrediction.append(high_sum)

CompleteYears = list(FordYears) + list(EmptyYears)
CompleteLowestPrices = list(Lowestprice) + LowestPrediction
CompleteMaximumPrices = list(Highestprice) + HighestPrediction

SortedArray = sorted(range(len(CompleteYears)), key=lambda k: CompleteYears[k])
SortedYears = [CompleteYears[i] for i in SortedArray]
CompleteLowestPrices = [CompleteLowestPrices[i] for i in SortedArray]
CompleteMaximumPrices = [CompleteMaximumPrices[i] for i in SortedArray]

#Construct The Graph
#Min Prices
fig1, ax1 = plt.subplots(figsize =(10,4))
ax1.plot(SortedYears, CompleteLowestPrices, marker = 'o', label = 'Minimum Price')
ax1.axvspan(2012, 2018, color='orange', alpha = 0.2, label = 'The Predicted Range')
ax1.set_title('Minimum Predicted Prices For 2012-2018')
ax1.set_xlabel('Year')
ax1.set_ylabel('Price ($)')
ax1.grid(True)
ax1.legend()
fig1.tight_layout()
plt.show()
#Max Prices
fig2, ax2 = plt.subplots(figsize =(10,4))
ax2.plot(SortedYears, CompleteMaximumPrices, marker = 'o', color = 'red', label = 'Maximum Price')
ax2.axvspan(2012, 2018, color='orange', alpha = 0.2, label = 'The Predicted Range')
ax2.set_title('Maximum Predicted Prices For 2012-2018')
ax2.set_xlabel('Year')
ax2.set_ylabel('Price ($)')
ax2.grid(True)
ax2.legend()
```

```
fig2.tight_layout()
plt.show()
```

Main.py

```
predict_missing(FordYears, Lowestprice, Highestprice, MinimumTheta, MaximumTheta, x)
```





Part 4 - Feature Scaling + Dynamic Learning Rate (20 points: 5 points per plot)
Note - This code is extended from Part 2 and 3, but I am only showing the code relevant to the question

Question: Did these adjustments improve the results? Why or why not? - Yes, the adjustments improved the results. The reason that it improved the results is because the dynamic learning model takes larger steps towards the beginning of the process since the learning rate is larger. This causes the graph to converge faster giving more reasonable and predictable results for the expected price. Overall, we produce a smoother graph without the loss of our accuracy in prediction.

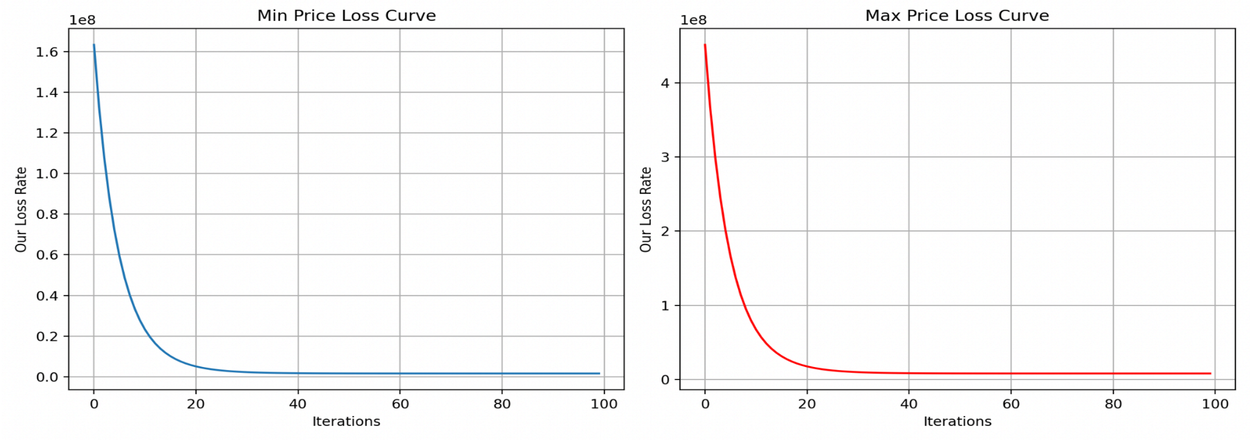
Part4_DynamicLearning.py

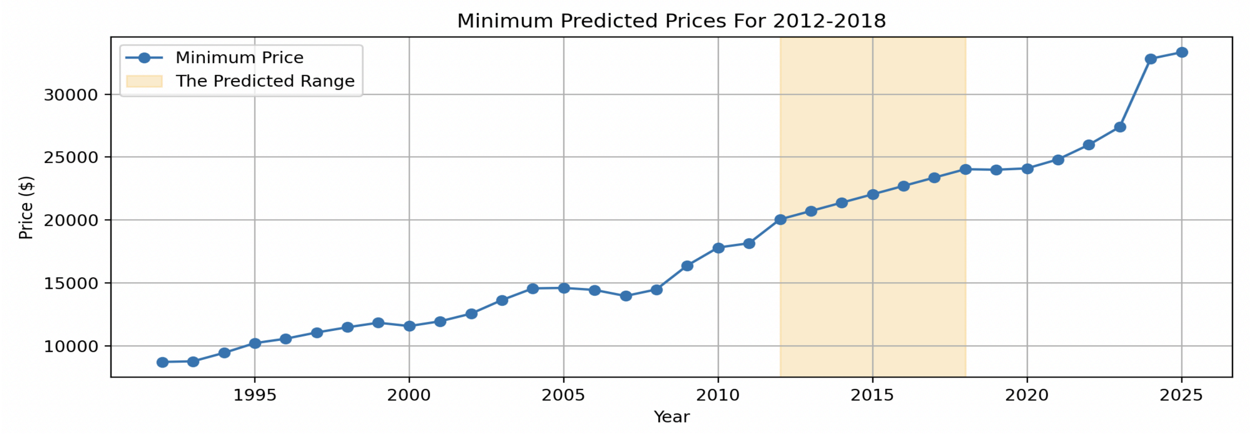
```
import numpy as np
import matplotlib.pyplot as plt

def DynamicLinearRegression(a, b, Initial = 0.1, decrease = 0.01, iterations = 100):
  c = len(a)
  theta = [0.0 for _ in range(len(a[0]))]
  history = []

  for d in range(iterations):
    learning = Initial / (1 + decrease * d)
    predictions = []
    for i in range(c):
      pred = 0.0
      for j in range(len(theta)):
        pred += theta[j] * a[i][j]
      predictions.append(pred)

    errorcounter = [predictions[i] - b[i] for i in range(c)]
    miss = sum(error ** 2 for error in errorcounter) / (2 * c)
    history.append(miss)

    grad = [0.0 for _ in range(len(theta))]
    for j in range(len(theta)):
      for i in range(c):
        grad[j] += errorcounter[i] * a[i][j]
      grad[j] /= c
    for j in range(len(theta)):
      theta[j] -= learning * grad[j]

  return np.array(theta), history

def plot_dynamic_losses(DynTrueLoss, DynTrueMax):
  fig, axes = plt.subplots(1, 2, figsize=(12,5))
  axes[0].plot(DynTrueLoss, label = 'The Model For Min Price - Dynamic Linear Regression')
  axes[0].set_title('Min Price Loss Curve')
  axes[0].set_xlabel('Iterations')
  axes[0].set_ylabel('Our Loss Rate')
  axes[0].grid(True)
  axes[1].plot(DynTrueMax, label = 'The Model for max Price -  Dynamic Linear Regression',
color='red')
  axes[1].set_title('Max Price Loss Curve')
  axes[1].set_xlabel('Iterations')
  axes[1].set_ylabel('Our Loss Rate')
```

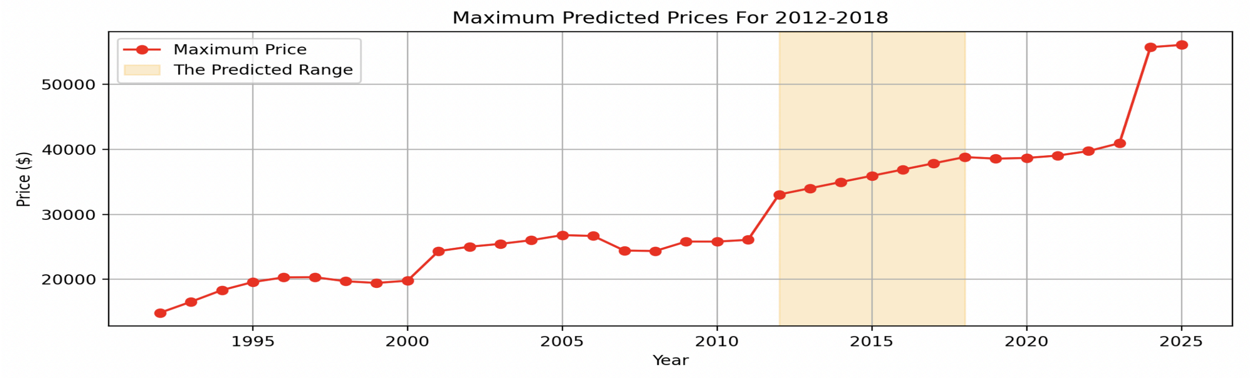
```
axes[1].grid(True)
fig.tight_layout()
plt.show()
```

Main.py

```
DynamicMinimumTheta, DynTrueLoss = DynamicLinearRegression(y, Lowestprice)
DynamicMaximumTheta, DynTrueMax = DynamicLinearRegression(y, Highestprice)
plot_dynamic_losses(DynTrueLoss, DynTrueMax)
predict_missing(FordYears, Lowestprice, Highestprice, DynamicMinimumTheta,
DynamicMaximumTheta, x)
```







Step 5 - Summary Paragraph Address all of the following in your PDF:

What is the predicted price of a 2026 Ford Ranger? Would you consider buying it?

The predicted price of a 2026 Ford Ranger is approximately $34,860 as a minimum and $58,640 as a maximum. From these predicted prices that came from the Dynamic Learning model, I would not consider buying this because the price is significantly higher than the previous prices from older models. The inflation of price would not be worth the car I am getting. I would consider buying a different brand or an older model, since the current price is historically overpriced.

Reflect on each model you trained in steps 2 & 4: Your thoughts on their performance. What could be done to improve them?

After reflecting on the models constructed in this assignment, I believe that the models performed as it was expected to. The initial model produced the correct answers but needed improvement due to the low learning rate which causes small convergences. When we incorporated the dynamic learning model, our learning rate converged much faster and provided a lot more accurate results in our predicted prices.

To improve future models, I would explore something like polynomial regression to better capture the increase in recent prices. Additionally, incorporating external economic factors or product lifecycle information such as production pauses could make the predictions more realistic. Outside environmental factors are a huge consideration when it comes to developing these models and providing that data would incorporate a much more accurate graph.