James Folk

DATA 527 – Predictive Modeling

Project 2

DEADLINE: May 6, 2024

Spring 2024

Overview

The purpose of this exercise is to implement a feed forward neural network to predict the pitch of an airplane based on previous values of altitude, indicated airspeed, and roll. The main goal is to work on data exploration and preparation beside the network model implementation. I am using a dataset that contains 3 columns with 5404 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, five hidden layers, and an output layer. Each hidden layer is composed of 20 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 airplane pitch values based on their attributes.
- **Normalize the dataset:** I opted to apply min max scaling normalization to scale each entry, ensuring effective handling of any anomalies.

Implementation

$articlealgorithmalgorithmic$

Pseudo-code for Car Price Predictor Functions

1 Pseudo-code for Each Function

1.1 Function: normalizeMinMaxScaling

normalizeMinMaxScaling(ary) [1] length of $ary > 0$ $max\_val \leftarrow$ maximum value in ary $min\_val \leftarrow$ minimum value in ary $new\_ary \leftarrow []$ each $item$ in ary $normalized\_item \leftarrow$ normalizeMinMaxScalingValue($item$, min_val, max_val) add $normalized\_item$ to new_ary new_ary, min_val, max_val ary, 0, 0

1.2 Function: normalizeMinMaxScalingValue

normalizeMinMaxScalingValue(val, $minimum$, $maximum$) [1] $(val - minimum)/(maximum - minimum)$

1.3 Function: open

open($filename$) [1] Open the file with the given $filename$ for reading Read the contents of the file Parse the data from the file Close the file

1.4 Function: predictPrice

predictPrice($mileage$, $fuelType$, $transmission$, $ownership$, $manufactureYear$, $engine$, $seats$) [1] input is valid Prepare the input tensor Use the trained model to predict the price predicted price Print "Invalid Input" None

1.5 Function: __learn

__learn($useMilage$, $useFuelType$, $useTransmission$, $useOwnership$, $useManufacture$, $useEngine$, $useSeats$) [1] Set the usage flags for data preprocessing Preprocess the data Normalize the features Create a neural network model Train the model Save the trained model

1.6 Function: __processCarName

__processCarName(var) [1] var

1.7 Function: __processCarPrice

__processCarPrice(var) [1] Parse the price value and scale from the input string Convert the price to a standard currency (e.g., USD) Update the price dictionary The converted price

1.8 Function: __processMilage

__processMilage(var) [1] Parse the mileage value from the input string Update the mileage dictionary The parsed mileage value

1.9 Function: __processFuelType

__processFuelType(var) [1] Update the fuel type dictionary The processed fuel type value

1.10 Function: __processTransmission

__processTransmission(var) [1] Update the transmission dictionary The processed transmission value

1.11 Function: __processOwnership

__processOwnership(var) [1] Update the ownership dictionary The processed ownership value

1.12 Function: __processManufacture

__processManufacture(var) [1] Calculate the age of the car based on the manufacturing year The calculated age

1.13 Function: __processEngine

__processEngine(var) [1] Parse the engine value from the input string Update the engine dictionary The parsed engine value

1.14 Function: __processSeats

__processSeats(var) [1] Update the seats dictionary The processed seats value

1.15 Function: process

process($useMilage$, $useFuelType$, $useTransmission$, $useOwnership$, $useManufacture$, $useEngine$, $useSeats$) [1] Call __learn function with provided usage flags

Results

Learning Rate: 0.001

Number of Iterations: 500

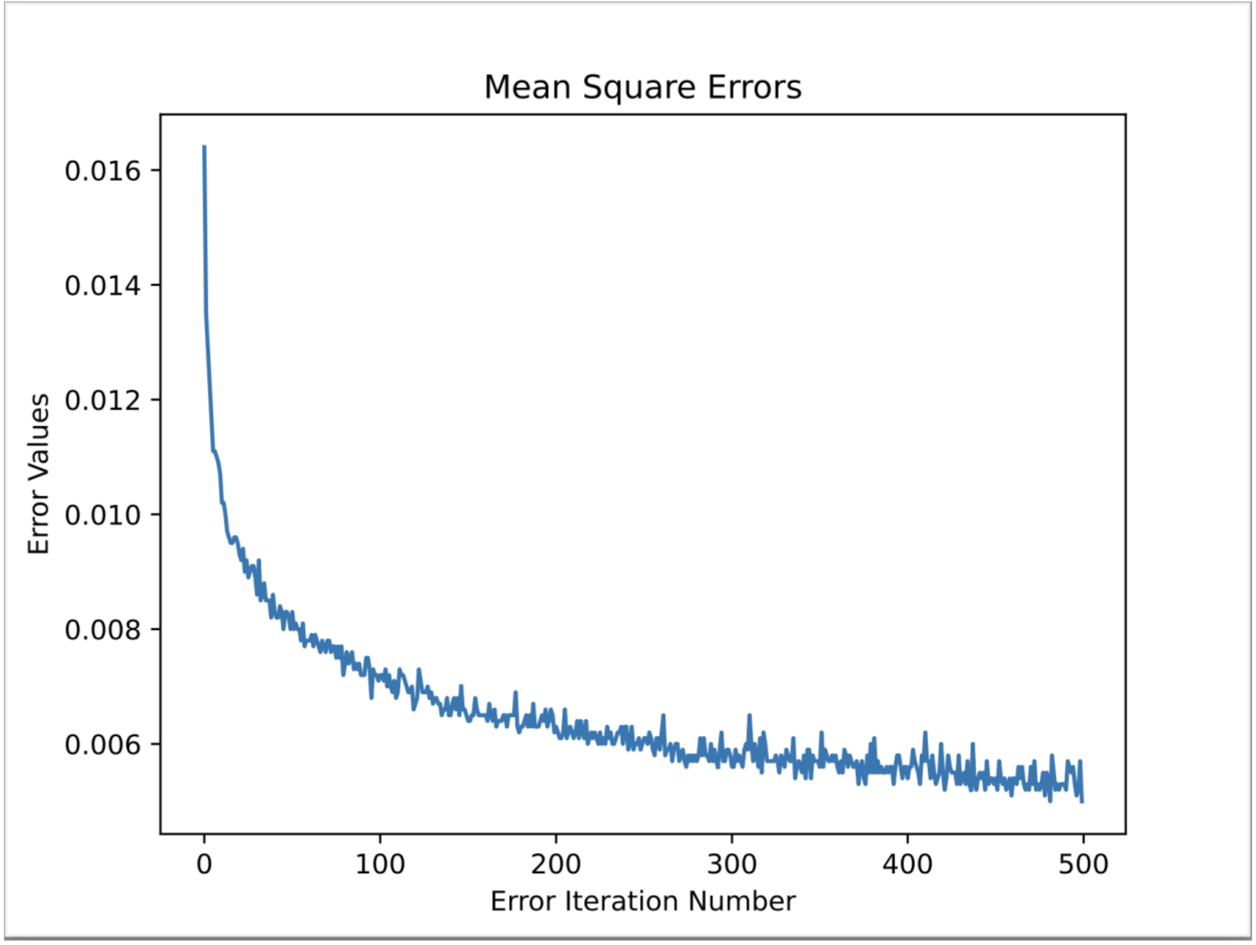
Final Coorelation Coefficent: 0.7293365435980965

Structure:

3 input nodes

5 hidden layers with 40 nodes each

1 output node



Mean Square Errors

Discussion

Challenges Faced and Solutions

- **Determining the Hidden Layer Structure:** Analyzing the neural network's optimal setup was the major issue encountered, specifically in determining the number of nodes and hidden layers in the model. This configuration consisted of 5 hidden layers, each with 40 nodes. The process was iterative which showed that increasing the number of layers and nodes had an improved accuracy in prediction. R-squared value of 0.4 was obtained when a single layer with 3 nodes was used which was very low. This structure was then adjusted to have 3 layers with each having 20 nodes resulting in R-squared equal to 0.6.

Conclusion

To enable the neural network to accurately determine pitch based on the provided data, this represents the culmination of extensive refinement efforts. Initially, structuring the model effectively posed a significant challenge, necessitating iterative adjustments, particularly concerning the allocation of hidden layers and their nodes. Progress was incremental, ultimately leading to the adoption of a configuration featuring five hidden layers, each comprising 40 nodes, resulting in a substantial enhancement in predictive accuracy. This is evidenced by the achieved final correlation coefficient of 0.729. Considering the potential for further optimization, it becomes imperative to continually enhance our models, as varied inquiries demand tailored solutions.

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.
2. Netron. (2024, April 28). e
3. Team, K. (n.d.). *Keras Documentation: Adam*. https://keras.io/api/optimizers/adam/