Porto Seguro's Safe Driver Prediction

Group 4: Luca, Marcel, Robin

Task Overview

Our task

- Task: Predict whether a driver will file an insurance claim in the next year

- Target Variable: Binary (target = 0 or 1)
  - 0 = No insurance claim
  - 1 = Insurance claim

Porto: An insurance company in Brazil

- Founded in 1945 (80 years old) in Sao Paulo, Brazil
- Now has +13,000 employees (2021)
- One of Brazil's largest insurance companies

- Porto Seguro
  - €5.17 billion revenue
  - €0.2 billion net income
- For comparison, Allianz:
  - €152.7 billion revenue
  - €6.9 billion net income

Dataset

	0	1	2	3	4
id	7.000000	9.000000	13.000000	16.000000	17.000000
target	0.000000	0.000000	0.000000	0.000000	0.000000
ps_ind_01	2.000000	1.000000	5.000000	0.000000	0.000000
ps_ind_02_cat	2.000000	1.000000	4.000000	1.000000	2.000000
ps_ind_03	5.000000	7.000000	9.000000	2.000000	0.000000
ps_ind_04_cat	1.000000	0.000000	1.000000	0.000000	1.000000
ps_ind_05_cat	0.000000	0.000000	0.000000	0.000000	0.000000
ps_ind_06_bin	0.000000	0.000000	0.000000	1.000000	1.000000
ps_ind_07_bin	1.000000	0.000000	0.000000	0.000000	0.000000
ps_ind_08_bin	0.000000	1.000000	1.000000	0.000000	0.000000
ps_ind_09_bin	0.000000	0.000000	0.000000	0.000000	0.000000
ps_ind_10_bin	0.000000	0.000000	0.000000	0.000000	0.000000
ps_ind_11_bin	0.000000	0.000000	0.000000	0.000000	0.000000
ps_ind_12_bin	0.000000	0.000000	0.000000	0.000000	0.000000
ps_ind_13_bin	0.000000	0.000000	0.000000	0.000000	0.000000
ps_ind_14	0.000000	0.000000	0.000000	0.000000	0.000000
ps_ind_15	11.000000	3.000000	12.000000	8.000000	9.000000

```
'id',
 'target',
```

Label
(y value)

```
'ps_ind_01',
'ps_ind_02_cat',
'ps_ind_03',
'ps_ind_04_cat',
'ps_ind_05_cat',
'ps_ind_06_bin',
'ps_ind_07_bin',
'ps_ind_08_bin',
'ps_ind_09_bin',
'ps_ind_10_bin',
'ps_ind_11_bin',
'ps_ind_12_bin',
'ps_ind_13_bin',
'ps_ind_14',
'ps_ind_15',
'ps_ind_16_bin',
'ps_ind_17_bin',
'ps_ind_18_bin',
```

Data about
insured person

```
'ps_reg_01',
'ps_reg_02',
'ps_reg_03',
```

Data related to
registration
process

```
'ps_car_01_cat',
 'ps_car_02_cat',
 'ps_car_03_cat',
 'ps_car_04_cat',
 'ps_car_05_cat',
 'ps_car_06_cat',
 'ps_car_07_cat',
 'ps_car_08_cat',
 'ps_car_09_cat',
 'ps_car_10_cat',
 'ps_car_11_cat',
 'ps_car_11',
 'ps_car_12',
 'ps_car_13',
 'ps_car_14',
 'ps_car_15',
```

Data related
insured car

```
'ps_calc_01',
'ps_calc_02',
'ps_calc_03',
'ps_calc_04',
'ps_calc_05',
'ps_calc_06',
'ps_calc_07',
'ps_calc_08',
'ps_calc_09',
'ps_calc_10',
'ps_calc_11',
'ps_calc_12',
'ps_calc_13',
'ps_calc_14',
'ps_calc_15_bin',
'ps_calc_16_bin',
'ps_calc_17_bin',
'ps_calc_18_bin',
'ps_calc_19_bin',
'ps_calc_20_bin'
```

no idea

Feature correlations

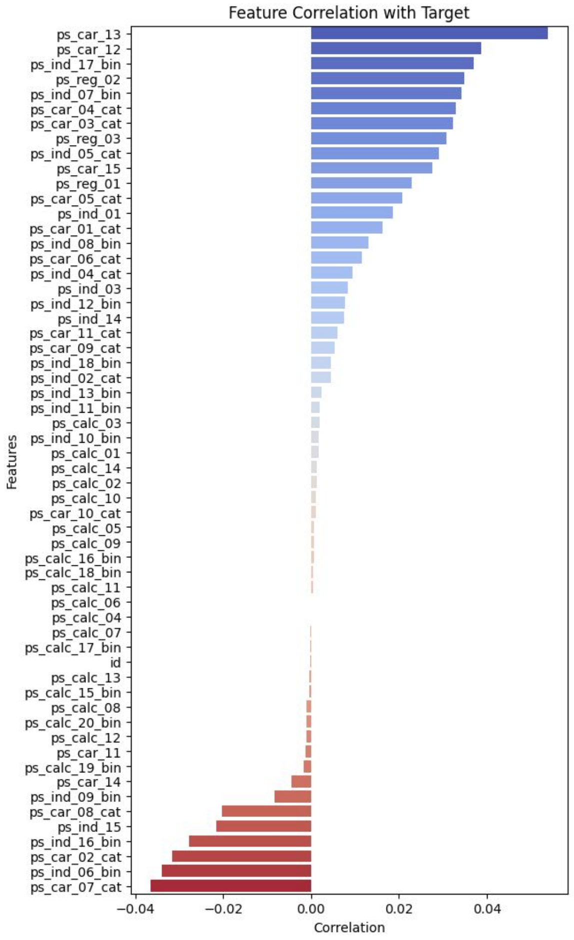
- All features have a very weak correlation of >-0.05 and < 0.05

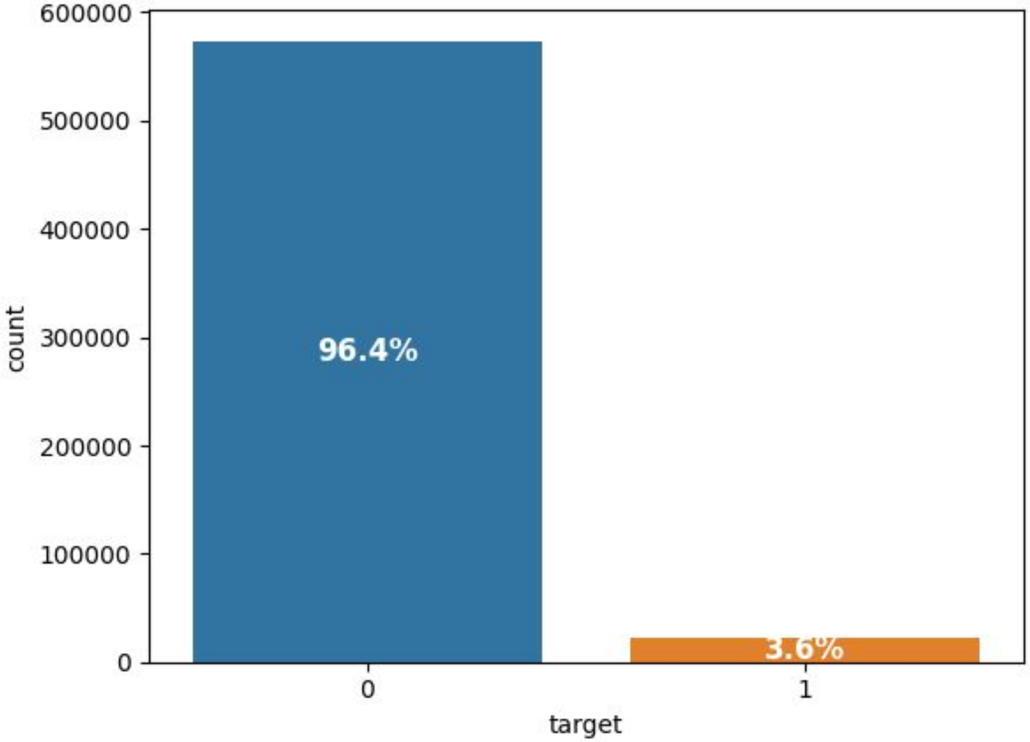
Good News:

- Little multicollinearity – the features are not strongly redundant.
- Models such as Random Forest, Gradient Boosting, or Neural Networks can benefit from this since each feature provides unique information.

Bad News:

- If the features also show little correlation with the target, the model may struggle to identify meaningful patterns.
- Linear models (e.g., Logistic Regression) may perform worse because they rely on linear relationships.



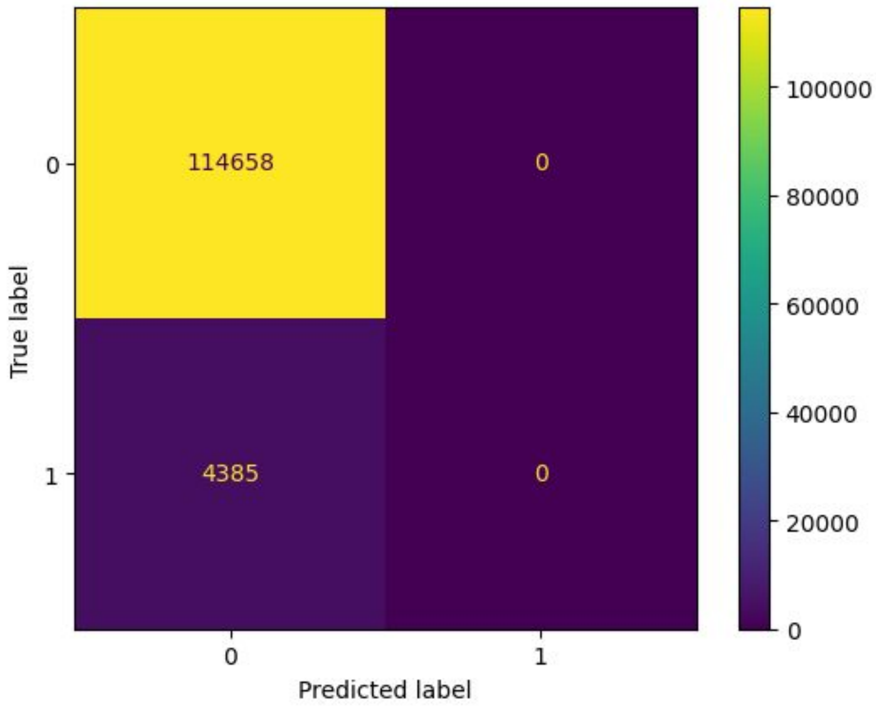


The dataset is unbalanced, there are only 3.6% of positive targets aka filed insurance claims.

Baseline Model

Baseline model

- No heuristics possible, since data is anonymized and no strong correlations found
- Decided for a dummy baseline model
- Predicts that all cases are equal to the majority value, so 0 (“no claim”)



Baseline model results

	precision	recall	f1-score	support
0	0.96	1.00	0.98	114658
1	0.00	0.00	0.00	4385
accuracy			0.96	119043
macro avg	0.48	0.50	0.49	119043
weighted avg	0.93	0.96	0.95	119043

Baseline model results

will go down as trade-off

needs to improve

	precision	recall	f1-score	support
0	0.96	1.00	0.98	114658
1	0.00	0.00	0.00	4385
accuracy			0.96	119043
macro avg	0.48	0.50	0.49	119043
weighted avg	0.93	0.96	0.95	119043

Baseline model results

	precision	recall	f1-score	support
0	0.96	1.00	0.98	114658
1	0.00	0.00	0.00	4385
accuracy			0.96	119043
macro avg	0.48	0.50	0.49	119043
AUC	0.6085			

most important metrics

Finding a better model

Model	Precision	Recall	F1-score	Accuracy	ROC-AUC
Dummy Classifier	0.00	0.00	0.00	0.96	0.61
Heuristic	0.07	0.04	0.05	0.95	0.51
Logistic Regression	0.00	0.00	0.00	0.96	0.62
KNN (k=5, numeric)	0.09	0.00	0.00	0.96	0.51
KNN + SMOTE (full)	0.74	1.00	0.85	0.82	0.95
KNN + SMOTE (train only)	0.04	0.36	0.07	0.66	0.52
Single Tree	0.05	0.05	0.05	0.93	0.49
Random Forest (base)	0.00	0.00	0.00	0.96	0.58
Random Forest (mod, no SMOTE)	0.07	0.59	0.13	0.72	0.72
Random Forest (tuned + SMOTE)	0.06	0.02	0.03	0.95	0.56
XGBoost (quick and dirty scale_pos_weight)	0.06	0.54	0.10	0.65	0.64
XGBoost (improved spw)	0.06	0.56	0.10	0.64	0.64
XGBoost (SMOTE, no spw)	0.07	0.00	0.01	0.96	0.50
Voting (soft, no weights)	0.05	0.27	0.08	0.78	0.56
Voting (soft, weighted 3-2-1)	0.06	0.09	0.07	0.91	0.57

Model	Precision	Recall	F1-score	Accuracy	ROC-AUC
Dummy Classifier	0.00	0.00	0.00	0.96	0.61
Heuristic	0.07	0.04	0.05	0.95	0.51
Logistic Regression	0.00	0.00	0.00	0.96	0.62
KNN (k=5, numeric)	0.09	0.00	0.00	0.96	0.51
KNN + SMOTE (full)	0.74	1.00	0.85	0.82	0.95
KNN + SMOTE (train only)	0.04	0.36	0.07	0.66	0.52
Single Tree	0.05	0.05	0.05	0.93	0.49
Random Forest (base)	0.00	0.00	0.00	0.96	0.58
Random Forest (mod, no SMOTE)	0.07	0.59	0.13	0.72	0.72
Random Forest (tuned + SMOTE)	0.06	0.02	0.03	0.95	0.56
XGBoost (quick and dirty scale_pos_weight)	0.06	0.54	0.10	0.65	0.64
XGBoost (improved spw)	0.06	0.56	0.10	0.64	0.64
XGBoost (SMOTE, no spw)	0.07	0.00	0.01	0.96	0.50
Voting (soft, no weights)	0.05	0.27	0.08	0.78	0.56
Voting (soft, weighted 3-2-1)	0.06	0.09	0.07	0.91	0.57

Model	Precision	Recall	F1-score	Accuracy	ROC-AUC
Dummy Classifier	0.00	0.00	0.00	0.96	0.61
KNN + SMOTE (full)	0.74	1.00	0.85	0.82	0.95
KNN + SMOTE (train only)	0.04	0.36	0.07	0.66	0.52
Random Forest (mod, no SMOTE)	0.07	0.59	0.13	0.72	0.72
XGBoost (improved spw)	0.06	0.56	0.10	0.64	0.64

SMOTE error

We used SMOTE on the dataset before splitting it. It should only be used for training, not testing!

We learned: First split the data, then use SMOTE for training!

Model	Precision	Recall	F1-score	Accuracy	ROC-AUC
Dummy Classifier	0.00	0.00	0.00	0.96	0.61
KNN + SMOTE (full)	0.74	1.00	0.85	0.82	0.95
KNN + SMOTE (train only)	0.04	0.36	0.07	0.66	0.52
Random Forest (mod, no SMOTE)	0.07	0.59	0.13	0.72	0.72
XGBoost (improved spw)	0.06	0.56	0.10	0.64	0.64

Model	Precision	Recall	F1-score	Accuracy	ROC-AUC
Dummy Classifier	0.00	0.00	0.00	0.96	0.61
KNN + SMOTE (train only)	0.04	0.36	0.07	0.66	0.52
Random Forest (mod, no SMOTE)	0.07	0.59	0.13	0.72	0.72
XGBoost (improved spw)	0.06	0.56	0.10	0.64	0.64

Model	Precision	Recall	F1-score	Accuracy	ROC-AUC
Dummy Classifier	0.00	0.00	0.00	0.96	0.61
KNN + SMOTE (train only)	0.04	0.36	0.07	0.66	0.52
Random Forest (mod, no SMOTE)	0.07	0.59	0.13	0.72	0.72
XGBoost (improved spw)	0.06	0.56	0.10	0.64	0.64

REVIEW error

The dataset wasn’t split, so the classification report was on the trained dataset.

We learned:
Do a proper review, always let a second person check what you’ve done!

Model	Precision	Recall	F1-score	Accuracy	ROC-AUC
Dummy Classifier	0.00	0.00	0.00	0.96	0.61
KNN + SMOTE (train only)	0.04	0.36	0.07	0.66	0.52
Random Forest (mod, no SMOTE)	0.07	0.59	0.13	0.72	0.72
Corrected Random Forest (mod, no SMOTE)	0.06	0.43	0.10	0.72	0.62
XGBoost (improved spw)	0.06	0.56	0.10	0.64	0.64

Model	Precision	Recall	F1-score	Accuracy	ROC-AUC
Dummy Classifier	0.00	0.00	0.00	0.96	0.61
KNN + SMOTE (train only)	0.04	0.36	0.07	0.66	0.52
Random Forest (mod, no SMOTE)	0.07	0.59	0.13	0.72	0.72
Corrected Random Forest (mod, no SMOTE)	0.06	0.43	0.10	0.72	0.62
XGBoost (improved spw)	0.06	0.56	0.10	0.64	0.64

Model	Precision	Recall	F1-score	Accuracy	ROC-AUC
Dummy Classifier	0.00	0.00	0.00	0.96	0.61
Corrected Random Forest	0.06	0.43	0.10	0.72	0.62
XGBoost (improved spw)	0.06	0.56	0.10	0.64	0.64

Our winning model

Basteline Model

	precision	recall	f1-score	support
0	0.96	1.00	0.98	114658
1	0.00	0.00	0.00	4385
accuracy			0.96	119043
macro avg	0.48	0.50	0.49	119043
weighted avg	0.93	0.96	0.95	119043

ROC AUC: 0.6085867220255219

XGboost Classification Report:

	precision	recall	f1-score	support
0	0.97	0.64	0.77	114704
1	0.06	0.56	0.10	4339
accuracy			0.64	119043
macro avg	0.52	0.60	0.44	119043
weighted avg	0.94	0.64	0.75	119043

ROC AUC: 0.6391

Random tree with modified parameters without SMOTE

XGBoost stands for **Extreme Gradient Boosting**.
XGBoost is an advanced tree-based algorithm that builds many small decision trees in sequence, each one improving on the last, to create a very strong predictive model.

It became popular because it consistently wins **Kaggle competitions** and performs well on many **classification, regression, and ranking problems**.