Capstone Project

Renan Hidani

Machine Learning Engineer Nanodegree

November 1st,2017

Definition

Project Overview

Car insurance is a common subject for anyone who has a car, it might be a brand-new or an old car, insurance is a financial protection for car crash, body injury or car theft. In some countries car insurance is optional, other countries is compulsory, the reason behind this is that the driver should have an insurance for third party liability.

Building an insurance proposal is not an easy task, many variables should be taken in consideration, more information related to features will be made in Analysis.

This project is based on Kaggle Competition Platform[1]. Porto Seguro has provided data for the Kaggle community to build a new fair classification algorithm to classify drivers that claim insurance.

Proposed Solution

The goal is to build a new classification algorithm for target feature, the results are supposed to be percentage of a driver that might claim insurance.

For this project will be considered the following classification techniques:

- Naïve Bayes – GaussianNB
- Stochastic Gradient Descent – SGDClassifier
- Random Forest – RandomForestClassifier
- Support Vector Machine (SVM) – SVC

Besides these 4 techniques, it will be performed a test in each group of features for each technique, this dataset has 4 group of features:

- Personal Information (Features _*ind*)
- Region Information (Features _*reg*)
- Auto Information (Features _*car*)
- Calculated Information (Features _*calc*)

Metrics

The algorithms will be evaluated according to the following metrics:

- Classification Report (Precision, Recall, F1 score)
- Confusion Matrix (FP, TP)
- AUC

[1] https://www.kaggle.com/c/porto-seguro-safe-driver-prediction#description

Analysis

Data Exploration

It has been provided 2 files to build a classification model: test.csv and train.csv

Test file should test your algorithm and train file should train it.

Test file has 595.213 records and Train file has 892.817 records, this is not very usual, training files usually have more records to build a better algorithm.

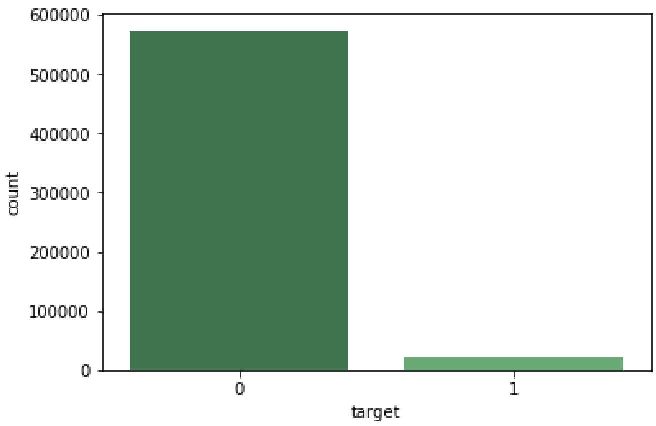
Test file has 58 features: ID, Target, 18 features related to personal information, 3 features related to region information, 15 features related to auto information and 20 features that are calculated.

Train file has 57 features: all features likewise test file, just dropping target, which is the feature to be predicted.

Data Visualization

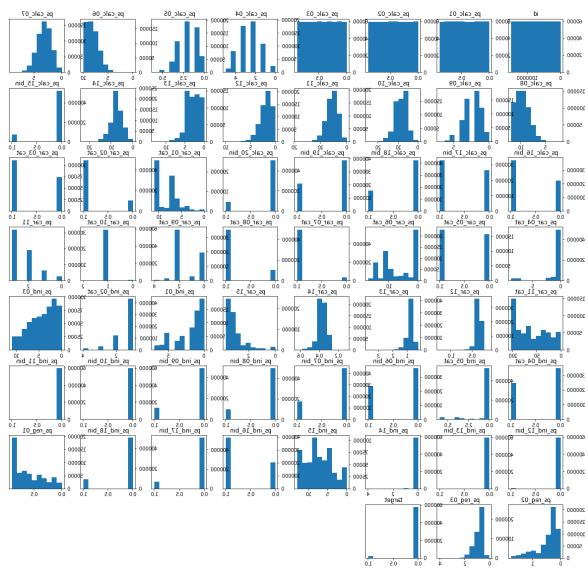
The first analysis to be made is to verify if feature target is balanced or not.

Figure 1 - Amount of Target Value



Target have 573.518 records which value is 0, this means a driver that doesn’t claim insurance, and 21.694 records which value is 1. This train file has many more records where the driver doesn’t claim insurance than the other way around (Unbalanced dataset, ratio 1:26.4 for target = 1).

Figure 2 - Histogram



The second analysis is to check the histogram, since there are binary and categorical values.

Since there are many features, it's difficult to check the values, so it will be left as an attachment the histogram (Attachment - 1).

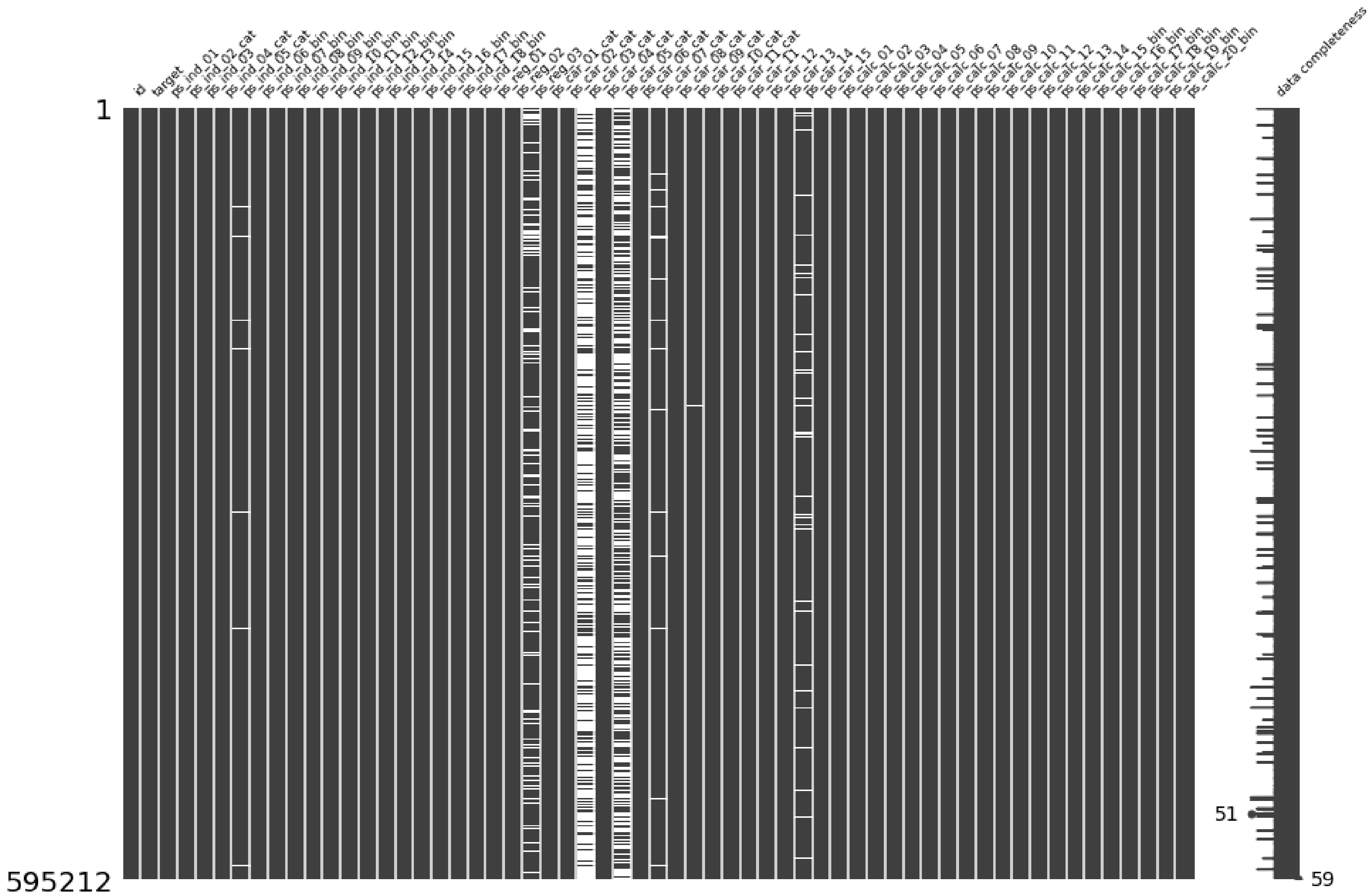
The histogram doesn't give too much information, some features have a normal distribution, and other features that are binary have almost all records with value 0.

The third is an important visualization, check the missing values with missingno library, the missing values are identified with value "-1" inside the dataset.

White lines indicate a missing value.

There are 56 features which 13 have missing values: ps_ind_02_cat, ps_ind_04_cat, ps_ind_05_cat, ps_reg_03, ps_car_01_cat, ps_car_02_cat, ps_car_03_cat, ps_car_05_cat, ps_car_07_cat, ps_car_09_cat, ps_car_11, ps_car_12 and ps_car_14.

Figure 3 - Missing Values



There are 2 features that have many records with missing values: ps_car_03_cat (411.231 records missing) and ps_cat_05 (266.551 records missing).

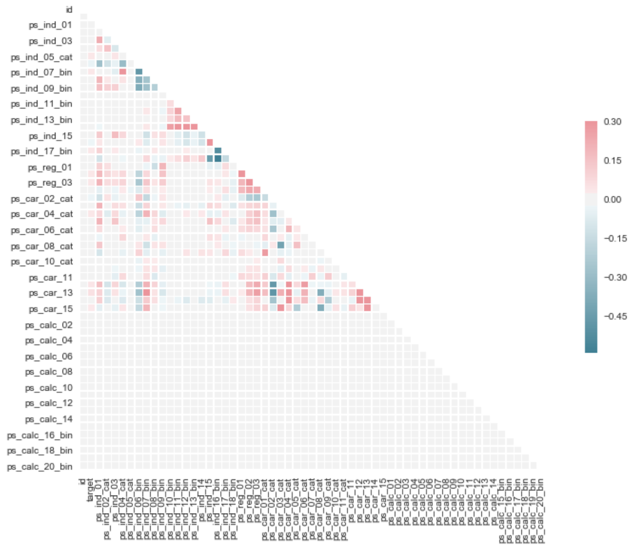
This is a big problem, since some classification algorithm can't process missing values.

It can be dropped rows that have missing values, or can be dropped features that have too many missing values, it can be considered the mean value from the feature as missing values, it can be imputed a value like -1 or -999 or it can be predicted through a regression algorithm.

It has been decided to clean all records with missing values (rows) and balance data (SMOTE - Oversampling), more information will be included in Methodology – Data Preprocessing.

Fourth data analysis has been correlation plot - Heatmap (Seaborn Library).

Figure 4 - Heatmap



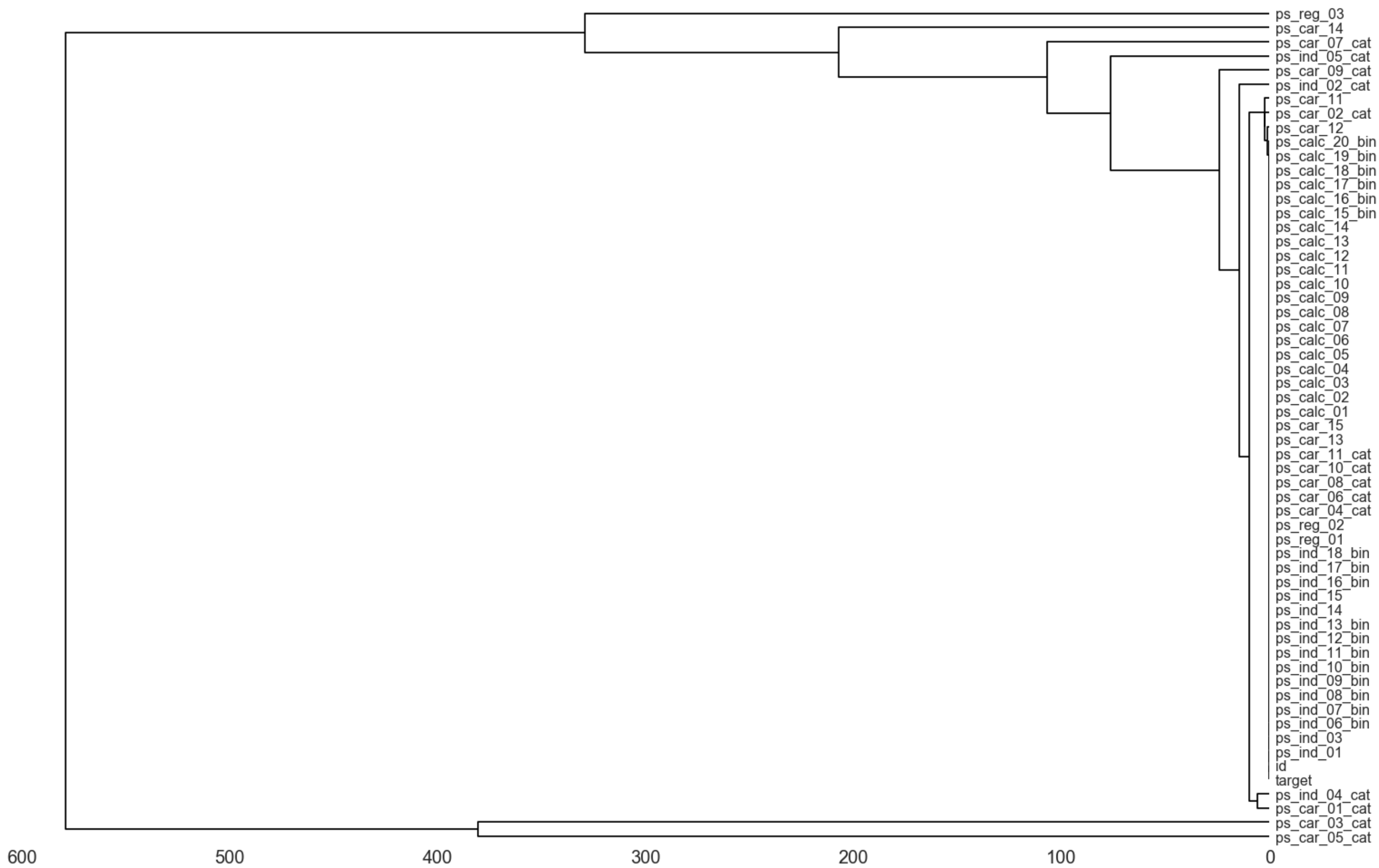
Since it's a big correlation plot, it will be attached to Attachment - 2.

Through this figure 4, it can be said that some features don't have any correlation, for example the calculation features.

The last figure is a dendrogram.

Dendrogram is more visual friendly. It can be concluded from dendrogram that ps_reg_03, ps_car14 and ps_car_07_cat is less correlated (ps_car_03_cat and ps_car_05_cat have many missing features).

Figure 5 - Dendrogram



Methodology

Data Preprocessing

For preprocessing it has been imported SMOTE and train-test split library.

The right way to use these two libraries is to train-test split library first and then applies SMOTE.[2]

Otherwise, training evaluation might score high but testing evaluation will score poorly.

SMOTE has been used to balance target feature to a ratio 1:1.

```
# Import SMOTE to balance data
from imblearn.over_sampling import SMOTE

# Import Train Test Split Library
from sklearn.model_selection import train_test_split

train_data_sm = train_data_clean.drop(['id','target'],axis =1)
train_target_sm = train_data_clean[['target']]

x_train, x_val, y_train, y_val = train_test_split(train_data_sm, train_target_sm, test_size = .1,
random_state = 0)

# Balance Data with Over-sampling
sm = SMOTE(ratio='auto',random_state = 0 , k=None, k_neighbors=5, m=None, m_neighbors=10, out_step=0.5,
kind='regular', svm_estimator=None, n_jobs=1)
x_train_data_balanced,y_train_data_balanced = sm.fit_sample(x_train, y_train.values.ravel())

# Shape from Balanced Data
x_train_data_balanced.shape
y_train_data_balanced.shape
```

```
x_train_data_balanced.shape
(214660L, 57L)
y_train_data_balanced.shape
(214660L,)
```

After applying SMOTE and train-test split it has been created a new balanced data with 214.660 records with all 57 features.

Feature Selection

For feature selection it has been imported SelectKBest, VarianceThreshold and SelectPercentile.

To select the features that best represent data, it has been made a cascade feature selection, which means that these 3 libraries has been applied in order:

1. VarianceThreshold
2. SelectPercentile
3. SelectKBest

To select features based on variance between features and then apply 2 features selection based on classification algorithm, it seems the best way to do this feature selection.

[2] https://beckernick.github.io/oversampling-modeling/

```
# Data Selection - Features

# Import the Feature Selection Library
from sklearn.feature_selection import SelectKBest, VarianceThreshold, SelectPercentile, chi2, f_classif

# Split train_data without missing values
x_train_data_selection = train_data_clean.drop(['id','target'], axis = 1)
y_train_data_selection = train_data_clean[['target']]

# New Array of Features without ID and Target
features_train_selection =
['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','ps_reg_01','ps_reg_02','ps_reg_03','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','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']
features_test_selection =
['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','ps_reg_01','ps_reg_02','ps_reg_03','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','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']

# Threshold for VarianceThreshold
thresholdforpreprocessing = 0.1

# Apply VarianceThreshold to train_data
selector = VarianceThreshold(threshold=thresholdforpreprocessing)
variance_train_data = selector.fit_transform(x_train_data_selection,y_train_data_selection)

# Get Features from train_data - VarianceThreshold
variance_array = selector.get_support()

# Create list for feaatures that are removed through VarianceThreshold
variance_values_removed = []
variance_values_maintained = []

# List to Store the Features that are removed
for i in range(len(variance_array)):
    if variance_array[i] == False:
        variance_values_removed.insert(i,features_train_selection[i])
    else:
        variance_values_maintained.insert(i,features_train_selection[i])
# Features removed through VarianceThreshold
print('Features removed through VarianceThreshold:')
print(variance_values_removed)

# Amount of features Removed
len(variance_values_removed)

#Features maintained
print('Features maintained:')
print(variance_values_maintained)

# Amount of features Maintained
len(variance_values_maintained)
```

```
Features removed through VarianceThreshold:
['ps_ind_10_bin', 'ps_ind_11_bin', 'ps_ind_12_bin', 'ps_ind_13_bin', 'ps_ind_14', 'ps_reg_01',
'ps_car_07_cat', 'ps_car_10_cat', 'ps_car_12', 'ps_car_13', 'ps_car_14', 'ps_calc_01', 'ps_calc_02',
'ps_calc_03']
Features maintained:
['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_15', 'ps_ind_16_bin', 'ps_ind_17_bin',
'ps_ind_18_bin', 'ps_reg_02', 'ps_reg_03', '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_08_cat', 'ps_car_09_cat', 'ps_car_11_cat',
'ps_car_11', 'ps_car_15', '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']
43
```

The VarianceThreshold get as parameter the threshold to select features that should be dropped, it has been selected “0.1” as a good variance.

It has been dropped 14 features from 57, remaining 43 features.

```
# Percentage of Features to keep (%)
select_percentile = 50

# Amount of features after SelectPercentile
percentile_features_maintained = int(0.5*len(variance_values_maintained))
print('Amount of features maintained after select percentile')
print(percentile_features_maintained)
print('Percentage of Features Removed:')
print(select_percentile)

# Drop the Features from VarianceThreshold
x_train_data_selectpercentile = x_train_data_selection.drop(variance_values_removed, axis=1)
y_train_data_selectpercentile = y_train_data_selection

# Apply SelectPercentile to train data - Evaluation chi2
percentile = SelectPercentile(chi2, percentile = select_percentile)
percentile_train_data = percentile.fit(x_train_data_selectpercentile,y_train_data_selectpercentile)

# Get Features from train_data_selection - SelectPercentile
percentile_array = percentile_train_data.get_support()

# Create list for feaatures that are removed through SelectPercentile
percentile_values_removed = []
percentile_values_maintained = []

# List to Store the Features that are removed
for i in range(len(percentile_array)):
    if percentile_array[i] == False:
        percentile_values_removed.insert(i,variance_values_maintained[i])
    else:
        percentile_values_maintained.insert(i,variance_values_maintained[i])
# Features removed through SelectPercentile
print('Features removed through SelectPercentile:')
print(percentile_values_removed)

# Amount of features Removed
len(percentile_values_removed)

#Features maintained
print('Features maintained:')
print(percentile_values_maintained)

# Amount of features Maintained
len(percentile_values_maintained)
```

```
Amount of features maintained after select percentile
21
Percentage of Features Removed:
50
Features removed through SelectPercentile:
['ps_ind_02_cat', 'ps_ind_18_bin', 'ps_car_05_cat', 'ps_car_09_cat', 'ps_car_11', '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']
Features maintained:
['ps_ind_01', '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_15', 'ps_ind_16_bin', 'ps_ind_17_bin', 'ps_reg_02',
'ps_reg_03', 'ps_car_01_cat', 'ps_car_02_cat', 'ps_car_03_cat', 'ps_car_04_cat', 'ps_car_06_cat',
'ps_car_08_cat', 'ps_car_11_cat', 'ps_car_15']
Out[12]: 21
```

After VarianceThreshold, it has been applied the SelectPercentile, which has been set to remove 50% of features based on chi2 classification algorithm.

A total of 22 features has been removed, remaining 21 features.

```python
# Amount of Features to keep - SelectKBest
kbest_features = int(0.8*(len(percentile_values_maintained)))

# Amount of features after K Best
print('Amount of features maintained after KBest')
print(kbest_features)

# Drop the Features from SelectPercentile
x_train_data_kbest = x_train_data_selectpercentile.drop(percentile_values_removed, axis=1)
y_train_data_kbest = y_train_data_selectpercentile

# Apply SelectKBest to data - Evaluation f_classif
selectK = SelectKBest(f_classif, k=kbest_features)
selectK_train_data = selectK.fit(x_train_data_kbest,y_train_data_kbest)

# Get Features from train_data_percentile - SelectKBest
selectK_array = selectK_train_data.get_support()

# Create list for feaatures that are removed through SelectKBest
selectK_values_removed = []
selectK_values_maintained = []

# List to Store the Features that are removed
for i in range(len(selectK_array)):
    if selectK_array[i] == False:
        selectK_values_removed.insert(i,percentile_values_maintained[i])
    else:
        selectK_values_maintained.insert(i,percentile_values_maintained[i])
# Features removed through SelectKBest
print('Features removed through SelectKBest:')
print(selectK_values_removed)

# Amount of features Removed
len(selectK_values_removed)

#Features maintained
print('Features maintained:')
print(selectK_values_maintained)

# Amount of features Maintained
len(selectK_values_maintained)
```

```
Amount of features maintained after KBest
16
Features removed through SelectKBest:
['ps_ind_03', 'ps_ind_04_cat', 'ps_car_06_cat', 'ps_car_08_cat', 'ps_car_11_cat']
Features maintained:
['ps_ind_01', 'ps_ind_05_cat', 'ps_ind_06_bin', 'ps_ind_07_bin', 'ps_ind_08_bin', 'ps_ind_09_bin',
'ps_ind_15', 'ps_ind_16_bin', 'ps_ind_17_bin', 'ps_reg_02', 'ps_reg_03', 'ps_car_01_cat',
'ps_car_02_cat', 'ps_car_03_cat', 'ps_car_04_cat', 'ps_car_15']
Out[13]: 16
```

The last feature selection is the SelectKBest, which it has been set to select the 80% best from the SelectPercentile, this is a total of 16 features.

5 features have been removed, and these 16 below are the ones that will be used to score in the Porto Seguro Competition.

['ps_ind_01', 'ps_ind_05_cat', 'ps_ind_06_bin', 'ps_ind_07_bin', 'ps_ind_08_bin', 'ps_ind_09_bin', 'ps_ind_15', 'ps_ind_16_bin', 'ps_ind_17_bin', 'ps_reg_02', 'ps_reg_03', 'ps_car_01_cat', 'ps_car_02_cat', 'ps_car_03_cat', 'ps_car_04_cat', 'ps_car_15']

Results

Model Evaluation and Validation

The model evaluation will base on the train-test split with 4 techniques (GaussianNB, SVC, SGDClassifier and RandomForestClassifier) applied each one to 6 different data (Personal Information only, Region Information only, Auto Information only, Calculated Information only, all features and best features).

This is a total of 24 test that will be printed as a matrix, each one with 3 metrics for evaluation.

One of metrics returns 3 variables: Precision, Recall and F1-score.

The other two are Confusion Matrix and Area under ROC curve.

		GaussianNB	SVC	SGDClassifier	RandomForestClassifier
Personal Information only	Precision	0.92	0.91	0.90	0.91
	Recall	0.27	0.87	0.19	0.94
	F1	0.38	0.89	0.26	0.93
	Confusion Matrix	[[2901 8995] [116 482]]	[[10776 1120] [546 52]]	[[1842 10054] [106 492]]	[[11754 142] 586 12]]
	AUC	0.53	0.50	0.49	0.50
Region Information only	Precision	0.91	0.90	0.91	0.91
	Recall	0.47	0.15	0.70	0.94
	F1	0.60	0.20	0.78	0.92
	Confusion Matrix	[[5543 6353] [254 344]]	[[1356 10540] [80 518]]	[[8524 3372] [387 211]]	[[11692 204] [584 14]]
	AUC	0.52	0.49	0.54	0.50
Auto Information only	Precision	0.92	0.91	0.91	0.91
	Recall	0.49	0.92	0.79	0.94
	F1	0.61	0.91	0.84	0.92
	Confusion Matrix	[[5715 6181] [231 367]]	[[11477 419] [583 15]]	[[9680 2216] [452 146]]	[[11673 223] [586 12]]
	AUC	0.55	0.50	0.53	0.50
Calculated Information only	Precision	0.91	0.91	0.00	0.91
	Recall	0.65	0.74	0.05	0.95
	F1	0.75	0.81	0.00	0.93
	Confusion Matrix	[[7939 3957] [405 193]]	[[9134 2762] [476 122]]	[[0 11896] [0 598]]	[[11896 0] [598 0]]
	AUC	0.50	0.49	0.5	0.5

Meaning of each variable

TP: Number of correct classification based on true target (target = 1)

TN: Number of correct classification based on false target (target = 0)

FP: Number of incorrect classification based on true target (target = 1)

FN: Number of incorrect classification based on false target (target = 0)

Precision: number of True Positives(TP) divided by TP and False Positives(FP)

Recall: number of TP divided by TP and False negative(FN), also called Sensitivity.

F1: measurement that combines precision and recall, its and harmonic mean.

F1 is calculated through equation below:

$$F1 = 2 \times \frac{precision\ \times recall}{precision + recall}$$

Confusion Matrix: $\begin{matrix} TP & FP \\ FN & TN \end{matrix}$

AUC (Area under the ROC curve): number that represents correlation between FP and TP, usually a ROC closer to one is better.

		GaussianNB	SVC	SGDClassifier	RandomForestClassifier
All Features	Precision	0.91	0.91	0.91	0.91
	Recall	0.51	0.94	0.88	0.95
	F1	0.63	0.92	0.90	0.93
	Confusion Matrix	[[5979 5917] [262 336]]	[[11703 193] [592 6]]	[[10918 978] [513 85]]	[[11894 2] [598 0]]
	AUC	0.53	0.50	0.53	0.50
Best Information Only	Precision	0.93	0.91	0.97	0.91
	Recall	0.60	0.80	0.53	0.95
	F1	0.71	0.85	0.66	0.93
	Confusion Matrix	[[7173 4758] [253 310]]	[[9946 1985] [469 94]]	[[6299 5632] [220 343]]	[[11898 33] [562 1]]
	AUC	0.58	0.5	0.57	0.50

Conclusion

Summary of Performance

GaussianNB – Naïve Bayes method

- Performed Poorly on almost every test.
- The Naïve Bayes Algorithm is the benchmark model, because it's simple, fast and powerful on application with a lot of data, like spam recognition.
- The Porto Seguro dataset is not so big, perhaps with more data this method would performed better.
- The best performance for GaussianNB occur when information passed through features selection.

SVC – Support Vector Machine method

- SVC performed poorly when tested with only region information (low recall score – 0.15). Since SVC construct a hyperplane or multiple hyperplanes to separate data, it seems that region information doesn't present this characteristic.
- Overall, SVC performed well on almost all techniques.

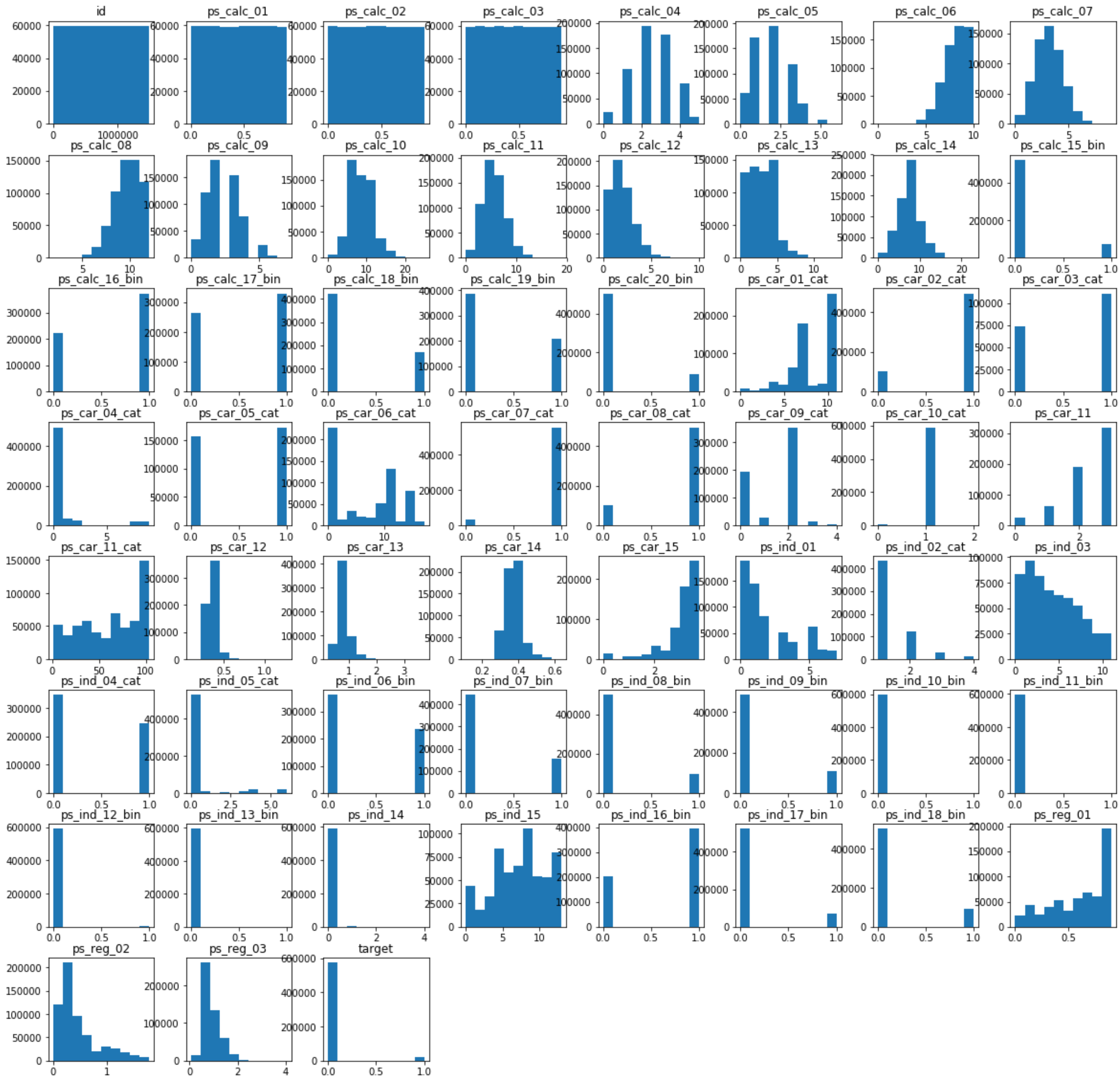
SGDClassifier – Stochastic Gradient Descent method

- Stochastic Gradient Descent performed poorly on 2 tests: Calculated Features only and Personal Features only.
- On other tests, SGDClassifier performed well.

RandomForestClassifier – Ensemble method (Decision Tree)

- Best method so far, since hasn't performed poorly on any tests.
- Solid Performance.
- This method has been chosen to predict the probability for insurance.

Attachment 1 – Histogram Plot



Attachment 2 – Heatmap