Executive Summary

Sairam Tangirala

Data sets

Train data

- 40,000 rows and 101 columns
- data set contains 1 target and 100 predictors which are a mix of float64(86), int64(3), object(12) datatypes

Test data

- 10,000 rows and 100 columns

Output Files

- "glmresults.csv" and "nonglmresults.csv" containing 10,000 rows. Each file has a single column representing the predicted probabilities (for label 1) for Logistic Regression (glm) and Random Forest (nonglm) models

Exploratory data analysis

Pre-Processing

- truncating weekday name in x3 to first 3 letters
- In 'x7', removing % sign and converting to numeric values
- In 'x19', removing ($ ##) and converting to numeric values with - sign
- Using histogram of predictors (Fig. 1),
  - Deleting 'x39' column as it has only one value = 5-10 miles
  - Deleting 'x58' column as it has a very high peak at 300.63 and its variation does not seem to affect the target
  - Deleting 'x67' column as it has a very high peak at 14.40 and its variation does not seem to affect the target
  - Deleting 'x71' column as it has a very high peak at -0.00023 and its variation does not seem to affect the target
  - Deleting 'x75' column as it has a very high peak around 0 and its variation does not seem to affect the target
  - Deleting 'x84' column as it has a very high peak at 2.13 and its variation does not seem to affect the target
  - Deleting 'x98' column as it is binary 0/1 and its variation does not seem to affect the target

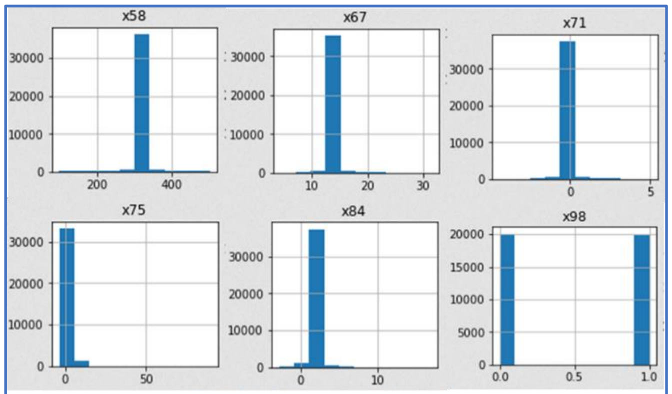


Fig.1: Select histograms of data columns

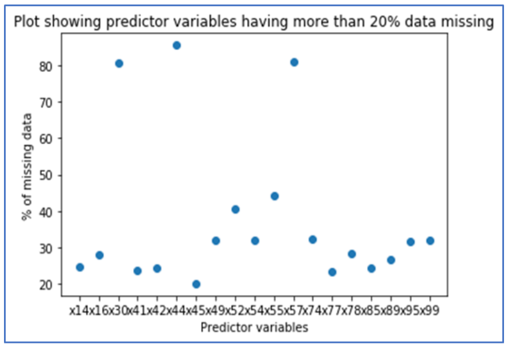


Fig. 2: Predictors missing more than 20% data

Removing 19 columns with > 20% missing data (Fig. 2)

- Identified 19 predictor data columns that have more than 20% data missing, I removed these columns from further analysis
  - Following data columns were deleted: 'x14', 'x16', 'x30', 'x41', 'x42', 'x44', 'x45', 'x49', 'x52', 'x54', 'x55', 'x57', 'x74', 'x77', 'x78', 'x85', 'x89', 'x95', 'x99'

Removing 1 (gender) categorical data (Fig. 3)

- Data column 'x24' was deleted as target was dependent on this gender category data

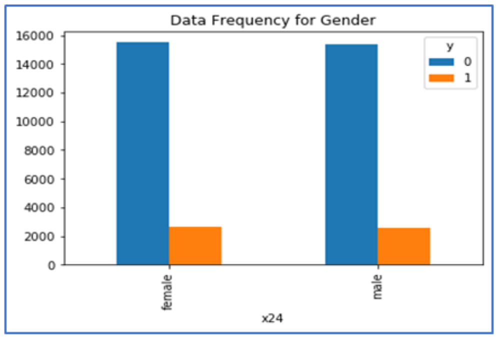
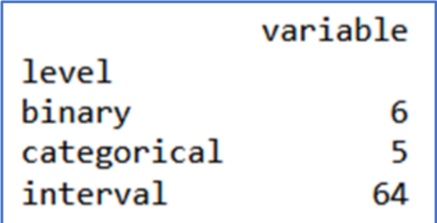


Fig. 3: No affect of Gender category on 'y'

Fig.4: Metadata dictionary generation

- binary: columns having 0/1, yes/no, male/female data, etc.

- categorical: columns having car make, month name, weekday, etc.
- interval: columns having float data types
- Used in imputing missing data



```
                 variable
level
binary                  6
categorical             5
interval               64
```

Fig. 4: Meta Data information

Imputing NA values for numeric data columns using mean()

- For 64 numeric (interval) data columns, all NA (missing values) were imputed using mean() of remaining rows

Categorical variables effect on target

- For each categorical variable, the sensitivity of target to itself was plotted (Fig. 5)
- Conclusion: only gender has no effect on the target and may be removed from further analysis

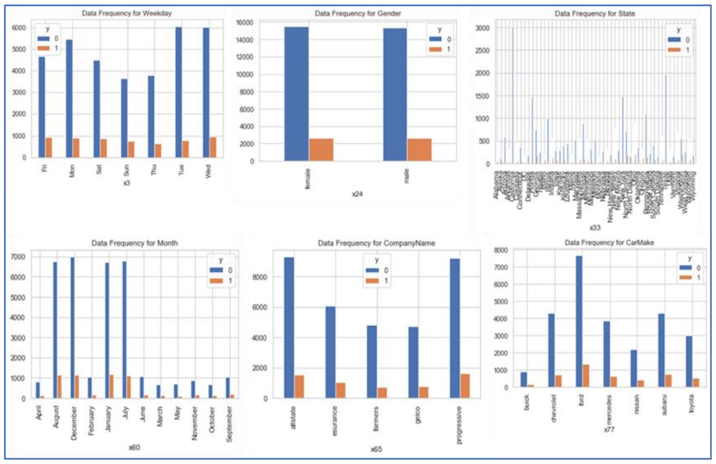


Fig.5: Sensitivity of target to categorical predictors

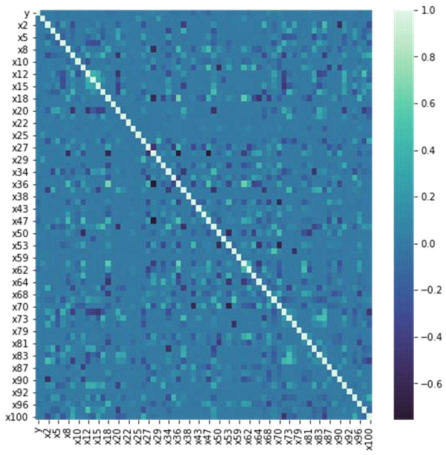


Fig.6: Correlations between a pair of columns

Fig. 6: Two-point correlations among numeric data columns

- I created a subset dataframe (numeric_df) that contains numerics only ('int16', 'int32', 'int64', 'float16', 'float32', 'float64') only and plotted correlations between a pair of numeric data columns
    - I did not observe strong correlations between any pairs of numeric data columns

Fig. 7: Removing 17 data columns using Multicollinearity calculations

- Even though, no two of the predictors are highly correlated with each other, I checked the possibility of one predictor being correlated with rest of the predictors. See Fig. 7
- There are 17 predictors having relatively high Variance Inflation Factor (VIF) scores
- In model development, I drop 17 predictors that have large VIF scores

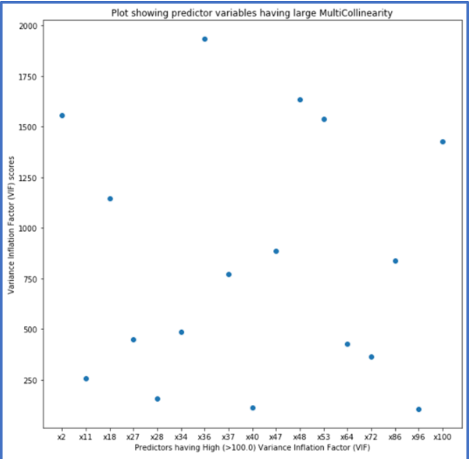


Fig. 7: Multicollinearity of Predictors

Fig. 8 Dataset after pre-processing

- Fig. 8 shows the heatmap of clean pre-processed data without any NAs

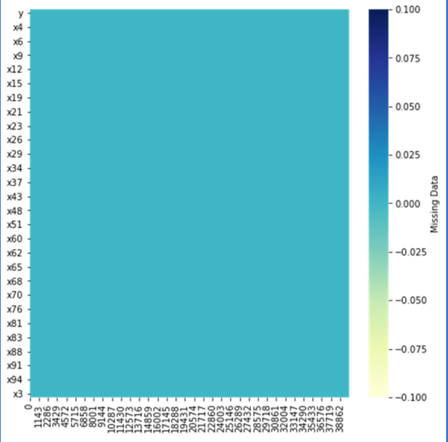


Fig. 8: Clean pre-processed dataset

Train data preparation

One-Hot Encoding (OHE)of 6 categorical data columns

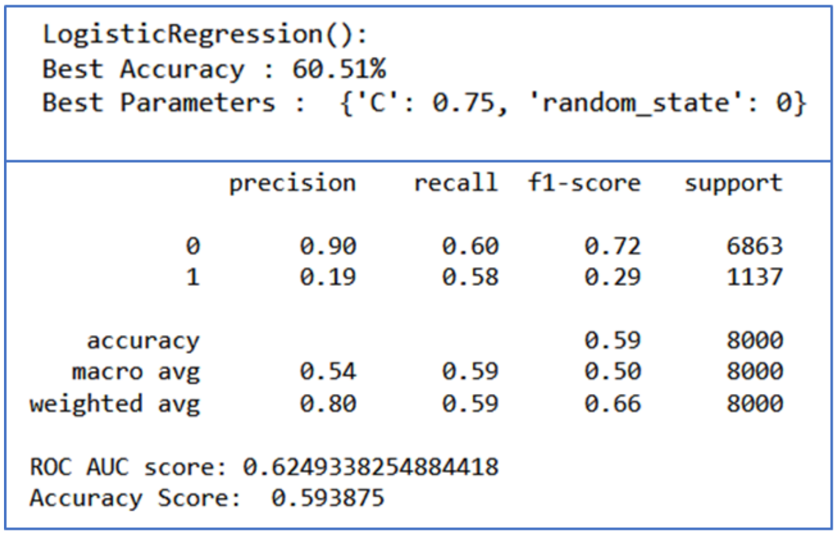
- The following data columns were one-hot-encoded,
    - x3: weekdays
    - x60: months
    - x65: insurance providers
- OHE vectors were added to cleaned dataset and corresponding categorical data columns were dropped

Oversampling to get balanced dataset

- The distribution of data columns in the target class is skewed (imbalanced data), I performed SMOTE oversampling to balance the data

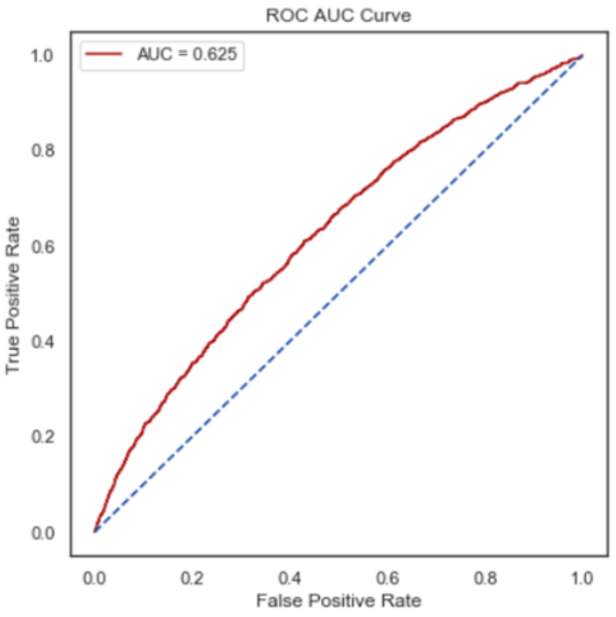
ML Models

Logistic Regression

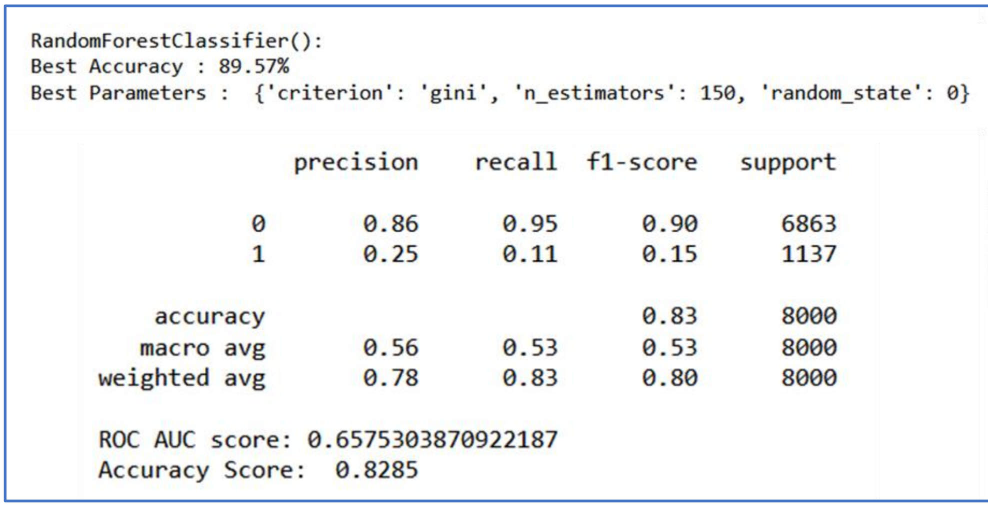


```
 LogisticRegression():
 Best Accuracy : 60.51%
 Best Parameters :  {'C': 0.75, 'random_state': 0}
```

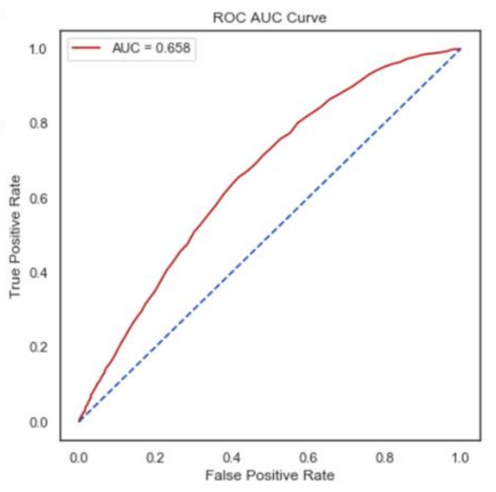
```
              precision    recall  f1-score   support

           0       0.90      0.60      0.72      6863
           1       0.19      0.58      0.29      1137

    accuracy                           0.59      8000
   macro avg       0.54      0.59      0.50      8000
weighted avg       0.80      0.59      0.66      8000

ROC AUC score: 0.6249338254884418
Accuracy Score:  0.593875
```



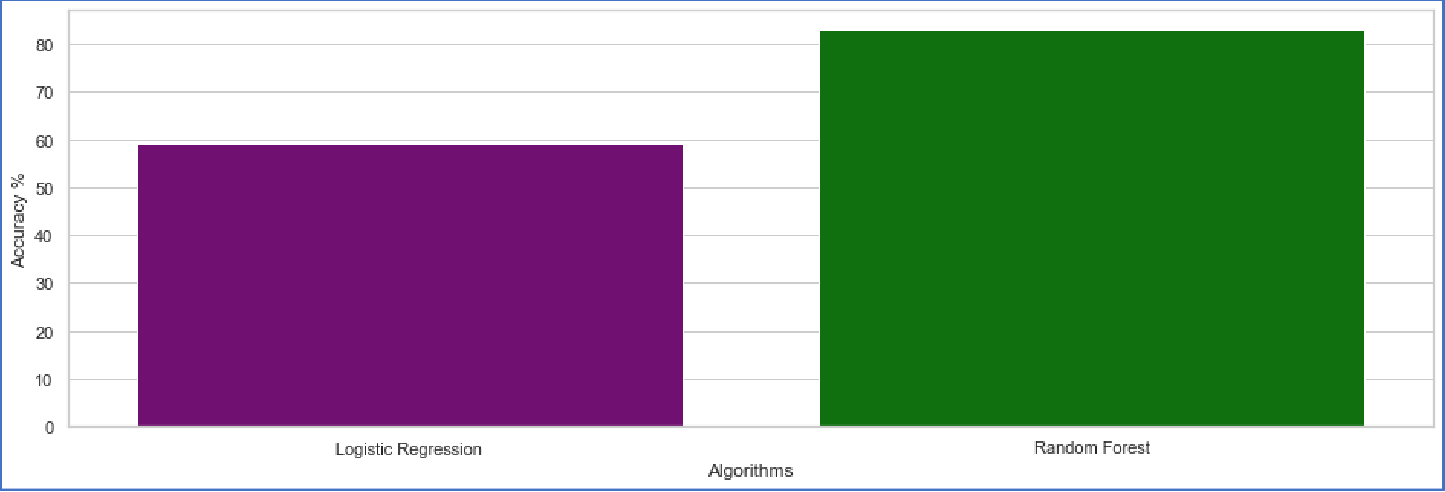
Random Forest

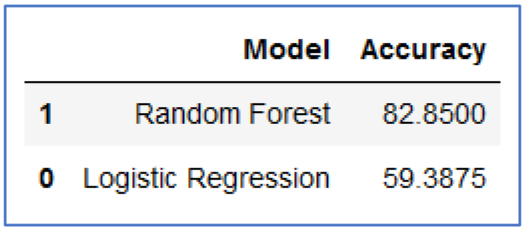


```
RandomForestClassifier():
Best Accuracy : 89.57%
Best Parameters :  {'criterion': 'gini', 'n_estimators': 150, 'random_state': 0}

              precision    recall  f1-score   support

           0       0.86      0.95      0.90      6863
           1       0.25      0.11      0.15      1137

    accuracy                           0.83      8000
   macro avg       0.56      0.53      0.53      8000
weighted avg       0.78      0.83      0.80      8000

ROC AUC score: 0.6575303870922187
Accuracy Score:  0.8285
```



Model Comparisons





	Model	Accuracy
1	Random Forest	82.8500
0	Logistic Regression	59.3875

In comparing logistic regression and random forest models, the later performed better for me. Random Forest is based on the Ensemble Learning technique called bagging. It creates as many trees on the subset of the data and combines the output of all the trees. In this way it reduces overfitting problem encountered in decision trees and also reduces the variance and improves the accuracy. As given data set contains several categorical predictors, this algorithm works well with both categorical and continuous variables. By its working, this approach also handles outliers.

Even for training 80% of the 40K dataset, my PC seemed to take a longer training time compared to other models. It is expected: Random Forest require much more time to train as compared to decision trees as it generates a lot of trees (instead of one tree in case of decision tree) and makes decision on the majority of votes.

Which algorithm may perform better?

I tested different models in this exercise, I did not prepare a NN model as I did not have time and additionally, I may not prefer NN model for this data set that deals with insurance related dataset. This is because, the NNs are not explainable and not easy for me to tune efficiently. However, I think, neural network model may perform better on this data set. The reasons for it are

- NNs can work with many different data types and in this exercise, I was given a variety of data columns
- I was able to easily scale the data, fill missing values. Data preprocessing for Neural Networks requires filling missing values and converting categorical data into numerical. There is a need for feature scaling. In the case of different ranges of features, there may be problems with model training. If you don't scale features into the same ranges then features with larger values will be treated as more important in the training, which is not desired.
- Care needs to be taken to make sure that the gradients values don't explode and the neurons can saturate which will make it impossible to train NN

Business partner conversation

I would put across a business advantage of Random Forest model that it offers

- ease of use, flexibility and explain-ability to users
- it handles both classification and regression problems efficiently and existing model knowledge may be used in extending to other data sets
- As multiple decision trees form an ensemble in the random forest model, they predict more accurate results, particularly when the individual trees are uncorrelated with each other