Used Car Price Prediction

P15: Kaksha Mhatre, Janvi Jitendra Phadtare, Saail Ganesh
Department of Computer Science, North Carolina State University
Raleigh, NC, 27695
{kpmhatre,jphadta,sganesh9}@ncsu.edu

1 Background

When you are selling a vehicle online, finding the right price for a vehicle is always a hassle. The goal of the project is to predict the price of used cars and determine whether they can be sold as a whole or just for their parts. Valuation of used cars is tricky as it not only depends on the specifications of the car but also on the current state of the car.

The price of a used car depends on attributes such as odometer reading, fuel type, years of use, wear, etc. The aim of the project is to efficiently and reliably calculate the price. Based on this valuation and the actual price of the car, the system will then decide whether it is better to sell the whole car or sell the car in parts.

1.1 Literature Survey

Here are the main points we learned by reading a variety of sources.

Table 1: Literature Survey

Sr. No.	Title of the paper	Conference/Year	Purpose/ Key findings
1	Techniques to Predict Price of Used Cars: Predictive Analysis in Retail.	2021 Second International Conference on Electronics and Sustainable Communication Systems(ICESC)	Best Performance metric for prediction models: RMSE, MAE, R2.
2	Price Prediction: A Deep Learning Approach	2021 8th International Conference on Signal Processing and Integrated Networks(SPIN)	Hyperparameter tuning for models such as LSTM, LGBM, XGB.

2 Proposed method

2.1 Rationale

Our intuition for this project was to use various ensemble learning techniques to predict the price of used cars. We chose to build our prediction model using ensemble learning because we wanted to create a robust model which would reduce the dispersion of the predictions and model performance. We also introspected that we could achieve better performance with an ensemble than with any single contributing model as this technique is widely used for prediction problems.

2.2 Algorithm Description

We need to look for a model that can anticipate the price of used cars as accurately as possible. For that, we have chosen a variety of models, each with advantages and disadvantages of its own.

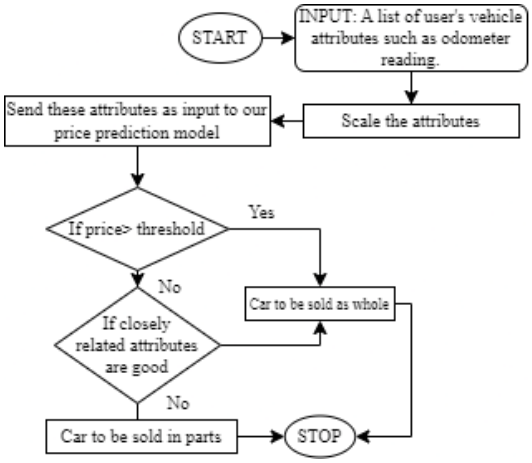

Figure 1: Flowchart

We chose a model that has the highest performance measures, and use it as out as our actual price prediction model. We want our model to be quick and not take a lot of time to run our model. After getting the output, we check if the important features of the car are worn out or not. If they are and the price of car is not a lot, we predict that it is better to sell the car in parts.

3 Plan & Experiment

3.1 Dataset

In our project, we have used the dataset: https://www.kaggle.com/datasets/austinreese/craigslist-carstrucks-data - a Used car dataset from Kaggle. The dataset contains over 426881 records of used cars from 42 different makes and 26 attributes. It is compiled by Austin Reese using Craiglist.

3.2 Hypothesis

As we had 25 independent attributes at our disposal to predict price, an intriguing question was how many of those attributes were actually related to the price. Our initial intuition was that attributes like the odometer, year, region, title_status, and posting_date would be a few of the most correlated attributes, and attributes like cylinder, drive, transmission, etc. would be some of the attributes with the least correlation to price. Our first instinct was that boosting algorithms would perform better than other algorithms as they reduce the error by decreasing bias since they aggregate output from many models.

3.3 Testbed

As our dataset is huge and prone to having many outliers, we planned to clean the dataset and perform EDA to find out irrelevant attributes. For model evaluation, we planned to start with simple regression models and move up to more complex models like ensemble models and neural networks.

3.4 Data Pre-Processing

Our approach begins with pre-processing of the dataset. After checking the data quality of the dataset, we remove all the columns that are not important to the objective, such as Site URLs, ID, Latitude and Longitude, etc. The dataset is cleaned for NaN, empty or duplicate values, and repaired by removing columns or rows so that we don't have empty values in the dataset.

The Bar graph shows the total NaN values present in all the features. Based on the results of the bar graph, we eliminated the attribute "size".

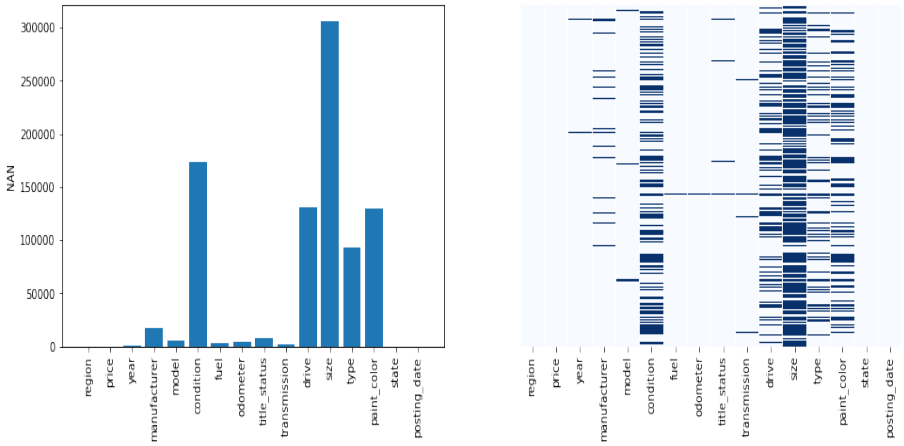

Figure 2: Bar Graph and Heat Map with total NaN values in each attribute.

After cleaning the dataset for empty or 0 values, we remove outliers so that they don't affect our data analysis. We found those outliers by creating box plots for our attributes and selecting the values from approximately the 20th percentile to the 80th percentile.

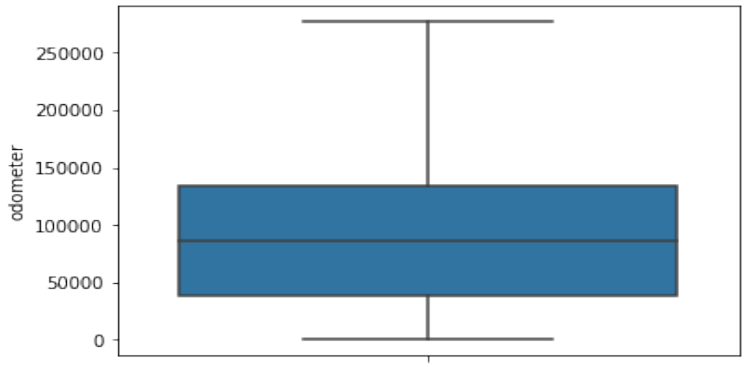

Figure 3: Box Plot for odometer.

Odometer Box Plot helps us understand the outliers in the attribute "odometer". We have used this box plot to filter out the outliers in the odometer by considering values greater than 30000 and less than 190000. We have chosen these values by considering the quantile range between 10 and 90 percentile.

Price Box Plot helps us understand the outliers in the attribute "price". We have used this box plot to filter out the outliers in the odometer by considering values greater than 4000 and less than 30000. We have chosen these values by considering the quantile range between the 10 and 90 percentile.

For further analysis, we convert attributes with object datatype to a usable format. We use label encoding to convert object datatype(categorical data).

We then drop the attributes that have a poor correlation with the price attribute. After all the data pre-processing steps, the columns - *year, manufacturer, model, condition, cylinders, fuel, odometer, transmission, drive*, and *type* were retained. The data was then standardized to make it consistent. Standardization was chosen over normalization as our data had outliers and standardization is immune to outliers.

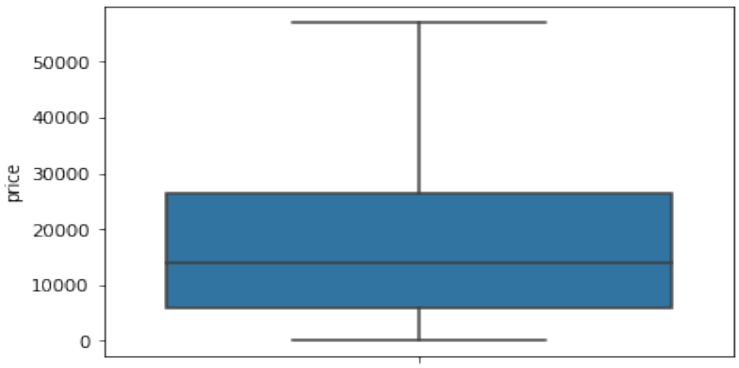

Figure 4: Box Plot for Price

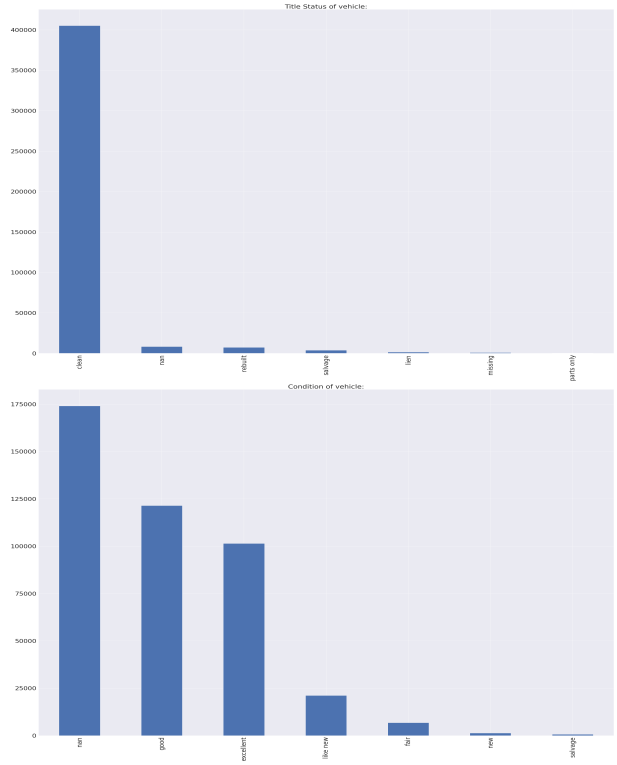

Figure 5: Title Status and Condition

From Title Status and Condition, we have selected condition over title status as we are getting more information from it.

3.5 Data Visualization

To provide us with insights and an understanding of the data, we would visualize the dataset in many forms. For a very large dataset, it is better to pre-process the dataset using various visualization. We plot the total NaN values in each attribute to find and remove attributes with exceedingly high NaN values. To find the outliers for our dataset, we use box plots.

We have visualized the dataset distribution and its correlation. We have also used bar plots to see the distribution of data for columns and selected columns that have a better distribution.

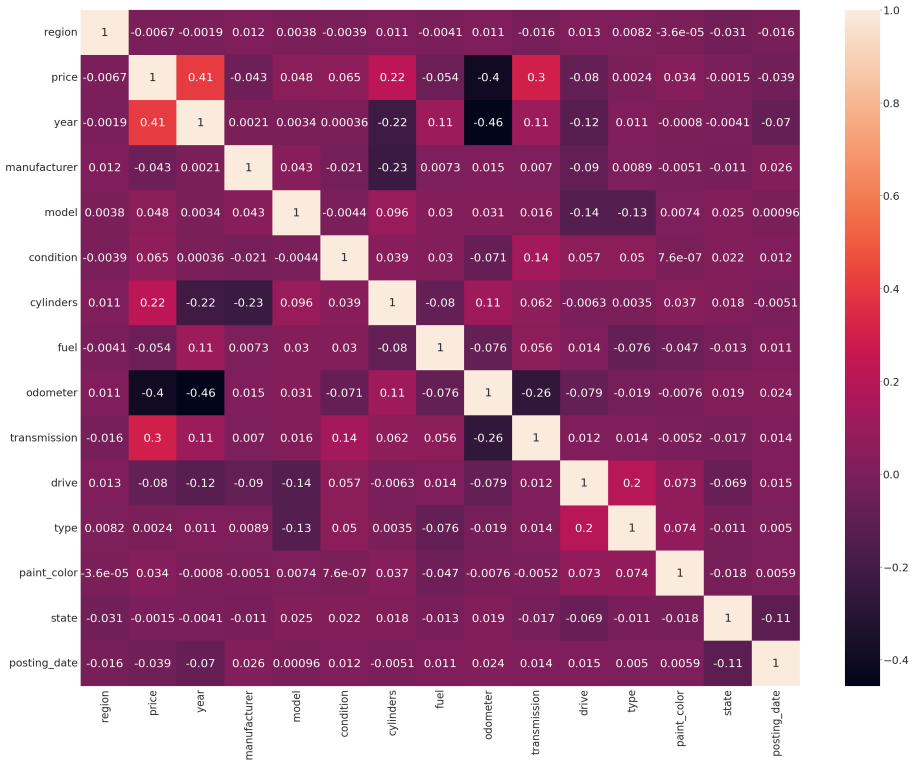

Figure 6: Correlation Matrix

The Correlation matrix represents a correlation between all the attributes. We can see that year, transmission, and cylinders are most positively correlated to price while odometer reading is most negatively correlated to price.

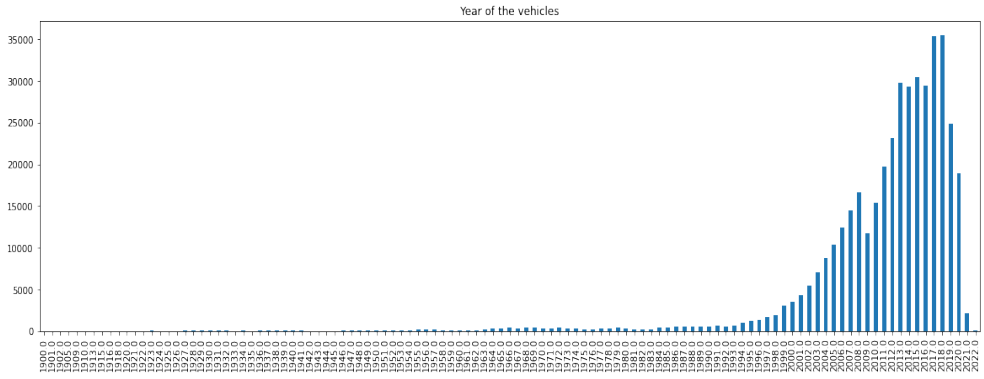

Figure 7: Distribution Graph for Year

The distribution graph shows the distribution of data over the year. We see that we have very little data before the year 1990, hence we have taken the data only for years greater than 1990.

4 Experiment Setup

The model construction, testing, and data analysis are done using Python 3. The data analysis and visualization for the dataset are done using Pandas, Matplotlib, and Seaborn libraries.

Firstly, we load the 'vehicles.csv' dataset into a Pandas dataframe. In the data pre-processing stage, we clean our dataset using the methods illustrated in figures 1,2,3,4. We check the NaN values and eliminate some attributes based on them.

Secondly, for finding outliers in the data, we use box plots that allow us to easily find outliers. We remove outliers for attributes such as odometer, year and price.

We then use Label Encoding to convert the categorical attributes into numeric attributes.

To remove attributes that have no correlation with each other, we plotted a correlation matrix.

After data pre-processing, we split our data into two parts namely, Training and Testing for regression models. For ANN models we split the data into Training, Validation, and Testing datasets. We then use the Sklearn library to train our linear regression model. To perform Standardization, we have used the StandardScaler module.

Finally, we test our model from the remaining dataset and use performance measures such as R2 Score, Mean Absolute Error, and RMSE. For Linear Regression, the accuracy is 34.75%.

The model obtained from Linear Regression will be used as a base model to compare with the other models that we have built.

5 Model Evaluation

After completely transforming our data into a usable format by removing or fixing redundant features, we use this data to identify the best model for price prediction. To find the best model using various performance metrics like R2 Score, Mean Absolute Error, and RMSE, we built various models like Linear Regression, SVM, LSTM, XGBoost, Random Forest, Adaboost, and LGBM. We split the dataset into training and testing datasets for regressors, and into training, validation, and testing datasets for the ANN models.

5.1 Linear Regression

Since here we have more than one dependent attributes, we use multivariable linear regression model here. We used this as a base model for our next models. It performed very poorly giving us a R2 score of mere 0.34.

5.2 SVM

We created a regression model using support vectors. We tried using both Polynomial and Radial Basis Function Kernel. For the Polynomial kernel, the degree was set to 3. The RBF kernel performed fairly better than the polynomial one. Though we got a better result than Linear Regression, the RMSE is still higher than 0.65.

5.3 Random Forest Regression

Random Forest Regression uses ensemble learning method for regression. Ensemble learning techniques are used to reduce the generalization error of the prediction. We have multiple decision trees on various sub-samples to control overfitting. At the end, the results from the decision trees are combined by taking the average and used for predicting the output. To improve the model's accuracy, we tried tuning the model using n_estimators which gives the number of trees in the forest. Criterion which is the loss function was set to squared_error. We achieved good accuracy using this model.

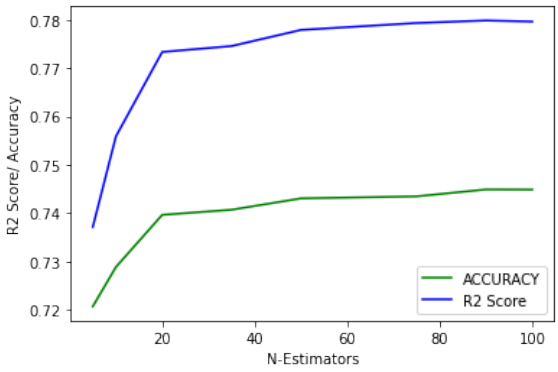

Figure 8: N-estimators affecting RFR accuracy and R2 score.

5.4 ANN using LSTM

For this, we used 2 LSTM layers, 1 dropout layer and 2 dense layers. The dropout was set to 0.2 to avoid overfitting. Early Stopping was also used to avoid overfitting and achieve better learning. At the 93rd epoch, we achieved the best accuracy of 68.29%.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 10, 128)           66560

 dropout (Dropout)           (None, 10, 128)           0

 lstm_1 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 32)                2080

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 118,081
Trainable params: 118,081
Non-trainable params: 0
_________________________________________________________________
```

Figure 9: ANN Model Summary

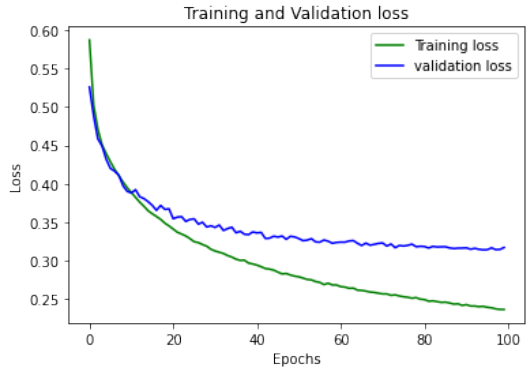

Figure 10: Validation vs Training Loss for 100 epochs

5.5 Light Gradient Boosting Machine (LGBM)

LGBM is a boosting framework which is a decision-tree-based algorithm. To get higher accuracy, we tuned the hyperparameters like num_leaves, learning_rate, and max_bin. To handle over-fitting we tuned the hyperparameters like bagging_fraction and bagging_freq. It was observed that with lower values of num_leaves, the accuracy of the model was low and for very high values the model had an overfitting problem. GridSearch CV was used was used for tuning hyper parameters like num_leaves. A smaller learning rate of 0.01 was used for better accuracy.

5.6 Extreme Gradient Boosting

XGB is also a decision tree-based algorithm and an ensemble learning technique. It is a boosting framework similar to LGBM, but LGBM grows leaf-wise and XGB grows depth-wise. We kept tuning the n_estimators parameter until we achieved good accuracy.

6 Results

Table 2: Results- Models sorted w.r.t. their R2 score

Sr. No.	Model	R2	MAE	Accuracy	RMSE
1	Random Forest Regressor	0.7797	0.2557	0.7442	0.4661
2	LGBM	0.7292	0.3252	0.6747	0.5167
3	XGB	0.6994	0.3468	0.6531	0.5445
4	ANN Using LSTM	0.6832	0.317	0.6829	0.559
5	SVR -RBF	0.5684	0.3979	0.602	0.6524
6	SVR- Polynomial	0.3914	0.5142	0.4827	0.7747
7	AdaBoost	0.3801	0.5974	0.4025	0.782
8	Linear Regression	0.3477	0.5756	0.4243	0.8021

7 Critical Evaluation

The goal of this project was to create a price prediction model that would help users find an accurate estimate for car prices in dollars. The dataset was huge and required a lot of pre-processing. Only the attributes necessary were selected and analyzed. We observed that price was highly skewed by outliers. We understand the correlation between price and various attributes such as odometer reading, year, etc. The linear regression model performed the worst mainly because it oversimplified the problem statement. We achieved the highest accuracy of 74.42% and R2 score of 77.97% with the Random Forest Regressor. Hence, we selected RFR as our price prediction model. We could have implemented additional learning techniques to further classify whether the car could be sold for just its parts. The only limitation was that we did not have enough relevant data in our dataset to achieve this efficiently.

8 Conclusion

At the start of the experiment, we had assumed that region would one of the most correlated attributes to price and cylinder to be the least correlated which was incorrect. We also expected XGBoost to perform better than RFR, but we learned that RFR was a better choice for our dataset as it works well even if there is a lot of missing data, while XGBoost is hard to work with when there is noisy data. RFR was also easier to tune compared to LGBM and XGB. Hence, we learned about different decision tree–based learning algorithms.

9 Link for Github

https://github.ncsu.edu/kpmhatre/engr-ALDA-Fall2022-H9/tree/main/PROJECT