**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. I found that the optimal combination is if you use Milage, Fuel Type, Transmission, Manufacture, and Engine. I have recorded all of the Correlation Coefficients for each of the different combinations in the file: data/r\_values.csv.

**Implementation**

A black background with a black square

Description automatically generated with medium confidence

**Results**

Learning Rate: 0.001

Number of Iterations: 2000

Final Coorelation Coefficent: 0.449060499

Structure:

5 input nodes

2 hidden layers with 14 nodes each

1 output node

Note: Neural Network can be viewed by loading the file data/\_milage\_fueltype\_transmission\_manufacture\_engine\_model.keras into https://netron.app/

Weights Hidden Layer 1

[

[

-20.052753448486328,

30.41869354248047,

-29.340452194213867,

-26.755163192749023,

-23.625890731811523,

-13.583880424499512,

-23.506765365600586,

-20.584434509277344,

-0.5824560523033142,

-30.154993057250977,

-13.516119003295898,

0.08546847105026245,

0.02979453280568123,

-7.395495414733887

],

[

-14.043724060058594,

8.590750694274902,

-19.176198959350586,

-13.518379211425781,

-15.981616020202637,

-11.147561073303223,

-21.66584587097168,

-13.772665023803711,

-6.756702899932861,

-19.642419815063477,

-12.052927017211914,

-0.16812050342559814,

-0.2175636887550354,

-15.7616548538208

],

[

8.07701301574707,

-4.358405113220215,

6.33493709564209,

8.354583740234375,

8.19600772857666,

8.607816696166992,

6.061176776885986,

8.667421340942383,

6.824980735778809,

6.4689788818359375,

8.51092529296875,

-0.5045139789581299,

-0.04380077123641968,

4.99353551864624

],

[

2.3172948360443115,

26.847137451171875,

-10.835719108581543,

-16.00691032409668,

-6.125864505767822,

-7.477907657623291,

-1.26982843875885,

-3.9746291637420654,

-15.358009338378906,

-11.698892593383789,

-1.0543980598449707,

-0.3652895390987396,

-0.15106476843357086,

-21.81543731689453

],

[

12.495899200439453,

2.8119852542877197,

0.3566901981830597,

4.2760329246521,

6.1253485679626465,

10.185563087463379,

8.476723670959473,

8.783164024353027,

3.3703739643096924,

-0.03924328461289406,

12.515473365783691,

-0.28745418787002563,

0.013660584576427937,

2.499056100845337

]

]

Weights Hidden Layer 2:

[

[

0.29018691182136536,

4.2458815574646,

0.28208664059638977,

-0.3593585789203644,

10.509090423583984,

11.1644287109375,

10.635229110717773,

11.109740257263184,

13.123650550842285,

11.346488952636719,

8.37028980255127,

3.9906511306762695,

-3.332224130630493,

12.415366172790527

],

[

-0.32660341262817383,

14.498351097106934,

-0.38898149132728577,

-0.4234330952167511,

-23.5913028717041,

-23.608083724975586,

-25.869813919067383,

-25.931623458862305,

-13.358104705810547,

-25.785335540771484,

14.957459449768066,

14.650092124938965,

26.8072566986084,

-11.116665840148926

],

[

-0.0036285454407334328,

0.8365309238433838,

0.3511177599430084,

-0.05414068326354027,

3.485286235809326,

3.575913429260254,

2.814467191696167,

2.819186210632324,

4.49971866607666,

3.3046510219573975,

-1.6195042133331299,

1.488612413406372,

-55.53550338745117,

4.683704376220703

],

[

0.1945963352918625,

-9.035855293273926,

-0.4516768455505371,

-0.20171284675598145,

9.488065719604492,

9.501992225646973,

9.468941688537598,

9.040873527526855,

10.212335586547852,

8.979424476623535,

3.7082860469818115,

-9.08509635925293,

-10.310453414916992,

9.826239585876465

],

[

-0.40215620398521423,

-1.4485477209091187,

0.27770987153053284,

-0.3232836127281189,

10.212333679199219,

9.403546333312988,

9.605029106140137,

9.540020942687988,

10.509519577026367,

9.809808731079102,

5.335829734802246,

-2.0409328937530518,

-10.018447875976562,

11.597524642944336

],

[

-0.3471307158470154,

-1.2035812139511108,

-0.43950963020324707,

0.053992658853530884,

8.27929401397705,

8.406825065612793,

8.35413646697998,

8.500100135803223,

10.937427520751953,

8.844594955444336,

5.806162357330322,

-1.1686549186706543,

-0.11704990267753601,

10.74338436126709

],

[

-0.36090466380119324,

8.024884223937988,

0.2406000792980194,

0.13039454817771912,

9.807088851928711,

9.674164772033691,

8.6563081741333,

8.087577819824219,

10.1550874710083,

8.997333526611328,

3.9617109298706055,

8.376510620117188,

-9.553631782531738,

10.553081512451172

],

[

-0.08214820921421051,

0.9656528830528259,

-0.2689742147922516,

-0.5125463604927063,

10.247309684753418,

10.16677188873291,

10.424769401550293,

10.52171802520752,

11.508100509643555,

10.740052223205566,

6.24664831161499,

0.6990911960601807,

-6.826329708099365,

12.260415077209473

],

[

0.014798125252127647,

-6.145018577575684,

-0.38072630763053894,

0.29146599769592285,

6.399423599243164,

5.80436372756958,

5.523807525634766,

5.29503870010376,

8.383744239807129,

5.125583171844482,

4.184746742248535,

-6.259167194366455,

6.021603107452393,

8.85302448272705

],

[

-0.2924669086933136,

0.7864699363708496,

0.24788862466812134,

-0.3676997125148773,

2.783468246459961,

2.5079805850982666,

2.817003011703491,

2.06425142288208,

3.871401786804199,

2.433901309967041,

-2.298661708831787,

0.807013213634491,

-61.54900360107422,

3.681018829345703

],

[

-0.1011292040348053,

1.0577698945999146,

0.25430625677108765,

-0.06071990355849266,

9.394059181213379,

9.66377067565918,

9.720389366149902,

9.529339790344238,

11.950811386108398,

9.086721420288086,

5.990633487701416,

1.7497689723968506,

-1.0540850162506104,

11.291552543640137

],

[

0.37094253301620483,

-0.22734840214252472,

-0.4270993769168854,

-0.3072996735572815,

0.2993580102920532,

-0.40661901235580444,

0.37257009744644165,

-0.18202435970306396,

-0.38177135586738586,

-0.10379579663276672,

-0.0755968689918518,

-0.31609398126602173,

0.15632355213165283,

-0.10490530729293823

],

[

-0.23029407858848572,

-0.3859885632991791,

0.2380625605583191,

0.22085869312286377,

-0.3932126462459564,

-0.39950668811798096,

-0.2945871651172638,

-0.4247731864452362,

-0.019371621310710907,

0.3383130729198456,

0.4607239365577698,

0.14808040857315063,

0.43212127685546875,

-0.10667785257101059

],

[

0.01466402132064104,

-8.595112800598145,

-0.38379967212677,

0.0597863495349884,

6.255191802978516,

6.566129684448242,

5.18014669418335,

4.954437732696533,

8.054862022399902,

4.987004280090332,

3.1765568256378174,

-9.229060173034668,

2.3204433917999268,

8.621097564697266

]

]

Model: "sequential\_51"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓

┃ Layer (type) ┃ Output Shape ┃ Param # ┃

┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩

│ dense\_153 (Dense) │ (None, 14) │ 84 │

├─────────────────────────────────┼────────────────────────┼───────────────┤

│ dense\_154 (Dense) │ (None, 14) │ 210 │

├─────────────────────────────────┼────────────────────────┼───────────────┤

│ dense\_155 (Dense) │ (None, 1) │ 15 │

└─────────────────────────────────┴────────────────────────┴───────────────┘

Total params: 929 (3.63 KB)

Trainable params: 309 (1.21 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 620 (2.43 KB)

A diagram of a function

Description automatically generated

**Neurl Network Layout**

**Discussion**

Challenges Faced and Solutions

* **Determining which variables are important:** I knew for sure that Car Name was unnecessary. I noticed that there was a unit of measurement on the prices, so I made sure to normalize that by scaling the number by the unit. I categorized the Fuel Type, Transmission, Ownership and Seats and labeled them with numbers. I thought that since there was so many engine sizes, it wouldn’t give a good measurement, so I was going to leave them out. I decided to train every combination of dimension between milage, fuel type, transmission, ownership, manufacture, engine, and seats.

**Conclusion**

It took several days to train the 127 different models, so I didn’t want to redo anything once I made my choices. I am curious if I can get a better correlation coefficient if I categorized the engine sizes. The combination that gave the best correlation coefficient between milage, fuel type, transmission, ownership, manufacture, engine, and seats were to omit the use of ownership and seats.

**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). https://netron.app/
3. Team, K. (n.d.). *Keras Documentation: Adam*. https://keras.io/api/optimizers/adam/