**Group Members:**

**Kaggle Id: rsv**

**Vasu Chaudhary –** [**vsc3**](mailto:vsc3@illinois.edu)

**Sanket Sinha -** [**ssinha27**](mailto:ssinha27@illinois.edu)

**Roshini Seshadri –** [**roshini3**](roshini3@illinois.edu)

**Introduction**

The goal of the project is to prognosticate the risk level (response variable) in providing insurance to new clients based on the historical data of various clients. One of the main challenges that we faced was cleaning the dataset due to excessive presence of null values. Secondly, trying out different algorithms to maximize the accuracy in predicting the response variable. After trying out various combinations of the preprocessing techniques and classification techniques, we were able to narrow down to Random Forest Classification, which gave us the maximum accuracy in predicting the response variable.

**Related work**

In the paper Random Decision Forests [1] by Tin Kam Ho, it is evident that the accuracy ofRandom Forests algorithm is better compared to using just decision trees which overfits the training data. Also, it highlights the fact that with a better choice of training data, the classifier can accommodate high accuracy on large datasets because it combines the results of multiple decision trees instead of depending on the output of a single decision tree.

**Implementation**

We used python’s scikit-learn machine learning library both for preprocessing and classifying the data.

**Preprocessing**- Due to presence of missing values in the dataset, it is unsuited to be passed directly to sklearn’s classifiers. Firstly, the training and testing set were concatenated in a pandas data frame. Initially, we performed techniques like removing columns which has more than 50% of null values and replacing the null values in continuous features with mean and categorical features with mode. However, this does not give us favorable results. So, we stuck with just converting all the columns like Product\_Info\_2 which are of object(text) type into categorical data. Apart from this, we used sklearn’s Imputer. It imputes the missing values (NaN) across all the columns with the mean (default) value.

**Classification** – sklearn’s Random Forest Classifier is used with parameters below:

* n\_estimators = 150 (No. of trees in the forest)
* criterion='gini' (Default value)
* verbosity = 1(A parameter which returns information about the tree building process, doesn’t contribute to the increase or decrease in accuracy)
* max\_features = ‘sqrt’ (The number of features each tree is randomly allocated)
* max\_depth = 50
* n\_jobs = -1 (Run the tasks in parallel on par with the number of cores present in your machine. This parameter enhances the performance.)

**Results**

Before narrowing down our approach we tried out the below algorithms:

1. Naïve Bayes

One of the main disadvantages of the approach is that it naively assumes the independence among variables.

0.43

1. Decision Trees

Disadvantageous because it over fits the data.

0.45

1. SVM

SVM technique does not directly predict the probabilities it performs an expensive cross validation.

Score: 0.41

1. Random Forests

It reduces over fitting of data and that makes it accurate.

0.53

**Work Distribution**

Vasu Chaudhary: Implementation of RandomForest and preprocessing

Sanket Sinha: Data Cleaning techniques and Imputer, SVM

Roshini Seshadri: Decision Trees and Naïve Bayes

**References:**

[1] Tin Kam Ho, "Random decision forests," *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Montreal, Que., 1995, pp. 278-282URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=598994&isnumber=13183>

[2] scikit-learn library: http://scikit-learn.org/stable/