CS 4372

ASSIGNMENT 1

Names of students in your group:

James Hooper

Hritik Panchasara

Number of free late days used: 0

Note: You are allowed a **total** of 4 free late days for the **entire semester**. You can use at most 2 for each assignment. After that, there will be a penalty of 10% for each late day.

Please list clearly all the sources/references that you have used in this assignment.

For SGD-Regressor model, metrics, & preprocessing data split

https://scikit-learn.org/stable/

For Adam Optimizer understanding

https://towardsdatascience.com/10-gradient-descent-optimisation-algorithms-86989510b5e9

https://hackernoon.com/demystifying-different-variants-of-gradient-descent-optimization-algorithm-19ae9ba2e9bc

https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam

For graphing data

https://seaborn.pydata.org/

https://matplotlib.org/

For data manipulation

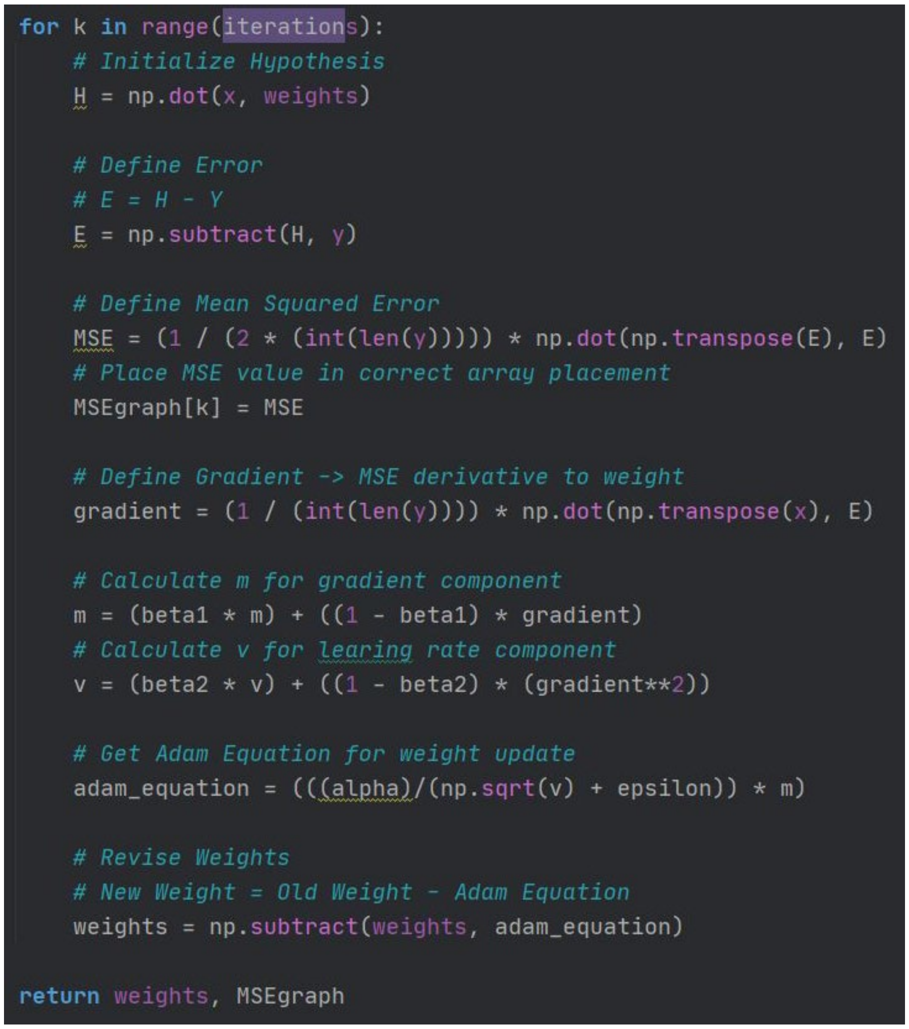
https://pandas.pydata.org/

https://numpy.org

Details of Enhanced Gradient Descent Algorithm: *Adam Optimizer*

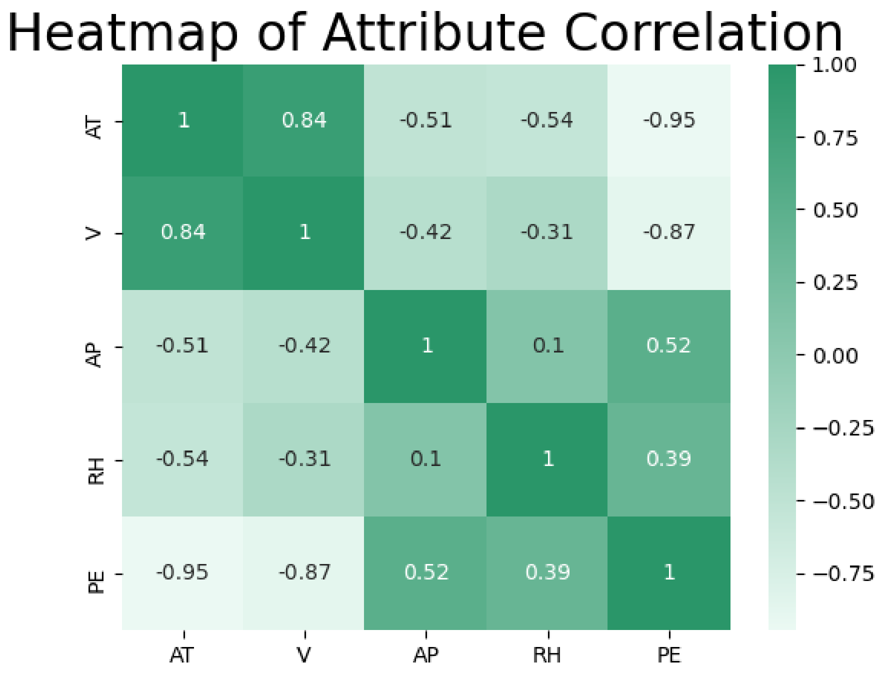
The Adam Optimizer, also known as Adaptive moment estimation, is an enhancement to the vanilla gradient descent model. Adam utilizes two key concepts: an exponential moving average of gradients that is akin to momentum & a type of adaptive learning rate that changes upon each iteration. The parameters/equations used for the Adam optimization are as follows:

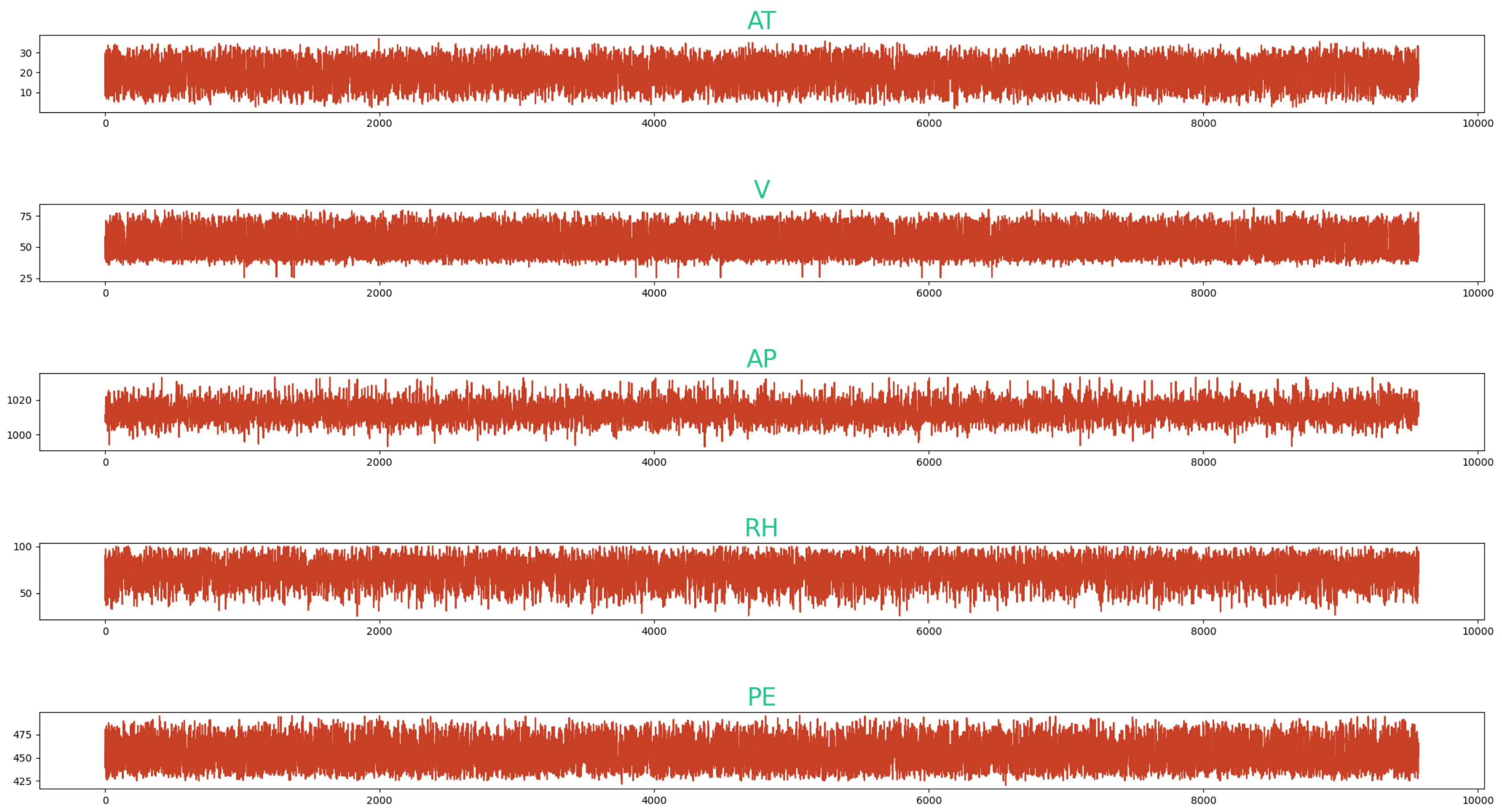
- alpha – the learning rate / step size
- beta1 – exponential decay rate for the first moment estimates
- beta2 – exponential decay rate for the second moment estimates
- epsilon – constant for numerical stability
- m & v – moving averages
    - m is "similar to the history used in Momentum GD"
    - v is "similar to the history used in RMSProp"
- g – gradient
- *The equations for the calculated values upon each iteration can be found in the snippet of code below.*



```
for k in range(iterations):
    # Initialize Hypothesis
    H = np.dot(x, weights)

    # Define Error
    # E = H - Y
    E = np.subtract(H, y)

    # Define Mean Squared Error
    MSE = (1 / (2 * (int(len(y))))) * np.dot(np.transpose(E), E)
    # Place MSE value in correct array placement
    MSEgraph[k] = MSE

    # Define Gradient -> MSE derivative to weight
    gradient = (1 / (int(len(y)))) * np.dot(np.transpose(x), E)

    # Calculate m for gradient component
    m = (beta1 * m) + ((1 - beta1) * gradient)
    # Calculate v for learing rate component
    v = (beta2 * v) + ((1 - beta2) * (gradient**2))

    # Get Adam Equation for weight update
    adam_equation = (((alpha)/(np.sqrt(v) + epsilon)) * m)

    # Revise Weights
    # New Weight = Old Weight - Adam Equation
    weights = np.subtract(weights, adam_equation)

return weights, MSEgraph
```

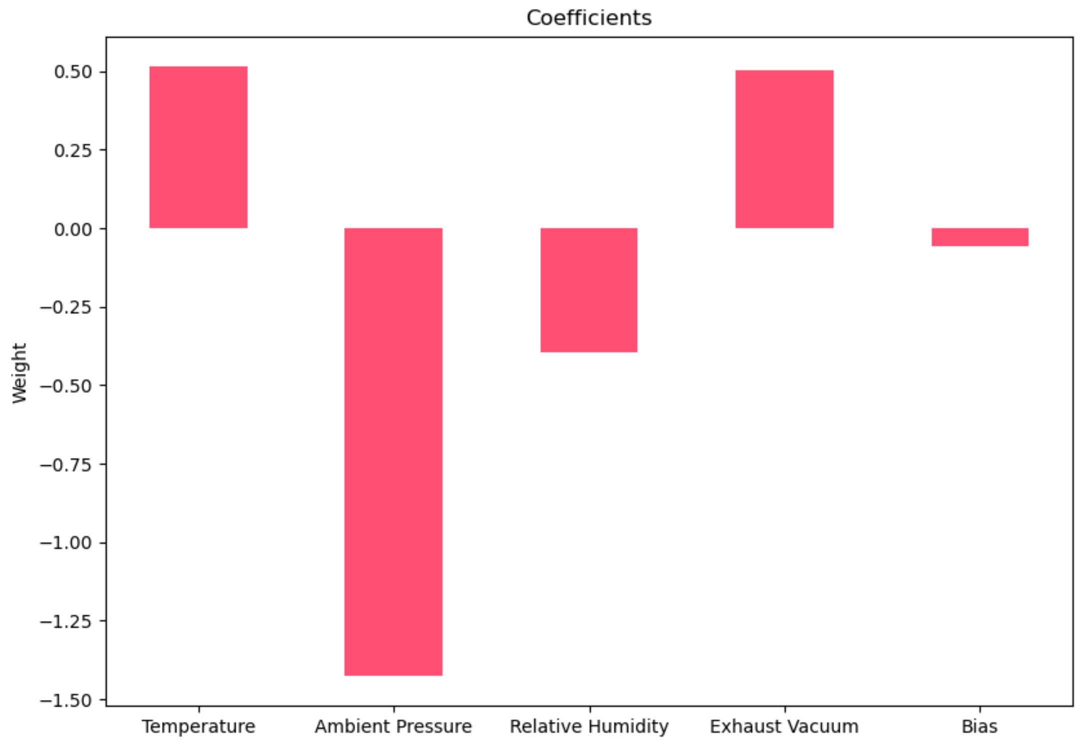
Here are the pre-processing graph results.

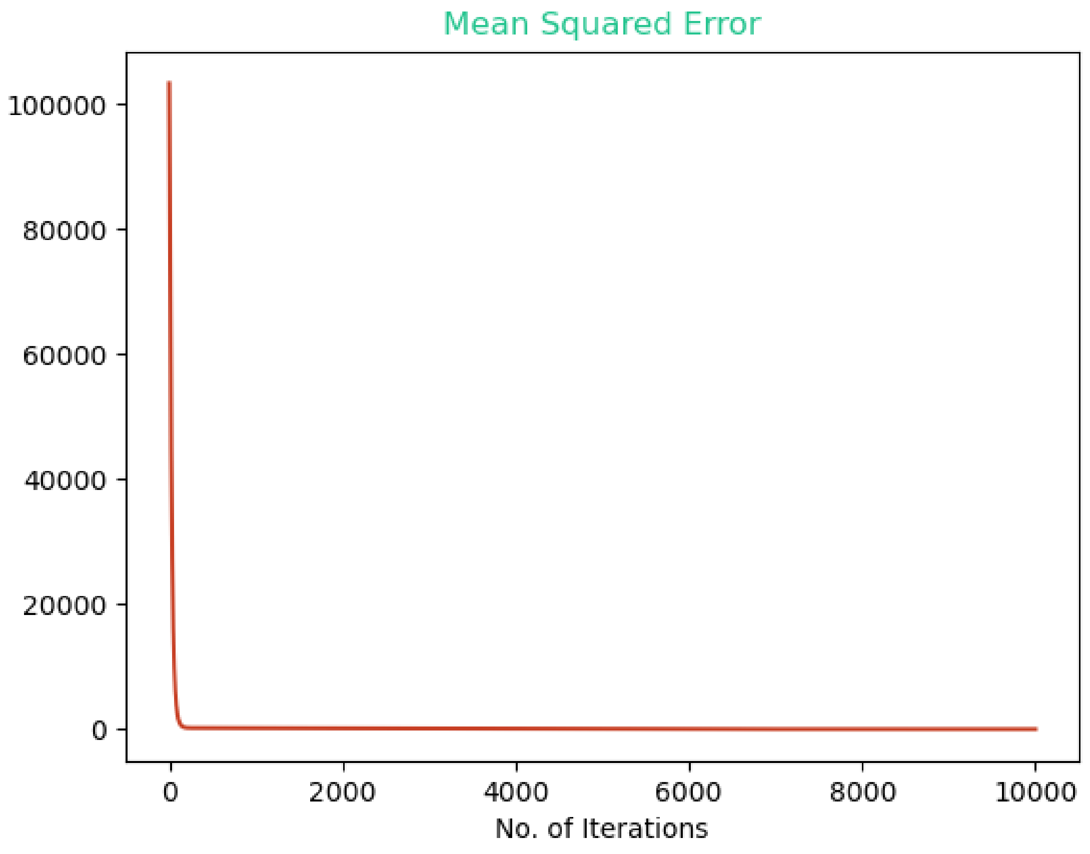


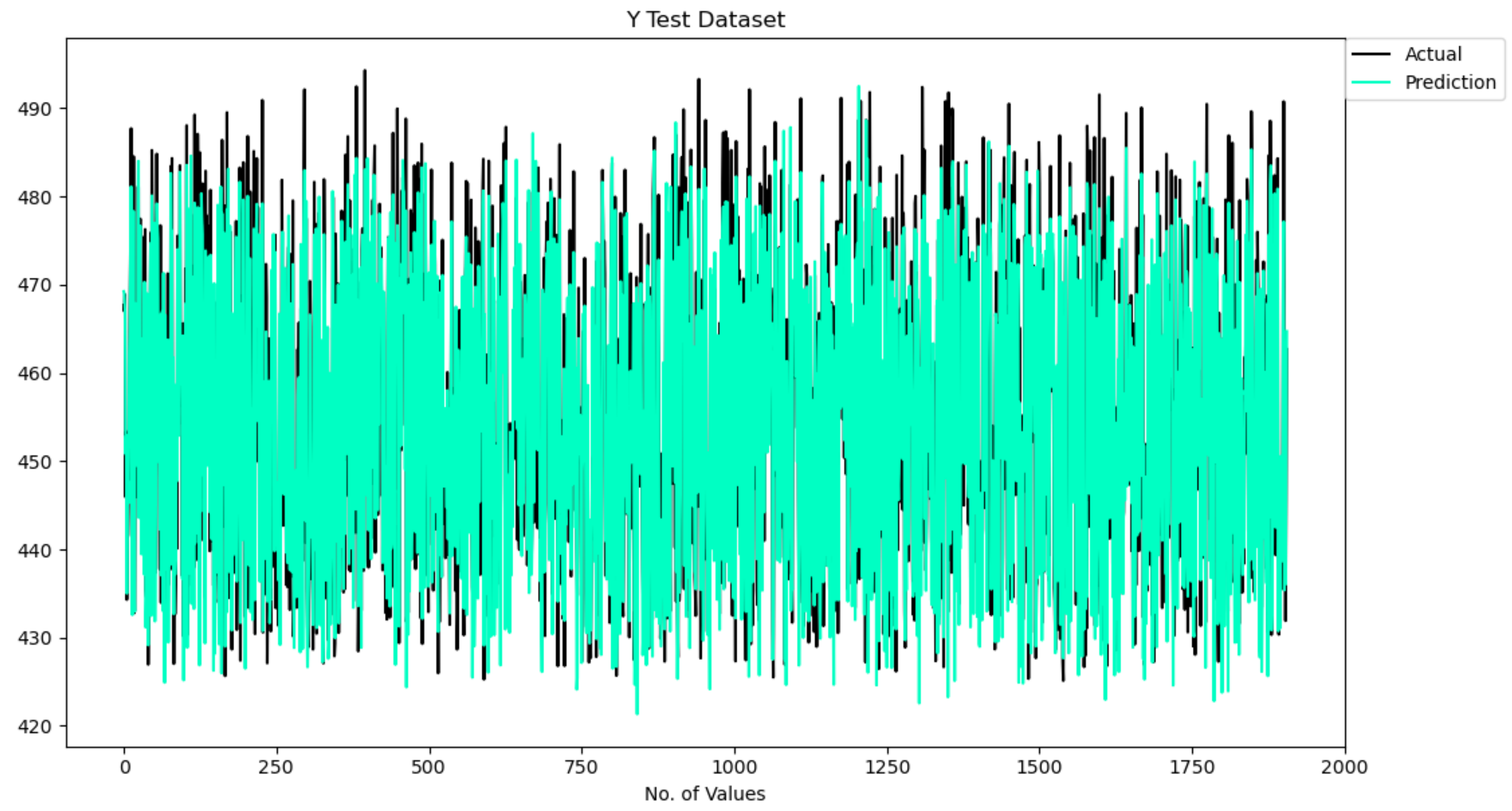


Plots of the attributes. The last plot is of the output value.

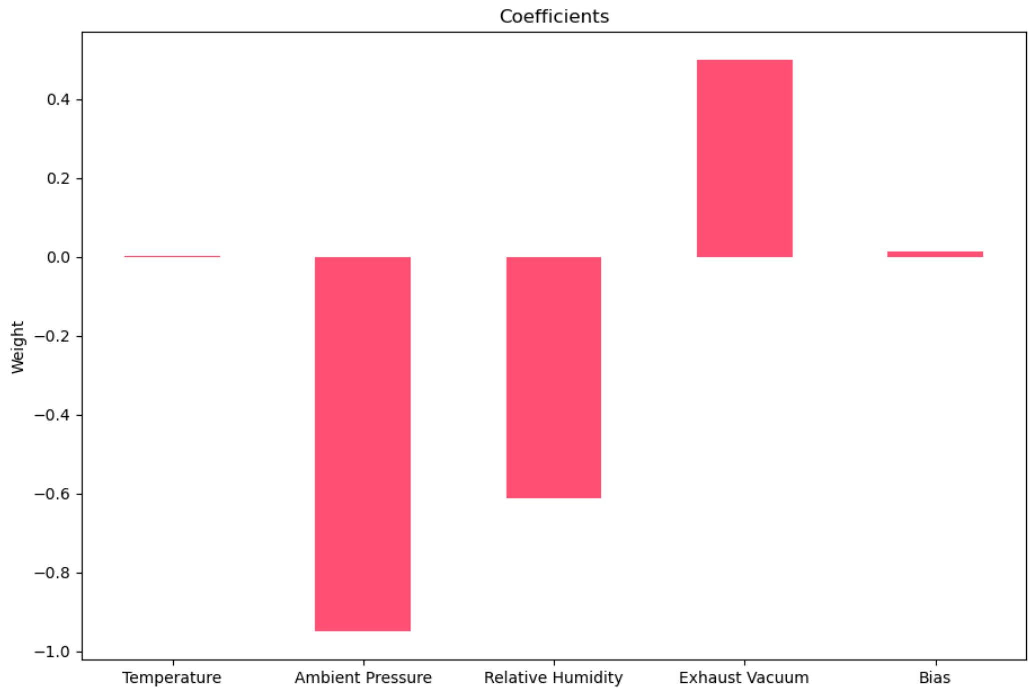
Part1 Plots for the trial with the best parameters found. State = 5.

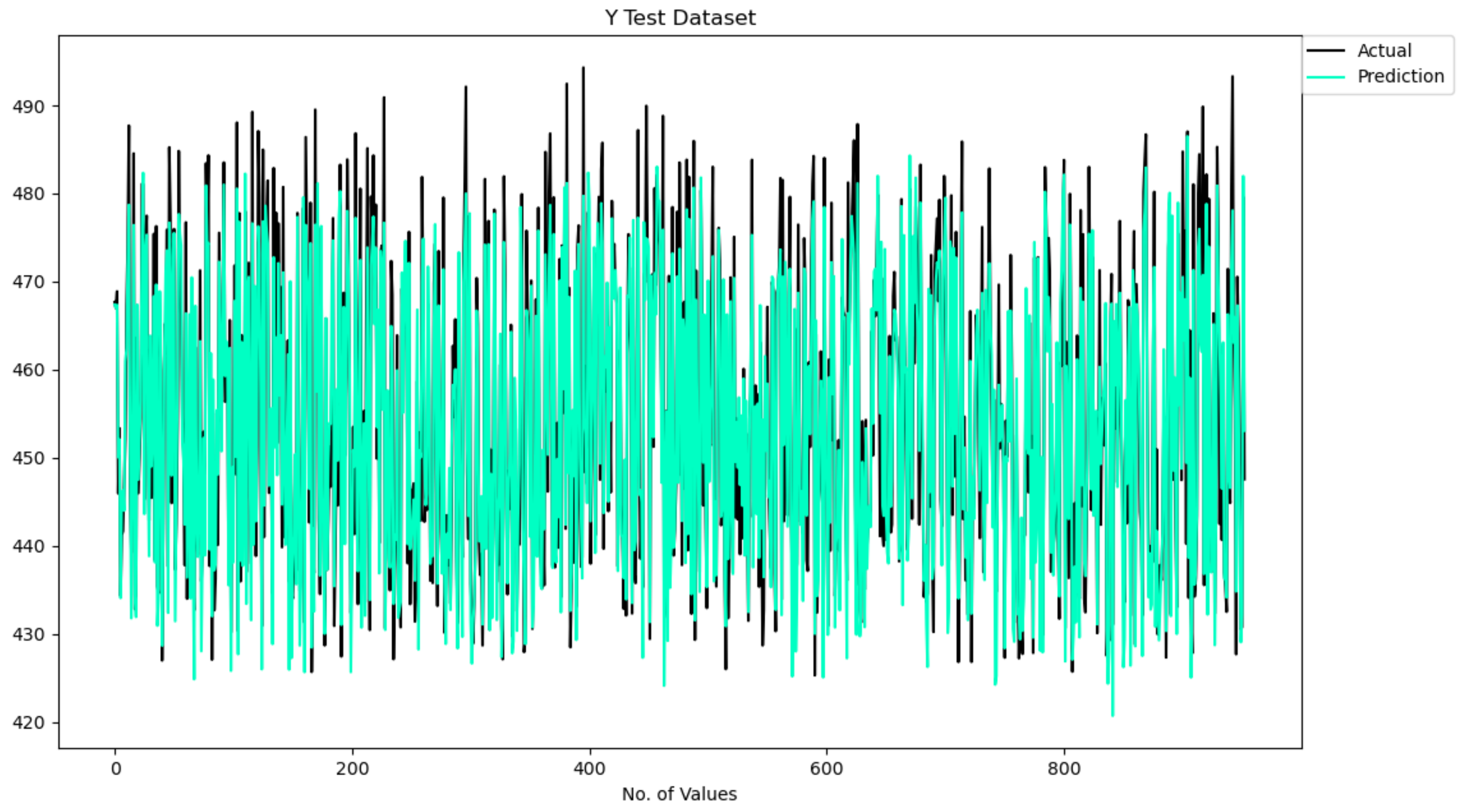





Y Test Dataset
Actual
Prediction
490
480
470
460
450
440
430
420
0
250
500
750
1000
1250
1500
1750
2000
No. of Values

Part2 Plots for the trial with the best parameters found. State = 5.





Part 1 of Logs

Tuning the Enhanced Gradient Descent Model for Best Parameters

```
Part 1 of Enhanced Gradient Descent
Parameters Used:
State:  5
Alpha: .001        |Beta1: .9          |Beta2: .999        |Epsilon: 10^-8    |m = 0    |v = 0
Iterations:  5000
Train Split:  80.0 %            |Test Split:  20.0 %
Coefficients:
 [0.4374861331222539, -0.6123631130942228, -0.26003467414341763, 0.44559254221717787, 0.3878470184438752]
Train Accuracy:
Mean Squared Error:  81.5735800926913
R^2 Value:  0.48285457574898716
Test Accuracy:
Mean Squared Error:  80.98996701556176
R^2 Value:  0.49817895252641486
```

```
Part 1 of Enhanced Gradient Descent
Parameters Used:
State:  5
Alpha: .001        |Beta1: .9          |Beta2: .999        |Epsilon: 10^-8    |m = 0    |v = 0
Iterations:  8000
Train Split:  80.0 %            |Test Split:  20.0 %
Coefficients:
 [0.48719170002575046, -1.118172756881196, -0.5029630792192917, 0.4937123219603581, 0.03896292313804325]
Train Accuracy:
Mean Squared Error:  29.51493201645292
R^2 Value:  0.8844960353362512
Test Accuracy:
Mean Squared Error:  29.646878731320733
R^2 Value:  0.8864923238960015
```

```
Part 1 of Enhanced Gradient Descent
Best one found.
Parameters Used:
State:  5
Alpha: .001        |Beta1: .9          |Beta2: .999        |Epsilon: 10^-8    |m = 0    |v = 0
Iterations:  10000
Train Split:  80.0 %            |Test Split:  20.0 %
Coefficients:
 [0.5144726250635316, -1.423992626029842, -0.39589981822794335, 0.5009286790132073, -0.058344123852125625]
Train Accuracy:
Mean Squared Error:  26.22033816483604
R^2 Value:  0.9019748858552826
Test Accuracy:
Mean Squared Error:  26.42235572271788
R^2 Value:  0.9030012797938725
```

```
Part 1 of Enhanced Gradient Descent
Drops attribute: 'AT' (this can be verified with the heatmap showing it's correlation with 'V')
Parameters Used:
State:  5
Alpha: .001        |Beta1: .9          |Beta2: .999        |Epsilon: 10^-8    |m = 0    |v = 0
```

```
Iterations:  10000
Train Split:  80.0 %            |Test Split:  20.0 %
Coefficients:
 [0.4648212353511957, -1.0136507011995473, 0.4906395683339943, 0.16063057719700888]
Train Accuracy:
Mean Squared Error:  57.38900595212531
R^2 Value:  0.7500325370547691
Test Accuracy:
Mean Squared Error:  55.812871327546695
R^2 Value:  0.7628366986014008
```

```
Part 1 of Enhanced Gradient Descent
Parameters Used:
State:  5
Alpha: .001          |Beta1: .9           |Beta2: .999        |Epsilon: 10^-8    |m = 0    |v = 0
Iterations:  10000
Train Split:  90.0 %            |Test Split:  10.0 %
Coefficients:
 [0.5140621380109054, -1.422759204876915, -0.39610464580669436, 0.5009524502521513, -0.05821683974215756]
Train Accuracy:
Mean Squared Error:  26.20013185328805
R^2 Value:  0.9024861152060023
Test Accuracy:
Mean Squared Error:  26.86479385463498
R^2 Value:  0.8985095209493589
```

Note:

- A key thing to mention here is that the Adam Optimizer is recommended to have the same variable values for Alpha, Beta1, Beta2, Epsilon, initial m, and initial v. These standard values are used through most implementations of using the Adam optimization. After trying to tune the parameters for the Enhanced Gradient Descent Model here were the best parameters found: iterations = 10000, train/test split = 90/10, and keeping all attributes.

Part 2 of Logs

Comparing Gradient Descent & Sklearn Stochastic Gradient Descent Model

```
Part 1 of Enhanced Gradient Descent
Parameters Used:
State:  1
Alpha: .001          |Beta1: .9            |Beta2: .999         |Epsilon: 10^-8    |m = 0    |v = 0
Iterations:  10000
Train Split:  80.0 %            |Test Split:  20.0 %
Coefficients:
 [0.5135360955599416, -1.4129702682082277, -0.4030913398665183, 0.5012177628402052, -0.0601577196111157]
Train Accuracy:
Mean Squared Error:  25.907627020055898
R^2 Value:  0.9030119875133472
Test Accuracy:
Mean Squared Error:  28.014235643608277
R^2 Value:  0.8973077744212778
```

```
Part 2 of Enhanced Gradient Descent
Parameters Used:
State:  1
Iterations:  10000
Learning Rate:  1e-06
Coefficients:
 [ 6.33944060e-04 -1.04579006e+00 -5.69972842e-01  5.00032224e-01 -1.42396194e-03]
Train Accuracy:
Mean Squared Error:  29.96784033730602
R^2 Value:  0.8827791382973325
Test Accuracy:
Mean Squared Error:  35.768615316443594
R^2 Test:  0.8706809198549961
```

```
Part 1 of Enhanced Gradient Descent
Parameters Used:
State:  2
Alpha: .001          |Beta1: .9            |Beta2: .999         |Epsilon: 10^-8    |m = 0    |v = 0
Iterations:  10000
Train Split:  80.0 %            |Test Split:  20.0 %
Coefficients:
 [0.5146539289431077, -1.4216915288677878, -0.3944771045499932, 0.5007776345920979, -0.05787278024584849]
Train Accuracy:
Mean Squared Error:  26.35812790742656
R^2 Value:  0.9008578375693331
Test Accuracy:
Mean Squared Error:  25.894430060307297
R^2 Value:  0.9052769558060673
```

```
Part 2 of Enhanced Gradient Descent
Parameters Used:
State:  2
Iterations:  10000
Learning Rate:  1e-06
Coefficients:
 [ 8.26639238e-04 -1.17934925e+00 -5.03116141e-01  5.00811753e-01  -2.21790217e-02]
```

Train Accuracy:
Mean Squared Error: 29.329251229280484
R^2 Value: 0.8865946508768744
Test Accuracy:
Mean Squared Error: 28.93303057712341
R^2 Test: 0.8899080573047872

Part 1 of Enhanced Gradient Descent
Parameters Used:
State: 3
Alpha: .001 |Beta1: .9 |Beta2: .999 |Epsilon: 10^-8 |m = 0 |v = 0
Iterations: 10000
Train Split: 80.0 % |Test Split: 20.0 %
Coefficients:
[0.514850161615456, -1.4175520120905034, -0.4032907854138991, 0.5010969321402667, -0.05667334052436697]
Train Accuracy:
Mean Squared Error: 26.049164151569435
R^2 Value: 0.90280841347468
Test Accuracy:
Mean Squared Error: 27.403502699222685
R^2 Value: 0.9015783901380442

Part 2 of Enhanced Gradient Descent
Parameters Used:
State: 3
Iterations: 10000
Learning Rate: 1e-06
Coefficients:
[5.79192509e-04 -9.85495454e-01 -5.98517524e-01 4.98454730e-01 7.34514243e-03]
Train Accuracy:
Mean Squared Error: 31.1965394854964
R^2 Value: 0.8778546902838902
Test Accuracy:
Mean Squared Error: 32.31482406991121
R^2 Test: 0.8783472782239139

Part 1 of Enhanced Gradient Descent
Parameters Used:
State: 4
Alpha: .001 |Beta1: .9 |Beta2: .999 |Epsilon: 10^-8 |m = 0 |v = 0
Iterations: 10000
Train Split: 80.0 % |Test Split: 20.0 %
Coefficients:
[0.5143122919765742, -1.424814866085679, -0.3971354213196689, 0.5011520085825308, -0.060557465214266824]
Train Accuracy:
Mean Squared Error: 26.236927421839752
R^2 Value: 0.9020796768035431
Test Accuracy:
Mean Squared Error: 26.35186963702148
R^2 Value: 0.9030362793154947

Part 2 of Enhanced Gradient Descent
Parameters Used:
State: 4
Iterations: 10000
Learning Rate: 1e-06

```
Coefficients:
 [ 6.18053193e-04 -1.02958395e+00 -5.75053426e-01  4.99599192e-01 -1.20913344e-03]
Train Accuracy:
Mean Squared Error:  30.428115367405606
R^2 Value:  0.8811448017908418
Test Accuracy:
Mean Squared Error:  29.810869047270845
R^2 Test:  0.8834203548265686
```

```
Part 1 of Enhanced Gradient Descent
Parameters Used:
State:  5
Alpha: .001        |Beta1: .9          |Beta2: .999       |Epsilon: 10^-8    |m = 0    |v = 0
Iterations:  10000
Train Split:  80.0 %           |Test Split:  20.0 %
Coefficients:
 [0.5144726250635316, -1.423992626029842, -0.39589981822794335, 0.5009286790132073, -0.058344123852125625]
Train Accuracy:
Mean Squared Error:  26.22033816483604
R^2 Value:  0.9019748858552826
Test Accuracy:
Mean Squared Error:  26.42235572271788
R^2 Value:  0.9030012797938725
```

```
Part 2 of Enhanced Gradient Descent
Parameters Used:
State:  5
Iterations:  10000
Learning Rate:  1e-06
Coefficients:
 [ 5.20305073e-04 -9.22055912e-01 -6.25737931e-01  4.97391759e-01 1.76156170e-02]
Train Accuracy:
Mean Squared Error:  33.3083715750111
R^2 Value:  0.8684416548402241
Test Accuracy:
Mean Squared Error:  33.33351926639194
R^2 Test:  0.8712252043979538
```

Notes:

- The chosen parameters for the SGD-Regressor kept the iterations the same as the ones used for the Enhanced Gradient Descent, set the learning rate to 1e-06 since it was the only Learning Rate that actually was optimal, set the loss function to “squared_loss”, and did not set any penalty.
- The states here are kept the same to properly compare the models.
- The obvious conclusion to be made here is that from the chosen parameters for both models, the Enhanced Gradient Descent model that implemented the Adam optimizer performed better than the sklearn SGD-Regressor model. Lower Mean Squared Error value and a higher R^2 value.

Answers to Questions

Are you satisfied that you have found the best solution? Explain.

- Overall, we are satisfied with the results we obtained from our enhanced version of vanilla Gradient Descent. The reason for this is that the accuracy achieved is something that can be interpreted as a success. Of course, there are other optimizations that can be used for the model to achieve even better results such as Normalizing or Standardizing the data before sending it through the model. There could also be other pre-processing analysis that can help modify the data to be more optimal as well. Since the accuracy seems sufficient and it is not only comparable to the SGD-Regressor model but achieves better results, we can be satisfied that we found a good solution.

Are you satisfied that the package has found the best solution? Explain.

- We feel that there is a lot more that could be done with the SGD-Regressor package that we didn't implement. There were a lot more parameters and other resources that could have helped fine tune the model. On top of Normalizing or Standardizing the data, we could have spent more time understanding all the inputs into the model that were offered. Furthermore, we are trying to compare the Adam optimizer to the SGD-Regressor model so it would be difficult to alter parameters without just having two almost incomparable models. Due to this concern we kept the SGD-Regressor model as simple as possible in order to have an easier understanding of what was directly being compared.