COMS 4995 Applied Machine Learning

Group 4 - Project Report

Anqi Xia (ax2171), Azam Khan (ak4973), Uttam Gurram (ug2146)
Xiaokun Yu (xy2550), Yupei Shu (ys3597)

1. Introduction

For an individual looking to purchase a car within budget, the expanding market of pre-owned cars is always a great place to look. So having a model that considers multiple car features to estimate the price point is valuable to buyers while also providing sellers with a competitive price point that can attract potential buyers. Hence, our main objective is to develop machine learning models that can accurately predict the price of a car, specifically through constructing a regression model. We will carefully consider which car features are most relevant for ensuring accurate and dependable price predictions.

2. Dataset (US Used cars dataset)

The US Used Cars dataset is vast, containing roughly **3 million** records about pre-owned cars listed since 2016. However, the valuation of a car depreciates over time, and at the same time, the actual listed price appreciates a bit because of inflation. So our model becomes unreliable if trained using the data from all the years, and we will limit our analysis to only the data of roughly **176K** records from 2021. The dataset contains 65 features, including numerical (e.g., $back\_legroom$, $engine\_displacement$, $daysonmarket$), categorical (e.g., $body\_type$, $city$, $engine\_cylinders$), boolean (e.g., $franchise\_dealer$, is_new) and textual description variables, providing a diverse range of information to utilize.

The target variable is **price** to be predicted.

3. Preliminary Data handling

3.1 Cleaning the dataset

Firstly, we processed the numerical features to remove information such as metrics (e.g., in, gal).

3.2 Feature extraction

We converted the $major\_options$ feature, initially representing a list of major options, to a numerical value representing the number of major options.

3.3 Feature selection

Based on our understanding, we discarded 18 features, 1 more compared to our previous understanding in **project part-2**, that are either repetitive or provide irrelevant information, leaving us with 47 meaningful features. Out of the remaining 47 features, we further removed 19 features that had more than 30% values missing.

4. Data Exploration with Insights

The **histogram** plots of numerical features reveal that some numerical features have skewed distributions and require transformations. The **side-by-side** bar plots of the categorical features show the *imbalance* in different feature values. The **box** plots of numerical variables reveal that the feature *mileage* has extreme outliers which need to be eliminated to improve its importance.

The distribution of the target variable **price** shows us that the variable is highly skewed because of *outliers*, so we apply *log* transformation to center the skewed distribution (**Note:** different from our analysis in **project part-2**, we decided to retain the *outliers* in **price** to allow the models to handle those outliers robustly). The **scatter** plots reveal the *complex* relationship between various numerical features and **price**. Finally, the **box** plots of various categorical features roughly reveal the impact of different feature values on the **price**.

5. Data Processing

5.1 Filling missing values

We utilized simple imputing methods to fill in the missing values. We imputed the missing values for the categorical features with the *mode* of the feature. Similarly, for the numerical features, we imputed the missing values with the *median* of the feature.

5.2 Outlier handling

As mentioned above, we observed that the *mileage* feature has extreme outliers, so we utilized *Inter Quartile Range* to detect (about 2% of the entire records) and remove those outliers. After removing outliers, the *savings_amount* feature has just 1 unique value, so we discard that feature.

5.3 Feature encoding

We utilized **boolean** encoding for the *feanchise_dealer* and *is_new* features. *body_type*, *wheel_system_display* features were encoded using **one hot** encoding. *engine_cylinders*, *fuel_type* features are encoded using **ordinal** encoding, and importance is assigned based on the usually observed sale prices of the cars with those different feature values. Initially, we used **target** encoding for all the remaining features. However, we observed that two features, *trim_name* and *model_name*, exhibit a high correlation with the target variable **price** which is not desired, so we utilized a **binary (base 2)** encoder for these two features.

5.4 Correlation matrix and Data Scaling

After feature encoding, we had 55 features and utilized the *correlation* matrix to identify highly (>= 0.9) correlated features. We identified 2 such pairs and eliminated 2 features, one from each pair. Finally, we utilized a **Standard** scaler to transform the features and reduce their skewness.

6. Implemented ML techniques

6.0 Metrics

We compute and report **R2-score**, several error metrics such as **RMSE, MAE**, and **RMSLE** on the **price** in the test dataset. Additionally, we compute the **AUROC** score such that the score reflects the model's ability to rank *bigger* values higher than *smaller* values. As mentioned above, the models are trained with $log(price)$ as the target values. The default and best results (after *hyperparameter* tuning) for each of the models are listed below.

Model	Test dataset metrics									
	default model metrics					*best* model metrics				
	R2-score	**RMSE**	**MAE**	**RMSLE**	**AUROC**	**R2-score**	**RMSE**	**MAE**	**RMSLE**	**AUROC**
Linear Regression	0.8085	14435.31	5278.08	0.1759	0.8596	-	-	-	-	-
Ridge	0.8085	14435.43	5277.86	0.1759	0.8596	0.8084	14436.84	5275.27	0.1759	0.8596
Elastic-Net	0.0	21084.97	12730.06	0.4019	0.5001	0.8085	14435.29	5278.06	0.1759	0.8596
Decision tree	0.97	5215.90	1468.17	0.0697	0.9573	0.9707	9332.59	1664.67	0.0688	0.9535
Random Forest	0.9806	9273.47	1323.02	0.0559	0.9653	0.9816	9521.17	1305.89	0.0545	0.9659
XGBoost	0.9706	10046.26	1842.57	0.0689	0.9503	0.9779	9042.07	1483.37	0.0597	0.9611
Neural Network	-	-	-	-	-	0.9604	12191.84	2125.88	0.08	0.9446
TabNet	-	-	-	-	-	0.9543	12186.04	2250.23	0.0859	0.9383

Table 1

6.1 Linear Regression

The linear regression model is utilized as a baseline. The **R2-score** on the $log(prices)$ in the test dataset is 0.8085. The distribution of residuals of the $log(prices)$ indicates that the log transformation helps in centering the mean of errors around 0 making the data **homoscedastic**. Additionally, we can observe from the $weight$ coefficients that the model heavily relies on the **one hot** encodings of the $body\_type$ feature and 1 or 2 other features to some extent. This heavy reliance wildly fluctuates the model output to changes or noise introduced in those very few features, thus making it **unreliable**.

6.2 Ridge Regression

We trained the default ridge regression model with $alpha = 1$. We also tuned the hyperparameter $alpha$ by performing a grid search to select the model with the best performance on the **validation** dataset. In both cases, the models showed the same performance (**R2-score** of 0.8085) as the linear regression model. However, by observing the $weight$ coefficients, we can understand that these models consider several features, compared to the linear regression model, to estimate the target **price**, thus making it more robust and **reliable**.

6.3 Elastic-Net Regression

We trained the default Elastic Net regression with $alpha = 0.5$ and $l1\_ratio = 0.5$. However, the resulting model performed the worst, did not learn anything, and pushed all the feature weights to 0. We tuned the hyperparameters $alpha$ and $l1\_ratio$ by performing a grid search to select the model with the best performance on the **validation** dataset. Once again, from the $weight$ coefficients, we observe that the best model performs similar to that of Linear and Ridge Regression models. Although the $L1$ and $L2$ penalties do not improve the computed test scores, they help the model consider more features to estimate the target **price**.

6.4 Decision Tree

We trained the default Decision Tree model, which trains to purity on the **development** dataset. Although training till purity is clearly overfitting, the model achieved an excellent **R2-score** of 0.97 on the test dataset. This is because the test dataset has a similar distribution as our development dataset. Then we searched over the hyperparameters max_depth, max_leaf_nodes, and $max\_features$ to see if we could reduce overfitting and improve the performance on the test dataset. The hyperparameter combination $max\_depth = 45$, $max\_leaf\_nodes = 2100$, and $max\_features = 35$ gave the best performance on the **validation** dataset, and the **R2-score** on the test dataset improved marginally to 0.9707 while other metrics had a slight degrade. However, from the $weight$ coefficients, it is noticeable that the model relies heavily on few features to estimate the target **price** which makes it sensitive to noise in those few features.

6.5 Random Forest

A similar framework was used to investigate the use of a Random Forest regression model to predict car prices using our dataset. We trained a random forest model with a default setting and measured its performance on the **test** set. We then performed a randomized search to select hyperparameters for the random forest model. We chose random values from distributions for max_depth, $max\_features$, and $num\_estimators$. Five unique combinations of hyperparameters are randomly selected from the space and ranked based on the validation **R^2-score**. The best hyperparameters were chosen and used to train the random forest model on the entire development set, the performance was then evaluated on the held-out **test** set, and the results can be seen in Table 1. Looking at the feature importance plot, we see that the three most important features for making a prediction are $horse\_power$, $width$, and $make\_name$. The feature $horse\_power$ is often used as an indicator of the car's performance. Therefore, it is not surprising that this feature is highly important in predicting car prices. A larger car may be more expensive than a smaller one, and the $width$ may be a good proxy for this factor. Lastly, the feature $make\_name$ indicates the brand or manufacturer of the car. Some brands are associated with higher quality or luxury, and this may affect the price of the car. Therefore, it is not surprising that this feature is also highly important in predicting car price. Overall, the feature importance analysis provides valuable insights into which features are most significant in predicting car prices and can help us better understand the factors that affect car **price**.

6.6 Gradient Boosting

We explored several boosting algorithms including AdaBoost, XGBoost, HistGradientBoost, and LightGBM. We performed grid search to find the best hyperparameters for each model and evaluated their performance based on the R^2 score on the test dataset. XGBoost achieved the best performance among all the boosting algorithms.

We first trained a default **XGBoost** regressor on the development dataset and measured its performance on the validation set. The model achieved an **R2-score** of 0.9706 showing similar performance as the default Decision tree model. We then performed a grid search to find the best hyperparameters for the XGBoost model. The set of parameters we experimented on are $learning\_rate$, max_depth and min_child_weight. The best hyperparameters for XGBoost regressor are $learning\_rate$: 0.3, max_depth: 9, min_child_weight: 1.0. The best XGBoost model achieved **R2-score** of 0.9779 on the validation set with the three most important features as $width$, $horse\_power$, and $body\_type\_sedan$.

6.7 Neural Network

For the neural nets, we utilized 4 different models with varying numbers of **layers**, **hidden nodes**, **dropout** probabilities, and **activation** functions. We trained these 4 different models with varying **learning rates** to observe its effect. In all the trainings, we utilized an $Adam$ optimizer, MSE as the loss on the $log(prices)$, and ran the trainings for 15 epochs. As noticeable in the plots, the trainings with a learning rate of 0.005 and 0.01 results in larger updates and accordingly more fluctuations in the training curves compared to the trainings with a learning rate of 0.001 showing that the learning rate of 0.001 is a better choice. Additionally, the models with $Tanh$ activation function consistently show better **validation** performance compared to their counterparts with $ReLU$ activation. The model with the best performance on the **validation** dataset was the 4^{th} model with 5 layers, $256, 128, 64, 16, 1$ hidden nodes, dropout probabilities of $0.3, 0.2, 0.1$ in the first 3 layers, and $Tanh, Leaky\_ReLU$ activations. It achieved an **R2-score** of 0.9446 showing the best performance out of all models.

6.8 TabNet

Finally, we consider the **TabNet** model developed by Google, which brings deep learning to tabular data. The model has $SOTA$ performance on several test scenarios. We utilize the functions from the $official$ implementation for our training and validation. We used an $Adam$ optimizer with a learning rate of $2e-3$, MSE as the loss on the $log(prices)$, and trained the model for 25 epochs. At the end of the training, the weights from the epoch with the best **validation** performance are automatically loaded into the model. Additionally, we can observe from the $importances$ that the model values features like $horsepower$, $franchise\_make$, and $width$ more, mimicking the thinking of a human buyer toward estimating the target **price**. The **TabNet** model performs similarly to the best neural network.

7. Conclusion and Future Work

From the trainings of various models with hyperparameter search, we observed that the $Random\ Forest$ model performed the best on the test dataset with an **R2-score** of 0.9816. The baseline **Linear** Regression model itself decently estimated the target **price** with an **R2-score** of 0.8085, and the **Ridge** and **Elastic-Net** Regression models did not provide quantitative improvements over it. The **Decision Tree** model showed significant improvement over them by attaining an **R2-score** of 0.9707, which is even better than the performance of **Neural Network** and **TabNet** models. Finally, the **XG Boost** model showed further improvements and was only slightly behind the best-performing model with an **R2-score** of 0.9779. All the above information indicates that our models do a great job at predicting the target **price**, but it should be noted that these predictions are only on the dataset from a $single\ year$. In the future, we can consider the dataset over different years and train the models to adjust their predictions by taking the time frame into account, which further improves the usability of our model.