Used Car Selling Price Prediction Report

Artificial Neural Networks and Deep Learning A

John Iwenofu

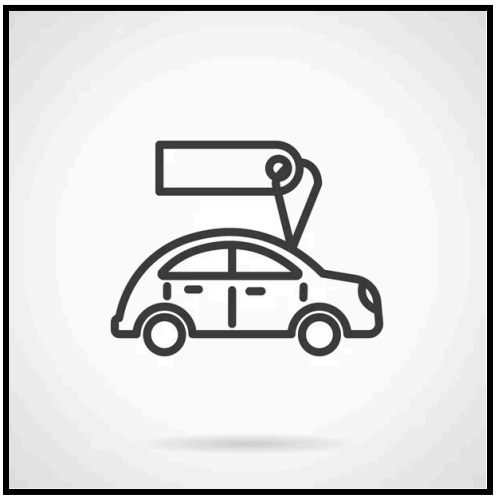
Beste Kuruefe

Abdi Kusata

Will Landers

Link to the GitHub Repository:

https://github.com/jiwenofu/Neural-Network-Project/tree/main



Predicting Used Car Prices Using Machine Learning

Have you ever wondered how much your car is worth? Or how dealerships determine the price of a used car? In this project, we seek out the answers to these questions using data and machine learning.

The goal was to build a model that could predict the price of a used car based on its features: model year, mileage, brand, reported accidents, transmission, exterior color, and fuel type. Along the way, we learned a lot about data cleaning, feature engineering, and the power of artificial intelligence.

This blog post will walk you through our process, the challenges we faced, and the results we achieved. The data and the code can be found on our GitHub repository.

The Dataset

The first step in any machine learning project is finding the right data. For this project, we used a dataset of 4009 used cars scraped from Kaggle. The dataset includes information about each car, such as its price, mileage, year, brand, model, color, transmission type, and engine type. The link for the dataset can be found here: https://www.kaggle.com/datasets/taeefnajib/used-car-price-prediction-dataset

However, raw data is rarely ready for analysis. This dataset was no exception; it had missing values, inconsistent formats, and outliers. Cleaning and preparing the data was a crucial part of the process. After the data cleaning process, we removed rows with missing features and outliers, resulting in a final training dataset with 3569 entries.

Cleaning and Preparing the Data

Processing uncleaned data is like comparing apples to oranges. To make the data useful, we had to:

- **Remove Features We Didn't Need:** We removed the features "internal color" and "clean title" because we saw that those were not necessary in predicting a price.
- **Handling Missing Values:** Rows with missing prices, mileage, etc, were removed. For other columns, we used logical rules to fill in missing values.
- **Standardized Formats:** Columns like "price" and "mileage" were stored as strings with symbols like "$" or "mi." We converted these to clean numerical values.
- **Group Categories:** Features like transmission and exterior color had too many unique values. We grouped similar categories together (e.g., "7-speed automatic" and "6-speed automatic" became "automatic").
- **Remove Outliers:** Cars priced above $150,000 were considered outliers and removed to avoid skewing the model.

Building the Model

With the data ready, it was time to build and test different machine learning models. We started by creating a baseline model to establish a performance benchmark. The goal was to evaluate how well the different approaches could predict car prices based on the features provided.

Baseline Models: Trying Three Approaches

We tested three different models as potential baselines:

- Linear Regression: A simple and interpretable model that assumes a linear relationship between features and the target variable. While it provided a quick solution, it struggled to capture the complex, non-linear relationships in the data.
- Random Forest Regressor: A powerful ensemble method that combines multiple decision trees. This model performed better than Linear Regression, as it could handle non-linear relationships and interactions between features.
- Poisson Regression: A model designed for count data or positively skewed distributions, such as car prices. It provided a balance between simplicity and the ability to handle the skewed nature of the target variable.

After evaluating the performance of these models using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and Mean Absolute Percentage Error (MAPE), we chose to move forward with the Poisson Regression, and we kept only Poisson Regression in our final code. It offered a better fit for the data compared to Linear Regression while being less complex and computationally intensive than Random Forest.

Moving forward with the Benchmark

Having established the benchmark with the Poisson Regression, we proceeded to train our neural network model to see how well it could perform in comparison. The goal was to determine whether the added complexity of a neural network could significantly improve performance over the baseline.

Neural Network Model

We built three versions of a feedforward neural network using TensorFlow and Keras. The general architecture for all models included:

- Input Layer: Matching the number of features in the dataset.
- Hidden Layers: Multiple layers with activation functions (ReLU for the first two trials and Tanh for the third) to capture non-linear relationships.
- Output Layer: A single neuron with no activation function for regression tasks.

All models were compiled using the Adaptive Moment Estimation (Adam) optimizer and Mean Absolute Error (MAE) as the loss function. While we initially considered using Mean Squared Error (MSE), we ultimately selected MAE because our primary objective was to improve real-world prediction accuracy rather than minimize occasional large errors.

To prevent overfitting, we used early stopping, which monitored the validation loss and stopped training when it stopped improving.

Training and Evaluation

The neural network was trained on the dataset for up to 100 epochs, with 20% of the training data used for testing. After training, the model was evaluated on the test set using MAE, MSE and MAPE (Mean Absolute Percentage Error).

Results

The first step was to preprocess the dataset and make it ready to train. When doing the preprocessing, we investigated some vehicle attributes in order to understand the dataset that we are handling. One important attribute is mileage, which plays a huge factor in the price.

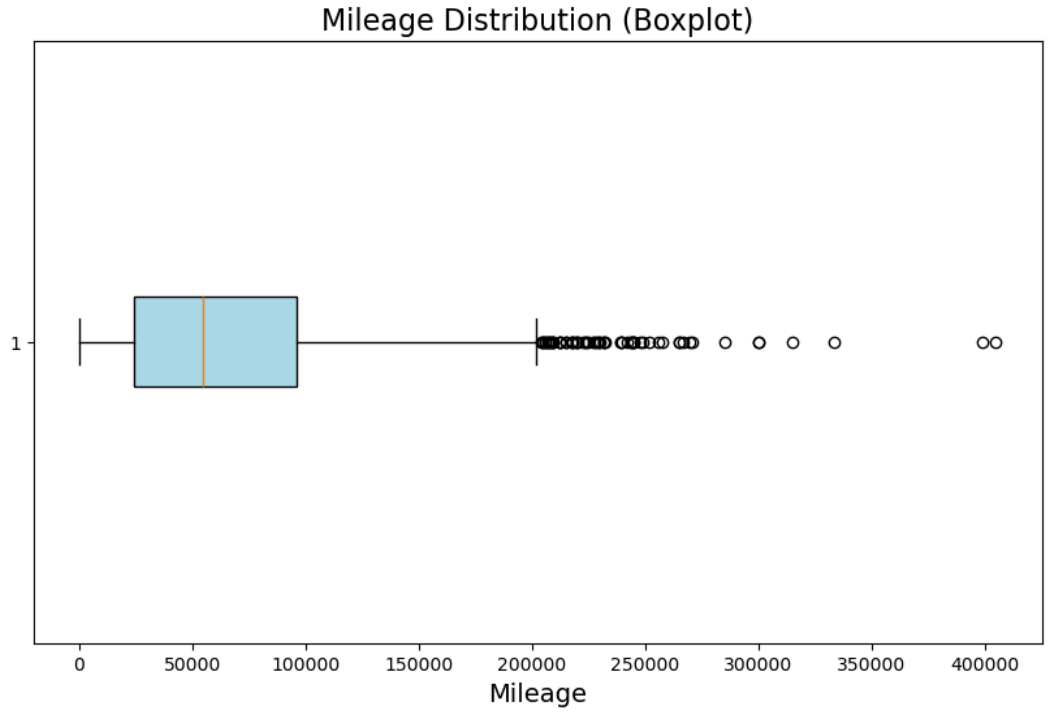


Figure 1. Mileage distribution boxplot, showing the overall mileage range in the dataset.

In Figure 1, the median mileage for the cars was 55,000 miles, with 50% of the cars being within the 25,000 - 100,000 miles range. The outline is right-skewed, meaning that more cars have a high mileage range between 100,000 to 150,000 miles, with some cars having a higher mileage that is above the range. Figure 2 shows the high frequency of cars within the 100,000 - 150,000

mile range, while showing the medium frequency of cars within both the low and medium mileage range.

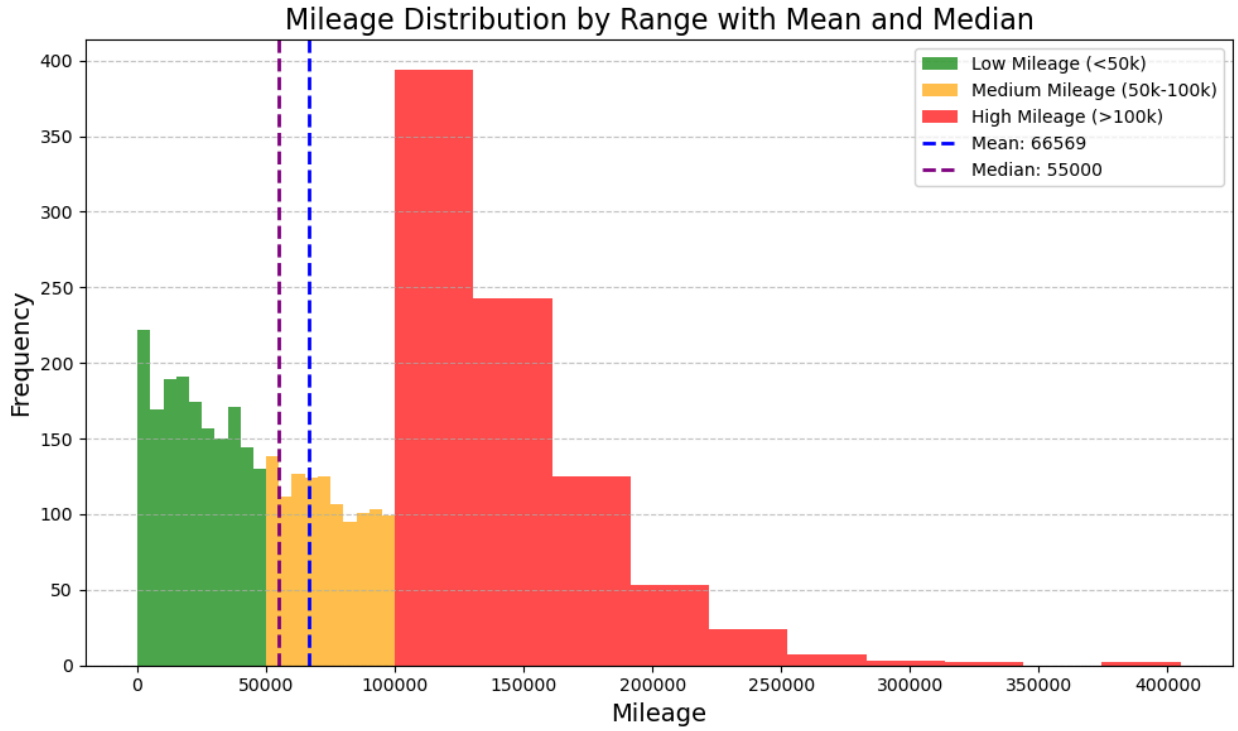


Figure 2. Bar chart of mileage distribution, segmented by low, medium, and high mileage.

From our model, we ran three trials. On the first trial, there are two hidden neuron layers with one output layer. For the second trial, there are four hidden neuron layers with one output layer. For the third trial, we used a different activation than the first two trials by using tanh. However, it has the same amount of hidden layers as the second trial. The codes for the three trials are shown in Figures 3, 4, and 5, respectively.

```
# Build the feedforward neural network
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)  # Output layer for regression (no activation or linear)
])
```

Figure 3. Neural network code for the first trial, which includes two hidden layers and ReLU activation.

After running the model for the first trial, we got the mean absolute error of $11,124, constituting a 38.9% error. This price represents the average deviation of how far off our model is estimated in comparison with the actual price.

```
model2 = tf.keras.Sequential([
    tf.keras.layers.Dense(256, activation='relu', input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])
```

Figure 4. Second trial neural network code, training it with four hidden layers.

When running the second trial with the four hidden layers as shown in Figure 4, our model got a better result than in the first trial. The second trial constituted $9,494 for the mean absolute error, representing a 31.6% error.

```
model2_tanh = tf.keras.Sequential([
    tf.keras.layers.Dense(256, activation='tanh', input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dense(128, activation='tanh'),
    tf.keras.layers.Dense(64, activation='tanh'),
    tf.keras.layers.Dense(32, activation='tanh'),
    tf.keras.layers.Dense(1)  # Output layer for regression
])
```

Figure 5. Neural network code for the third trial.

The third trial constitutes a different activation with the same hidden layers as the second trial, as shown in Figure 5. That trial performed the worst out of the two previous trials, with our tanh model's mean absolute error bringing in $34,441. This led to a percentage error of 98.7%, which is very high compared to 38.9% and 31.6% from the two respective trials. Figure 6 shows the comparison of the three trials, detailing their MAE, MSE, and MAPE.

```
Comparison of Model Trials:
            Model    MAE (in $)           MSE   MAPE (%)
0  Trial 1 (ReLU)  11124.348603  3.043538e+08  38.867140
1  Trial 2 (ReLU)   9494.312073  2.086249e+08  31.605176
2  Trial 3 (Tanh)  34441.478631  1.811847e+09  98.726767
```

Figure 6. Comparison of the three trials, showing Trial 2 performing better overall.

```
# Initialize the Poisson Regressor
poisson_model = PoissonRegressor(alpha=1e-12, max_iter=1000)

# Train the model
poisson_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred_poisson = poisson_model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred_poisson)
mse = mean_squared_error(y_test, y_pred_poisson)
mape = np.mean(np.abs((y_test - y_pred_poisson) / y_test)) * 100  # Mean Absolute Percentage Error
```

Figure 7. Poisson Regression baseline model setup code.

For the baseline model in Figure 7, we used the Poisson Regression to go against our model. Training the dataset with the baseline model, we got a mean absolute error of $9,959, constituting a 36.8% error. The baseline model performed worse than our second model trial, considering that the percentage error for that trial was 31.6%. This comparison is shown in detail in Figure 8.

```
Comparison of Best Trial and Baseline Model:
                Model  MAE (in $)           MSE  MAPE (%)
0  Poisson Regressor     9959.45  2.097562e+08     36.80
1        Trial 2 ANN     9494.31  2.086249e+08     31.61
```

Figure 8. Comparison between the baseline model and the Trial 2 model.

Discussion

As talked about in previous sections, we used a baseline model along with three trials with the ANNs:

- **Baseline Poisson Regression**
    - The Poisson Regression served as a starting point, achieving a Mean Absolute Error (MAE) of approximately $9,959. While it provided a reasonable benchmark, it struggled to capture the non-linear relationships in the data, which are crucial for predicting car prices accurately.
- **Trial 1 ANN**
    - The first ANN, which had a simple architecture with 64 and 32 neurons in its two hidden layers, produced a mean absolute error (MAE) of approximately $11,124—worse than the baseline model. This suggests that while a neural network can potentially yield good predictions, this particular architecture was still limited in its capacity.
- **Trial 2 ANN**
    - By increasing the number of hidden layers and neurons (256, 128, 64, and 32 neurons), the second ANN reduced the MAE to $9,494 and outperformed the baseline Poisson Regression model. This improvement highlighted the benefits of a deeper network in capturing intricate relationships between features and price.
- **Trial 3 ANN (with tanh activation)**
    - Replacing the ReLU activation function with tanh in the same architecture as Trial 2 resulted in worse performance, with an MAE of $34,441. This suggests that ReLU was better suited for this dataset, likely due to its ability to handle sparse activations and avoid vanishing gradient issues.

Initially, our model did not perform well the first time we tested it. There was a slight gap between the MAE and MAPE for the Poisson Regression baseline model and the first trial. Although it was not a big gap (2% difference), we noted that it could be improved by adding more hidden layers. For the second trial, we added two more hidden layers and increased the number of neurons to accommodate it. As a result, our second model performed not only better than our first model, but it also performed better than the baseline model. This meant that the car

prediction was more accurate with our second model than the Poisson Regression model, with a less percentage error overall.

Noting the success with the second model, we wanted to investigate the performance of a different activation by creating a new model while using the same hidden layers from the second model. Using the tanh activation, the third model performed worse than our first two models and the baseline model. Therefore, it was concluded that the tanh activation was not sufficient to accurately handle and train the used car dataset, citing the error percentage of nearly 100% for that model.

We encountered several challenges during development. A major one was **dealing with missing or inconsistent data** in our dataset. Another difficulty was **handling categorical data**, especially variables like car brands. There were thousands of unique individual brands that made it near impossible to group them categorically.

In the future, we plan to develop a user friendly interface that allows users to easily input vehicle information and customize the model's parameters. This would make the tool more accessible and intuitive for everyday users, not just developers.

Conclusion

Predicting used car prices is a complex but rewarding challenge that blends data science, machine learning, and real-world economics. Through this project, we developed and evaluated several models from a Poisson Regression baseline to multiple trials of artificial neural networks (ANNs). Among these, our second neural network trial performed the best, achieving the lowest mean absolute error and outperforming the benchmark model. This result confirmed that deeper networks with more layers and ReLU activation are better suited to capturing the non-linear relationships between car features and their market value.

Along the way, we tackled real-world challenges such as messy data, categorical overload, and inconsistent formatting. These issues are common in data science projects, and resolving them

was essential to achieving meaningful results. Grouping similar categories and removing outliers allowed us to simplify the problem while keeping the core predictive power of the dataset.

While our model is not yet production ready, it opens the door for further development. Future improvements could include more refined feature engineering, expanding the dataset, and integrating external data sources like regional pricing trends or dealership listings. Most importantly, we aim to build a user friendly interface so that individuals can easily estimate their vehicle's worth without needing technical knowledge.

In short, our work demonstrates the power of machine learning in everyday decisions. With cleaner data, smarter models, and better accessibility, predictive tools like ours can make the used car market more transparent and efficient for everyone.

Sources

D'Allegro, Joe. "Just What Factors into the Value of Your Used Car?" *Investopedia,* Investopedia, www.investopedia.com/articles/investing/090314/just-what-factors-value-your-used-car.asp. Accessed 5 May 2025.

Frost, Jim. "Poisson Regression Analysis Overview with Example." *Statistics by Jim*, 21 Sept. 2024, statisticsbyjim.com/regression/poisson-regression-analysis/.

KumarI, Ajitesh. "MSE vs RMSE vs Mae vs MAPE vs R-Squared: When to Use?" *Analytics Yogi*, 18 Aug. 2024, vitalflux.com/mse-vs-rmse-vs-mae-vs-mape-vs-r-squared-when-to-use/.

Najib, Taeef. "Used Car Price Prediction Dataset." *Kaggle*, 15 Sept. 2023, www.kaggle.com/datasets/taeefnajib/used-car-price-prediction-dataset.

"Tanh vs. Sigmoid vs. Relu." *GeeksforGeeks*, GeeksforGeeks, 3 Oct. 2024, www.geeksforgeeks.org/tanh-vs-sigmoid-vs-relu/.