Candidate: Teh Tian Yan
Client: Datium Insights

Section A

The dataset consisted of 130 covariates and 62,192. 53 covariates have more than 20% of data missing (Figure 1). In addition, the covariate “MonthGroup” had 73.3% of its data valued as “0”, which is assumed as missing data given that the range is from 1 to 12 representing months.

Candidate: Teh Tian Yan
Client: Datium Insights

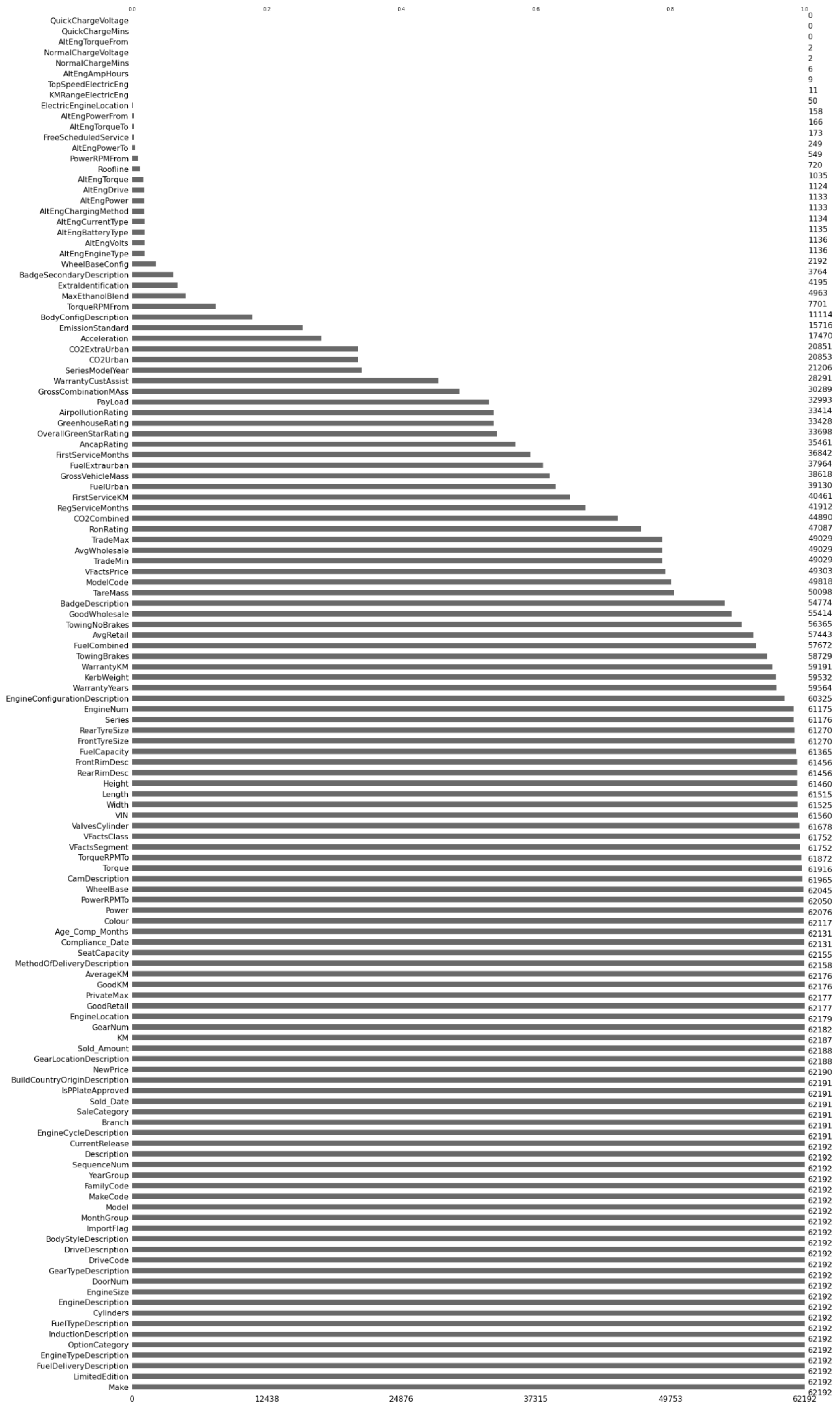


Figure 1: Missing data

Candidate: Teh Tian Yan
Client: Datium Insights

Certain categorical data were found to be correlated with each other. For instance, “”MakeCode”, “FamilyCode", "DriveCode", “GoodKM” are found to be collinear with “Make”, “Model”, “DriveDescription”, and “KM” respectively.

To address these concerns, covariates that have more than 20% missing data were dropped (Figure 2). This addresses the data sparsity. Further, it’s not practical to have a regression model designed for predictors that are typically not available.

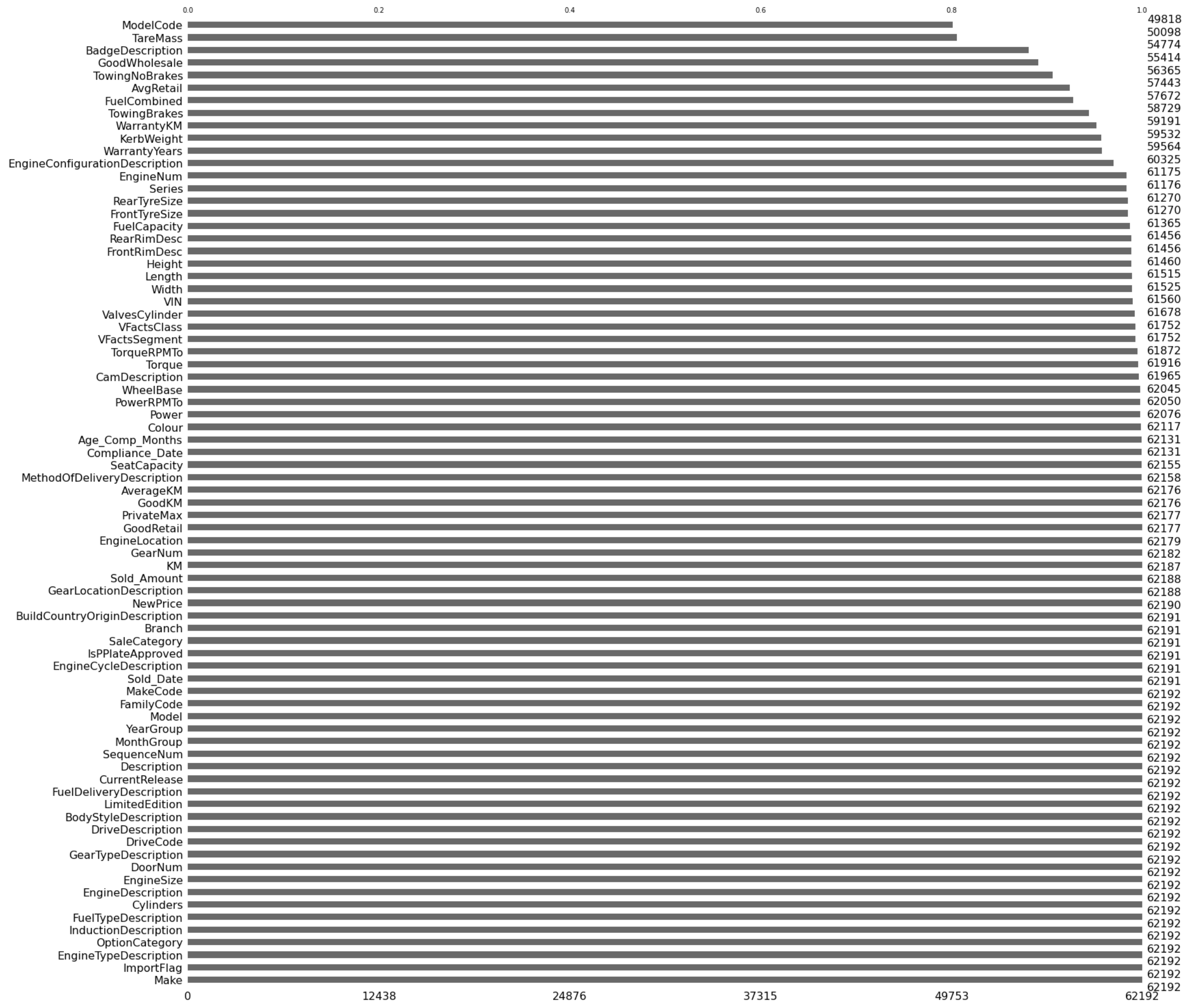


Figure 2: Missing data bar chart after dropping covariates with > 20% missing data.

There are 8 covariates that are collinear to the response variable, including “NewPrice” in addition to the 7 features which required removal (“AvgWholesale”, “AvgRetail”, “GoodWholesale”, “GoodRetail”, “TradeMin”, “TradeMax”, “PrivateMax”). This is to avoid data leaks of the response variable during the model training.

The missing data matrix revealed missing data in common rows. This suggests that the missing data may not be at random. Certain data points exhibit missing data across several columns consistently. A correlation plot of the missing data confirmed that missing data is heavily correlated to the absence of values in other covariates

(Figure 3). This is classified as missing not at random. In order to avoid modelling a biased estimate, a listwise deletion - or deletion of the entire record when there's absence of a single value - is done.

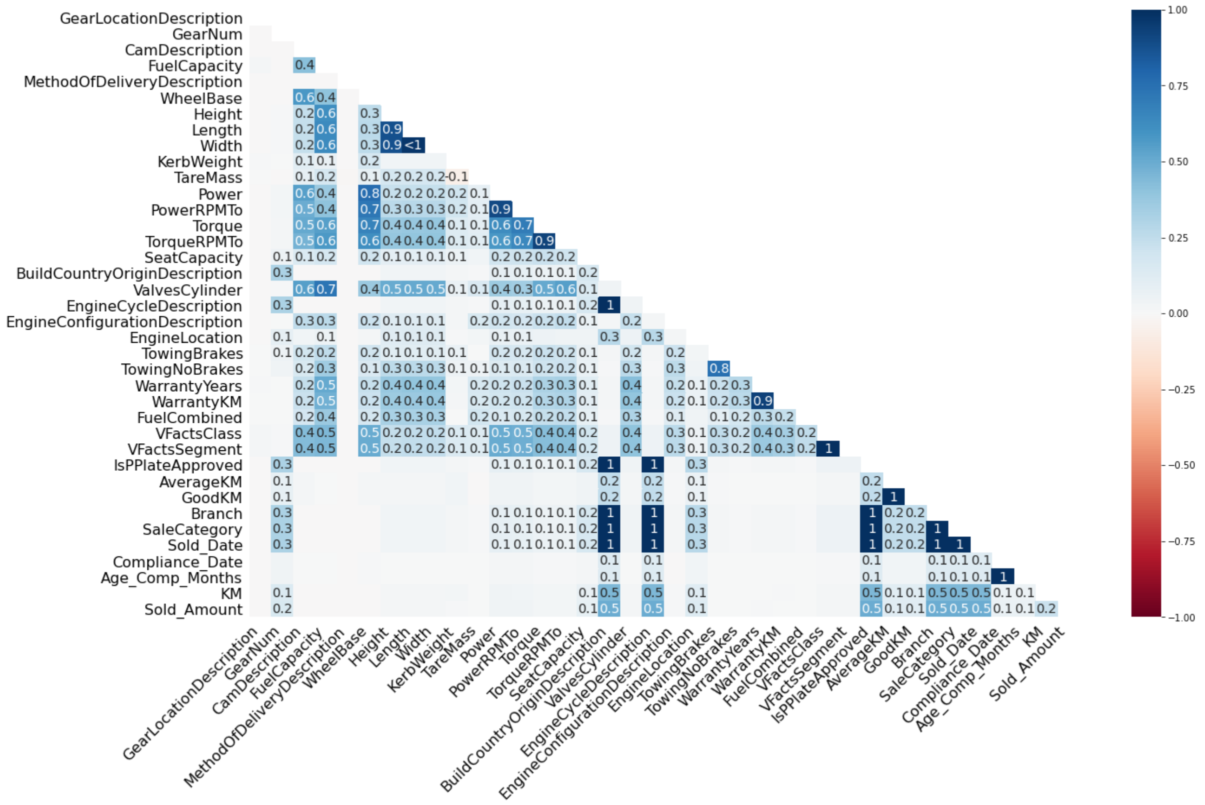


Figure 3: Missing data correlation plot.

Finally, the response variable "Sold_Amount" had 4 missing values. It's also noted that many are listed as "0" values which are assumed to be missing values just the same. These are removed as they serve no purpose in model training.

The distribution for the response variable "Sold_Amount" is right skewed (Figure 4).

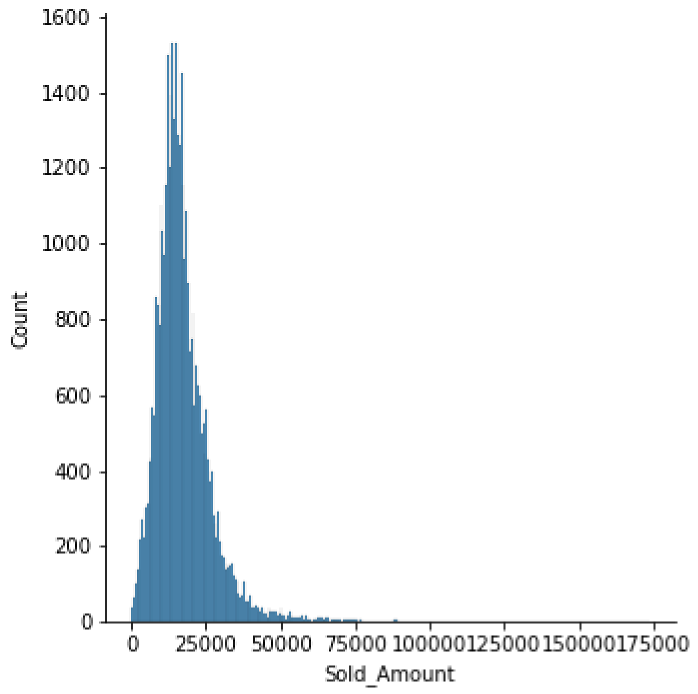


Figure 4: Response variable right skewed.

In order to unskew it, the log transformation is applied to the response variable (Figure 5).

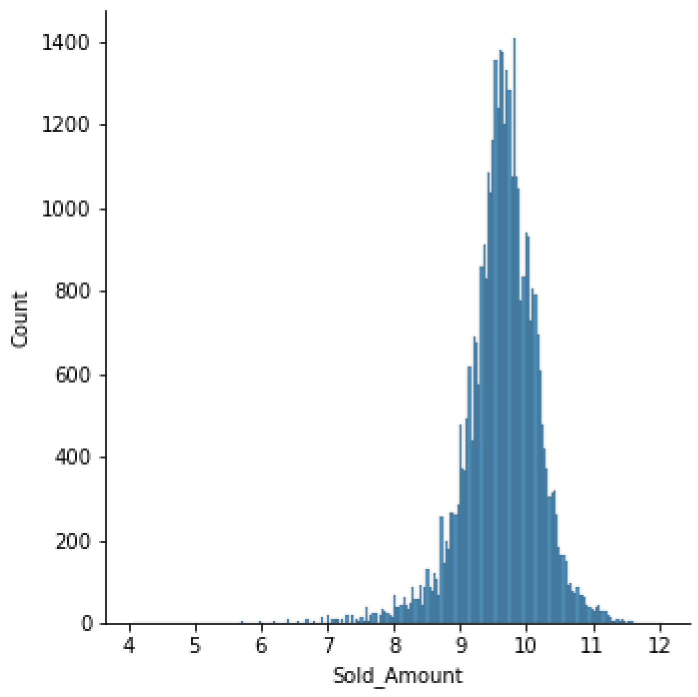


Figure 5: Log transformed response variable

Candidate: Teh Tian Yan
Client: Datium Insights

For feature selection, a Pearson's correlation plot (Figure 6) was used to examine the continuous covariates that had a high correlation to the response variable. At the same time, the collinearity with other covariates is examined as it would inflate the error metric when the model is fitted. The goal is to maximize the correlation to the response variable whilst eliminating as much collinearity between the covariates as much as possible for the feature extraction. The Variance Inflation Factor analysis (Figure 7) helps to detect collinearity. Typically any covariate with a VIF score > 5 should be removed to avoid multicollinearity.

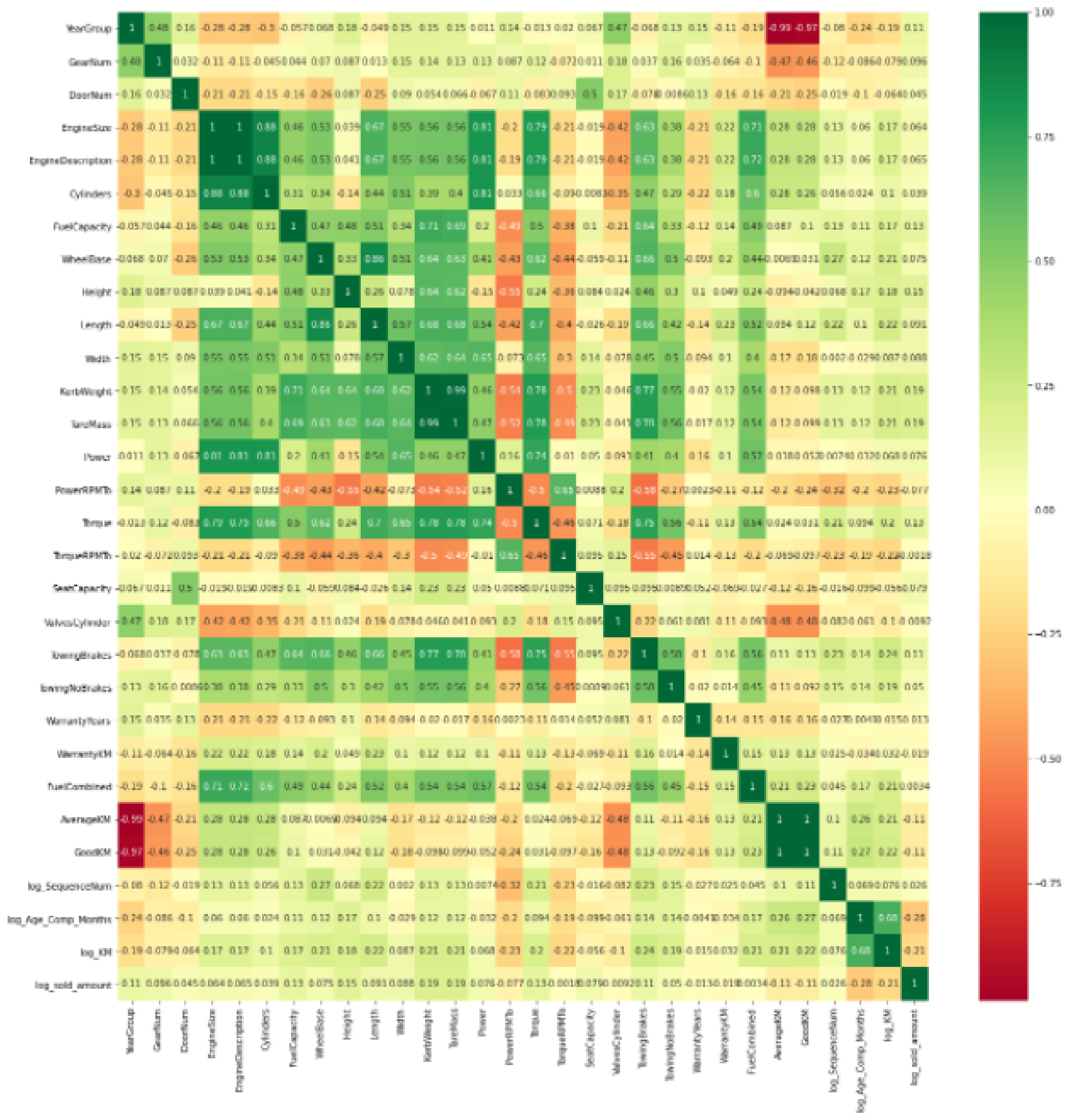
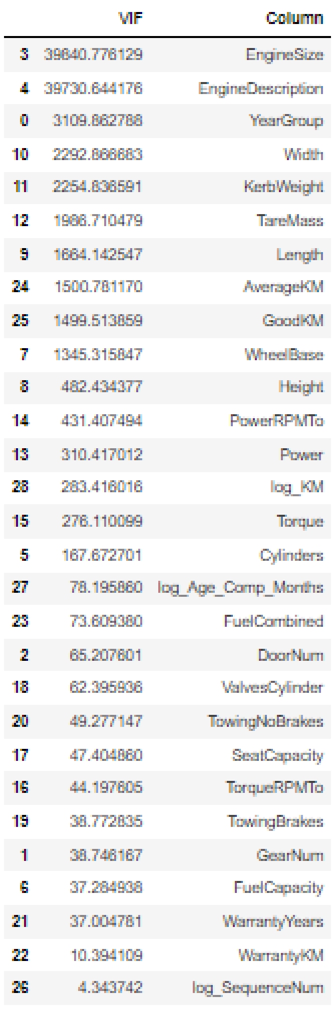


Figure 6: Pearson's Correlation Plot



	VIF	Column
3	39840.776129	EngineSize
4	39730.644176	EngineDescription
0	3109.862788	YearGroup
10	2292.866683	Width
11	2254.836591	KerbWeight
12	1986.710479	TareMass
9	1664.142547	Length
24	1500.781170	AverageKM
25	1499.513859	GoodKM
7	1345.315847	WheelBase
8	482.434377	Height
14	431.407494	PowerRPMTo
13	310.417012	Power
28	283.416016	log_KM
15	276.110099	Torque
5	167.672701	Cylinders
27	78.195860	log_Age_Comp_Months
23	73.609380	FuelCombined
2	65.207601	DoorNum
18	62.395936	ValvesCylinder
20	49.277147	TowingNoBrakes
17	47.404860	SeatCapacity
16	44.197605	TorqueRPMTo
19	38.772835	TowingBrakes
1	38.746167	GearNum
6	37.284938	FuelCapacity
21	37.004781	WarrantyYears
22	10.394109	WarrantyKM
26	4.343742	log_SequenceNum

Figure 7: Variation Inflation Factor

In this model fitting, I did not complete the feature engineering as it was an iterative process when selecting the variable. I attempted one-hot-encoding for the categorical variables but that resulted in the dataset becoming very huge and it slowed down the fitting process tremendously. An area of improvement to assist with the encoding would be to use Principal Component Analysis (PCA) given the size of the dataset, however, I was not able to implement it at this time. As a crude baseline for model fitting, I used all the continuous variables and omitted all the categorical variables.

The training and testing data was divided in a 75:25 ratio and was fitted to 4 regression models. Namely multiple linear regression, decision tree regressor, support vector regression, and random forest regressor. The metric used was a simple RMSE, in which, the multiple linear regression had the lowest RMSE of 1.03 and with an R-squared value of 0.987 (thereby accounting for majority of the data points).

Candidate: Teh Tian Yan
Client: Datium Insights

An area of improvement for this evaluation would be to create loss curves and to conduct hyperparameter tuning with a validation set using gridsearch.

Candidate: Teh Tian Yan
Client: Datium Insights

Section B

Goal: To build an algorithm to identify damaged areas of the vehicle.

R-CNN (Region-based Convolutional Neural Network) and YOLO (You Only Look Once) are two families of state-of-the-art object detection models used in car damage detection [1]. Both are based on CNN (Convolutional Neural Networks or ConvNets).

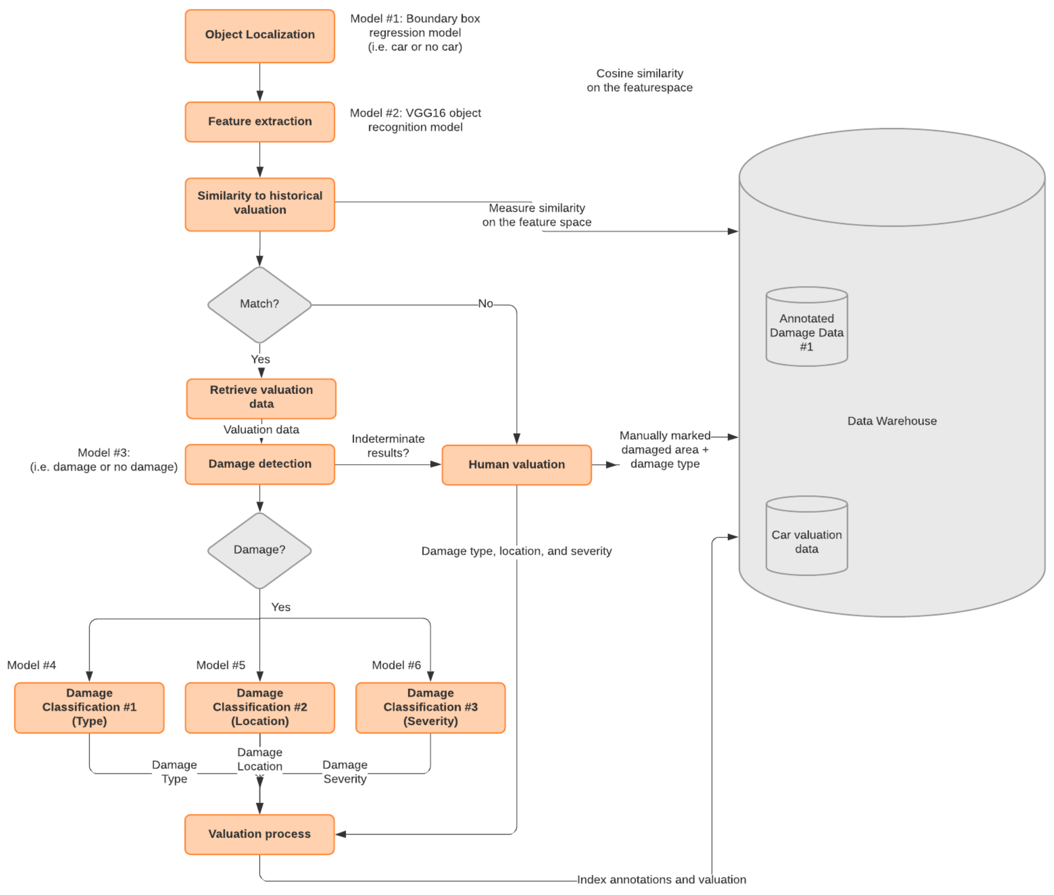
The R-CNN family of techniques include R-CNN, Fast R-CNN, Faster R-CNN and the state-of-the-art Masked R-CNN [2]. These are multistage models and are known for their accuracy.

YOLO, on the other hand, is a single-stage detector (as with SSD or Single-shot Multibox detection [3]) which determines the presence of an object, calculates the class probability, and regresses the boundary box coordinates in a single run. It’s variations include YOLOv1, YOLOv2, and YOLOv3. These are known for their ability to perform object detection in real time [4].

The preprocessing techniques involve zero-centering and scaling the images [5], the equivalent of normalizations. There are also mentions of the use of PCA (Principal Component Analysis), whitening, resizing, reduction of glare to improve the accuracy of damage detection. Not all images are fit for damage detection, notable exclusions from small damage claims include images with missing parts, total wreckages, and low quality / obstructed images.

The implementations of these algorithms are transfer learning options using pre-trained models or 3rd party implementations of pre-trained models (such as Keras). Transfer learning re-uses the weights from a general deep learning model (i.e. car classification) and applies it to a more specific problem (i.e. damage detection) [5]. This is a favored approach because deep-learning architectures are not a practical approach given the constraints on computational power and time. It requires a lot of training data with annotations, storage and computational resources, and multiple forward and backward propagations to train the correct weights. The trade-off in transfer learning is that fine-tuning is required to achieve a higher accuracy due to the fact that the pre-trained models were trained on more general problem cases. For instance, pre-trained models can achieve a 99.0% accuracy in recognizing cars in images [5] and this can be useful as a feature extraction method (e.g. VGG16 models). For the pre-trained models to be suitable for the specific problem case like damage detection, it has to be finetuned. The architecture of the pre-trained model could be reused where the weights are initialized randomly and the model trained on a damaged car dataset. Or, partial retraining is done where certain layers are frozen and others retrained [6].

In the context of damage detection done over an app, a single-stage detector architecture using YOLO would be appropriate given the real-time requirement of the problem. While the flowchart below depicts a multistage architecture for damage detection, it is only used for the sole purpose of illustrating the thought process of an architecture design. A single-stage damage detection architecture would only have a single model run instead of the 6 models depicted here.


Object Localization
Model #1: Boundary box regression model (i.e. car or no car)
Cosine similarity on the featurespace
Feature extraction
Model #2: VGG16 object recognition model
Similarity to historical valuation
Measure similarity on the feature space
Match?
No
Yes
Annotated Damage Data #1
Retrieve valuation data
Valuation data
Data Warehouse
Model #3: (i.e. damage or no damage)
Damage detection
Indeterminate results?
Human valuation
Manually marked damaged area + damage type
Damage?
Damage type, location, and severity
Car valuation data
Yes
Model #4
Model #5
Model #6
Damage Classification #1 (Type)
Damage Classification #2 (Location)
Damage Classification #3 (Severity)
Damage Type
Damage Location
Damage Severity
Valuation process
Index annotations and valuation

Proposed Damage Detection Architecture

Sources:

1. https://machinelearningmastery.com/object-recognition-with-deep-learning/
2. https://machinelearningmastery.com/how-to-perform-object-detection-in-photographs-with-mask-r-cnn-in-keras/
3. https://iq.opengenus.org/single-shot-detection-ssd-algorithm/
4. https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/

5. Jeffrey de Deijn, Automatic Car Damage Recognition using Convolutional Neural Networks
6. https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/
7. https://github.com/neokt/car-damage-detective/blob/master/neokt-car-damage-detective-121416.pdf
8. Pei Li, Bingyu Shen, Weishan Dong, An Anti-fraud System for Car Insurance Claim Based on Visual Evidence, https://arxiv.org/abs/1804.11207
9. Mahavir Dwivedi, Malik Hashmat Shadab, SN Omkar, Edgar Bosco Monis, Bharat Khanna, Satya Ranjan Samal, Ayush Tiwari, Aditya Rathi, Deep Learning Based Car Damage Classification and Detection, DOI: 10.13140/RG.2.2.18702.51525
10. Maria Raap, Vehicle Damage Detection using Semi-Supervised Object Detection
11. Kundjanasith Thonglek, Norawit Urailertprasert, Patchara Pattiyathanee, Chantana Chantrapornchai, Vehicle Part Damage Analysis Platform for Autoinsurance Application, DOI: 10.37936/ecti-cit.2021153.223151