Name: Eshaq Rahmani | Student Number: 22086790

Data exploration:

Plots:

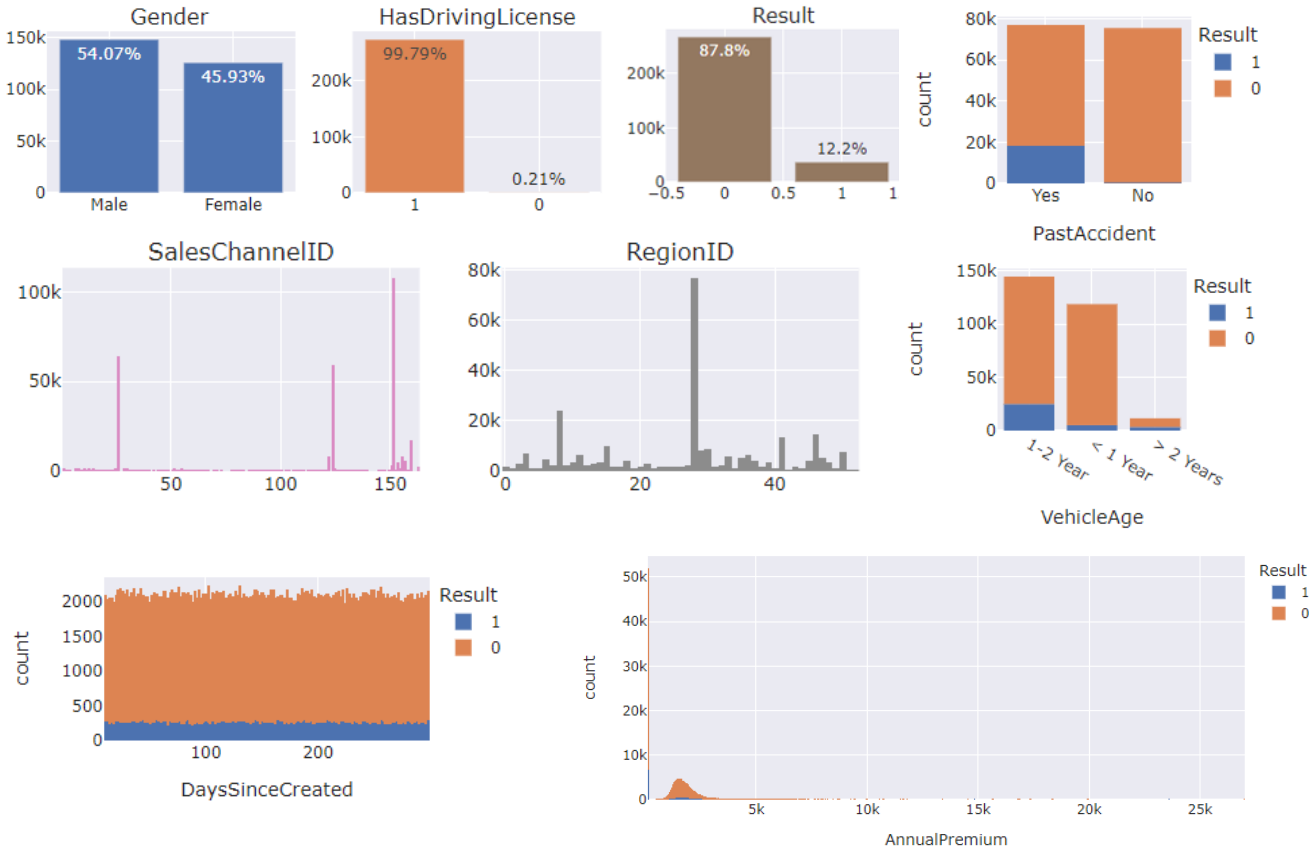
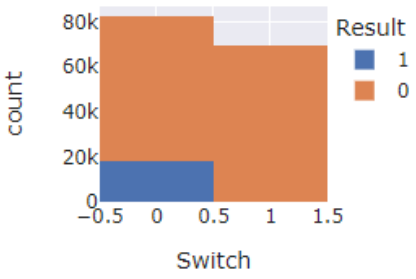


Figure 1. Histogram plots for feature and target variable analysis



Missing Values:

Table 1. Features with missing values

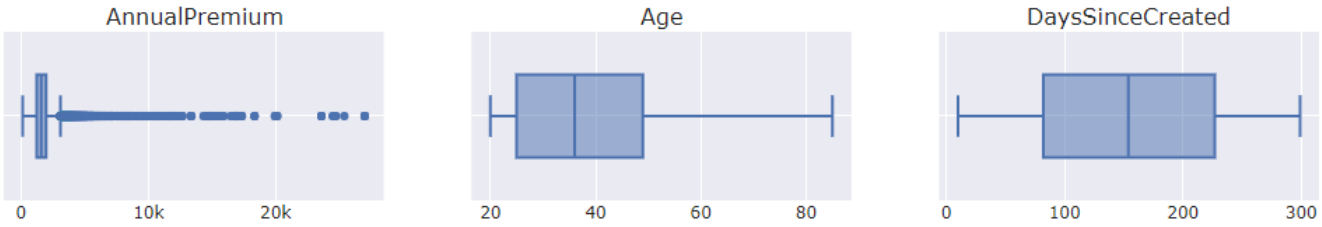
Feature	Missing Data
PastAccident	152465
Switch	152330
Gender	30427
Age	30548
HasDrivingLicense	30488
RegionID	30553
VehicleAge	30441

Analysis:

- The "Gender" feature has a higher percentage of "Male" values, which may influence missing data imputation.

- The "HasDrivingLicense" feature has mostly 1 (positive) values, with 99.79% being 1, indicating that it may be dropped.
- "DaySinceCreated" has no relationship with "Result" and can be dropped.
- "PastAccident" and "Result" show a positive correlation, and "Switch" and "Result" show a negative correlation.
- "VehicleAge" data for cars 1-2 years old and under 1 year is similar, but 1-2 year old cars are more likely to want insurance. This indicates positive correlation with VehicleAge and Result.
- The features “AnnualPremium”, “Age”, ”SalesChannelID”, and “RegionID” have significantly different scales compared to the other features. This may need scaling.
- The "Result" variable is imbalanced (87.8% 0) and missing data (particularly for "PastAccident" and "Switch") must be addressed before training a model.
- Missing data (particularly for "PastAccident" and "Switch") must be addressed before training a model.

Box Plots showing outliers:



Analysis: Only AnnualPremium feature has outliers which might need to be addressed in the pre-processing step.

Data pre-processing:

As the data processing will not The data pre-processing steps were carried out in the following order:

1. Cleaning data

We clean the AnnualPremium column by removing the currency symbol.

2. Dealing with Missing Data

We imputed missing values in 'Switch' and 'PastAccident' by creating a separate category ('Missing') based on the preservation of the original distribution and performance. The remaining column's missing value rows were imputed with mode and forward fill. We compared this method to KNN and iterative imputation (Bayesian estimator) based on performance, and selected iterative imputation due to its best results.

3. Outlier Removal and Feature Scaling

We used the IQR method to remove outliers from the AnnualPremium column. To reduce training time and improve model performance, we scaled “AnnualPremium” and “Age”.

4. Encoding Features

We used one-hot encoding for the categorical variables "Gender," "Switch," and "PastAccident" to reduce model complexity. We used ordinal encoding for the "VehicleAge" variable because of its hierarchical nature.

5. Feature Selection – Chi score (categorical) and Pearson Correlation (numerical)

We used the chi-square test to evaluate the association between each categorical feature and the target. The "DaySinceCreated" feature had the lowest score and was therefore dropped. Pearson correlation for "DaySinceCreated" verified this analysis. The "HasDrivingLicense" feature had the next lowest score, but the p-value was not significant for a 0.05 alpha score. However, we decided to drop this feature because it had all identical values. We also dropped the "id" feature because it was not useful for modelling.

Table 2. Chi square score for feature selection

Feature	Chi Score	p-value
DaySinceCreated	14.53	0.0
HasDrivingLicense	0.05	0.83

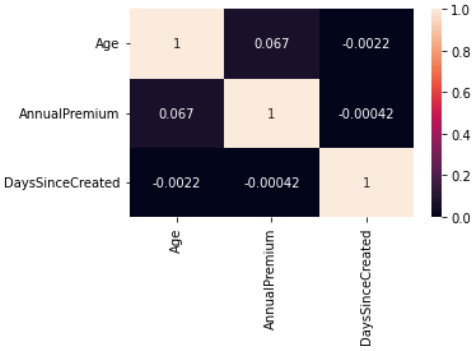


Figure 2. Pearson correlation for numerical features

6. Handle imbalanced data

To address the issue of imbalance in our dataset, we compared 3 different techniques: SMOTEENN, SMOTENC and hyperparameter class weights. We selected the method best suitable for each model based on performance. See “Model Implementation” section for performance values.

Model implementation

Three suitable models chosen are: XGBoost, Random Forest and KNN.

XGBoost

Justification: XGBoost is a powerful tree-based ensemble machine learning algorithm with a history of strong prediction accuracy and performance. XGBoost tends to perform well when there is a combination of categorical and numerical features in the data, making it well-suited for this type of dataset. Additionally, the algorithm's ability to accurately classify a single target variable makes it a strong choice for this type of classification problem.

Hyperparameter Optimization is done using **RANDOMIZEDSEARCHCV**

best hyperparameter grid for the given dataset:

{subsample=1.0, reg_lambda=0.1, reg_alpha=1, max_depth=7, learning_rate=0.1, colsample_bytree=0.8, n_estimators=3, scale_pos_weight = ratio}

KNN

Justification: KNN is a distance-based algorithm useful for classification tasks with non-linear feature relationships, particularly when the target classes are well-separated and there are a small number of features. Quality of predictions depends on the distance measure, so domain knowledge is important to select an appropriate measure. We will standardize the input variables to improve performance and convergence rate.

Hyperparameter Optimization is done using **RANDOMIZEDSEARCHCV**

best hyperparameter grid for the given dataset: `{'n_neighbors': 3}`

Random Forest

Justification: The algorithm's ability to handle a mixture of categorical and numeric features makes it well-suited for this type of data. Its ability to detect non-linear relationships can be particularly useful for complex datasets, and its low-bias nature means it is less likely to overfit the data. Overall, these characteristics make Random Forest a strong choice for classifying a structured dataset with a single target variable.

Hyperparameter Optimization is done using **RANDOMIZEDSEARCHCV**

best hyperparameter grid for the given dataset:

```
{'n_estimators': 10, 'min_samples_split': 3, 'min_samples_leaf': 6, 'max_features': 2, 'max_depth':
None}
```

Performance evaluation

We evaluate the top three models based on PR-AUC and ROC-AUC, as these metrics are sensitive to imbalanced target variables, which is the case with our dataset. The goal is to maximize these metrics. We also use K-Fold Cross Validation on training data and partition it into 5 folds. The mean ROC_AUC across all folds is then calculated.

1. XGBoost - scale_pos_weight model

Cross validating training data: ROC_AUC mean score: 0.84.

A high degree of accuracy on the training data achieved, even considering we used the imbalanced training data (due to the use of scale_pos_weight).

Performance values:

Table 3. Performance values for XGBoost

Balancing Method	PR-AUC	ROC -AUC	Cohen's Kappa score
SMOTEEN	0.587	0.772	0.259
SMOTENC	0.587	0.772	0.258
scale_pos_weight	0.600	0.775	0.248

Results: As can be observed from Table 3, the scale_pos_weight parameter of XGBoost handled the imbalance data the best compared to standard oversampling methods. A Cohen's Kappa score of 0.248 show a moderate level of agreement.

2. Random Forest Classifier – SMOTEEN model

Cross validating training data: ROC_AUC mean score: 0.84. A high degree of accuracy on the training data achieved.

Performance values:

Table 4. Performance values for Random Forest Classifier

Balancing Method	PR-AUC	ROC -AUC	Cohen's Kappa score
SMOTEEN	0.559	0.763	0.288
SMOTENC	0.497	0.721	0.296
class_weight='balanced'	0.534	0.748	0.297

Results: For this model the SMOTEEN method achieved highest AUC values, and seems to work best for this particular data set and model. A Cohen's Kappa score of 0.288 show a moderate level of agreement.

3. KNN – SMOTEEN with k = 3 model

Cross validating training data: ROC_AUC mean score: 0.90. A very high degree of accuracy on the training data achieved.

Performance values:

Table 5. Performance values for Random Forest Classifier

Balancing Method	PR-AUC (K=3, K =1)	ROC -AUC (K=3, K =1)	Cohen's Kappa score (K=3, K =1)
SMOTEEN	(0.502, 0.490)	(0.718, 0.711)	(0.256, 0.257)
SMOTENC	(0.419 ,0.378)	(0.660 ,0.631)	(0.234, 0.228)

Results: KNN performed best with K=3 and using SMOTEEN. We compare it with baseline K=1 which achieved worse performance results.

Result analysis and discussion

XGBoost had the best performance, with a PR-AUC score of 0.600 and a ROC-AUC score of 0.775. This suggests that the model moderately well distinguished between the positive and negative classes and had moderately good precision and recall.

Random Forest Classifier performed next, with a PR-AUC score of 0.559 and a ROC-AUC score of 0.763.

KNN had the lowest performance, with a PR-AUC score of 0.502 and a ROC-AUC score of 0.718.

Overall, our results suggest that the XGBoost model is the best choice for this task, as it had the highest PR-AUC and ROC-AUC scores. The Random Forest Classifier was also a strong performer, and could be considered as an alternative. On the other hand, the KNN model did not perform as well and may not be the best choice for this task.

It is important to note that the hyperparameters did not change across the different balancing methods, which may have influenced the results. It is essential to carefully consider the hyperparameter tuning process and ensure fair and accurate comparison of the balancing methods