CS235 Fall'23 Project Midterm Report

LIAM Y. HSIEH (LHSIE013), University of California, Riverside, U.S.A

This midterm report provides an overview of the ongoing CS235 project focused on car sales price prediction. It encompasses the exploration of linear regression as a baseline and delves into the evaluation of 81 distinct Artificial Neural Network (ANN) models. The comparative analysis is based on Mean Absolute Error (MAE) and aims to identify optimal predictive modeling approaches.

Additional Key Words and Phrases: Car sales price prediction, Neural Networks, Mean Absolute Error (MAE)

1 CURRENT STATE

The high-level flow for proposed approach is shown as Fig. 1.

The presented flowchart outlines the sequential steps in a data-driven predictive modeling process. It begins with raw data, progresses through data cleaning and preprocessing, including MinMax normalization. The dataset is then split into training and testing sets with an 80:20 ratio. Two main branches of modeling are pursued: one involves Linear Regression as baseline, and the other explores 81 distinct Artificial Neural Network (ANN) models. The final step involves a comprehensive evaluation of model performance using Mean Absolute Error, providing a comparative analysis between Linear Regression and various ANN models.

2 EXPERIMENTAL CONFIGURATION

Regarding accurately predicting car sales prices, the existing methods often rely on traditional statistical approaches like linear regression, which has been seen as an acceptable method due to a significant linear relationship being found. The problem at hand involves utilizing Artificial Neural Networks (ANNs), a powerful tool in machine learning, to predict car sales prices to further improve performance. By exploring different ANN architectures and comparing them with a baseline linear regression model, we aim to identify the most effective approach for predicting car sales prices accurately.

2.1 Linear Regression

In this predictive modeling task, we aim to utilize linear regression as the baseline method to estimate the car purchase amount based on a set of five input factors. Except for gender which is a binary factor, the other input factors, after MinMax normalization, include the individual's age , annual salary, credit card debt, and net worth, all normalized to fall within the range of 0 to 1.

Linear regression is a statistical method that assumes a linear relationship between the input features and the target variable. In our case, the linear regression model can be expressed as:

$$y_{\text{price}} = \beta_0 + \beta_1 \cdot X_{\text{gender}} + \beta_2 \cdot X_{\text{age}} + \beta_3 \cdot X_{\text{salary}} + \beta_4 \cdot X_{\text{debt}} + \beta_5 \cdot X_{\text{net\_worth}} + \epsilon$$

where:

- y_{price} is the predicted car price.
- β_0 is the intercept term.
- $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$ are the coefficients associated with each factor.
- $X_{\text{gender}}, X_{\text{age}}, X_{\text{salary}}, X_{\text{debt}}, X_{\text{net\_worth}}$ are input factors.

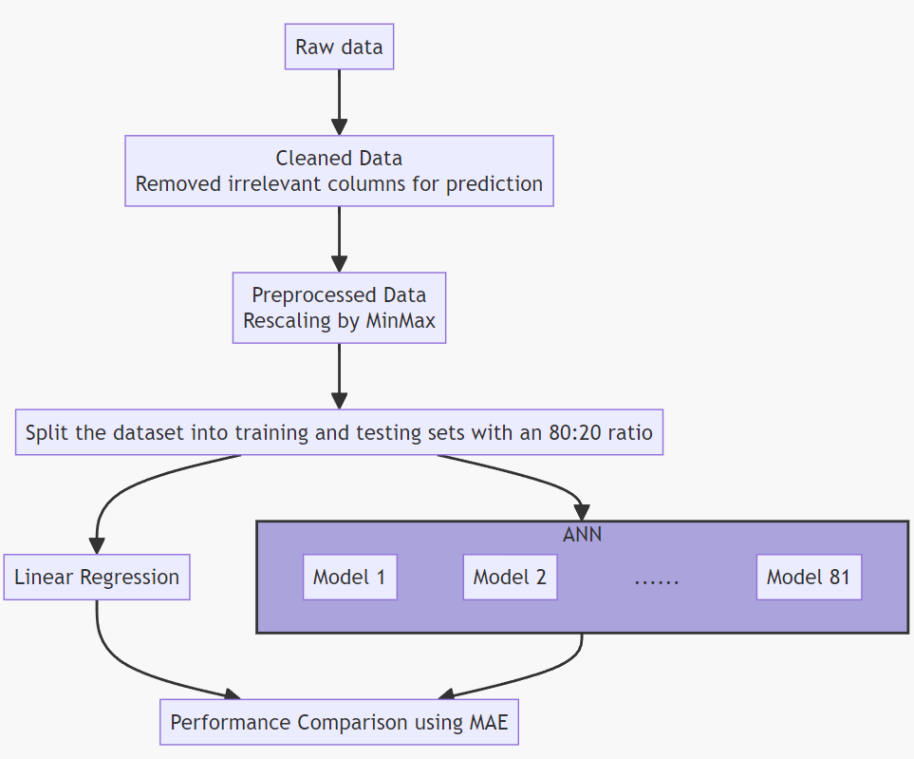

Fig. 1. Project Overview.

- ϵ represents the error term, accounting for unobserved factors.

The coefficients $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5$ are estimated during the training phase, aiming to minimize the difference between the predicted car price and the actual values in y_{price}. Once trained, the linear regression model can be used to make predictions on new data, providing insights into the expected car price based on the given input factors.

2.2 Artificial Neuron Network

In this project, we systematically investigate the performance of different ANN structures and optimization methods to identify the optimal configuration for predicting our target variable. We consider a range of activation functions, including *relu*,*sigmoid*, and *linear* and vary the number of neurons in each layer with choices of 5, 10, and 15 neurons. Additionally, we explore different optimizers, such as *adam*,*sgd*, and *nadam*. We generate ANN models based on the combinations of activation functions, neuron counts, and optimizers. All models are compiled using the mean absolute error loss function and mean absolute error as the evaluation metric. For each model we created, we train it on a training set for 100 epochs with a validation split of 20%.

2.2.1 *Optimizer.*

Adam (Adaptive Moment Estimation): Adam maintains two moving averages,m_t(mean) and v_t(uncentered variance), of the gradients g_t. The hyperparameters β_1 and β_2 control the decay rates of these moving averages. The $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected estimates. The learning rate η is adaptive for each parameter [3].

Here is how it update the value:

$$
\begin{aligned}
m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \\
v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \\
\hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \\
\theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t
\end{aligned}
$$

Stochastic Gradient Descent (SGD): As a classic optimization algorithm, it is a computationally efficient option. SGD updates the parameters θ in the direction of the negative gradient g_t with a fixed learning rate η.

Here is how SGD update the value:

$$\theta_{t+1} = \theta_t - \eta \cdot g_t$$

Nadam (Nesterov-accelerated Adaptive Moment Estimation): Nadam incorporates Nesterov momentum into Adam. It combines the advantages of Adam and Nesterov accelerated gradient (NAG) by using the NAG update in the final step [1].

Here is how Nadam update the value:

$$
\begin{aligned}
m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \\
v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \\
\hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \\
\theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot (\beta_1 \cdot \hat{m}_t + (1 - \beta_1) \cdot g_t)
\end{aligned}
$$

2.2.2 *activation function.*

Here, we briefly summary activation functions existing in this experiment:

Activation Function	**Mathematical Expression**
ReLU (Rectified Linear Unit)	$f(x) = \max(0, x)$
Sigmoid (Logistic Function)	$f(x) = \frac{1}{1+e^{-x}}$
Linear Activation	$f(x) = x$

The experiment aims to analyze and compare the performance of these different configurations, helping us identify the most suitable ANN architecture and optimization strategy for our predictive modeling task.

3 PRELIMINARY RESULTS

The code for this project can be found at project repository on GitHub [2].

All the output results have been illustrated by Fig. 2. Although linear regression model has relatively low MAE on both training and testing data, we still identify a few ANN models outperforming our baseline. The models with the lowest Test MAE values (indicating better performance on unseen data) are (linear, 5, 15, adam) and (linear, 5, 10, adam), with Test MAE values of 0.000908 and 0.001653, respectively. Models with *adam* as the optimizer generally perform better compared to other optimizers, especially *sgd*. Also, models with a linear activation function tend to perform better than other activation functions. Increasing the number of neurons in the layers does not always result in better performance. Some models with fewer neurons outperform those with more neurons. The worst-performing models are those using *sigmoid* function with *sgd* optimizor.

All the output results have been illustrated by Fig. 2. Although the linear regression model has a relatively low MAE on both training and testing data, we still identify a few ANN models outperforming our baseline. The models with the lowest Test MAE values (indicating better performance on unseen data) are **(linear, 5, 15, adam)** and **(linear, 5, 10, adam)**, with Test MAE values of 0.000908 and 0.001653, respectively. Models with *adam* as the optimizer generally perform better compared to other optimizers, especially *sgd*. Also, models with a linear activation function tend to perform better than other activation functions. Increasing the number of neurons in the layers does not always result in better performance. Some models with fewer neurons outperform those with more neurons. The worst-performing models are those using the *sigmoid* function with *sgd* optimizer.

We then conduct statistical tests for the best ANN model, **(linear, 5, 15, adam)**,

Metric	**Value**
Size of Testing Data	100
Variance of Predicted Samples	0.0239
Variance of Actual Values	0.0242
Standard Deviation of Predicted Samples	0.1545
Ratio for Checking Nearly Equal Variance	0.9867
T-Statistics	0.0142
P Value	0.9887

Table 1. Statistical Test Results

Table 1 summarizes the results of a two-tailed test comparing the variances between the predicted samples and the actual values in the testing dataset. The size of the testing data is 100. The variance of the predicted samples is found to be 0.0239, while the variance of the actual values is 0.0242. The standard deviation of the predicted samples is 0.1545. The ratio for checking nearly equal variance is 0.9867, suggesting a close match between the variances. The calculated t-statistics is 0.0142, and the associated p-value is 0.9887. With a p-value greater than the commonly used significance level of 0.05, we do not have sufficient evidence to reject the null hypothesis. Therefore, we conclude that there is no significant difference in variances between the predicted samples and the actual values; the prediction model we proposed can perform a satisfactory output.

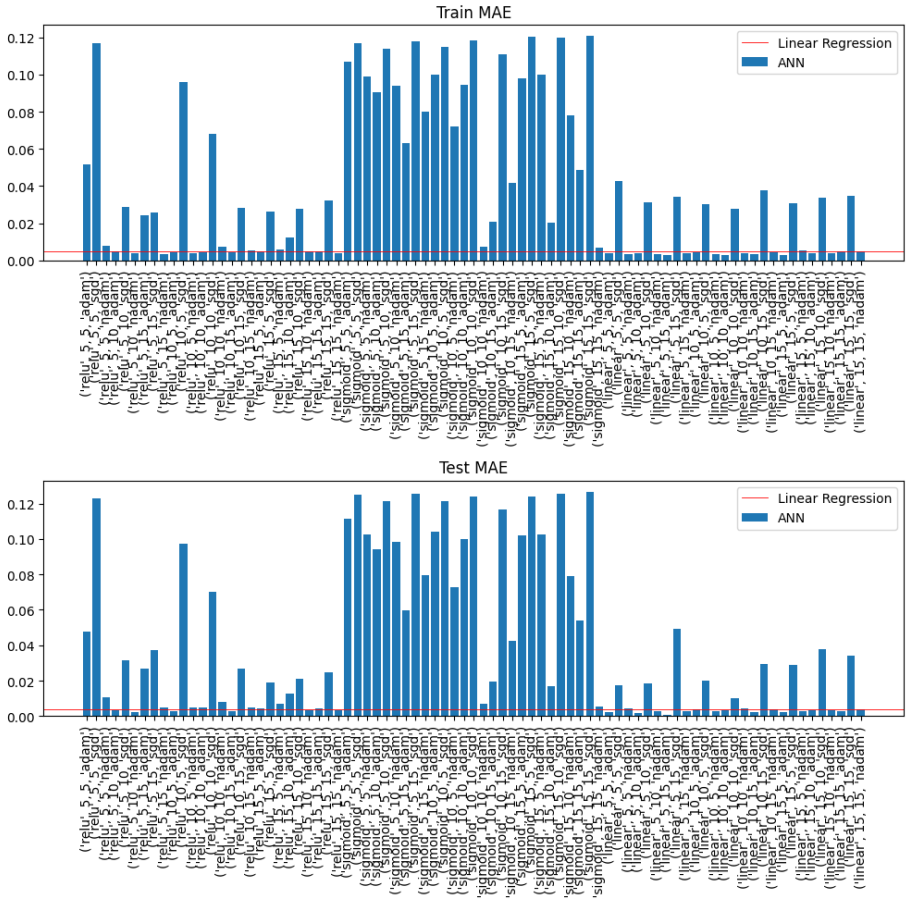

Fig. 2. Output Overview

REFERENCES

[1] Timothy Dozat. 2016. Incorporating nesterov momentum into adam. (2016).

[2] Liam Hsieh. 2023. Source code for final project. https://github.com/liam-hsieh/CS235/blob/main/project/code/car_price_prediction.ipynb.

[3] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).