PREDICTING ENERGY BEHAVIOR OF PROSUMERS

Presented by: Minda, Evan, Andy

OVERVIEW OF THE PROJECT

- Objective

  Develop models to predict the energy usage patterns of prosumers.

- Significance

  Accurate predictions can lead to reduced operational costs and more efficient energy use.

  As more solar energy is used, this issue will be increasingly prevalent.

- Description
  - Challenge/competition on Kaggle hosted by Enefit
  - Prosumers' energy use is currently not predictable to a satisfactory accuracy
- Goal
  - To best create a model that can accurately predict the energy behavior of prosumers
  - Scored and evaluated using MSE (mean squared error)

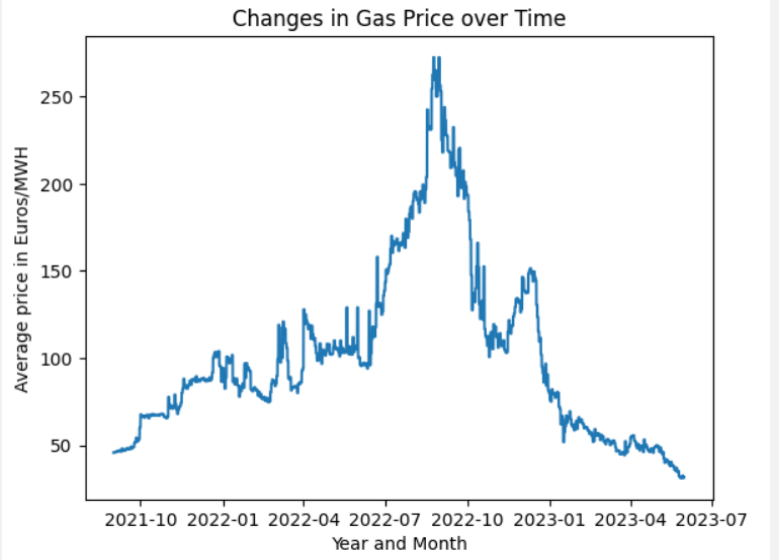
DATASET & DATA CLEANING

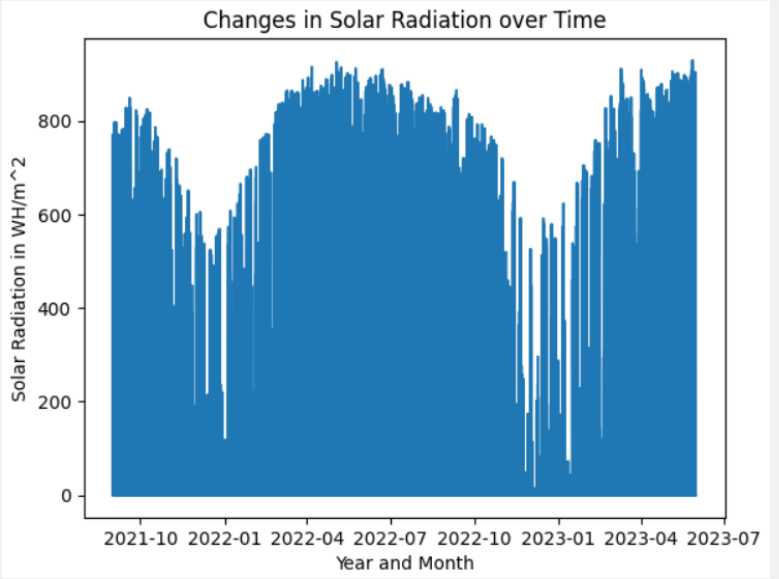
Data are given in 5 different files:

- Train – including 8 features
- Electricity prices – including 4 features
- Gas prices – including 4 features
- Forecast weather – including 18 features
- Weather station to county mapping

Important features:

- Product type
- Energy prices
- Direct solar radiation
- Wind speeds
- Precipitation





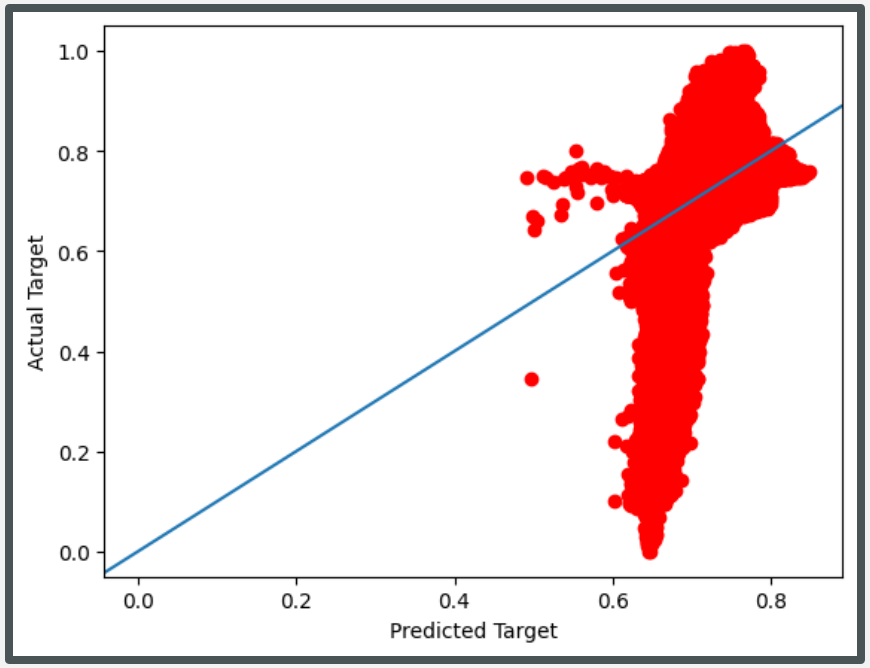
DATA & DATA CLEANING

- Merge all data to a single DataFrame
- Data ranged from 2021-09-01 to 2023-05-30
- Removed missing values & normalized data
- After merging and cleaning, 495,638 total entries
- Number of features: 23
- Target: net presumption or consumption

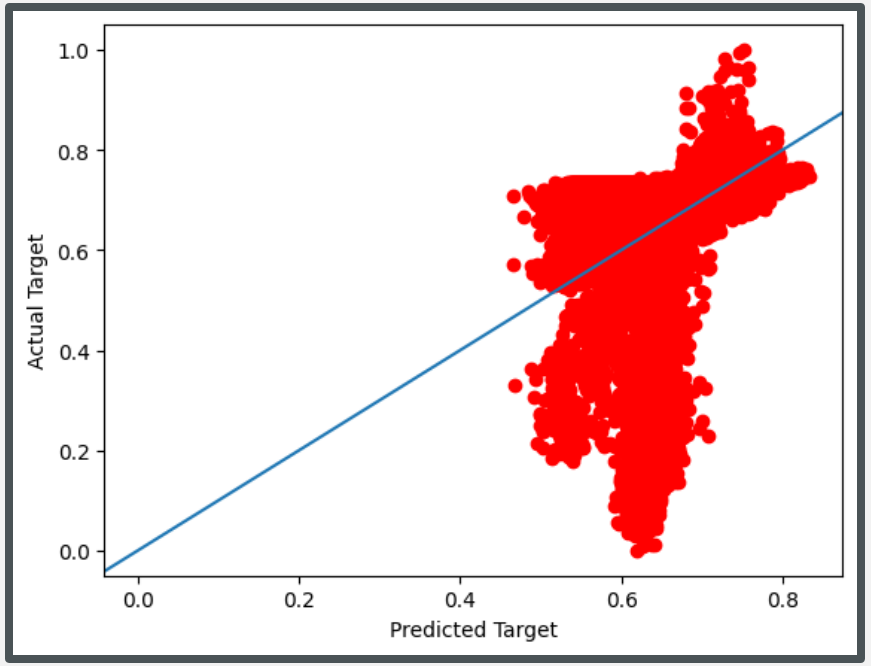
LINEAR MODEL

- Testing MSE: 0.0054
- Training MSE: 0.0042

- As expected – not great



Training

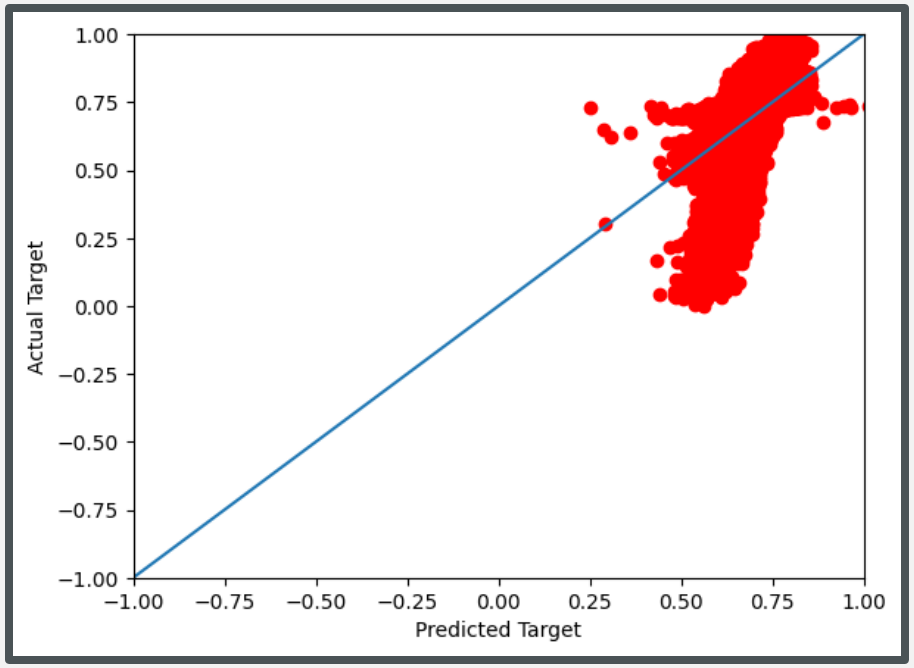


Testing

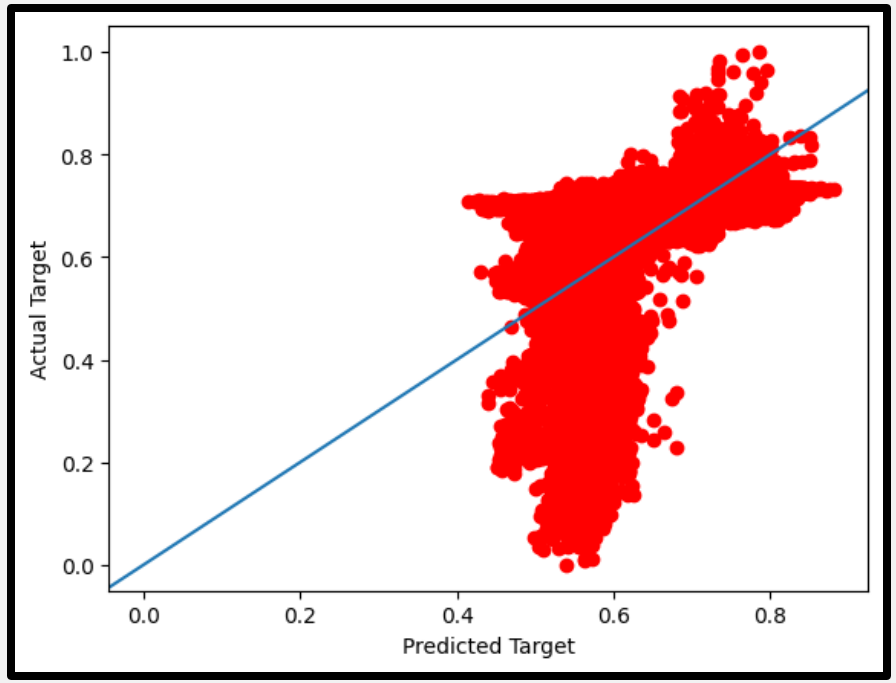
QUADRATIC MODEL

- Testing MSE: 0.00419
- Training MSE: 0.0039

- Much lower MSE vs linear



Training

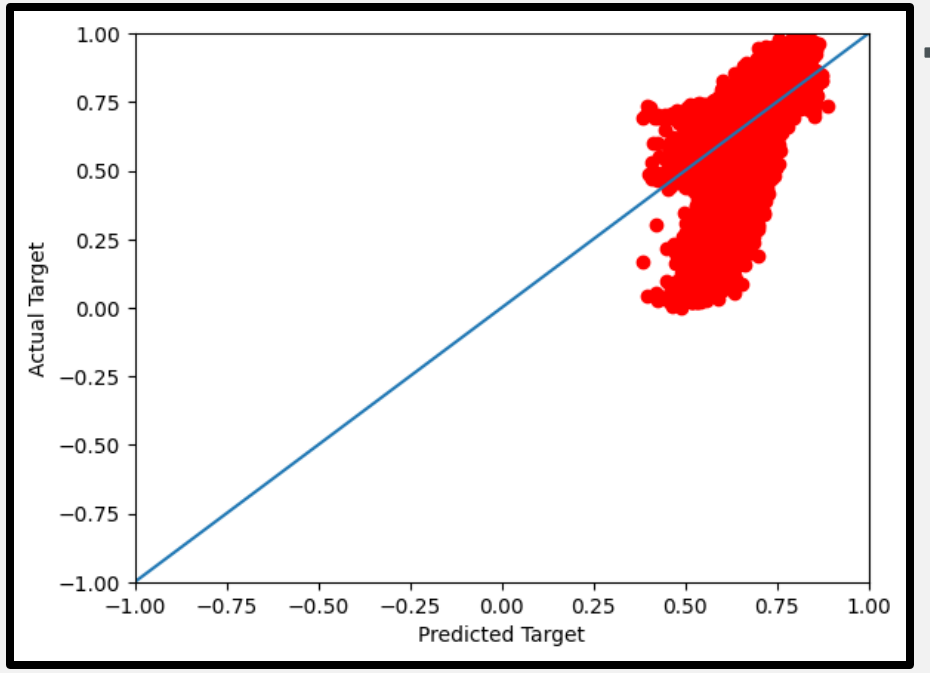


Testing

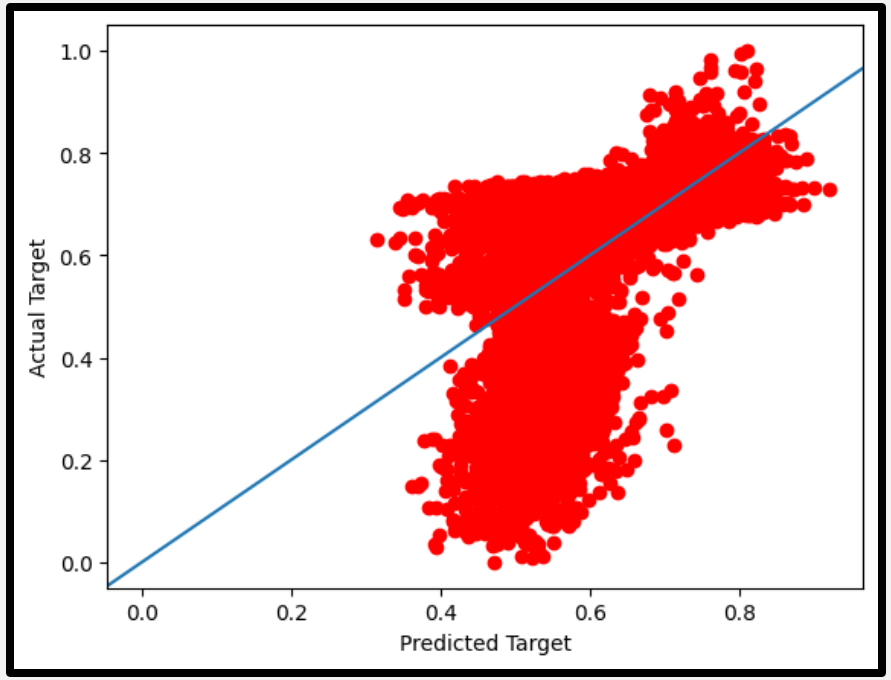
CUBIC

- Testing MSE: 0.00446
- Training MSE: 0.00367
  - Overfitting?

- Too many feature variables
  - Limited by system RAM
  - Using 50% of data to train



Training

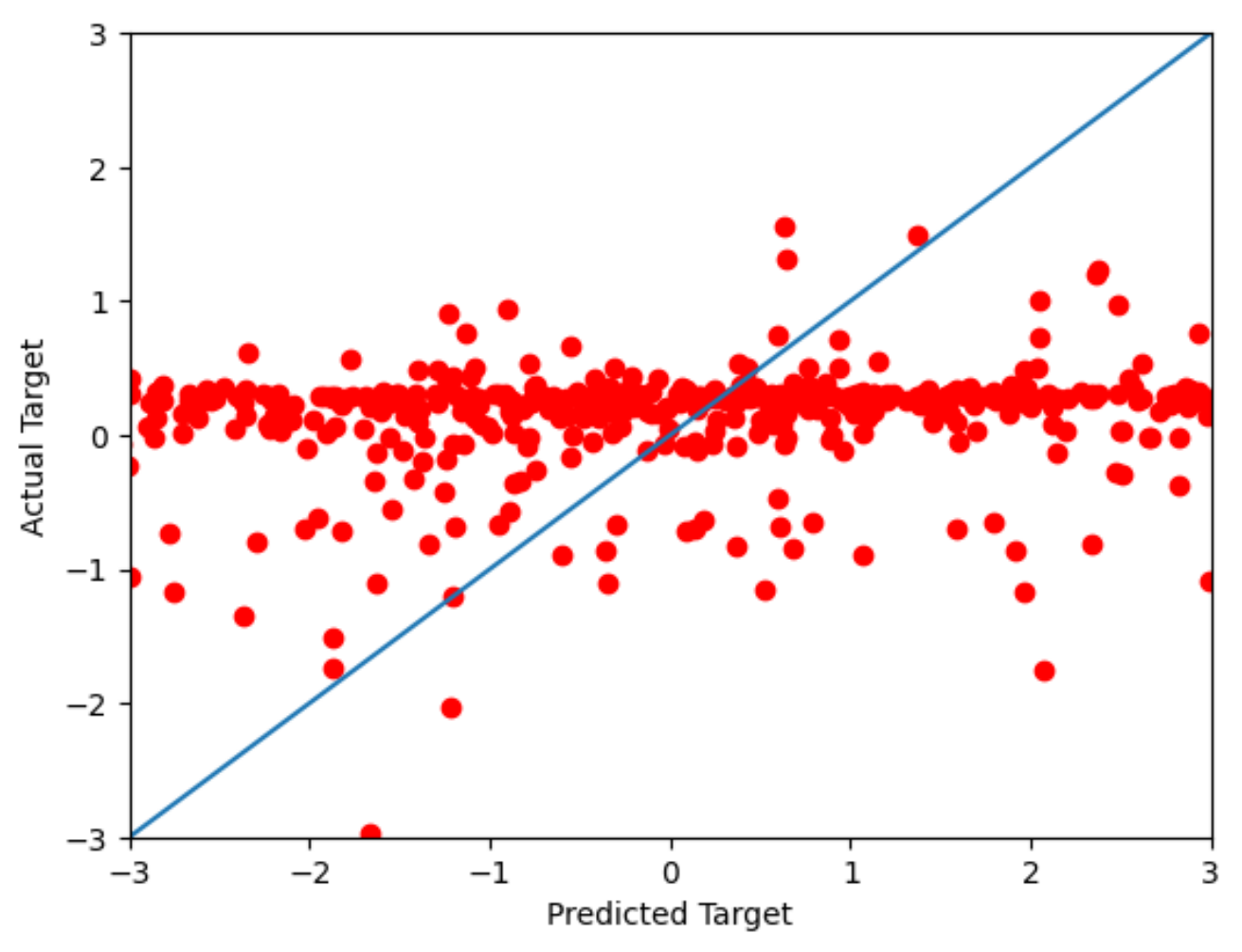


Testing

QUARTIC

- Testing MSE: 0.008755
- Training MSE: 0.00793

- Starting to get much worse



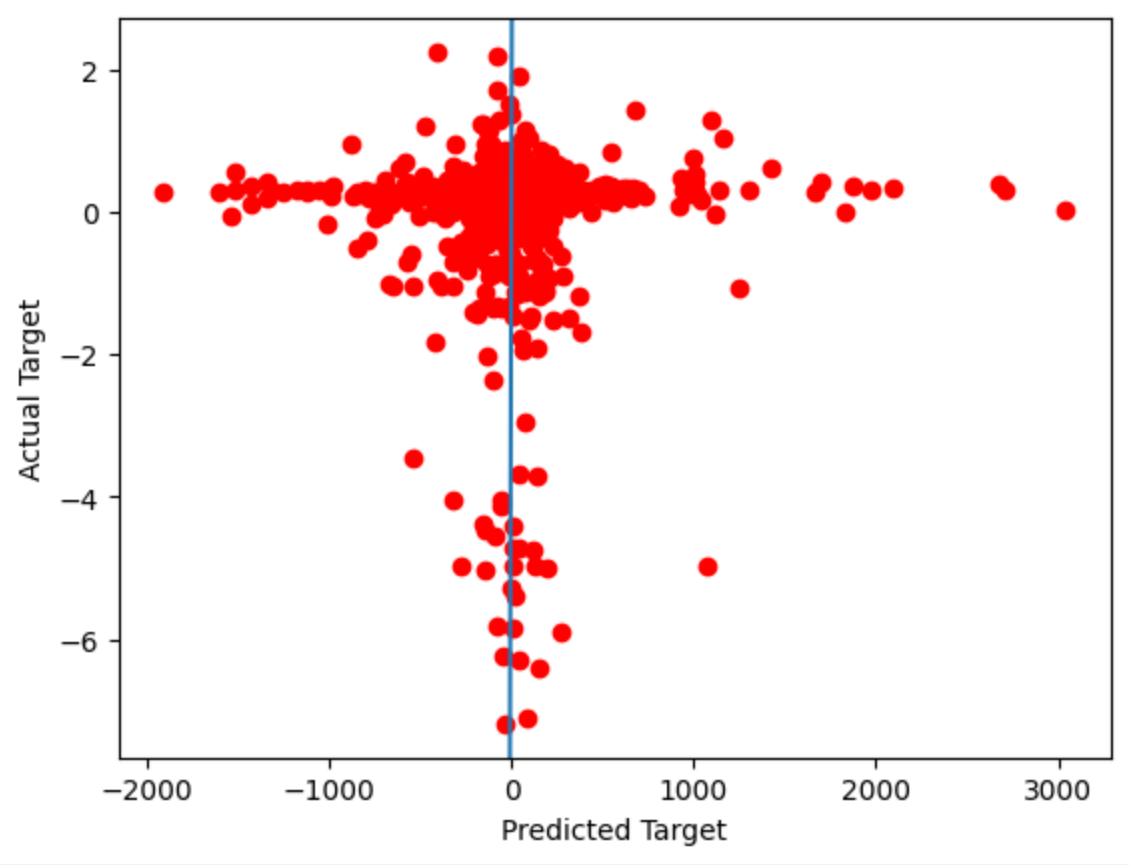
QUINTIC

```
print(len(poly.get_feature_names_out())) # get feature names
#print(polyFeatureNames)
poly5 = solve_normal_eq(xPoly, yTrain) # solve normal equation
```

26334

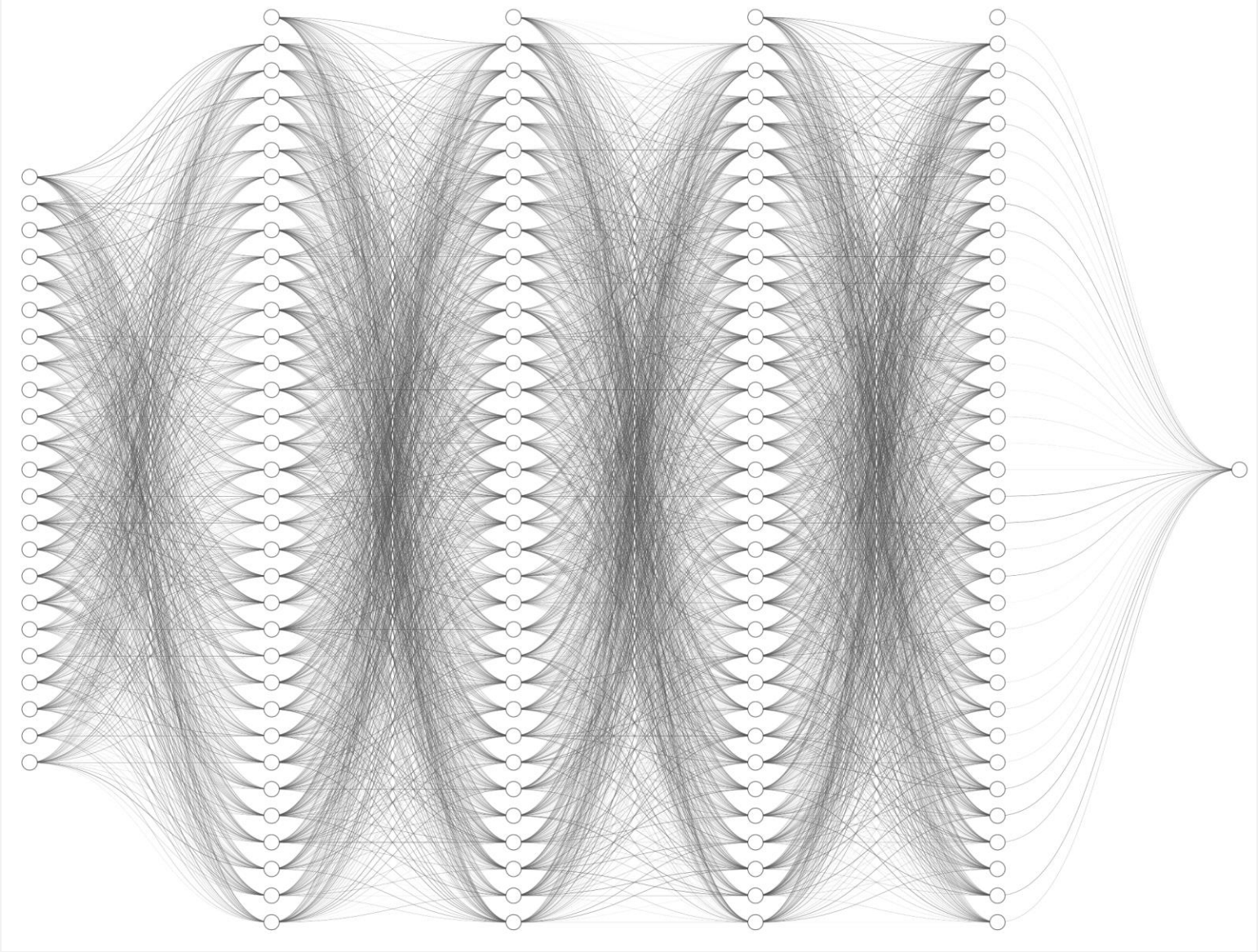
Your session crashed for an unknown reason. View runtime logs

25,000+ feature variables



NEURAL NETWORK

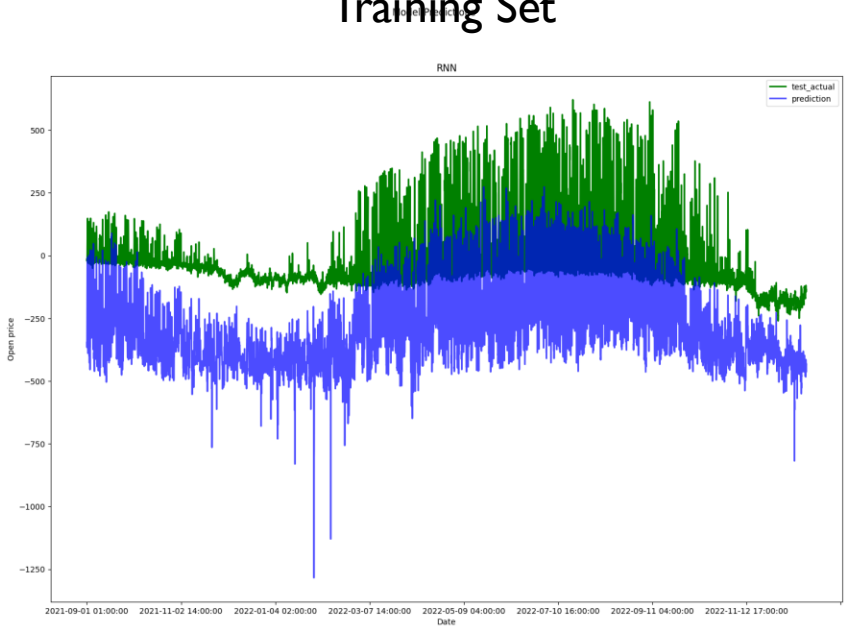
- Recurrent Neural Network
- Input layer – 23 neurons
- Hidden layers – 50 neurons
- Output layer – 1 neuron
- Batch Size – 10000

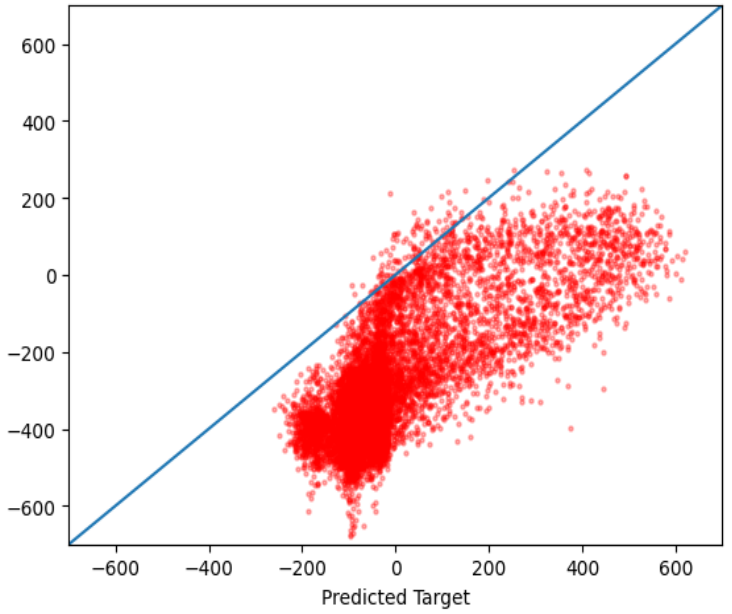


4 HIDDEN LAYERS 20 EPOCH TANH

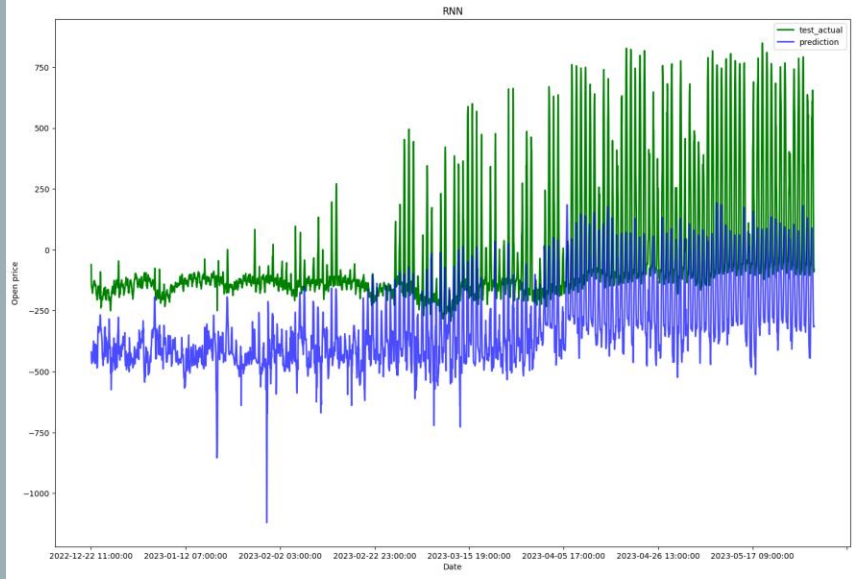
Training MSE – normalized: 0.0017
Testing MSE – normalized: 0.0029

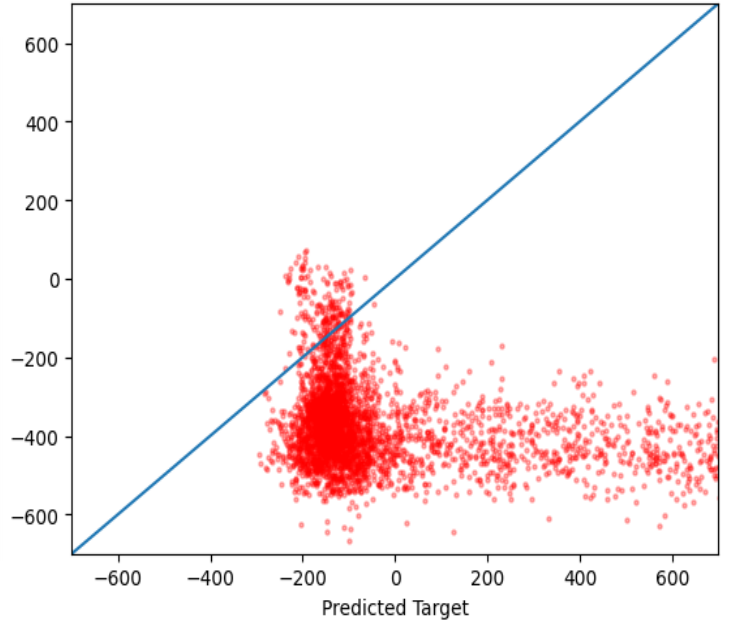
Training Set





Testing Set

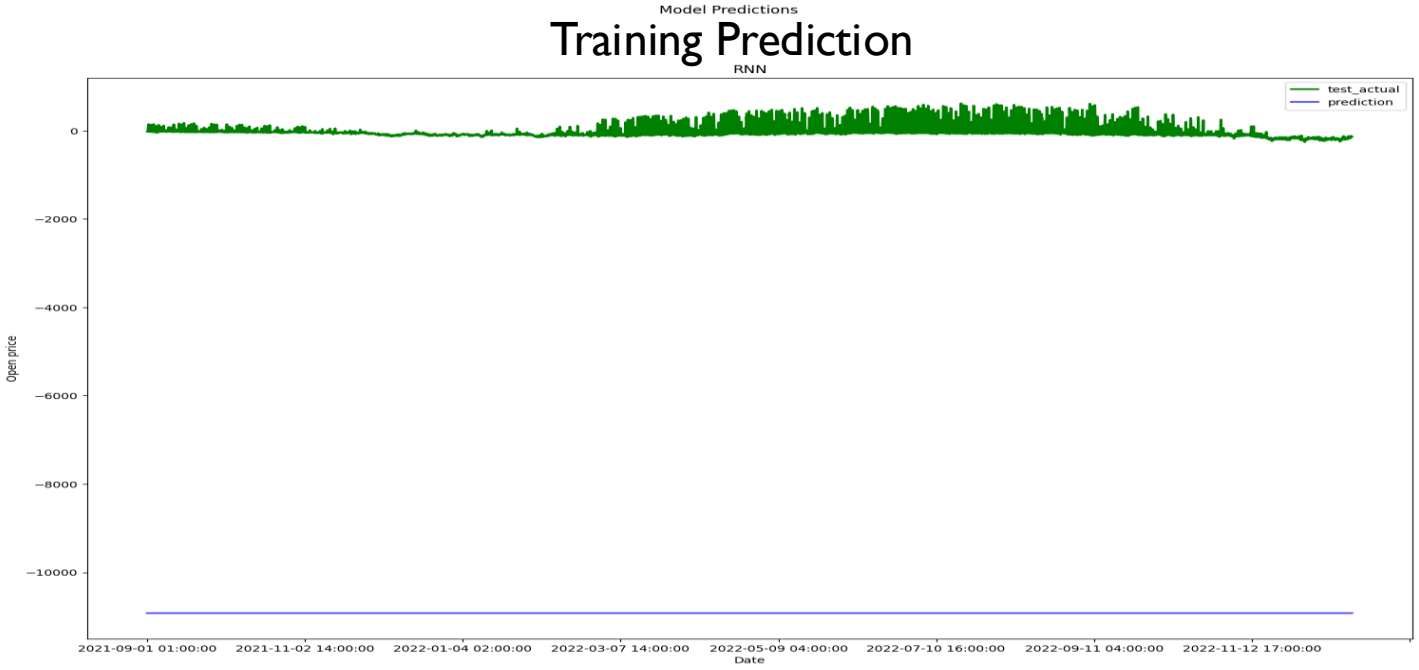




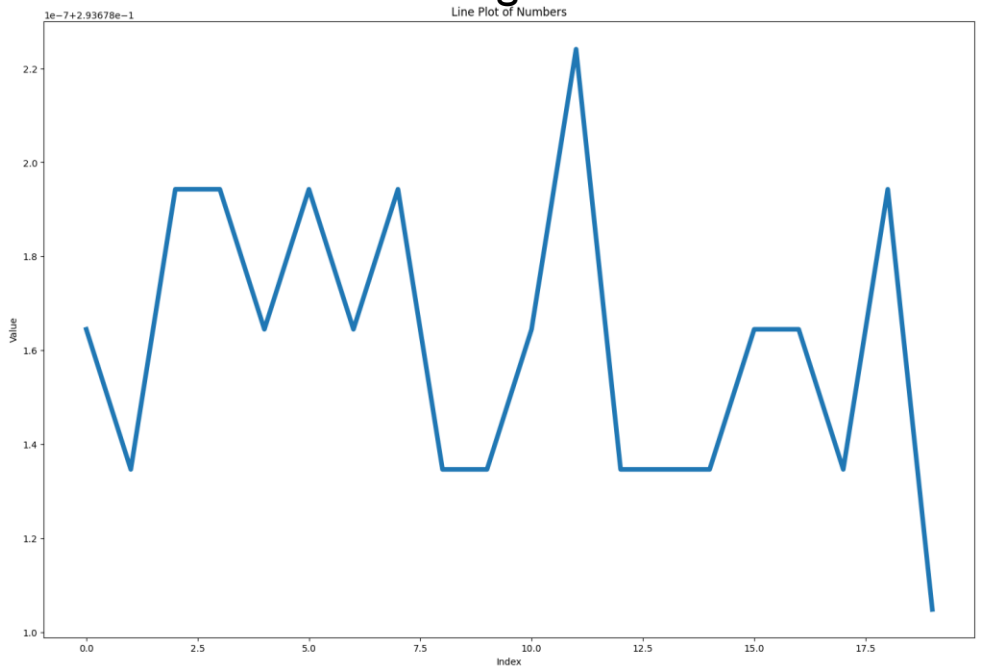
4 HIDDEN LAYERS
20 EPOCH
RELU

Training MSE – normalized: 0.2937
Testing MSE – normalized: 0.2926

Training Prediction



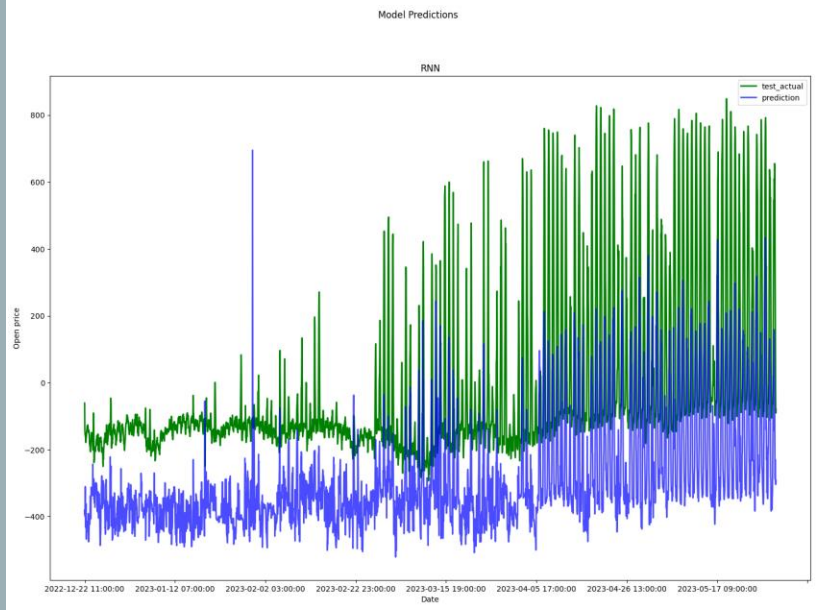
Training Loss

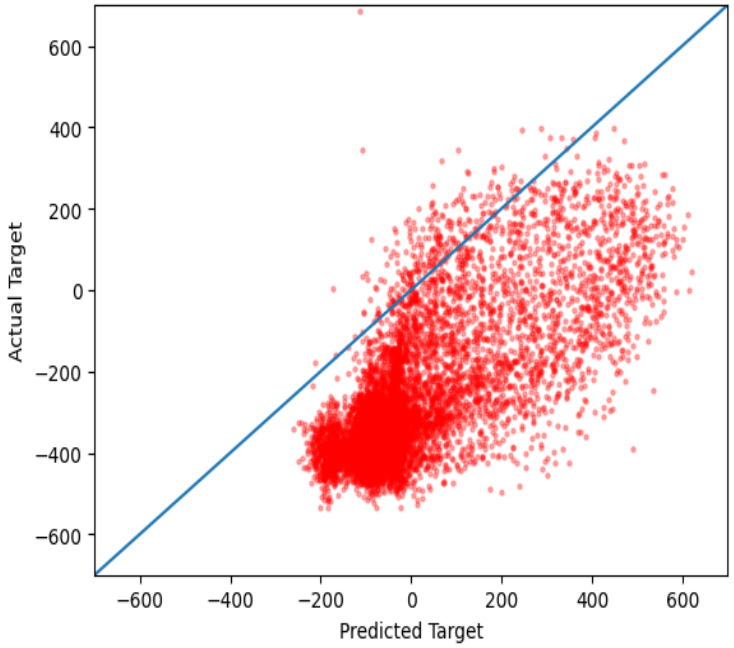


4 HIDDEN LAYERS 50 EPOCH TANH

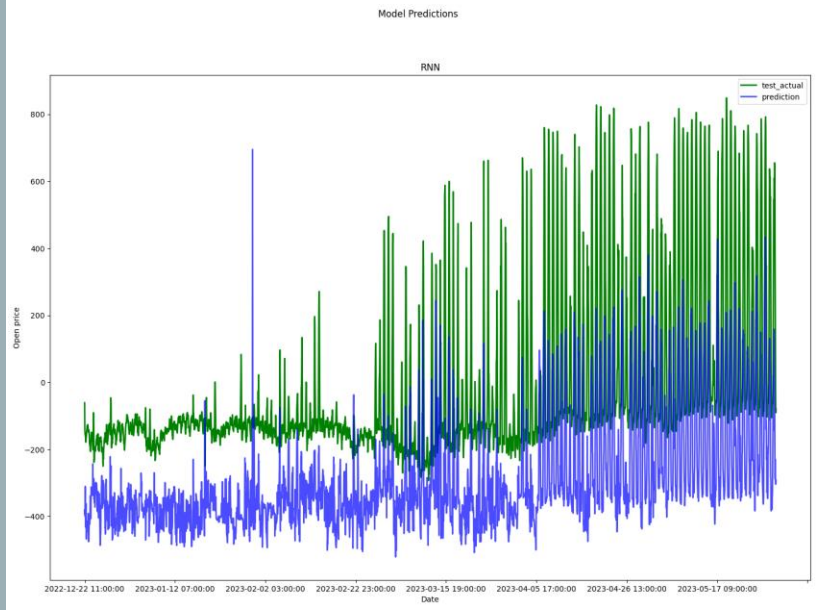
Training MSE – normalized: 0.0015
Testing MSE – normalized: 0.0026

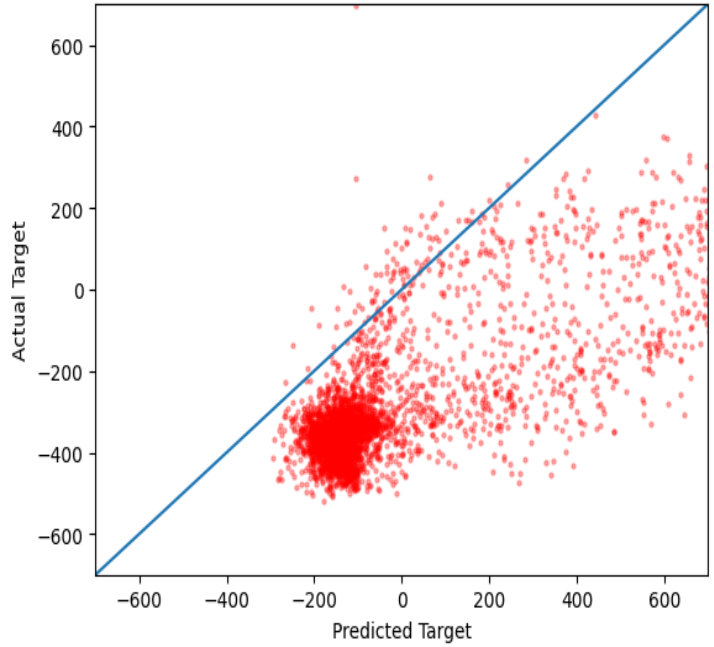
Training Set





Testing Set

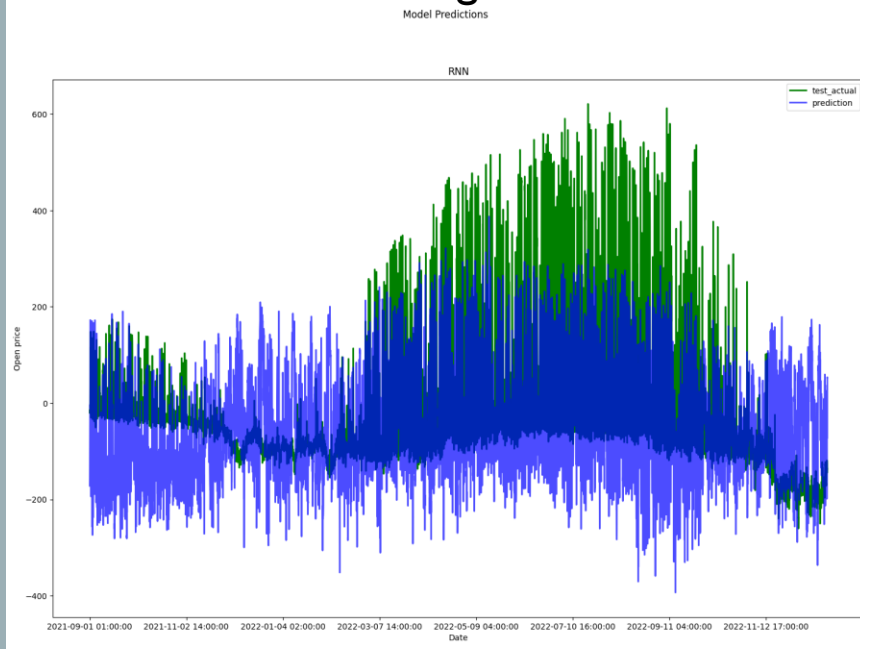


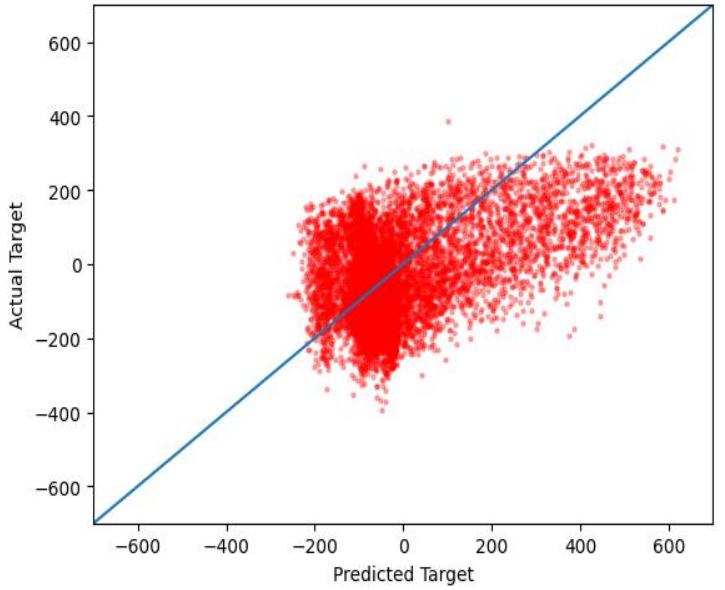


7 HIDDEN LAYERS 50 EPOCH TANH

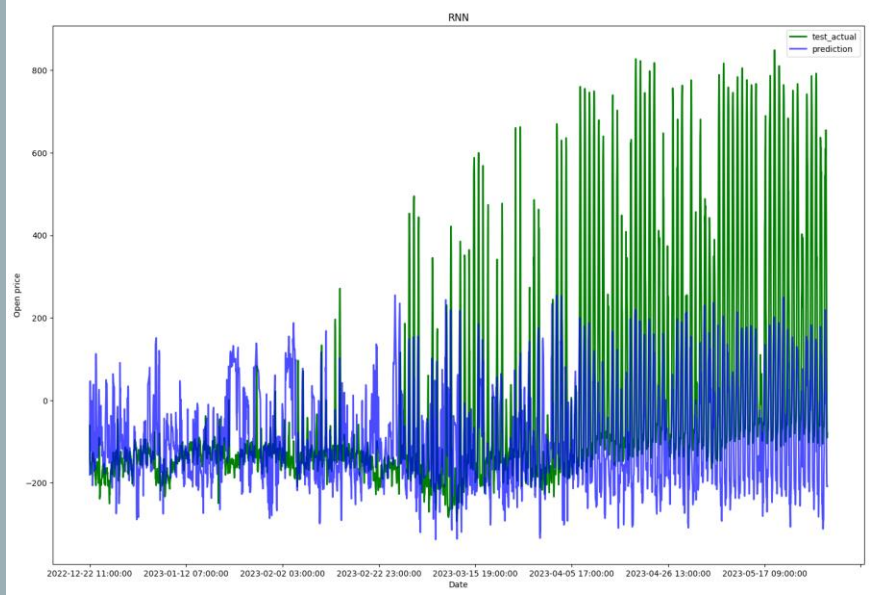
Training MSE – normalized: 0.0012
Testing MSE – normalized: 0.0023

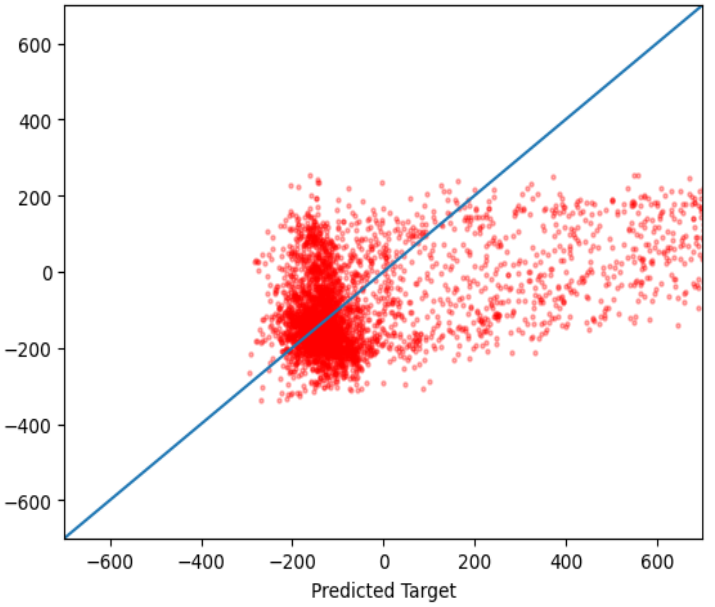
Training Set





Testing Set





MODEL RESULTS