SMU
SINGAPORE MANAGEMENT UNIVERSITY

School of
Computing and
Information Systems

CS610: Applied Machine Learning

Car-eful Consideration: An AML approach to purchasing cars

Group 10

AISHWARYA SANJAY MALOO

CHUA WEE YUAN

LAU HO YIN AMANDA FAITH

PRACHI RAJENDRA ASHANI

UTHARA VENKATACHARI

CONTENTS

INTRODUCTION

Buying a used car or a good-as-new car is often a difficult task for the uninitiated. Those who do not have any or little prior experience are faced with a range of options and price points. To gauge if the car is priced fairly may be difficult for the average buyer as they would not know the different factors that impact the pricing and may need extensive research to get acquainted with them.

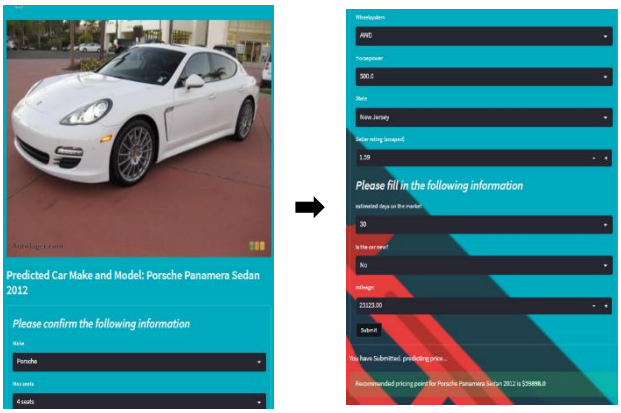
Nonetheless, most buyers can benefit significantly by understanding how similar cars have been priced previously and ascertain whether they are considering the purchase of a used car at a fair deal. Equipping the buyers with indicators can help them estimate a reasonable value for a car and place them in a better position to negotiate the final selling price with their dealers.

PROJECT OBJECTIVES

We seek to bring clarity to the decision-making process whilst equipping buyers with the knowledge to negotiate a better price. Therefore, by using images of cars labelled with their make and model along with historical pricing data, we're able to build a pipeline of 2 ml models to, from an image of a car, determine its make and model (image multi-classification) and determine its selling price (price prediction model).

PROJECT SCOPE

In this project scope, we aimed to build a minimum viable product that can be readily deployed for use. We have successfully achieved this by using the datasets from Stanford cars (196 car make and model) and US used car price from Kaggle. The image on the right, shows our deployed product on Streamlit.



With the make and model determined, we were able to determine certain features unique to make and model of the car (such as wheel system and horsepower). With the user inputting additional information like mileage, we were able to output an accurate price of the car based on these features.

DATA SOURCE

Data	Car Images dataset	US Used Car Dataset
Purpose	Used for car vision model	Used for price prediction model
Source	Stanford	Kaggle
Sample size	16,185	3,000,040
Features	N/A	66

The Stanford Cars dataset is a collection of 16,185 images of different types of cars. The dataset is organized into 196 classes, where each class represents a specific type of car. The classes are typically categorized based on the make, model, and year of the car. For instance, a 2012 Tesla Model S or a 2012 BMW M3 coupe would belong to different classes. The dataset is further split into two subsets, a training set and a testing set with a 50-50 split between the training and testing sets. For ease of training, the images are grouped into their respective class folders making it easy to locate images of a specific type of car in the dataset for training and testing.

The US used car dataset contains details of 3 million real world used cars obtained by crawling through Cargurus inventory in September 2020. The data includes features such as the vehicle's VIN, body type, engine type, fuel economy, legroom, cabin and bed size (in the case of pickup trucks), exterior and interior color, mileage, horsepower, and whether it has been involved in any accidents or has a damaged frame. Other features include the dealer's location, the listing date, and whether the vehicle is certified or new.

METHODOLOGY

Computer Vision

Data cleaning and pre-processing

Both the train and test set was subjected to a resizing of 299x299 pixels and the images are normalized using mean and standard deviation values of [0.470, 0.460, 0.455] and [0.267, 0.266, 0.270], respectively. Data augmentation was performed on the training set to increase size of training set and help the model learn more robust features from various viewpoints and perspectives. Below summarizes the various data augmentation technique applied to the training dataset.

- RandomResizedCrop: randomly crops the images to a smaller size and then resizes them back to 299x299 pixels with some randomness in the scale and aspect ratio.
- RandomRotation: randomly rotates the images up to 35 degrees.
- RandomHorizontalFlip: randomly flips the images horizontally with a probability of 0.5.
- ColorJitter: randomly adjusts the brightness, contrast, saturation, and hue of the images with some randomness.
- RandomAdjustSharpness: randomly sharpens the images with a probability of 0.5.
- RandomGrayscale: randomly converts the images to grayscale with a probability of 0.5.
- RandomPosterize: randomly reduces the number of bits used to represent the color of the images to 2 with a probability of 0.5.
- RandomPerspective: randomly (probability = 0.5) applies a perspective transformation to the images.

Model creation

To train this 196 multi-class image classification model, we evaluated different pre-trained backbones from the torchvision package, including ResNet50, ResNet152, EfficientNetV2_L, and InceptionV3. For each model, training was done for around 100 epochs, of which 20 epochs with the base layer frozen, and another 80 epochs with the base layer unfrozen to finetune the weights. The model was saved after each epoch and its performance evaluated based on precision, recall, and accuracy metrics.

To further optimize the performance of the best model, hyperparameters were varied. Specifically, the FC layer was modified by changing its depth to 2 layers and using the ReLU activation function, and a dropout rate of 0.3. The model was trained using the Adam optimizer with a learning rate scheduler of cosine annealing, and a batch size of 32. The resolution of the input images was also varied to see if it affects model performance.

By trialing different models and hyperparameters, our goal was to select the best performing model for the given dataset and achieve the highest possible accuracy.

Model evaluation

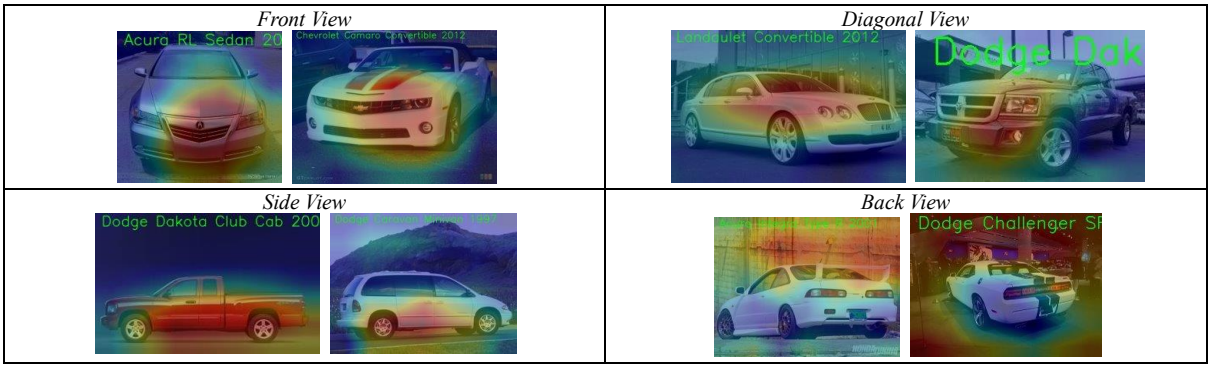
Model	Resnet50 input size (224x244)	Resnet152 w FC Input size (299x299)	Efficientnetv2_L w FC Input size (299x299)	Inceptionv3 w FC Input size (299x299)
Precision	77%	90%	93%	85%
Recall	77%	89%	92%	84%
Accuracy	79%	89%	92%	84%

Results from our model trials showed that the best performing model is the efficientv2_l with a fully connected layer and an input size of (299x299). We were able to achieve a 93% weighted average precision score, 92% weighted average recall score and 92% weighted average accuracy. This is unsurprising as efficientv2 achieves state-of-the-art performance on multiple image classification benchmarks (Mingxing Tan, 2021). It uses a novel architecture that combines various techniques such as compound scaling, SE-Net, and Swish activation function, resulting in improved performance and better generalization ability.

Class Activation Mapping

Class Activation Mapping (CAM) is a technique that allows for the visualization of which regions of an image are important for a deep learning model's classification decision. This can be useful in to help us understand and interpret the model's behaviour, debug the model by identifying misclassifications, and identifying potential weaknesses or biases in the model's decision-making process (Zhoubolei, 2016).

When we overlay the class activation heat map with some of the model testing images, we can see below that the model is able to learn the important features of the class of the car by focusing on headlight/fog light, front bumper and bonnet. From the side view, it's the doors and wheels. However, the most misclassifications come from the back view where the model is unable to focus well on parts of the car as seen below. One possible way to improve the model is to train the car on more images of back views of the car so it can learn the relevant features.



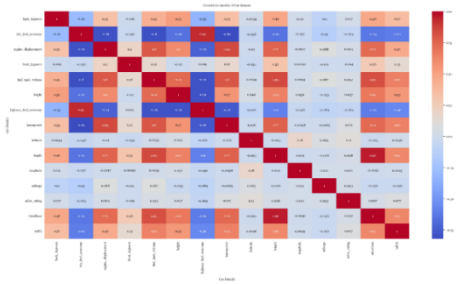
Price prediction

Data cleaning and pre-processing

Observation	Solution
'has_accidents' variable has three categories, 'True', 'False' and 'NA'.	The 'NA' values were recoded to 'Unknown' to retain the variable as that was deemed an important feature in determining the final pricing.
Torque, a numerical variable, is taken as 'object' and due to '@' present in its data, it will be considered categorical by Python	Split the variable by '@' and keep the former half, retaining the feature name 'torque'.
Engine type had information on the kind of engine as well as the number of cylinders.	The variable was split into two to get 'engine type', an alpha value, and 'cylinders', a numerical value, as new variables.
The cities the cars were listed in, had over 4000 categories. Too many categories could have hampered the accuracy of the prediction models.	The state of the corresponding city is extracted and converted to a new variable 'State'. The categories of the state variables are much fewer than city variable and it could be an important differentiator from a price standpoint. For instance, cars are known to be more expensive in New York vis-a-vis Arizona.
Year (assumed as the purchase year of the car) is a temporal variable which may not best suited as a feature in a price prediction model.	The age of the car can be an important determinant of price. This variable can be extracted from the purchase year of the car. A new variable called 'age' is created.
Features such as car VIN, and listing_id, are unique values that cannot help in predicting the target variable. Furthermore, year and listing_date are temporal variables that may not offer much insight in prediction models.	These features are dropped.
There were several features that had over 20% missing data.	The options were to drop these values, impute them, or treat them as target variables and predict their values. The last two solutions could have aggravated multicollinearity in the data. Hence, all the 'NA' values were dropped.
Post removing variables with missing values, the dataset still had many rows with at least one feature marked as NA. NA values do not bode well for machine learning models.	The options were to drop these values, impute them, or treat them as target variables and predict their values. The last two solutions could have aggravated multicollinearity in the data. Hence, owing to the paucity of time - all the 'NA' values were dropped.
The data type is 'object' for all the features.	They are converted to either numerical or categorical based on their original data types.

Exploratory Data Analysis

A. Correlation Analysis:



Numerical variables: The correlation plot visually shows the bivariate relationship between variables. In comparison, Variation Inflation Factor (VIF) helps quantify the correlation of one independent variables to other independent variables. VIF is used in conjunction with correlation plot to decide what variable to out of two highly correlated variables. Post the omission of multiple variables.

Figure 1: Correlation Plot for Continuous Variables

Categorical variables: Cramer's V is used to measure of association between categorical variables. Unlike correlation, the output is in the range of 0 and 1, that is, there is no negative correlation. However, this serves as a preliminary metric to eliminate highly correlated categorical variables (Zychlinski, 2018).

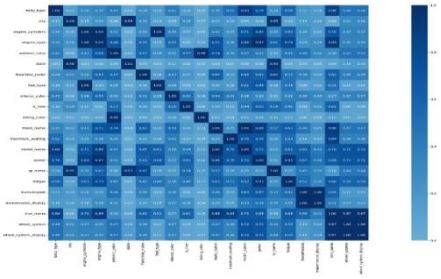


Figure 2: Cramer's V for Categorical Variables

B. Data Distribution:

(i) Continuous variables: As observed (see Appendix for charts), most of the features do not follow parametric distribution (skewed on either side). Such a data distribution can hamper the model accuracy and, therefore, there is a need for normalising the data.

(ii) Categorical Variables: We observed that certain categorical variables have a higher frequency for some of their categories (see Appendix for charts). Such a significant imbalance can cause bias in modelling. For instance, the dataset has much higher frequency of gasoline fuel_type cars compared to diesel, compressed natural gas, hybrid etc. fuel_type cars. Additionally, features such as 'has_accidents' and 'franchise_dealer' have only one category, and hence serve no purpose in the prediction models and were omitted.

(iii) Outlier Analysis:

a. Numerical variables: Car prices, in ascending order, while seem mostly linear, there is a sharp upturn most likely from the values of top echelon cars. On further analysis of the car prices for the bottom 99% (see figure 4), we can tell that the car prices follow a cubic graph shape. But with the majority of price points (~100k mark to 1.4mil mark) follow a linear graphical relationship and are thus retained in the dataset for further analysis.

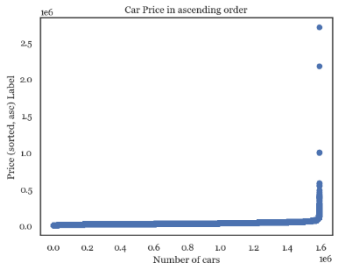


Figure 3: Car Prices for Bottom 99%

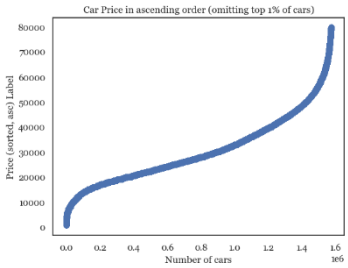


Figure 4: Car Prices for Top 1%

b. Categorical variables: From the graph below, we can see that there are outliers present in the data that can adversely affect the model accuracy. Thus, such extremities need to be handled in using the available scaling methods in Python.

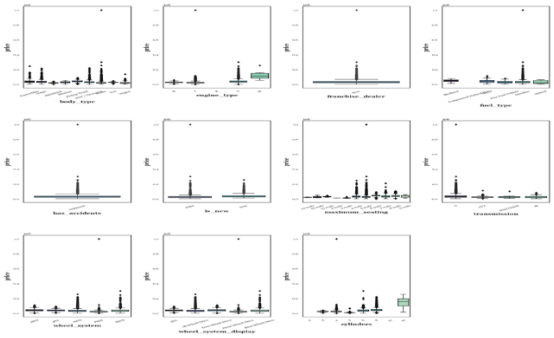


Figure 5: Outlier Analysis for Categorical Variables

Processing post EDA

After all the variables are finalised: (Zychlinski, 2018)

1. Dummy variables are created from the remaining categorical variables.
2. The data is split into training, validation, and test data in the ratio 6:2:2.

The final dataset had over 1.5 million rows and 111 features including car price (target variable). Both robust and standard scaler were applied to the input variables as they worked better together than if they used in silo for this dataset. Robust scaling helped account for the outliers that were observed during our outlier analysis whereas standard scaling was used to centre the target variable around a mean value of 0.

Feature Selection and Decision Tree Model

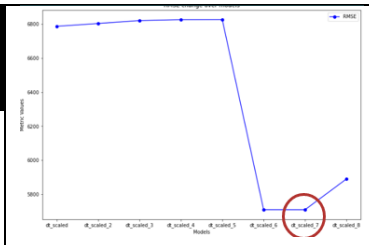
Feature selection was conducted considering the large number of variables to be dealt with that could potentially increase the model complexity with only marginal improvements in scores and explainability. First, we omitted the variable savings_amount since its definition is ambiguous and from a customer perspective, we would not have been able to explain with confidence on what it meant.

Next, we ran a Decision Tree model with the initial dataset post cleaning and omission of *savings_amount* and plotted the feature importance diagram to analyse the variables. The process was repeated iteratively and sequentially by dropping a new feature with low importance on the input dataset containing one less feature from the previous omission. Each Decision Tree model built per iteration was evaluated on the metrics R^2, RMSE, MAPE and bias-variance changes. For the purpose of consistency, the same metrics were used for the evaluation of Multilinear Regression and Neural Networks performed afterwards. R^2 was used as it showed the variance explained by the model. RMSE gives a dollar value representation to the size of the error putting context to the business problem. Without a baseline, however, understanding whether the error size was considered large or small was a challenge. Hence, we included MAPE as a metric as well. MAPE scales the error to be a percentage of the target variable. This managed the fact that car prices are typically numerically large (and went up to the thousands). Therefore, this scaling helped provide more information on how concerning the error amount was in relation to the car price and provided a band for negotiations.

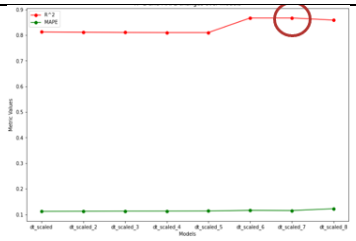
In total, 8 Decision Tree models were built, but the 7th model was used moving forward (with all feature omissions up until that point) as the 8th model had the variable *State* omitted and we saw the results worsen. Figure 6 shows the evaluation metrics tracked against the gradual omission of variables and the table on the left of Figure 6 shows the variable omission sequence in each Decision Tree Model.

Metrics	Test set - XGBoost
RSME	10,516.08
MAPE	53.67%
R^2	17.78%

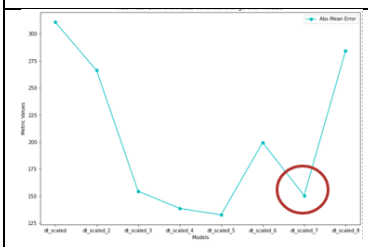
Legend	
Decision Tree	Variable Omitted
1.dt_scaled	1. savings_amount
2.dt_scaled_2	2. transmission_type
3.dt_scaled_3	3. fuel_type
4.dt_scaled_4	4. maximum_seating
5.dt_scaled_5	5. engine_type
6.dt_scaled_6	6. body_type
7.dt_scaled_7	7. lisitng_colour
8.dt_scaled_8	8. state



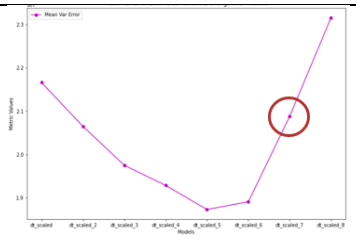
RMSE change over models



R^2 and MAPE change over models



Absolute Mean Error (Bias) change over models



Mean Variance Error (Variance) change over models

Figure 6: Model Performance throughout Feature Omission

Modelling

Final Model Variables

The variables included are: 'is_new' (whether it's a first-hand car), 'make_name' (brand of car), 'maximum_seating' (seats in the car: 4,5,6,7,8), 'wheel_system' (4x2, All Wheel, Forward, Rear Drive), 'state' (US state), 'horsepower' (engine power), 'days_on_market' (duration car has been for sale), 'mileage' (distance travelled), 'seller_rating' (proxy for seller trustworthiness).

Decision Tree

Using the default hyperparameters in *sklearn's* DecisionTreeRegressor, we ran the model on our test data and got an RMSE of 5255.19, MAPE of 12.03% and R^2 or 88.43% which indicates that the model performed well in generalizing unseen data.

Metrics	Train set	Test set
RSME	5,709.27	5,255.19
MAPE	11.53%	12.03%
R^2	86.74%	88.43%

Figure 7: Training and Test Set Results for Decision Tree

K Fold cross validation (see table below for results) was used to perform hyperparameter tuning with n_splits = 10, shuffle = True, random_state = 2023 for varying training-validation splits. The validation set had an average R^2 score of 84.6% and average MAPE of 10%.

Fold	1	2	3	4	5	6	7	8	9	10
R^2	88.67%	90.60%	72.04%	88.91%	91.29%	81.20%	86.10%	90.77%	78.94%	76.99%
MAPE	9.972%	9.977%	10.04%	9.972%	10.02%	9.978%	10.04%	10.04%	10.06%	10.04%

Figure 8: Test Set Results for XGBoost

As mentioned, we interpret this MAPE to be a negotiation margin for the customer. Based on the recommended price predicted by the model, the customer has approximately a 10% margin to negotiate the car price.

XGBoost

A Gradient Boosting model was explored using XGBRegressor with parameters, objective reg:squarederror, eval_metric = mape, learning_rate = 0.3, lambda = 2, and n_estimators = 300.

Four learning rates – 0.001, 0.01, 0.1, 0.2 and four estimators – 200, 300, 400, 500. Using a scoring of negative MAPE in gridsearch, we found that the best estimator had a learning rate of 0.2 and number of estimators as 500. Figure 8 shows the evaluation metrics of applying the optimal hyperparameters on the test set. Since our gradient boosting model did not perform as well as the Decision Tree model, it was dropped from our model comparison, and we proceeded with attempting Multilinear Regression and Neural Networks instead.

Multilinear Regression (MLR)

The dataset was split 70-30 to include only a train and test set since validation was not intended for this approach. Using the split dataset containing the final set of features upon scaling, sklearn's LinearRegression was used to train the model and produced the test results as seen in Figure 9. To avoid overfitting and reduce complexity, we performed Lasso regularisation with different alpha values (0.001, 0.01, 0.1, 1, 10) and found that an alpha of one was the most optimal in terms of providing a low RMSE, ideal complexity that neither overfits nor underfits the data, and low regularized cost.

Metrics	Test set - MLR
RSME	6,953.79
MAPE	79.93%
R^2	17.94%

Figure 9: Test Set Results for MLR

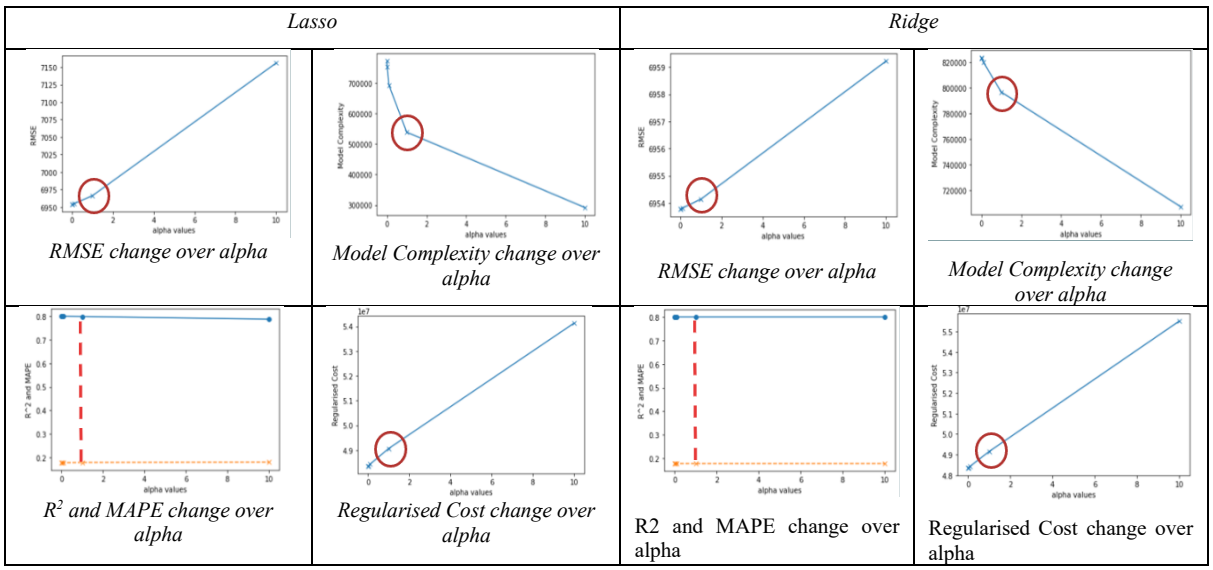


Figure 10: Lasso & Ridge Regularisation Alpha

Ridge was not preferred over Lasso since it considers all features as relevant to the model, which may not be the case. Nonetheless, it was used to compare the performance of Lasso and we found that the outcomes for all the metrics used indicate an optimal alpha of one.

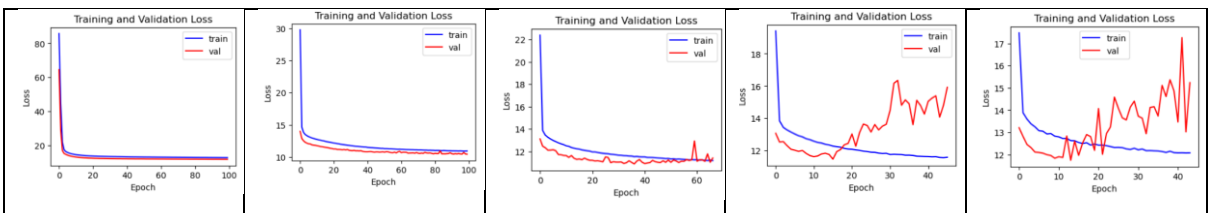
Overall, while Multilinear Regression is not a preferred approach for price prediction due to its simplicity and lack of consideration for nonlinear relationships, it still fared comparably well – likely because much of the price data was near linear (see figure 4 for more details).

Neural Network

The neural network was built using the Tensorflow and Keras package of python. Taking the lead from the decision tree model, 110 features (post one-hot encoding) were considered for the analysis. The data was transformed using both Robust and Standard scalers to address the outliers and non-parametric distribution issues. This dataset was split in train-test-validation set with ratio 60:20:20.

The neural network had an input layer with 128 units/neurons, 110 input dimensions – matching the size of input features. There was one hidden layer with 64 units/neurons and the output layer had one unit/neuron as this was a price prediction model. Dropout of 0.2 was set between the input and the hidden layer to regularize the model. See below the details of the network modelled.

Half the dataset (750,000 datapoints) was used to fit the neural network model as computing hardware limited the speed of analysis possible. Five learning rates (0.001, 0.01, 0.025, 0.05, 0.1) were used to develop five neural networks with 100 epochs and a batch size of 1,024. The Leaky ReLU activation function was used as this accounts for the vanishing gradient problem. Tanh and ReLU functions were avoided as Tah function leads to the problem of vanishing gradient, while ReLU function does not perform well with negative values. Below are the results derived from each of the learning rates.

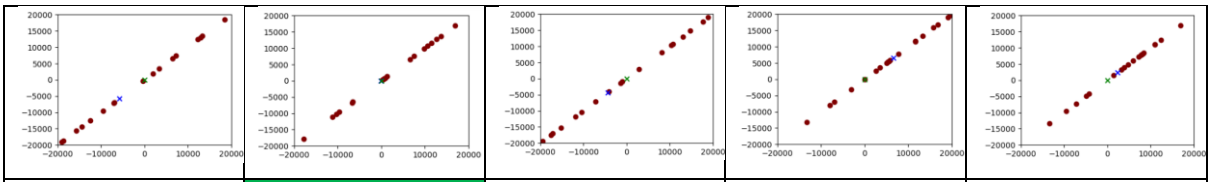


Learning rate: 0.001	Learning rate: 0.01	Learning rate: 0.025	Learning rate: 0.05	Learning rate: 0.1

Figure 11: Training and Validation Set Loss Plots

Metrics	Train set				
LR	0.001	0.01	0.025	0.05	0.1
RSME	6,720.82	6,110.62	6,561.90	8,149.58	7,968.67
MAPE	11.62%	10.34%	11.29%	15.89%	15.24%
R^2	80.82%	84.14%	81.71%	71.79%	73.03%
	Validation set				
RSME	5,836.34	5,117.44	5,756.32	7,549.01	7,301.64
MAPE	11.68%	10.44%	11.47%	15.92%	15.23%
R^2	85.13%	88.57%	85.53%	75.12%	76.72%
	Test set				
RSME	5,675.82	4,943.93	5,577.01	7,426.58	7,186.72
MAPE	11.74%	10.49%	11.47%	16.04%	15.35%
R^2	85.78%	89.21%	86.27%	75.65%	77.20%

We also analysed the bias-variance for each of these learning rates.



Learning rate: 0.001 Bias-Variance: -5,751.00	Learning rate: 0.01 Bias-Variance: -49.17	Learning rate: 0.025 Bias-Variance: -4293.37	Learning rate: 0.05 Bias-Variance: 6,617.93	Learning rate: 0.1 Bias-Variance: 2,377.51

Figure 12: Bias Variance Plots

Next, K Fold cross validation (see table below for results) helped determine the overall performance of the neural network model with learning rate 0.01. Again, the K Fold parameters used were same as those used in the Decision Tree model (n_splits = 10, shuffle = True, random_state = 2023) to maintain consistency and comparability across the different models created thus far. The average R^2 score for test set was 85.7% and average MAPE was 10.4%.

Fold	1	2	3	4	5	6	7	8	9	10
R^2	89.57%	89.57%	88.82%	89.94%	84.28%	88.51%	89.12%	88.28%	88.11%	60.65%
MAPE	10.34%	10.20%	10.19%	10.21%	10.19%	10.49%	10.56%	10.34%	10.37%	10.65%

Prediction models comparison

From the three models developed, we believe each of the models bode well for the purpose of the car price prediction as each have an R^2 score of over 80.0% and MAPE of at most 12.0% for the test set. However, given that the neural network offered marginally better results than the other two models, the team chose neural network as the final car price prediction model. See below for the comparison across all three models.

Model	Decision Tree	Multi Linear Regression	Neural Network
RMSE	5,255	6,954	4,943
R^2 Score	88.4%	79.9%	89.2%
MAPE	12.0%	17.9%	10.4%

PROJECT LIMITATIONS

Insufficient computing power: For both image classification and feature analysis model, the number of hyperparameter trials, epochs and the size of the dataset had to be limited. Given access to better GPUs would increase computing power thereby allowing us to further tune these hyperparameters parameters, running sufficient epochs until convergence with various batch sizes, testing out different combinations of layers in the fully connected layer of the model, and running different regularized Decision Tree and MLR models. Such an exercise can result in highly optimized price prediction and image classification models.

Large number categories of car make and models: In the real world, the number of car makes such as BMW, Audi, Porsche and models such as iX, i7, i5, i4 etc. for BMW are extremely large. Additionally, local brands, such as Maruti cars in India, add to this complexity. It is challenging to capture all combinations of every make and model of cars that have existed to date to accurately predict each of them through image classification.

Geographical constraints: The pricing data is from the United States. Pertinent factors such as certain make and models of the cars, and the state (California, New York, etc.) they were purchased in are privy only to the US.

This renders the pricing model unsuitable to be applied for other countries. For instance, India has local brands such as Mahindra and Maruti that are extensively used in the country are unavailable in our current dataset.

Difficulty in quantifying hidden variables: The condition of a used car can greatly affect the price it should be sold at. For example, two cars of the same make, model, and year could have vastly different conditions and their prices will be reflective of that. Better maintenance often translates to a better price. However, condition is a subjective variable that is difficult to quantify in the model. In the future, the condition of the car can be a categorical variable based on inputs from the computer vision model – that is, if the computer vision model can evaluate how well the car is maintained by the look of it, it can recommend an appropriate 'condition' category.

Market conditions are also not reflected in this model and can be considered in future iterations. For example, periods of economic uncertainty, interest rate fluctuations and supply and demand changes can affect the value of the car. For example, with the supply of new cars at a near standstill at the height of the coronavirus pandemic, used car prices in the United States of America have been on the rise (Wayland, 2023).

FUTURE WORK AND CONCLUSION

Bolster the data: By adding more images and car classes to the dataset, the CV model can capture the full range of variability among different car classes, learn a wider range of features and improve its pattern recognition that differentiates each car class from the others. In this way, the predicted prices of cars become more precise. With a larger dataset, we can group the car brands and models into different categories and train each grouped category with its specific image classification model. This can enhance the prediction power as we narrow the model's focus on a smaller set of classes. This approach, called "fine-grained classification," allows the model to focus on a smaller set of classes and learn more specialized features that are specific to each group. For example, a model trained to classify sedans may learn to recognize the specific features that differentiate sedans from other types of cars, such as SUVs or sports cars. However, it is important to note that adding more data and creating more specialized models also increases the complexity and computational cost of the overall system.

Trial various transfer learning backbone: Pre-trained model (backbone) such as resnet, Efficientnetv2 and inceptionV3 were evaluated for the image classification model. More transfer learning backbones could be evaluated to further improve the performance of the image classification model on the Stanford car images dataset. There are a wide range of pre-trained models available for transfer learning in computer vision, such as VGG, DenseNet, MobileNet, ResNeXt, and many more. Each of these models has its own strengths and weaknesses and may perform differently depending on the specific dataset and task so it may be useful to try out several other different pre-trained models as backbones for the image classification model and evaluate their performance on the Stanford car images dataset. This can help to identify the best backbone for the task at hand.

Higher resolution images for image detection model: During the pre-processing of car images for the image classification model, an input resolution of 299x299 was used. This increase in resolution from 244x244 gave us better model performance as it allowed to model to better learn the necessary car features. More experiments can be done to further increase the input resolution to see if it increases model performance. However, it is also important to consider the potential trade-offs between model performance and computational cost when deciding on the input resolution. A higher resolution may lead to better accuracy, but it may also make the model slower and more resource-intensive, which can be a practical concern in real-world scenarios.

Hyperparameter tuning to Optimise neural networks: To improve the accuracy of both the computer vision model and the price prediction model, Neural Networks can be optimized through various techniques, including pruning, experimentation with different batch sizes, running a higher number of epochs til convergence, using different optimizers, and activation functions. By implementing these hyperparameter tuning methods, it is possible to significantly enhance the models' performance.

Performing polynomial regression: Polynomial regression is a potential model to explore as it is suitable when there is a non-linear relationship between the independent variables and target variable. However, there could be an issue of overfitting if too many polynomial terms are involved.

Regularisation for decision tree: For the current iteration we ran XGBoost for Decision Tree model. However, the R^2 score was lower (53.7%) and the MAPE (17.8%,) was higher compared to our other models. Regularisation in decision tree requires adding an objective function that can reduce tree complexity and thus prevent the problem of overfitting.

In conclusion, while this project has provided a minimum viable product for a car detection and price prediction consumer product, there is still room for improvement to make the predictions more precise – from lower accuracy for car detection, to a lower MAPE for a smaller negotiation margin.

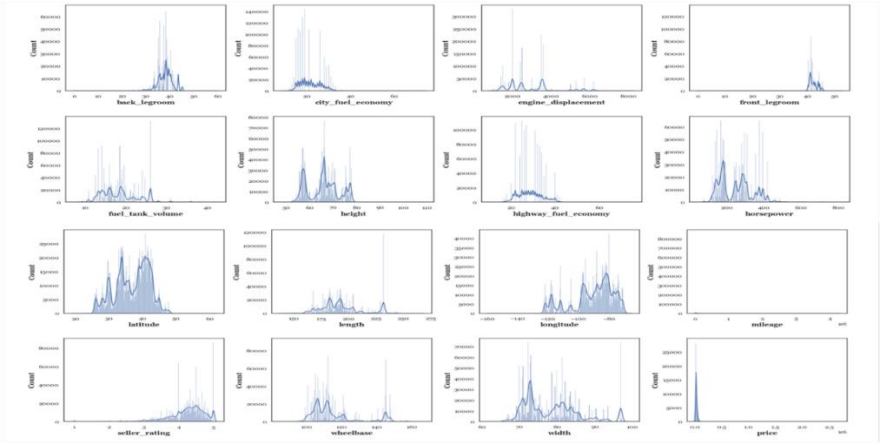
APPENDIX

The original dataset had 66 features. We derived or modified some features such as age, state, engine type, and number of cylinders. Many features needed to be omitted for the one of the following reasons – the features were unique identifiers, were descriptors, had large amounts of missing values, their purpose was unknown, or were highly correlated.

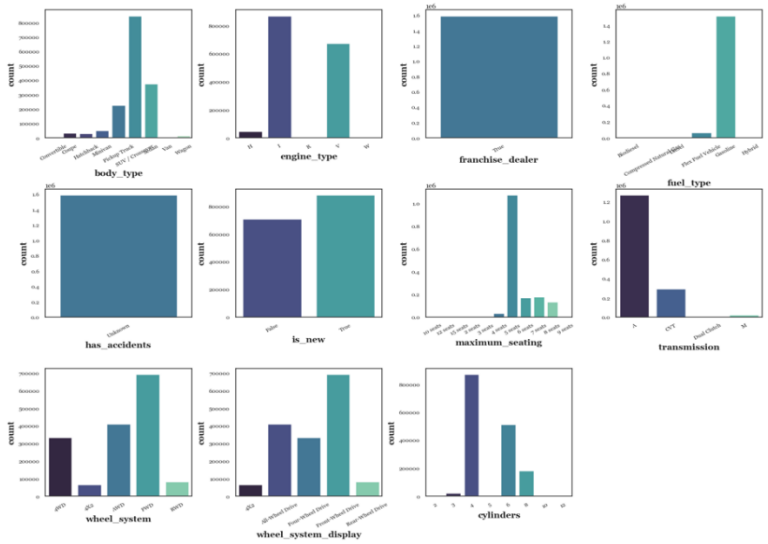
Attribute	Data Type	Retention Status	Description
back_legroom	Continuous	Dropped	Legroom in the rear seat
bed	Categorical	Dropped	Category of bed size (open cargo area) in pickup truck. Null usually means the vehicle isn't a pickup truck
bed_height	Continuous	Dropped	Height of bed in inches
bed_length	Continuous	Dropped	Length of bed in inches
body_type	Categorical	Dropped	Body Type of the vehicle. Like Convertible, Hatchback, Sedan, etc
cabin	Categorical	Dropped	Category of cabin size (open cargo area) in pickup truck
city	Categorical	Dropped	city where the car is listed
state	Categorical	Derived & Retained	
city_fuel_economy	Continuous	Dropped	Fuel economy in city traffic in km per litre
combine_fuel_economy	Continuous	Dropped	Combined fuel economy is a weighted average of City and Highway fuel economy in km per litre
daysonmarket	Continuous	Retained	Days since the vehicle was first listed on the website
dealer_zip	Continuous	Dropped	Zipcode of the dealer
description	String	Dropped	Vehicle description on the vehicle's listing page
engine_cylinders	Categorical	Dropped	The engine configuration
engine_displacement	Continuous	Dropped	engine_displacement is the measure of the cylinder volume swept by all of the pistons of a piston engine, excluding the combustion chambers
engine_type		Dropped	The engine configuration
engine_type	Categorical	Derived & Dropped	
cylinder	Continuous	Derived & Dropped	Number of cylinders in an engine
exterior_color	Categorical	Dropped	Exterior color of the vehicle, usually a fancy one same as the brochure
fleet	Boolean	Dropped	Whether the vehicle was previously part of a fleet
frame_damaged	Boolean	Dropped	Whether the vehicle has a damaged frame
franchise_dealer	Boolean	Dropped	Whether the dealer is a franchise dealer
franchise_make	Categorical	Dropped	The company that owns the franchise
front_legroom	Continuous	Dropped	The legroom in inches for the passenger seat
fuel_tank_volume	Continuous	Dropped	Fuel tank's filling capacity in gallons
fuel_type	Categorical	Dropped	Dominant type of fuel ingested by the vehicle
has_accidents	Categorical	Dropped	Whether the vin has any accidents registered
height	Continuous	Dropped	Height of the vehicle in inches
highway_fuel_economy	Continuous	Dropped	Fuel economy in highway traffic in km per litre
horsepower	Continuous	Retained	Horsepower is the power produced by an engine

interior_color	Categorical	Dropped	Interior color of the vehicle, usually a fancy one same as the brochure
is_certified	Boolean	Dropped	Whether the vehicle is certified. Certified cars are covered through warranty period
is_cpo	Boolean	Dropped	Pre-owned cars certified by the dealer. Certified vehicles come with a manufacturer warranty for free repairs for a certain time period
is_new	Boolean	Retained	If True means the vehicle was launched less than 2 years ago
is_oemcpo	Boolean	Dropped	Pre-owned cars certified by the manufacturer
isCab	Boolean	Dropped	Whether the vehicle was previously taxi/cab
latitude	Continuous	Dropped	Latitude from the geolocation of the dealership
length	Continuous	Dropped	Length of the vehicle in inches
listed_date	Date	Dropped	The date the vehicle was listed on the website. Does not make days_on_market obsolete. The prices is days_on_market days after the listed date
listing_color	Categorical	Dropped	Dominant color group from the exterior color
listing_id	Unique Identifier	Dropped	Listing id from the website
longitude	Continuous	Dropped	Longitude from the geolocation of the dealership
main_picture_url	String	Dropped	
major_options	String	Dropped	
make_name	Categorical	Retained	
maximum_seating	Categorical	Partially Retained	
mileage	Continuous	Retained	
model_name	Categorical	Dropped	
owner_count	Continuous	Dropped	
power	String	Dropped	
price	Continuous	Target	
salvage	Boolean	Dropped	
savings_amount	Continuous	Dropped	
seller_rating	Continuous	Retained	
sp_id	Unique Identifier	Dropped	
sp_name	Categorical	Dropped	
theft_title	Categorical	Dropped	
torque	Continuous	Updated & Dropped	
transmission	Categorical	Partially Retained	
transmission_display	Categorical	Dropped	
trim_name	Categorical	Dropped	
trimId	Unique Identifier	Dropped	
vehicle_damage_category	Categorical	Dropped	
vin	Unique Identifier	Dropped	Vehicle Identification Number is a unique encoded string for every vehicle
wheel_system	Categorical	Partially Retained	
wheel_system_display	Categorical	Dropped	

wheelbase	Continuous	Dropped	
width	Continuous	Dropped	
year	Date	Dropped	
age	Continuous	Derived & Dropped	



Appendix Figure 1: Distribution of Categorical Variables



Appendix Figure 2: Distribution using Histogram for Categorical Variables

REFERENCES:

Mingxing Tan, Q. V. (2021). EfficientNetV2: Smaller Models and Faster Training. *Computer Vision and Pattern Recognition*.

Wayland, M. (7 March, 2023). *Used vehicle prices rising at an unseasonably strong rate*. Retrieved from CNBC: https://www.cnbc.com/2023/03/07/used-vehicle-prices-rising-at-an-unseasonably-strong-rate.html

Zhoubolei. (2016). *Learning Deep Features for Discriminative Localization*. Retrieved from Computer Vision and Pattern Recognition: https://github.com/zhoubolei/CAM

Zychlinski, S. (24 February, 2018). *The Search for Categorical Correlation*. Retrieved from Towards Data Science: https://towardsdatascience.com/the-search-for-categorical-correlation-a1cf7f1888c9