Porto Seguro's Safe Driver Prediction

Team Name: ForkHead
Salmaan Pehlari (011409307) Shrey Bhatt (011489777) Vansh Zaveri (011463322)
GitHub Link : https://github.com/ShreyBhatt/Project_256

Ch.1 Introduction

Motivation

- It is mandatory for each and every driver to have a car insurance because of the principle of "Safety First". But, what truly decides the amount of insurance premium one pays? Taking more insurance premium from a good driver and less from a bad one can cause great loss to any auto insurance company; as well as to the drivers who drive safely and still pays more insurance. To precisely decide the auto insurance coverage and rates for wide range of drivers, there should be a model that can predict whether a driver will claim an insurance in the next year or not.

Objective

- In this project, one of the Brazil's largest auto and homeowner insurance company; Porto Seguro, is challenging Kaggle's Machine Learning Community to build a machine learning model that can predict the probability of a driver claiming an insurance in the coming year. The company Porto Seguro has already used machine learning from past 20 years.
- Our objective is to build a machine learning model by using data mining and machine learning methods and classification algorithms, resulting in a more accurate prediction which will help an insurance company to better decide their insurance charges and can serve more drivers where the good ones pays less car insurance than the bad drivers. A win win situation for both; the consumer as well as the insurance company.

Ch.2 System Design & Implementation details

Algorithms Selected: Logistic Regression, Gradient Boosting Classifier, Gaussian Naive Bayes, Bernoulli Naive Bayes, SGD Classifier, Random Forest Classifier, Extra Trees Classifier , Decision Trees Classifier

- We started with exploring the dataset and see the amount of people who have claimed insurance compared to those who have not and saw there is a wide imbalance in them.

- We saw the features of the dataset which were a mix of categorical and binary features and saw that there were some features which had lots of missing values in them.
- We tried to establish relationships between various features of the dataset using Pearson's correlation matrix and saw some features which have high correlation.
- We began with the simplest approach of Logistic Regression and SGD Classifier without doing a lot of preprocessing on the dataset to see what kind of results can be seen.
- On getting unsatisfactory results we spent a lot of time in analyzing the dataset and preprocessing the dataset using techniques like imputing missing values, converting categorical features to dummy features and removing features with high missing values.
- After that we tried Naive Bayes classifiers like GaussianNB and BernouliNB from sklearn.naive_bayes and Decision Tree Classifier and Extra Tree Classifier from sklearn.tree
- Next, we tried ensemble algorithms like Random Forest Classifier, AdaBoost Classifier, Gradient Boosting Classifier from sklearn.ensemble. Gradient Boosting Classifier improved the evaluation score. We tried almost all the algorithms from sklearn library of python to compare the results and get the idea which algorithm can work for the classification of imbalanced data.

  Tools and Technologies Used: python, jupyter notebook, sklearn, imblearn
  We can code efficiently using jupyter notebook in python and we are also familiar with python so we decided to use that.

System Architecture Design

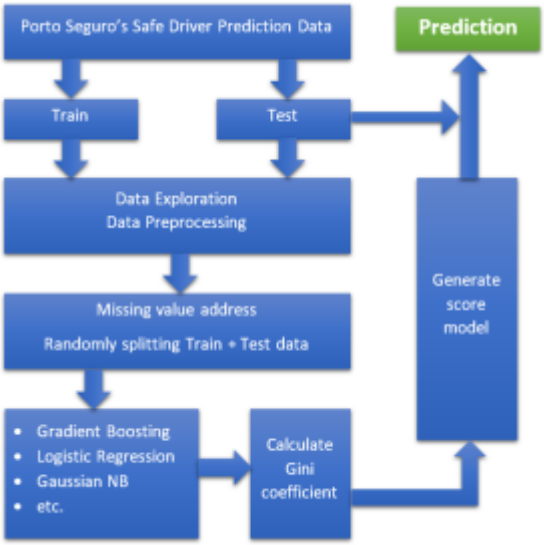


Figure 1 System Architecture

- First, we did some data exploration so that we can get a clear understanding about the entire dataset.
- In the data-preprocessing, we removed columns having more of -1 values because they might not be the good contributors in making the right predictions.

Basically, we applied most of the classification algorithms for making our classification model

1) Logistic Regression Classifier
2) SGD Classifier
3) Gaussian Naive Bayes Classifier
4) Bernoulli Naive Bayes Classifier
5) Random Forest Classifier
6) Extra Trees Classifier
7) Decision Trees Classifier
8) Gradient Boosting Classifier

- We calculated the Gini coefficients for evaluating our classification model.
- We applied our test data in the training model to generate the scores of Gini values for the target column prediction in the test data.

Ch.3 Experiments / Proof of concept evaluation

Dataset Used: Porto Seguro's Safe Driver Prediction Dataset - Kaggle
Size: train.csv (110 MB), test.csv (164 MB)
Dimensions of data: train data (595212 * 59), test data (892816 * 58)

Data description

- There are total 59 variables in the training dataset, which includes the policyholder's ID as well as the target column. So, in total there are 57 distinct features which are anonymized. As per the description provided in the competition, the features are tagged as per similar groupings which is stated in its name prefix, and hence can be categorized and differentiated based on its naming convention.
- The prefix in the variable names includes ind, reg, car and calc. 'Ind' stands for individual/policyHolder; 'reg' stands for region; 'car' for the vehicle and 'calc' for the calculated values. The features are either binary, categorical, continuous or ordinal, which can be derived from its name.
- Values of -1 in the variable indicates that the particular feature is missing in the policy holder's feature set.

Preprocessing decisions

- We dropped the ‘id’ columns as they won’t contribute in predicting the probability of insurance claim i.e the target variable.
- We then replaced -1 with NaN for missing values and then calculated number of null or missing values in every features. We then dropped the top 3 columns having really high number of missing values as they will not add anything worthwhile to the prediction. Then we imputed the missing values with the idxmax() i.e the value having maximum number of occurrence over the requested axis.
- Now, from the training data we only used dummy object values using pd.get_dummies() of pandas for the ‘categorical’ features. Now we splitted the train data into 70% training and 30% testing using train_test_split() function of sklearn.model_selection.
- The class weight was unbalanced in the train data. We tried to run the Logistic Regression, SGD Classifier, Extra Trees Classifier and Decision Trees Classifier using this data and the score was lower.
- Then we also tried to balance the classes by providing the parameter class_weight = 'balanced' in the model and the score increased for all models than using SMOTETomek.
- We also tried dimensionality reduction using TruncatedSVD() with n_components = 40 and n_components = 50 and compared the Gini co-efficients.
- The dataset was highly imbalanced and we had only few 1s in the target variable.
- No of people who claimed insurance = 21694 (3.644752%) People who did not claim insurance = 573518 (96.355248%)
- We tried to oversample the 1s and undersample the 0s using the re-sampling technique. We used SMOTETomek() from imblearn.combine for 50000 and 100000 records since using all records would fail on our systems. This did not give the best performance but improved it over the baseline.

Methodology followed

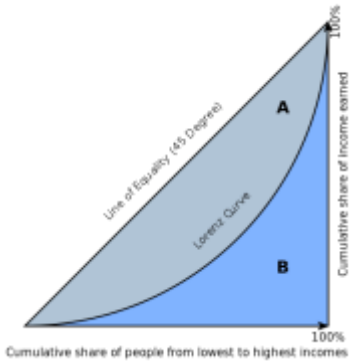
1) Data Exploration

- We noticed that there were many missing values in the dataset which were identified with -1 values. These values can be problematic when we apply classification algorithms. We tried to identify the number of missing values in each columns using the bar plots.
- Then we explored the target variable for which we wanted to predict the probabilities for the test data.
- From the target variable in the training data we found following result after plotting the bar and pie chart for the target variable.

- No of people who claimed insurance: 21694 (3.644752%) People who did not claim insurance: 573518 (96.355248%)
- We explored the features which have binary attributes.
- We plotted a heatmap of Pearson correlation of continuous features and we noticed that columns ps_car_13 and ps_car_12 have high correlation, as well as columns ps_reg_02 and ps_reg_03.

2) Evaluation Metric

- The metric to assess the result in the competition is Normalized Gini Coefficient. It is most commonly used in measuring inequality.
- Gini is defined based on the Lorenz curve. The sample graph shown is from Wikipedia, which shows the proportion of total income of the population which is earned by the bottom x% of the population.
- The Gini coefficient in this case can be defined as the ratio of the area between Line of Equality and Lorenz curve, over the total area of the triangle formed. It can range from 0 i.e. complete equality to 1 i.e. complete inequality.



$$G = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|x_i - x_j|}{2\sum_{i=1}^{n}\sum_{j=1}^{n}x_j} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|x_i - x_j|}{2n\sum_{i=1}^{n}x_i}$$

3) Linear Classification Models

- **Logistic Regression**

  We started with the Logistic regression classifier as it can be simple and intuitive approach to easily solve the classification problem. We tried to solve it without balancing the class but we got 0.0 Gini score which shows that the algorithm results were random guesses. Then we balanced the class and got the Gini score in the range of 0.230-0.239.

  We also tried dimensionality reduction using components = 40 and components = 50 but it did not help improve the score much.

- **SGD Classifier**

  We tried SGD Classifier after Logistic Regression because it was also in sklearn.linear_model. It also gave 0.0 gini score without class imbalance. After balancing the class weight it gave the Gini score in the range 0.042-0.062 which was significantly less than what we got with Logistic Regression.

4) Naive Bayes Classifiers

- **Bernoulli Naive Bayes**
  Bernoulli Naive Bayes gave the Gini score in the range of 0.189-0.219. We tried to perform dimensionality reduction but it did not increase the score.

- **Gaussian Naive Bayes**
  In Gaussian Naive Bayes dimensionality helped in increasing the score. Initially, the score was 0.161 which increased with dimensionality reduction to 0.206 with components = 50 and further to 0.210 with components = 40.

5) Ensemble algorithm

- **Random Forest Classifier**
  Random forest did not perform well in this case. It gave the gini scores in the range of 0.062-0.067. In did not even worked well with dimensionality reduction or re-sampling.

- **Gradient Boosting Classifier**
  Gradient Boosting performed the best in all the cases. It gave the gini score in the range of 0.232-0.262. It did not performed well with dimensionality reduction.

6) Tree Classifier algorithms

- **Extra Trees Classifier and Decision Trees Classifier**
  Classification algorithms from sklearn.tree like **Extra Trees Classifier** and **Decision Trees Classifier** did not perform well in target prediction. They both gave the gini score in range 0.005-0.01.

7) Comparison of all Algorithms

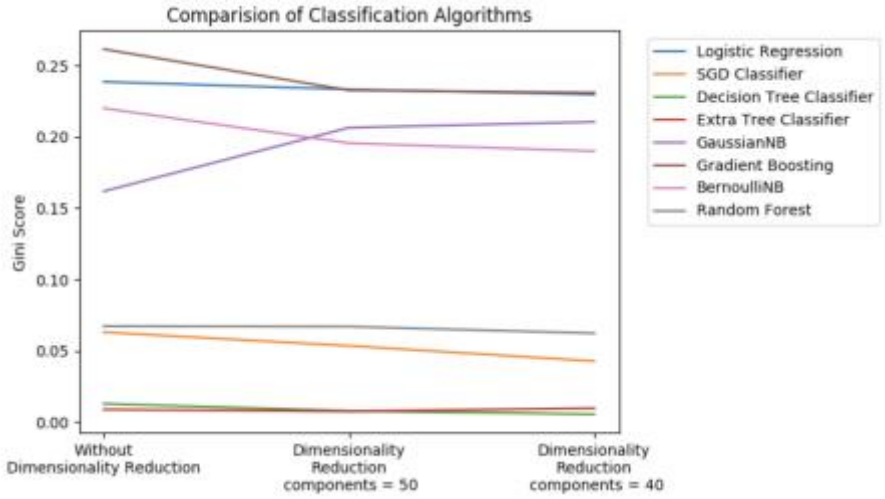


Figure 2 Comparison of all Algorithm

8) Validation

We used K fold cross validation with K=5 for validation of the results. We split the dataset as 70% training and 30% testing which was split for every fold of the validation. We took the mean of all 5 folds and used that result as our final answer for that particular model.

Analysis of results

- With the help of results and the evaluation graphs we can say that the Logistic Regression can provide good results for solving this kind of classification problems.
- Naive Bayes algorithms like Bernoulli Naive Bayes and Gaussian Naive Bayes give slightly lower scores than Logistic Regression.
- Gradient Boosting performs way better than the Logistic Regression and Naive Bayes Classifiers.
- SGD Classifier, Extra Trees Classifier, Random Forest Classifier and Decision Trees Classifier give very low score compared to Logistic Regression and Naive Bayes algorithms.
- We can also see that performing dimensionality reduction does not guarantee an increase in the score. As we saw it depends on the algorithm and model.

Ch.4 Discussion & Conclusions

Decisions made

- We decided to drop some of the columns which featured higher number of missing values. Our intuition was that since majority of the rows would show the value to be missing it would not give us any better results.
- We used cross validation techniques and took the average of the results.
- We converted the columns which contained categorical values into dummy columns by using the pandas get_dummies function and then dropped the original columns.
- We saw that applying dimensionality reduction techniques doesn't offer an improvement over all algorithms but some algorithms benefit by reducing the number of dimensions to 40. We used TruncatedSVD to apply dimensionality reduction.
- We decided to train the model with 70% of the dataset and used 30% for testing as we were limited to 5 attempts per day with the test data from Kaggle.

Difficulties faced

- The dataset had a large amount of class imbalance with only 3.61% actual claims. This made it difficult to test normal techniques.
- The number of rows in the dataset caused memory errors when testing some algorithms. This caused us to test some alternate techniques like sampling to test the algorithms on a smaller scale.
- We had difficulty in setting up ensemble algorithms which contain a stacker algorithms and different levels of models.

Things that worked well

- Data preprocessing gave us much better results and also allowed us to understand the data more.
- We received better results when we handled missing values properly by dropping 3 columns which showed the highest number of values.
- Converting the categorical values to dummy columns also gave us better results.
- We were able to save the model and its parameters on disk as a pickle store and were able to reuse it later for testing it. This saved us a lot of time.

Things that didn't work well

- We tried to sample the dataset by taking a smaller part of it and training the algorithm based on that data. This did not work very well as we did not maintain the original class proportion in the sampled data. On taking sampled data by accounting for the imbalanced classes we got better results but not by a lot.
- We also tried to use oversampling techniques which try to increase the proportion of the classes using the SMOTETomek algorithm which showed even better results but still not as good as the original dataset.
- In the end we decided to use the whole dataset for training which required much time but we found a solution for it later.
- We tried several algorithms like KNearestNeighbours, AdaBoostClassifier, LinearSVC, MLPClassifier and more but did not get satisfactory results. Some algorithms like KNearestNeighbours took a really long time to run with the size of the dataset so we choose not to select them

Conclusion

- We tried to explore the dataset and see relationships between features. Data exploration also helped us in finding out features which have missing values and categorical columns which could be converted to binary columns.
- We learned how to handle a highly imbalanced dataset and the techniques employed to apply classification problems based on it. We also learned about the

Gini evaluation metric and why the normalized Gini metric is used in dataset which has high imbalanced classes.

- We concluded that gradient boosting classifier performed the best after preprocessing data and applying cross validation with logistic regression coming second.

Ch.5 Project Plan / Task Distribution

Porto Seguro's Safe Driver Prediction	
Task	**Responsibility**
Dataset Selection	All
Data Exploration	All
Data Preprocessing	All
Research on Classification Algorithms	All
Logistic Regression	Shrey, Vansh
SGDClassifier	Salmaan, Shrey
GaussianNB	Salmaan, Shrey
BernoulliNB	Vansh, Salmaan
Extra Tree Classifier	Salmaan, Vansh
Decision Tree Classifier	Vansh, Shrey
Random Forest Classifier	Salmaan, Shrey
Gradient Boosting Classifier	Shrey, Salmaan
Documentation	All

References

1. https://www.kaggle.com/c/porto-seguro-safe-driver-prediction
2. http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
3. http://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html
4. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
5. https://www.kaggle.com/c/ClaimPredictionChallenge/discussion/703
6. https://www.kaggle.com/bertcarremans/data-preparation-exploration
7. https://www.kaggle.com/batzner/gini-coefficient-an-intuitive-explanation
8. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

9. http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
10. http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html
11. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html
12. http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
13. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
14. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
15. http://contrib.scikit-learn.org/imbalanced-learn/stable/combine.html