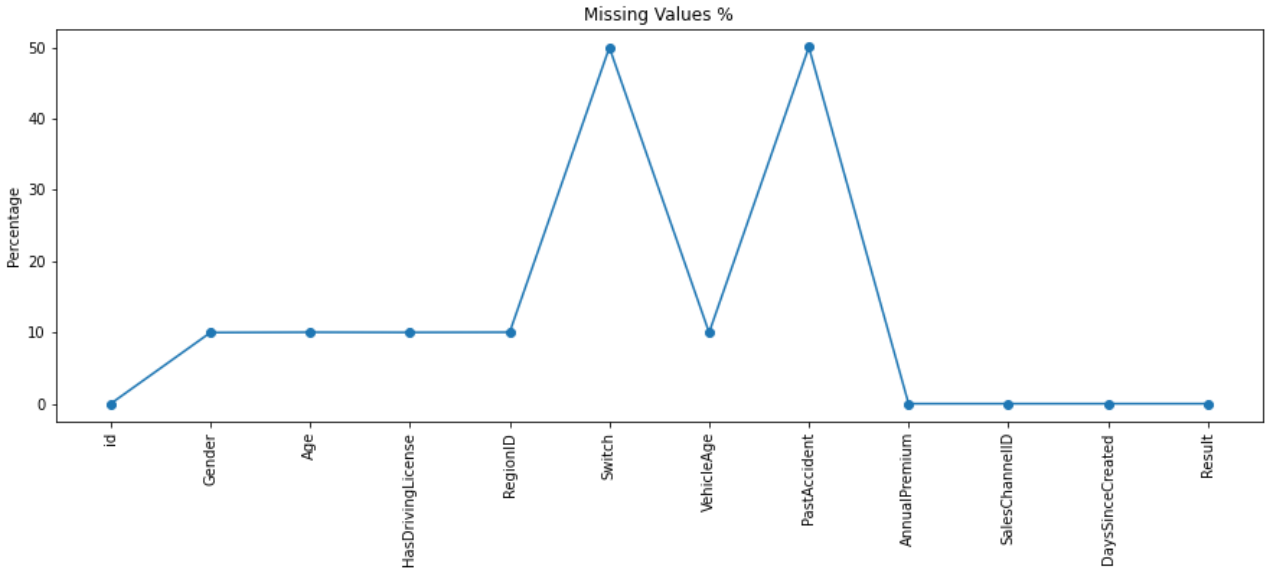
CMT 07
Applied Machine Learning
Coursework 1

Machine Learning Model for Prediction of Customer Purchase of Car Insurance using Data from Health Insurance Provider

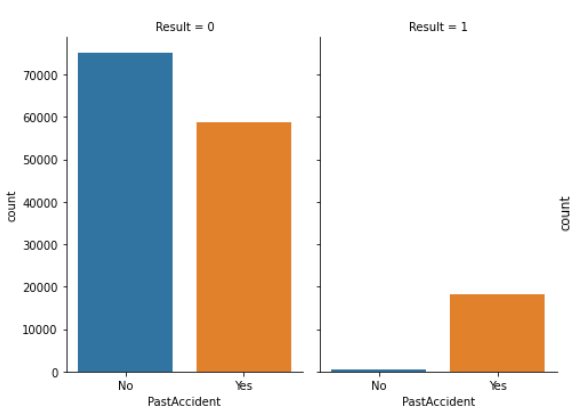
Student Number: C22086815

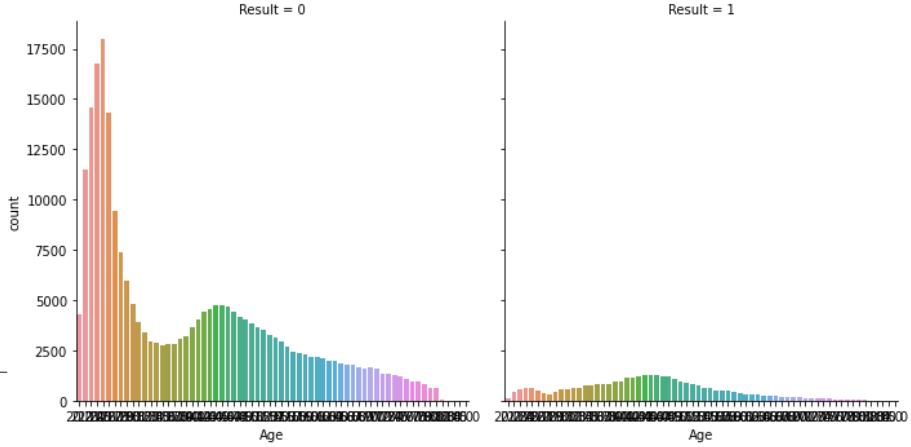
1. Data Exploration

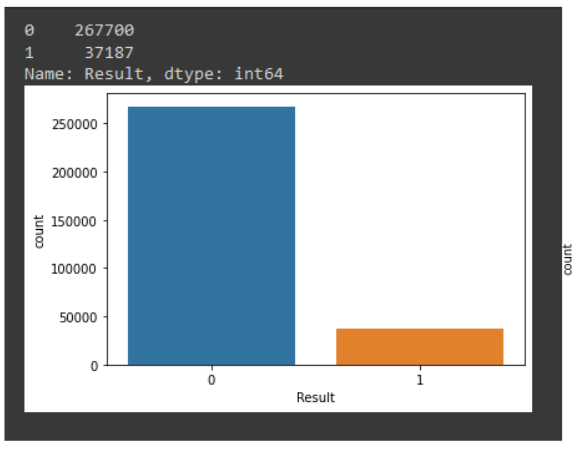
The dataframe has 304887 data points in total. Most variables have 10% missing values or less. **Switch and Past accident** have over 50% missing variables. Identifiable information such as ID is dropped from the data. The variable types clearly have some anomalies and need to be modified as follows to obtain more useful insights with data exploration. AnnualPremium was changed to a float data type. None of the numerical data follow a normal distribution, and this means that we can stick to **MinMaxScaler** in pre-processing.

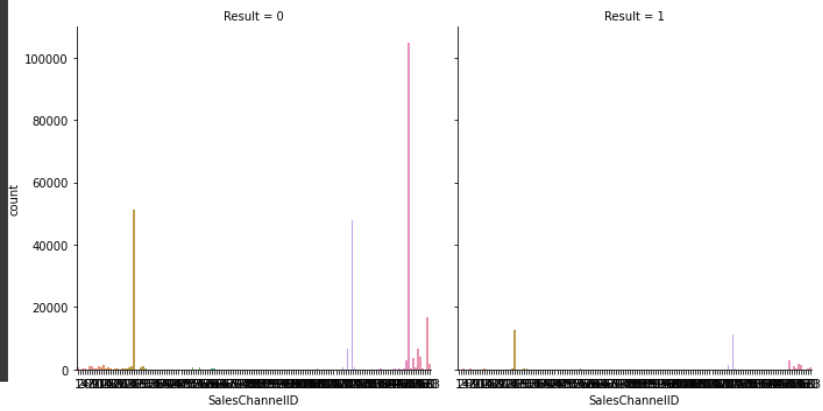


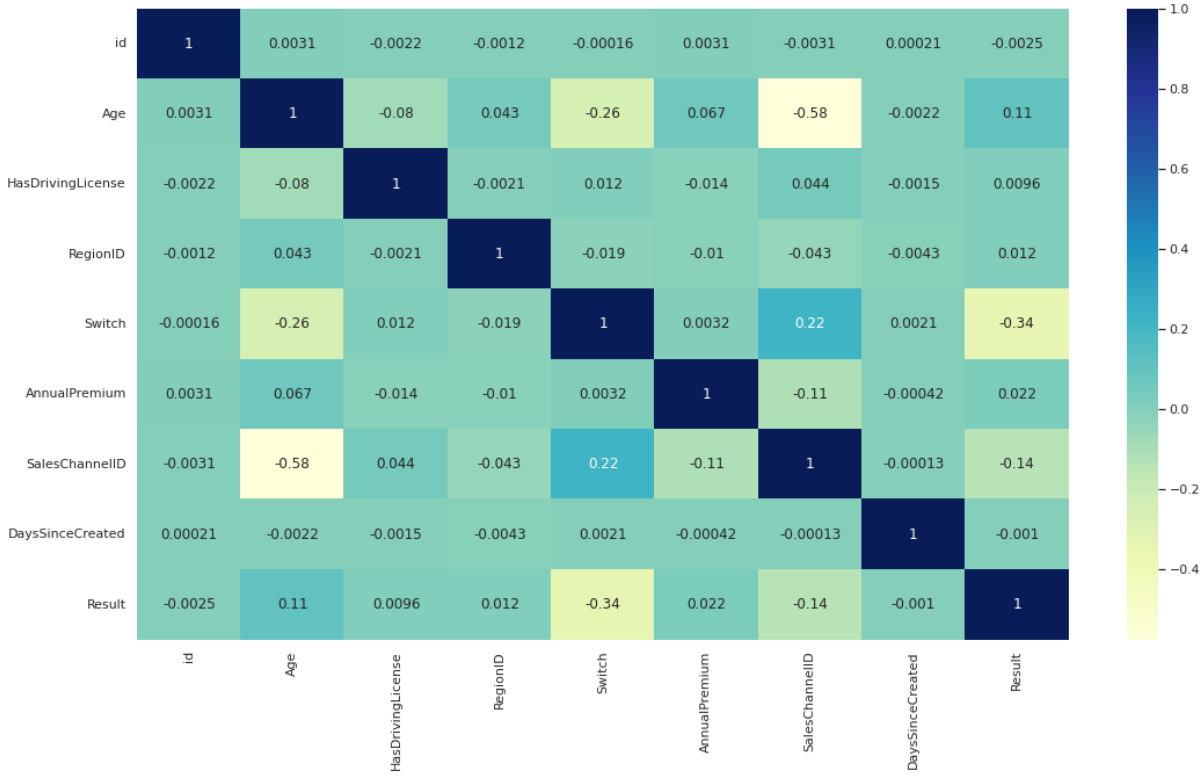
Results: The data is very imbalanced to being with. This has to be corrected before we start training any model.
Past Accident: Expectedly, people that have had car accidents previously are more likely to get car insurance.
Gender: The distribution of Result is pretty homogeneous across Male and Female gender.
Age : The proportion of younger drivers within the set of people without car insurance is expectedly higher.
SalesChannelID: Some of the Sales Channels perform poorly in comparison to the rest, and need to be revised.











2. Data Pre-processing

Thinking ahead, we shall perform different types of pre-processing based on the models that we are most likely to use. This is due to the fact that different classifiers deal with certain issues such as duplicates, missing values or with different levels of success. Many of the changes in this section were made after testing different versions of pre-processed data against the chosen classifier models, eg dropping **Switch**. They are summarised in the table below:

Feature	Pre-processing steps
id	Dropped
PastAccident	Data instances with missing values dropped
Age	Missing values imputed with median, MinMaxScaled between 0 and 1
Gender	Missing values imputed with 'most frequent', encoded into 1 binary column
HasDrivingLicense	Missing values imputed with 'most frequent', one hot encoded in 1 binary columns
RegionID	One hot encoded into N-1 binary columns
SalesChannelID	One hot encoded into N-1 binary columns
Switch	Data instances with missing values dropped
VehicleAge	Missing values imputed with 'most frequent', one hot encoded in 2 binary columns
AnnualPremium	MinMaxScaled between 0 and 1
DaysSinceCreated	MinMaxScaled between 0 and 1

The data is now ready to be used by most Machine Learning Classifier models. We first need to split the data into Training and Test sets. We note that the data is quite imbalanced, and we need to use methods such as undersampling, oversampling or to correct for it. We opt for oversampling the minority group with SMOTE.

```
y.value_counts()
```

```
0    133796
1     18626
Name: Result, dtype: int64
```

190	191	192	193	194	195	196	197	198	Age	HasDrivingLicense	AnnualPremium	DaysSinceCreated	Result
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.430769	1.0	0.088759	0.615917	0
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.000000	1.0	0.083113	0.726644	0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.046154	1.0	0.000000	0.529412	0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.876923	1.0	0.000000	0.944637	0
0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.215385	1.0	0.000000	0.792388	0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.369231	1.0	0.040053	0.508651	1

3. Model Implementation

For classification purposes, we opt for the following models: **Logistic Regression**, **Decision Tree** and **Ensemble**. There were several back-and-forth's between pre-processing to create new train-test splits, which resulted in numerous attempts in optimising the model.

1. Logistic Regression

For a new Machine Learning Initiate, Logistic Regression is one the most elementary methods for Classification. It pertains strongly to material recently covered in MAT022, .We create a logistic regression model with parameters fit purely from the training data. The performance is increased upon using GridSearchCV to optimise the hyperparameters of the model.

2. Decision Tree

A decision tree model is implemented for this classifier. RandomizedSearchCV with RepeatedStratifiedKfold is used to optimize the hyperparameters of the model. It is noted that a better performance is achieved in this case using the regular imbalanced training dataset before SMOTE; therefore we preserve it for this model.

3. Ensemble Method

Here, we tested as many classifier models in 1 go, testing out their performance. Only the top models were kept in the final code, for the sake of efficiency especially in the total training time for this method. Hyper optimising parameters for such models takes very long, and thus was not necessarily repeated after each trial of modifications made in the pre-processing due to run time constraints and efficiency.

4. Performance and Evaluation

Some additional modifications for the dataset used for each model were made in an effort to improve the metrics presented below, such as using a different starting dataframe, or completely kick some variable columns from the model. For a classifier model, the ideal metric to measure performance is a confusion matrix. As this is a model based on the decision of buying insurance package, False Negative and True Positive are important values as they give an indication of the minimum projected revenue for the company.

5. Results Analysis and Discussion

The first metric for the 2 models below is the accuracy of each model accessed with **classification_report**.

Model 1

This model remains somewhat poor, especially when it comes to predicting data with Result '1'. This demonstrates that there is an element of luck in the model rather than a proper fit, given that the number of Result '0' data instances happens to be high.

```
0.6838629256238109
              precision    recall  f1-score   support

           0       0.98      0.65      0.78     40072
           1       0.27      0.91      0.42      5655

    accuracy                           0.68     45727
   macro avg       0.63      0.78      0.60     45727
weighted avg       0.89      0.68      0.74     45727
```

Model 2

The same issue can be noted here. The quality of the metrics degrades rapidly when True Result is 1. The overall accuracy has improved but it is not satisfactory. Recall and F1-score in particular being low shows that the model is not reliable to predict income from customer purchase.

```
0.8761563190237716
              precision    recall  f1-score   support

           0       0.88      1.00      0.93     40072
           1       0.47      0.01      0.02      5655

    accuracy                           0.88     45727
   macro avg       0.67      0.50      0.48     45727
weighted avg       0.83      0.88      0.82     45727
```

Model 3

	Model	Accuracy Score Train	ROC-AUC Score Train	f1-score Train	Accuracy Score Test	ROC-AUC Score Test
0	Logistic	0.791863	0.841913	0.791863	0.687945	0.831552
1	KNN	0.898809	0.971919	0.898809	0.760586	0.772325
2	Bagging	0.992372	0.999826	0.992372	0.841178	0.752766
3	Gradient Boost	0.830243	0.915446	0.830243	0.722547	0.830791
4	XGB	0.829213	0.913759	0.829213	0.721248	0.831060
5	SVM	0.812501	0.881311	0.812501	0.695582	0.817549

Based on training with the data set with these base models, hyperparameter optimising is performed for Bagging, with RandomisedSearchCV. Due to time constraints, further optimisation was not feasible, and the last recorded results is as follows using a few versions of pre-processed data are:

0.6318211702827088

	precision	recall	f1-score	support
0	0.99	0.59	0.74	20128
1	0.24	0.98	0.38	2687
accuracy			0.63	22815
macro avg	0.62	0.78	0.56	22815
weighted avg	0.91	0.63	0.70	22815

0.8761563190237716

	precision	recall	f1-score	support
0	0.88	1.00	0.93	40072
1	0.47	0.01	0.02	5655
accuracy			0.88	45727
macro avg	0.67	0.50	0.48	45727
weighted avg	0.83	0.88	0.82	45727

Despite numerous optimisation attempts for the hyper parameters and compromising between the performance metrics for all the models, the performance does not improve considerably. This shows that the faultiness in this model lies in the pre-processing area. The biggest issue was dealing with Missing Values. Alternate methods of dealing with the features with many categories can be considered, such as binning. RegionID and SalesChannelID were tackled in various manners during previous trials, given that they are categorical features with large number of categories. Many of the features were straightforward to pre-process with direct scaling to [0,1] and OneHotEncoding for categoricals. But the number of features increases drastically with OneHotEncoding, which makes the training time very arduous for all classifiers, for both training and the hyperparameter optimisation. Methods of Classification were picked on familiarity developed from previous experience and from the material covered in class, where we had some understanding of the methods instead of trying too hard to completely emulate new things from the internet.