Project 31: Predict if driver will file an insurance claim next year.

Phase 2: Exploratory Data Analysis:

Type of Machine Learning problem: We are trying to predict whether the driver will claim insurance next year or not. So this is binary classification problem which we are trying to solve.

Here we are referring the “Target” column present in train.csv data set which is having two values as 0 or 1.

0 -> Represents that Insurance Claim is not filed.

1 -> Represents that Insurance Claim is filed.

Doing through EDA is necessary to solve any kind of machine learning problem. If we don’t perform EDA then we might run in to many issues while applying machine learning models. No data is perfect there might be some features which is not required for building the model so that we can drop those features to improve the performance of the model. There might be some missing values in our data set we need to handle missing values as well we also need to see if there is any outlier in our data set.

Loading the data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
import warnings
train=pd.read_csv('train.csv',index_col='id')
test=pd.read_csv('test.csv')
```

We have used python libraries like Numpy,Pandas,Matplotlib,seaborn.

Train variable will load all the training data set.

Test variable will load all the test data set.

- Train data set consists of 59 features including ‘ID’ and ‘Target’. So there are 57 features which we can use for our prediction. All features present in our data set in anonymous due to some data privacy reason.
- ‘bin’ features refers to binary features.
- ‘cat’ feature refers to categorical features.
- ‘calc’ refers to extra calculated features.

We have divided the features in our dataset to different categories

1. Numerical features
2. Categorical features
3. Binary features

```
# We are now preparing the list of all different types of features in our data set
# 1. Numberical features
# 2. Categorical features
# 3.Binary features
# All features
all_features=train.columns.tolist()
all_features.remove('target')
# numeric features
num_features=[n for n in all_features if n[-3:] not in ['bin','cat']]
# Categorical features
cat_features=[c for c in all_features if c[-3:]=='cat']
# Binary features
bin_features=[b for b in all_features if b[-3:]=='bin']
```

Here in the above code we have removed the ‘target’ column from our train data set as it is dependent variable.

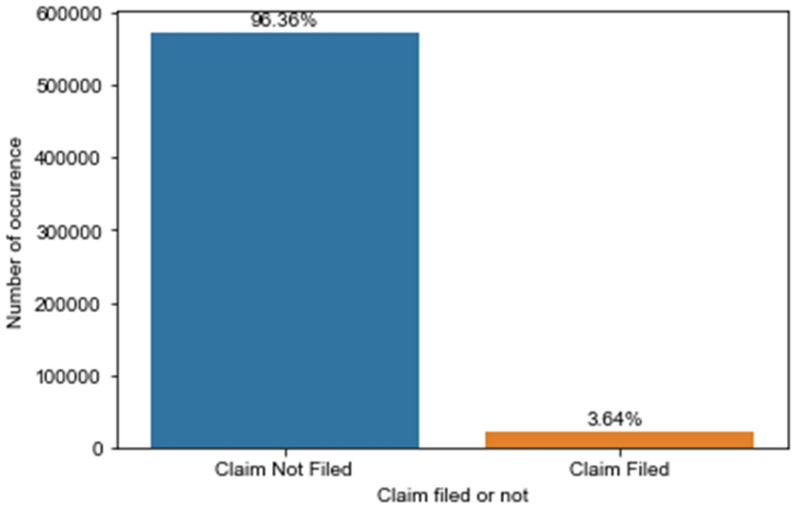
Num_features refers to numerical feature.

Cat_features refers to Categorical features

Bin_features refers to binary features.

```
]: # We need to map target columns as Insurance claimed or not Claimed
   # We are considering '0' as not claimed '1' as Claimed
   train['Insurance_Claimed'] = train['target'].map({0: 'Claim Not Filed', 1: 'Claim Filed'})
```

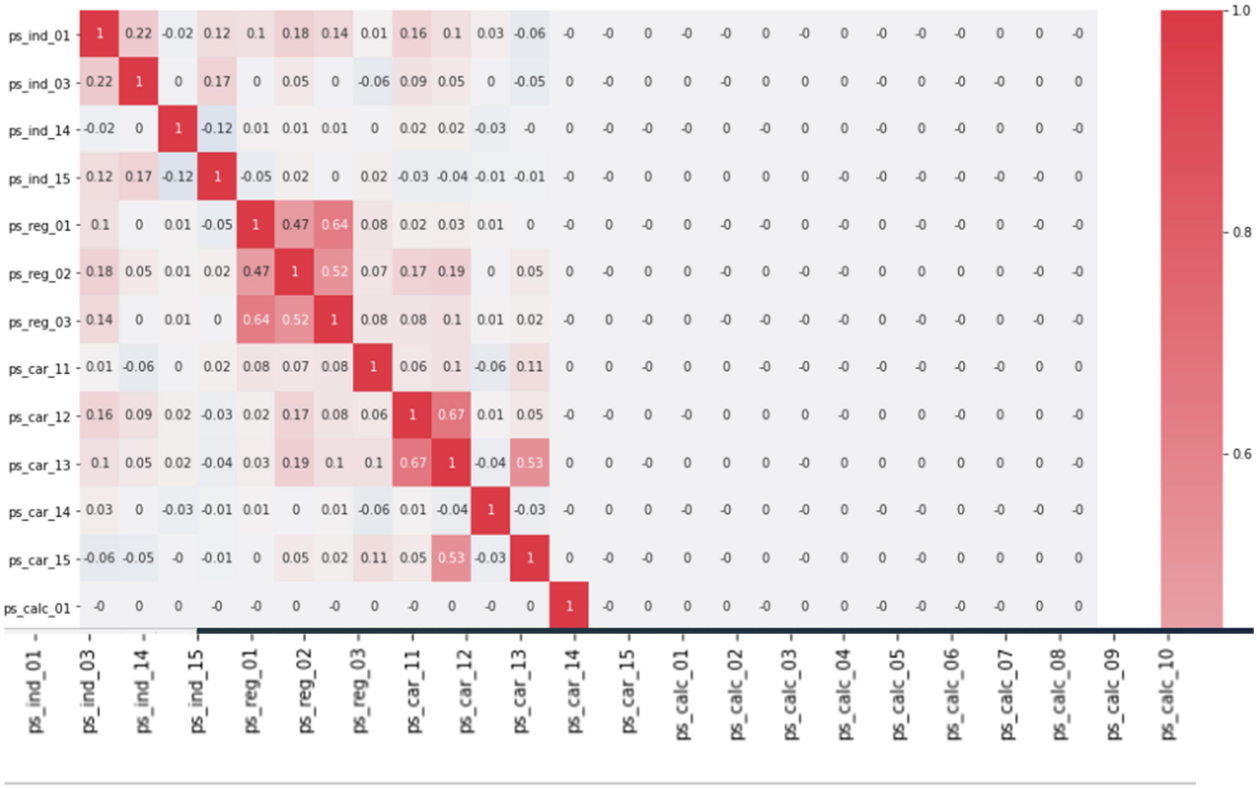
Here in above code snippet we have encoded Claim filed as 1 and Claim not filed as 0.



In the above it is clear that 96.36% driver have filed insurance claim and only 3.64% driver have filed insurance claim. So from this it is clear that data set is quite imbalanced.

Corelation between features

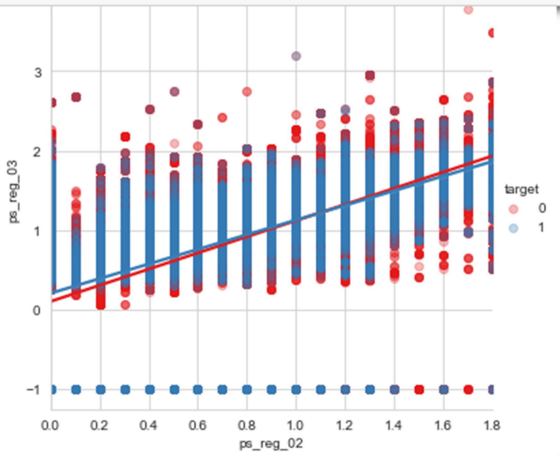
We have plotted heat map to find the correlation between the features.



1. Ps_car_13 and ps_car_12 we can see are highly correlated (0.67).
2. Ps_reg_02 and ps_reg_03 are highly correlated(0.7).
3. Ps_car_13 and pas_car_15 are also highly correlated(0.53).
4. Ps_car_12 and ps_car_14 have high correlation(0.58)

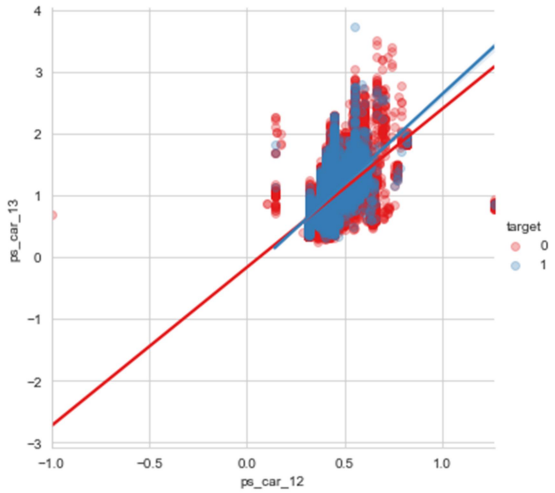
```
ps_reg_02 and ps_reg_03
```

```
3]: # We are plotting scatter plot to show the correlation between region 2 and region 3.
sns.lmplot(x='ps_reg_02', y='ps_reg_03', data=train, hue = 'target', palette = 'Set1', scatter_kws={'alpha':0.3})
plt.show()
```



```
ps_car_12 and ps_car_13 correlation.
```

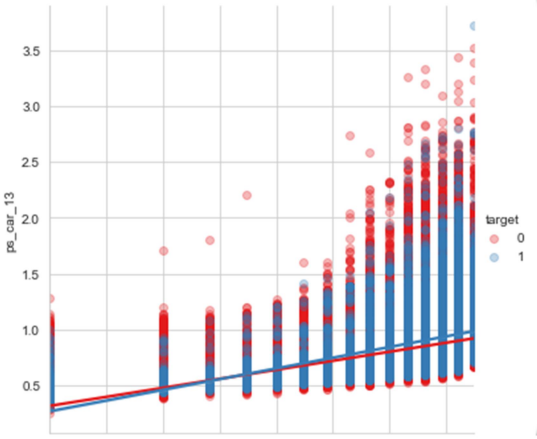
```
5]: # We are trying to plot scatter plot to show the correlation between car 12 and car 13.
sns.lmplot(x='ps_car_12', y='ps_car_13', data=train, hue = 'target', palette = 'Set1', scatter_kws={'alpha':0.3})
plt.show()
```



Scatter plot for ps_car_12 and ps_car_13.

```
ps_car_15 & ps_car_13
```

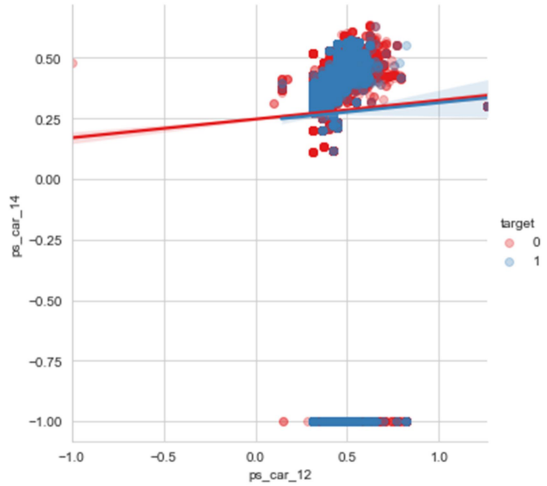
```
]: # We are trying to plot scatter plot to show the correlation between car 15 and car 13.
sns.lmplot(x='ps_car_15', y='ps_car_13', data=train, hue = 'target', palette = 'Set1', scatter_kws={'alpha':0.3})
plt.show()
```



Scatter plot between ps_car_15 and ps_car_13.

```
ps_car_12 & ps_car_14
```

```
# We are trying to plot scatter plot to show the correlation between car 12 and car 14.
sns.lmplot(x='ps_car_12', y='ps_car_14', data=train, hue = 'target', palette = 'Set1', scatter_kws={'alpha':0.3})
plt.show()
```



Scatter plot between ps_car_12 and ps_car_14.

```
# We checking the missed value by percentage
miss_values=train.eq(-1).sum()
column_name=train.columns
for i in range(len(miss_values)):
    if miss_values[i]!=0:
        print("The missing value % in column",column_name[i],"is", miss_values[i]*100/595212,'%')
```

```
The missing value % in column ps_ind_02_cat is 0.03628959093566662 %
The missing value % in column ps_ind_04_cat is 0.013944611331760785 %
The missing value % in column ps_ind_05_cat is 0.975954785857812 %
The missing value % in column ps_reg_03 is 18.106489788512327 %
The missing value % in column ps_car_01_cat is 0.01797678810239041 %
The missing value % in column ps_car_02_cat is 0.0008400368272145051 %
The missing value % in column ps_car_03_cat is 69.08983689844963 %
The missing value % in column ps_car_05_cat is 44.78253126617071 %
The missing value % in column ps_car_07_cat is 1.9302366215734899 %
The missing value % in column ps_car_09_cat is 0.09559619093701067 %
The missing value % in column ps_car_11 is 0.0008400368272145051 %
The missing value % in column ps_car_12 is 0.000168007365442901 %
The missing value % in column ps_car_14 is 7.160473915176441 %
```

Here are few features which have missing value.

- Since the data given to us was anonymized for different privacy reason so domain based feature engineering is not possible.
- We can drop those columns which is having missing values greater than 85% in our data set.

- For features which is having missing value between 5-25% we can impute them using mean or mode method or any other method.
- We can also do feature importance to select the important features which is useful for prediction and rest others we can drop from our train data set so that when we are building machine learning model it doesn't impact the performance of our model.

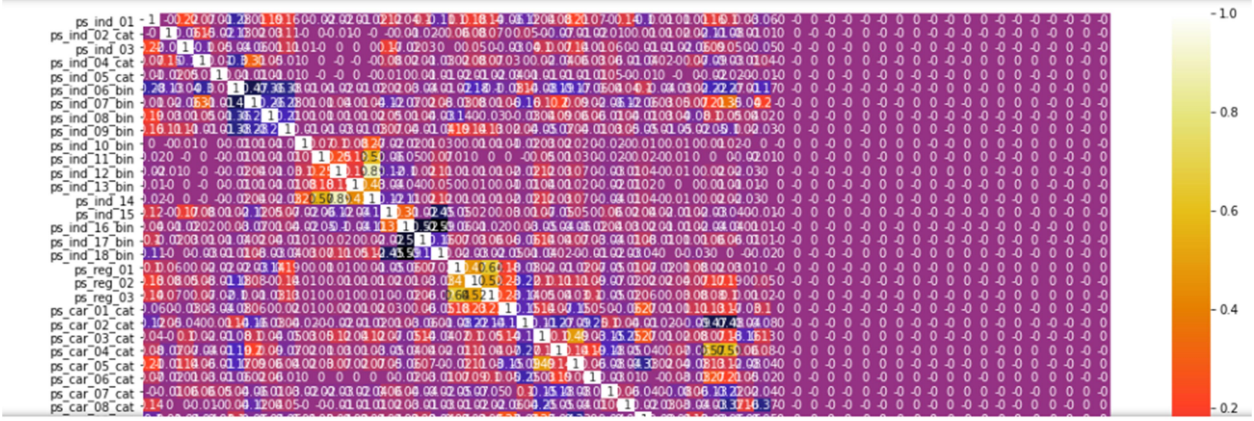
Feature Importance

```
# Here we are assigning target variable to y
y=train['target']
train=train.drop(['target'],axis=1) # dropping the target column which is a dependent variable from our train dataset.
X_train, X_test, y_train, y_test = train_test_split(train, y, test_size=0.1, random_state=42)
```

Pearson correlation can be used for find correlation between features and remove the features which have high correlation.

We have tried Pearson correlation to find out highly correlated feature so that we can drop those features from our data set.

```
]: # plotting the correlation between the features in train data set.
plt.figure(figsize=(18,12))
cor=X_train.corr().round(2)
sns.heatmap(cor,annot=True,cmap=plt.cm.CMRmap)
plt.show()
```



Here is the Pearson Correlation heatmap to find all the highly correlated features.

```
# Below code is used to calcualte Pearson-Corelation
def correlation(dataset,threshold):
    col_corr=set()
    corr_matrix=dataset.corr()
    for a in range(len(corr_matrix.columns)):
        for b in range(a):
            if abs(corr_matrix.iloc[a,b]) > threshold:
                col_name=corr_matrix.columns[a]
                col_corr.add(col_name)
        return col_corr
```

```
corr_features=correlation(X_train,0.9)
len(set(corr_features))
```

```
0
```

```
corr_features=correlation(X_test,0.9)
len(set(corr_features))
```

```
0
```

Here the above code is used to calculate the correlation between the features.

We can see that Pearson correlation doesn't show any features which is highly correlated. So we will be trying information gain technique to find the important features in our data set.

```
# We are trying to select important feature using information gain for classification in ML.
from sklearn.feature_selection import mutual_info_classif
# Determin the mutualinformation.
mutual_info=mutual_info_classif(X_train,y_train)
mutual_info
```

The above code we are using information gain to find the important feature for classification in ML.

All the features having high information gain can be considered as important features.

```
ps_car_10_cat     0.120373
ps_car_07_cat     0.107228
ps_car_02_cat     0.086017
ps_car_08_cat     0.085716
ps_car_09_cat     0.064991
ps_car_03_cat     0.063292
ps_ind_02_cat     0.056535
ps_ind_16_bin     0.055452
ps_car_11         0.051068
ps_calc_16_bin    0.048772
ps_calc_17_bin    0.038438
ps_car_05_cat     0.035659
ps_car_01_cat     0.028745
ps_calc_04        0.028121
ps_calc_05        0.026799
ps_calc_06        0.026069
ps_calc_08        0.025230
ps_calc_09        0.024938
ps_calc_12        0.023670
ps_calc_07        0.021774
ps_ind_04_cat     0.020964
ps_ind_06_bin     0.020010
ps_calc_13        0.017828
ps_car_06_cat     0.014815
ps_calc_19_bin    0.014602
ps_calc_11        0.013297
ps_ind_01         0.012672
ps_reg_01         0.011997
```

We have found around printed all features with information gain and now we are going to select top 10 features for our prediction.

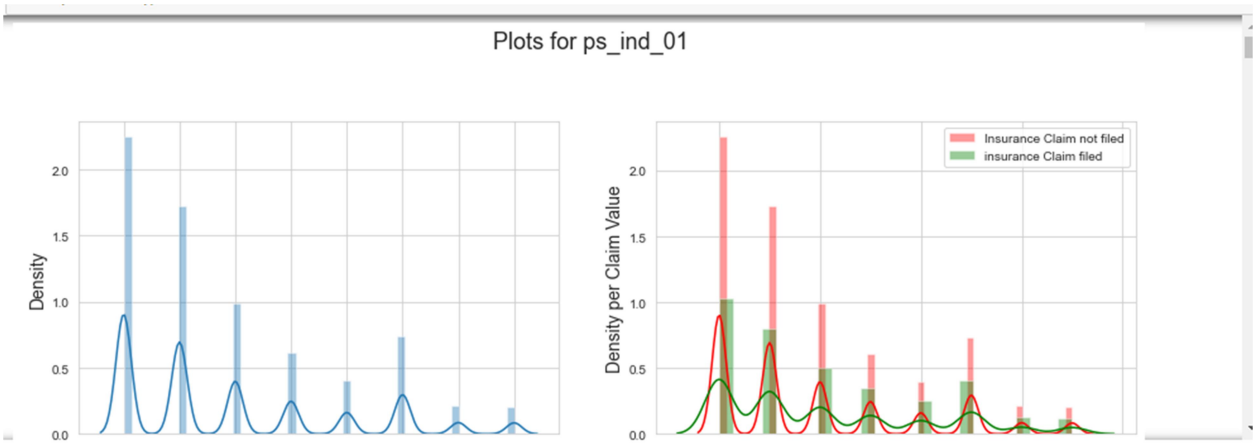
```
# we are selecting top 10 important features
select_ten_col=SelectKBest(mutual_info_classif,k=10)
select_ten_col.fit(X_train,y_train)
X_train.columns[select_ten_col.get_support()]
```

```
Index(['ps_ind_02_cat', 'ps_ind_16_bin', 'ps_car_02_cat', 'ps_car_03_cat',
       'ps_car_07_cat', 'ps_car_08_cat', 'ps_car_09_cat', 'ps_car_10_cat',
       'ps_car_11', 'ps_calc_16_bin'],
      dtype='object')
```

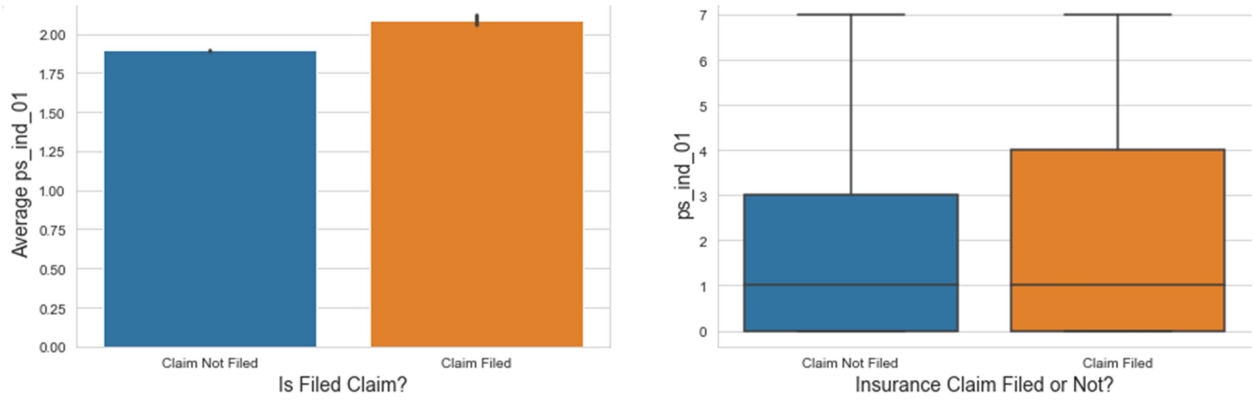
These are the top 10 features which we have selected as top 10 features for prediction.

Numerical Feature EDA

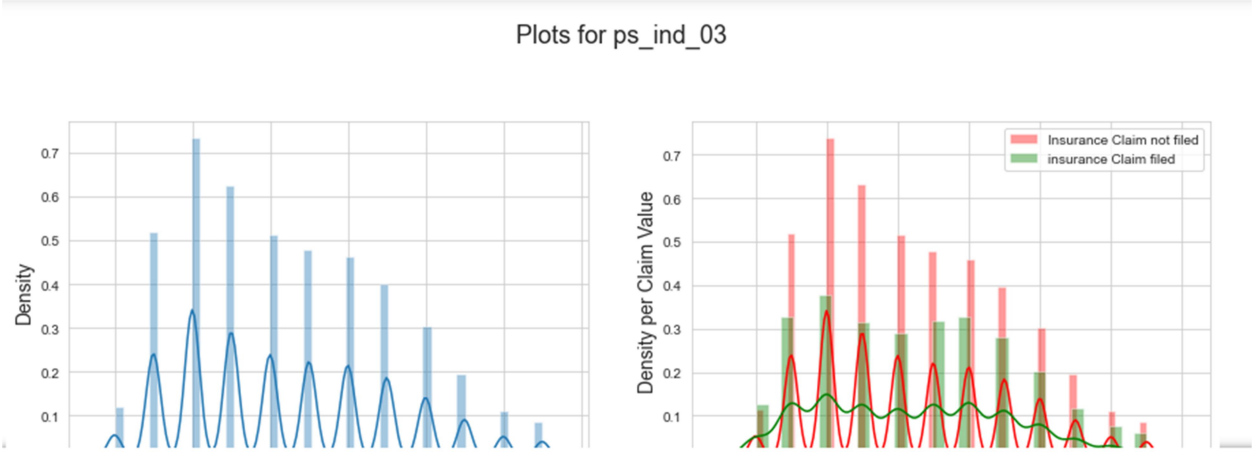
Now are going to detail data analysis on numerical features present in our data set.

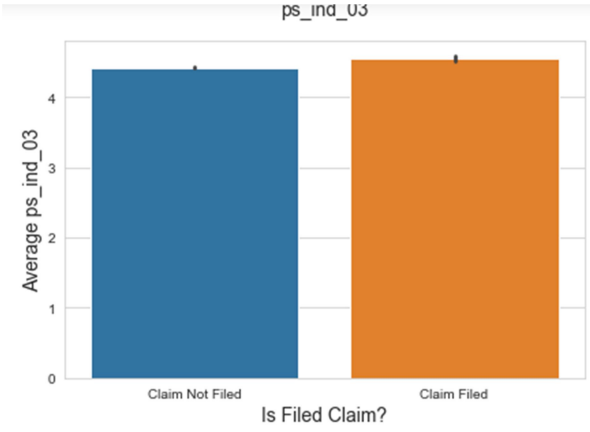


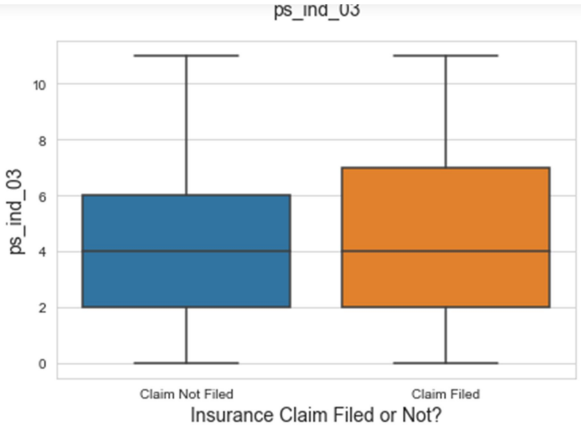
Here we have plotted the density and density per claim.

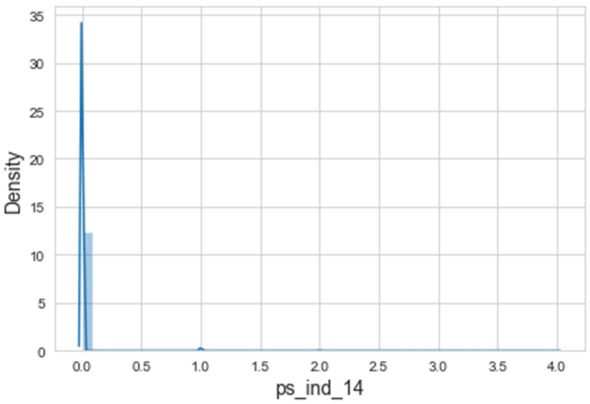


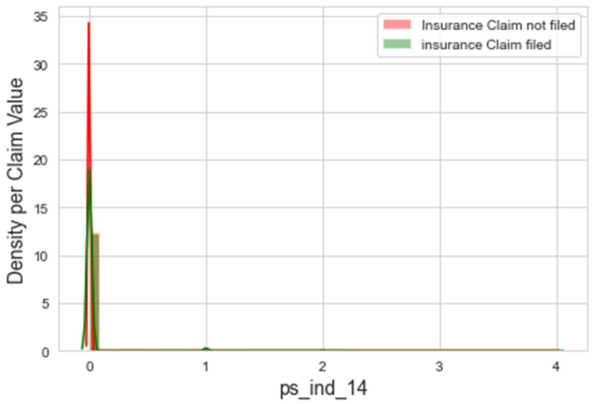
We have used box plot for Average value for all the numerical features.

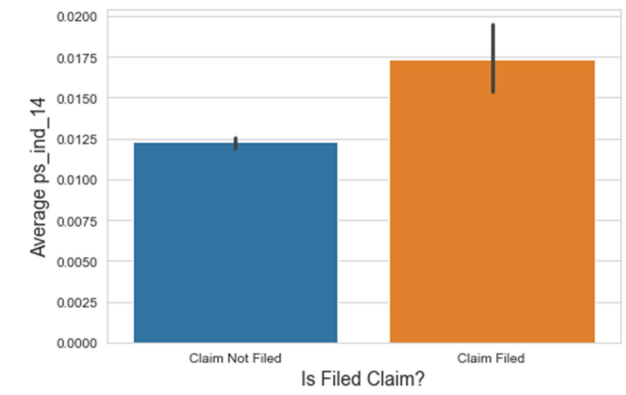


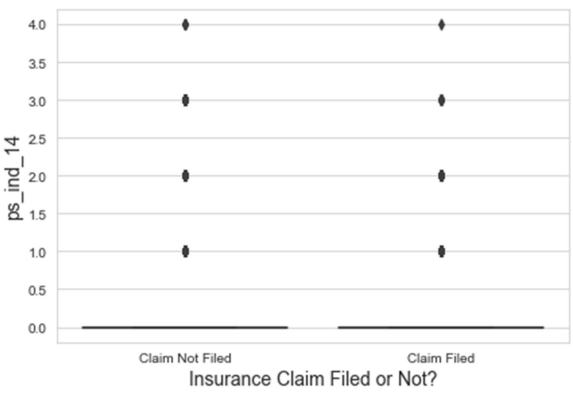

ps_ind_03
Average ps_ind_03
0
1
2
3
4
Claim Not Filed
Claim Filed
Is Filed Claim?

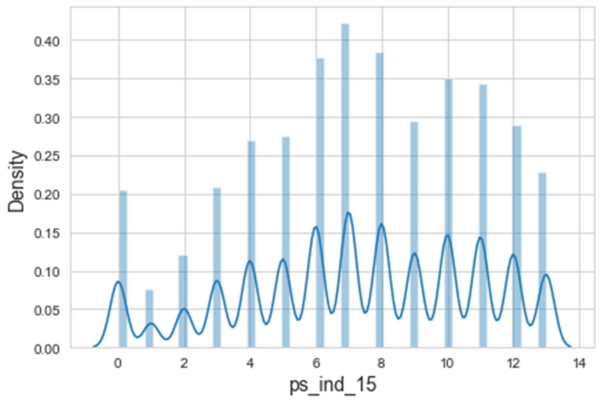

ps_ind_03
ps_ind_03
0
2
4
6
8
10
Claim Not Filed
Claim Filed
Insurance Claim Filed or Not?

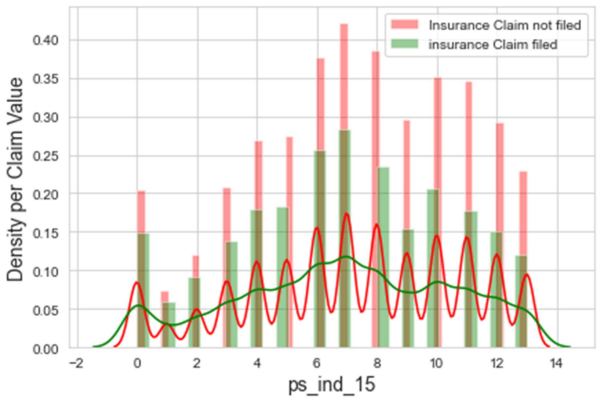

Density
0
5
10
15
20
25
30
35
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
ps_ind_14

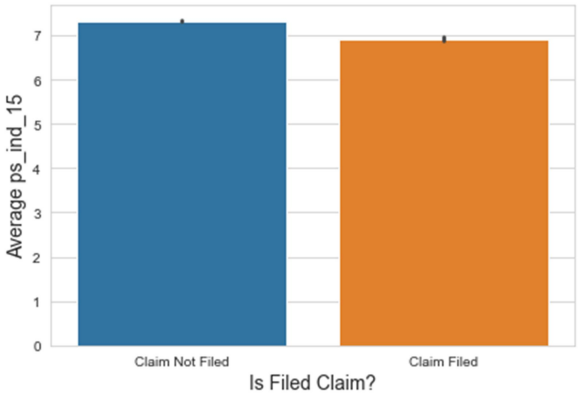

Insurance Claim not filed
insurance Claim filed
Density per Claim Value
0
5
10
15
20
25
30
35
0
1
2
3
4
ps_ind_14

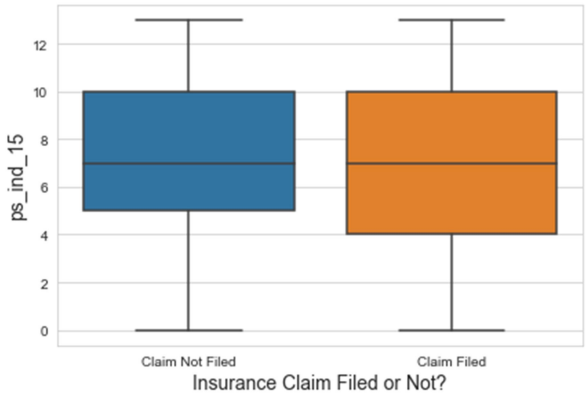

Average ps_ind_14
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0.0200
Claim Not Filed
Claim Filed
Is Filed Claim?

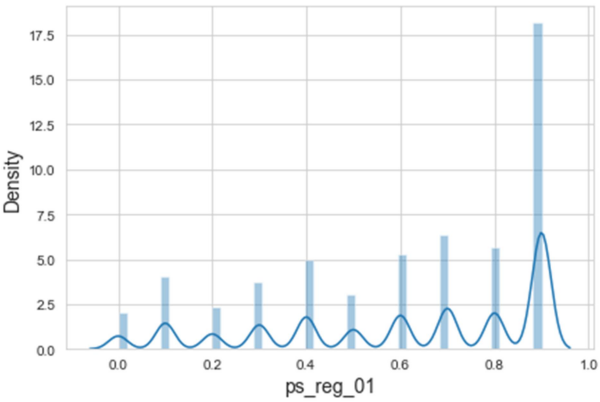

ps_ind_14
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Claim Not Filed
Claim Filed
Insurance Claim Filed or Not?

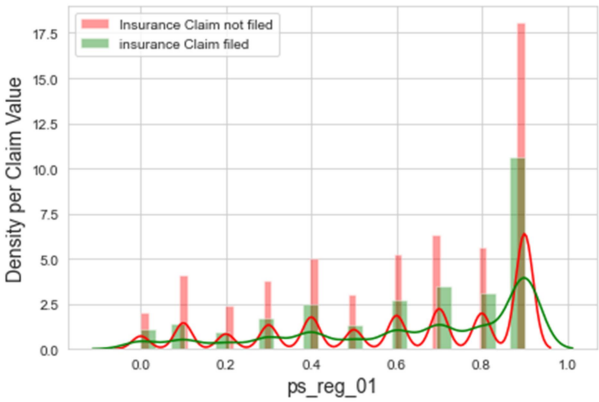

Density
ps_ind_15

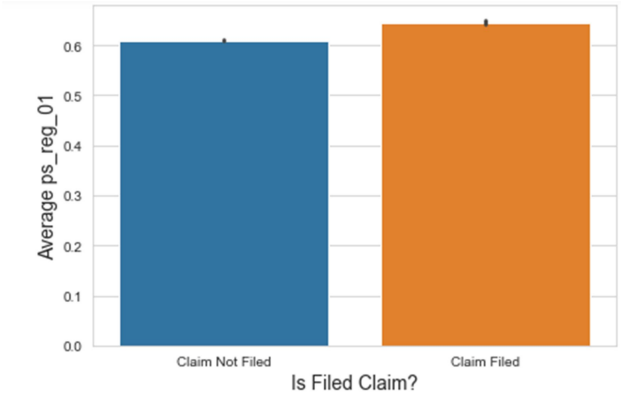

Insurance Claim not filed
insurance Claim filed
Density per Claim Value
ps_ind_15

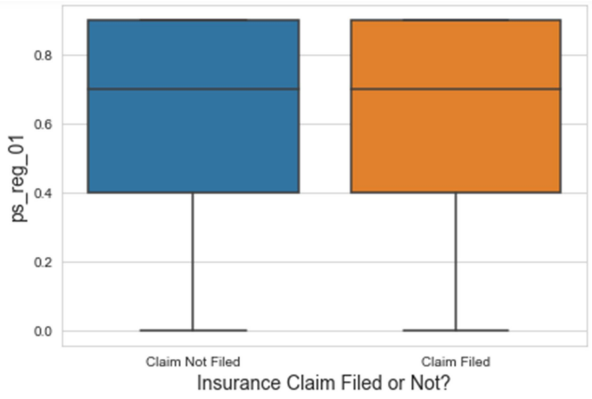

Average ps_ind_15
Claim Not Filed
Claim Filed
Is Filed Claim?

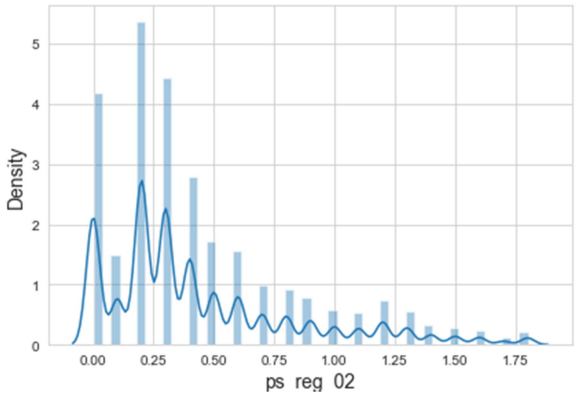

ps_ind_15
Claim Not Filed
Claim Filed
Insurance Claim Filed or Not?

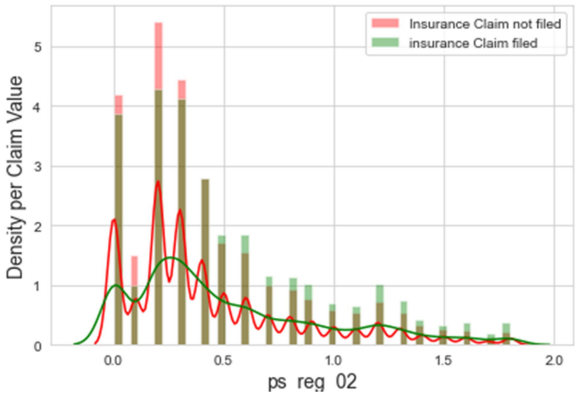

Density
ps_reg_01

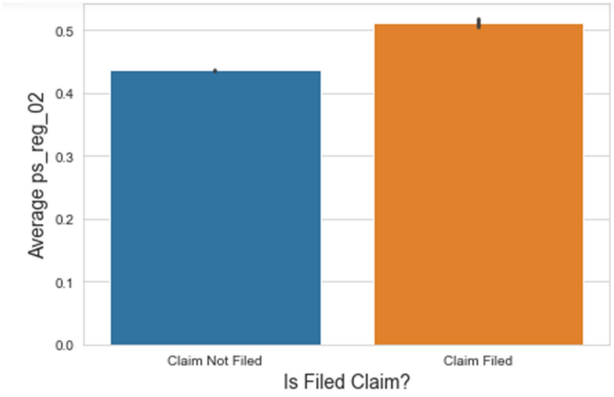

Insurance Claim not filed
insurance Claim filed
Density per Claim Value
ps_reg_01

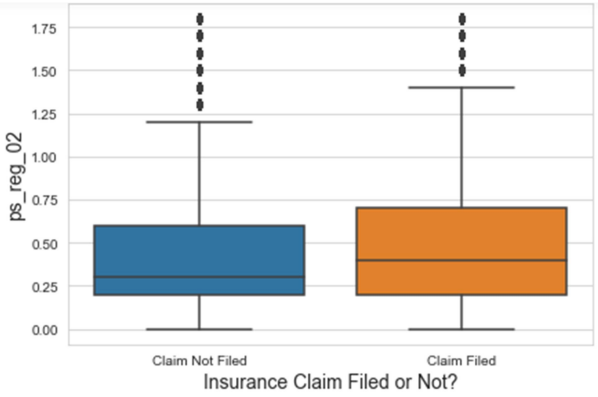

Average ps_reg_01
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Claim Not Filed
Claim Filed
Is Filed Claim?

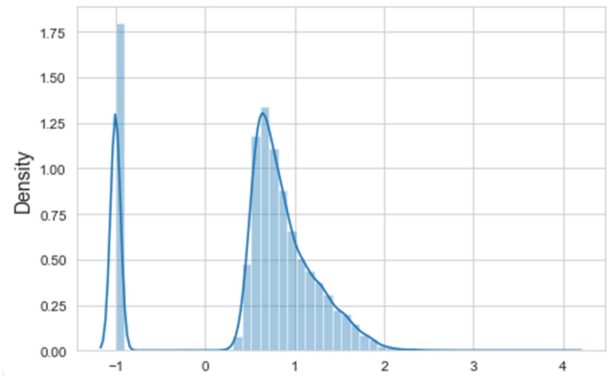

ps_reg_01
0.0
0.2
0.4
0.6
0.8
Claim Not Filed
Claim Filed
Insurance Claim Filed or Not?

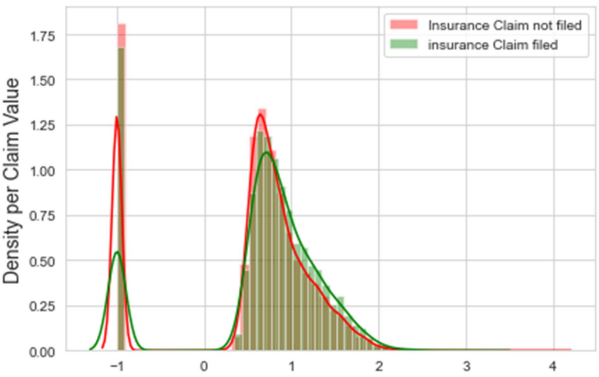

Density
0
1
2
3
4
5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
ps_reg_02

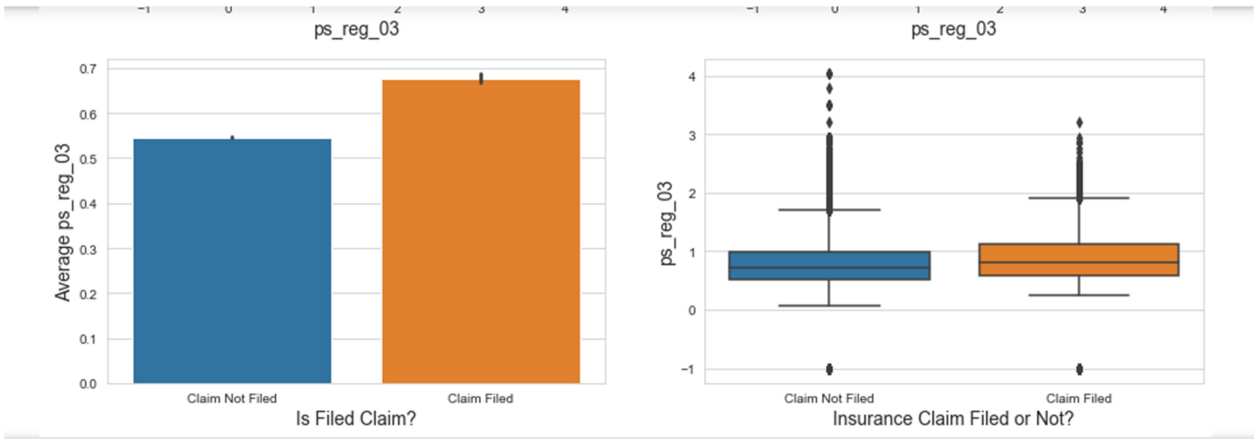

Insurance Claim not filed
insurance Claim filed
Density per Claim Value
0
1
2
3
4
5
0.0
0.5
1.0
1.5
2.0
ps_reg_02


Average ps_reg_02
0.0
0.1
0.2
0.3
0.4
0.5
Claim Not Filed
Claim Filed
Is Filed Claim?

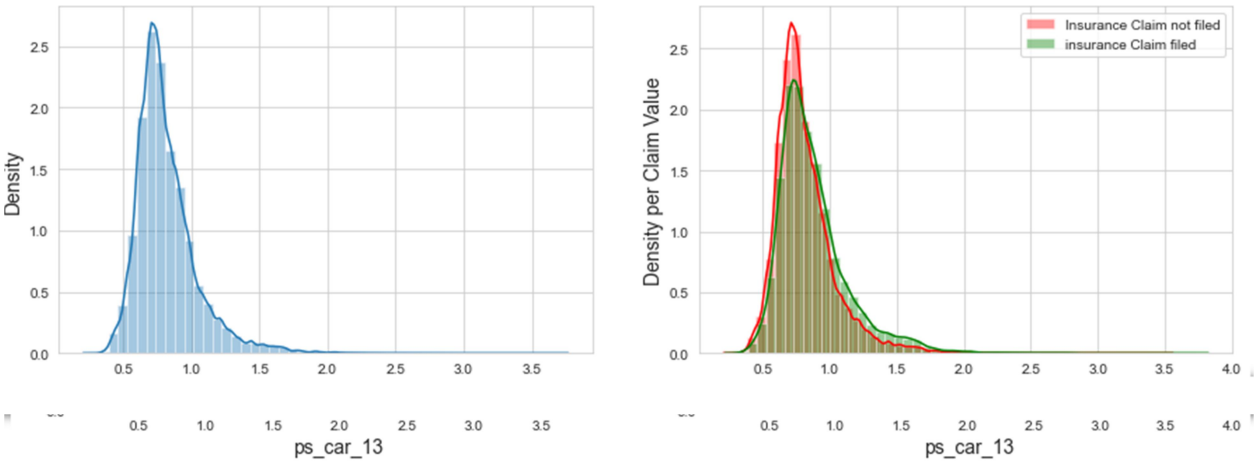

ps_reg_02
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Claim Not Filed
Claim Filed
Insurance Claim Filed or Not?

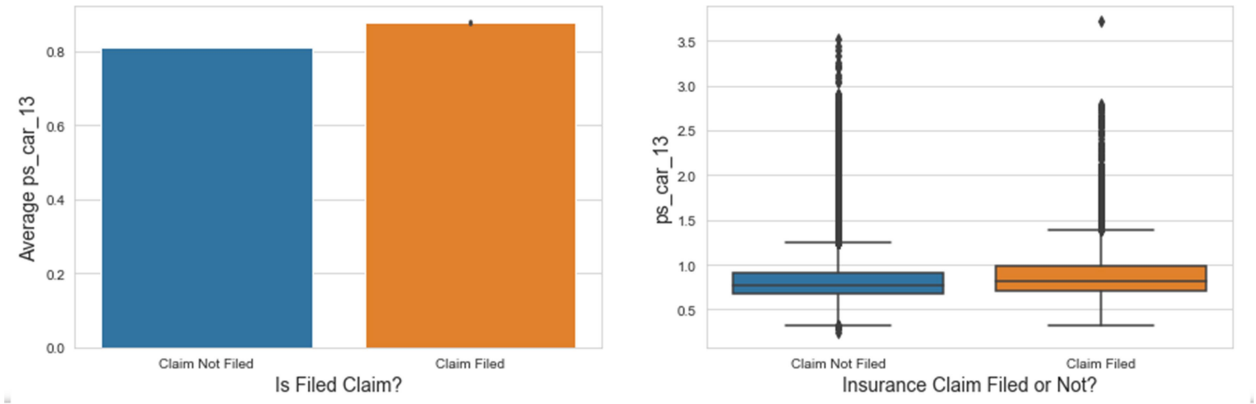

Density
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
−1
0
1
2
3
4

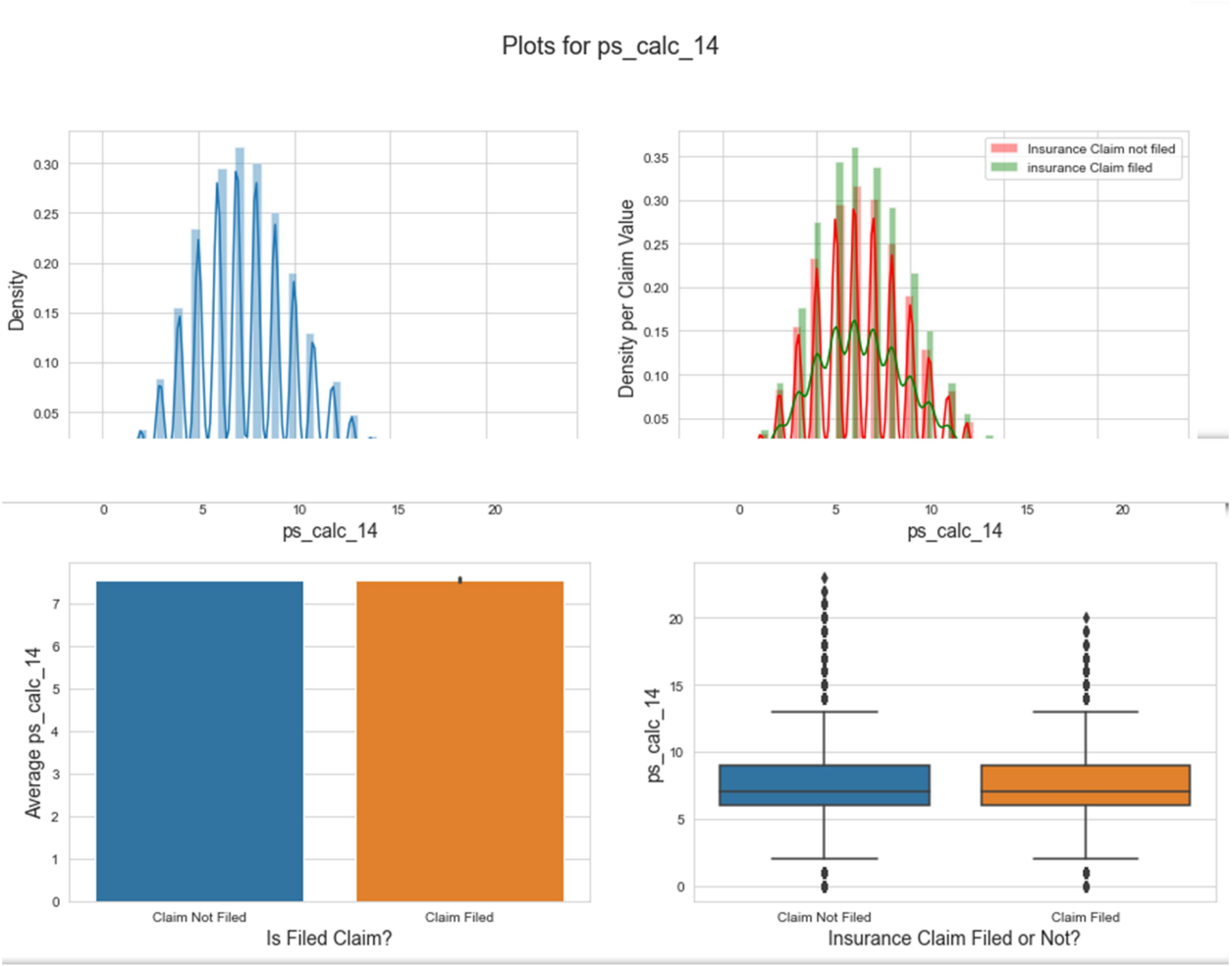

Insurance Claim not filed
insurance Claim filed
Density per Claim Value
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
−1
0
1
2
3
4



Plots for ps_car_13

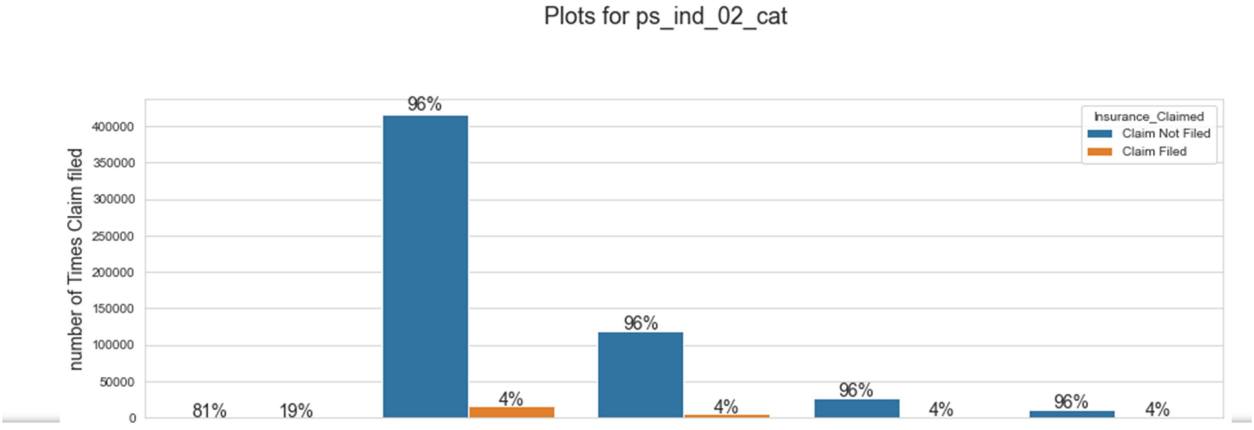




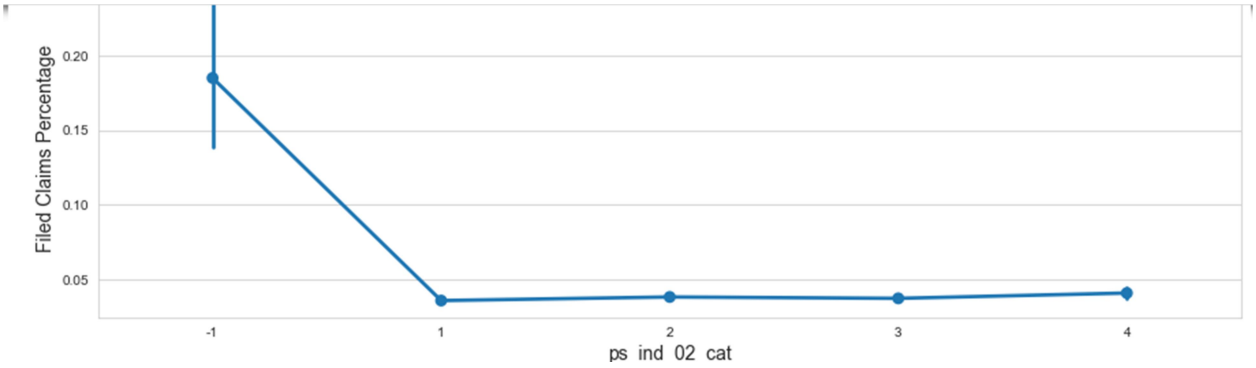


Categorical Feature EDA

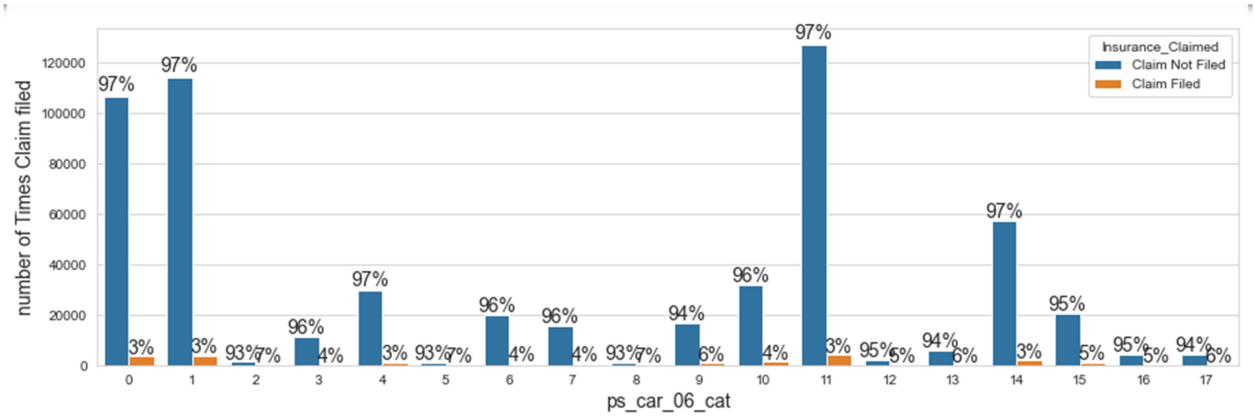
Now we are going to plot the graphs for all the categorical features present in our data set.



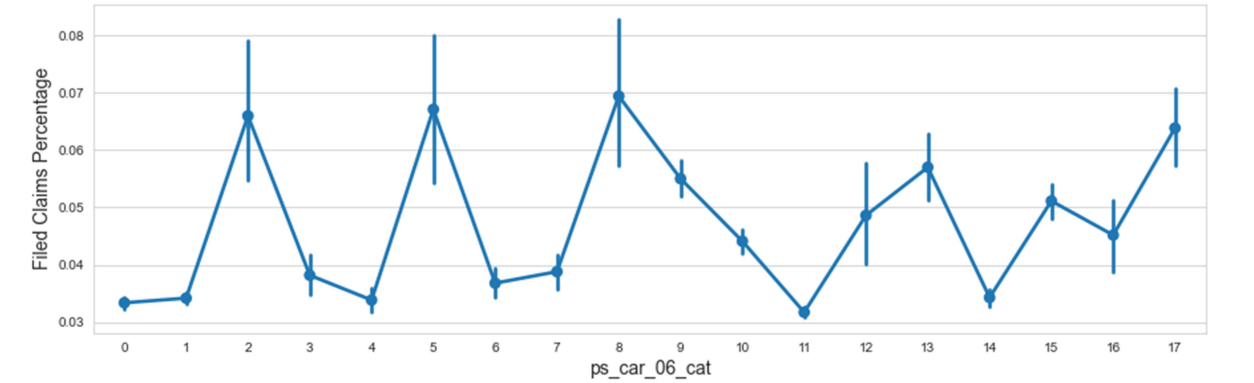
Here we can see that the individual driver who is categorized as ps_in_02 has not filed the insurance claim most of the time.



Filed Claims percentage is almost same for the same driver.

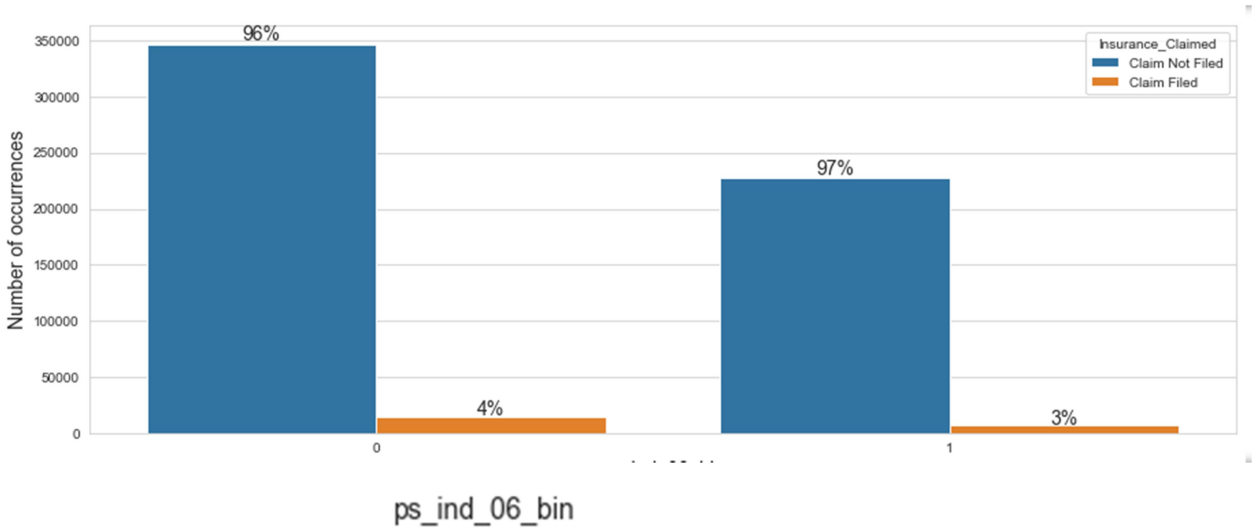


Here we can see number of time car_06 has filed the insurance for different years. Most of the time driver has not filed insurance claim as we can see in graph that 97% is claim not filed and 3%-5% claim is filed.



This is the Filed Claims percentage for ps_car_06 feature in our data set.

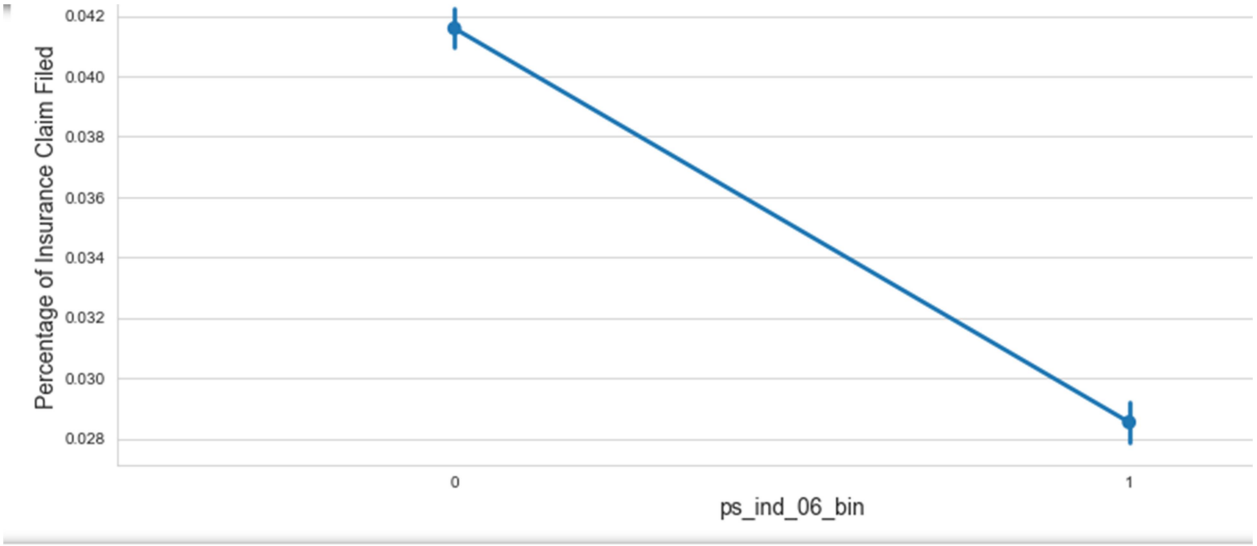
Binary Features EDA



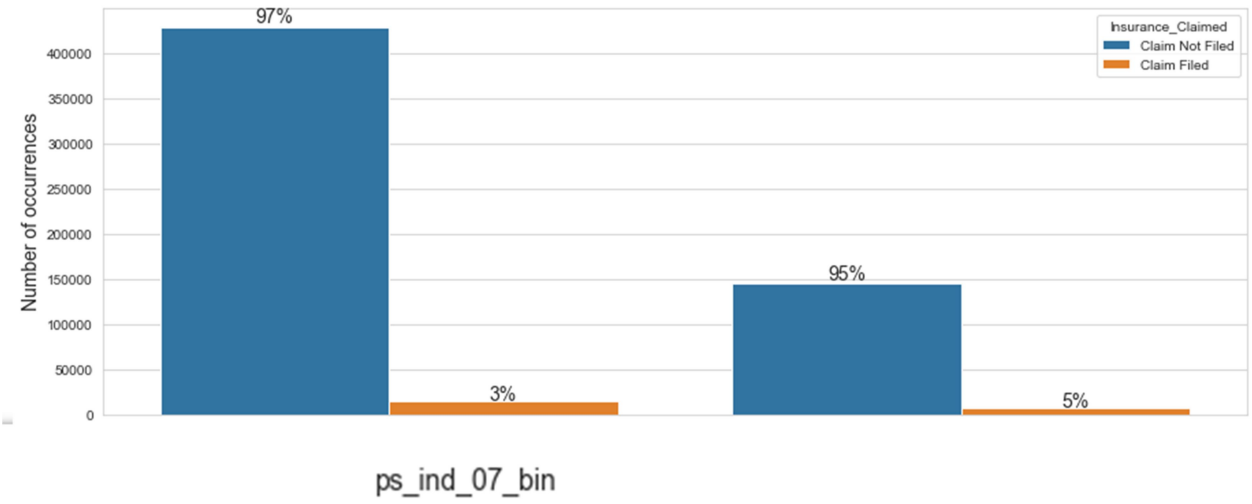
Here we have plotted bar graph for one of the binary features as ps_ind_06_bin.

We are showing the number of occurrence for insurance claim filed or not filed.

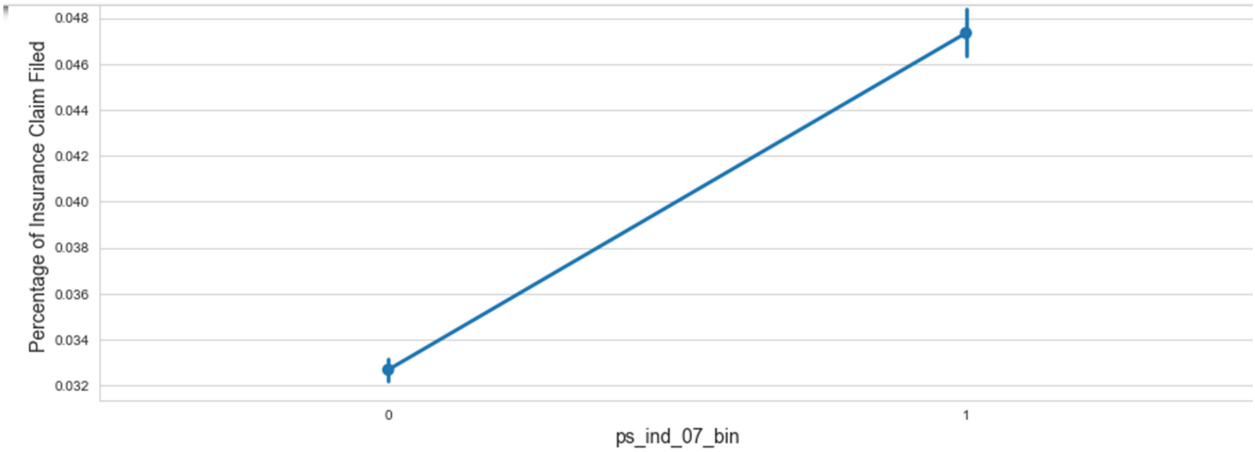
We can see almost 97% insurance claim is not filed and only 3%-4% insurance claim is filed.



In the above graph we can see that last year claim was around 4% and next year claiming filed percentage is dropped to 3%.



So here for another feature ps_ind_07 we can see that there is increase in the percentage of insurance claimed from last year. Even the percentage for insurance claim not filed has also dropped from last year which was around 97% last year and next year it is dropped to 95%.



Ps_ind_07 feature we can see that last year insurance claim filed was around 3% and now next year it is increased to almost 5%.

References

1. https://www.kaggle.com/code/emilyjiminroh/data-exploration-porto-seguro-s-safe-driver
2. https://hemanthsagar1995.medium.com/porto-seguros-safe-driver-prediction-a1b9401da661

3. https://www.kaggle.com/code/ryujungsoo/porto-sequro-s-safe-driver-data-analysis
4. https://medium.com/analytics-vidhya/porto-seguros-safe-driver-prediction-a-machine-learning-case-study-1f2dd3f79d9e
5. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.r_regression.html
6. https://www.geeksforgeeks.org/python-pearson-correlation-test-between-two-variables/
7. https://www.youtube.com/watch?v=81JSbXZ26Ls&list=PLZoTAELRMXVPgjwJ8VyRoqmfNs2CJwhVH&index=3