**James Folk**

**DATA 527 – Predictive Modeling**

**Assignment 4**

**DEADLINE: April 11, 2024**

**Spring 2024**

**Overview**

The purpose of this exercise is to implement a feed forward neural network to predict the car market prices. The main goal is to work on data exploration and preparation beside the network model implementation. I am using a dataset that contains 9 columns with 5512 entries. I leveraged the Tensor SDK to implement the feed forward neural network.

Create the neural network using the Tensor SDK

Clean the dataset.

Determine which data is relevant

**Methodology**

* **Create the Neural Network:** The neural network architecture was established with a Keras sequential model, comprising an input layer, two hidden layers, and an output layer. Each hidden layer is composed of 14 neurons employing a Rectified Linear Unit (ReLU) activation function, chosen due to the positive nature of all inputs. The input layer accommodates the number of input neurons corresponding to the dimensions utilized in training the model. Conversely, the output layer consists of a single neuron, as the objective is solely to predict car prices based on their attributes.
* **Clean the dataset:** Upon data loading, I meticulously traversed each column, constructing a dictionary for tallying the occurrence of each value. This process aimed to uncover any aberrations within the dataset. Instances exhibiting nonsensical anomalies prompted the exclusion of their respective rows. Entries devoid of numerical values were handled through enumerations. As for numerical entries, I initially planned to utilize them directly, but upon observation, I noted price entries were suffixed to denote their scale. Leveraging this suffix, I scaled the prices accordingly. Notably, the manufacturing year column provided numerical data, which I utilized to gauge the age of the vehicles rather than relying on the actual calendar year. I opted to apply z-score normalization to scale each entry, ensuring effective handling of any anomalies.
* **Determine the relevant data:** Initially, it was evident not to utilize the name column for training the neural network. Upon running the algorithm to tally occurrences in the column, I observed numerous distinct engine sizes. Omitting the engine size column from training led to price predictions closer to reality. Consequently, I enhanced the program's flexibility to allow parameter selection via arguments. Leveraging TensorFlow's capability to save models to disk, I systematically iterated through various parameter combinations to train the neural network, aiming to identify the optimal combination.
* **Calculation the error:** Implemented the estimate\_r\_squared function, utilizing the trained model to predict dependent values based on the dependent values.

**Implementation**

A black background with a black square

Description automatically generated with medium confidence

**Results**

**A screenshot of a computer

Description automatically generated**

**Neural Network Topography**

There are three layers.

**A screen shot of a graph

Description automatically generated**

**Errors per iteration**

{

"learningRate": 0.001,

"iterations": 10000,

"final mse": 0.6220643031721588,

"r value": -0.5478559975622943,

"neural network": {

"layers": [

{

"neurons": [

{

"bias": 1,

"weights": [

0.18144856003867058,

0.5915557182523438

]

},

{

"bias": 1,

"weights": [

0.5705911571075168,

0.34895927025115503

]

}

]

},

{

"neurons": [

{

"bias": 0.6796229808028796,

"weights": [

0.6849640803187985

]

},

{

"bias": 0.7192036709557639,

"weights": [

0.7431559071997575

]

}

]

},

{

"neurons": [

{

"bias": 0.9999977865596916,

"weights": []

}

]

}

]

}

}

**Discussion**

Challenges Faced and Solutions

* **Determining the Neural Network structure:** I am not quite sure if I am doing what is expected for the stochastic and batch gradient descent functionality in the tests.
* **Determine how to use the data in the Neural Network:** As I was trying to determine the best learning rate, it seemed not follow a linear cause and effect.
* **Determine which variables to use:** It might have not been the best topology to have only one hidden layer with two nodes.

**Conclusion**

In this work, we investigated the prediction of dependent values using a neural network, with a focus on independent values.

**References**

1. Pfeiffer, Simon. “Creating Your First Neural Network in Python w/ Tensorflow.” *DEV Community*, DEV Community, 14 Aug. 2021, dev.to/codesphere/creating-your-first-neural-network-in-python-w-tensorflow-4l5p.