#C22107414

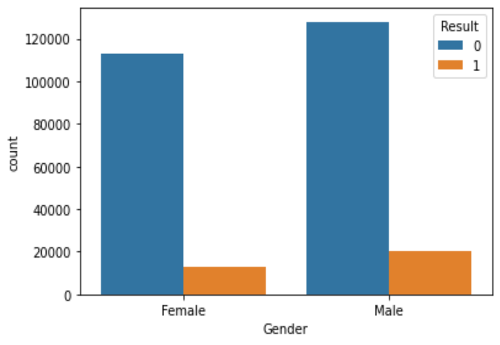
REPORT

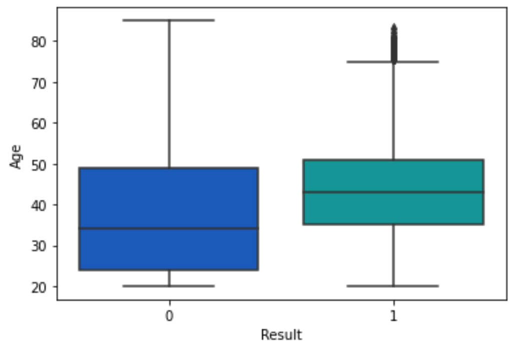
1. Dataset Description & Preparation

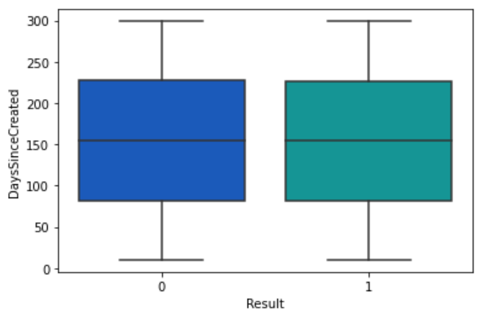
The dataset contained data about 304887 customers (no duplicates were detected). The target variable was whether a customer would buy car insurance (Result). The dataset was highly imbalanced in regard to the target variable with only 12.2% interested in buying car insurance.

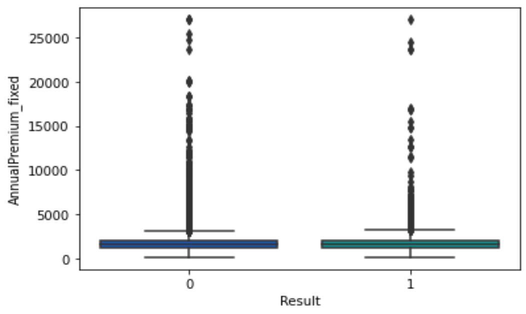
2. Exploratory Data Analysis

Gender	Did not seem to play a significant role whether someone is interested in buying car insurance or not (the proportions are fairly similar – see the figure below)
Having a driving license	It is a useless feature because only 0.2% of clients did not have a driving licence
Number of days since created	The mean was the same for both groups so most likely this feature will not significantly contribute to model performance and do not have a lot of predictive power (see the figure below)
Annual Premium	The mean was the same for both groups so most likely this feature will not significantly contribute to model performance and do not have a lot of predictive power (see the figure below)
Age	The mean for those who are interested is higher, which means that older customers are more likely to be interested in the purchase
Important observation →	The two most important features were past accidents and switch. We can see below that only those who did not switch were interested in car insurance, and almost everyone who had a past accident was interested









3. Imputing missing values, Pre-processing and Sampling

For the numerical variables we chose the KNN imputer, whereas for the categorical we chose to impute missing values with the most frequent value. For the two variables with 50% missing data, this was not an ideal solution and likely led to a significant number of wrong imputations. We then standardised the numerical features using scikit-learn's Standard Scaler[1] and encoded categorical variables using one-hot encoding.

Using the whole, pre-processed dataset we set a baseline using a Support Vector Machine (SVM) using 80% of the data for training and 20% as a test set. The results show that the model is struggling with the minority class with an F1 score of only 0.258 which leads to a macro average F1 score of 0.570 on the test set.

```
              precision    recall  f1-score   support

           0      0.898     0.865     0.881     53541
           1      0.232     0.292     0.258      7437

    accuracy                          0.795     60978
   macro avg      0.565     0.579     0.570     60978
weighted avg      0.817     0.795     0.805     60978
```

Hence, we tried three sampling methods to balance the dataset and one additional approach where we did not impute missing values but dropped them and set 4 more baselines. We tried (1) undersampling the majority class, (2) oversampling the minority class, and (3) SMOTE[2]. Oversampling and undersampling produced almost identical results, whereas SMOTE did not perform as good compared to the other two approaches. The results for undersampling (left) and SMOTE (right) are shown below.

```
              precision    recall  f1-score   support

           0      0.664     0.636     0.650      7359
           1      0.658     0.686     0.671      7516

    accuracy                          0.661     14875
   macro avg      0.661     0.661     0.661     14875
weighted avg      0.661     0.661     0.661     14875
```

```
              precision    recall  f1-score   support

           0      0.380     0.241     0.295      9981
           1      0.446     0.607     0.514     10019

    accuracy                          0.425     20000
   macro avg      0.413     0.424     0.404     20000
weighted avg      0.413     0.425     0.405     20000
```

We decided to proceed using undersampling given it resulted in a smaller dataset and hence faster model training. Finally, given the problem mentioned above regarding the high probability of wrong imputations, we created a dataset where all missing values were dropped, resulting in 85% of the data being lost. The data was still very imbalanced, so we used undersampling again. The performance was significantly higher and confirmed our concern about wrong imputations.

```
              precision    recall  f1-score   support

           0      0.929     0.660     0.771      1072
           1      0.742     0.951     0.834      1105

    accuracy                          0.808      2177
   macro avg      0.836     0.805     0.803      2177
weighted avg      0.834     0.808     0.803      2177
```

We proceeded with 2 approaches: the undersampled with imputed values and the undersampled with no imputed values. All variables were used apart from the customer ID and driving licence.

4. Model Selection using undersampled with imputed values

The test set was set aside and model choice and hyperparameter tuning was done via 5-fold cross validation on the training set. We compared KNN, decision tree, multilayer perceptron (MLP), logistic regression, random first and AdaBoost to our SVM baseline which had a weighted F1 score of 0.632 on the training set. The maximum depth of the decision tree was set to 10 to avoid overfitting, and the hidden layer neuron size of the MLP was set to to (compared to the default 100). For the other classifiers scikit-learn's default parameters were used.

Classifier	*Weighted F1 Score*
KNN	0.721
Decision Tree	0.752
MLP	0.759
Logistic Regression	0.757
Random Forest	0.736
AdaBoost	0.757

All classifiers outperformed our SVM baseline but most of the models showed similar performance. To cover several different models, we proceeded with a linear method (logistic regression), a tree-based method (decision tree) and an ensemble method (AdaBoost), which we found interesting to compare to a decision tree given the base estimator for AdaBoost is a decision tree classifier with an initialised maximum depth of 1.

5. Feature Selection

Given the observations discussed in Section 2 and the small number of features, we performed feature selection manually using a logistic regression model. We first deleted features which we believed had no impact on the model like gender, days since created, and annual premium. As we presumed, the F1 score remained unchanged when using 5-fold cross validation (0.755). Another two features which did not lead to a drop in the F1 score was the region and the age of the vehicle. Features which led to a drop in performance were the sales channel (0.73), switch (0.73) and past accidents (0.729). Age and vehicle age led to a slight drop (to 0.754 and 0.75 respectively), but we decided to keep them too. The decision tree classifier and Adaboost both scored 0.757 when trained on those five features.

6. Hyperparameter Tuning

We performed a grid search on the hyperparameters outlined in the table below for each model. The last column shows the ones that scored the best.

Model	Hyperparameters	Final Choice
Logistic Regression	C, solver	C: 0,1, solver: newton-cg
Decision Tree	Maximum depth, criterion	Depth: 10, criterion: entropy
AdaBoost	Number of estimations, algorithm	Number of estimators: 150, algorithm: SMME.R

7. Model validation on Test Set

When testing our final models, using only the five features described above, and the hyperparameters which produced the best results, we achieved the following results on the test set:

Logistic Regression

```
              precision    recall  f1-score   support

           0      0.848     0.628     0.722      7359
           1      0.710     0.890     0.790      7516

    accuracy                          0.760     14875
   macro avg      0.779     0.759     0.756     14875
weighted avg      0.778     0.760     0.756     14875
```

Decision Tree

```
              precision    recall  f1-score   support

           0      0.827     0.651     0.728      7359
           1      0.717     0.867     0.785      7516

    accuracy                          0.760     14875
   macro avg      0.772     0.759     0.757     14875
weighted avg      0.771     0.760     0.757     14875
```

AdaBoost

```
              precision    recall  f1-score   support

           0      0.830     0.655     0.732      7359
           1      0.720     0.868     0.787      7516

    accuracy                          0.763     14875
   macro avg      0.775     0.762     0.760     14875
 eighted avg      0.774     0.763     0.760     14875
```

One can see that all models achieved similar results, but AdaBoost overperformed both having the highest macro and averaged F1 score.

8. AdaBoost trained on undersampled data without imputed values

Finally, we wanted to use the best model (in our case AdaBoost) on the dataset where no data was imputed. Given we knew the most important features, we only selected those five from the raw data, and dropped the observations which contained a missing value, leaving us with 20% of the data, 15% of which were used as a test set. We then used undersampling to balance the dataset and searched for the best hyperparameters for AdaBoost, which were the SAMME algorithm and 100 estimators, using 5-fold cross validation on the training set. Our final results on the test set are shown below:

```
              precision    recall  f1-score   support

           0      0.922     0.639     0.754      1090
           1      0.738     0.949     0.830      1166

    accuracy                          0.799      2256
   macro avg      0.830     0.794     0.792      2256
weighted avg      0.827     0.799     0.794      2256
```

9. Discussion

By dropping all missing values and not imputing them and using only a small balanced dataset of 15k observations we could significantly increase the performance of our baseline model which had a macro F1 score of only 0.570. The reasons being (1) the imbalance and (2) that the two main predictors contained most of the missing values. However, the problem for the future remains: the new data that the trained model would be used on, will most likely also be imbalanced and it is questionable how well the model, which was trained on balanced data, would perform on real world data. But it is important for the health insurer to obtain switch and past accidents data, given these are the most important features.

10. References

[1] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[2] https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html