Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

CS 559 – Machine learning Final Project Report

Submission by:

Vidya Sagar Polaki

CWID: 10430970

vpolaki1@stevens.edu

Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

Table of Contents

Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

1. Project at a glance:

1.1. Purpose

Nothing ruins the thrill of buying a brand-new car more quickly than seeing your new insurance bill. The sting's even more painful when you know you're a good driver. It doesn't seem fair that you must pay so much if you've been cautious on the road for years.

Inaccuracies in car insurance company's claim predictions raise the cost of insurance for good drivers and reduce the price for bad ones

1.2. Business Need

To allow the auto insurance companies to further tailor their prices, and hopefully make auto insurance coverage more accessible to more drivers.

1.3. Project Lifecycle

- Start Date: November 10, 2018
- Anticipated End Date: December 8, 2018
- Total Duration: 28 days

1.4. Software dependencies

- Python 3
- Anaconda 3
- Pandas
- Matplotlib
- Seaborn
- NumPy
- Scikit-learn
- XG Boost

2. Motivation:

As of 2015 there were over 268 million registered vehicles on the roads in the United States. Of those millions of registered vehicles, each year there are also millions of vehicle crashes. In 2015, there were 32,166 fatalities, 1,715,000 injuries and 4,548,000 car crashes which involved property damage. Of these fatalities, there are far

more driver deaths, than passenger, pedestrian or motorcyclist deaths. So, while many of us feel secure in our vehicles, the statistics indicate the importance of automobile insurance and in most cases, auto insurance is required by law. Auto insurance is important because it not only covers any physical damage that may occur in an accident, but also any damage or injury that might be caused because of a vehicular accident or which may be done upon oneself or one's vehicle by another vehicle or accident – a falling tree for example.

Of course, auto insurance comes at a cost to the car owner and the younger the owner the more expensive car insurance typically is. In 2016, the average cost of car insurance for 19 year old males was around 2,154 U.S. dollars and for 19 year old females it was around 1,930 U.S. dollars. The price of car insurance drops significantly as people get older and the gap in price between the genders also levels out.

A machine learning model to predict whether a driver files claim in a year after taking the insurance can classify the driver as good or bad driver based on results. Once the companies have an idea about whom they are insuring they can tailor their policy based on driver stats.

3. Objective:

To build a model that predicts the probability that a driver will initiate an auto insurance claim in one year after taking a new auto insurance policy.

4. Introduction

4.1. Insurance Claim Predictor

This project is based on Kaggle competition "**Porto Seguro's Safe Driver Prediction**" objective of the competition and this project are same, which is to predict the probability that a driver will initiate an auto insurance claim in one year after taking a new auto insurance policy

5. About Dataset

Both test and training datasets are provided and from the provided description its clear that,

Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

- Features that belong to similar groupings are tagged as such in the feature names (e.g., ind, reg, car, calc).
- Feature names include the postfix bin to indicate binary features and cat to indicate categorical features.
- Features without these designations are either continuous or ordinal.
- Values of -1 indicate that the feature was missing from the observation.
- The target column signifies whether a claim was filed for that policy holder.

5.1. Dataset Description

Training dataset: Training dataset has 595212 rows and 59 columns (features).

	id	target	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
0	7	0	2	2	5	1	0	0	1
1	9	0	1	1	7	0	0	0	0
2	13	0	5	4	9	1	0	0	0
3	16	0	0	1	2	0	0	1	0
4	17	0	0	2	0	1	0	1	0

Testing dataset: Testing dataset has 892816 rows and 58 columns (features).

	id	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
0	0	0	1	8	1	0	0	1
1	1	4	2	5	1	0	0	0
2	2	5	1	3	0	0	0	0
3	3	0	1	6	0	0	1	0
4	4	5	1	7	0	0	0	0

After careful observation of the datasets, we can see that the datasets contain

- binary variables
- categorical variables of which the category values are integers
- other variables with integer or float values
- variables with -1 representing missing values
- the target variable and an ID variable

Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

5.2. Data Preprocessing

5.2.1. Handling missing values

We know that missing values are represented as -1

```
Variable ps_ind_02_cat has 74 records with missing values
Variable ps_ind_04_cat has 41 records with missing values
Variable ps_ind_05_cat has 1246 records with missing values
Variable ps_reg_03 has 18720 records with missing values
Variable ps_car_01_cat has 46 records with missing values
Variable ps_car_02_cat has 1 records with missing values
Variable ps_car_03_cat has 73240 records with missing values
Variable ps_car_05_cat has 47407 records with missing values
Variable ps_car_07_cat has 2456 records with missing values
Variable ps_car_09_cat has 126 records with missing values
Variable ps_car_14 has 7973 records with missing values
In total, there are 11 variables with missing values
```

- ps_car_03_cat and ps_car_05_cat are category variables and have a large proportion of records with missing values. Removing these variables.
- For the other categorical variables with missing values, leaving the missing value -1 as such.
- ps_reg_03 (continuous) has 18720 missing values. Replacing then by the mean.
- ps_car_11 (ordinal) has only 5 records with missing values. Replacing by the mode.
- ps_car_12 (continuous) has only 1 record with missing value. Replacing by the mean.
- ps_car_14 (continuous) has 7973 missing values of all records. Replacing by the mean.

5.2.2. Dataset Sampling

From the mean value of target variable, we can clearly say that there are only 3.644% of 1s and 96.356 % of 0s which leaves our data highly unbalanced. To handle this, we must either over sample or under sample the training data. Which can be done as shown

- oversampling records with target=1
- under sampling records with target=0

Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

As our training dataset is large, is good to do under sampling for target = 0. we will drop some rows of target 0 to obtain balance between classes.

After sampling,

```
Number of samples with target 0 573518
Number of samples with target 1 21694
undersampling_rate 0.15130475416639058
number of rows selected 86776
Rate to undersample records with target=0: 0.15130475416639058
Number of records with target=0 after undersampling: 86776
```

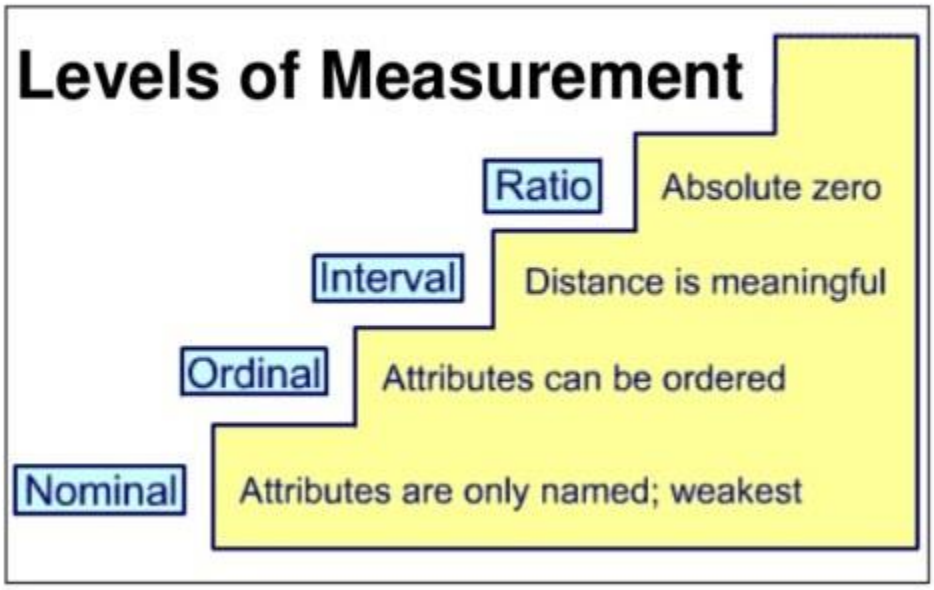
```
print('shape of train dataset after sampling', train.shape)
```

```
shape of train dataset after sampling (108470, 59)
```

```
import numpy as np
print(np.mean(train['target']))
```

```
0.2
```

5.2.3. Dataset Classification into levels



Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

- For ease of data handling, based on the above levels of measurement figure, we classify the data by storing metadata related to data in a separate data frame as shown below so that EDA can be done efficiently.
    - **variable type**: input, ID, target
    - **level**: nominal, interval, ordinal, binary
    - **required**: True or False
    - **datatype**: int, float, str

	role	level	count
0	id	nominal	1
1	input	binary	17
2	input	interval	10
3	input	nominal	14
4	input	ordinal	16
5	target	binary	1

Calculating mean, median and standard deviation on categorical variables so we will find those values for the features of int and float data types

	ps_reg_01	ps_reg_02	ps_reg_03	ps_car_12	ps_car_13	ps_car_14	ps_car_15	ps_calc_01
count	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000
mean	0.610991	0.439184	0.551102	0.379945	0.813265	0.276256	3.065899	0.449756
std	0.287643	0.404264	0.793506	0.058327	0.224588	0.357154	0.731366	0.287198
min	0.000000	0.000000	-1.000000	-1.000000	0.250619	-1.000000	0.000000	0.000000
25%	0.400000	0.200000	0.525000	0.316228	0.670867	0.333167	2.828427	0.200000
50%	0.700000	0.300000	0.720677	0.374166	0.765811	0.368782	3.316625	0.500000
75%	0.900000	0.600000	1.000000	0.400000	0.906190	0.396485	3.605551	0.700000
max	0.900000	1.800000	4.037945	1.264911	3.720626	0.636396	3.741657	0.900000

There are 10 features of float datatype

- ps_reg_03, ps_car_12 and ps_car_15 have missing values in float types

Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

	ps_ind_01	ps_ind_03	ps_ind_14	ps_ind_15	ps_car_11	ps_calc_04	ps_calc_05
count	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000
mean	1.900378	4.423318	0.012451	7.299922	2.346072	2.372081	1.885886
std	1.983789	2.699902	0.127545	3.546042	0.832548	1.117219	1.134927
min	0.000000	0.000000	0.000000	0.000000	-1.000000	0.000000	0.000000
25%	0.000000	2.000000	0.000000	5.000000	2.000000	2.000000	1.000000
50%	1.000000	4.000000	0.000000	7.000000	3.000000	2.000000	2.000000
75%	3.000000	6.000000	0.000000	10.000000	3.000000	3.000000	3.000000
max	7.000000	11.000000	4.000000	13.000000	3.000000	5.000000	6.000000

There are 16 int datatypes that are neither binary or categorical values

- Only ps_car_11 has missing values

	target	ps_ind_06_bin	ps_ind_07_bin	ps_ind_08_bin	ps_ind_09_bin	ps_ind_10_bin	ps_ind_11_bin
count	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000
mean	0.036448	0.393742	0.257033	0.163921	0.185304	0.000373	0.001692
std	0.187401	0.488579	0.436998	0.370205	0.388544	0.019309	0.041097
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

There are 18 features that are binary with target variable included.

5.2.4. Categorical variables

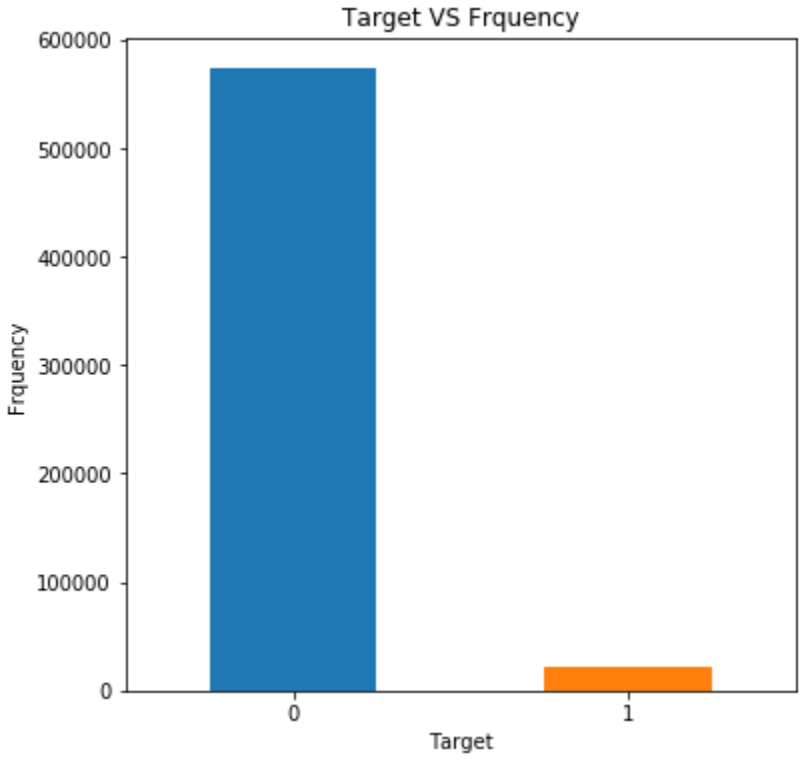
```
Variable ps_ind_02_cat has 5 distinct values
Variable ps_ind_04_cat has 3 distinct values
Variable ps_ind_05_cat has 8 distinct values
Variable ps_car_01_cat has 13 distinct values
Variable ps_car_02_cat has 3 distinct values
Variable ps_car_04_cat has 10 distinct values
Variable ps_car_06_cat has 18 distinct values
Variable ps_car_07_cat has 3 distinct values
Variable ps_car_08_cat has 2 distinct values
Variable ps_car_09_cat has 6 distinct values
Variable ps_car_10_cat has 3 distinct values
Variable ps_car_11_cat has 104 distinct values
```

Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

6. Exploratory Data Analysis

EDA was performed on the entire Training dataset.

6.1. Target class distribution

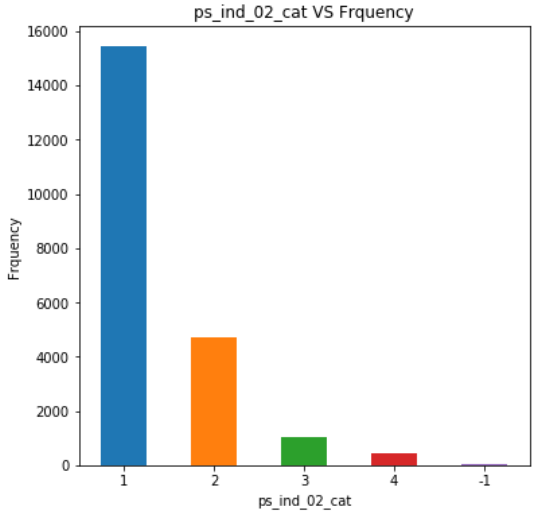


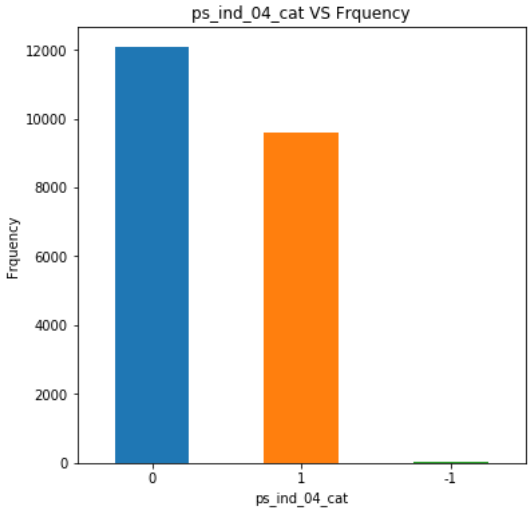
A new data frame is created where target variable is equal to 1 which means insurance is claimed so that we can better understand the patterns for claims.

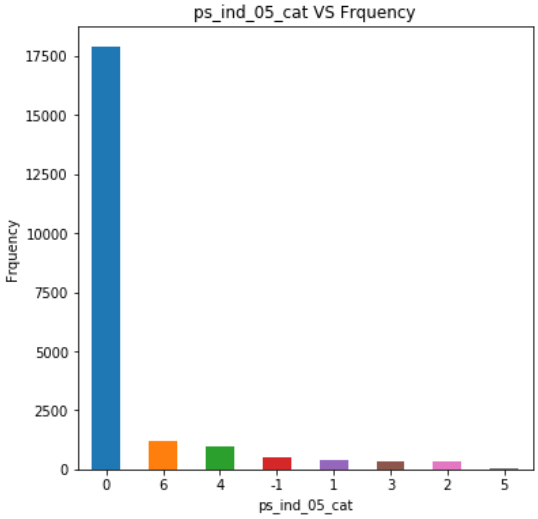
6.2. Categorical variable plots

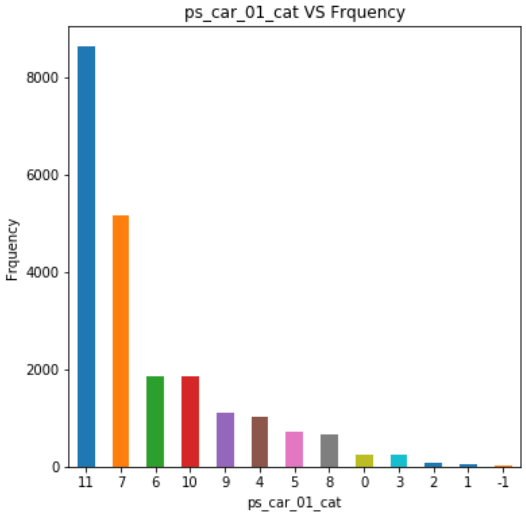
Various plots from categorical variables are plotted as shown below

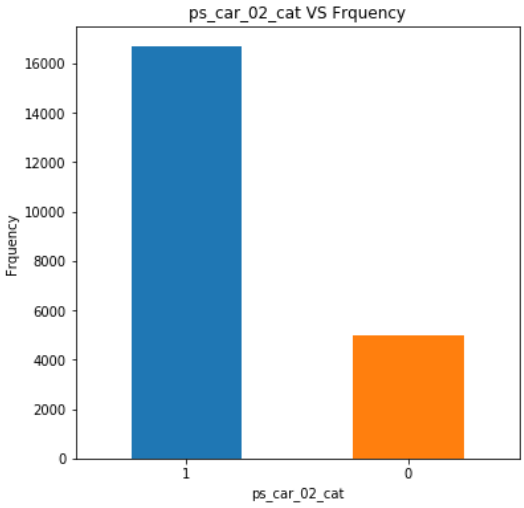
Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

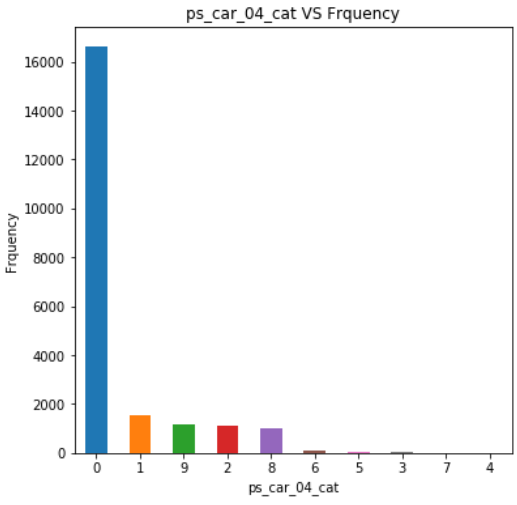




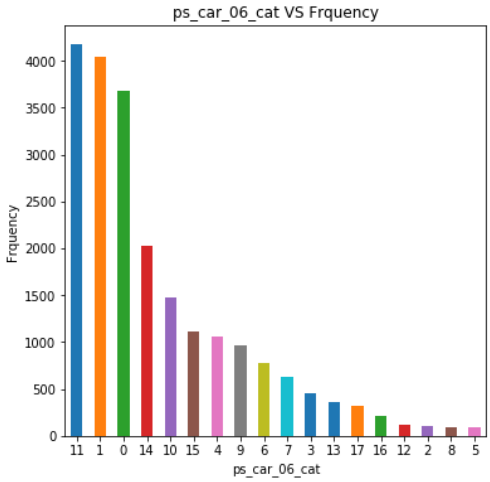


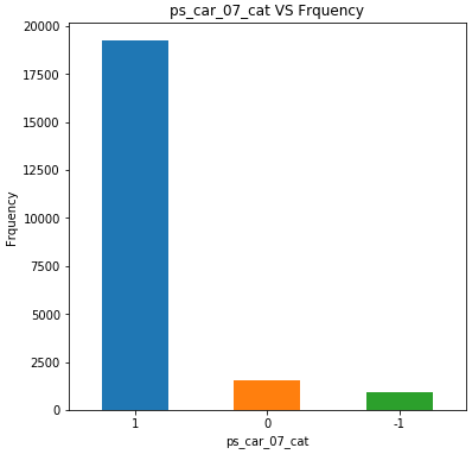


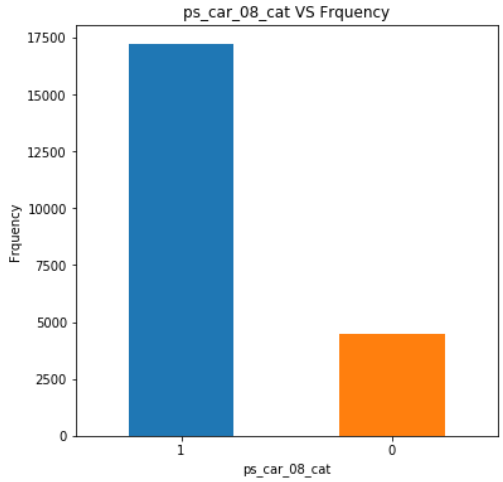


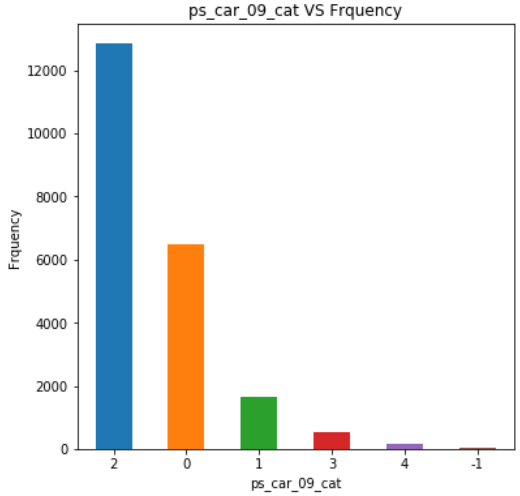


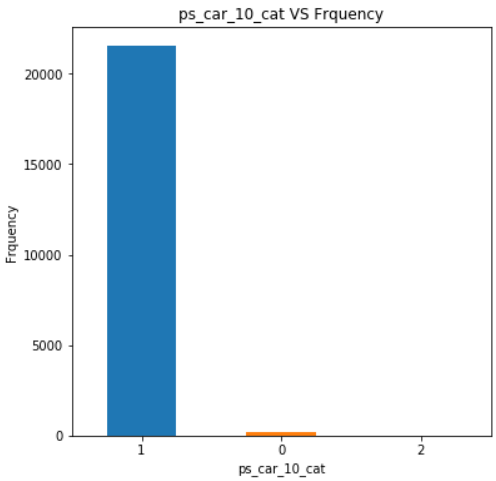
Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu







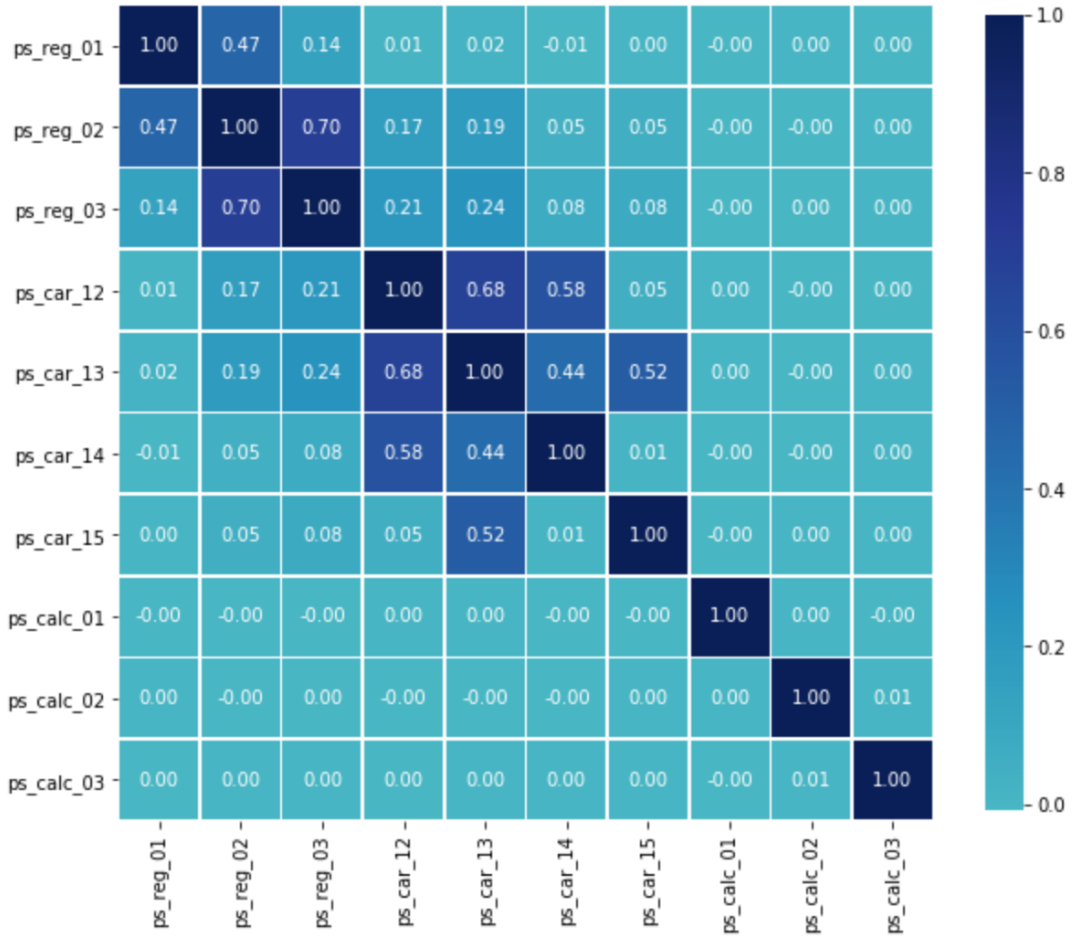




Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

From the above plots we can clearly say that missing values have effect on filing an insurance claim. And most of the categories its value 1 dominant over others.

6.3. Correlation between float data types

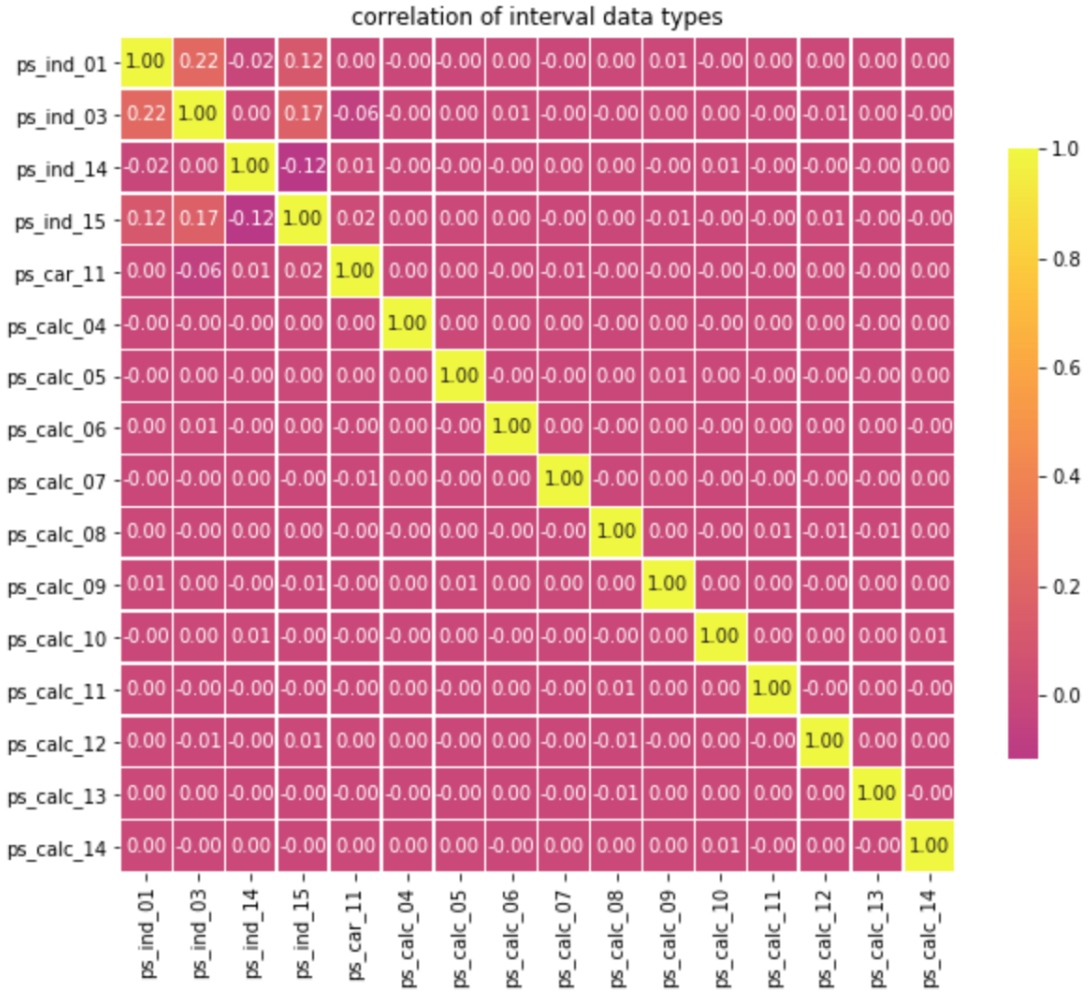


There is a strong correlation between some variables:

- ps_reg_02 and ps_reg_03 (0.7)
- ps_car_12 and ps_car13 (0.68)
- ps_car_12 and ps_car14 (0.58)
- ps_car_12 and ps_car14 (0.52)

Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

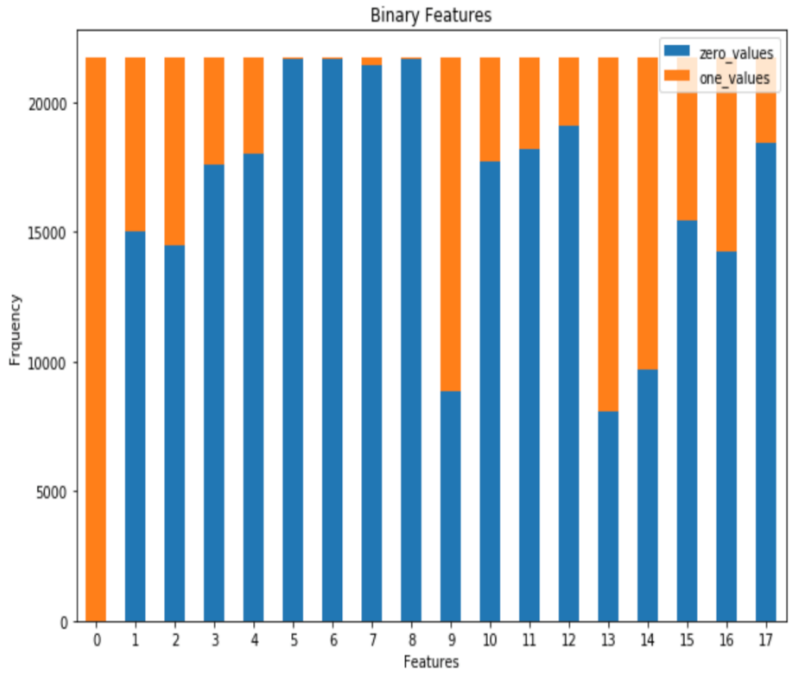
6.4. Correlation between int data types



We can see that no strong correlations are present, and some negative correlations are present as well

Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

6.5. Binary data types plot:



	Feature
0	target
1	ps_ind_06_bin
2	ps_ind_07_bin
3	ps_ind_08_bin
4	ps_ind_09_bin
5	ps_ind_10_bin
6	ps_ind_11_bin
7	ps_ind_12_bin
8	ps_ind_13_bin
9	ps_ind_16_bin
10	ps_ind_17_bin
11	ps_ind_18_bin
12	ps_calc_15_bin
13	ps_calc_16_bin
14	ps_calc_17_bin
15	ps_calc_18_bin
16	ps_calc_19_bin
17	ps_calc_20_bin

7. Models

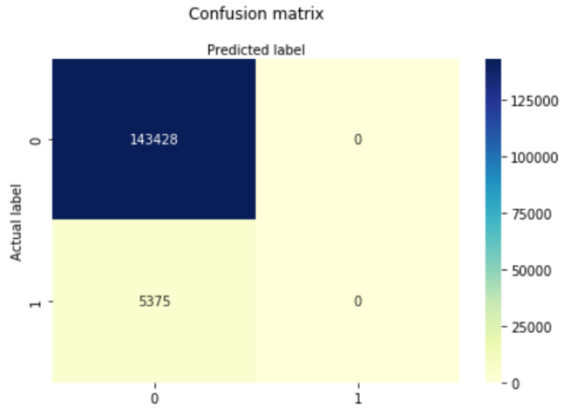
7.1. Algorithms

7.1.1. Logistic Regression

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature

- Logistic regression on the entire dataset without sampling (under sampling or oversampling)

Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu




```
Accuracy: 0.9638784164297762
Precision: 0.0
Recall: 0.0
```

Clearly, we can see that even though the accuracy is very high, precision and recall are very less. So there is a need for sampling

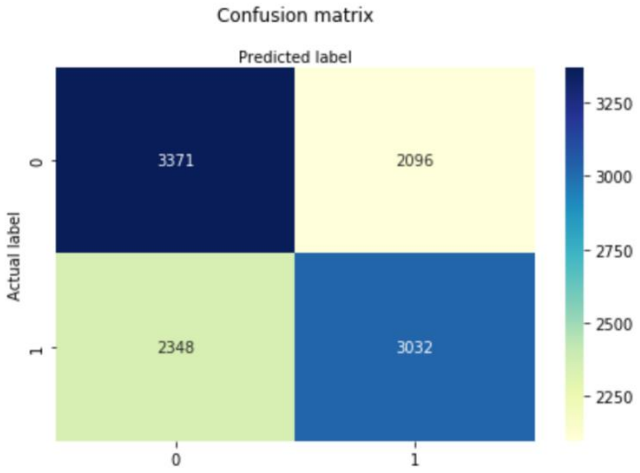
After running a loop for sample sizes, we obtained

```
(216940, 59)
Accuracy is:  0.8992716880243385 when split ratio is 0.1
Precision is:  0.25 when split ratio is 0.1
Recall is:  0.00018311664530305805 when split ratio is 0.1

(108470, 59)
Accuracy is:  0.8016815399365735 when split ratio is 0.2
Precision is:  0.3695652173913043 when split ratio is 0.2
Recall is:  0.009546986147510296 when split ratio is 0.2

(72313, 59)
Accuracy is:  0.7045743680513302 when split ratio is 0.3
Precision is:  0.5150078988941548 when split ratio is 0.3
Recall is:  0.060820895522388056 when split ratio is 0.3

(54234, 59)
Accuracy is:  0.62607861936721 when split ratio is 0.4
Precision is:  0.5716259788116076 when split ratio is 0.4
Recall is:  0.23062627764356067 when split ratio is 0.4

(43388, 59)
Accuracy is:  0.5903014658430903 when split ratio is 0.5
Precision is:  0.5912636505460218 when split ratio is 0.5
Recall is:  0.5635687732342007 when split ratio is 0.5

(36156, 59)
Accuracy is:  0.614669764354464 when split ratio is 0.6
Precision is:  0.6258056030514271 when split ratio is 0.6
Recall is:  0.8817642698295033 when split ratio is 0.6

(30991, 59)
Accuracy is:  0.6938564790913784 when split ratio is 0.7
Precision is:  0.6946833463643471 when split ratio is 0.7
Recall is:  0.994589552238806 when split ratio is 0.7
```

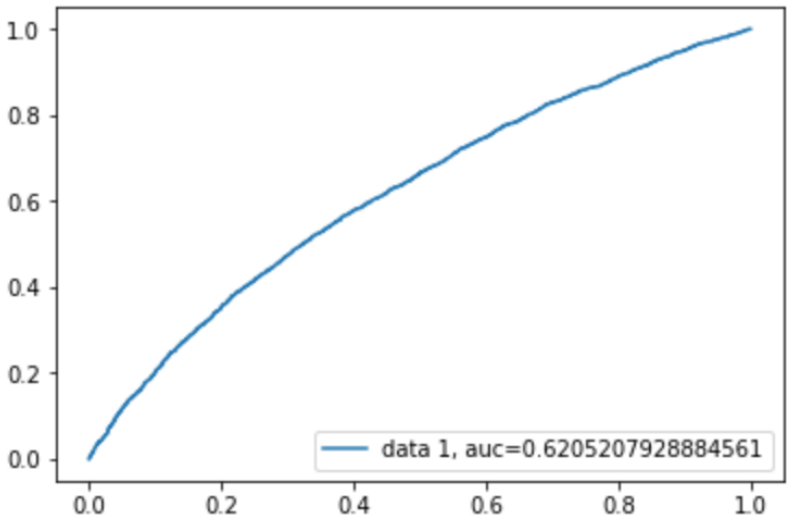
Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

```
(27117, 59)
Accuracy is:  0.7952802359882006 when split ratio is 0.8
Precision is:  0.7952802359882006 when split ratio is 0.8
Recall is:  1.0 when split ratio is 0.8

(24104, 59)
Accuracy is:  0.8941254563557915 when split ratio is 0.9
Precision is:  0.8941254563557915 when split ratio is 0.9
Recall is:  1.0 when split ratio is 0.9
```

Clearly, we can see that model is predicting all outcomes as zeros when class ratio is too low and it is predicting all values as one when class ratio is high. So, we will settle for an average ratio which is 0.5 and the results are as shown




```
Accuracy: 0.5903014658430903
Precision: 0.5912636505460218
Recall: 0.5635687732342007
```



Gini = 2 (AUC) -1 = (2 * 0.6205) – 1 = 0.25

Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

7.1.2. Feature selection using Random forest

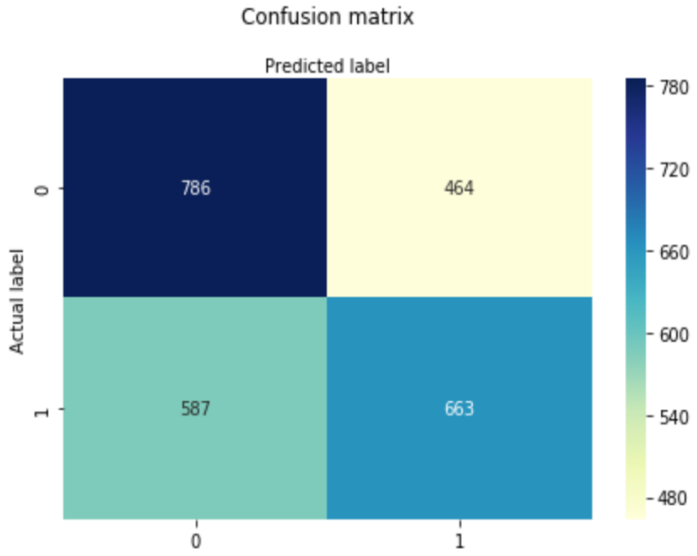
Selected top 10 features based on feature importance score and saved them as a separate dataset for further analysis and modeling.

	id	target	ps_car_11	ps_reg_01	ps_car_12	ps_car_09_cat	ps_ind_13_bin	ps_ind_01	ps_calc_08	ps_calc_12	ps_car_15	ps_calc_09
0	494228	0	3	0.0	0.316228	0	0	2	10	0	3.464102	4
1	549468	0	2	0.6	0.424264	3	0	1	6	0	2.000000	0
2	695518	0	2	0.8	0.400000	0	0	0	11	0	3.464102	2
3	1237762	0	2	0.3	0.316228	2	0	0	10	1	2.828427	3
4	664144	0	2	0.9	0.374166	2	0	0	7	2	3.741657	3

Features with top importance score are selected and further analysis is carried on these features.

7.1.3. Support Vector Machine (SVM)

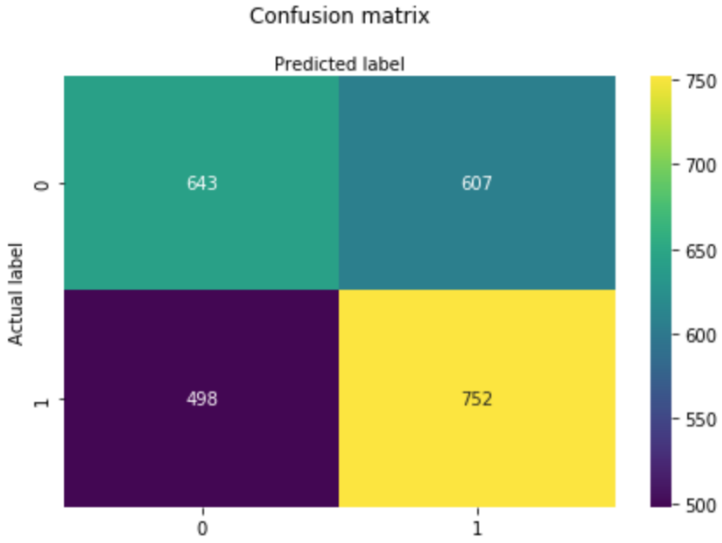
7.1.3.1 Linear Kernel




```
SVM with linear kernel Accuracy: 0.5632
SVM with linear kernel Precision: 0.5572463768115942
SVM with linear kernel Recall: 0.6152
AUC with linear kernel Recall: 0.5632
```

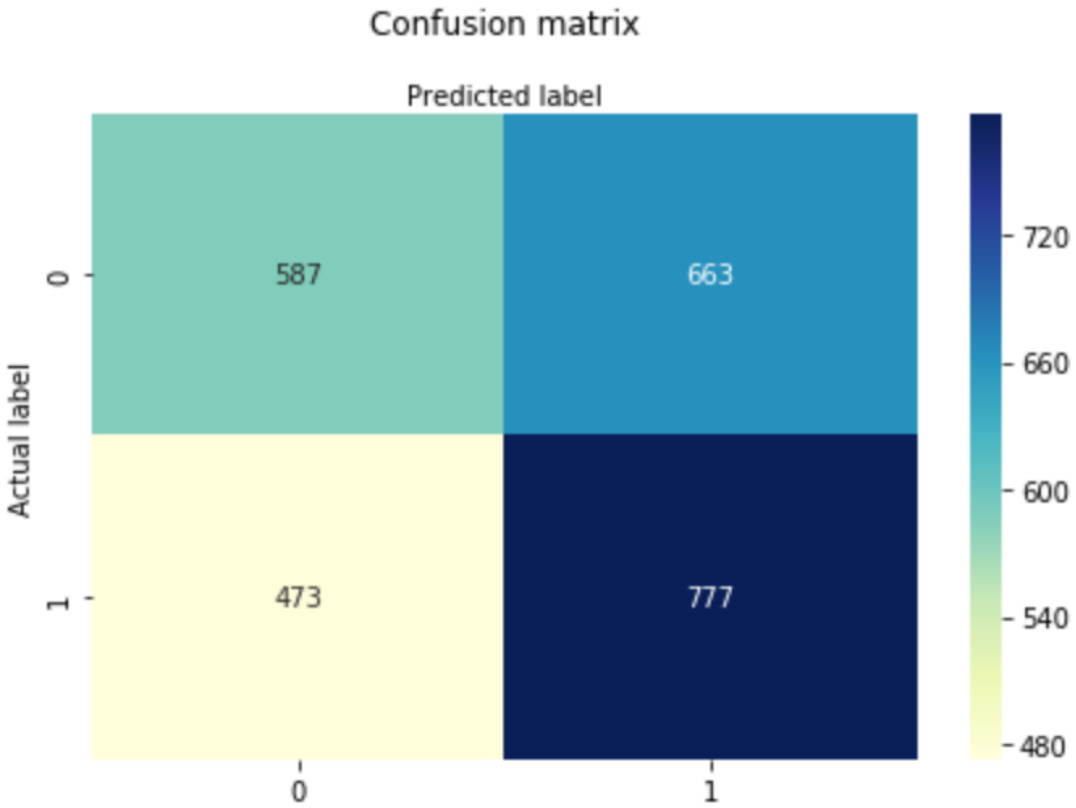
7.1.3.2 Polynomial Kernel

Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu




```
SVM with polynomial kernel Accuracy: 0.558
SVM with polynomial kernel Precision: 0.5533480500367918
SVM with polynomial kernel Recall: 0.6016
AUC with polynomial kernel Recall: 0.5579999999999999
```

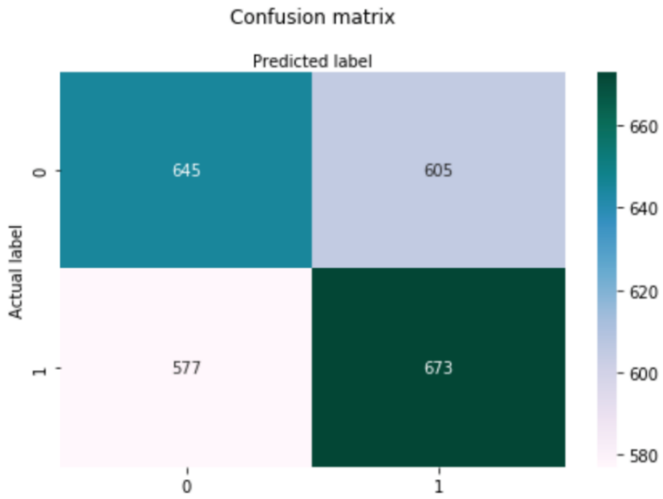
7.1.3.3 Radial Bias Function Kernel (RBF)



Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

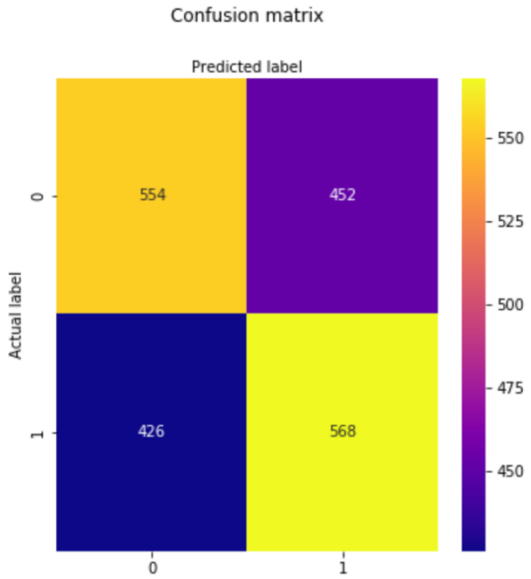
```
SVM with RBF kernel Accuracy: 0.5456
SVM with RBF kernel Precision: 0.5395833333333333
SVM with RBF kernel Recall: 0.6216
AUC with RBF kernel Recall: 0.5456
```

7.1.4. Random Forest




```
Random forest Accuracy: 0.5656
Random forest Precision: 0.5682196339434277
Random forest Recall: 0.5464
Random forest AUC: 0.5656
```

7.1.5. XG Boost

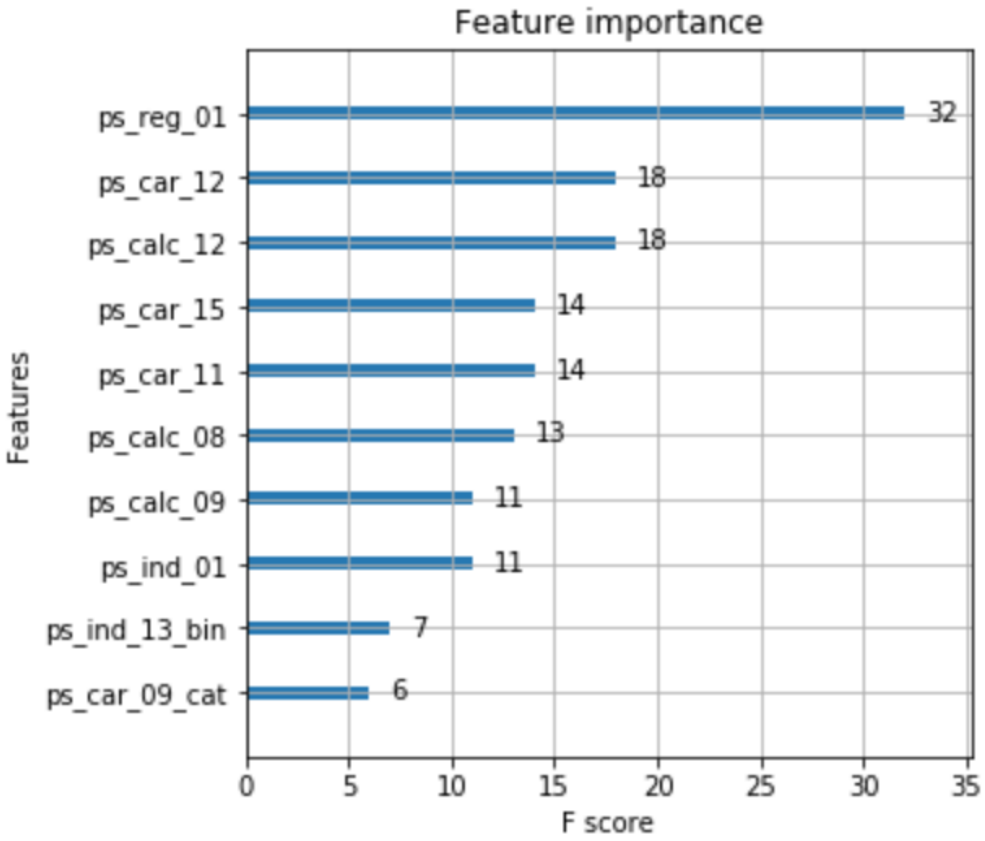


Vidya Sagar Polaki
CWID – 10430970
vpolaki1@stevens.edu

```
XG Boost Accuracy: 0.561
XG Boost Precision: 0.5568627450980392
XG Boost Recall: 0.5714285714285714
XG Boost AUC: 0.5610621982391366
```

	train-auc-mean	train-auc-std	test-auc-mean	test-auc-std
49	0.620203	0.001603	0.588849	0.038065
50	0.620420	0.001579	0.588880	0.038092
51	0.620699	0.001594	0.589006	0.038046
52	0.620946	0.001591	0.589237	0.038060
53	0.621164	0.001602	0.589374	0.038213

Best parameters using XG Boost



7.2. Performance Evaluation

We used Precision, Recall and Accuracy as our performance measuring parameters.

- Precision: percentage of true cases among the predicated true cases
- Recall: percentage of true cases that have been retrieved over the total number of true cases
- Accuracy: It is the number of correct predictions made divided by the total number of predictions made
- AUC: AUC represents the probability that a random positive sample is positioned to the right of a random negative sample. it measures the likelihood that given two random points — one from the positive and one from the negative class — the classifier will rank the point from the positive class higher than the one from the negative one.

8 Analysis of Experiments Results

8.1 What Worked & Didn't Work

- Missing value imputations worked well.
- Could identify correlation between variables clearly.
- Sampling issues might have affected the model. I'm still figuring out the issue.
- Different split values didn't work well for the data.
- Multiple algorithms were tested and as No Free Lunch theorem states, computational complexity and optimization, for certain types of mathematical problems, the computational cost of finding a solution, averaged over all problems in the class, is the same for any solution method.
- XG Boost is better than other classifiers in this instance with AUC score of 0.5893

8.2 Challenges Faced

- Dataset is highly imbalanced. With target variables 1 totaling only 3.64% of data. Balanced the dataset using under sampling and reduced the number of rows with target variable 0

8.3 Scope of Improvement

- There is a lot of scope for improvement as the designed models hardly reached 60% accuracy which is mostly because of the imbalances in the dataset

8.4 Business Insights & Application

Model upon further tuning can be used to tailor the insurance policies for specific drivers. If the driver was predicted to file a claim, then company can raise his insurance cost so that it will not incur any loss. If the driver was predicted not to file a claim, company can reduce his insurance cost so that user will be happy, and it can retain the user for longer period. It's a win-win situation for both the company and the driver.

9 Conclusion

Here due to imbalances in dataset, we select AUC value as the performance evaluation parameter.

accuracy depends on the threshold chosen, whereas the AUC considers all possible thresholds. Because of this it is often preferred as it provides a "broader" view of the performance of the classifier, but they still measure different things and as such using one or the other is problem-dependent.

There is a lot of scope for improvement and I would love to work on the model even though the semester ended to improve it and if possible, make a working product out of it to integrate with auto insurance companies.

10 Future Scope

Once a successful model is generated, it can be applied to various other insurances like housing, health, appliances etc. Scope of this idea is very large, and this project just handled one area of a vast domain.

11 References

1. https://www.kaggle.com/bertcarremans/data-preparation-exploration
2. https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis
3. https://www.marcoaltini.com/blog/dealing-with-imbalanced-data-undersampling-oversampling-and-proper-cross-validation
4. https://www.kaggle.com/residentmario/undersampling-and-oversampling-imbalanced-data
5. https://www.analyticsvidhya.com/blog/2016/01/12-pandas-techniques-python-data-manipulation/

6. https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html
7. https://chrisalbon.com/machine_learning/preprocessing_structured_data/impute_missing_values_with_means/
8. https://towardsdatascience.com/logistic-regression-using-python-sklearn-numpy-mnist-handwriting-recognition-matplotlib-a6b31e2b166a
9. https://www.datacamp.com/community/tutorials/xgboost-in-python
10. https://www.datacamp.com/community/tutorials/random-forests-classifier-python
11. https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python
12. https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python