# 

**Assessment 2**

**Programming Task**

**Intelligent Systems**

ISY503

**Learning Facilitator: Dr. Farshid Keivanian**

Student name: Nguyen Bui

# Student ID: A00129444

# Table of Contents

[**Table of Contents 1**](#_x4cmmis3a40x)

[**1. Introduction 2**](#_somxyvwv5b03)

[**2. Models and Techniques Used 2**](#_uutzz7f3l233)

[**3. Impact on Data and Performance 3**](#_mdzpk3u10ke9)

[3.1. Model 1: Best Model with Numeric Features (No Normalization) 3](#_qsl38qw7h1jh)

[3.2. Model 2: Adding Normalization to Numeric Features 4](#_y0c9jpbsq19l)

[3.3. Model 3: Using Only Categorical Features 5](#_29ij3aq054ik)

[3.4. Model 4: Best Model Using All Features (Numeric and Categorical) 6](#_o2tgxy80m0r9)

[**4. Evaluation of Efficiency 6**](#_b4xjzlbgv6oh)

[**5. Conclusion (50 words) 7**](#_36phe10vxxz)

[**6. References 7**](#_k8wi6gsraj15)

# 

# 1. Introduction

In machine learning, selecting the right model and hyperparameters is crucial for optimizing predictions. This project focuses on predicting car prices using various models, including the DNN Regressor and DNNLinearCombinedRegressor. We explore how hyperparameter tuning, feature engineering, and model selection impact prediction accuracy and overall performance.

# 2. Models and Techniques Used

Various models and hyperparameters were experimented with to optimize car price prediction. The standard model used was a DNN Regressor with 64 hidden units and a batch size of 16. Different configurations and approaches were tested to improve upon this baseline:

* **Learning Rate:**

Values ranging from 0.1 to 0.0005 were tested. A learning rate of 0.005 provided the best balance between training speed and model accuracy, offering stable convergence without overfitting.

* **Batch Size:**

Batch sizes of 10, 16, and 32 were tried. The batch size of 16 consistently provided the best results, balancing model stability and computational efficiency.

* **Hidden Layers:**
  + 128, 64 units: This configuration was effective for handling non-normalized data, allowing the model to capture complex feature relationships and varying scales.
  + 128, 64, 32 units: This deeper setup offered more flexibility in modeling intricate patterns, leading to improved accuracy without overfitting.
* **Training Steps:**

A range from 10,000 to 30,000 steps was experimented with. While 20,000 steps yielded solid results, extending to 30,000 steps led to further performance improvements, especially when training on non-normalized data.

* **Model Selection:**

To enhance performance, the DNN Regressor was replaced with the DNNLinearCombinedRegressor (Maplesoft, n.d.). This combined model uses a DNN for numeric features and a linear model for categorical features, allowing it to capture both non-linear and linear relationships in the data. This hybrid approach was especially effective for a dataset with mixed feature types and scales, improving both generalization and prediction accuracy.

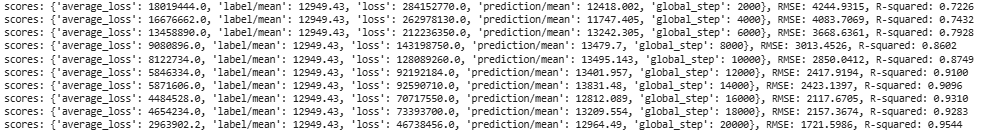
* **Feature Transformations:**

Normalization can improve model quality, but its impact depends on the features and how it's applied. Initially, using Z-score normalization on all features increased the loss, likely due to inappropriate scaling of certain features. After selectively applying normalization to specific features, the loss decreased (Google Developers, n.d.). Retuning hyperparameters after normalization is crucial, as it helps the model adapt to the transformed data for better performance. Various feature transformation techniques were applied to numeric variables:

* + Z-score normalization for features like width, height, weight, length, and highway-mpg.
  + Min-Max scaling for symboling and wheel-base.
  + Robust scaling (using median and IQR) for city-mpg, stroke, normalized-losses, engine-size, and horsepower.
  + Log transformation for skewed features such as peak-rpm and compression-ratio.

# 3. Impact on Data and Performance

## **3.1. Model 1: Best Model with Numeric Features (No Normalization)**

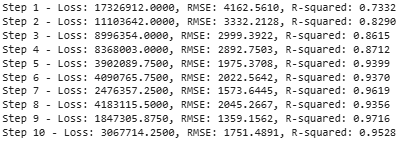
****

* **Loss:** 46,738,456
* **RMSE:** 1721.5986
* **R-Squared:** 0.9544

This model used only numeric features and adjust hyperparameter without applying any normalization.Despite achieving a strong R² score, the relatively high RMSE indicates that there was still a significant prediction error. The model likely struggled due to the wide range of scales in the numeric features, which can distort learning.

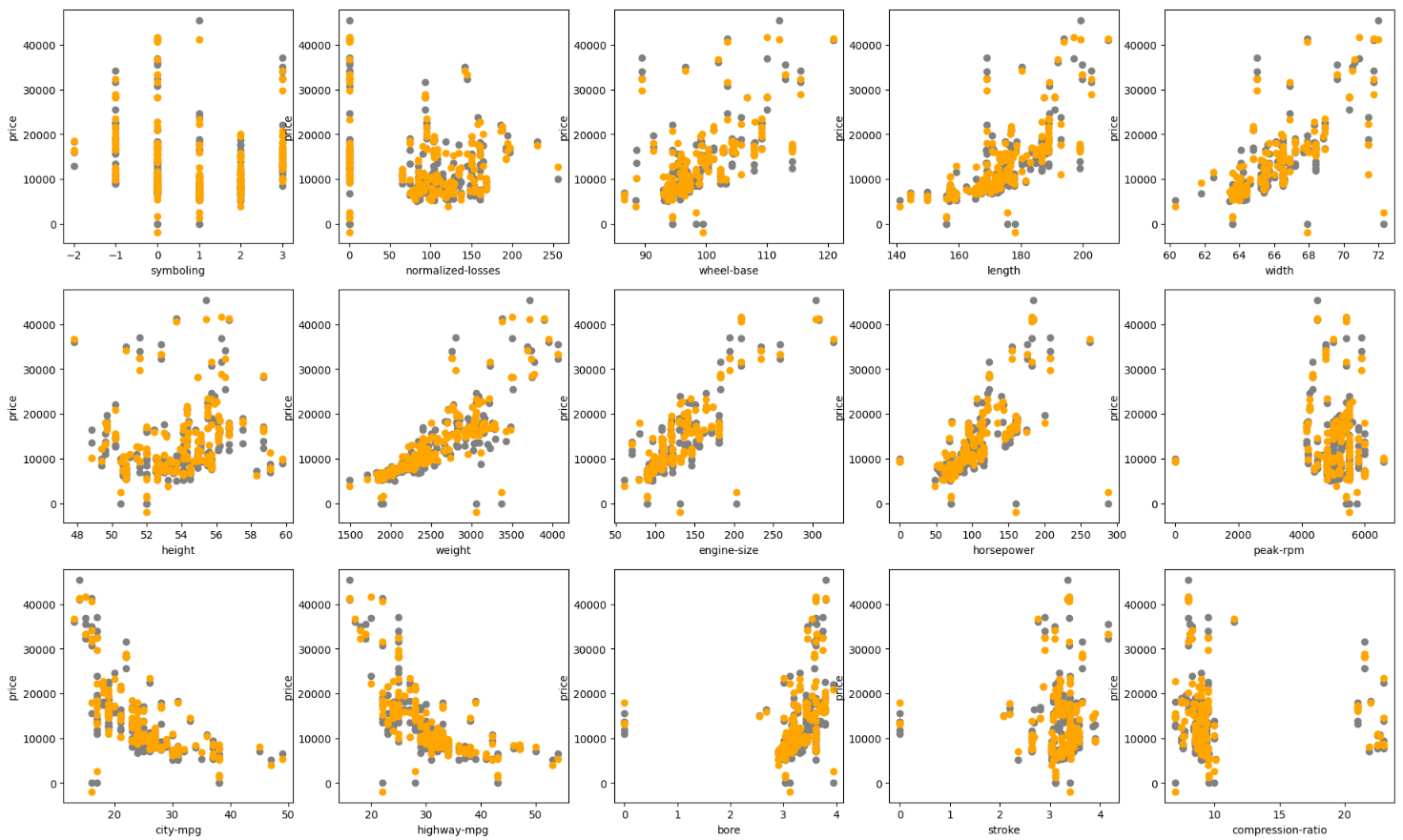
## 

## 3.2. Model 2: Adding Normalization to Numeric Features

****

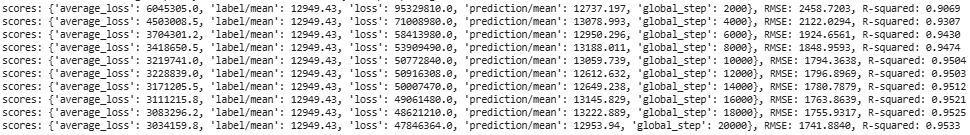
* **Loss:** 3,067,714
* **RMSE:** 1751.4891
* **R-Squared:** 0.9528

In this model, normalization techniques such as **Z-score**, **Min-Max**, and **Robust scaling** were applied to numeric features. While the RMSE slightly increased, the **overall loss decreased significantly**, and the model became more stable. This suggests that normalization improved generalization and helped avoid overfitting or biased weight updates.



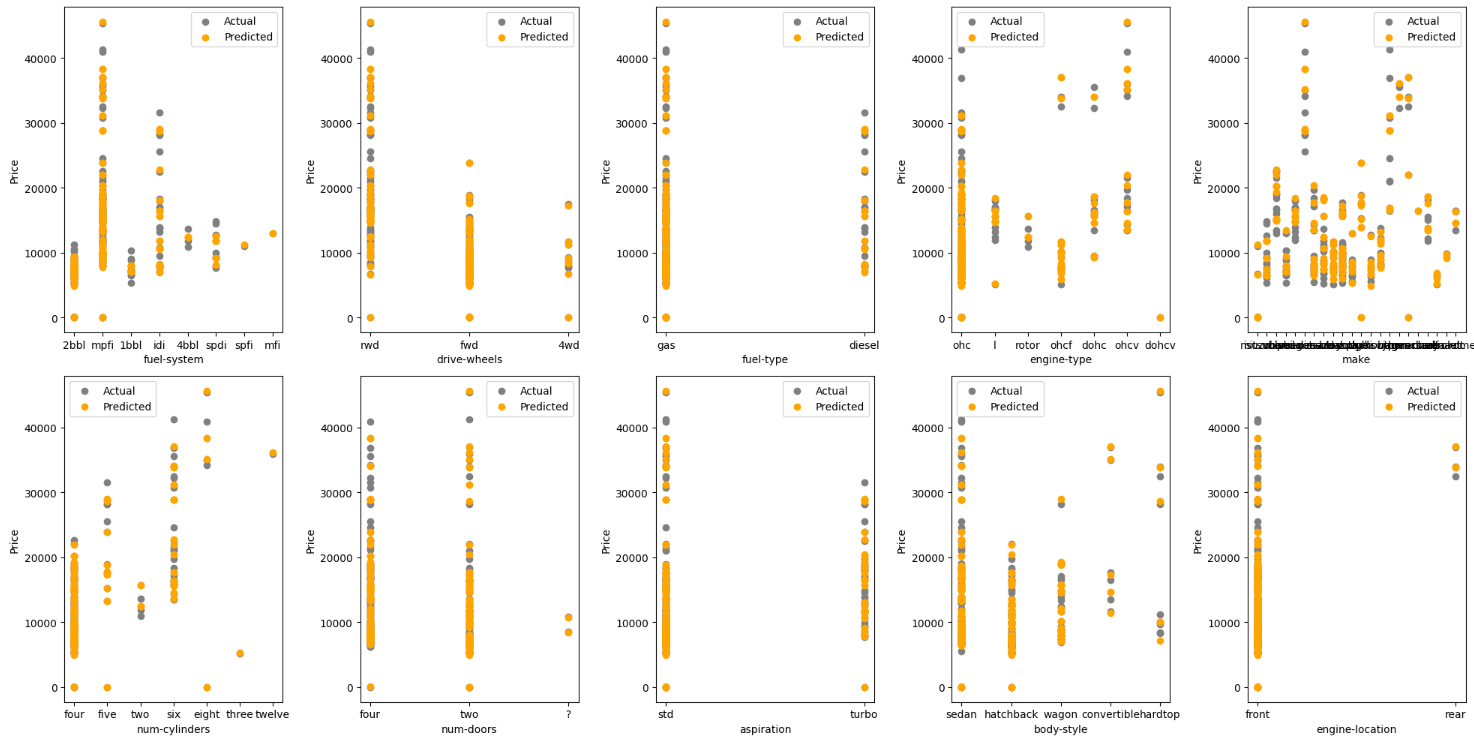
## 

## 3.3. Model 3: Using Only Categorical Features

****

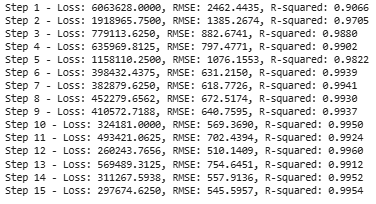
* **Loss:** 47,846,364
* **RMSE:** 1741.8849
* **R-Squared:** 0.9553

This model was trained using only categorical features, represented through indicator columns. It performed reasonably well, achieving a high R² value, which means categorical features contain meaningful information. However, since it lacked numeric inputs, it couldn't capture all the variance in the data, resulting in higher loss and RMSE compared to the best model.



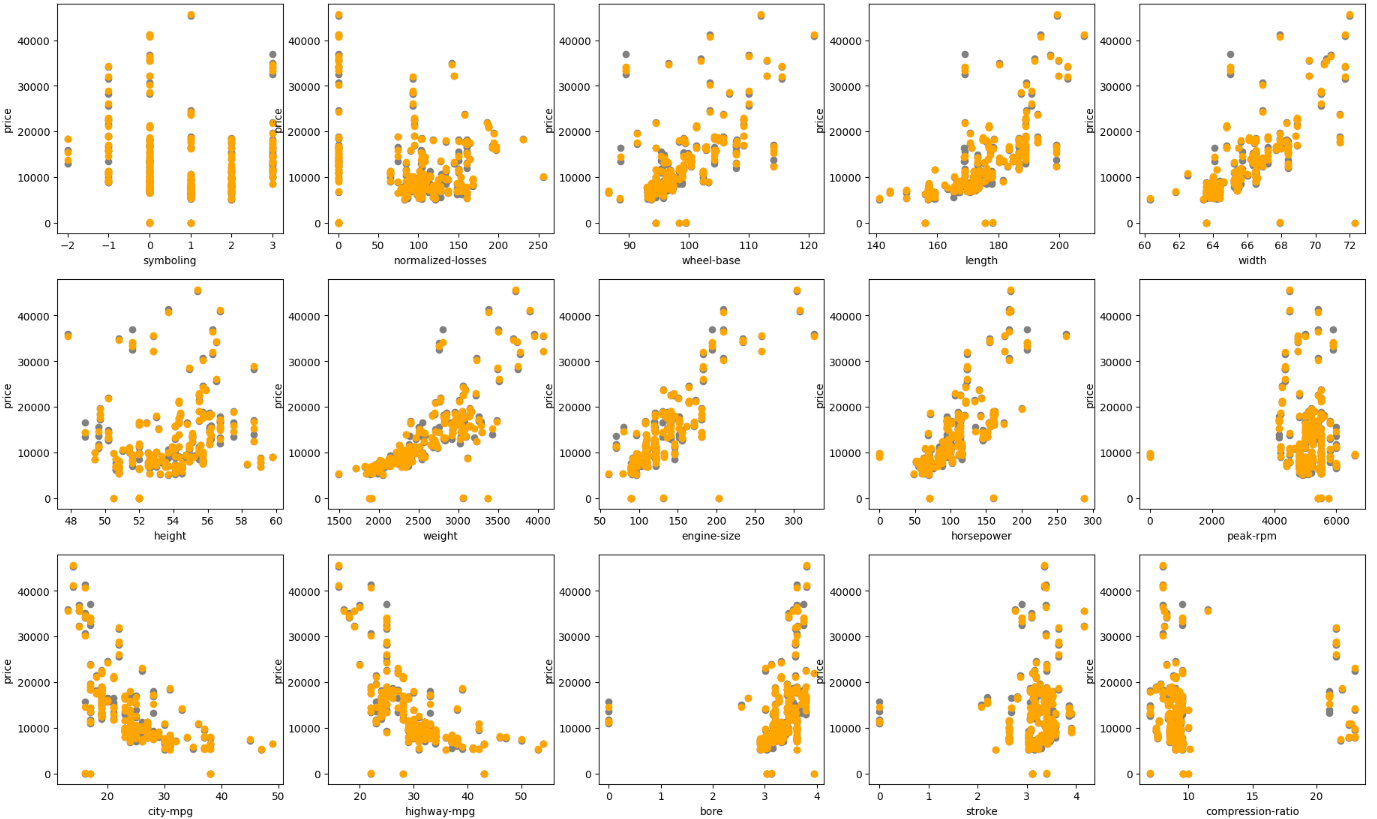
## 

## 3.4. Model 4: Best Model Using All Features (Numeric and Categorical)

****

* **Loss:** 297,674
* **RMSE:** 545.5957
* **R-Squared:** 0.9954

This model combined normalized numeric features and categorical features using a DNNLinearCombinedRegressor. It delivered the best performance across all metrics. The low RMSE and near-perfect R² indicate that the model captured complex and linear relationships in the data very effectively. The combined use of feature engineering and hybrid modeling significantly boosted accuracy.



# 4. Evaluation of Efficiency

The best-performing model was the DNNLinearCombinedRegressor using both normalized numeric and encoded categorical features. It achieved the lowest RMSE (545.60) and highest R² (0.9954), indicating excellent predictive accuracy. The most impactful hyperparameters were the learning rate (0.005) and hidden layers (128, 64, 32), which enabled deep learning while preventing overfitting. Normalization and log transformation also significantly boosted performance. A key challenge was balancing training time and accuracy, especially when tuning learning rates and model depth. Additionally, handling skewed distributions required experimentation to ensure stable convergence and avoid distorted predictions.

# 5. Conclusion

This project highlighted the importance of combining feature engineering and model selection to achieve high accuracy. The DNNLinearCombinedRegressor with normalized numeric features and categorical encoding performed best. Future work could explore additional feature transformations, model architectures, and optimization techniques to further improve prediction performance and reduce errors.

# 6. References

1. Google Developers. (n.d.). *Numerical data: Normalization*. Google for Developers. https://developers.google.com/machine-learning/crash-course/numerical-data/normalization

2. Maplesoft. (n.d.). *DNNRegressor*. https://www.maplesoft.com/support/help/maple/view.aspx?path=DeepLearning%2FDNNRegressor