

Executive Summary

This project on Data Mining for “Predicting if a driver will file an insurance claim next year” was completed as part of Fall 2017 course curriculum for “Introduction to Data Science” under the guidance of Prof. Daisy Zhe Wang at University of Florida, Gainesville was done by Sanket Kumar and Bharat Alva (Group 23).

Problem Statement:

The goal of this project is to build a statistical model that can accurately predict whether a driver will initiate an auto insurance claim next year based on the dataset provided by Kaggle for a competition called “Porto Seguro’s Safe Driver Prediction” which ended in Nov-2017 with a prize money of US \$25,000. A more accurate prediction will allow Porto Seguro to further tailor their prices, and hopefully make auto insurance coverage more accessible to more drivers.

Data/benchmark Summary:

The Kaggle data set consists of 3 files: **train.csv**, **test.csv**, and **sample_submission.csv**. Since the insurance claim responses for the test set is not available, we split the training set “**train.csv**” into two sets: training (70%) and testing (30%) for model building, testing, and validation.

This dataset was complex due to large number of columns, masked column description, limited data type information from Kaggle, large missing data, skewed data distribution and lots of categorical features. There was huge imbalance in the target data: 96.36% for claims NOT initiated vs 3.64% for claims initiated. This required that we balance the data, both under and over.

Summary of dataset **train.csv**

- | | | | |
|--------------------------------|--|-------------------------------|------------------------------|
| • Size: | <i>Original: 110 MB</i> | <i>Oversampled: 264 MB</i> | <i>Under-sampled: 8 MB</i> |
| • Total no. of rows: | <i>Original: 595,212</i> | <i>Oversampled: 1,094,174</i> | <i>Under-sampled: 43,388</i> |
| • Total no. of columns: | <i>59 (Categorical data type: 54, Continuous data type: 5)</i> | | |

Approach/Algorithms Used:

The approach included data pre-processing, data cleaning, data exploration and visualization, model building, model evaluation, conclusion, and presentation of findings under various combination of train and test data sets. Jupyter Notebook with Python3 was used as the programming tool. The algorithms used were Logistic Regression with one hot encoder, Random Forest, and Gradient Boost Classifier.

Result and Evaluation Metrics:

We used AUC, Confusion Matrix, Accuracy and F1 score for model evaluation for this binary classification (claim filed or not filed). All the result sets were compiled for original, under-sampled and over-sampled data sets. “**Logistic Regression**” with One Hot Encoding was chosen as the final model as it provided the best combination of Accuracy and F1 score (Accuracy: 59%, F1 score: 58%) under the most likely situation: Train the model with balanced data and make Prediction on original data. The total time for running this combination was 5 minutes approximately. The end-to-end run time from data pre-processing to model result was 18 minutes.

Improvement after the last presentation

1. Using PANDAS_PROFILING for better understanding of the data: We could eliminate more irrelevant features based on this report combined with the eliminations based on missing data and multicollinearity.
2. Using ONE HOT ENCODING for categorical features. It improved the accuracy, F1 score and AUC of Logistic Regression.
3. Using Median (instead of mean used earlier) for missing value imputation in continuous variables. It resulted in performance improvement of all models with all combinations.
4. We dropped using over-sampled data because the results were similar to under-sampled data. What was important was to use balanced data set. This saved time.

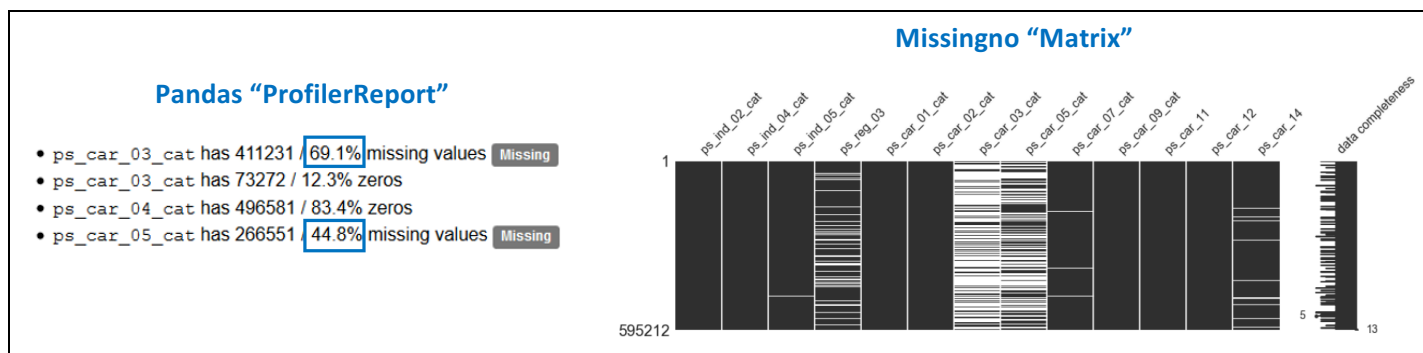
Details on Problem Statement and Data

The dataset was provided by Kaggle (<https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/data>). We first analysed the **train.csv** data and had the following observations:

Response variable: This information was available from Kaggle. The target variable is "target", which is 1 if a claim was filed, and 0 otherwise. This is clearly a case of binary prediction. This column is highly imbalanced having 96.36% of "0" values and 3.64% of "1" values. We balanced the data by Under-sampling and Oversampling.

Feature/Predictor variables: As per Kaggle, the variables ending with "cat" are categorical, and those ending with "bin" are binary, rest are either ordinal or continuous data types. We used the highest number of unique values of a stated "_cat" variable to differentiate between ordinal and continuous variables. This number was 104 for the variable "ps_car_11_cat". Variables with number of unique values <= 104 was considered ordinal and the rest were treated as continuous variables. As a result, we had a total of 54 categorical and 5 continuous datatypes including "id" and "target".

Missing and Skewed Data in the variables: We used PANDAS PROFILING along with other techniques to study the presence of this type of data.

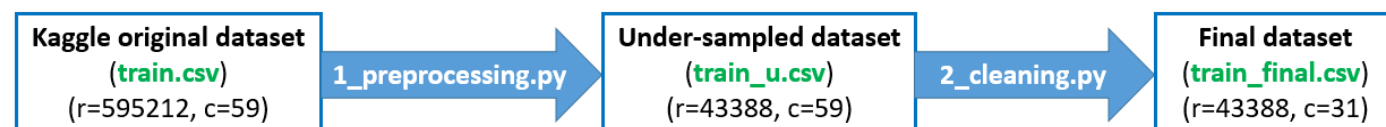


The following columns were deleted for various reasons:

- | | |
|---|--|
| (1) ps_car_03_cat (69.1% missing values) | (5) ps_ind_11_bin (99.8% zeros, highly skewed) |
| (2) ps_car_05_cat (44.8% missing values) | (6) ps_ind_12_bin (99.1% ZEROS) |
| (3) ps_car_10_cat (99.2% data has only one value) | (7) ps_ind_13_bin (99.9% zeros, highly skewed) |
| (4) ps_ind_10_bin (100% zeros, highly skewed) | (8) ps_ind_14 (98.9% zeros) |

Detecting Multicollinearity in the data set: We used Correlation Matrix to study the multicollinearity in the dataset, which suggested that all the "_calc" columns (20 columns) be dropped.

Our Data Preparation Journey: Finally, our data exploration, visualization, and analysis resulted in the following data preparation flow to produce the final dataset for future steps. In the diagram below, "r" stands for number of rows, and "c" stands for number of columns in the dataset.



Data Processing Scripts: Two python scripts were created for the following purposes:

- 1_preprocessing.py:** This script under-samples the original data. We had experimented with oversampling as well, but we discarded it as the results were similar to under-sampled data.
- 2_cleaning.py:** This script drops the variables identified above and handles missing data: mode for categorical variables and median for continuous variables.

Details of the Process

Approach: When we looked at the data and the information provided at Kaggle, we had to settle with the fact that we can't apply business logic to explore the variables in terms of applying any time lag or create other derivatives. So, we focussed completely on the data analysis using technical tools. Our approach included data quality assessment and handling, data analysis and preparation from algorithm's perspective, model building, model evaluation, conclusion, and presentation of findings.

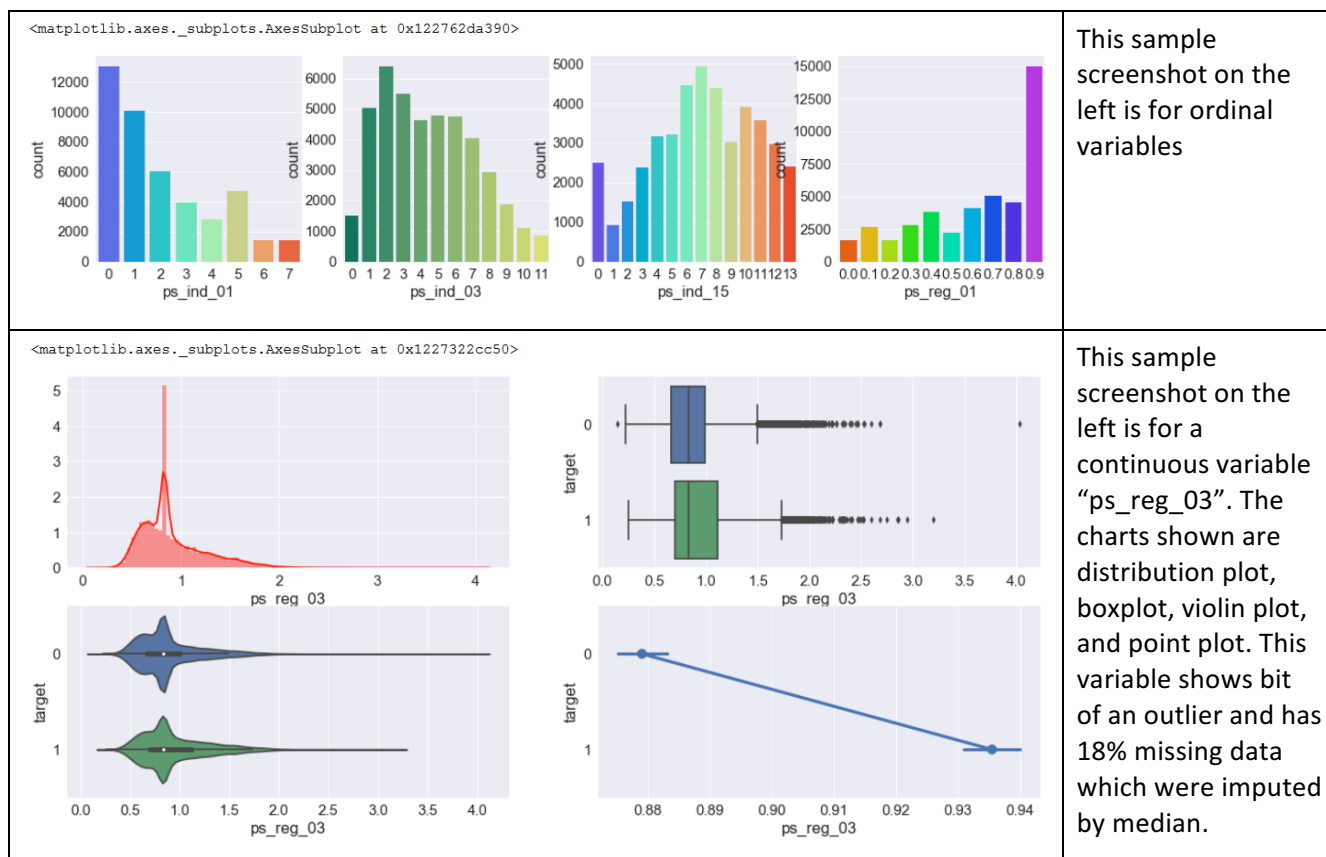
As mentioned in the previous section, the dataset **train_final.csv** was prepared after handling imbalances in "target" variable, dropping the unwanted variables and handling the missing data. Further handling and study of the data was done in "**3b_exploring_and_visualizing_after_data_cleaning.ipynb**" Jupyter file in the following manner:

1. Separate the target and features variables for further calculation and visualization
2. Segregate features into 4 parts: "_cat", "_bin", "_ord", and "_con" for categorical, binary, ordinal, and continuous data types
3. The "_cat", "_bin", and "_ord" features were converted into categorical data type

```
16 train.dtypes.value_counts()
Out[99]: category    26
         float64     4
         dtype: int64
```

The screenshot on the left shows count of columns of various types. Our file **train_final.csv** had 31 columns including "id". We had dropped "id" column earlier in the code set.

4. With these categorized features, we carried out intensive data visualization and analysis. Sample screenshots are shown here.



5. This data was found to be fit for statistical algorithms discussed next.

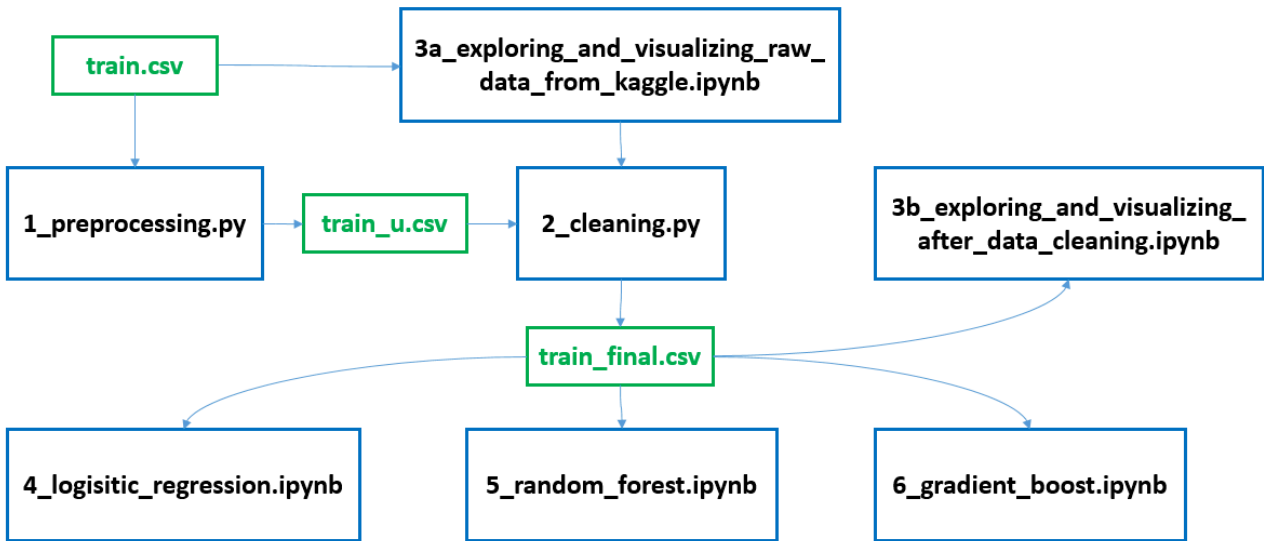
Algorithms and Programming Tools Used: We used Python script for data pre-processing and cleaning. For the rest of the steps, we used Jupyter notebook with Python3. We used three statistical algorithms: Logistic Regression, Random Forest, and Gradient Boost classifier on three types of data sets: original, under-sampled, over-sampled.

One Hot Encoder was used in Logistic Regression to transform categorical features to a format that works better with classification and regression algorithms in general. However, algorithms like random forests and gradient boost, handle categorical values natively and hence, one hot encoding was not used for them.

Source Code Structure: The source code has been organized in the following manner:

<div><div>group_23 > source_code</div><div><div>Search source_code</div><div><div>Name</div><div><div>1_preprocessing.py</div><div>2_cleaning.py</div><div>3a_exploring_and_visualizing_raw_data_from_kaggle.ipynb</div><div>3b_exploring_and_visualizing_after_data_cleaning.ipynb</div><div>4_logisitic_regression.ipynb</div><div>5_random_forest.ipynb</div><div>6_gradient_boost.ipynb</div><div>Readme</div></div></div></div></div>	<p>1_preprocessing.py: generates under-sampled data</p> <p>2_cleaning.py: drops identified columns, handles missing data</p> <p>3a_exploring_and_visualizing_raw_data_from_kaggle.ipynb: this code set gives important information to carry out data handling in 2_cleaning.py.</p> <p>3b_exploring_and_visualizing_after_data_cleaning.ipynb: this code set helps in data visualization and analysis for further validations to ensure that the data is fit for statistical algorithms.</p>
<p>4_logisitic_regression.ipynb: this code set does required data transformation including one hot encoding for Logistic Regression model building, performance evaluation, and plots ROC</p> <p>5_random_forest.ipynb: this code set handles model building, evaluation, and features importance for Random Forest classifier</p> <p>6_gradient_boost.ipynb: this code set handles model building, evaluation, and features importance for Gradient Boost classifier.</p>	

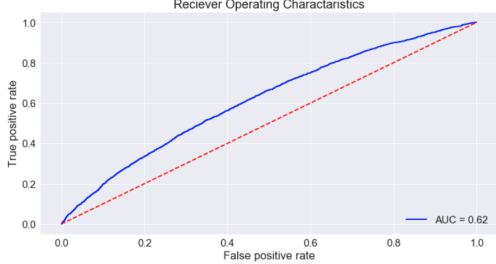
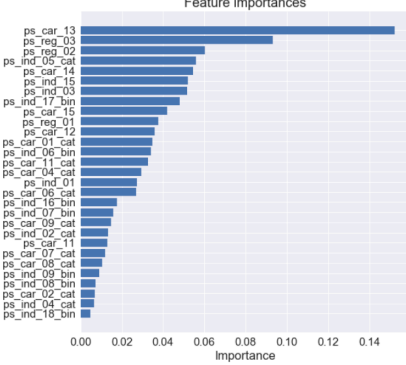
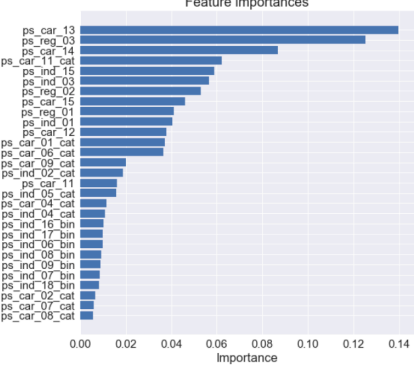
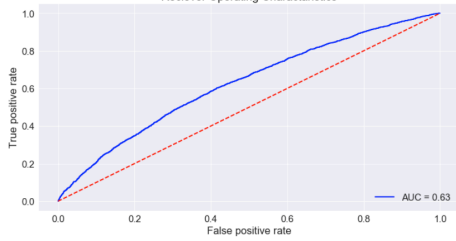
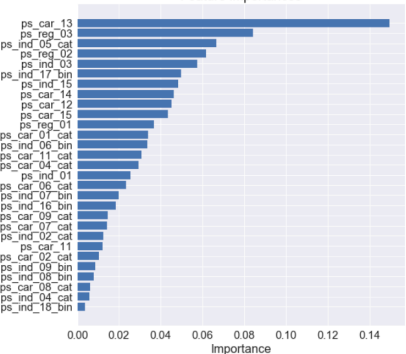
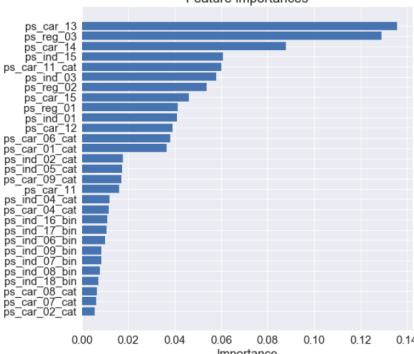
End-to-end System diagram: The end-to-end system diagram is shown below:



Details of the Evaluation

Metrics and Evaluation Steps: The following criteria was used for model evaluation

- Accuracy
- Cross Validation (CV) score
- AUC
- F1 score

Logistic Regression	Random Forest	Gradient Boost
Model built and tested on Under-sampled data		
LR Confusion matrix <pre>[[2638 1700] [1932 2406]]</pre> LR Accuracy: 58.1373905025 % LR Mean cv Score: 61.9989919882 %	RF Confusion Matrix <pre>----- Predicted 0 1 Actual 0 3951 2575 1 2742 3749 -----</pre> RF Accuracy: 59.1534147653 % RF f1 score: 58.5095591104 %	GB Confusion Matrix: <pre>Predicted 0 1 Actual 0 3946 2580 1 2823 3668 -----</pre> GB Accuracy: 58.4927402627 % GB f1 score: 57.5869377502 %
		
Model built on Under-sampled data and tested on Original data		
LR Confusion matrix <pre>[[2689 1649] [1893 2445]]</pre> LR Accuracy: 59.1747349009 % LR Mean cv Score: 62.4916901108 %	RF Confusion Matrix <pre>----- Predicted 0 1 Actual 0 104951 67028 1 2642 3943 -----</pre> RF Accuracy: 60.9831769002 % RF f1 score: 10.1681365723 %	GB Confusion Matrix: <pre>Predicted 0 1 Actual 0 105219 66760 1 1464 5121 -----</pre> GB Accuracy: 61.7929705876 % GB f1 score: 13.0527871945 %
		
Model built and tested on Original data		
LR Confusion matrix <pre>[[71242 43461] [1879 2459]]</pre> LR Accuracy: 61.9122823229 % LR Mean cv Score: 63.4902537372 %	RF Confusion Matrix <pre>----- Predicted 0 1 Actual 0 113447 58532 1 3171 3414 -----</pre> RF Accuracy: 65.4448825071 % RF f1 score: 9.9633742394 %	GB Confusion Matrix: <pre>Predicted 0 1 Actual 0 171974 5 1 6584 1 -----</pre> GB Accuracy: 96.3100064963 % GB f1 score: 0.0303444090426 %

Detailed Analysis of the Results

Analysis of Results: As it is evident from the screenshots of the results obtained, **Logistic Regression** has given the best result followed by Random Forest Classifier and Gradient Boost Classifier based on the Accuracy + F1 score. We also see that the re-sampling of data it required to balance the target variable distribution. The most likely situation will be testing on original data. That leaves us from with two choices: train on balanced data and train on original data. In this case, the best F1 score is 57.99% from Logistic Regression.

Train set	Test set	Algorithm	Accuracy	F1 Score	AUC
Under-sampled	Under-samples	Logistic Regression	58.14%	56.98%	62%
		Random Forest	59.15%	58.51%	---
		Gradient Boost	58.49%	57.59%	---
Under-sampled	ORIGINAL	Logistic Regression	59.17%	57.99%	62.49%
		Random Forest	60.98%	10.17%	---
		Gradient Boost	61.79%	13.05%	---
Original	Original	Logistic Regression	61.91%	9.79%	63.49%
		Random Forest	65.44%	9.96%	---
		Gradient Boost	96.31%	0.03%	---

Comparisons of Different Approaches: Logistic Regression has given the most consistent result across our three scenarios. Gradient Boost has shown the largest variation in both accuracy (from 58.49% to 96.31%) as well as F1 score (from 0.03% to 57.59%). Random Forest Classifier has performed somewhere in-between.

Logistic regression is the generalization of linear regression. It is used primarily for predicting binary or multiclass dependent variables. Because the response variable is discrete, it cannot be modelled directly by linear regression. Therefore, rather than predicting point estimate of the event itself, it builds the model to predict the odds of its occurrence. In two class problem odds greater than 50% would mean that the case is assigned to the class designed as "1" and "0" otherwise. While logistic regression is a powerful modelling tool, it assumes that the response variable is linear in the coefficients of the predictor variable.

Random forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

The feature importance has similar characteristics for all the three algorithms, which indicates that the predictors were selected correctly.

We also conclude that the sample balancing was a must. Even an under-sampled training set has performed better in predicting the original data than the original itself.

Conclusion on Challenges and Lessons learned

Challenges: It is complex project because of the type and quality of data sets and lack of business metadata. These can be summarized as:

- The data set has large no of variables (59).
- No clear data type has been mentioned for Nominal/Continuous data types. We have made our own decision based on unique number of values in a column for the largest known categorical feature.
- The data set has large no of categorical variables (54).
- Large missing values and highly skewed data. It required heavy imputation.
- Business metadata is unknown. Hence no intuitive business validation could be carried out. It is pecially true for “_calc” columns where a mistake can easy be made regarding its data type.

Lessons Learnt and Future Direction:

- The 2nd most important features “ps_reg_03” has **18.1%** missing data. Its missing values were imputed by median. If time permitted, we could have looked into better methods for missing value imputation for such an important feature.



- In conclusion, we can prove that it is possible to use a superior analytics algorithm through the use of different sampling method to correctly classify drivers according to their probability of filing insurance claim.

References

1. <https://www.youtube.com/watch?v=suhHIXHPkksk>
2. https://www.youtube.com/watch?v=JFJIQO_2ijg
3. <http://stackoverflow.com/questions/30002013/error-in-confusion-matrix-the-data-and-reference-factors-must-have-the-same-nu>
4. <http://www.ijarcce.com/upload/2016/march-16/IJARCCE%20128.pdf>
5. <http://www.informatica.si/index.php/informatica/article/viewFile/148/140>
6. <https://pdfs.semanticscholar.org/cade/435c88610820f073a0fb61b73dff8f006760.pdf>
7. Jiawei Han, Micheline Kamber, Jian Pei (2011). Data Mining: Concepts and Techniques, Third Edition.
8. Dursun Delen, Glenn Walker, Amit Kadam (2004). Predicting breast cancer survivability: a comparison of thee data mining methods. Artificial Intelligence in Medicine, 34, 113-127.
9. Mazurowski M.A., Habas P.A., Zurada J.M., Lo J.Y., Baker J.A., Tourassi G.D. (2008). Training neural network classifiers for medical decision making: The effects of imbalanced dataset on classification performance. Neural Network, 21, 427-436.