Deep Learning – HW1 – Joe Ferguson

Step 1: Data – Questions

1) How many data samples are included in the dataset?

There are a total of 3047 data samples in the dataset.

2) Which problem will this dataset try to address?

We are trying to build a model to predict the TARGET_deathRate feature of the dataset according to the other features in the data set.

3) What is the minimum value and the maximum value in the dataset?

The min value of the entire dataset is 0 for multiple features (we don't have any nonzero data), the min value of target_deathrate is 59.7. The max value of the entire dataset is 10170292 for population Estimate 2015, the max value of target_deathrate is 362.8.

4) How many features in each data samples?

There are 33 features for each data sample, excluding the target_deathrate feature which would bring it to 34.

5) Does the dataset have any missing information? E.g., missing features.

Yes, for the features PctSomeCol18_24 , PctEmployed16_Over, and PctPrivateCoverageAlone we have some missing/null values. To be exact those numbers are 2285, 152, 609 respectively. Some percentage-based features have 0's as well, but those appear to be accurate.

6) What is the label of this dataset?

The label is TARGET_deathRate.

7) How many percent of data will you use for training, validation and testing?

As I have not taken machine learning at UC, I went off of the link below to Geeks for Geeks website to learn which split I should use. I thought 80% for training would be a little too much so I went for a 75% split for training. With the remaining 25 % split I split equally for validation and testing making the final split 75/12.5/12.5 (training, validation, testing)

Training vs Testing vs Validation Sets - GeeksforGeeks

8) What kind of data pre-processing will you use for your training dataset?

As I have not taken machine learning at UC, I went off of the link below to Geeks for Geeks website to learn what data preprocessing I should do.

One of the features I had to transform was the binnedInc feature, which was a range of floats, i.e [345.6, 56.76], I just replaced these values with the midpoint of the range.

For the geographical strings I used skl.LabelEncoder() to encode those values.

Data Preprocessing in Python - GeeksforGeeks

After I partitioned that data into training, validation, and testing, I dropped the label from the training data and then I converted all missing values to their respective means for each feature using df.fillna(df.mean()) from pandas and then I log-normalized the data for the features using np.log1p from numpy.

Finally, I used scikit-learn's StandardScaler() to standardize all features.

Step 2: Model

Model	Test R-squared	HyperParametersUsed
Linear Regression	0.7611	BatchSize=16, Learning Rate=0.012, epochs=100
DNN-16	0.8078	LeakyReLU(0.01), BatchSize=16, Learning Rate=0.0001, epochs=100
DNN-30-8	0.8197	LeakyReLU(0.01), BatchSize=16, Learning Rate=0.0001, epochs=100
DNN-30-16-8	0.8247	LeakyReLU(0.01), BatchSize=16, Learning Rate=0.0001, epochs=100

DNN-30-16-8-4	0.8733	LeakyReLU(0.01), BatchSize=16, Learning Rate=0.0001, epochs=100
DNN-8-4 (my personal)	0.8221	LeakyReLU(0.01), BatchSize=16, Learning Rate=0.001, epochs=100

Step 3: Objective

I used MSE as required from scikit-learn's library.

```
try:
    mse = mean_squared_error(y_np, preds)
except Exception:
    mse = np.nan
try:
    r2 = r2_score(y_np, preds)
except Exception:
    r2 = np.nan
```

Step 4: Optimization

I used SGD from torch.optim as required.

```
def train_model(model, bs, lr, epochs, verbose=True):
    criterion = nn.MSELoss()
    optimizer = optim.SGD(model.parameters(), lr=lr)
    train_losses, val_losses = [], []
```

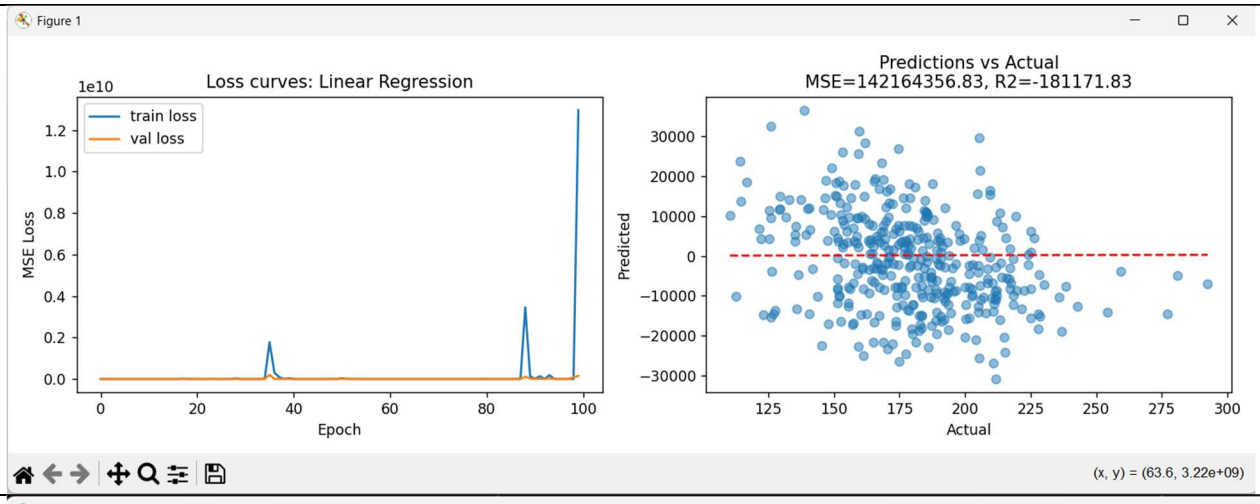
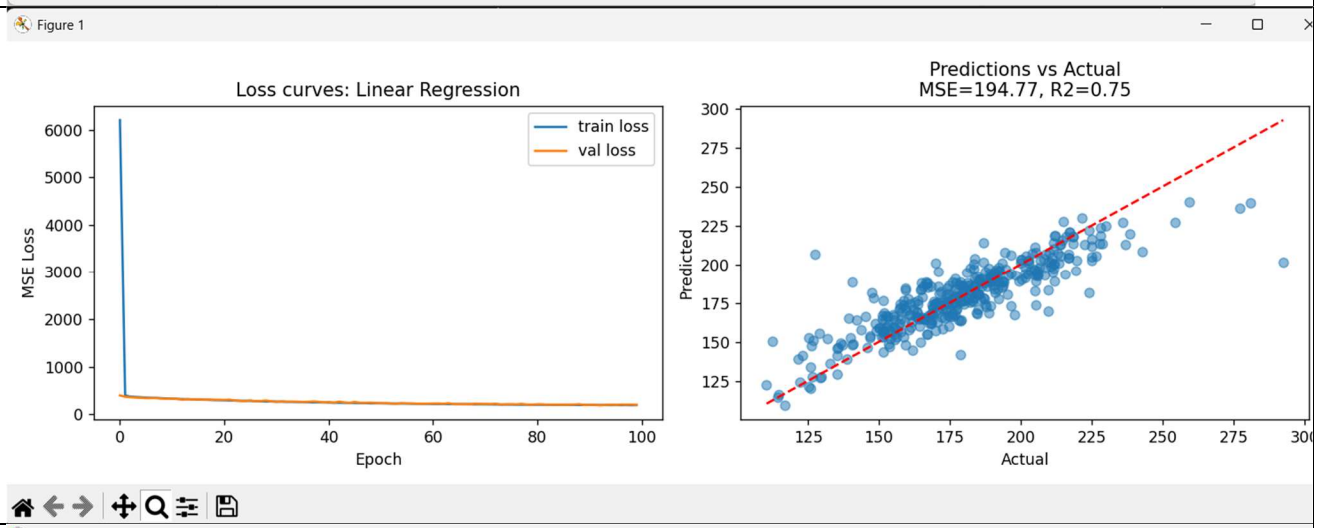
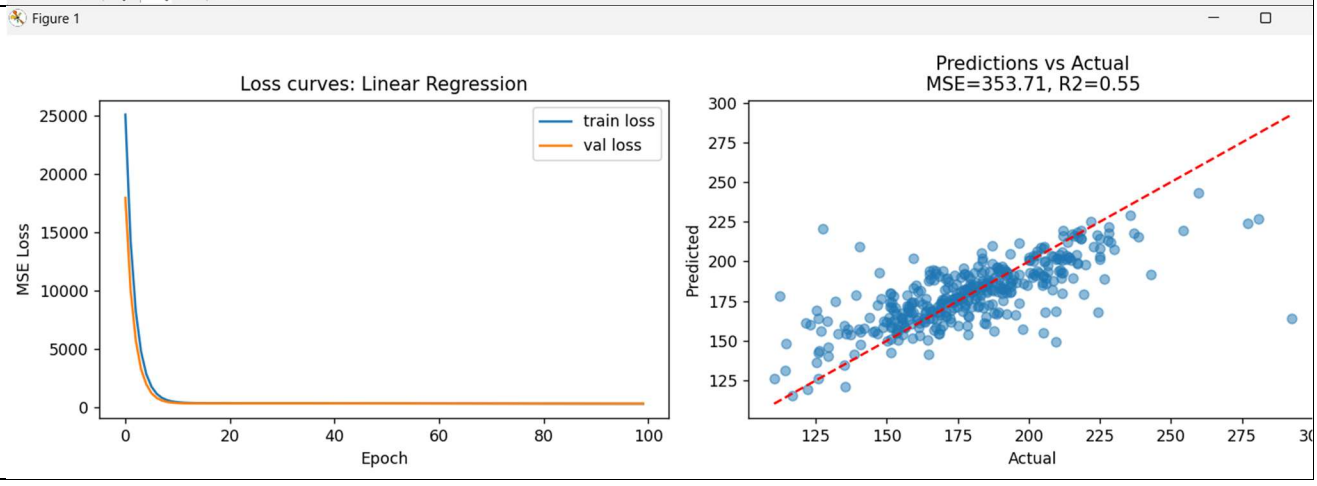
Step 5: Model Selection

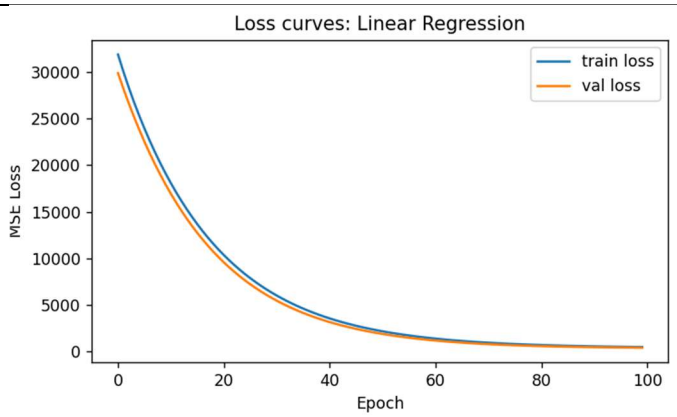
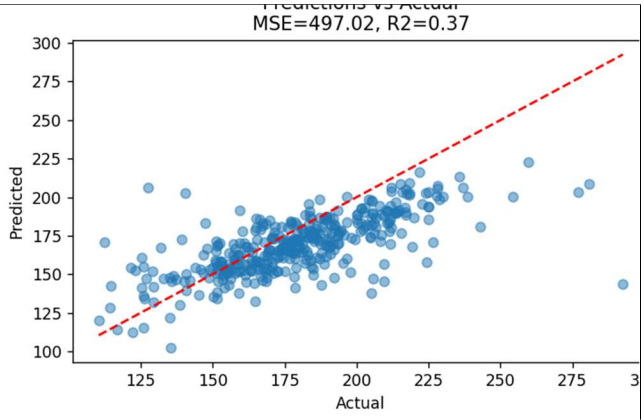
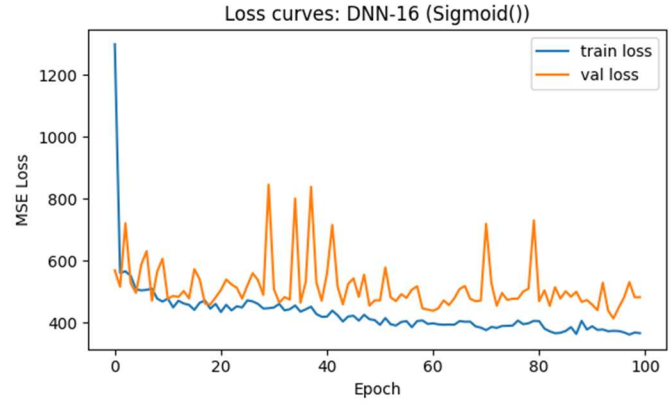
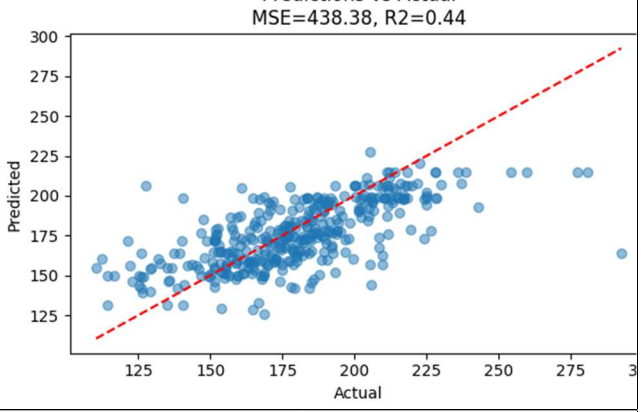
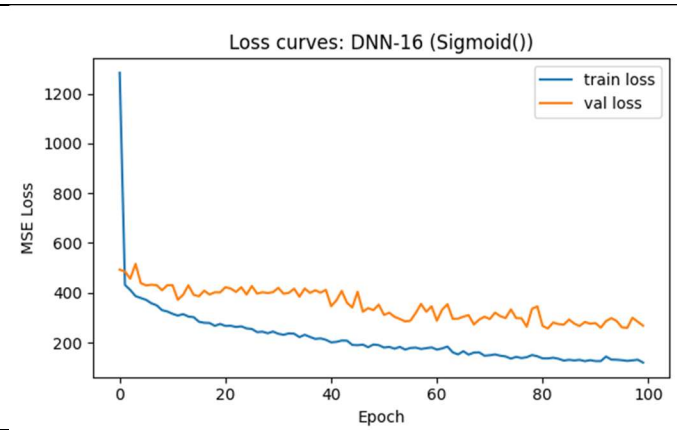
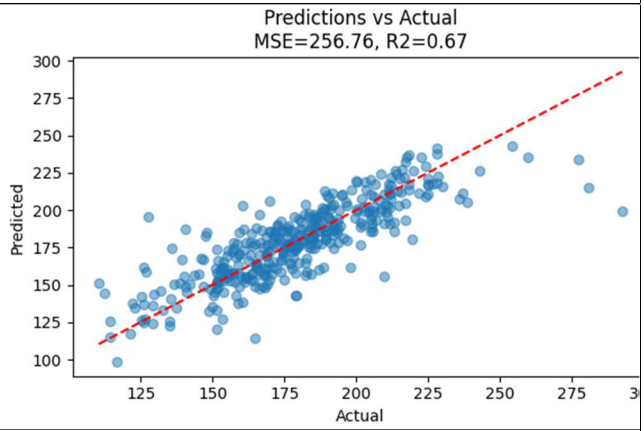
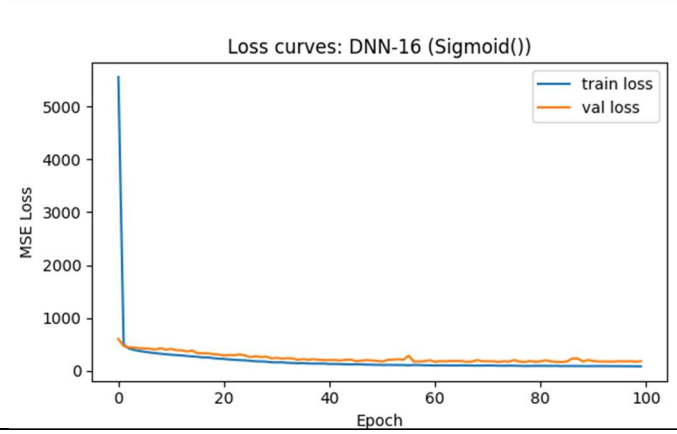
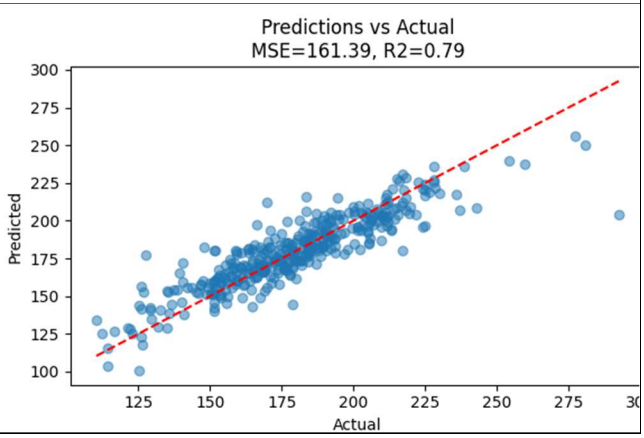
For these I will use Sigmoid – besides for Linear Regression, a batch size of 16, and 100 epochs. I used Sigmoid because ReLU and LeakyReLU have gradient "explosions" for high learning rates.

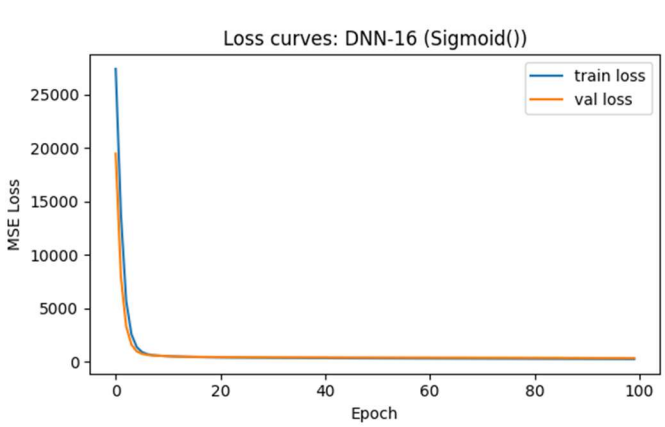
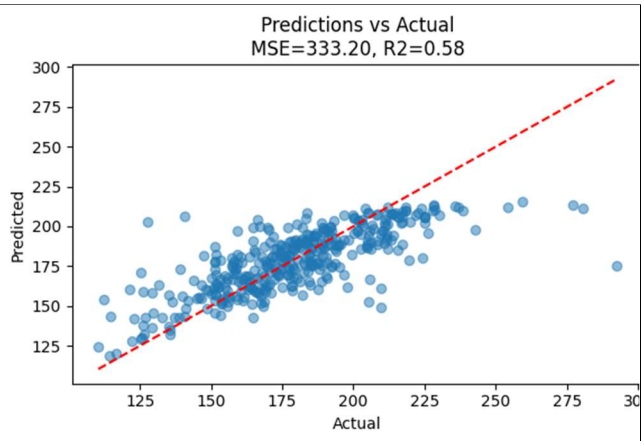
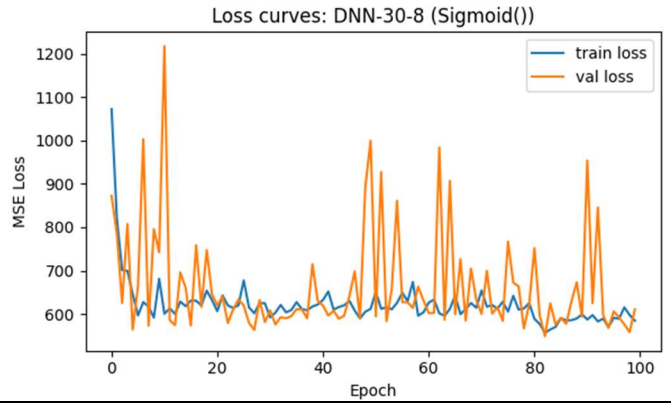
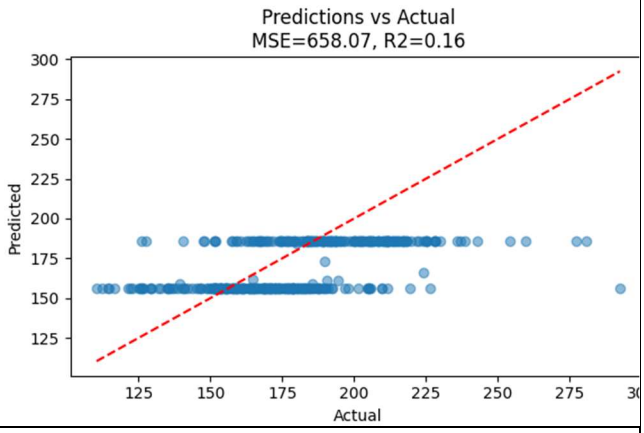
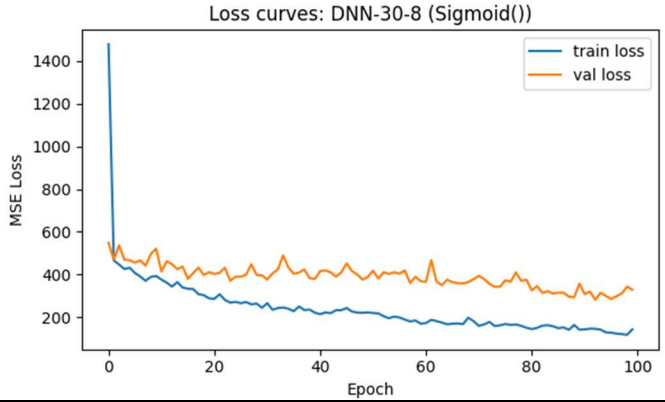
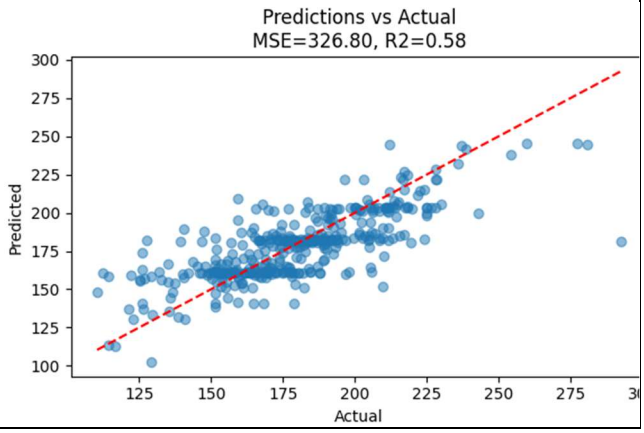
Model	LR: 0.1 (R^2)	LR: 0.01 (R^2)	LR: 0.001 (R^2)	LR: 0.0001 (R^2)
Linear Regression	-181171.8315 (The gradient exploded)	0.75	0.55	0.37
DNN-16	0.44	0.67	0.79	0.58
DNN-30-8	0.16	0.58	0.80	0.68
DNN-30-16-8	-2.48	0.57	0.78	0
DNN-30-16-8-4	-0.01	0.51	0.5	0

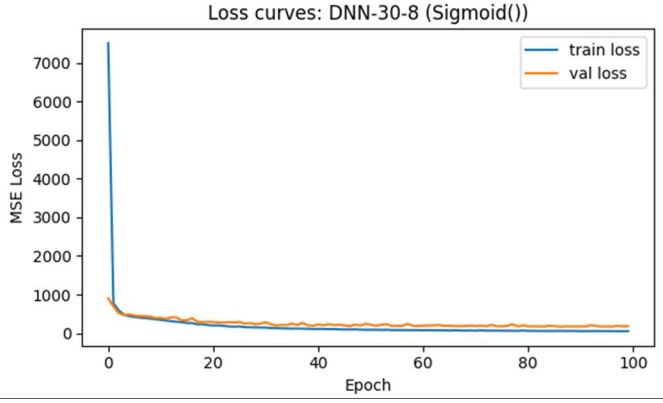
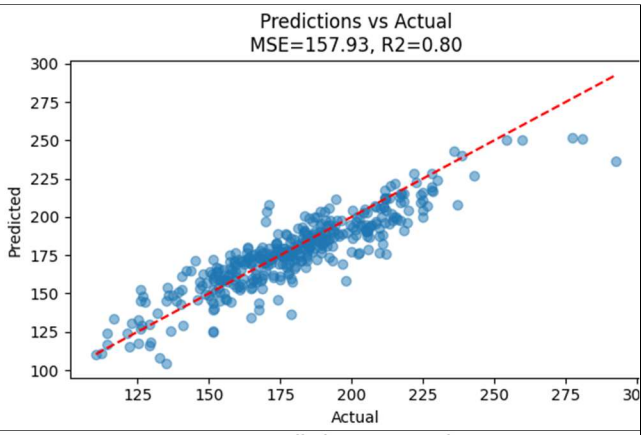
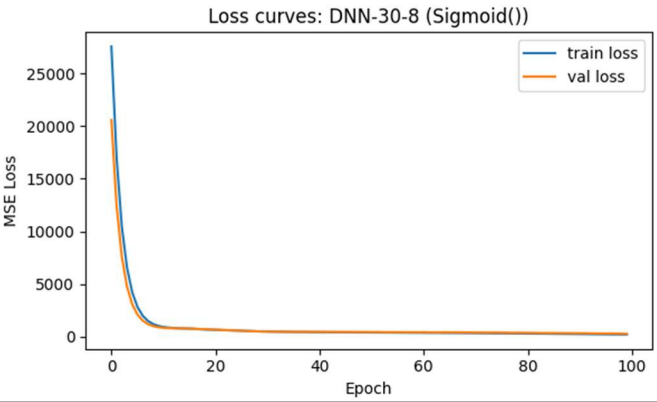
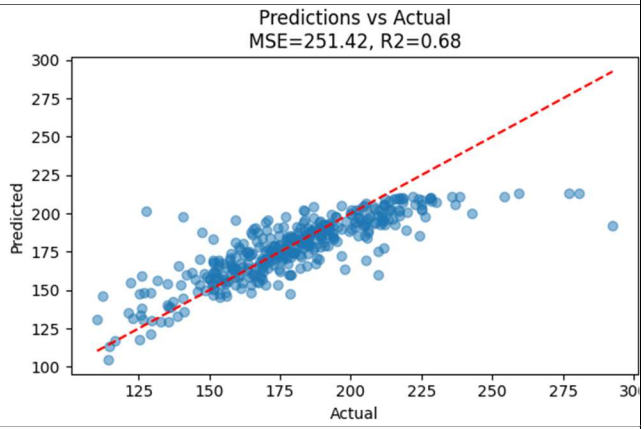
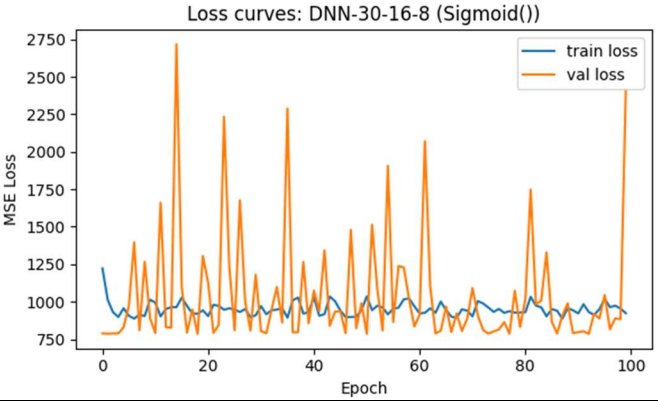
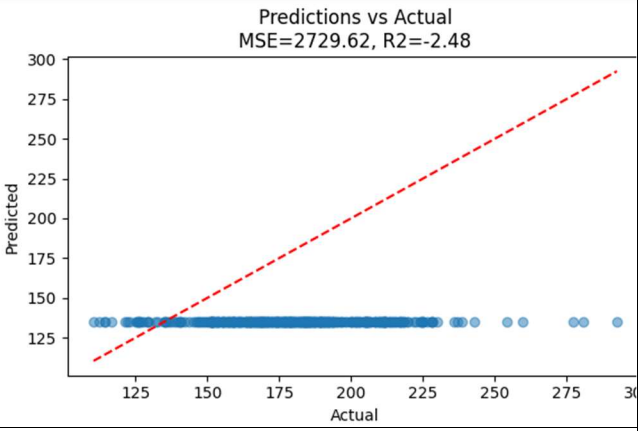
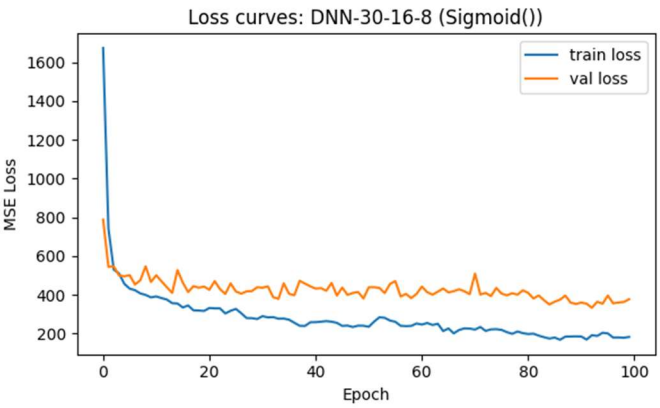
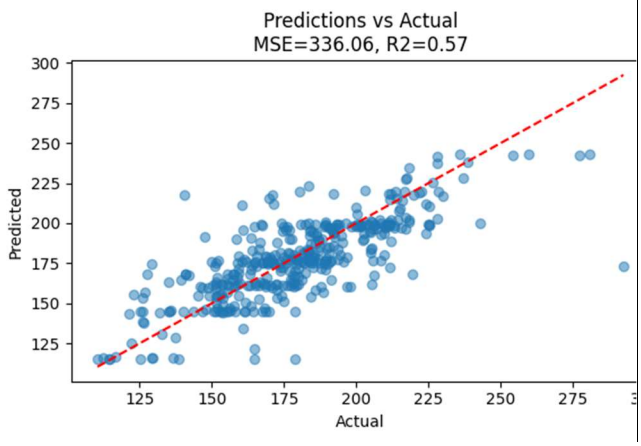
DNN-8-4	0.2	0.57	0.82	0.62

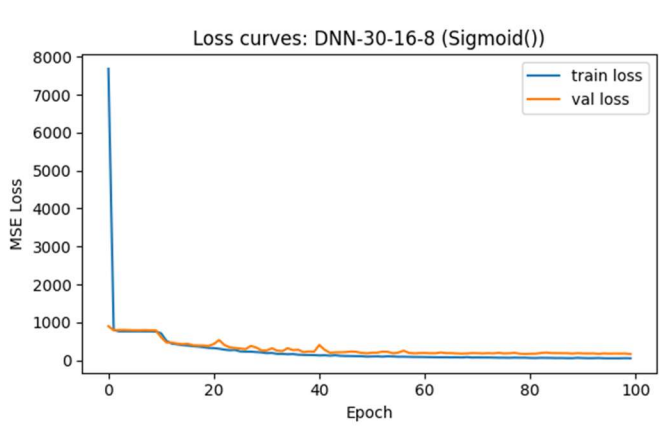
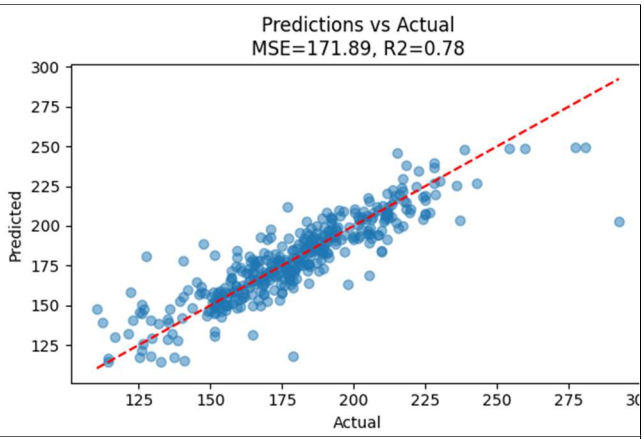
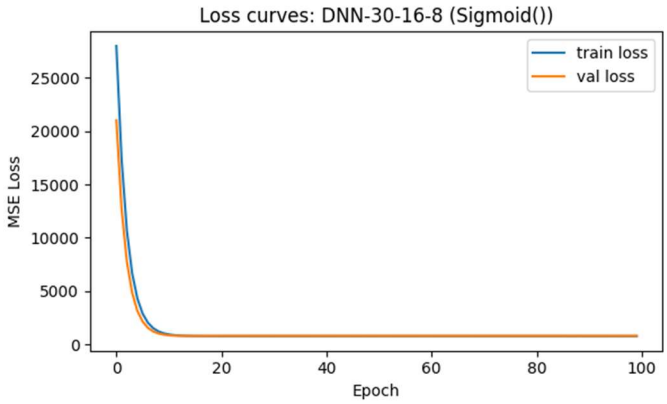
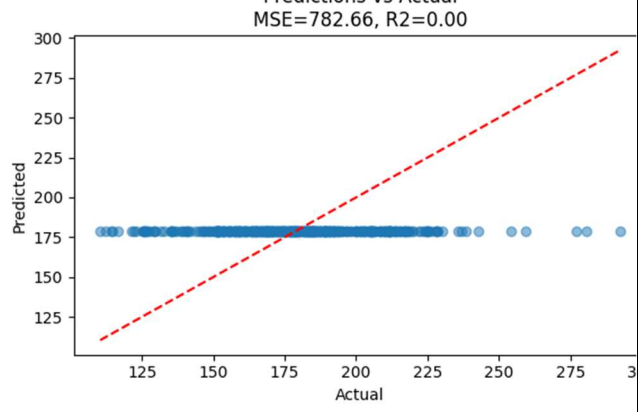
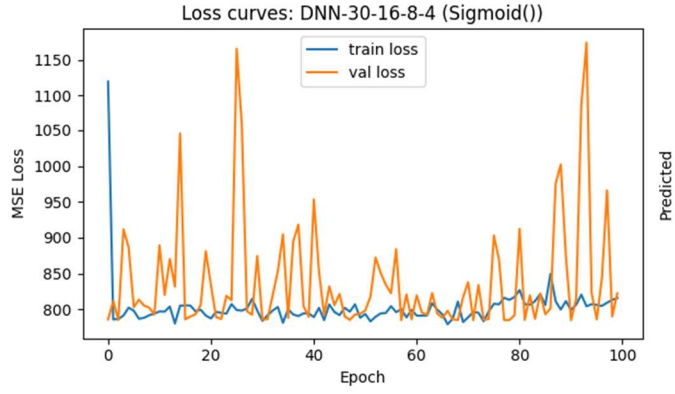
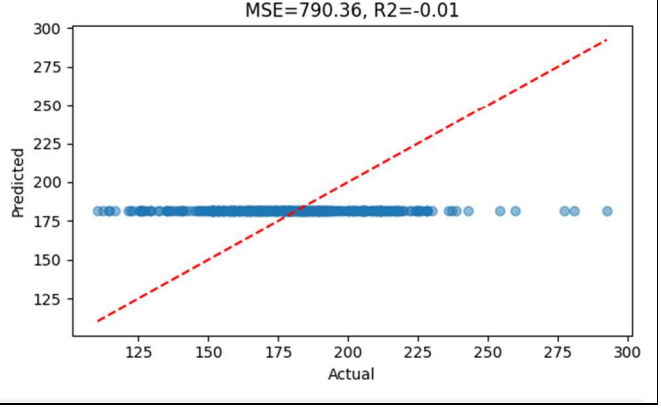
Step 6: Model Performance

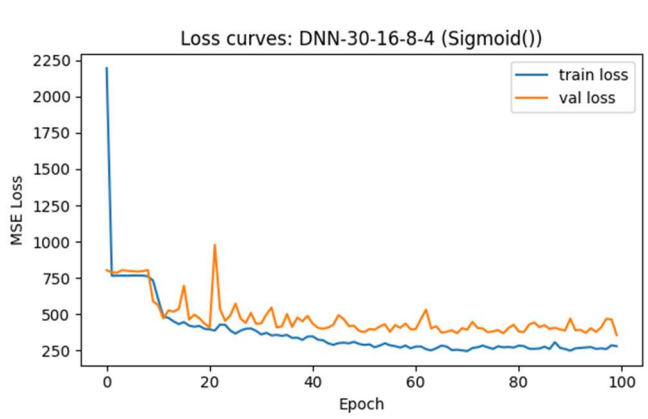
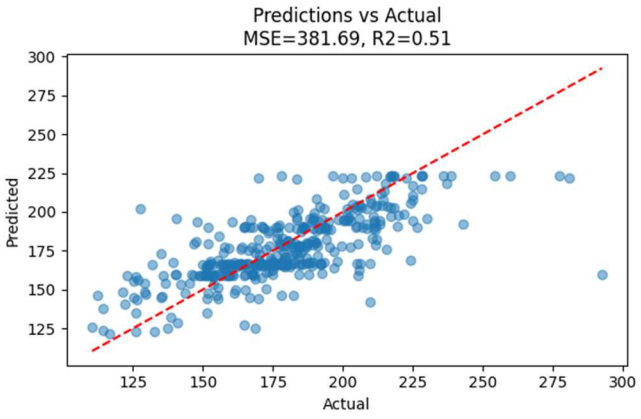
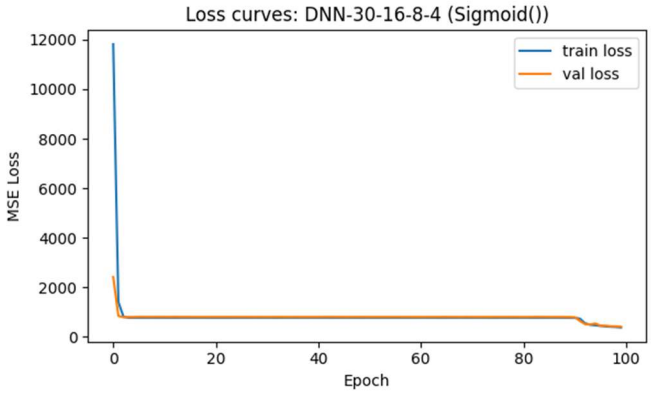
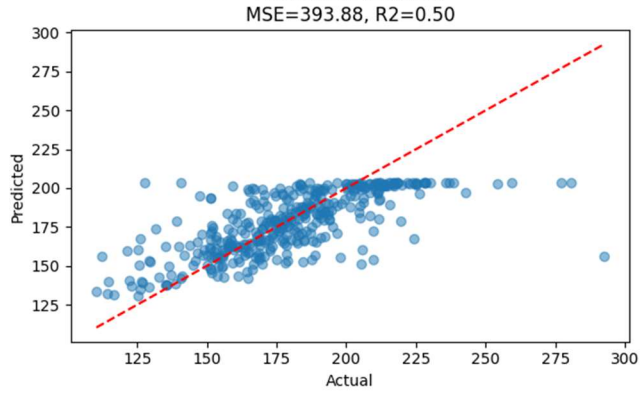
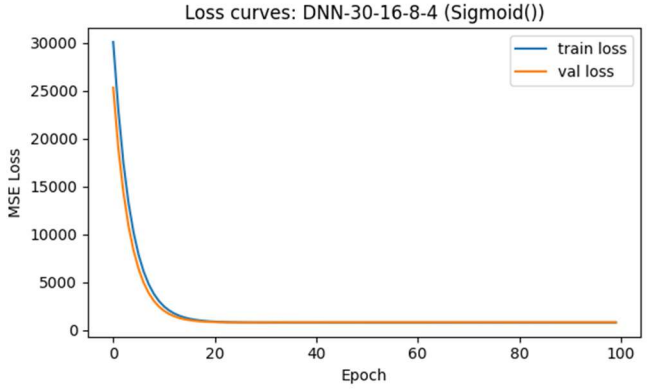
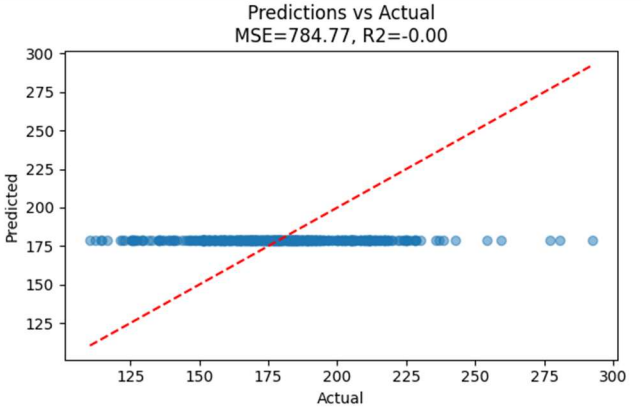
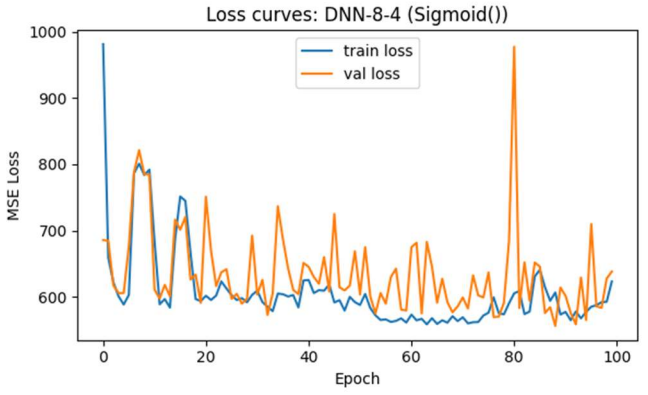
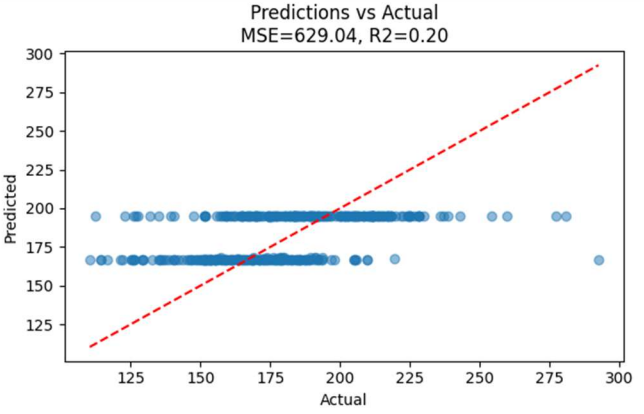
Model	Performance + Plot
Linear Regression LR:0.1	
Linear Regression LR:0.01	
Linear Regression LR:0.001	

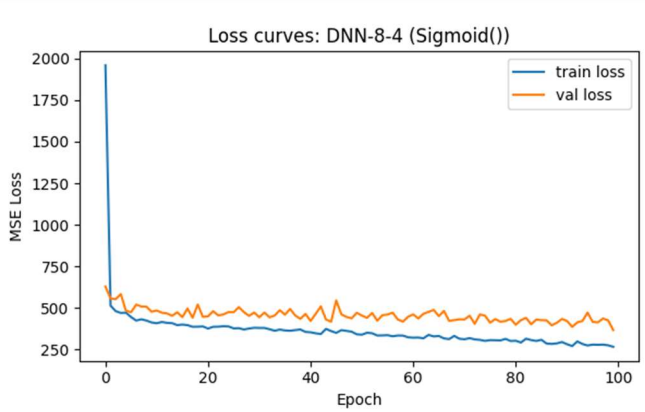
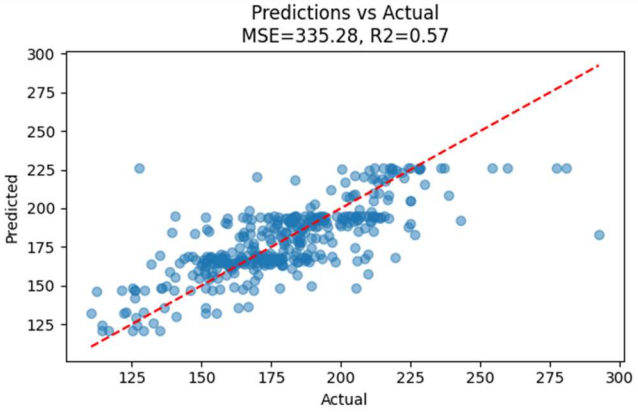
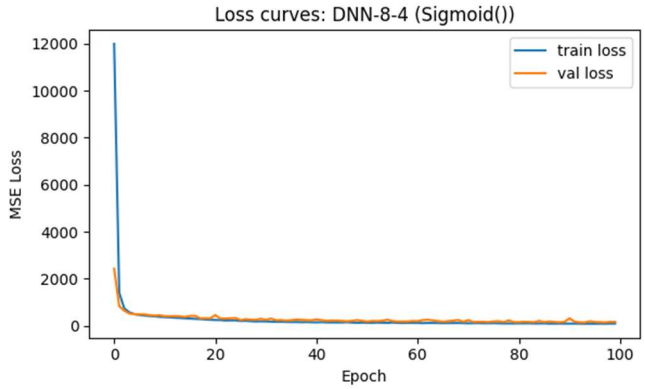
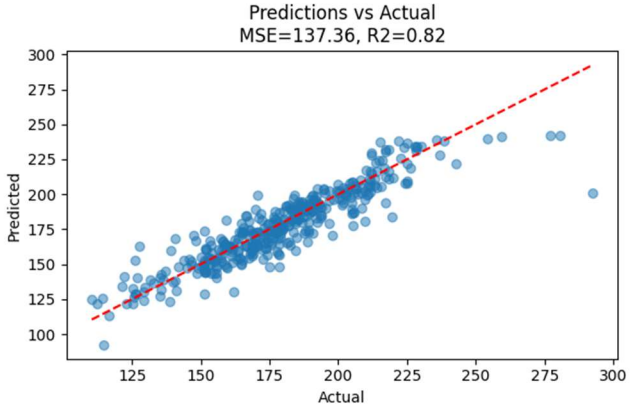
Linear Regression LR:0.0001		
DNN-16 LR:0.1		
DNN-16 LR: 0.01		
DNN-16 LR: 0.001		

DNN-16 LR: 0.0001	 
DNN-30-8 LR:0.1	 
DNN-30-8 LR:0.01	 

DNN-30-8 LR:0.001		
DNN-30-8 LR:0.0001		
DNN-30-16-8 LR:0.1		
DNN-30-16-8 LR:0.01		

DNN-30-16-8 LR:0.001	 
DNN-30-16-8 LR:0.0001	 
DNN-30-16-8-4 LR:0.1	 

DNN-30-16-8-4 LR:0.01	 
DNN-30-16-8-4 LR:0.001	 
DNN-30-16-8-4 LR:0.0001	 
DNN-8-4 LR:0.1	 

DNN-8-4 LR:0.01	 
DNN-8-4 LR:0.001	 
DNN-8-4 LR:0.0001