TEL AVIV UNIVERSITY
Department of Economics
Data Science Seminar
Lecturer: Dr. Dan Zeltzer

Comparison of machine learning models for the imbalanced classification problem of fraud detection in car insurance claims

Authors: Elad Golan & Dov Tuch

Email: eladg010@gmail.com , dovboristuch@mail.tau.ac.il

Date: 01/05/2022

Abstract

Automobile Insurance Fraud is one of the main challenges for insurance companies, causing yearly substantial financial losses worldwide and impairing the speed and quality of services to insured customers. In recent years, applications of machine learning models for fraud detection have been studied to deal with this problem. Often fraud detection problem is a private case of an imbalanced classification problem. A classification problem that concerns many other areas, such as credit card, health insurance, financial statements, disease detection, etc. Traditional classification algorithms can be limited in their performance on highly unbalanced datasets. In this study, we focus on comparing 3 machine learning models - Random Forest, Support Vector Machine, Naïve Bayes; and the combination of those models with imbalance classification techniques - Synthetic Minority Oversampling Technique, Near Miss Undersampling, cost sensitivity learning and feature selection decomposition. We evaluate models' performance by a few relevant metrics – Area Under ROC curve (AUC-ROC), F-scoring, G-mean and Area Under Precision/Recall Curve (AUC-PR). Of the 15 methods and models combinations considered in this paper, Random Forest with Cost Sensitivity technique achieved the best performance metrics. We have shown that the test results of the chosen model can be used as a preliminary filter of identifying a claim fraud for insurance companies and a high confidence that a classified non-fraudulent claim is indeed so.

Introduction

Insurance fraud is a deliberate deception perpetrated against or by an insurance company or agent for the purpose of financial gain. Common frauds include "padding" or inflating claims; misrepresenting facts on an insurance application; submitting claims for injuries or damage that never occurred; and staging accidents.

As economists our mission is to maximize overall social welfare. The lack of genuine information about the insurance claim causes an increase in policy

prices and extends the time of receiving compensation, which also affects honest policy holder.
In the case of public institutions, such as Social Security / Ministry of Defense, the unnecessary payments are taken from public funds.
In this study, we compare machine learning models and common practices for fraud detection classification problems on car insurance claims dataset. This is important because developing accurate discrimination systems will prevent fraud success and will identify fraud faster thus reducing the overall time to handle claims and save money. We have chosen to focus on car insurance claims, but our work can be applied for detection classification problems (with some adjustments) in other fields, such as health insurance, credit card payment etc.

Literature overview

1) *SVMs Modeling for Highly Imbalanced Classification [1]*

In this work, different approaches of SVM model for dealing with an imbalance sample problem have been applied, SVM–WEIGHT, SVM-SMOTH, SVM-RANDU and a new based SVM method was introduced, Granular Support Vector Machines- Repetitive Undersampling algorithm (GSVM-RU). SVM–WEIGHT implements cost-sensitive learning for SVM modeling, the basic idea is to assign a larger penalty value to FNs than FPs. SVM-SMOTH adopts the SMOTE algorithm to generate more pseudo positive samples and then builds an SVM on the oversampled dataset. SVM-RANDU randomly selects a few negative samples and then builds an SVM on the undersampled dataset. GSVM-RU splits the training set into multiple negative information granules, in each iteration SVM is modelled to extract Negative Local Support Vectors (NLSVs) then these NLSVs are removed from the original training set. An aggregation operation is then executed to selectively aggregate the samples in these negative information granules with all positive samples to complete the undersampling process. Finally, an SVM is modelled on the aggregated dataset for classification. GSVM-RU gains a smart choice of undersampling, instead of a random choice as is done in SVM-RANDU.
Seven highly imbalanced datasets are collected from related works. The authors compared the SVM models and the previous approaches of the related

works. The performance comparison was performed by 4 different metrics, G-mean, AUC-ROC, F-Measures, AUC-RR, better suited to binary imbalanced classification problems than the other traditional metrics. The result shows the GSVM-RU outperforms the most of previous best approach and it is also an efficient method. In addition, the results show that SVM-WEIGHT is also highly effective, but less efficient, therefore it can be the first SVM modelling method of choice if the dataset is not exceptionally large.
The above study contributed to our personal understanding, by exposing us to problems that can arise in imbalanced data and introducing us to relevant methods for solving. In addition, the article reinforced our understanding that it is important to look at a correct metric, depending on the specific classification / prediction problem.

2) *Feature selection for high-dimensional imbalanced data [2]*
In this paper a discussion on the problems of using feature selection with highly imbalanced data is conducted. By showing that the traditional model-independent feature selection methods have a biased influence toward the majority class. For example, the calculation of Fisher scores will be mainly influenced by the majority class and therefore will lower the metric score in the end. To reduce the bias, a model-independent decomposition-based feature selection method is proposed.
The proposed method is divided into three phases. First, a decomposition of the majority class into smaller sub-classes, using a clustering algorithm that can produce clusters with balanced size (Expectation-Maximization, k-means-clustering, etc...). This makes the data multi-class balanced. In The second phase, a traditional feature selection is used to select the best-m features from the multiclass data. In the third phase the sub-classes are relabeled to the original major class and the original labeled data with the selected features is returned.
Multiple data sets were used from different domains: CNS, LYMPH, OVARY, NIPS each data set has 7129, 7129 ,6000 and 13,649 features with class imbalance ratios of 2:1, 58:19, 25:8, 4:1, respectively. The performance metrics of each data set with decomposition feature selection and regular feature selection were compared using Naive Bayes, SVM and decision trees with

Bayesian learning (LIBSVM and C4.5 algorithms respectively). The results showed that the performance of decomposition-based FS became better than traditional FS when the number of m features selected is m >45. This study illuminated the importance of understanding how imbalanced data can influence traditional methods that are not necessarily the models themselves but also the processes we do beforehand.

3) *Mining corporate annual reports for intelligent detection of financial statement fraud–A comparative study of machine learning methods. [3]*
A comparison of different machine learning models for predicting fraudulent financial statements. The original data consisted of 311 fraudulent statements gathered from US SEC in AAER. The data was manually balanced by getting 311 non-fraudulent reports from firms with corresponding market capitalization and industry membership. 30 stratified samples (explain this) have been created from the original data. Divided to 75% training and 25% test. On each data partition a correlation-based selection filter combined with a forward-selection was used to find subsets of low inter-correlation and high correlation with the class. This filter was used for two reasons: highly inter-correlated data ($P < 0.05$) and model-independence. The average number of selected samples was 7.87 with S.D of 0.73
Afterwards, on each sample, a comparison of the performances of 14 different learning techniques was applied using Accuracy, TP rate, TN rate, MC (combination of FP and FN rates), F-measure and AUC. The comparison of the metrics of each method was implemented by a statistical paired t-test. The results show that ensemble methods had the best performance in terms of correctly classifying fraudulent claims, the MC metric of fraudulent firms was significantly higher than non-fraudulent.
Bayesian belief networks and Decision Table/Naïve Bayes (DTNB) hybrid classifier provided the highest accuracy on non-fraudulent firms in terms of TN rate. This suggests that correct prediction of non-fraudulent firms is less complex, where the detection of fraudulent firms requires more complex (and less interpretable) machine learning methods.
In the discussion they proposed the possibility of designing loss functions where misclassified fraudulent firms get a much bigger penalty 1:2, 1:10, 1:20,

the rates: used in the study, recommended for regulators and investors, respectively [15]. We learned that NB and RF are good models for fraud detection, and we used those models in our comparison as well as cost-sensitive learning.

Data description and preprocessing

The Vehicle Insurance Claim Fraud Detection data was taken from Kaggle[1] contains 15,000 vehicle claims reports between 1994 to 1996. Each report contains 33 variables, 1 continues, 32 categorical (ordinal and nominal), see Table 1[2]. Only 6% of observations have been labeled as fraud, as can be seen in Figure 1, hence it is an imbalanced classification problem. As seen in Table 1 and 2, there are features with many categories and there is a noticeably substantial difference between the number of observations in each category under both fraud and non-fraud, thus there are groups with very few observations.

Table 1:
Variables description

Variable name	Description	Values
FraudFound_P	Indicates whether the claim was fraud (1) or not (0)	1,0
Sex	------	male, female
Age	Age of driver who made the accident	numeric
Fault	Categorization of who deemed fault	Policy Holder, Third Party
Marital Status	Marital status of claimant	Single, Married ,Widow Divorced
Driver rating	Not Specified which which is better	Ordinal from 1-4
VehiclePrice	ranges of vehicle prices	less than 20000, 20000 to 29000, 30000 to 39000, 40000 to 59000, 60000 to 69000, more than 69000
BasePolicy	type of insurance coverage	Liability, Collision, All Perils
Days_Policy_Claim	the number of days between when the policy was purchased and the claim filled	None, 8 to 15, 15 to 30, More than 30
PastNumberOfClaims	previous number of claims filed by policy holder	None, '1', 2 to 4, more than 4

[1] https://www.kaggle.com/datasets/shivamb/vehicle-claim-fraud-detection
[2] Main variables only.

PoliceReportFiled	indicates whether a police report was filed for the accident	yes, no
WitnessPresent	indicated whether a witness was present.	yes, no
VehicleCategory	Categorization of vehicle type	sport, sedan, utility

Figure 1:

Bar plot of Fraud and Not-Fraud frequency

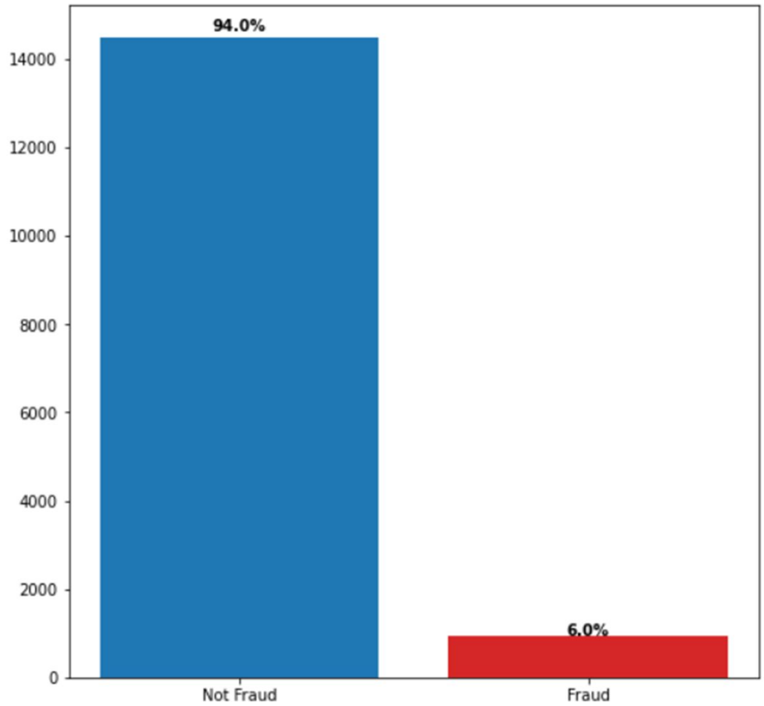


***Table 2**:*
Descriptive statistics

Variable name	category	Not fraud	Fraud
Sex	female	2315 (16%)	105 (11%)
	Male	12182 (84%)	818 (89%)
AccidentArea	Rural	1465 (10%)	133 (14%)
	Urban	13032 (90%)	790 (86%)
VehicleCategory	Sedan	8876 (61%)	795 (86%)
	Sport	5724 (36%)	84 (9%)
	Utillty	347 (3%)	44 (5%)
BasePolicy	All Perlis	3997 (28%)	452 (49%)
	Collision	5527 (38%)	435 (47%)
	Liability	4793 (34%)	36 (4%)
Fault	Policy Holder	10344 (71%)	886 (96%)
	Third Party	4153 (29%)	37 (4%)
VehiclePrice	Less than 20000	7658 (53%)	421 (46%)
	20000 to 29000	3358 (23%)	175 (19%)
	30000 to 39000	430 (3%)	31 (3%)
	40000 to 59000	83 (0.5%)	4 (0.5%)
	60000 to 69000	993 (7%)	103 (11%)
	More than 69000	1975 (13.5%)	189 (20.5%)
PastNumberOfClaimes	none	4013 (28%)	339 (37%)
	1	3351 (23%)	222 (24%)
	2 to 4	5191 (36%)	294 (32%)

	More than 4	1942 (13%)	68 (7%)
PoliceReportField	No	14085 (97%)	907 (98%)
	Yes	412 (3%)	16 (2%)
AgeOfVehicle	2 years	70 (0.5%)	3 (0.5%)
	3 years	139 (1%)	13 (1.5%)
	4 years	208 (1.5%)	21 (2%)
	5 years	1262 (9%)	95 (10%)
	6 years	3220 (22%)	228 (25%)
	7 years	5482 (38%)	325 (35%)
	More than 7	3775 (26%)	206 (22%)
	new	341 (2%)	32 (4%)

missing values: 320 of the observations were recorded with Age 0 and one observation was recorded with a 0 value for DayOfWeekClaimed and MonthClaimed. We assumed 0 indicates missing values, hence, to avoid dropping these observations, we replaced these values with the mean. We dropped PolicyType, PolicyNumber, AgeOfPolicyHolder and Make (car manufacturer) variable, because PolicyNumber is an ID of each Policy and PolicyType appears to be a concatenation of VehicleCategory and BasePolicy. AgeOfPolicyHolder and Age has correlation of 96%. Binary features replaced with zero one coding, ordinal feautres replaced with ordinal numbers or average of each category, using OneHot encoder for the rest categorical features. Afterwards, we split the data into train, validation, and test set in ratio of 70:20:10. Notice OneHot encoder application and the number of observations is relatively low in certain groups limitation may lead to sparse features matrix and cause issues in machine learning models like overfitting, inaccurate feature importance and high variance [4].

Imbalanced classification

The case when a data set is dominated by a major class or classes which have significantly more observations than the other rare/minor classes in the is known as class imbalance [1]. Imbalanced classifications pose a challenge for predictive modeling as most of the machine learning algorithms used for classification were designed around the assumption of an equal number of examples for each class [2]. It is possible that minority examples may be treated as noise by the learning model. This results in models that have poor predictive performance, specifically for the minority class. This is a problem because typically, the minority class is more important, therefore the problem is more sensitive to classification errors for the minority than the majority.

This problem becomes even more severe when the dimensionality of the data is high. Also known as the curse of dimensionality [3] which is out of the scope of this paper.

Table 3:
Confusion Matrix

		Real labeled	
		Positive (Fraud)	Negative (Not-Fraud)
predicted	Positive (Fraud)	TP	FP
	Negative (Non-Fraud)	FN	TN

To properly classify the minority class (fraud in our case) we must choose the right metric to optimize fraud detection. The common metrics are based on confusion matrix as shown at Table 3. When using balanced data, a common metric that is chosen is the accuracy measure. With highly skewed data distribution, the overall accuracy metric (1) is not sufficient anymore**.** Accuracy is a metric that is defined as the percentage of correctly classified samples across all classes [4]. It is useful when all classes are of equal importance, but in our case the important class is the minority (fraudulent claim). It is easy to get a high accuracy score by simply classifying all observations as the majority class [4].

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \qquad (1)$$

More suitable metrics are the Roc-Auc ,Area Under sensitivity/true positive rate (2) and specificity/true negative rate (3) Curve, F1-score (7), G-mean (4) and AUC-PR [5]. The tradeoff between sensitivity and specificity allows us to optimize our imbalanced classification problem relative to both classes, independent of class distributions. In fact, instead of maximizing the overall accuracy of the model we try to maximize true positive rate (true fraud classification) while considering the 1-specficty (false fraud classification) rate that we do not want to be too high. The Geometric Mean (G-Mean) is a metric that measures the balance between classification performances on both the majority and minority classes [6]. ROC-AUC can also indicate balanced classification ability between sensitivity and specificity, but unlike G-Mean by changing the threshold ROC-AUC can be used to choose between high sensitivity or low 1-specificity according to our specific problem [7].

$$sensitivity = TP/(TP + FN) \qquad (2)$$

$$specificity = TN/(TN + FP) \qquad (3)$$

$$G - Mean = \sqrt{sensitivity * specificity} \qquad (4)$$

In our case, the goal is to identify frauds, but we do not want to classify observations as fraud at all costs, because this will hurt honest customers. For this reason, we consider F1-score as it combines precision and recall into one metric by calculating the harmonic mean between those two [8]. It is actually a special case of the more general function F-beta[3], where by changing beta, you can give more weight to recall or precision. Another reason for using F1-score in imbalance classification problem is that we get an indication that the accuracy is irrelevant, when F1 is equal to zero. Similarly to AUC-ROC, AUC-PR allows a trade-off between recall and precision using a threshold.

$$precision = TP/(TP + FP) \qquad (5)$$

$$recall = TP/(TP + FN) \qquad (6)$$

$$F - Measure = \frac{2 * precision * recall}{precision + recall} \qquad (7)$$

After choosing the right model metrics for performance evaluation, some methods like resampling, cost-sensitivity learning and feature selection for imbalanced data (mentioned in the literature review) can help improve a model's performance.

For resampling methods, we use SMOTE: Synthetic Minority Oversampling Technique and NearMiss undersampling. SMOTE produces a new observation of the minority class by randomly picking a point from the minority class and computing the k-nearest neighbors for this point. Synthetic points are added between the chosen point and its neighbors. SMOTE's disadvantage is that it increases the likelihood of overfitting since it replicates the minority class [9]. NearMiss undersampling removes observations from the majority, when two points that belong to different classes are close to each other in the

[3] Multiply F1 by $\frac{1+\beta^2}{\beta^2}$

distribution. NearMiss can help improve the runtime of the model and solve memory problems by reducing the number of training data samples when the training data set is big, however it can lead to discarding useful information and getting a biased sample [10].
Cost-sensitive learning gives different misclassifications costs to each class. misclassification costs may be described by cost matrix *C,* with C(i, j) [11] being the cost
of predicting that an example belongs to class *i* when in fact it belongs to class *j*. In our case the cost of misclassifying fraud is greater than the cost of misclassifying non-fraud claim. It is often recommended to use the inverse rate (IR) in the data to weigh the cost ratio [12]. In our sample our IR is (1:16) cost to non-fraud and fraud misclassification, respectively. ratios of 1:10 or 1:20 were recommended also [13]. A disadvantage of cost-sensitive learning is that the true misclassification costs are often not known, and we must use heuristics. Another disadvantage is that it increases the time complexity of the model [16].
For imbalanced feature selection we will use the decomposition-based feature selection previously discussed [maamar number], in the second phase we will use kmean clusters and mutual information with Kbest.

As a result of what we explained above, using standard machine learning models will lead to poor results in imbalance classification problems, therefore we will combine the metrics and methods we presented along with the models we chose to use.
We apply 3 machine learning models - Random Forest, Support Vector Machine, Naïve Bayes using sklearn python package. The models' and methods' hyper-parameters were chosen by heuristics.

Experimental Results

Table 4 summarizes the results for all models and methods. The best results between the models with applied methods are marked with bold font. The first row shows the results of a baseline RF model with no resampling or cost-sensitivity methods being used. The accuracy of the baseline models is the highest with 94.1% while the Roc-Auc, G-mean, F1, F2 Scores are the lowest of all other methods. The results show that RF-SMOTH had the highest F1-score, SVM-SMOTH with decomposition feature selection had the highest accuracy. RF-CS outperformed the remaining models in ROC-AUC, G-mean F2-score and Auc-PR. Comparing the classification metrics between “conventional” and decomposition-based feature selection, conventional NB-SMOTH, NB-Nearmiss SVM-CS, surpassed D-based by across almost all metrics. D-SVM-SMOTH had the highest accuracy of all models but underperformed SVM-SMOTH in all other metrics. D-SVM-NearMiss had higher accuracy than SVM-NearMiss but underperformed in all other metrics.

Table 4:

Models' performance

Model	Feature Selection	Accuracy	Roc-Auc	G-mean	F1-score	F2-score	Auc-PR
RF baseline	-	94.49 %	50%	0	0	0	52.76%
RF-SMOTH	-	75.11%	67.44%	66.89%	**20.66%**	33.83%	36.81%
RF–NearMiss	-	52.22%	66.13%	64.25%	15.87%	30.73%	45.78%
RF–CS	-	59.64%	**75.6%**	**73.44%**	20.34%	**38.35%**	**52.65%**
NB-SMOTH	Kbest-MI	62.23%	67.0%	66.79%	17.43%	32.01%	41.89%
NB-SMOTH	D-Kbest-MI	40.91%	64.02%	58.52%	14.37%	28.99%	49.18%
NB–NearMiss	Kbest-MI	56.73%	65.19%	59.26%	15.98%	30.25%	42.52%
NB-NearMiss	D-Kbest-MI	54.39%	60.08%	59.74%	13.84%	26.36%	38.01%
NB-CS	Kbest-MI	56.14%	72.36 %	70.03%	18.54%	35.47%	50.72%
NB-CS	D-Kbest-MI	39.48%	63.82%	57.66%	14.24%	28.84%	49.69%
SVM-SMOTH	Kbest-MI	72.93%	67.4%	67.11%	19.93%	33.48%	37.61 %
SVM-SMOTH	D-Kbest-MI	**78.35%**	55.31%	48.87%	13.02%	19.56%	20.83%
SVM–NearMiss	Kbest-MI	45.96%	60.33%	58.13%	13.48%	26.66%	42.58 %
SVM-NearMiss	D-Kbest-MI	53.74%	53.92%	53.92%	11.42%	21.69%	31.52 %
SVM-CS	Kbest-MI	60.39%	74.61%	72.88%	20.13%	37.75%	51.22%
SVM-CS	D-Kbest-MI	41.75%	63.36%	58.52%	14.21%	28.59%	48.03%

Figure 3 compares the F1-score, precision and recall of each model-method on the validation sample. SVM-SMOTH and RF-SMOTH have the highest precision (12.53%, 11.91%). RF-CS, SVM-CS, NB-CS with the highest recall (93.5%,90.5%,90.5%).

Figure 3:

Bar plot comparison of models' F1-scoring, precision, recall

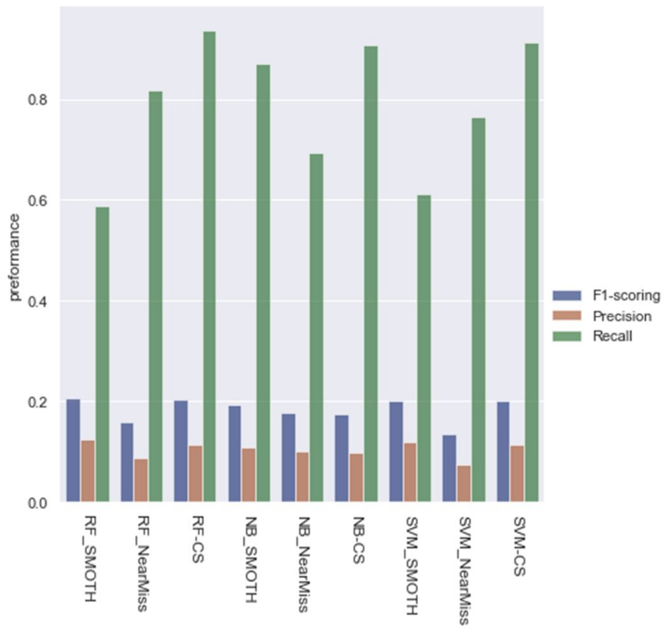
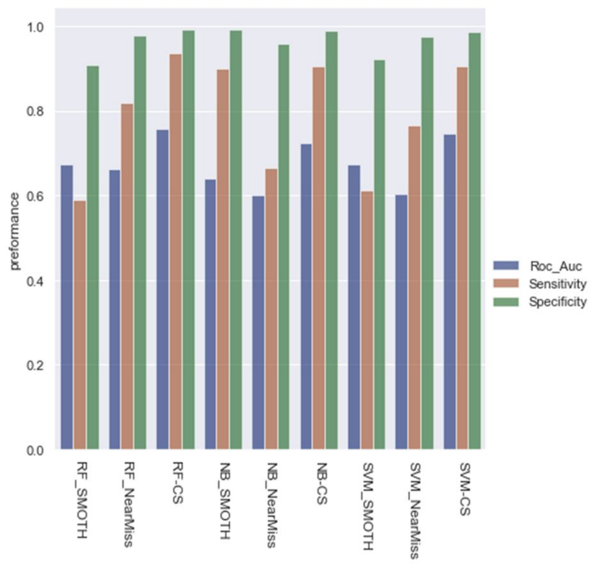


Figure 4 compares the Roc-Auc, Sensitivity and Specificity of each model-method on the validation sample. Roc-AUC and Sensitivity (recall) were discussed previously.

RF-CS, NB-SMOTH and NB-CS have the highest specificity rates (99.1%,99%, 98.81%)

Figure 4:

Bar plot comparison of models' Roc_Auc, Sensitivity, Specificity.



The chosen model is *RF-CS* because it performed best in most metrics. The scores and confusion matrix of the test set are:

- Accuarcy: 62.52
- Roc_Auc: 79.57 %
- G-mean: 77.18 %
- F1-score: 23.75 %
- F2-score: 43.65 %
- AUC-PR: 56.23 %

Figure 5:

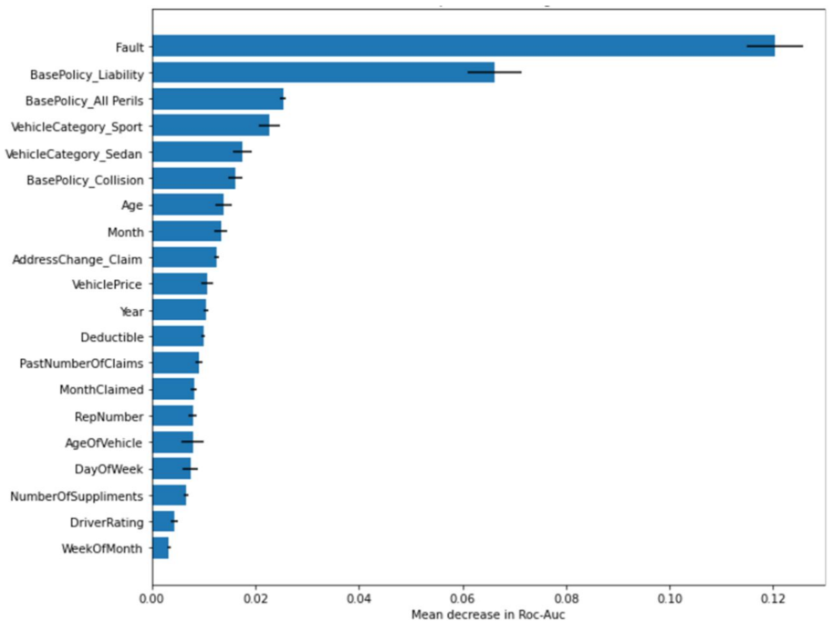
Confusion matrix of the chosen model RF-CS.

Figure 6 summarizes the results of feature importance using the mean decrease in Roc-auc of the chosen model (RF-CS), as can be seen from figure 2, the information about whose fault it was: Policy Holder or Third Party had the biggest influence on the decrease in roc auc by a large margin. The second most noticeable feature was whether the base policy was liability (damages to other people's property as well as medical costs).

Figure 6:

Feature Importance of the chosen model using Roc-Auc.

The black line represents Standard error.



Discussion

Elad Golan's Discussion:

We implement and compare machine learning methods on car insurance claims highly imbalanced dataset under 5 metrics (G-mean, AUC-ROC, F1-scoring, F2-scoring and AUC-PR). In previous works, imbalanced classification methods can be categorized into: oversampling, undersampling, cost sensitivity [1] [3] and imbalanced feature selection methods [2]. We establish our models based on the literature review.

The results show that the performances of three types of models are the lowest when we use under-sampling Near-Miss technique. A feasible cause for this result is the limitation of our data in terms of the relatively small groups in some categorial features and the disadvantage of using OneHot encoder (getting a Sparse feature matrix) as mentioned In Data description and preprocessing section. it may lead to discarding important samples and information when implementing under-sampling.

As seen in previous research [1] GSVM-RU and SVM-RANDU have good performance. Hence, we still recommend considering the implementation of under-sampling techniques, depending on the type and size of data available to the car insurance companies, besides it is the most efficient method.

The significant decrease in models' performances as a result of decomposition-based filter selection application may derive from the use K-means as part of the algorithm that performs not well when dataset consists of categorical, ordinal, and numeric variables that have not been normalized before using the method [19].

RF-CS (The chosen model) cannot be applied as an automated fraud detection algorithm because of low precision; however, it classifies right almost certainly as a fraudulent the test observation (few FN), therefore it can be implemented as a first filter for fraud detection. Only observations identified as fraud will be investigated further, thus reducing the number of claims that need to undergo credibility testing. As a result, insurance companies will save money, time and be able to invest more effort in identifying real potential fraud claims.

For further research, can be helpful examining additional methods that have been presented in the literature, improving the preprocessing, exploring of feature engineering applications and using methods such as cross validation to select hyper parameters in order to maximize each models' performance and for "fair" comparison.

Dov Tuch's Discussion:

As mentioned in the literature review, a comparison between learning models on a balanced data set and a discussion on the importance of feature selection was held [3]. Two of the other studies suggested different approaches when modeling highly imbalanced data. Resampling methods, cost-sensitive learning [1] and decomposition-based feature selection [2]. The present study was therefore designed to compare the combined effects of the best performing models from study [3] and the proposed methods in studies [1], [2]. The results of our study agree with the empirical evidence of study [3]. The cost sensitive ensemble method RF-CS performed best in terms of sensitivity (TP rate) and extremely high specificity rate as well as NB-SMOTH. Which means that if a claim is classified as non-fraudulent there is high probability that it is indeed so [3]. This could be useful for insurance companies to quickly handle deemed-honest claims, removing unnecessary processing time and allocating more resources to find the fraudulent claims. A possible explanation to RF-CS outperforming RF-SMOTH is that for the large data sets, cost-sensitive learning does often yield better results than oversampling [16] because the larger amount of training data makes it easier to estimate the class-membership probabilities more accurately.

The results of traditional feature selection beating decomposition based are concurrent with study [2]. It was shown there that only for features selection of k >= 45 decomposition outperforms traditional one. A possible explanation is that in study [2] they used data sets with at least 6,000 features before preprocessing. Our data had 33 features before preprocessing and we used feature selection of k <= 14. This suggests that we should not extrapolate those findings for data sets below 6,000. Another explanation is that by using OneHot encoding we made sparse features (no data points are missing, but most of them have zero value). It was shown [20] that k-means algorithm is not robust to sparse features and instead we should have used the entropy-weighted k-means that is better suited for this problem. Perhaps a future study about the effectiveness of decomposition methods for data sets with a lower number of features should be held.

References

[1] Y. Tang, Y. Zhang, N. V. Chawla and S. Krasser, "SVMs Modeling for Highly Imbalanced Classification," in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 39, no. 1, pp. 281-288, Feb. 2009, doi: 10.1109/TSMCB.2008.2002909.
[2] J. Van Hulse, T. M. Khoshgoftaar, A. Napolitano and R. Wald, "Feature Selection with High-Dimensional Imbalanced Data," 2009 IEEE International Conference on Data Mining Workshops, 2009, pp. 507-514, doi: 10.1109/ICDMW.2009.35.
[3] Petr Hajek, Roberto Henriques, Mining corporate annual reports for intelligent detection of financial statement fraud – A comparative study of machine learning methods, Knowledge-Based Systems, Volume 128, 2017, Pages 139-152, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2017.05.001.
[4] Arushi Prakash ,Working With Sparse Features In Machine Learning Models, www.kdnuggets.com/2021/01/sparse-features-machine-learning-models.html
[5] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," Intelligent Data Analysis, vol. 6, no. 5, pp. 429–449, 2002.
[6] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: special issue on learning from imbalanced data sets," SIGKDD Explorations, vol. 6, no. 1, pp. 1–6, 2004.
[7] Pes B, Lai G. 2021. Cost-sensitive learning strategies for high-dimensional and imbalanced data: a comparative study. PeerJ Computer Science 7:e832.
[8] L. A. Jeni, J. F. Cohn and F. De La Torre, "Facing Imbalanced Data--Recommendations for the Use of Performance Metrics," 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction, 2013, pp. 245-251, doi: 10.1109/ACII.2013.47.
[9] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: one-sided selection," in Proc. of the 14th International Conference on Machine Learning (ICML1997), pp. 179–186, 1997.
[10] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms.," Pattern Recognition, vol. 30, no. 7, pp. 1145–1159, 1997.
[11] C. J. Van Rijsbergen, Information Retrieval, 2nd edition. London: Butterworths, 1979.
[9] Jason Brownlee, "SMOTE for Imbalanced Classification with Python", January.2020 in Imbalanced Classification, machine learning mastery website blog.
[12] Jason Brownlee, "Undersampling Algorithms for Imbalanced Classification", January.2020 in Imbalanced Classification, machine learning mastery website blog.
[13] Metacost: A general method for making classifiers cost-sensitive, P Domingos, Cited by 1970 s.
[14] H. He and E. A. Garcia, "Learning from Imbalanced Data," in IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pp. 1263-1284, Sept. 2009, doi: 10.1109/TKDE.2008.239.

3979 paper citations.
[15] Abbasi, Ahmed, Conan Albrecht, Anthony Vance, and James Hansen. "MetaFraud: A Meta-Learning Framework for Detecting Financial Fraud." MIS Quarterly 36, no. 4 (2012): 1293–1327. https://doi.org/10.2307/41703508.
[16] McCarthy, Kate & Zabar, Bibi & Weiss, Gary. (2005). Does Cost-Sensitive Learning Beat Sampling for Classifying Rare Classes. Learning. 10.1145/1089827.1089836. 210 citations
[17] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning: with Applications in R. New York: Springer, 2013.
[18] ebb, G. I.; Boughton, J.; Wang, Z. (2005). "Not So Naive Bayes: Aggregating One-Dependence Estimators". Machine Learning. 58 (1): 5–24. doi:10.1007/s10994-005-4258-6
[19] Ahmad, Amir & Dey, Lipika. (2007). A k-mean clustering algorithm for mixed numeric and categorical data. Data & Knowledge Engineering. 63. 503-527. 10.1016/j.datak.2007.03.016.
[20] L. Jing, M. K. Ng and J. Z. Huang, "An Entropy Weighting k-Means Algorithm for Subspace Clustering of High-Dimensional Sparse Data," in IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 8, pp. 1026-1041, Aug. 2007, doi: 10.1109/TKDE.2007.1048.

Appendix

Link to GitHub repo of our work: https://github.com/Eladgo10/DS-Seminar-project-

Link to Jupyter notebook: https://github.com/Eladgo10/DS-Seminar-project-/blob/main/seminer-project.ipynb

seminer-project._1.0

April 30, 2022

1 Welcome

1.0.1 Seminar in data sceince, project notebook

- Tel Aviv university, department of economics
- Authors: Elad Golan & Dov Tuch
- Fraud Detection in car insurance

2 Part 0 - Load packages and dataset

2.0.1 0.a - Packages

[ ]:
```
#imort packages
import numpy as np # linear algebra
import pandas as pd # Data frames
import seaborn as sns # plots
import matplotlib.pyplot as plt #plots
import sklearn # Data science package
from sklearn.impute import SimpleImputer # for replace NA with avarge
from sklearn.model_selection import train_test_split  # for spliting the data
from sklearn.ensemble import RandomForestClassifier # RF classifier
from sklearn.tree import plot_tree
from sklearn.inspection import permutation_importance  #feature importance
from sklearn.pipeline import make_pipeline
from sklearn.svm import SVC # SVM classifier
from sklearn.feature_selection import SelectKBest, chi2,mutual_info_classif␣
 ↪#feature selection
from sklearn.preprocessing import StandardScaler # normlize features
from sklearn.naive_bayes import BernoulliNB , ComplementNB, CategoricalNB #NB␣
 ↪classifiers
# for evaluating preformance
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score, accuracy_score, confusion_matrix,␣
 ↪ConfusionMatrixDisplay, f1_score,fbeta_score, precision_recall_curve,auc
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
```

```
##
from collections import Counter
import researchpy as rp #for crosstab
import scipy.stats as stats # chi^2 test
from scipy.stats import pearsonr #correlation
import random #  for comuting random seed
 #preprocessing
from category_encoders.ordinal import OrdinalEncoder
from category_encoders.binary import BinaryEncoder
from category_encoders.one_hot import OneHotEncoder
##
import imblearn
from imblearn.over_sampling import SMOTE #oversampling
from imblearn.metrics import geometric_mean_score #metric
from imblearn.under_sampling import NearMiss #undersampling
```

[ ]:
```
np.random.seed(2020)
```

2.0.2 0.b - Load Vechicle Insures Dataset

[ ]:
```
url = "https://raw.githubusercontent.com/Eladgo10/DS-Seminar-project-/main/
 ↪fraud_oracle.csv" #github link
dataset = pd.read_csv(url)
dataset
```

[ ]:
```
      Month  WeekOfMonth  DayOfWeek     Make AccidentArea DayOfWeekClaimed  \
0       Dec            5  Wednesday    Honda        Urban          Tuesday
1       Jan            3  Wednesday    Honda        Urban           Monday
2       Oct            5     Friday    Honda        Urban         Thursday
3       Jun            2   Saturday   Toyota        Rural           Friday
4       Jan            5     Monday    Honda        Urban          Tuesday
…     …          …          …      …       …            …
15415   Nov            4     Friday   Toyota        Urban          Tuesday
15416   Nov            5   Thursday  Pontiac        Urban           Friday
15417   Nov            5   Thursday   Toyota        Rural           Friday
15418   Dec            1     Monday   Toyota        Urban         Thursday
15419   Dec            2  Wednesday   Toyota        Urban         Thursday

      MonthClaimed  WeekOfMonthClaimed     Sex MaritalStatus  …  \
0              Jan                   1  Female        Single  …
1              Jan                   4    Male        Single  …
2              Nov                   2    Male       Married  …
3              Jul                   1    Male       Married  …
4              Feb                   2  Female        Single  …
…            …                 …     …           …   …
15415          Nov                   5    Male       Married  …
15416          Dec                   1    Male       Married  …
```

```
15417          Dec                   1    Male        Single  …
15418          Dec                   2  Female       Married  …
15419          Dec                   3    Male        Single  …

      AgeOfVehicle AgeOfPolicyHolder PoliceReportFiled WitnessPresent  \
0          3 years          26 to 30                No             No
1          6 years          31 to 35               Yes             No
2          7 years          41 to 50                No             No
3      more than 7          51 to 65               Yes             No
4          5 years          31 to 35                No             No
…              …               …                …             …
15415      6 years          31 to 35                No             No
15416      6 years          31 to 35                No             No
15417      5 years          26 to 30                No             No
15418      2 years          31 to 35                No             No
15419      5 years          26 to 30                No             No

      AgentType  NumberOfSuppliments  AddressChange_Claim  NumberOfCars  Year  \
0      External                 none               1 year        3 to 4  1994
1      External                 none            no change     1 vehicle  1994
2      External                 none            no change     1 vehicle  1994
3      External          more than 5            no change     1 vehicle  1994
4      External                 none            no change     1 vehicle  1994
…         …                  …                   …            …   …
15415  External                 none            no change     1 vehicle  1996
15416  External          more than 5            no change        3 to 4  1996
15417  External               1 to 2            no change     1 vehicle  1996
15418  External          more than 5            no change     1 vehicle  1996
15419  External               1 to 2            no change     1 vehicle  1996

       BasePolicy
0       Liability
1       Collision
2       Collision
3       Liability
4       Collision
…           …
15415   Collision
15416   Liability
15417   Collision
15418  All Perils
15419   Collision

[15420 rows x 33 columns]
```

[ ]:
```
dataset.columns
```

```
[ ]: Index(['Month', 'WeekOfMonth', 'DayOfWeek', 'Make', 'AccidentArea',
       'DayOfWeekClaimed', 'MonthClaimed', 'WeekOfMonthClaimed', 'Sex',
       'MaritalStatus', 'Age', 'Fault', 'PolicyType', 'VehicleCategory',
       'VehiclePrice', 'FraudFound_P', 'PolicyNumber', 'RepNumber',
       'Deductible', 'DriverRating', 'Days_Policy_Accident',
       'Days_Policy_Claim', 'PastNumberOfClaims', 'AgeOfVehicle',
       'AgeOfPolicyHolder', 'PoliceReportFiled', 'WitnessPresent', 'AgentType',
       'NumberOfSuppliments', 'AddressChange_Claim', 'NumberOfCars', 'Year',
       'BasePolicy'],
      dtype='object')
```

3 Part 1 - Desprective Statistics

3.0.1 1.a Describe numeric variables

```
[ ]: dataset.describe()
```

```
[ ]:         WeekOfMonth  WeekOfMonthClaimed           Age  FraudFound_P  \
count  15420.000000        15420.000000  15420.000000  15420.000000
mean       2.788586            2.693969     39.855707      0.059857
std        1.287585            1.259115     13.492377      0.237230
min        1.000000            1.000000      0.000000      0.000000
25%        2.000000            2.000000     31.000000      0.000000
50%        3.000000            3.000000     38.000000      0.000000
75%        4.000000            4.000000     48.000000      0.000000
max        5.000000            5.000000     80.000000      1.000000

       PolicyNumber     RepNumber    Deductible  DriverRating          Year
count  15420.000000  15420.000000  15420.000000  15420.000000  15420.000000
mean    7710.500000      8.483268    407.704280      2.487808   1994.866472
std     4451.514911      4.599948     43.950998      1.119453      0.803313
min        1.000000      1.000000    300.000000      1.000000   1994.000000
25%     3855.750000      5.000000    400.000000      1.000000   1994.000000
50%     7710.500000      8.000000    400.000000      2.000000   1995.000000
75%    11565.250000     12.000000    400.000000      3.000000   1996.000000
max    15420.000000     16.000000    700.000000      4.000000   1996.000000
```

3.0.2 1.b Describe qualtive features

```
[ ]: dataset.describe(include=['object'])
```

```
[ ]:         Month DayOfWeek     Make AccidentArea DayOfWeekClaimed MonthClaimed  \
count   15420     15420    15420        15420            15420        15420
unique     12         7       19            2                8           13
top       Jan    Monday  Pontiac        Urban           Monday          Jan
freq     1411      2616     3837        13822             3757         1446
```

```
          Sex MaritalStatus         Fault         PolicyType  …  \
count   15420         15420         15420              15420  …
unique      2             4             2                  9  …
top      Male       Married  Policy Holder  Sedan - Collision  …
freq    13000         10625         11230               5584  …

       PastNumberOfClaims AgeOfVehicle AgeOfPolicyHolder PoliceReportFiled  \
count               15420        15420             15420             15420
unique                  4            8                 9                 2
top                2 to 4      7 years          31 to 35                No
freq                 5485         5807              5593             14992

       WitnessPresent AgentType NumberOfSuppliments AddressChange_Claim  \
count           15420     15420               15420               15420
unique              2         2                   4                   5
top                No  External                none           no change
freq            15333     15179                7047               14324

       NumberOfCars BasePolicy
count         15420      15420
unique            5          3
top       1 vehicle  Collision
freq          14316       5962

[4 rows x 24 columns]
```

3.0.3 1.c intresting variablas distrabution - bar plots, chi^2 test, common distrabution

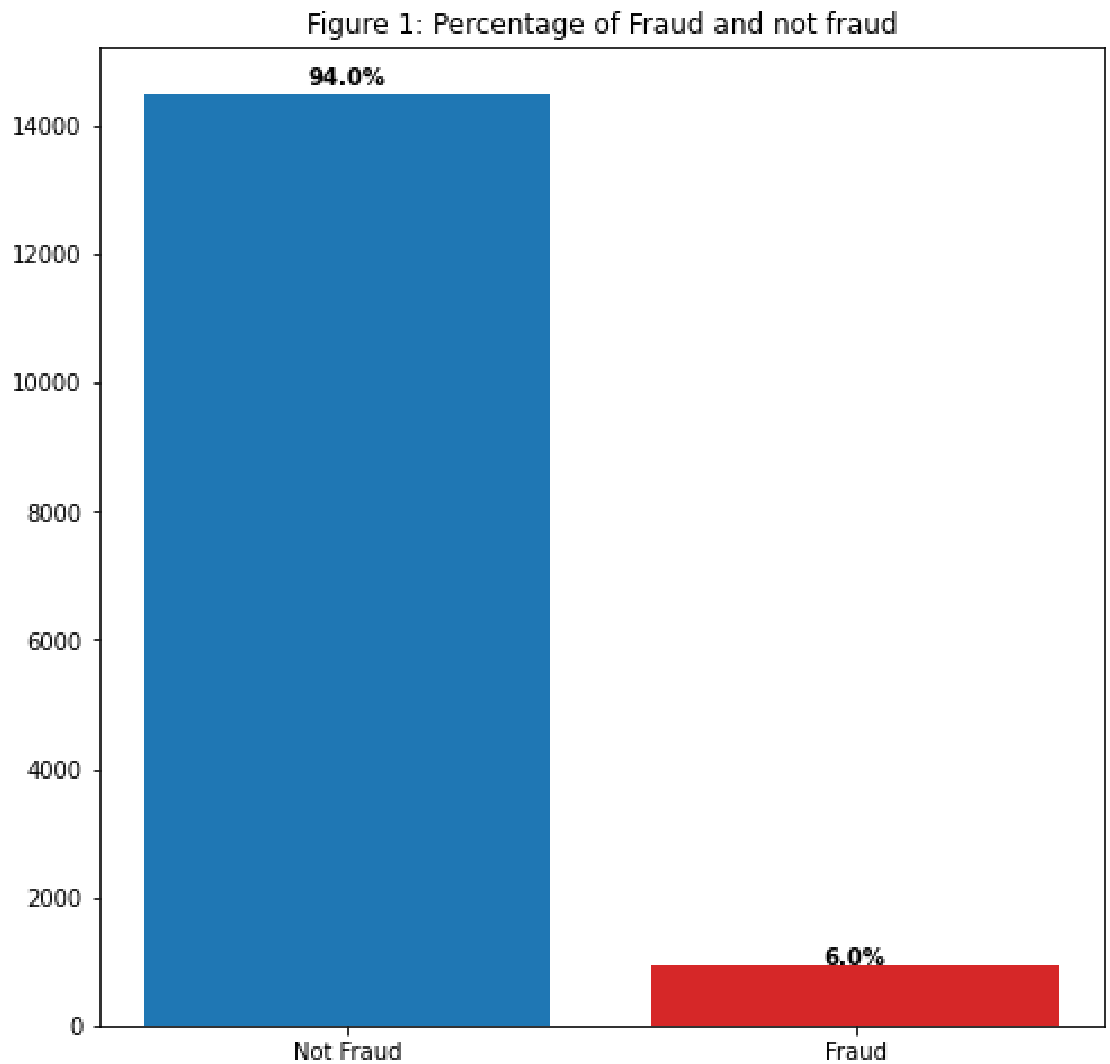
```
[ ]: print(dataset['FraudFound_P'].value_counts(),'\n') # 923 frauds and 14497 not␣
     ↪fraud - outcome
     print(dataset['AgeOfPolicyHolder'].value_counts(),'\n')
     print(dataset['WitnessPresent'].value_counts(), '\n')
     print(dataset['PoliceReportFiled'].value_counts())
```

```
0    14497
1      923
Name: FraudFound_P, dtype: int64

31 to 35    5593
36 to 40    4043
41 to 50    2828
51 to 65    1392
26 to 30     613
over 65      508
16 to 17     320
21 to 25     108
```

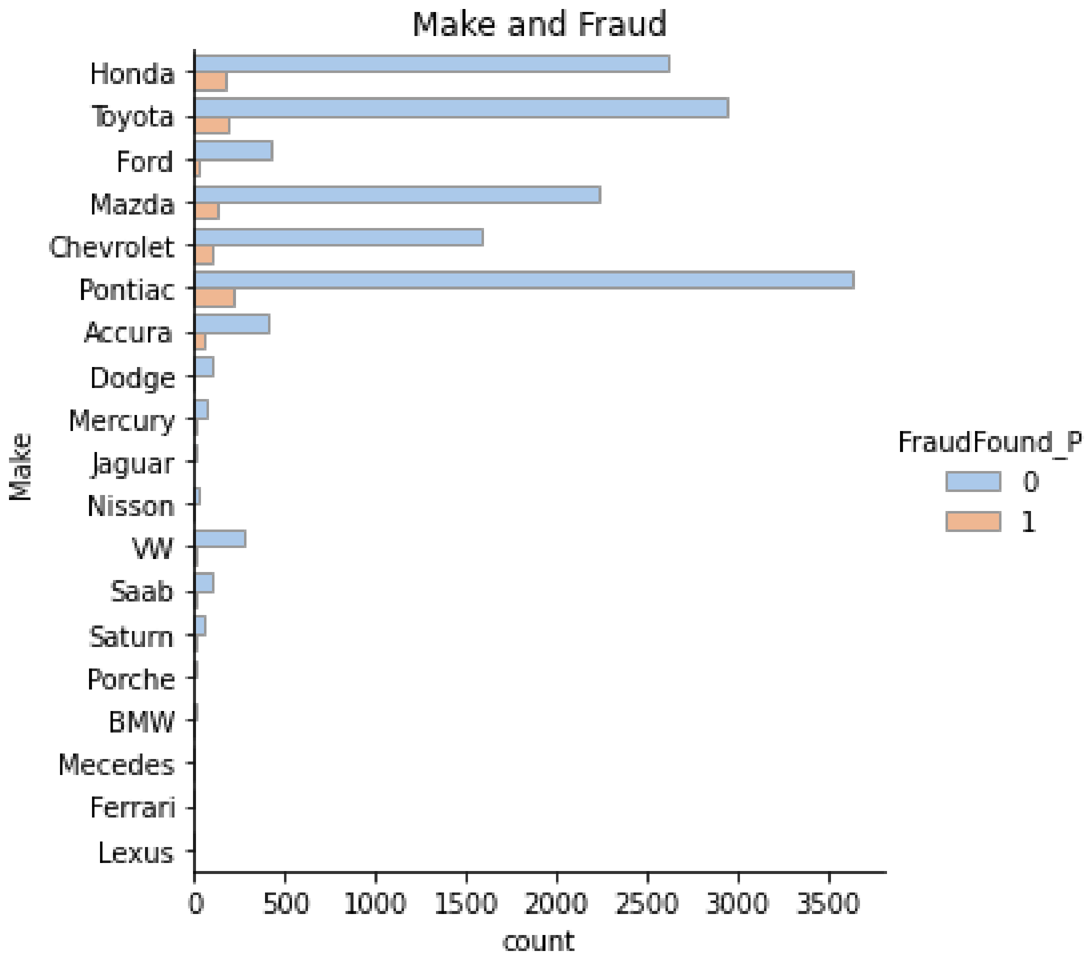
```
18 to 20      15
Name: AgeOfPolicyHolder, dtype: int64

No     15333
Yes       87
Name: WitnessPresent, dtype: int64

No     14992
Yes      428
Name: PoliceReportFiled, dtype: int64
```

bar plots

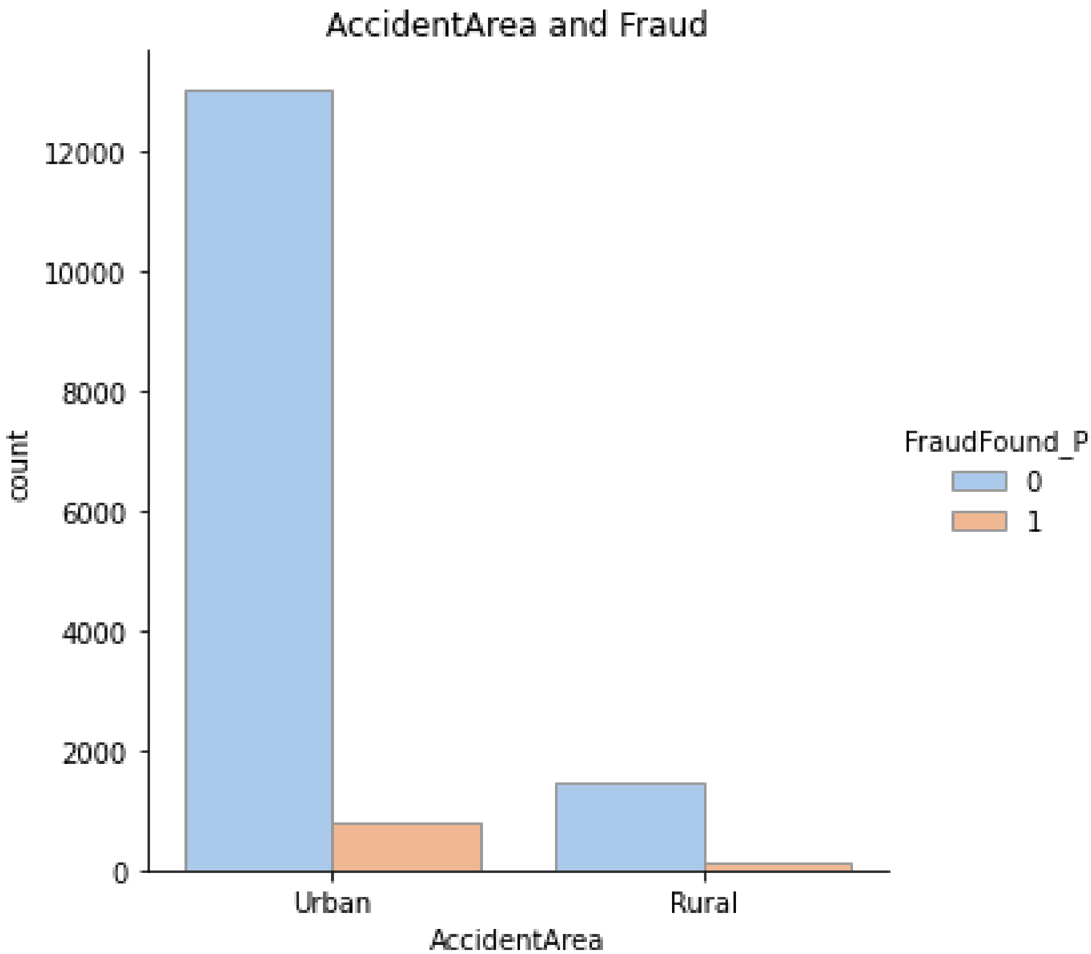
[ ]:
```
# creating data for the plot
data_FraudFound_P = pd.DataFrame({'category':['Not Fraud', 'Fraud'],
                     'counts': dataset['FraudFound_P'].value_counts().values,
                     'percentage': [round(sum(dataset.FraudFound_P == 0)/
 ↪len(dataset), 3)*100 ,
                                    round(sum(dataset.FraudFound_P == 1)/
 ↪len(dataset), 3)*100]
                    })


plt.figure(figsize=(8,8))
colors_list = ['tab:blue', 'tab:red']
graph = plt.bar(data_FraudFound_P.category,data_FraudFound_P.counts, color =␣
 ↪colors_list)
plt.title("Figure 1: Percentage of Fraud and not fraud")

i = 0
for p in graph:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    plt.text(x+width/2,
             y+height*1.01,
             str(data_FraudFound_P.percentage[i])+'%',
             ha='center',
             weight='bold')
    i+=1
plt.show()
```



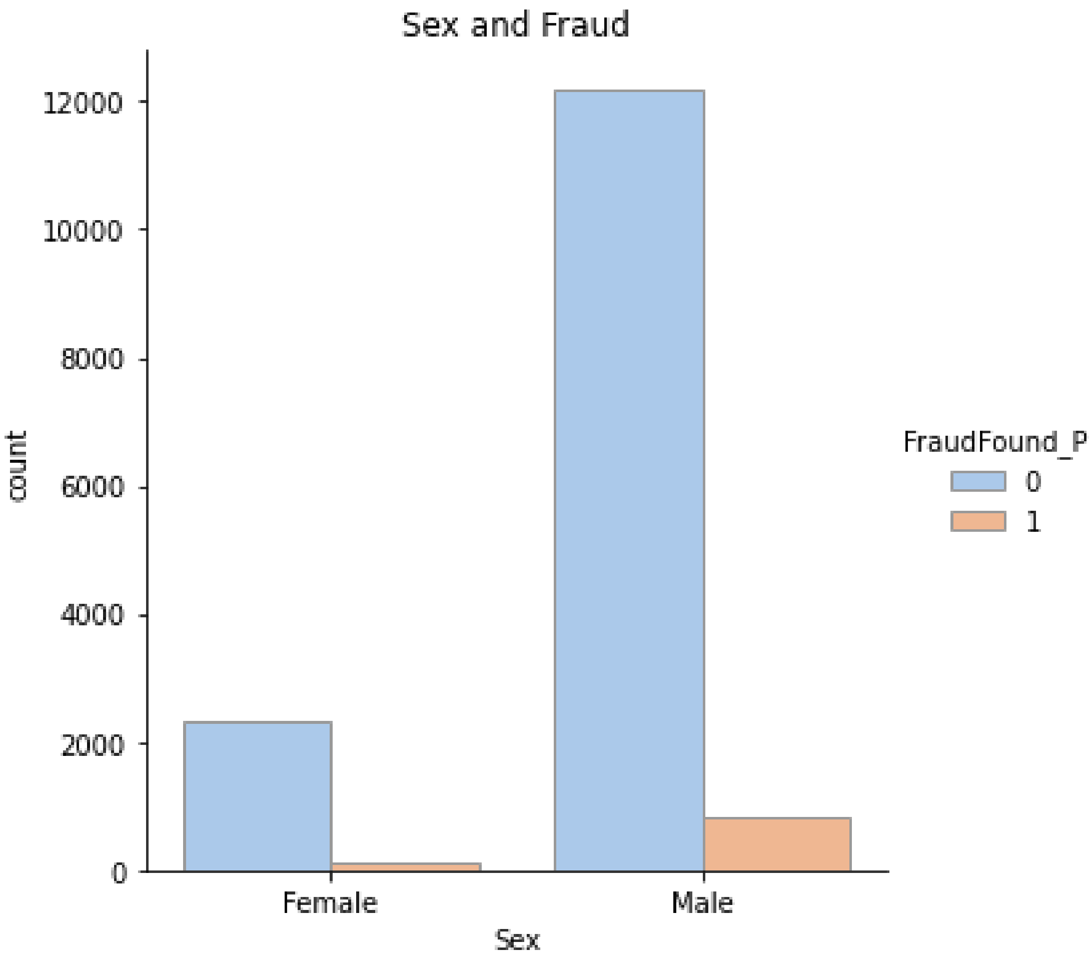
[ ]:
```
sns.catplot(y="Make", hue="FraudFound_P", kind="count",
            palette="pastel", edgecolor=".6",
            data=dataset).set(title = "Make and Fraud")
plt.show()
```



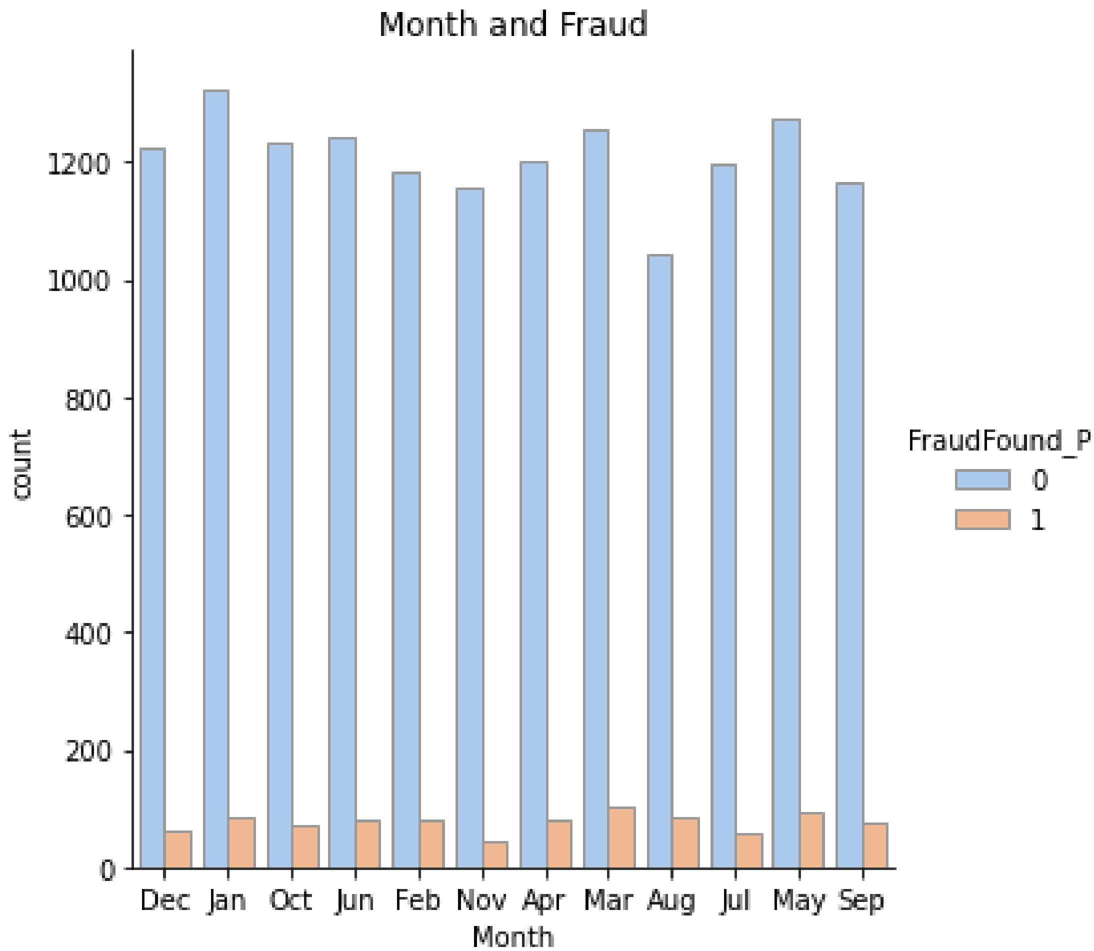
[ ]:
```
sns.catplot(x="AccidentArea", hue="FraudFound_P", kind="count",
            palette="pastel", edgecolor=".6",
            data=dataset).set(title = "AccidentArea and Fraud")
plt.show()
```



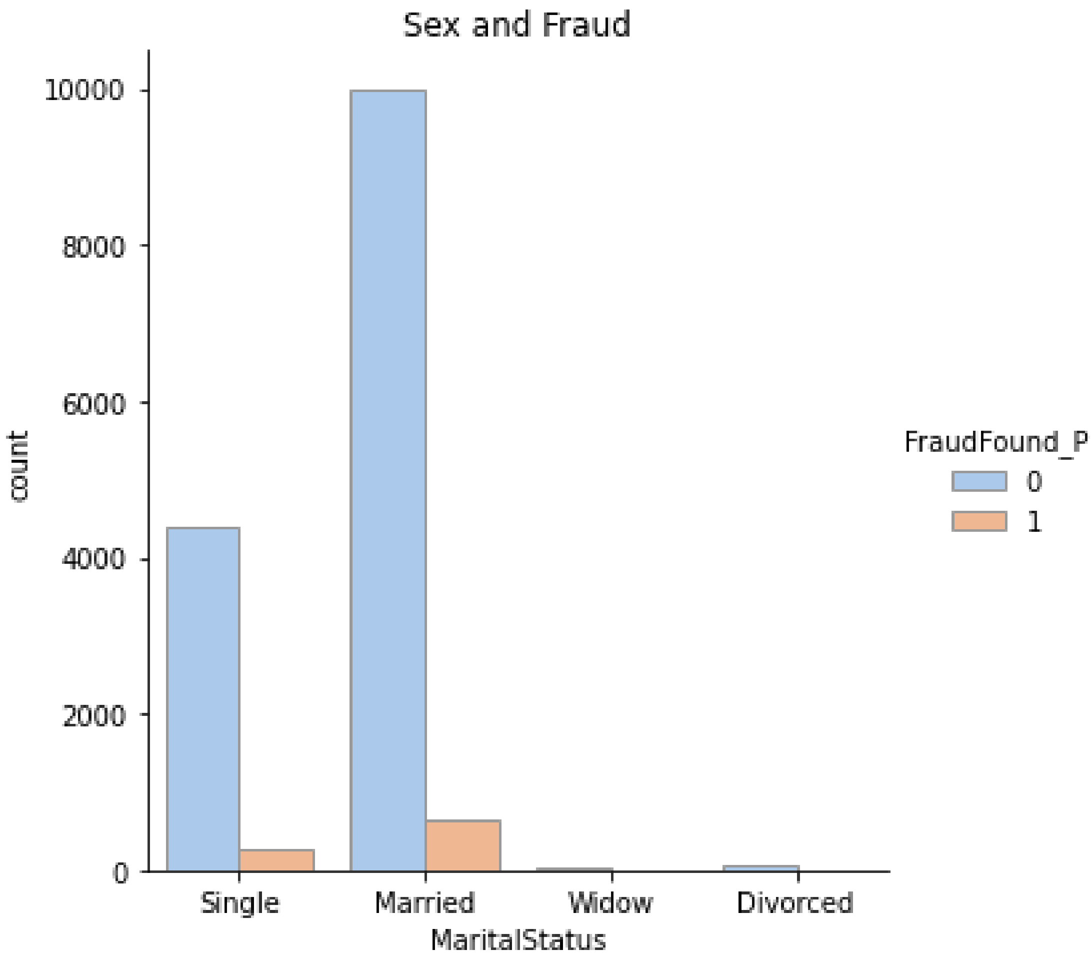
[ ]:
```
sns.catplot(x="Sex", hue="FraudFound_P", kind="count",
            palette="pastel", edgecolor=".6",
            data=dataset).set(title = "Sex and Fraud")
plt.show()
```



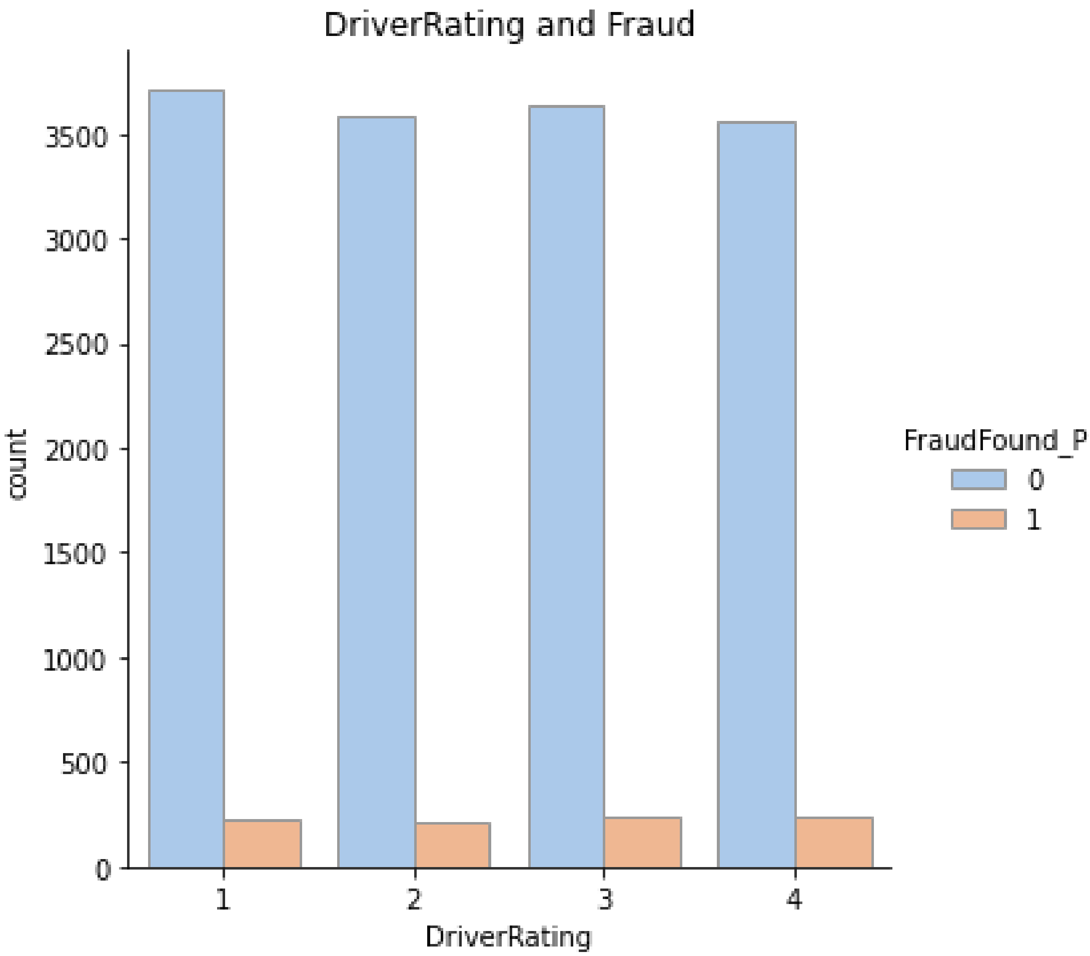

```
[ ]: sns.catplot(x="Month", hue="FraudFound_P", kind="count",
                 palette="pastel", edgecolor=".6",
                 data=dataset).set(title = "Month and Fraud")
     plt.show()
```



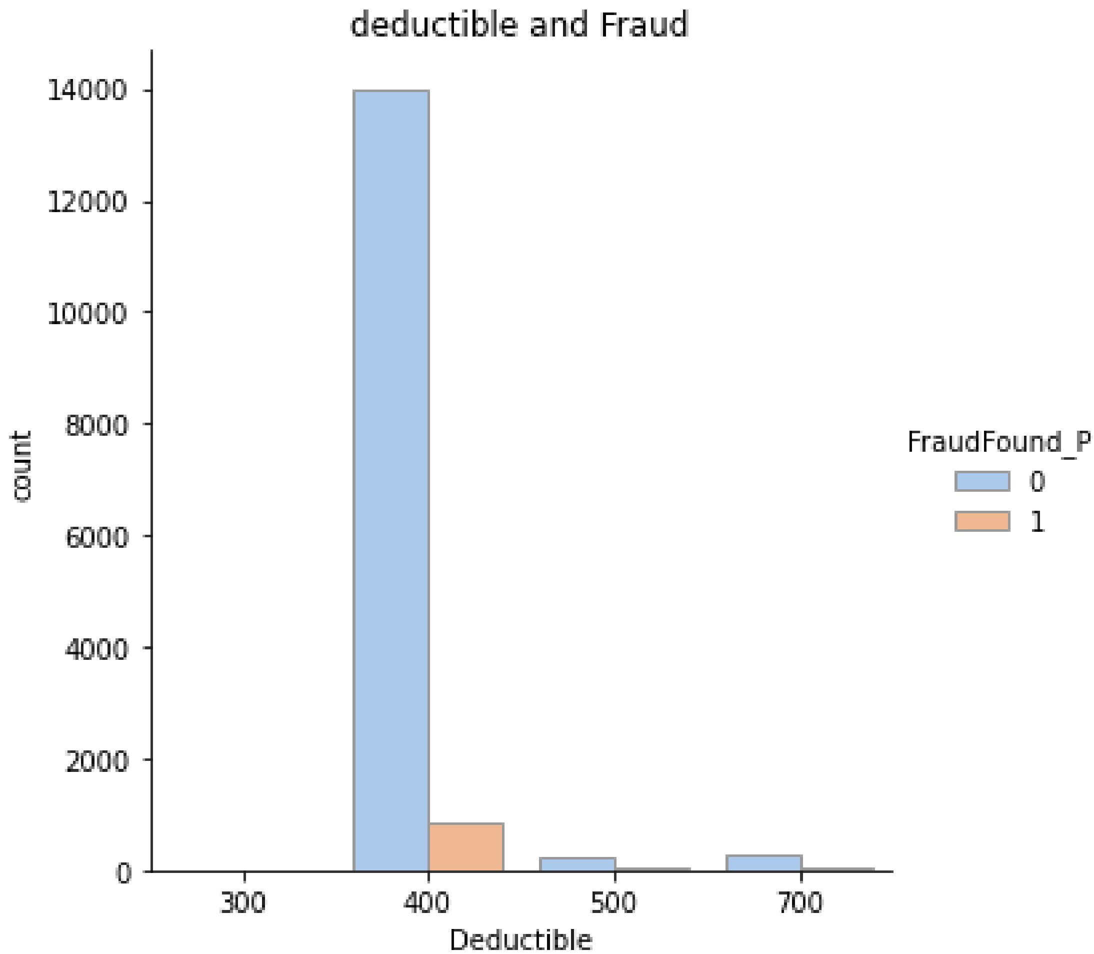
[ ]:
```
sns.catplot(x="MaritalStatus", hue="FraudFound_P", kind="count",
            palette="pastel", edgecolor=".6",
            data=dataset).set(title = "Sex and Fraud")
plt.show()
```



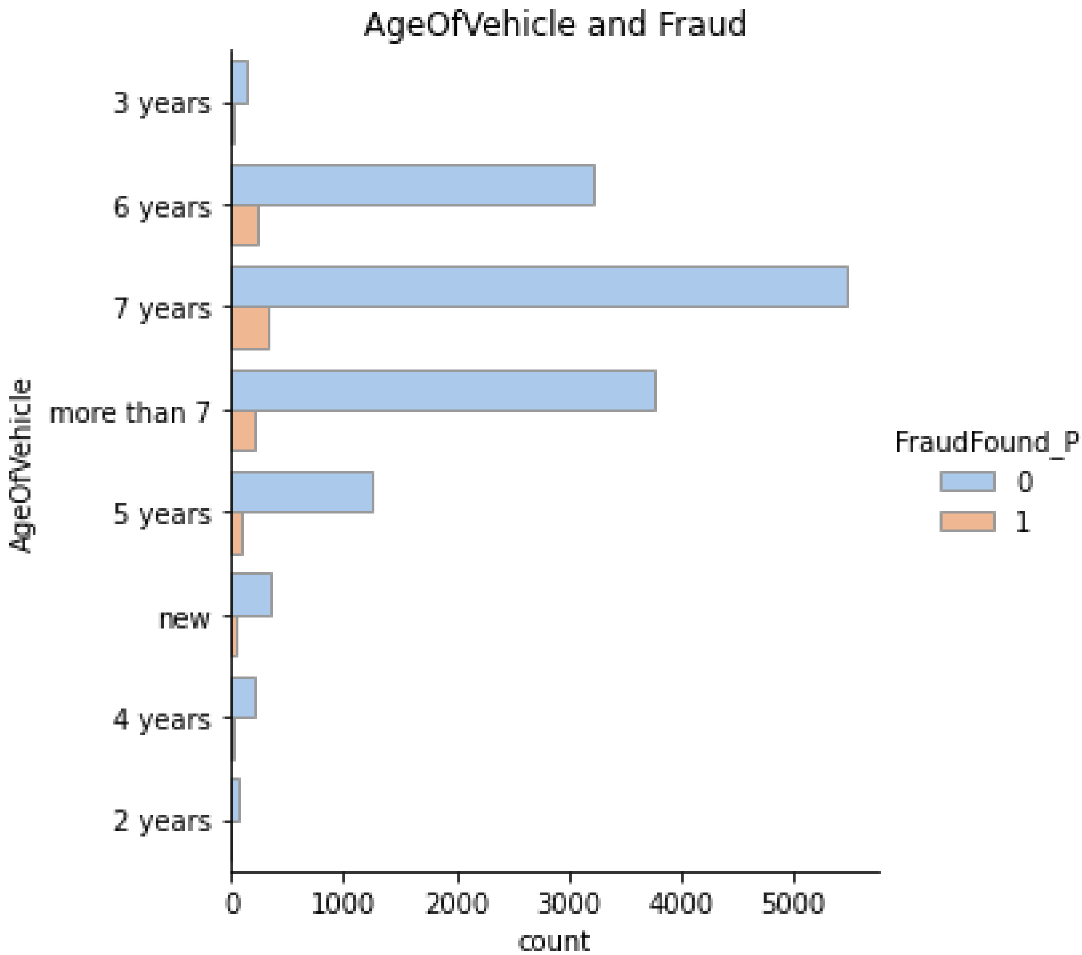

```
[ ]: sns.catplot(x="DriverRating", hue="FraudFound_P", kind="count",
                 palette="pastel", edgecolor=".6",
                 data=dataset).set(title = "DriverRating and Fraud")
     plt.show()
```



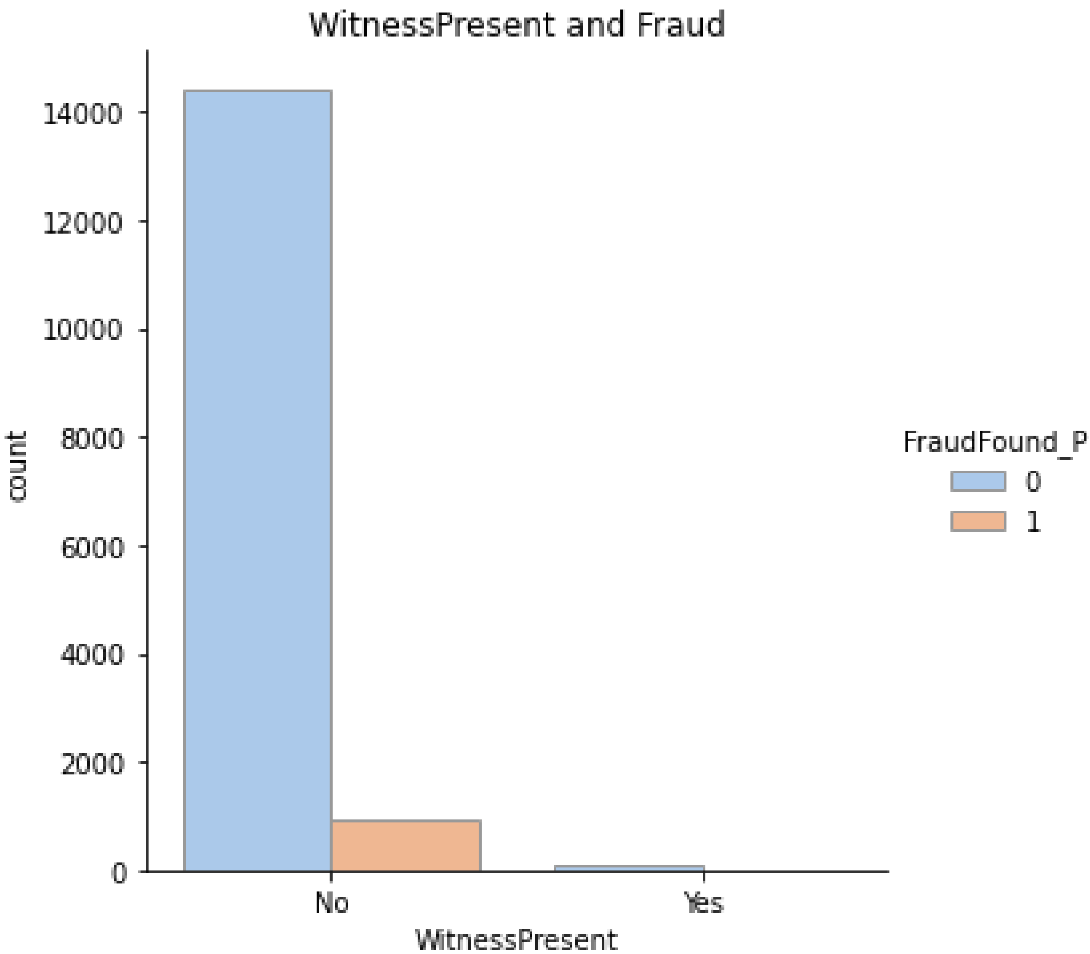

```
[ ]: sns.catplot(x="Deductible", hue="FraudFound_P", kind="count",
                 palette="pastel", edgecolor=".6",
                 data=dataset).set(title = "deductible and Fraud")
     plt.show()
```



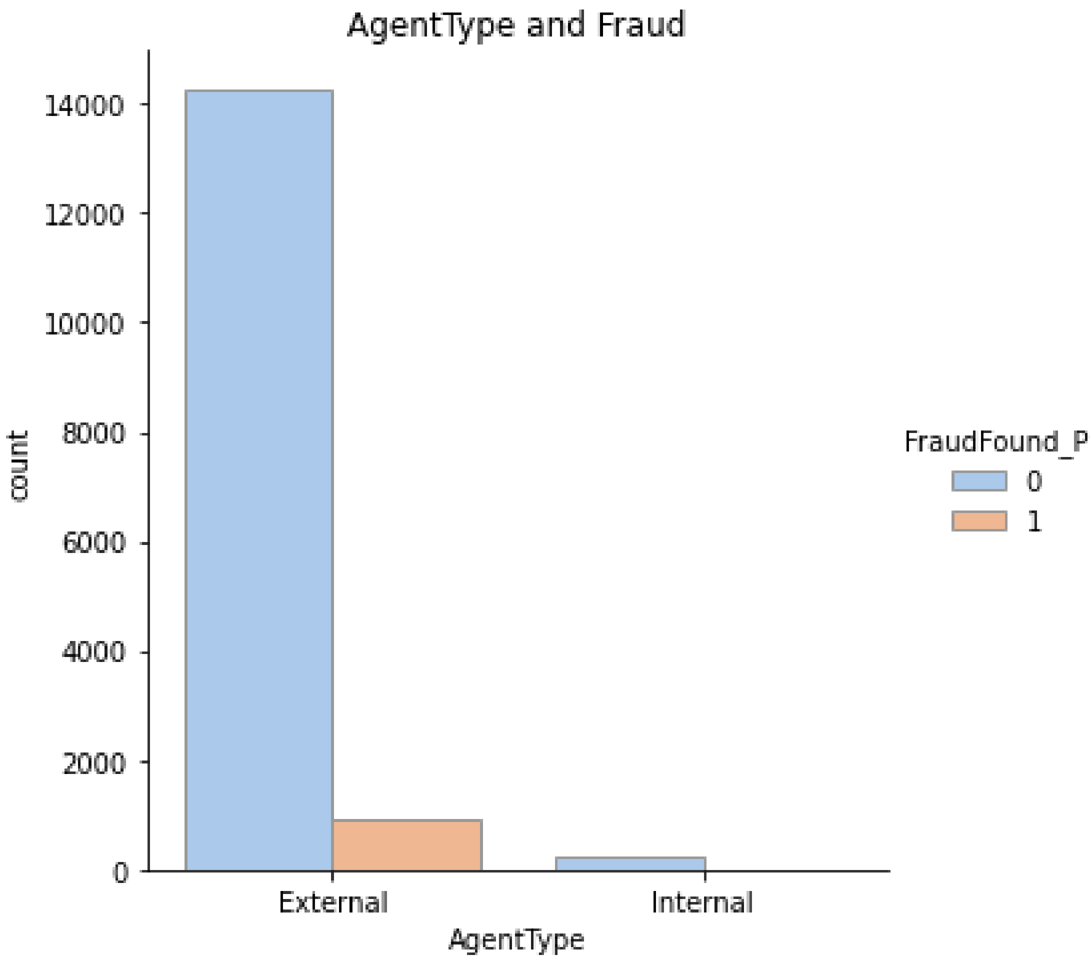
[ ]:
```
sns.catplot(y="AgeOfVehicle", hue="FraudFound_P", kind="count",
            palette="pastel", edgecolor=".6",
            data=dataset).set(title = "AgeOfVehicle and Fraud")
plt.show()
```



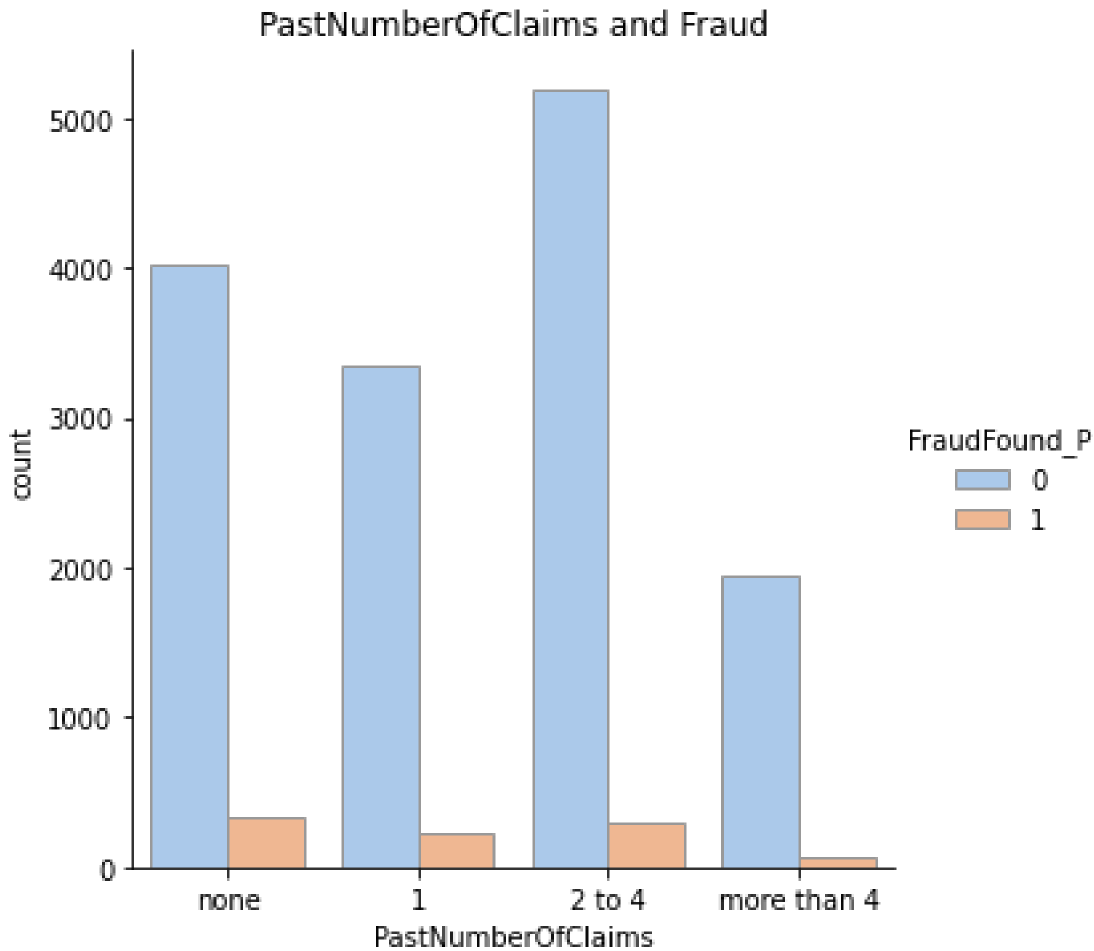
[ ]:
```
sns.catplot(x="WitnessPresent", hue="FraudFound_P", kind="count",
            palette="pastel", edgecolor=".6",
            data=dataset).set(title = "WitnessPresent and Fraud")
plt.show()
```



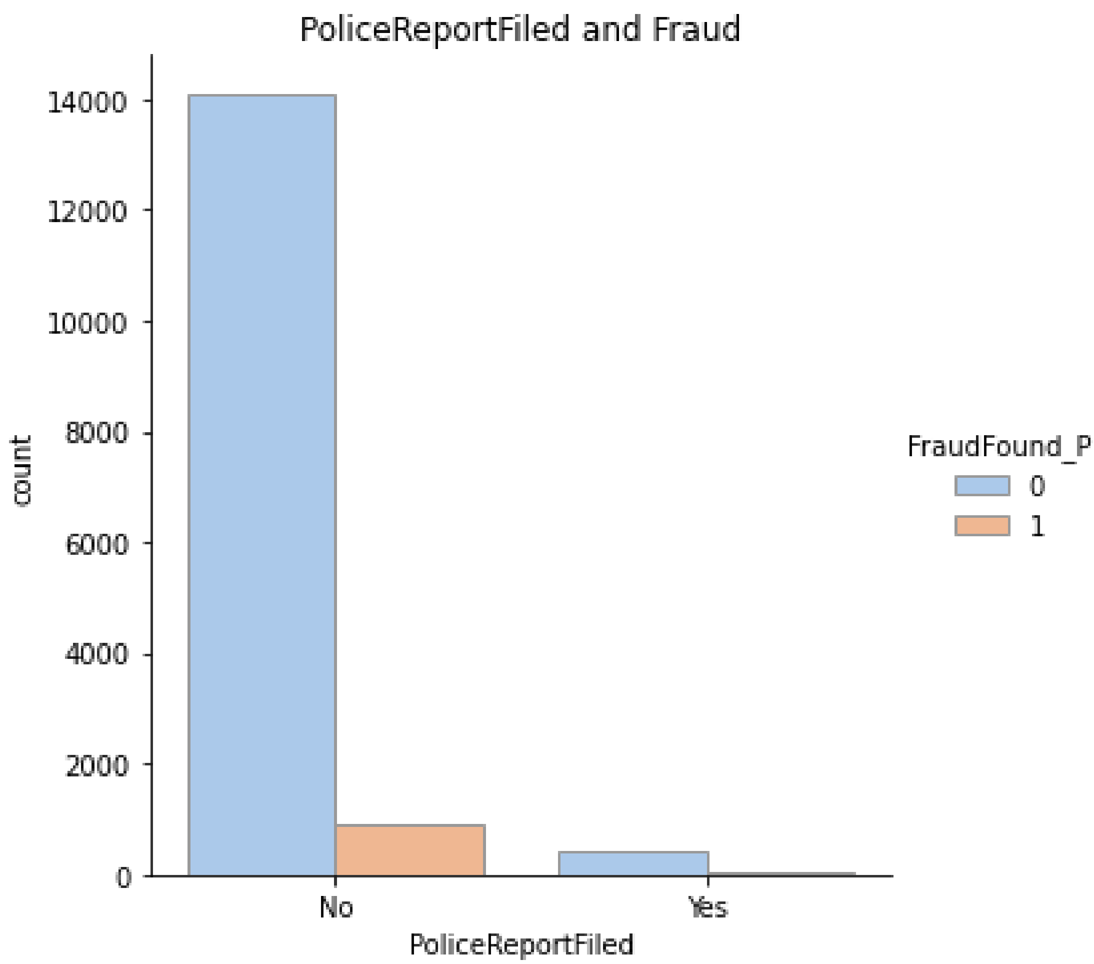
[ ]:
```
sns.catplot(x="AgentType", hue="FraudFound_P", kind="count",
            palette="pastel", edgecolor=".6",
            data=dataset).set(title = "AgentType and Fraud")
plt.show()
```



[ ]:
```
sns.catplot(x="PastNumberOfClaims", hue="FraudFound_P", kind="count",
            palette="pastel", edgecolor=".6",
            data=dataset).set(title = "PastNumberOfClaims and Fraud")
plt.show()
```



[ ]:
```
sns.catplot(x="PoliceReportFiled", hue="FraudFound_P", kind="count",
            palette="pastel", edgecolor=".6",
            data=dataset).set(title = "PoliceReportFiled and Fraud")
plt.show()
```



[ ]:
```
dataset.columns[~dataset.columns.isin(["Age", 'Month'])]
```

[ ]:
```
Index(['WeekOfMonth', 'DayOfWeek', 'Make', 'AccidentArea', 'DayOfWeekClaimed',
       'MonthClaimed', 'WeekOfMonthClaimed', 'Sex', 'MaritalStatus', 'Fault',
       'PolicyType', 'VehicleCategory', 'VehiclePrice', 'FraudFound_P',
       'PolicyNumber', 'RepNumber', 'Deductible', 'DriverRating',
       'Days_Policy_Accident', 'Days_Policy_Claim', 'PastNumberOfClaims',
       'AgeOfVehicle', 'AgeOfPolicyHolder', 'PoliceReportFiled',
       'WitnessPresent', 'AgentType', 'NumberOfSuppliments',
       'AddressChange_Claim', 'NumberOfCars', 'Year', 'BasePolicy'],
      dtype='object')
```

^2 test

[ ]:
```
col_names = dataset.columns[~dataset.columns.isin(['Age', 'FraudFound_P'])]
for i in dataset.columns[~dataset.columns.isin(['Age', 'FraudFound_P'])]:
    col_names = col_names[col_names != i]
```

```python
    for j in col_names:
        crosstab, test_results, expected = rp.crosstab(dataset[i], dataset[j],
                                                       test= "chi-square",
                                                       expected_freqs= True,
                                                       prop= "cell")
        if test_results['results'][1] < 0.05:
            print(i + ' ' + j + ':')
            print(test_results)
            print('\n')
    print('****** new ******')
```

```
Month WeekOfMonth:
                 Chi-square test   results
0  Pearson Chi-square ( 44.0) =   149.1212
1                     p-value =     0.0000
2                  Cramer's V =     0.0492


Month DayOfWeek:
                 Chi-square test   results
0  Pearson Chi-square ( 66.0) =   201.1752
1                     p-value =     0.0000
2                  Cramer's V =     0.0466


Month DayOfWeekClaimed:
                 Chi-square test   results
0  Pearson Chi-square ( 77.0) =   199.4474
1                     p-value =     0.0000
2                  Cramer's V =     0.0430


Month MonthClaimed:
                  Chi-square test     results
0  Pearson Chi-square ( 132.0) =   94726.2341
1                      p-value =       0.0000
2                   Cramer's V =       0.7473


Month WeekOfMonthClaimed:
                 Chi-square test   results
0  Pearson Chi-square ( 44.0) =   234.5356
1                     p-value =     0.0000
2                  Cramer's V =     0.0617
```

```
Month PolicyType:
                 Chi-square test   results
0  Pearson Chi-square ( 88.0) =   127.0620
1                     p-value =     0.0041
2                  Cramer's V =     0.0321


Month VehiclePrice:
                 Chi-square test  results
0  Pearson Chi-square ( 55.0) =   78.5796
1                     p-value =    0.0201
2                  Cramer's V =    0.0319


Month AgeOfVehicle:
                 Chi-square test   results
0  Pearson Chi-square ( 77.0) =   269.3303
1                     p-value =     0.0000
2                  Cramer's V =     0.0500


Month AgeOfPolicyHolder:
                 Chi-square test   results
0  Pearson Chi-square ( 88.0) =   212.3422
1                     p-value =     0.0000
2                  Cramer's V =     0.0415


Month PoliceReportFiled:
                 Chi-square test  results
0  Pearson Chi-square ( 11.0) =   46.8558
1                     p-value =    0.0000
2                  Cramer's V =    0.0551


Month NumberOfCars:
                 Chi-square test   results
0  Pearson Chi-square ( 44.0) =   265.3381
1                     p-value =     0.0000
2                  Cramer's V =     0.0656


Month Year:
                 Chi-square test   results
0  Pearson Chi-square ( 22.0) =   113.4837
1                     p-value =     0.0000
2                  Cramer's V =     0.0607
```

```
Month BasePolicy:
                 Chi-square test  results
0  Pearson Chi-square ( 22.0) =   44.5829
1                    p-value =    0.0030
2                 Cramer's V =    0.0380


****** new ******
WeekOfMonth DayOfWeek:
                 Chi-square test  results
0  Pearson Chi-square ( 24.0) =   39.4612
1                    p-value =    0.0244
2                 Cramer's V =    0.0253


WeekOfMonth MonthClaimed:
                 Chi-square test   results
0  Pearson Chi-square ( 48.0) =   121.6322
1                    p-value =     0.0000
2                 Cramer's V =     0.0444


WeekOfMonth WeekOfMonthClaimed:
                 Chi-square test    results
0  Pearson Chi-square ( 16.0) =   9935.7556
1                    p-value =      0.0000
2                 Cramer's V =      0.4014


WeekOfMonth Fault:
                Chi-square test  results
0  Pearson Chi-square ( 4.0) =   10.1096
1                   p-value =    0.0386
2                Cramer's V =    0.0256


WeekOfMonth PolicyType:
                 Chi-square test  results
0  Pearson Chi-square ( 32.0) =   67.2744
1                    p-value =    0.0003
2                 Cramer's V =    0.0330


WeekOfMonth VehicleCategory:
                Chi-square test  results
0  Pearson Chi-square ( 8.0) =   17.3118
1                   p-value =    0.0270
```

```
2                 Cramer's V =    0.0237


WeekOfMonth Days_Policy_Accident:
                 Chi-square test  results
0  Pearson Chi-square ( 16.0) =   29.0664
1                    p-value =    0.0235
2                 Cramer's V =    0.0217


****** new ******
DayOfWeek AccidentArea:
                Chi-square test  results
0  Pearson Chi-square ( 6.0) =   14.9448
1                   p-value =    0.0207
2                Cramer's V =    0.0311


DayOfWeek DayOfWeekClaimed:
                 Chi-square test    results
0  Pearson Chi-square ( 42.0) =   1959.6262
1                    p-value =      0.0000
2                 Cramer's V =      0.1455


DayOfWeek MaritalStatus:
                 Chi-square test  results
0  Pearson Chi-square ( 18.0) =   30.9558
1                    p-value =    0.0291
2                 Cramer's V =    0.0259


DayOfWeek Fault:
                Chi-square test  results
0  Pearson Chi-square ( 6.0) =   29.0767
1                   p-value =    0.0001
2                Cramer's V =    0.0434


DayOfWeek PolicyType:
                 Chi-square test   results
0  Pearson Chi-square ( 48.0) =   127.6210
1                    p-value =     0.0000
2                 Cramer's V =     0.0371


DayOfWeek VehicleCategory:
                 Chi-square test  results
```

```
0  Pearson Chi-square ( 12.0) =   96.8021
1                    p-value =    0.0000
2                 Cramer's V =    0.0560


DayOfWeek VehiclePrice:
                 Chi-square test  results
0  Pearson Chi-square ( 30.0) =   51.9935
1                    p-value =    0.0076
2                 Cramer's V =    0.0260


DayOfWeek Deductible:
                 Chi-square test  results
0  Pearson Chi-square ( 18.0) =   32.2414
1                    p-value =    0.0206
2                 Cramer's V =    0.0264


DayOfWeek PastNumberOfClaims:
                 Chi-square test  results
0  Pearson Chi-square ( 18.0) =   29.6607
1                    p-value =    0.0409
2                 Cramer's V =    0.0253


DayOfWeek AgeOfPolicyHolder:
                 Chi-square test  results
0  Pearson Chi-square ( 48.0) =   92.4484
1                    p-value =    0.0001
2                 Cramer's V =    0.0316


DayOfWeek BasePolicy:
                 Chi-square test  results
0  Pearson Chi-square ( 12.0) =   69.4465
1                    p-value =    0.0000
2                 Cramer's V =    0.0475


****** new ******
Make AccidentArea:
                 Chi-square test  results
0  Pearson Chi-square ( 18.0) =   50.7382
1                    p-value =    0.0001
2                 Cramer's V =    0.0574
```

```
Make Sex:
                 Chi-square test   results
0  Pearson Chi-square ( 18.0) =   100.7645
1                    p-value =     0.0000
2                 Cramer's V =     0.0808


Make MaritalStatus:
                 Chi-square test   results
0  Pearson Chi-square ( 54.0) =   265.6970
1                    p-value =     0.0000
2                 Cramer's V =     0.0758


Make Fault:
                 Chi-square test  results
0  Pearson Chi-square ( 18.0) =   51.6409
1                    p-value =    0.0000
2                 Cramer's V =    0.0579


Make PolicyType:
                  Chi-square test    results
0  Pearson Chi-square ( 144.0) =   3508.2304
1                     p-value =      0.0000
2                  Cramer's V =      0.1686


Make VehicleCategory:
                 Chi-square test    results
0  Pearson Chi-square ( 36.0) =   1174.8102
1                    p-value =      0.0000
2                 Cramer's V =      0.1952


Make VehiclePrice:
                 Chi-square test    results
0  Pearson Chi-square ( 90.0) =   5307.6281
1                    p-value =      0.0000
2                 Cramer's V =      0.2624


Make PastNumberOfClaims:
                 Chi-square test   results
0  Pearson Chi-square ( 54.0) =   106.7208
1                    p-value =     0.0000
2                 Cramer's V =     0.0480
```

```
Make AgeOfVehicle:
                  Chi-square test    results
0  Pearson Chi-square ( 126.0) =   2147.2538
1                    p-value =      0.0000
2                 Cramer's V =      0.1410
```

```
Make AgeOfPolicyHolder:
                  Chi-square test    results
0  Pearson Chi-square ( 144.0) =   1924.9858
1                    p-value =      0.0000
2                 Cramer's V =      0.1249
```

```
Make AgentType:
                 Chi-square test  results
0  Pearson Chi-square ( 18.0) =   35.1848
1                   p-value =    0.0090
2                Cramer's V =    0.0478
```

```
Make NumberOfSuppliments:
                 Chi-square test   results
0  Pearson Chi-square ( 54.0) =   173.2723
1                   p-value =     0.0000
2                Cramer's V =     0.0612
```

```
Make NumberOfCars:
                 Chi-square test   results
0  Pearson Chi-square ( 72.0) =   117.1254
1                   p-value =     0.0006
2                Cramer's V =     0.0436
```

```
Make BasePolicy:
                 Chi-square test   results
0  Pearson Chi-square ( 36.0) =   438.5126
1                   p-value =     0.0000
2                Cramer's V =     0.1192
```

```
****** new ******
AccidentArea DayOfWeekClaimed:
                Chi-square test  results
0  Pearson Chi-square ( 7.0) =   17.2348
1                  p-value =    0.0159
```

```
2                 Cramer's V =    0.0334
```

```
AccidentArea MonthClaimed:
                 Chi-square test  results
0  Pearson Chi-square ( 12.0) =   24.7296
1                    p-value =    0.0162
2                 Cramer's V =    0.0400
```

```
AccidentArea Sex:
                Chi-square test  results
0  Pearson Chi-square ( 1.0) =   17.6210
1                   p-value =    0.0000
2               Cramer's phi =    0.0338
```

```
AccidentArea PolicyType:
                Chi-square test  results
0  Pearson Chi-square ( 8.0) =   78.5019
1                   p-value =    0.0000
2                 Cramer's V =    0.0714
```

```
AccidentArea VehicleCategory:
                Chi-square test  results
0  Pearson Chi-square ( 2.0) =   65.1495
1                   p-value =    0.0000
2                 Cramer's V =    0.0650
```

```
AccidentArea Days_Policy_Claim:
                Chi-square test  results
0  Pearson Chi-square ( 3.0) =    9.3663
1                   p-value =    0.0248
2                 Cramer's V =    0.0246
```

```
AccidentArea PastNumberOfClaims:
                Chi-square test  results
0  Pearson Chi-square ( 3.0) =   60.8933
1                   p-value =    0.0000
2                 Cramer's V =    0.0628
```

```
AccidentArea WitnessPresent:
                Chi-square test  results
0  Pearson Chi-square ( 1.0) =   12.4043
```

```
1                 p-value =    0.0004
2            Cramer's phi =    0.0284


AccidentArea AddressChange_Claim:
               Chi-square test  results
0  Pearson Chi-square ( 4.0) =   13.8445
1                 p-value =    0.0078
2              Cramer's V =    0.0300


AccidentArea BasePolicy:
               Chi-square test  results
0  Pearson Chi-square ( 2.0) =   50.1098
1                 p-value =    0.0000
2              Cramer's V =    0.0570


****** new ******
DayOfWeekClaimed MonthClaimed:
                Chi-square test     results
0  Pearson Chi-square ( 84.0) =   15649.1595
1                  p-value =        0.0000
2               Cramer's V =        0.3808


DayOfWeekClaimed WeekOfMonthClaimed:
                Chi-square test   results
0  Pearson Chi-square ( 28.0) =   102.6120
1                  p-value =      0.0000
2               Cramer's V =      0.0408


DayOfWeekClaimed PolicyType:
                Chi-square test   results
0  Pearson Chi-square ( 56.0) =   140.5674
1                  p-value =      0.0000
2               Cramer's V =      0.0361


DayOfWeekClaimed VehicleCategory:
                Chi-square test  results
0  Pearson Chi-square ( 14.0) =   26.3675
1                  p-value =    0.0232
2               Cramer's V =    0.0292


DayOfWeekClaimed RepNumber:
```

```
                 Chi-square test   results
0  Pearson Chi-square ( 105.0) =   132.0635
1                     p-value =     0.0381
2                  Cramer's V =     0.0350


DayOfWeekClaimed Days_Policy_Claim:
                 Chi-square test     results
0  Pearson Chi-square ( 21.0) =   15438.0791
1                    p-value =       0.0000
2                 Cramer's V =       0.5777


DayOfWeekClaimed AgeOfVehicle:
                 Chi-square test  results
0  Pearson Chi-square ( 49.0) =   90.7528
1                    p-value =    0.0003
2                 Cramer's V =    0.0290


DayOfWeekClaimed AgeOfPolicyHolder:
                 Chi-square test   results
0  Pearson Chi-square ( 56.0) =   113.5637
1                    p-value =     0.0000
2                 Cramer's V =     0.0324


DayOfWeekClaimed AgentType:
                Chi-square test  results
0  Pearson Chi-square ( 7.0) =   14.5105
1                   p-value =    0.0428
2                Cramer's V =    0.0307


****** new ******
MonthClaimed WeekOfMonthClaimed:
                 Chi-square test   results
0  Pearson Chi-square ( 48.0) =   288.3509
1                    p-value =     0.0000
2                 Cramer's V =     0.0684


MonthClaimed PolicyType:
                 Chi-square test   results
0  Pearson Chi-square ( 96.0) =   133.4310
1                    p-value =     0.0069
2                 Cramer's V =     0.0329
```

```
MonthClaimed VehiclePrice:
                 Chi-square test   results
0  Pearson Chi-square ( 60.0) =   107.1590
1                    p-value =     0.0002
2                 Cramer's V =     0.0373


MonthClaimed Days_Policy_Claim:
                 Chi-square test     results
0  Pearson Chi-square ( 36.0) =   15435.2169
1                    p-value =       0.0000
2                 Cramer's V =       0.5776


MonthClaimed PastNumberOfClaims:
                 Chi-square test  results
0  Pearson Chi-square ( 36.0) =   57.4361
1                    p-value =    0.0131
2                 Cramer's V =    0.0352


MonthClaimed AgeOfVehicle:
                 Chi-square test   results
0  Pearson Chi-square ( 84.0) =   354.1054
1                    p-value =     0.0000
2                 Cramer's V =     0.0573


MonthClaimed AgeOfPolicyHolder:
                 Chi-square test   results
0  Pearson Chi-square ( 96.0) =   340.5870
1                    p-value =     0.0000
2                 Cramer's V =     0.0525


MonthClaimed PoliceReportFiled:
                 Chi-square test  results
0  Pearson Chi-square ( 12.0) =   64.2098
1                    p-value =    0.0000
2                 Cramer's V =    0.0645


MonthClaimed AgentType:
                 Chi-square test  results
0  Pearson Chi-square ( 12.0) =   22.7712
1                    p-value =    0.0297
2                 Cramer's V =    0.0384
```

```
MonthClaimed NumberOfCars:
                 Chi-square test   results
0  Pearson Chi-square ( 48.0) =   151.7854
1                    p-value =     0.0000
2                 Cramer's V =     0.0496


MonthClaimed Year:
                 Chi-square test   results
0  Pearson Chi-square ( 24.0) =   113.5557
1                    p-value =     0.0000
2                 Cramer's V =     0.0607


MonthClaimed BasePolicy:
                 Chi-square test  results
0  Pearson Chi-square ( 24.0) =   57.1723
1                    p-value =    0.0002
2                 Cramer's V =    0.0431


****** new ******
WeekOfMonthClaimed VehicleCategory:
                Chi-square test  results
0  Pearson Chi-square ( 8.0) =   16.5586
1                   p-value =    0.0350
2                Cramer's V =    0.0232


WeekOfMonthClaimed VehiclePrice:
                 Chi-square test  results
0  Pearson Chi-square ( 20.0) =   35.3163
1                    p-value =    0.0185
2                 Cramer's V =    0.0239


WeekOfMonthClaimed PastNumberOfClaims:
                 Chi-square test  results
0  Pearson Chi-square ( 12.0) =   22.6916
1                    p-value =    0.0305
2                 Cramer's V =    0.0221


WeekOfMonthClaimed PoliceReportFiled:
                Chi-square test  results
0  Pearson Chi-square ( 4.0) =   12.3583
```

```
1                   p-value =    0.0149
2                Cramer's V =    0.0283


****** new ******
Sex MaritalStatus:
                Chi-square test   results
0  Pearson Chi-square ( 3.0) =   377.0490
1                   p-value =     0.0000
2                Cramer's V =     0.1564


Sex PolicyType:
                Chi-square test   results
0  Pearson Chi-square ( 8.0) =   138.8970
1                   p-value =     0.0000
2                Cramer's V =     0.0949


Sex VehicleCategory:
                Chi-square test   results
0  Pearson Chi-square ( 2.0) =   106.8475
1                   p-value =     0.0000
2                Cramer's V =     0.0832


Sex VehiclePrice:
                Chi-square test   results
0  Pearson Chi-square ( 5.0) =   336.3477
1                   p-value =     0.0000
2                Cramer's V =     0.1477


Sex AgeOfVehicle:
                Chi-square test   results
0  Pearson Chi-square ( 7.0) =   700.5206
1                   p-value =     0.0000
2                Cramer's V =     0.2131


Sex AgeOfPolicyHolder:
                Chi-square test   results
0  Pearson Chi-square ( 8.0) =   289.7156
1                   p-value =     0.0000
2                Cramer's V =     0.1371


Sex BasePolicy:
```

```
                Chi-square test  results
0  Pearson Chi-square ( 2.0) =   75.2780
1                   p-value =    0.0000
2                Cramer's V =    0.0699


****** new ******
MaritalStatus PolicyType:
                 Chi-square test   results
0  Pearson Chi-square ( 24.0) =   113.2053
1                    p-value =     0.0000
2                 Cramer's V =     0.0495


MaritalStatus VehicleCategory:
                Chi-square test  results
0  Pearson Chi-square ( 6.0) =   67.2819
1                   p-value =    0.0000
2                Cramer's V =    0.0467


MaritalStatus VehiclePrice:
                 Chi-square test   results
0  Pearson Chi-square ( 15.0) =   245.7704
1                    p-value =     0.0000
2                 Cramer's V =     0.0729


MaritalStatus Deductible:
                Chi-square test  results
0  Pearson Chi-square ( 9.0) =   25.6227
1                   p-value =    0.0024
2                Cramer's V =    0.0235


MaritalStatus Days_Policy_Accident:
                 Chi-square test  results
0  Pearson Chi-square ( 12.0) =   30.1381
1                    p-value =    0.0027
2                 Cramer's V =    0.0255


MaritalStatus PastNumberOfClaims:
                Chi-square test  results
0  Pearson Chi-square ( 9.0) =   17.7580
1                   p-value =    0.0381
2                Cramer's V =    0.0196
```

```
MaritalStatus AgeOfVehicle:
                 Chi-square test    results
0  Pearson Chi-square ( 21.0) =   3288.8123
1                    p-value =      0.0000
2                 Cramer's V =      0.2666


MaritalStatus AgeOfPolicyHolder:
                 Chi-square test    results
0  Pearson Chi-square ( 24.0) =   4299.7930
1                    p-value =      0.0000
2                 Cramer's V =      0.3049


MaritalStatus NumberOfSuppliments:
                Chi-square test  results
0  Pearson Chi-square ( 9.0) =   29.1260
1                   p-value =    0.0006
2                Cramer's V =    0.0251


MaritalStatus BasePolicy:
                Chi-square test  results
0  Pearson Chi-square ( 6.0) =   47.3191
1                   p-value =    0.0000
2                Cramer's V =    0.0392


****** new ******
Fault PolicyType:
                Chi-square test  results
0  Pearson Chi-square ( 8.0) =   895.858
1                   p-value =     0.000
2                Cramer's V =     0.241


Fault VehicleCategory:
                Chi-square test   results
0  Pearson Chi-square ( 2.0) =   544.8769
1                   p-value =     0.0000
2                Cramer's V =     0.1880


Fault VehiclePrice:
                Chi-square test  results
0  Pearson Chi-square ( 5.0) =   37.5873
1                   p-value =    0.0000
```

```
2                 Cramer's V =    0.0494


Fault Days_Policy_Accident:
                Chi-square test  results
0  Pearson Chi-square ( 4.0) =   17.8731
1                    p-value =    0.0013
2                 Cramer's V =    0.0340


Fault Days_Policy_Claim:
                Chi-square test  results
0  Pearson Chi-square ( 3.0) =    7.8594
1                    p-value =    0.0490
2                 Cramer's V =    0.0226


Fault PastNumberOfClaims:
                Chi-square test   results
0  Pearson Chi-square ( 3.0) =   250.4110
1                    p-value =     0.0000
2                 Cramer's V =     0.1274


Fault AgeOfVehicle:
                Chi-square test  results
0  Pearson Chi-square ( 7.0) =   36.1858
1                    p-value =    0.0000
2                 Cramer's V =    0.0484


Fault AgeOfPolicyHolder:
                Chi-square test  results
0  Pearson Chi-square ( 8.0) =   55.2858
1                    p-value =    0.0000
2                 Cramer's V =    0.0599


Fault PoliceReportFiled:
                Chi-square test  results
0  Pearson Chi-square ( 1.0) =   11.4467
1                    p-value =    0.0007
2               Cramer's phi =    0.0272


Fault WitnessPresent:
                Chi-square test  results
0  Pearson Chi-square ( 1.0) =   57.4464
```

```
1                  p-value =    0.0000
2              Cramer's phi =    0.0610


Fault NumberOfSuppliments:
               Chi-square test  results
0  Pearson Chi-square ( 3.0) =   12.9618
1                  p-value =    0.0047
2                Cramer's V =    0.0290


Fault BasePolicy:
               Chi-square test   results
0  Pearson Chi-square ( 2.0) =   659.5151
1                  p-value =     0.0000
2                Cramer's V =     0.2068


****** new ******
PolicyType VehicleCategory:
                Chi-square test  results
0  Pearson Chi-square ( 16.0) =   30840.0
1                   p-value =       0.0
2                 Cramer's V =       1.0


PolicyType VehiclePrice:
                Chi-square test    results
0  Pearson Chi-square ( 40.0) =   5558.1464
1                   p-value =      0.0000
2                 Cramer's V =      0.2685


PolicyType Deductible:
                Chi-square test    results
0  Pearson Chi-square ( 24.0) =   1943.0273
1                   p-value =      0.0000
2                 Cramer's V =      0.2049


PolicyType Days_Policy_Accident:
                Chi-square test  results
0  Pearson Chi-square ( 32.0) =   48.6201
1                   p-value =    0.0301
2                 Cramer's V =    0.0281


PolicyType PastNumberOfClaims:
```

```
                 Chi-square test    results
0  Pearson Chi-square ( 24.0) =   2506.1265
1                    p-value =      0.0000
2                 Cramer's V =      0.2328


PolicyType AgeOfVehicle:
                 Chi-square test   results
0  Pearson Chi-square ( 56.0) =   810.0619
1                    p-value =     0.0000
2                 Cramer's V =     0.0866


PolicyType AgeOfPolicyHolder:
                 Chi-square test    results
0  Pearson Chi-square ( 64.0) =   1459.5007
1                    p-value =      0.0000
2                 Cramer's V =      0.1088


PolicyType PoliceReportFiled:
                Chi-square test  results
0  Pearson Chi-square ( 8.0) =   34.2248
1                   p-value =    0.0000
2                Cramer's V =    0.0471


PolicyType WitnessPresent:
                Chi-square test  results
0  Pearson Chi-square ( 8.0) =   40.1169
1                   p-value =    0.0000
2                Cramer's V =    0.0510


PolicyType AgentType:
                Chi-square test   results
0  Pearson Chi-square ( 8.0) =   153.7383
1                   p-value =     0.0000
2                Cramer's V =     0.0999


PolicyType NumberOfSuppliments:
                 Chi-square test   results
0  Pearson Chi-square ( 24.0) =   136.2349
1                    p-value =     0.0000
2                 Cramer's V =     0.0543
```

```
PolicyType AddressChange_Claim:
                 Chi-square test   results
0  Pearson Chi-square ( 32.0) =   107.6380
1                    p-value =     0.0000
2                 Cramer's V =     0.0418


PolicyType NumberOfCars:
                 Chi-square test  results
0  Pearson Chi-square ( 32.0) =   55.0367
1                    p-value =    0.0069
2                 Cramer's V =    0.0299


PolicyType Year:
                 Chi-square test  results
0  Pearson Chi-square ( 16.0) =   32.2181
1                    p-value =    0.0094
2                 Cramer's V =    0.0323


PolicyType BasePolicy:
                 Chi-square test  results
0  Pearson Chi-square ( 16.0) =   30840.0
1                    p-value =       0.0
2                 Cramer's V =       1.0


****** new ******
VehicleCategory VehiclePrice:
                 Chi-square test    results
0  Pearson Chi-square ( 10.0) =   2530.6226
1                    p-value =      0.0000
2                 Cramer's V =      0.2865


VehicleCategory Days_Policy_Accident:
                Chi-square test  results
0  Pearson Chi-square ( 8.0) =   18.8717
1                   p-value =    0.0156
2                Cramer's V =    0.0247


VehicleCategory PastNumberOfClaims:
                Chi-square test    results
0  Pearson Chi-square ( 6.0) =   1744.3221
1                   p-value =      0.0000
2                Cramer's V =      0.2378
```

```
VehicleCategory AgeOfVehicle:
                 Chi-square test   results
0  Pearson Chi-square ( 14.0) =   153.6338
1                    p-value =     0.0000
2                 Cramer's V =     0.0706


VehicleCategory AgeOfPolicyHolder:
                 Chi-square test   results
0  Pearson Chi-square ( 16.0) =   193.5867
1                    p-value =     0.0000
2                 Cramer's V =     0.0792


VehicleCategory PoliceReportFiled:
                Chi-square test  results
0  Pearson Chi-square ( 2.0) =   25.2812
1                   p-value =    0.0000
2                Cramer's V =    0.0405


VehicleCategory WitnessPresent:
                Chi-square test  results
0  Pearson Chi-square ( 2.0) =   11.8448
1                   p-value =    0.0027
2                Cramer's V =    0.0277


VehicleCategory AgentType:
                Chi-square test  results
0  Pearson Chi-square ( 2.0) =   30.1847
1                   p-value =    0.0000
2                Cramer's V =    0.0442


VehicleCategory NumberOfSuppliments:
                Chi-square test  results
0  Pearson Chi-square ( 6.0) =   28.4876
1                   p-value =    0.0001
2                Cramer's V =    0.0304


VehicleCategory BasePolicy:
                Chi-square test     results
0  Pearson Chi-square ( 4.0) =   14293.1690
1                   p-value =       0.0000
```

```
2                 Cramer's V =       0.6808


****** new ******
VehiclePrice Days_Policy_Accident:
                 Chi-square test  results
0  Pearson Chi-square ( 20.0) =   34.7076
1                    p-value =    0.0217
2                 Cramer's V =    0.0237


VehiclePrice PastNumberOfClaims:
                 Chi-square test   results
0  Pearson Chi-square ( 15.0) =   392.2822
1                    p-value =     0.0000
2                 Cramer's V =     0.0921


VehiclePrice AgeOfVehicle:
                 Chi-square test    results
0  Pearson Chi-square ( 35.0) =   2886.6801
1                    p-value =      0.0000
2                 Cramer's V =      0.1935


VehiclePrice AgeOfPolicyHolder:
                 Chi-square test    results
0  Pearson Chi-square ( 40.0) =   2499.9984
1                    p-value =      0.0000
2                 Cramer's V =      0.1801


VehiclePrice AgentType:
                Chi-square test   results
0  Pearson Chi-square ( 5.0) =   148.2960
1                   p-value =     0.0000
2                Cramer's V =     0.0981


VehiclePrice NumberOfSuppliments:
                 Chi-square test   results
0  Pearson Chi-square ( 15.0) =   153.5315
1                    p-value =     0.0000
2                 Cramer's V =     0.0576


VehiclePrice Year:
                 Chi-square test  results
```

```
0  Pearson Chi-square ( 10.0) =   37.7982
1                    p-value =    0.0000
2                 Cramer's V =    0.0350


VehiclePrice BasePolicy:
                 Chi-square test    results
0  Pearson Chi-square ( 10.0) =   1294.7487
1                    p-value =      0.0000
2                 Cramer's V =      0.2049


****** new ******
****** new ******
RepNumber AgeOfVehicle:
                  Chi-square test   results
0  Pearson Chi-square ( 105.0) =   144.5109
1                     p-value =     0.0064
2                  Cramer's V =     0.0366


RepNumber AgeOfPolicyHolder:
                  Chi-square test   results
0  Pearson Chi-square ( 120.0) =   148.6219
1                     p-value =     0.0392
2                  Cramer's V =     0.0347


****** new ******
Deductible AgeOfVehicle:
                 Chi-square test   results
0  Pearson Chi-square ( 21.0) =   296.7951
1                    p-value =     0.0000
2                 Cramer's V =     0.0801


Deductible AgeOfPolicyHolder:
                 Chi-square test   results
0  Pearson Chi-square ( 24.0) =   147.9834
1                    p-value =     0.0000
2                 Cramer's V =     0.0566


Deductible AddressChange_Claim:
                 Chi-square test     results
0  Pearson Chi-square ( 12.0) =   13202.0328
1                    p-value =       0.0000
2                 Cramer's V =       0.5342
```

```
Deductible NumberOfCars:
                Chi-square test  results
0  Pearson Chi-square ( 12.0) =   62.1425
1                    p-value =    0.0000
2                 Cramer's V =    0.0367


****** new ******
****** new ******
Days_Policy_Accident Days_Policy_Claim:
                Chi-square test    results
0  Pearson Chi-square ( 12.0) =   7675.3042
1                    p-value =      0.0000
2                 Cramer's V =      0.4073


Days_Policy_Accident PastNumberOfClaims:
                Chi-square test  results
0  Pearson Chi-square ( 12.0) =   52.2123
1                    p-value =    0.0000
2                 Cramer's V =    0.0336


Days_Policy_Accident AgeOfVehicle:
                Chi-square test   results
0  Pearson Chi-square ( 28.0) =   102.2116
1                    p-value =     0.0000
2                 Cramer's V =     0.0407


Days_Policy_Accident AgeOfPolicyHolder:
                Chi-square test  results
0  Pearson Chi-square ( 32.0) =   52.1698
1                    p-value =    0.0136
2                 Cramer's V =    0.0291


Days_Policy_Accident WitnessPresent:
               Chi-square test  results
0  Pearson Chi-square ( 4.0) =   49.1640
1                   p-value =    0.0000
2                Cramer's V =    0.0565


Days_Policy_Accident NumberOfSuppliments:
                Chi-square test   results
```

```
0  Pearson Chi-square ( 12.0) =   148.1611
1                    p-value =     0.0000
2                 Cramer's V =     0.0566


Days_Policy_Accident NumberOfCars:
                 Chi-square test   results
0  Pearson Chi-square ( 16.0) =   165.0298
1                    p-value =     0.0000
2                 Cramer's V =     0.0517


Days_Policy_Accident BasePolicy:
                Chi-square test  results
0  Pearson Chi-square ( 8.0) =   26.0831
1                   p-value =    0.0010
2                Cramer's V =    0.0291


****** new ******
Days_Policy_Claim PastNumberOfClaims:
                Chi-square test  results
0  Pearson Chi-square ( 9.0) =   38.2365
1                   p-value =    0.0000
2                Cramer's V =    0.0287


Days_Policy_Claim AgeOfVehicle:
                 Chi-square test  results
0  Pearson Chi-square ( 21.0) =   74.7111
1                    p-value =    0.0000
2                 Cramer's V =    0.0402


Days_Policy_Claim AgeOfPolicyHolder:
                 Chi-square test  results
0  Pearson Chi-square ( 24.0) =   75.9610
1                    p-value =    0.0000
2                 Cramer's V =    0.0405


Days_Policy_Claim NumberOfSuppliments:
                Chi-square test  results
0  Pearson Chi-square ( 9.0) =   70.1265
1                   p-value =    0.0000
2                Cramer's V =    0.0389
```

```
Days_Policy_Claim BasePolicy:
                Chi-square test  results
0  Pearson Chi-square ( 6.0) =   14.1933
1                    p-value =    0.0275
2                 Cramer's V =    0.0215


****** new ******
PastNumberOfClaims AgeOfVehicle:
                 Chi-square test  results
0  Pearson Chi-square ( 21.0) =   83.7227
1                     p-value =    0.0000
2                  Cramer's V =    0.0425


PastNumberOfClaims AgeOfPolicyHolder:
                 Chi-square test  results
0  Pearson Chi-square ( 24.0) =   69.7237
1                     p-value =    0.0000
2                  Cramer's V =    0.0388


PastNumberOfClaims AgentType:
                Chi-square test  results
0  Pearson Chi-square ( 3.0) =   11.4980
1                    p-value =    0.0093
2                 Cramer's V =    0.0273


PastNumberOfClaims NumberOfSuppliments:
                Chi-square test   results
0  Pearson Chi-square ( 9.0) =   210.2259
1                    p-value =     0.0000
2                 Cramer's V =     0.0674


PastNumberOfClaims BasePolicy:
                Chi-square test    results
0  Pearson Chi-square ( 6.0) =   2197.7416
1                    p-value =      0.0000
2                 Cramer's V =      0.2670


****** new ******
AgeOfVehicle AgeOfPolicyHolder:
                 Chi-square test     results
0  Pearson Chi-square ( 56.0) =   30849.6640
1                     p-value =       0.0000
```

```
2                  Cramer's V =       0.5346


AgeOfVehicle WitnessPresent:
                Chi-square test  results
0  Pearson Chi-square ( 7.0) =   18.8519
1                    p-value =    0.0087
2                 Cramer's V =    0.0350


AgeOfVehicle NumberOfSuppliments:
                 Chi-square test   results
0  Pearson Chi-square ( 21.0) =   631.7692
1                     p-value =     0.0000
2                  Cramer's V =     0.1169


AgeOfVehicle BasePolicy:
                 Chi-square test   results
0  Pearson Chi-square ( 14.0) =   295.9225
1                     p-value =     0.0000
2                  Cramer's V =     0.0980


****** new ******
AgeOfPolicyHolder NumberOfSuppliments:
                 Chi-square test   results
0  Pearson Chi-square ( 24.0) =   440.4423
1                     p-value =     0.0000
2                  Cramer's V =     0.0976


AgeOfPolicyHolder Year:
                 Chi-square test  results
0  Pearson Chi-square ( 16.0) =   32.3318
1                     p-value =    0.0091
2                  Cramer's V =    0.0324


AgeOfPolicyHolder BasePolicy:
                 Chi-square test   results
0  Pearson Chi-square ( 16.0) =   404.3574
1                     p-value =     0.0000
2                  Cramer's V =     0.1145


****** new ******
PoliceReportFiled WitnessPresent:
```

```
                Chi-square test   results
0  Pearson Chi-square ( 1.0) =   605.1143
1                    p-value =     0.0000
2               Cramer's phi =     0.1981


PoliceReportFiled AgentType:
                Chi-square test  results
0  Pearson Chi-square ( 1.0) =    8.3486
1                    p-value =    0.0039
2               Cramer's phi =    0.0233


PoliceReportFiled NumberOfSuppliments:
                Chi-square test  results
0  Pearson Chi-square ( 3.0) =    8.4281
1                    p-value =    0.0379
2                 Cramer's V =    0.0234


PoliceReportFiled NumberOfCars:
                Chi-square test  results
0  Pearson Chi-square ( 4.0) =   11.6517
1                    p-value =    0.0201
2                 Cramer's V =    0.0275


PoliceReportFiled Year:
                Chi-square test  results
0  Pearson Chi-square ( 2.0) =    8.2065
1                    p-value =    0.0165
2                 Cramer's V =    0.0231


PoliceReportFiled BasePolicy:
                Chi-square test  results
0  Pearson Chi-square ( 2.0) =   29.3742
1                    p-value =    0.0000
2                 Cramer's V =    0.0436


****** new ******
WitnessPresent Year:
                Chi-square test  results
0  Pearson Chi-square ( 2.0) =    6.0565
1                    p-value =    0.0484
2                 Cramer's V =    0.0198
```

```
WitnessPresent BasePolicy:
                Chi-square test  results
0  Pearson Chi-square ( 2.0) =   23.8709
1                   p-value =    0.0000
2                Cramer's V =    0.0393


****** new ******
AgentType NumberOfSuppliments:
                Chi-square test  results
0  Pearson Chi-square ( 3.0) =   17.6963
1                   p-value =    0.0005
2                Cramer's V =    0.0339


AgentType AddressChange_Claim:
                Chi-square test  results
0  Pearson Chi-square ( 4.0) =   11.2244
1                   p-value =    0.0242
2                Cramer's V =    0.0270


AgentType NumberOfCars:
                Chi-square test  results
0  Pearson Chi-square ( 4.0) =   14.6981
1                   p-value =    0.0054
2                Cramer's V =    0.0309


AgentType Year:
                Chi-square test  results
0  Pearson Chi-square ( 2.0) =    6.0853
1                   p-value =    0.0477
2                Cramer's V =    0.0199


AgentType BasePolicy:
                Chi-square test   results
0  Pearson Chi-square ( 2.0) =   106.9915
1                   p-value =     0.0000
2                Cramer's V =     0.0833


****** new ******
NumberOfSuppliments BasePolicy:
                Chi-square test  results
0  Pearson Chi-square ( 6.0) =   43.5975
```

```
1                     p-value =    0.0000
2                  Cramer's V =    0.0376


****** new ******
AddressChange_Claim NumberOfCars:
                 Chi-square test     results
0  Pearson Chi-square ( 16.0) =   13501.6367
1                     p-value =       0.0000
2                  Cramer's V =       0.4679


****** new ******
****** new ******
Year BasePolicy:
                Chi-square test  results
0  Pearson Chi-square ( 4.0) =   10.9377
1                    p-value =    0.0273
2                 Cramer's V =    0.0188


****** new ******
****** new ******
```

[ ]:
```
for i in dataset.columns[~dataset.columns.isin(['Age', 'FraudFound_P'])]:
    crosstab, test_results, expected = rp.crosstab(dataset[i],␣
↪dataset['FraudFound_P'],
                                              test= "chi-square",
                                              expected_freqs= True,
                                              prop= "cell")

    if test_results['results'][1] < 0.05:
        print(i + ':')
        print(test_results)
        print('\n')
```

```
Month:
                 Chi-square test  results
0  Pearson Chi-square ( 11.0) =   29.7964
1                     p-value =    0.0017
2                  Cramer's V =    0.0440


Make:
                 Chi-square test  results
0  Pearson Chi-square ( 18.0) =   59.8100
1                     p-value =    0.0000
```

```
2                  Cramer's V =    0.0623
```

```
AccidentArea:
                Chi-square test  results
0  Pearson Chi-square ( 1.0) =   17.3045
1                   p-value =    0.0000
2                Cramer's phi =    0.0335
```

```
MonthClaimed:
                 Chi-square test  results
0  Pearson Chi-square ( 12.0) =   42.2667
1                    p-value =    0.0000
2                 Cramer's V =    0.0524
```

```
Sex:
                Chi-square test  results
0  Pearson Chi-square ( 1.0) =   13.8348
1                   p-value =    0.0002
2                Cramer's phi =    0.0300
```

```
Fault:
                Chi-square test   results
0  Pearson Chi-square ( 1.0) =   266.1974
1                   p-value =     0.0000
2                Cramer's phi =     0.1314
```

```
PolicyType:
                Chi-square test   results
0  Pearson Chi-square ( 8.0) =   437.4019
1                   p-value =     0.0000
2                 Cramer's V =     0.1684
```

```
VehicleCategory:
                Chi-square test   results
0  Pearson Chi-square ( 2.0) =   290.9421
1                   p-value =     0.0000
2                 Cramer's V =     0.1374
```

```
VehiclePrice:
                Chi-square test  results
0  Pearson Chi-square ( 5.0) =   67.7683
```

```
1                   p-value =    0.0000
2                Cramer's V =    0.0663
```

```
Deductible:
                Chi-square test  results
0  Pearson Chi-square ( 3.0) =   72.4152
1                   p-value =    0.0000
2                Cramer's V =    0.0685
```

```
Days_Policy_Accident:
                Chi-square test  results
0  Pearson Chi-square ( 4.0) =   11.5716
1                   p-value =    0.0208
2                Cramer's V =    0.0274
```

```
PastNumberOfClaims:
                Chi-square test  results
0  Pearson Chi-square ( 3.0) =   53.5008
1                   p-value =    0.0000
2                Cramer's V =    0.0589
```

```
AgeOfVehicle:
                Chi-square test  results
0  Pearson Chi-square ( 7.0) =   21.9290
1                   p-value =    0.0026
2                Cramer's V =    0.0377
```

```
AgeOfPolicyHolder:
                Chi-square test  results
0  Pearson Chi-square ( 8.0) =   33.0033
1                   p-value =    0.0001
2                Cramer's V =    0.0463
```

```
PoliceReportFiled:
                Chi-square test  results
0  Pearson Chi-square ( 1.0) =    3.9512
1                   p-value =    0.0468
2              Cramer's phi =    0.0160
```

```
AgentType:
                Chi-square test  results
```

```
0  Pearson Chi-square ( 1.0) =    8.1417
1                    p-value =    0.0043
2               Cramer's phi =    0.0230


NumberOfSuppliments:
                Chi-square test  results
0  Pearson Chi-square ( 3.0) =   18.1406
1                    p-value =    0.0004
2                 Cramer's V =    0.0343


AddressChange_Claim:
                Chi-square test   results
0  Pearson Chi-square ( 4.0) =   104.7338
1                    p-value =     0.0000
2                 Cramer's V =     0.0824


Year:
                Chi-square test  results
0  Pearson Chi-square ( 2.0) =    9.5780
1                    p-value =    0.0083
2                 Cramer's V =    0.0249


BasePolicy:
                Chi-square test   results
0  Pearson Chi-square ( 2.0) =   402.8519
1                    p-value =     0.0000
2                 Cramer's V =     0.1616
```

[ ]:
```python
for i in dataset.columns[~dataset.columns.isin(['Age', 'FraudFound_P'])]:
    crosstab, test_results, expected = rp.crosstab(dataset[i],␣
 ↪dataset['FraudFound_P'],
                                               test= "chi-square",
                                               expected_freqs= True,
                                               prop= "cell")

    if test_results['results'][1] > 0.05:
        print(i + ':')
        print(test_results)
        print('\n')
```

```
WeekOfMonth:
                Chi-square test  results
0  Pearson Chi-square ( 4.0) =    2.4474
```

```
1                    p-value =    0.6541
2                 Cramer's V =    0.0126


DayOfWeek:
                Chi-square test  results
0  Pearson Chi-square ( 6.0) =   10.1506
1                    p-value =    0.1185
2                 Cramer's V =    0.0257


DayOfWeekClaimed:
                Chi-square test  results
0  Pearson Chi-square ( 7.0) =    5.1596
1                    p-value =    0.6405
2                 Cramer's V =    0.0183


WeekOfMonthClaimed:
                Chi-square test  results
0  Pearson Chi-square ( 4.0) =    3.3723
1                    p-value =    0.4976
2                 Cramer's V =    0.0148


MaritalStatus:
                Chi-square test  results
0  Pearson Chi-square ( 3.0) =    1.0135
1                    p-value =    0.7980
2                 Cramer's V =    0.0081


PolicyNumber:
                    Chi-square test     results
0  Pearson Chi-square ( 15419.0) =   15420.0000
1                        p-value =       0.4962
2                     Cramer's V =       1.0000


RepNumber:
                 Chi-square test  results
0  Pearson Chi-square ( 15.0) =   11.8196
1                     p-value =    0.6926
2                  Cramer's V =    0.0277


DriverRating:
                Chi-square test  results
```

```
0  Pearson Chi-square ( 3.0) =    1.6494
1                    p-value =    0.6482
2               Cramer's V =    0.0103


Days_Policy_Claim:
                Chi-square test  results
0  Pearson Chi-square ( 3.0) =    4.8812
1                    p-value =    0.1807
2               Cramer's V =    0.0178


WitnessPresent:
                Chi-square test  results
0  Pearson Chi-square ( 1.0) =    1.0011
1                    p-value =    0.3171
2             Cramer's phi =    0.0081


NumberOfCars:
                Chi-square test  results
0  Pearson Chi-square ( 4.0) =    2.4161
1                    p-value =    0.6597
2               Cramer's V =    0.0125
```

common distrabution

[ ]:
```
pd.crosstab(dataset.Sex, dataset.FraudFound_P)
```

[ ]:
```
FraudFound_P      0    1
Sex
Female         2315  105
Male          12182  818
```

[ ]:
```
pd.crosstab(dataset.Sex, dataset.FraudFound_P, normalize = 'columns')
```

[ ]:
```
FraudFound_P         0         1
Sex
Female        0.159688  0.113759
Male          0.840312  0.886241
```

[ ]:
```
pd.crosstab(dataset.AccidentArea, dataset.FraudFound_P)
```

[ ]:
```
FraudFound_P      0    1
AccidentArea
Rural          1465  133
```

```
Urban          13032  790
```

[ ]:
```
pd.crosstab(dataset.AccidentArea, dataset.FraudFound_P, normalize = 'columns')
```

[ ]:
```
FraudFound_P         0         1
AccidentArea
Rural         0.101055  0.144095
Urban         0.898945  0.855905
```

[ ]:
```
pd.crosstab(dataset.VehicleCategory, dataset.FraudFound_P)
```

[ ]:
```
FraudFound_P        0    1
VehicleCategory
Sedan            8876  795
Sport            5274   84
Utility           347   44
```

[ ]:
```
pd.crosstab(dataset.VehicleCategory, dataset.FraudFound_P, normalize =␣
 ↪'columns')
```

[ ]:
```
FraudFound_P            0         1
VehicleCategory
Sedan            0.612265  0.861322
Sport            0.363799  0.091008
Utility          0.023936  0.047671
```

[ ]:
```
pd.crosstab(dataset.VehicleCategory, dataset.FraudFound_P, normalize =␣
 ↪'columns')
```

[ ]:
```
FraudFound_P            0         1
VehicleCategory
Sedan            0.612265  0.861322
Sport            0.363799  0.091008
Utility          0.023936  0.047671
```

[ ]:
```
for i in dataset.columns[~dataset.columns.isin(['Age', 'FraudFound_P'])]:
    print(pd.crosstab(dataset[i], dataset.FraudFound_P))
    print(pd.crosstab(dataset[i], dataset.FraudFound_P, normalize = 'columns'))
```

```
FraudFound_P     0    1
Month
Apr           1200   80
Aug           1043   84
Dec           1223   62
Feb           1184   82
Jan           1324   87
```

```
Jul            1197   60
Jun            1241   80
Mar            1258  102
May            1273   94
Nov            1155   46
Oct            1235   70
Sep            1164   76
FraudFound_P          0         1
Month
Apr            0.082776  0.086674
Aug            0.071946  0.091008
Dec            0.084362  0.067172
Feb            0.081672  0.088841
Jan            0.091329  0.094258
Jul            0.082569  0.065005
Jun            0.085604  0.086674
Mar            0.086777  0.110509
May            0.087811  0.101842
Nov            0.079672  0.049837
Oct            0.085190  0.075840
Sep            0.080292  0.082340
FraudFound_P      0     1
WeekOfMonth
1              2987  200
2              3333  225
3              3425  215
4              3206  192
5              1546   91
FraudFound_P          0         1
WeekOfMonth
1              0.206043  0.216685
2              0.229910  0.243770
3              0.236256  0.232936
4              0.221149  0.208017
5              0.106643  0.098592
FraudFound_P      0     1
DayOfWeek
Friday         2291  154
Monday         2456  160
Saturday       1850  132
Sunday         1623  122
Thursday       2053  120
Tuesday        2180  120
Wednesday      2044  115
FraudFound_P          0         1
DayOfWeek
Friday         0.158033  0.166847
Monday         0.169414  0.173348
```

```
Saturday      0.127613  0.143012
Sunday        0.111954  0.132178
Thursday      0.141616  0.130011
Tuesday       0.150376  0.130011
Wednesday     0.140995  0.124594
FraudFound_P     0    1
Make
Accura         413   59
BMW             14    1
Chevrolet     1587   94
Dodge          107    2
Ferrari          2    0
Ford           417   33
Honda         2622  179
Jaguar           6    0
Lexus            1    0
Mazda         2231  123
Mecedes          3    1
Mercury         77    6
Nisson          29    1
Pontiac       3624  213
Porche           5    0
Saab            97   11
Saturn          52    6
Toyota        2935  186
VW             275    8
FraudFound_P         0         1
Make
Accura        0.028489  0.063922
BMW           0.000966  0.001083
Chevrolet     0.109471  0.101842
Dodge         0.007381  0.002167
Ferrari       0.000138  0.000000
Ford          0.028765  0.035753
Honda         0.180865  0.193933
Jaguar        0.000414  0.000000
Lexus         0.000069  0.000000
Mazda         0.153894  0.133261
Mecedes       0.000207  0.001083
Mercury       0.005311  0.006501
Nisson        0.002000  0.001083
Pontiac       0.249983  0.230769
Porche        0.000345  0.000000
Saab          0.006691  0.011918
Saturn        0.003587  0.006501
Toyota        0.202456  0.201517
VW            0.018969  0.008667
FraudFound_P     0    1
```

```
AccidentArea
Rural          1465  133
Urban         13032  790
FraudFound_P         0         1
AccidentArea
Rural         0.101055  0.144095
Urban         0.898945  0.855905
FraudFound_P         0    1
DayOfWeekClaimed
0                    1    0
Friday            2333  164
Monday            3541  216
Saturday           117   10
Sunday              49    3
Thursday          2516  144
Tuesday           3177  198
Wednesday         2763  188
FraudFound_P             0         1
DayOfWeekClaimed
0                 0.000069  0.000000
Friday            0.160930  0.177681
Monday            0.244257  0.234020
Saturday          0.008071  0.010834
Sunday            0.003380  0.003250
Thursday          0.173553  0.156013
Tuesday           0.219149  0.214518
Wednesday         0.190591  0.203684
FraudFound_P     0    1
MonthClaimed
0                1    0
Apr           1189   82
Aug           1034   92
Dec           1097   49
Feb           1209   78
Jan           1354   92
Jul           1169   56
Jun           1215   78
Mar           1251   97
May           1309  102
Nov           1239   46
Oct           1266   73
Sep           1164   78
FraudFound_P         0         1
MonthClaimed
0             0.000069  0.000000
Apr           0.082017  0.088841
Aug           0.071325  0.099675
Dec           0.075671  0.053088
```

```
Feb           0.083397  0.084507
Jan           0.093399  0.099675
Jul           0.080637  0.060672
Jun           0.083810  0.084507
Mar           0.086294  0.105092
May           0.090295  0.110509
Nov           0.085466  0.049837
Oct           0.087328  0.079090
Sep           0.080292  0.084507
FraudFound_P          0    1
WeekOfMonthClaimed
1                  3230  220
2                  3512  208
3                  3362  221
4                  3224  209
5                  1169   65
FraudFound_P              0         1
WeekOfMonthClaimed
1                  0.222805  0.238353
2                  0.242257  0.225352
3                  0.231910  0.239437
4                  0.222391  0.226436
5                  0.080637  0.070423
FraudFound_P      0    1
Sex
Female         2315  105
Male          12182  818
FraudFound_P         0         1
Sex
Female        0.159688  0.113759
Male          0.840312  0.886241
FraudFound_P      0    1
MaritalStatus
Divorced         73    3
Married        9986  639
Single         4406  278
Widow            32    3
FraudFound_P          0         1
MaritalStatus
Divorced       0.005036  0.003250
Married        0.688832  0.692308
Single         0.303925  0.301192
Widow          0.002207  0.003250
FraudFound_P       0    1
Fault
Policy Holder  10344  886
Third Party     4153   37
FraudFound_P          0         1
```

```
Fault
Policy Holder  0.713527  0.959913
Third Party    0.286473  0.040087
FraudFound_P            0    1
PolicyType
Sedan - All Perils    3676  411
Sedan - Collision     5200  384
Sedan - Liability     4951   36
Sport - All Perils      22    0
Sport - Collision      300   48
Sport - Liability        1    0
Utility - All Perils   299   41
Utility - Collision     27    3
Utility - Liability     21    0
FraudFound_P                  0         1
PolicyType
Sedan - All Perils    0.253570  0.445287
Sedan - Collision     0.358695  0.416035
Sedan - Liability     0.341519  0.039003
Sport - All Perils    0.001518  0.000000
Sport - Collision     0.020694  0.052004
Sport - Liability     0.000069  0.000000
Utility - All Perils  0.020625  0.044420
Utility - Collision   0.001862  0.003250
Utility - Liability   0.001449  0.000000
FraudFound_P       0    1
VehicleCategory
Sedan            8876  795
Sport            5274   84
Utility           347   44
FraudFound_P            0         1
VehicleCategory
Sedan            0.612265  0.861322
Sport            0.363799  0.091008
Utility          0.023936  0.047671
FraudFound_P       0    1
VehiclePrice
20000 to 29000   7658  421
30000 to 39000   3358  175
40000 to 59000    430   31
60000 to 69000     83    4
less than 20000   993  103
more than 69000  1975  189
FraudFound_P            0         1
VehiclePrice
20000 to 29000   0.528247  0.456121
30000 to 39000   0.231634  0.189599
40000 to 59000   0.029661  0.033586
```

```
60000 to 69000   0.005725  0.004334
less than 20000  0.068497  0.111593
more than 69000  0.136235  0.204767
FraudFound_P  0  1
PolicyNumber
1             1  0
2             1  0
3             1  0
4             1  0
5             1  0
…           .. ..
15416         0  1
15417         1  0
15418         0  1
15419         1  0
15420         0  1

[15420 rows x 2 columns]
FraudFound_P         0         1
PolicyNumber
1             0.000069  0.000000
2             0.000069  0.000000
3             0.000069  0.000000
4             0.000069  0.000000
5             0.000069  0.000000
…                  …         …
15416         0.000000  0.001083
15417         0.000069  0.000000
15418         0.000000  0.001083
15419         0.000069  0.000000
15420         0.000000  0.001083

[15420 rows x 2 columns]
FraudFound_P    0   1
RepNumber
1             924  63
2             901  55
3             889  60
4             862  50
5             935  52
6             876  66
7             995  74
8             879  52
9             934  65
10            920  66
11            892  56
12            930  47
13            834  58
```

```
14             884  57
15             928  49
16             914  53
FraudFound_P          0         1
RepNumber
1              0.063737  0.068256
2              0.062151  0.059588
3              0.061323  0.065005
4              0.059461  0.054171
5              0.064496  0.056338
6              0.060426  0.071506
7              0.068635  0.080173
8              0.060633  0.056338
9              0.064427  0.070423
10             0.063461  0.071506
11             0.061530  0.060672
12             0.064151  0.050921
13             0.057529  0.062839
14             0.060978  0.061755
15             0.064013  0.053088
16             0.063048  0.057421
FraudFound_P       0    1
Deductible
300                6    2
400            13982  856
500              216   47
700              293   18
FraudFound_P          0         1
Deductible
300            0.000414  0.002167
400            0.964475  0.927411
500            0.014900  0.050921
700            0.020211  0.019502
FraudFound_P      0    1
DriverRating
1              3712  232
2              3587  214
3              3642  242
4              3556  235
FraudFound_P          0         1
DriverRating
1              0.256053  0.251354
2              0.247431  0.231853
3              0.251224  0.262189
4              0.245292  0.254605
FraudFound_P               0    1
Days_Policy_Accident
1 to 7                    13    1
```

```
15 to 30                46    3
8 to 15                 50    5
more than 30         14342  905
none                    46    9
FraudFound_P                0         1
Days_Policy_Accident
1 to 7               0.000897  0.001083
15 to 30             0.003173  0.003250
8 to 15              0.003449  0.005417
more than 30         0.989308  0.980498
none                 0.003173  0.009751
FraudFound_P           0    1
Days_Policy_Claim
15 to 30              50    6
8 to 15               18    3
more than 30       14428  914
none                   1    0
FraudFound_P              0         1
Days_Policy_Claim
15 to 30           0.003449  0.006501
8 to 15            0.001242  0.003250
more than 30       0.995240  0.990249
none               0.000069  0.000000
FraudFound_P           0    1
PastNumberOfClaims
1                   3351  222
2 to 4              5191  294
more than 4         1942   68
none                4013  339
FraudFound_P               0         1
PastNumberOfClaims
1                   0.231151  0.240520
2 to 4              0.358074  0.318527
more than 4         0.133959  0.073673
none                0.276816  0.367281
FraudFound_P     0    1
AgeOfVehicle
2 years         70    3
3 years        139   13
4 years        208   21
5 years       1262   95
6 years       3220  228
7 years       5482  325
more than 7   3775  206
new            341   32
FraudFound_P         0         1
AgeOfVehicle
2 years       0.004829  0.003250
```

```
3 years       0.009588  0.014085
4 years       0.014348  0.022752
5 years       0.087052  0.102925
6 years       0.222115  0.247021
7 years       0.378147  0.352113
more than 7   0.260399  0.223185
new           0.023522  0.034670
FraudFound_P          0    1
AgeOfPolicyHolder
16 to 17            289   31
18 to 20             13    2
21 to 25             92   16
26 to 30            580   33
31 to 35           5233  360
36 to 40           3806  237
41 to 50           2684  144
51 to 65           1322   70
over 65             478   30
FraudFound_P              0         1
AgeOfPolicyHolder
16 to 17           0.019935  0.033586
18 to 20           0.000897  0.002167
21 to 25           0.006346  0.017335
26 to 30           0.040008  0.035753
31 to 35           0.360971  0.390033
36 to 40           0.262537  0.256771
41 to 50           0.185142  0.156013
51 to 65           0.091191  0.075840
over 65            0.032972  0.032503
FraudFound_P          0    1
PoliceReportFiled
No                 14085  907
Yes                  412   16
FraudFound_P              0         1
PoliceReportFiled
No                 0.97158  0.982665
Yes                0.02842  0.017335
FraudFound_P       0    1
WitnessPresent
No              14413  920
Yes                84    3
FraudFound_P          0        1
WitnessPresent
No              0.994206  0.99675
Yes             0.005794  0.00325
FraudFound_P     0    1
AgentType
External      14260  919
```

```
Internal        237    4
FraudFound_P          0         1
AgentType
External      0.983652  0.995666
Internal      0.016348  0.004334
FraudFound_P             0    1
NumberOfSuppliments
1 to 2                2330  159
3 to 5                1920   97
more than 5           3672  195
none                  6575  472
FraudFound_P                 0         1
NumberOfSuppliments
1 to 2                0.160723  0.172264
3 to 5                0.132441  0.105092
more than 5           0.253294  0.211268
none                  0.453542  0.511376
FraudFound_P              0    1
AddressChange_Claim
1 year                  159   11
2 to 3 years            240   51
4 to 8 years            598   33
no change             13499  825
under 6 months            1    3
FraudFound_P                 0         1
AddressChange_Claim
1 year                0.010968  0.011918
2 to 3 years          0.016555  0.055255
4 to 8 years          0.041250  0.035753
no change             0.931158  0.893824
under 6 months        0.000069  0.003250
FraudFound_P      0    1
NumberOfCars
1 vehicle     13466  850
2 vehicles      666   43
3 to 4          343   29
5 to 8           20    1
more than 8       2    0
FraudFound_P          0         1
NumberOfCars
1 vehicle     0.928882  0.920910
2 vehicles    0.045941  0.046587
3 to 4        0.023660  0.031419
5 to 8        0.001380  0.001083
more than 8   0.000138  0.000000
FraudFound_P     0    1
Year
1994          5733  409
```

```
1995           4894  301
1996           3870  213
FraudFound_P          0         1
Year
1994           0.395461  0.443120
1995           0.337587  0.326111
1996           0.266952  0.230769
FraudFound_P      0    1
BasePolicy
All Perils     3997  452
Collision      5527  435
Liability      4973   36
FraudFound_P          0         1
BasePolicy
All Perils     0.275712  0.489707
Collision      0.381251  0.471289
Liability      0.343036  0.039003
```

4 Part 2 - Data preproccesing

4.0.1 2.a drop irrelevant colmuns

Function for dropping Irrelevant_colmuns - PolicyType, PolicyNumber

[ ]:
```python
def Irrelevant_col(df , drop):
    df.drop(drop, axis=1, inplace=True)


#drop in my data set
drop = ["PolicyType","PolicyNumber"]
dataset_new = dataset
Irrelevant_col(dataset_new, drop)

dataset_new
```

[ ]:
```
      Month  WeekOfMonth  DayOfWeek     Make AccidentArea DayOfWeekClaimed  \
0       Dec            5  Wednesday    Honda        Urban          Tuesday
1       Jan            3  Wednesday    Honda        Urban           Monday
2       Oct            5     Friday    Honda        Urban         Thursday
3       Jun            2   Saturday   Toyota        Rural           Friday
4       Jan            5     Monday    Honda        Urban          Tuesday
…       …           …        …     …         …              …
15415   Nov            4     Friday   Toyota        Urban          Tuesday
15416   Nov            5   Thursday  Pontiac        Urban           Friday
15417   Nov            5   Thursday   Toyota        Rural           Friday
15418   Dec            1     Monday   Toyota        Urban         Thursday
15419   Dec            2  Wednesday   Toyota        Urban         Thursday
```

```
      MonthClaimed  WeekOfMonthClaimed     Sex MaritalStatus  …  \
0              Jan                   1  Female        Single  …
1              Jan                   4    Male        Single  …
2              Nov                   2    Male       Married  …
3              Jul                   1    Male       Married  …
4              Feb                   2  Female        Single  …
…            …                 …     …             …   …
15415          Nov                   5    Male       Married  …
15416          Dec                   1    Male       Married  …
15417          Dec                   1    Male        Single  …
15418          Dec                   2  Female       Married  …
15419          Dec                   3    Male        Single  …

       AgeOfVehicle AgeOfPolicyHolder PoliceReportFiled WitnessPresent  \
0           3 years          26 to 30                No             No
1           6 years          31 to 35               Yes             No
2           7 years          41 to 50                No             No
3       more than 7          51 to 65               Yes             No
4           5 years          31 to 35                No             No
…             …               …                …             …
15415       6 years          31 to 35                No             No
15416       6 years          31 to 35                No             No
15417       5 years          26 to 30                No             No
15418       2 years          31 to 35                No             No
15419       5 years          26 to 30                No             No

       AgentType  NumberOfSuppliments  AddressChange_Claim  NumberOfCars  \
0       External                 none               1 year        3 to 4
1       External                 none            no change     1 vehicle
2       External                 none            no change     1 vehicle
3       External          more than 5            no change     1 vehicle
4       External                 none            no change     1 vehicle
…          …                   …                    …            …
15415   External                 none            no change     1 vehicle
15416   External          more than 5            no change        3 to 4
15417   External               1 to 2            no change     1 vehicle
15418   External          more than 5            no change     1 vehicle
15419   External               1 to 2            no change     1 vehicle

       Year  BasePolicy
0      1994   Liability
1      1994   Collision
2      1994   Collision
3      1994   Liability
4      1994   Collision
…      …        …
15415  1996   Collision
```

```
15416  1996   Liability
15417  1996   Collision
15418  1996  All Perils
15419  1996   Collision

[15420 rows x 31 columns]
```

2.b dealing non available values changing the zero values to nan values in columns: Age, DayOfWeekClaimed, weekclaimed

[ ]:
```
#Age
print(sum(dataset['Age']  == 0))
dataset.loc[dataset['Age'] == 0, 'Age'] = np.nan

#DayOfWeekClaimed
print(dataset['DayOfWeekClaimed'].unique())
dataset[dataset['DayOfWeekClaimed'] == '0'] # obs 1516 has a
dataset.loc[dataset['DayOfWeekClaimed'] == 0, 'DayOfWeekClaimed'] = np.nan

#MonthClaimed
print(sum(dataset['MonthClaimed']  == '0'))
dataset.loc[dataset['MonthClaimed'] == '0', 'DayOfWeekClaimed'] = np.nan
```

```
320
['Tuesday' 'Monday' 'Thursday' 'Friday' 'Wednesday' 'Saturday' 'Sunday'
 '0']
1
```

Null values of age, day of week - replacing with mean

[ ]:
```
# removing rows
dataset_new_rem = dataset_new.dropna(subset = ['Age'])
dataset_new_rem = dataset_new_rem.dropna(subset =['MonthClaimed'␣
 ↪,'DayOfWeekClaimed'])
#print(dataset_new_rem.isnull().sum())

# avereging
imputer = SimpleImputer(missing_values=np.NaN, strategy='mean')
# We instantiated a SimpleImputer object looking for missing values that are␣
 ↪represented
#by np.NaN and asking Scikit-Learn to use the 'mean' as its strategy.
#This means that any np.NaN values will be imputed by the columns mean.

dataset_new_avg = dataset_new
imputer=imputer.fit(dataset_new_avg[['Age']])
dataset_new_avg[['Age']]=imputer.transform(dataset_new_avg[['Age']])
```

```
dataset_new_avg = dataset_new_avg.dropna(subset =['MonthClaimed'␣
↪,'DayOfWeekClaimed'])
#print(dataset_new_avg.isnull().sum())
```

4.0.2 2.c Dealing with categorials features

[ ]:
```
dataset_new.dtypes # can we see most of the variabales are categorial
```

[ ]:
```
Month                    object
WeekOfMonth               int64
DayOfWeek                object
Make                     object
AccidentArea             object
DayOfWeekClaimed         object
MonthClaimed             object
WeekOfMonthClaimed        int64
Sex                      object
MaritalStatus            object
Age                     float64
Fault                    object
VehicleCategory          object
VehiclePrice             object
FraudFound_P              int64
RepNumber                 int64
Deductible                int64
DriverRating              int64
Days_Policy_Accident     object
Days_Policy_Claim        object
PastNumberOfClaims       object
AgeOfVehicle             object
AgeOfPolicyHolder        object
PoliceReportFiled        object
WitnessPresent           object
AgentType                object
NumberOfSuppliments      object
AddressChange_Claim      object
NumberOfCars             object
Year                      int64
BasePolicy               object
dtype: object
```

[ ]:
```
#make a copy of the data for making changes
y =  dataset_new.FraudFound_P.copy()
X = dataset_new.drop('FraudFound_P', axis = 1, inplace=False ).copy()
```

Binary variables zero one coding:

[ ]:
```python
cols = ['AccidentArea', 'Sex', 'Fault', 'PoliceReportFiled', 'WitnessPresent',␣
 ↪'AgentType']
y_val = ['Urban','Female','Policy Holder', 'Yes', 'Yes','External']
x_val = ['Rural', 'Male', 'Third Party', 'No', 'No', 'Internal']

for i in range(len(cols)):
    X_idx1 = X[cols[i]]==y_val[i]
    X_idx2 = X[cols[i]]==x_val[i]

    X.loc[list(X_idx1),cols[i]]=1
    X.loc[list(X_idx2),cols[i]]=0

for i in range(len(cols)):
    X[cols[i]] = X[cols[i]].astype('int')

X.dtypes
```

[ ]:
```
Month                     object
WeekOfMonth                int64
DayOfWeek                 object
Make                      object
AccidentArea               int64
DayOfWeekClaimed          object
MonthClaimed              object
WeekOfMonthClaimed         int64
Sex                        int64
MaritalStatus             object
Age                      float64
Fault                      int64
VehicleCategory           object
VehiclePrice              object
RepNumber                  int64
Deductible                 int64
DriverRating               int64
Days_Policy_Accident      object
Days_Policy_Claim         object
PastNumberOfClaims        object
AgeOfVehicle              object
AgeOfPolicyHolder         object
PoliceReportFiled          int64
WitnessPresent             int64
AgentType                  int64
NumberOfSuppliments       object
AddressChange_Claim       object
NumberOfCars              object
Year                       int64
BasePolicy                object
```

```
dtype: object
```

Ordianal categorial featurs:

```
[ ]: col_ordering = [{'col':'Month','mapping':{'Jan':1,'Feb':2,'Mar':3,'Apr':4,'May':
     ↪5,'Jun':6,'Jul':7,'Aug':8,'Sep':9,'Oct':10,'Nov':11,'Dec':12}},
         {'col':'DayOfWeek','mapping':{'Monday':1,'Tuesday':2,'Wednesday':
     ↪3,'Thursday':4,'Friday':5,'Saturday':6,'Sunday':7}},
         {'col':'DayOfWeekClaimed','mapping':{'Monday':1,'Tuesday':2,'Wednesday':
     ↪3,'Thursday':4,'Friday':5,'Saturday':6,'Sunday':7}},
         {'col':'MonthClaimed','mapping':{'Jan':1,'Feb':2,'Mar':3,'Apr':4,'May':
     ↪5,'Jun':6,'Jul':7,'Aug':8,'Sep':9,'Oct':10,'Nov':11,'Dec':12}},
         {'col':'PastNumberOfClaims','mapping':{'none':0 ,'1':1,'2 to 4':2,'more␣
     ↪than 4':5 }},
         {'col':'NumberOfSuppliments','mapping':{'none':0,'1 to 2':1,'3 to 5':
     ↪3,'more than 5':6}},
         {'col':'VehiclePrice','mapping':{'more than 69000':69001,'20000 to 29000':
     ↪24500,'30000 to 39000':34500,'less than 20000':19999,
                                          '40000 to 59000':49500,'60000 to 69000':
     ↪64500}},
         {'col':'AgeOfVehicle','mapping':{'3 years':3,'6 years':6,'7 years':7,'more␣
     ↪than 7':8,'5 years':5,'new':0,'4 years':4,'2 years':2}},
     ]
     ord_encoder = OrdinalEncoder(mapping = col_ordering, return_df=True)
     X2 = ord_encoder.fit_transform(X)
     X2.loc[X2["DayOfWeekClaimed"] == -1.0,"DayOfWeekClaimed"] = 0
     X2.loc[X2["MonthClaimed"] == -1.0,"MonthClaimed"] = 0
```

ordianal categorial featurs - taking the avg for each category

```
[ ]: col_map = [{'Days_Policy_Accident':{'more than 30': 31,'15 to 30': 22.5,'none':␣
     ↪0,'1 to 7': 4,'8 to 15': 11.5}},
         {'Days_Policy_Claim':{'more than 30': 31,'15 to 30': 22.5,'8 to 15': 11.
     ↪5,'none': 0}},
         {'AgeOfPolicyHolder':{'26 to 30':28,'31 to 35':33,'41 to 50':45.5,'51 to␣
     ↪65':58,'21 to 25':23,'36 to 40':38,'16 to 17':16.5,
                                               'over 65':66,'18 to 20':19}},
         {'AddressChange_Claim':{'1 year':1,'no change':0,'4 to 8 years':6,'2 to 3␣
     ↪years':2.5,'under 6 months':0.5}},
         {'NumberOfCars':{'3 to 4':3.5,'1 vehicle':1,'2 vehicles':2,'5 to 8':6.
     ↪5,'more than 8':9}},
     ]

     X3 = X2.copy()
     for i in range(len(col_map)):
         X3.replace(col_map[i], inplace=True)
```

```
X3.dtypes
```

```
[ ]: Month                     int64
     WeekOfMonth               int64
     DayOfWeek                 int64
     Make                     object
     AccidentArea              int64
     DayOfWeekClaimed        float64
     MonthClaimed            float64
     WeekOfMonthClaimed        int64
     Sex                       int64
     MaritalStatus            object
     Age                     float64
     Fault                     int64
     VehicleCategory          object
     VehiclePrice              int64
     RepNumber                 int64
     Deductible                int64
     DriverRating              int64
     Days_Policy_Accident    float64
     Days_Policy_Claim       float64
     PastNumberOfClaims        int64
     AgeOfVehicle              int64
     AgeOfPolicyHolder       float64
     PoliceReportFiled         int64
     WitnessPresent            int64
     AgentType                 int64
     NumberOfSuppliments       int64
     AddressChange_Claim     float64
     NumberOfCars            float64
     Year                      int64
     BasePolicy               object
     dtype: object
```

One hot encoder for the categorial features

```
[ ]: #implementing one-hot encoding
     one_hot_encoder = OneHotEncoder(cols =␣
      ↪['Make','MaritalStatus','VehicleCategory','BasePolicy'],use_cat_names=True,␣
      ↪return_df=True)

     #implementing label encoding, with random assignment of integers to each label
     # assumes no natrual underlying order to the feature labels
     ord_encoder1 = OrdinalEncoder(cols =␣
      ↪['Make','MaritalStatus','VehicleCategory','BasePolicy'],return_df=True)

     #implementing binary encoding
```

```
#represents the data
bi_encoder = BinaryEncoder(cols =␣
 ↪['Make','MaritalStatus','VehicleCategory','BasePolicy'], return_df=True)

#implimented a simple switch to change how I wanted to encode the variables
#Allowed for a updating the independet variables quickly and not missing a code␣
 ↪box

switch_val = 0
if switch_val ==0:
    X4 = one_hot_encoder.fit_transform(X3)
elif switch_val==1:
    X4 = ord_encoder1.fit_transform(X3)
else:
    X4 = bi_encoder.fit_transform(X3)

X4.dtypes
```

[ ]:
```
Month                        int64
WeekOfMonth                  int64
DayOfWeek                    int64
Make_Honda                   int64
Make_Toyota                  int64
Make_Ford                    int64
Make_Mazda                   int64
Make_Chevrolet               int64
Make_Pontiac                 int64
Make_Accura                  int64
Make_Dodge                   int64
Make_Mercury                 int64
Make_Jaguar                  int64
Make_Nisson                  int64
Make_VW                      int64
Make_Saab                    int64
Make_Saturn                  int64
Make_Porche                  int64
Make_BMW                     int64
Make_Mecedes                 int64
Make_Ferrari                 int64
Make_Lexus                   int64
AccidentArea                 int64
DayOfWeekClaimed           float64
MonthClaimed               float64
WeekOfMonthClaimed           int64
Sex                          int64
MaritalStatus_Single         int64
MaritalStatus_Married        int64
```

```
MaritalStatus_Widow           int64
MaritalStatus_Divorced        int64
Age                         float64
Fault                         int64
VehicleCategory_Sport         int64
VehicleCategory_Utility       int64
VehicleCategory_Sedan         int64
VehiclePrice                  int64
RepNumber                     int64
Deductible                    int64
DriverRating                  int64
Days_Policy_Accident        float64
Days_Policy_Claim           float64
PastNumberOfClaims            int64
AgeOfVehicle                  int64
AgeOfPolicyHolder           float64
PoliceReportFiled             int64
WitnessPresent                int64
AgentType                     int64
NumberOfSuppliments           int64
AddressChange_Claim         float64
NumberOfCars                float64
Year                          int64
BasePolicy_Liability          int64
BasePolicy_Collision          int64
BasePolicy_All Perils         int64
dtype: object
```

[ ]:
```python
print(pearsonr(X4.Age, X4.AgeOfPolicyHolder))
print(pearsonr(X4.MonthClaimed, X4.Month))
print(pearsonr(X4.BasePolicy_Liability,X4.VehicleCategory_Sport))
```

```
(0.8995052651641743, 0.0)
(0.8335242937029943, 0.0)
(0.944432189599651, 0.0)
```

[ ]:
```python
X4.corr()
```

[ ]:
```
                        Month  WeekOfMonth  DayOfWeek  Make_Honda  \
Month                1.000000     0.031442   0.000968   -0.021027
WeekOfMonth          0.031442     1.000000  -0.013370    0.012041
DayOfWeek            0.000968    -0.013370   1.000000   -0.000321
Make_Honda          -0.021027     0.012041  -0.000321    1.000000
Make_Toyota         -0.003369     0.004741   0.002423   -0.237332
Make_Ford            0.002855    -0.004448   0.000286   -0.081684
Make_Mazda           0.005397    -0.009569  -0.000881   -0.199975
Make_Chevrolet      -0.002603    -0.004139  -0.006622   -0.164797
```

Make_Pontiac	0.018575	−0.001375	0.014497	−0.271162
Make_Accura	0.004674	−0.001817	−0.013943	−0.083719
Make_Dodge	−0.004390	0.018664	−0.006279	−0.039752
Make_Mercury	0.000380	−0.002377	−0.001814	−0.034659
Make_Jaguar	−0.004119	−0.004422	0.001915	−0.009295
Make_Nisson	−0.003285	0.001534	−0.005380	−0.020801
Make_VW	0.004612	−0.005693	−0.015286	−0.064420
Make_Saab	−0.005218	−0.003121	0.006975	−0.039568
Make_Saturn	−0.011811	−0.007190	0.001683	−0.028949
Make_Porche	0.006266	−0.008233	−0.016450	−0.008485
Make_BMW	0.003070	0.005124	0.010386	−0.014701
Make_Mecedes	−0.004136	−0.003610	0.003598	−0.007589
Make_Ferrari	0.005274	−0.011399	−0.013280	−0.005366
Make_Lexus	−0.010184	0.013832	−0.011424	−0.003794
AccidentArea	0.002140	0.009116	−0.025699	0.018914
DayOfWeekClaimed	−0.006647	0.010013	−0.056893	−0.006690
MonthClaimed	0.833524	0.013915	−0.003859	−0.025446
WeekOfMonthClaimed	0.053917	0.275400	0.004716	0.002430
Sex	0.007397	−0.005314	−0.000990	−0.025704
MaritalStatus_Single	−0.009206	0.017114	0.017814	0.094790
MaritalStatus_Married	0.009206	−0.015966	−0.018147	−0.093388
MaritalStatus_Widow	0.000114	0.006773	0.000499	−0.004799
MaritalStatus_Divorced	−0.000472	−0.011459	0.002622	−0.001934
Age	−0.023429	−0.011781	−0.004563	−0.056550
Fault	0.003619	−0.025456	−0.011941	−0.012439
VehicleCategory_Sport	−0.012695	−0.003411	−0.049847	0.087863
VehicleCategory_Utility	0.011504	−0.008759	−0.029359	−0.066362
VehicleCategory_Sedan	0.008761	0.006206	0.058630	−0.064946
VehiclePrice	−0.036599	−0.004793	−0.028024	0.116680
RepNumber	0.009520	0.005283	0.002350	0.005572
Deductible	−0.003074	−0.003993	0.000393	−0.018293
DriverRating	0.008318	−0.016817	0.001387	−0.010871
Days_Policy_Accident	0.003888	−0.032973	0.000707	−0.016398
Days_Policy_Claim	0.004077	−0.017198	0.018507	−0.016439
PastNumberOfClaims	−0.023655	−0.009283	−0.032424	−0.000574
AgeOfVehicle	0.027530	−0.009798	−0.008550	−0.280443
AgeOfPolicyHolder	0.006801	−0.002224	0.002434	−0.159659
PoliceReportFiled	0.047896	0.013026	0.015406	−0.017145
WitnessPresent	−0.001515	0.013713	0.004251	−0.008541
AgentType	−0.023576	−0.006477	−0.003516	0.014614
NumberOfSuppliments	0.024617	0.000177	−0.001544	−0.066027
AddressChange_Claim	0.001477	0.000147	0.006422	−0.006243
NumberOfCars	−0.015607	0.002901	−0.010573	−0.001082
Year	0.048852	−0.003906	0.007275	−0.008792
BasePolicy_Liability	−0.011205	−0.004198	−0.055095	0.023751
BasePolicy_Collision	0.032236	−0.004401	0.039266	0.035583
BasePolicy_All Perils	−0.023067	0.009069	0.014744	−0.062796

	Make_Toyota	Make_Ford	Make_Mazda	Make_Chevrolet	\
Month	-0.003369	0.002855	0.005397	-0.002603	
WeekOfMonth	0.004741	-0.004448	-0.009569	-0.004139	
DayOfWeek	0.002423	0.000286	-0.000881	-0.006622	
Make_Honda	-0.237332	-0.081684	-0.199975	-0.164797	
Make_Toyota	1.000000	-0.087339	-0.213818	-0.176205	
Make_Ford	-0.087339	1.000000	-0.073591	-0.060646	
Make_Mazda	-0.213818	-0.073591	1.000000	-0.148470	
Make_Chevrolet	-0.176205	-0.060646	-0.148470	1.000000	
Make_Pontiac	-0.289933	-0.099789	-0.244297	-0.201322	
Make_Accura	-0.089514	-0.030809	-0.075424	-0.062156	
Make_Dodge	-0.042503	-0.014629	-0.035813	-0.029513	
Make_Mercury	-0.037058	-0.012755	-0.031225	-0.025732	
Make_Jaguar	-0.009939	-0.003421	-0.008374	-0.006901	
Make_Nisson	-0.022241	-0.007655	-0.018740	-0.015444	
Make_VW	-0.068879	-0.023707	-0.058037	-0.047828	
Make_Saab	-0.042307	-0.014561	-0.035647	-0.029377	
Make_Saturn	-0.030953	-0.010653	-0.026081	-0.021493	
Make_Porche	-0.009072	-0.003123	-0.007644	-0.006300	
Make_BMW	-0.015719	-0.005410	-0.013245	-0.010915	
Make_Mecedes	-0.008114	-0.002793	-0.006837	-0.005634	
Make_Ferrari	-0.005737	-0.001975	-0.004834	-0.003984	
Make_Lexus	-0.004057	-0.001396	-0.003418	-0.002817	
AccidentArea	0.003407	-0.016896	0.017126	-0.016928	
DayOfWeekClaimed	0.000462	0.017840	0.002158	-0.004756	
MonthClaimed	0.003542	0.002124	0.007731	-0.000558	
WeekOfMonthClaimed	-0.009855	-0.008962	-0.003094	0.005526	
Sex	0.017835	-0.035614	0.036963	0.009259	
MaritalStatus_Single	-0.014755	-0.049172	0.019583	-0.022905	
MaritalStatus_Married	0.014124	0.048220	-0.018698	0.023701	
MaritalStatus_Widow	0.009890	0.024116	-0.008878	-0.007938	
MaritalStatus_Divorced	-0.003186	-0.012202	0.001025	-0.000847	
Age	-0.006317	0.064238	-0.048459	0.031309	
Fault	-0.005060	-0.003225	-0.014332	-0.004318	
VehicleCategory_Sport	-0.038119	-0.020521	0.044325	-0.002228	
VehicleCategory_Utility	-0.080226	0.060265	-0.028319	0.116977	
VehicleCategory_Sedan	0.063618	0.000615	-0.034441	-0.035836	
VehiclePrice	-0.158961	0.121485	-0.034307	0.115709	
RepNumber	0.006306	0.001385	-0.008413	-0.004676	
Deductible	0.012321	0.004673	-0.005070	-0.000241	
DriverRating	-0.002516	0.018409	-0.005847	0.005576	
Days_Policy_Accident	0.009426	-0.004023	0.006525	0.006661	
Days_Policy_Claim	0.011277	0.007675	0.007660	0.008044	
PastNumberOfClaims	-0.036762	0.004081	0.001717	0.033660	
AgeOfVehicle	0.036972	0.068598	0.019212	0.060899	
AgeOfPolicyHolder	0.024686	0.068059	-0.022503	0.043510	

PoliceReportFiled	-0.004546	0.017613	0.010605	0.010567
WitnessPresent	-0.007777	0.002372	-0.007899	-0.001345
AgentType	0.008820	-0.021641	0.006964	-0.023030
NumberOfSuppliments	-0.004340	0.024850	0.008689	0.018211
AddressChange_Claim	0.003370	-0.008070	0.007222	0.012737
NumberOfCars	0.004898	-0.010908	0.002509	-0.003452
Year	0.000350	0.000522	-0.001274	-0.003507
BasePolicy_Liability	-0.015102	-0.012486	0.023229	0.019971
BasePolicy_Collision	-0.019789	-0.027682	0.065110	-0.023050
BasePolicy_All Perils	0.036880	0.042660	-0.093993	0.004131

	Make_Pontiac	Make_Accura	…	PoliceReportFiled \
Month	0.018575	0.004674	…	0.047896
WeekOfMonth	-0.001375	-0.001817	…	0.013026
DayOfWeek	0.014497	-0.013943	…	0.015406
Make_Honda	-0.271162	-0.083719	…	-0.017145
Make_Toyota	-0.289933	-0.089514	…	-0.004546
Make_Ford	-0.099789	-0.030809	…	0.017613
Make_Mazda	-0.244297	-0.075424	…	0.010605
Make_Chevrolet	-0.201322	-0.062156	…	0.010567
Make_Pontiac	1.000000	-0.102274	…	0.000456
Make_Accura	-0.102274	1.000000	…	0.002061
Make_Dodge	-0.048562	-0.014993	…	0.004592
Make_Mercury	-0.042340	-0.013072	…	-0.007034
Make_Jaguar	-0.011355	-0.003506	…	-0.003334
Make_Nisson	-0.025411	-0.007846	…	-0.007460
Make_VW	-0.078697	-0.024297	…	0.003368
Make_Saab	-0.048337	-0.014924	…	-0.014190
Make_Saturn	-0.035365	-0.010919	…	-0.010382
Make_Porche	-0.010366	-0.003200	…	-0.003043
Make_BMW	-0.017960	-0.005545	…	-0.005272
Make_Mecedes	-0.009271	-0.002862	…	-0.002722
Make_Ferrari	-0.006555	-0.002024	…	-0.001924
Make_Lexus	-0.004635	-0.001431	…	-0.001361
AccidentArea	-0.009039	-0.003812	…	0.001754
DayOfWeekClaimed	-0.009994	0.014267	…	-0.010992
MonthClaimed	0.009767	0.004594	…	0.056724
WeekOfMonthClaimed	0.008607	0.001928	…	0.023510
Sex	0.011474	-0.044583	…	0.007413
MaritalStatus_Single	-0.020723	-0.046151	…	0.012010
MaritalStatus_Married	0.020141	0.047801	…	-0.011863
MaritalStatus_Widow	-0.002235	-0.008475	…	0.000237
MaritalStatus_Divorced	0.004474	-0.007130	…	-0.000617
Age	0.021635	0.034908	…	-0.009089
Fault	0.003240	0.040838	…	-0.027246
VehicleCategory_Sport	-0.044810	-0.052978	…	-0.038732
VehicleCategory_Utility	-0.014593	0.155737	…	-0.007164

VehicleCategory_Sedan	0.048869	0.001537	…	0.040469
VehiclePrice	-0.175717	0.128124	…	0.009106
RepNumber	-0.006369	0.004411	…	0.006107
Deductible	0.004569	-0.004595	…	0.009005
DriverRating	0.011696	-0.011854	…	0.015947
Days_Policy_Accident	0.005114	-0.013999	…	-0.017292
Days_Policy_Claim	-0.003102	-0.010459	…	-0.009508
PastNumberOfClaims	-0.000443	0.004945	…	-0.005798
AgeOfVehicle	0.080781	0.057802	…	-0.001556
AgeOfPolicyHolder	0.050527	0.036382	…	-0.005590
PoliceReportFiled	0.000456	0.002061	…	1.000000
WitnessPresent	0.022733	0.011746	…	0.198096
AgentType	0.006009	-0.004926	…	-0.023268
NumberOfSuppliments	0.028510	-0.004655	…	0.005256
AddressChange_Claim	-0.009144	-0.000589	…	-0.007406
NumberOfCars	-0.008096	0.004527	…	-0.024648
Year	0.014070	-0.011705	…	0.021206
BasePolicy_Liability	-0.019347	-0.043670	…	-0.041331
BasePolicy_Collision	-0.014644	-0.044448	…	0.034467
BasePolicy_All Perils	0.035737	0.092914	…	0.005675

	WitnessPresent	AgentType	NumberOfSuppliments	\
Month	-0.001515	-0.023576	0.024617	
WeekOfMonth	0.013713	-0.006477	0.000177	
DayOfWeek	0.004251	-0.003516	-0.001544	
Make_Honda	-0.008541	0.014614	-0.066027	
Make_Toyota	-0.007777	0.008820	-0.004340	
Make_Ford	0.002372	-0.021641	0.024850	
Make_Mazda	-0.007899	0.006964	0.008689	
Make_Chevrolet	-0.001345	-0.023030	0.018211	
Make_Pontiac	0.022733	0.006009	0.028510	
Make_Accura	0.011746	-0.004926	-0.004655	
Make_Dodge	-0.006356	-0.008091	0.002071	
Make_Mercury	-0.005541	0.009269	0.001484	
Make_Jaguar	-0.001486	0.002486	-0.003123	
Make_Nisson	-0.003326	0.005563	0.005496	
Make_VW	-0.010300	-0.021724	0.018432	
Make_Saab	0.004056	0.010582	0.003355	
Make_Saturn	-0.004628	-0.009340	-0.003591	
Make_Porche	-0.001357	0.002269	-0.009155	
Make_BMW	-0.002351	-0.012840	0.003464	
Make_Mecedes	-0.001213	0.002030	-0.013393	
Make_Ferrari	-0.000858	0.001435	-0.000269	
Make_Lexus	-0.000607	0.001015	-0.003443	
AccidentArea	-0.028362	0.005189	-0.018695	
DayOfWeekClaimed	-0.002415	-0.005900	0.003279	
MonthClaimed	0.001021	-0.032042	0.034534	

WeekOfMonthClaimed	0.009369	0.011314	0.011589
Sex	0.005585	0.012681	-0.010698
MaritalStatus_Single	0.010492	-0.003176	-0.026884
MaritalStatus_Married	-0.011123	0.003455	0.029001
MaritalStatus_Widow	0.014601	-0.015964	-0.005530
MaritalStatus_Divorced	-0.005301	0.008868	-0.011388
Age	-0.007501	0.007343	0.034230
Fault	-0.061036	0.005306	-0.011595
VehicleCategory_Sport	-0.027693	-0.044206	0.023742
VehicleCategory_Utility	0.004373	0.007021	0.003354
VehicleCategory_Sedan	0.025848	0.041248	-0.024469
VehiclePrice	0.010224	-0.018714	0.007826
RepNumber	0.007521	-0.005630	0.005764
Deductible	0.000586	0.004244	0.010341
DriverRating	0.010489	0.000262	0.001989
Days_Policy_Accident	-0.023680	-0.010436	0.074091
Days_Policy_Claim	-0.003147	-0.008176	0.051378
PastNumberOfClaims	-0.011111	0.011332	0.102582
AgeOfVehicle	-0.008419	-0.020840	0.151374
AgeOfPolicyHolder	-0.001870	0.003847	0.068239
PoliceReportFiled	0.198096	-0.023268	0.005256
WitnessPresent	1.000000	-0.011450	-0.015419
AgentType	-0.011450	1.000000	-0.033347
NumberOfSuppliments	-0.015419	-0.033347	1.000000
AddressChange_Claim	-0.005944	0.026692	-0.008348
NumberOfCars	-0.012535	-0.000356	-0.004290
Year	-0.015503	-0.018993	0.014884
BasePolicy_Liability	-0.039307	-0.055502	0.041335
BasePolicy_Collision	0.020201	-0.021279	-0.006172
BasePolicy_All Perils	0.018916	0.080241	-0.036091

	AddressChange_Claim	NumberOfCars	Year \
Month	0.001477	-0.015607	0.048852
WeekOfMonth	0.000147	0.002901	-0.003906
DayOfWeek	0.006422	-0.010573	0.007275
Make_Honda	-0.006243	-0.001082	-0.008792
Make_Toyota	0.003370	0.004898	0.000350
Make_Ford	-0.008070	-0.010908	0.000522
Make_Mazda	0.007222	0.002509	-0.001274
Make_Chevrolet	0.012737	-0.003452	-0.003507
Make_Pontiac	-0.009144	-0.008096	0.014070
Make_Accura	-0.000589	0.004527	-0.011705
Make_Dodge	0.003389	0.018415	0.006316
Make_Mercury	-0.007733	-0.009234	-0.002115
Make_Jaguar	0.011178	0.019147	-0.009001
Make_Nisson	-0.000738	-0.007452	0.005507
Make_VW	0.002166	-0.001995	0.006489

Make_Saab	0.004546	0.025127	-0.012177
Make_Saturn	-0.007866	0.004039	0.000982
Make_Porche	-0.004454	-0.004283	-0.001490
Make_BMW	-0.004330	0.011969	0.010366
Make_Mecedes	0.004210	-0.003831	-0.007349
Make_Ferrari	-0.002816	-0.002709	0.008982
Make_Lexus	-0.001991	-0.001915	0.001339
AccidentArea	-0.017305	-0.005518	0.002284
DayOfWeekClaimed	-0.001619	0.006807	0.003126
MonthClaimed	0.001160	-0.009481	0.053457
WeekOfMonthClaimed	0.009449	0.005903	0.012175
Sex	0.001692	0.001159	-0.000413
MaritalStatus_Single	0.002666	-0.004618	-0.015370
MaritalStatus_Married	-0.000171	0.002773	0.013732
MaritalStatus_Widow	-0.011795	-0.011344	0.007928
MaritalStatus_Divorced	-0.008360	0.019707	0.004782
Age	-0.006791	-0.003163	0.016506
Fault	0.007316	0.009841	-0.011158
VehicleCategory_Sport	0.001876	-0.004146	0.000584
VehicleCategory_Utility	0.006275	-0.006745	-0.005028
VehicleCategory_Sedan	-0.003887	0.006276	0.001060
VehiclePrice	-0.004111	-0.013551	-0.031859
RepNumber	-0.002964	-0.011262	0.009338
Deductible	0.061006	-0.000276	-0.001170
DriverRating	0.007585	0.010840	-0.013890
Days_Policy_Accident	0.004967	-0.001763	-0.006536
Days_Policy_Claim	-0.001585	-0.008215	-0.003559
PastNumberOfClaims	-0.015250	-0.005299	0.013785
AgeOfVehicle	-0.003621	0.008260	0.020945
AgeOfPolicyHolder	-0.006269	-0.001538	0.025193
PoliceReportFiled	-0.007406	-0.024648	0.021206
WitnessPresent	-0.005944	-0.012535	-0.015503
AgentType	0.026692	-0.000356	-0.018993
NumberOfSuppliments	-0.008348	-0.004290	0.014884
AddressChange_Claim	1.000000	0.392163	-0.000286
NumberOfCars	0.392163	1.000000	-0.003109
Year	-0.000286	-0.003109	1.000000
BasePolicy_Liability	0.005068	0.000251	0.009971
BasePolicy_Collision	0.000856	0.000814	0.004989
BasePolicy_All Perils	-0.006158	-0.001135	-0.015669

	BasePolicy_Liability	BasePolicy_Collision	\
Month	-0.011205	0.032236	
WeekOfMonth	-0.004198	-0.004401	
DayOfWeek	-0.055095	0.039266	
Make_Honda	0.023751	0.035583	
Make_Toyota	-0.015102	-0.019789	

Make_Ford	-0.012486	-0.027682
Make_Mazda	0.023229	0.065110
Make_Chevrolet	0.019971	-0.023050
Make_Pontiac	-0.019347	-0.014644
Make_Accura	-0.043670	-0.044448
Make_Dodge	0.010897	0.014077
Make_Mercury	-0.018853	-0.007446
Make_Jaguar	0.007379	-0.008912
Make_Nisson	-0.011769	-0.007855
Make_VW	0.026897	-0.028196
Make_Saab	-0.013421	0.008372
Make_Saturn	-0.019999	-0.007451
Make_Porche	0.002891	0.000494
Make_BMW	0.005008	-0.007688
Make_Mecedes	-0.011173	0.003750
Make_Ferrari	-0.007900	0.014345
Make_Lexus	-0.005586	0.010143
AccidentArea	0.056380	-0.038078
DayOfWeekClaimed	-0.011861	-0.006669
MonthClaimed	-0.006066	0.040181
WeekOfMonthClaimed	0.018027	-0.012527
Sex	0.061632	-0.006469
MaritalStatus_Single	0.031454	0.016498
MaritalStatus_Married	-0.028541	-0.019580
MaritalStatus_Widow	-0.006895	0.015300
MaritalStatus_Divorced	-0.013224	0.010678
Age	-0.001519	-0.104799
Fault	0.197380	-0.057170
VehicleCategory_Sport	0.944432	-0.482044
VehicleCategory_Utility	-0.093382	-0.102649
VehicleCategory_Sedan	-0.899640	0.508049
VehiclePrice	0.006224	-0.023956
RepNumber	0.003231	-0.011935
Deductible	0.018619	-0.016765
DriverRating	-0.000548	-0.007651
Days_Policy_Accident	0.037864	-0.025045
Days_Policy_Claim	0.021651	-0.018703
PastNumberOfClaims	0.357783	-0.143489
AgeOfVehicle	-0.010160	-0.021768
AgeOfPolicyHolder	-0.018736	-0.076071
PoliceReportFiled	-0.041331	0.034467
WitnessPresent	-0.039307	0.020201
AgentType	-0.055502	-0.021279
NumberOfSuppliments	0.041335	-0.006172
AddressChange_Claim	0.005068	0.000856
NumberOfCars	0.000251	0.000814
Year	0.009971	0.004989

BasePolicy_Liability	1.000000	-0.550713
BasePolicy_Collision	-0.550713	1.000000
BasePolicy_All Perils	-0.441710	-0.505597

	BasePolicy_All Perils
Month	-0.023067
WeekOfMonth	0.009069
DayOfWeek	0.014744
Make_Honda	-0.062796
Make_Toyota	0.036880
Make_Ford	0.042660
Make_Mazda	-0.093993
Make_Chevrolet	0.004131
Make_Pontiac	0.035737
Make_Accura	0.092914
Make_Dodge	-0.026394
Make_Mercury	0.027490
Make_Jaguar	0.001951
Make_Nisson	0.020608
Make_VW	0.002504
Make_Saab	0.004874
Make_Saturn	0.028680
Make_Porche	-0.003519
Make_BMW	0.003086
Make_Mecedes	0.007519
Make_Ferrari	-0.007253
Make_Lexus	-0.005128
AccidentArea	-0.017349
DayOfWeekClaimed	0.019428
MonthClaimed	-0.036919
WeekOfMonthClaimed	-0.005169
Sex	-0.056752
MaritalStatus_Single	-0.050245
MaritalStatus_Married	0.050546
MaritalStatus_Widow	-0.009319
MaritalStatus_Divorced	0.002192
Age	0.114212
Fault	-0.142571
VehicleCategory_Sport	-0.458081
VehicleCategory_Utility	0.206853
VehicleCategory_Sedan	0.383832
VehiclePrice	0.019315
RepNumber	0.009489
Deductible	-0.001226
DriverRating	0.008790
Days_Policy_Accident	-0.012219
Days_Policy_Claim	-0.002277

```
PastNumberOfClaims                   -0.215590
AgeOfVehicle                          0.033899
AgeOfPolicyHolder                     0.101130
PoliceReportFiled                     0.005675
WitnessPresent                        0.018916
AgentType                             0.080241
NumberOfSuppliments                  -0.036091
AddressChange_Claim                  -0.006158
NumberOfCars                         -0.001135
Year                                 -0.015669
BasePolicy_Liability                 -0.441710
BasePolicy_Collision                 -0.505597
BasePolicy_All Perils                 1.000000

[55 rows x 55 columns]
```

Drop Make_% for avoiding sparse Matrix Drop AgePolicyHolder beacuse 0.96 corr with Age

```
[ ]: drop = ['Make_Honda', 'Make_Toyota','Make_Toyota',␣
      ↪'Make_Ford','Make_Mazda','Make_Chevrolet','Make_Pontiac',
             'Make_Accura', 'Make_Accura','Make_Dodge', 'Make_Mercury',␣
      ↪'Make_Jaguar','Make_Nisson','Make_VW','Make_Saab',
            ␣
      ↪'Make_Saturn','Make_Porche','Make_BMW','Make_BMW','Make_Mecedes','Make_Ferrari','Make_Lexus
             'AgeOfPolicyHolder']
     X5 = X4.copy()
     Irrelevant_col(X5, drop)
```

4.0.3 2.d splitting our data to train, validation and test sets

```
[ ]: # splitting function
     def train_val_test_split(X, y, train_ratio, validation_ratio, test_ratio):
         X_train, X_test, y_train, y_test = train_test_split(X, y,␣
      ↪test_size=test_ratio, random_state = 3)
         X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train,
                                                              test_size=␣
      ↪validation_ratio/(train_ratio+validation_ratio), random_state =2022)
         return X_train, y_train, X_valid, y_valid,  X_test, y_test
```

```
[ ]: # with out drop AgeOfPolictyHolder and Make_%
     X_train, y_train, X_valid, y_valid,  X_test, y_test = train_val_test_split(X=␣
      ↪X4, y = y, train_ratio= 0.7, validation_ratio= 0.2, test_ratio = 0.1)

     print(X_train.shape)
     print(X_test.shape)
     print(X_valid.shape)
```

```
print(y_train.shape)
print(y_test.shape)
print(y_valid.shape)
print(len(y_valid)/len(y))
```

```
(10793, 55)
(1542, 55)
(3085, 55)
(10793,)
(1542,)
(3085,)
0.20006485084306097
```

[ ]:
```
#final splitting including all preproccesing
X_train1, y_train1, X_valid1, y_valid1,  X_test1, y_test1 =␣
 ↪train_val_test_split(X= X5, y = y, train_ratio= 0.7, validation_ratio= 0.2,␣
 ↪test_ratio = 0.1)

print(X_train1.shape)
print(X_test1.shape)
print(X_valid1.shape)
print(y_train1.shape)
print(y_test1.shape)
print(y_valid1.shape)
print(len(y_valid1)/len(y))
```

```
(10793, 35)
(1542, 35)
(3085, 35)
(10793,)
(1542,)
(3085,)
0.20006485084306097
```

5 Part 3 - Techniques for imbalance dataset

5.0.1 3.a oversampling

[ ]:
```
print('Original traim shape %s' % Counter(y_train1))
sm = SMOTE(random_state=2022)
X_res1, y_res1 = sm.fit_resample(X_train1, y_train1)
print('Resampled train shape %s' % Counter(y_res1))
```

```
Original train shape Counter({0: 10131, 1: 662})
Resampled train shape Counter({0: 10131, 1: 10131})
```

5.0.2 3.b undersampling

[ ]:
```
undersample = NearMiss(version=1, n_neighbors=3)
# transform the dataset
X_under1, y_under1 = undersample.fit_resample(X_train1, y_train1)
# summarize the new class distribution
print('Resampled train shape %s' % Counter(y_under1))
```

```
Resampled train shape Counter({0: 662, 1: 662})
```

5.0.3 3.c Feature selection decompositon

[ ]:
```
# input: train sample, m - number of fetures to select, K vec of clusters for␣
↪each class, La - label for train
# K = (num_clusters_maority, num_clusters minority)
#  first phase: local clustering
    # for each class
        # kmeans_cluster(tr(class), K[class])
        # relab;e(label(class), label(new_subclass))
# 2nd_phase: feature sellect
    # select M best of mutual information method
#    return train_M_best
from sklearn.cluster import KMeans

def decomp_fs_names(X, y, k, m):
    X_majority1 = X[y == 0]
    X_minority1 = X[y == 1]
    k_cluster = KMeans(n_clusters= k, random_state=2022)
    k_cluster = k_cluster.fit(X_majority1)
    labels= k_cluster.labels_ + 2
    labels = np.append(labels, np.repeat(1, sum(y == 1)))
    X_new1= X_majority1.copy()
    X_new1= pd.concat([X_new1 , X_minority1 ])
    X_comp_filter1 = SelectKBest(mutual_info_classif, k=m).fit(X_new1, labels)
    # names chosen by compision based Feature Selection
    comp_names = X_comp_filter1.feature_names_in_[X_comp_filter1.
↪get_support(indices=True)]
    return comp_names
```

6 Part 4 - Build classifiers

RF - baseline

[ ]:
```
# Random forest
rf = RandomForestClassifier(n_estimators = 100, max_features ='sqrt',␣
↪random_state = 3, max_depth =5)
rf.fit(X_train1, y_train1)
```

```
rf_pred = rf.predict(X_valid1)
```

RF - SMOTH

```
[ ]: rf_smoth1 = RandomForestClassifier(n_estimators = 100, max_features ='sqrt',␣
      ↪random_state = 2022, max_depth =5)
     rf_smoth1.fit(X_res1, y_res1)
     rf_smoth1_pred = rf_smoth1.predict(X_valid1)
```

RF-NearMiss

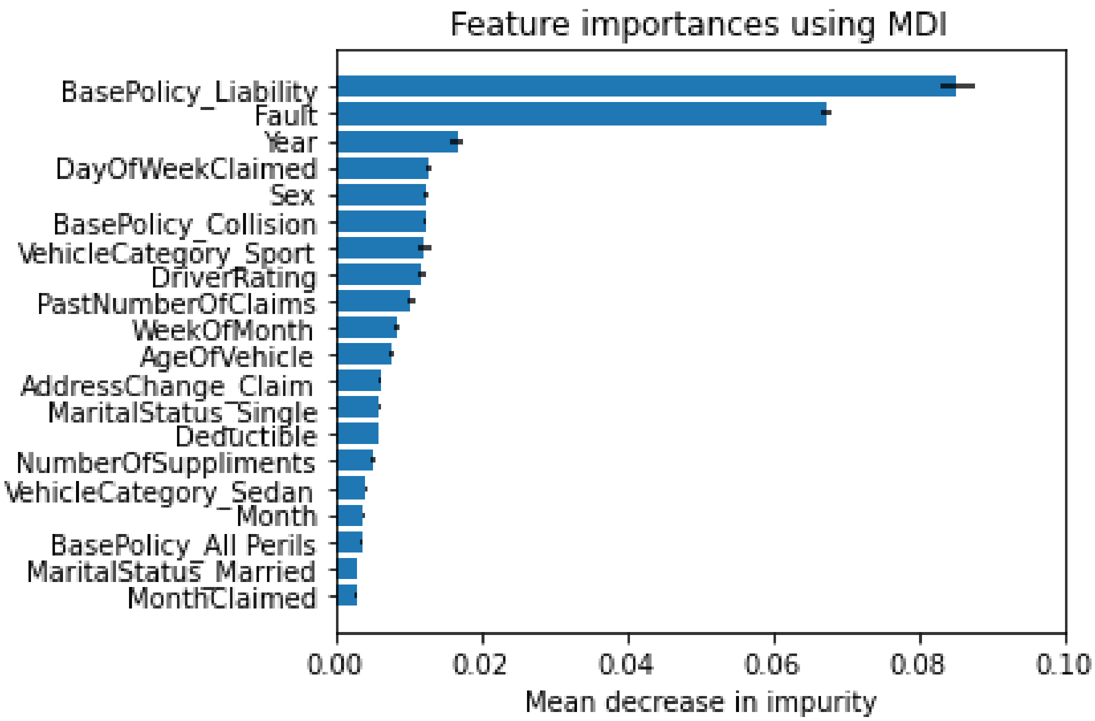
```
[ ]: rf_nearmiss1 = RandomForestClassifier(n_estimators = 100, max_features ='sqrt',␣
      ↪random_state = 2022, max_depth =5)
     rf_nearmiss1.fit(X_under1, y_under1)
     rf_nearmiss1_pred = rf_nearmiss1.predict(X_valid1)
```

RF- CS

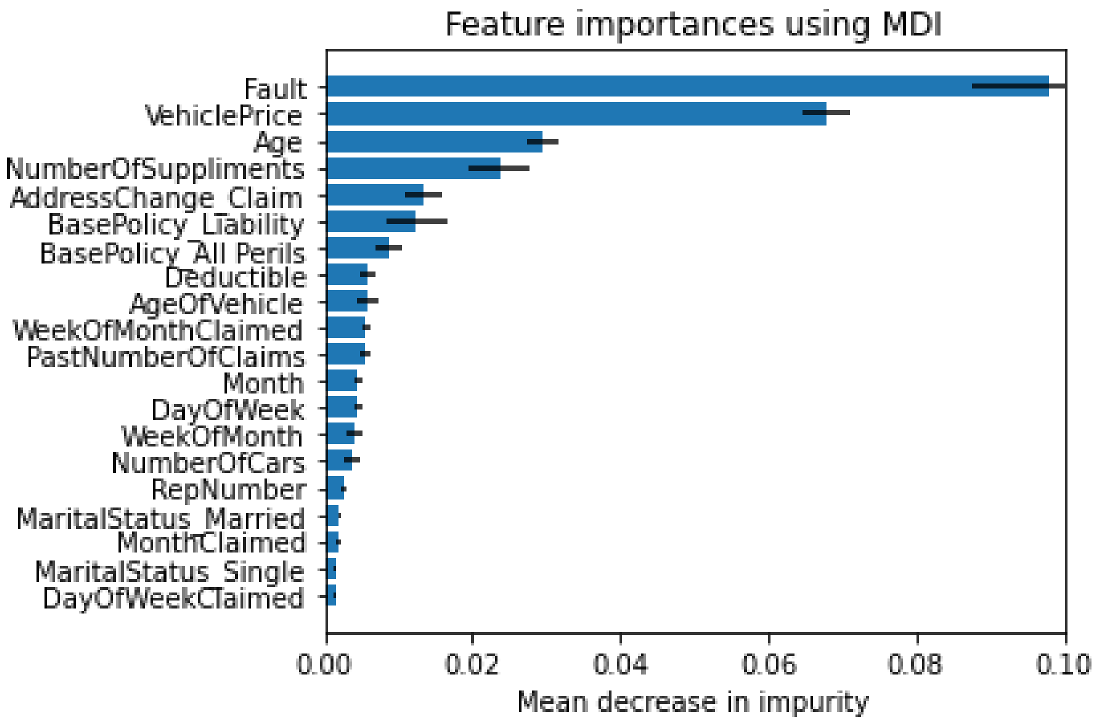
```
[ ]: rf_cs1 = RandomForestClassifier(n_estimators = 100, max_features ='sqrt',␣
      ↪random_state = 2022, max_depth =5,
                                            class_weight={0: 1 ,1:16})
     rf_cs1.fit(X_train1, y_train1)
     rf_cs1_pred = rf_cs1.predict(X_valid1)
```

feature importance for RF classifiers

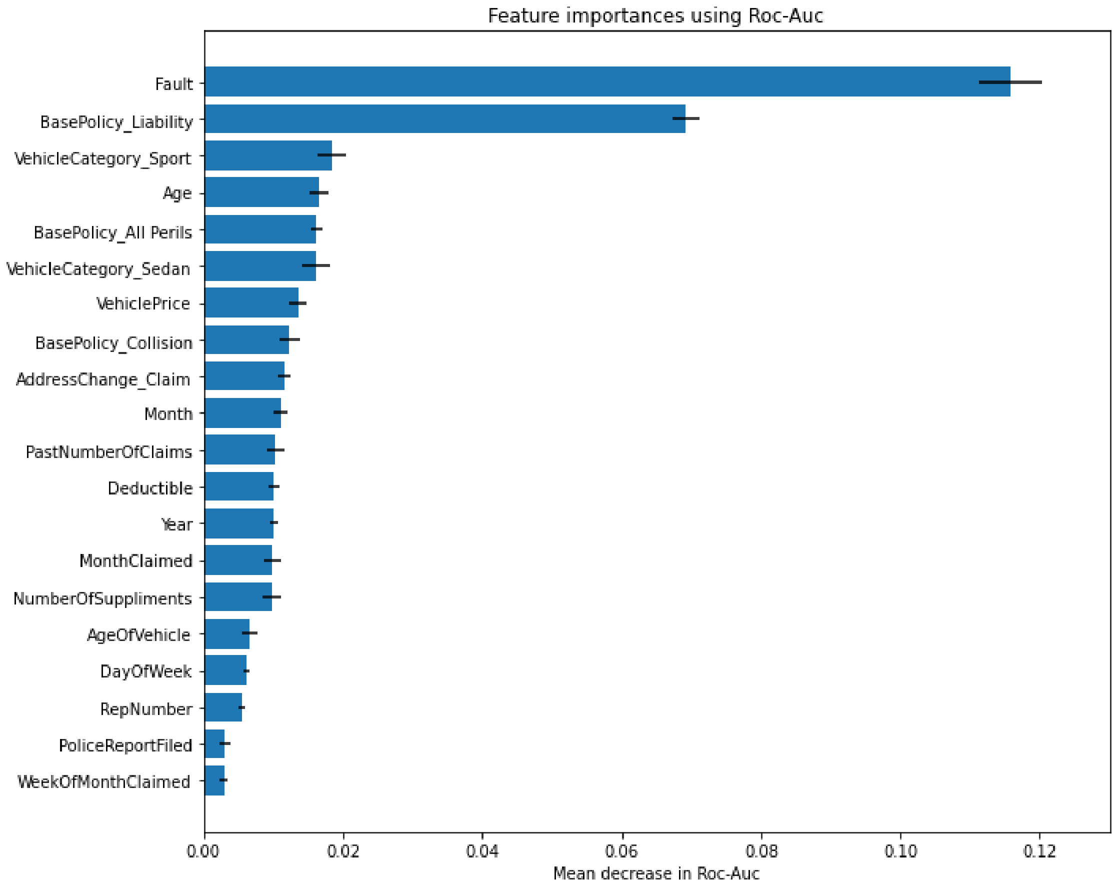
```
[ ]: # RF-SMOTH-X5
     feature_names = [f"feature {i}" for i in range(X_res1.shape[1])]
     results = permutation_importance(rf_smoth1, X_res1, y_res1, scoring='roc_auc')
     # get importance
     importance = results.importances_mean
     std = results.importances_std
     std = pd.Series(std, index = X_res1.columns)
     forest_importances = pd.Series(importance, index=X_res1.columns)
     forest_importances = forest_importances.sort_values(ascending=False)
     std = std[forest_importances.index[0:20]]
     fig, ax = plt.subplots()
     ax.barh(forest_importances.index[0:20], forest_importances[0:20], xerr=std,␣
      ↪align='center')
     ax.invert_yaxis()  # labels read top-to-bottom
     ax.set_title("Feature importances using MDI")
     ax.set_xlabel("Mean decrease in impurity")
     plt.xlim(0,0.1)
     fig.tight_layout()
```



[ ]:
```
# RF-NearMiss-X5
feature_names = [f"feature {i}" for i in range(X_under1.shape[1])]
results = permutation_importance(rf_nearmiss1, X_under1, y_under1,␣
 ↪scoring='roc_auc')
# get importance
importance = results.importances_mean
std = results.importances_std
std = pd.Series(std, index = X_under1.columns)
forest_importances = pd.Series(importance, index=X_under1.columns)
forest_importances = forest_importances.sort_values(ascending=False)
std = std[forest_importances.index[0:20]]
fig, ax = plt.subplots()
ax.barh(forest_importances.index[0:20], forest_importances[0:20], xerr=std,␣
 ↪align='center')
ax.invert_yaxis()  # labels read top-to-bottom
ax.set_title("Feature importances using MDI")
ax.set_xlabel("Mean decrease in impurity")
plt.xlim(0,0.1)
fig.tight_layout()
```



[ ]:
```python
# RF-CS-X5
feature_names = [f"feature {i}" for i in range(X_train1.shape[1])]
results = permutation_importance(rf_cs1, X_train1, y_train1, scoring='roc_auc')
# get importance
importance = results.importances_mean
std = results.importances_std
std = pd.Series(std, index = X_train1.columns)
forest_importances = pd.Series(importance, index=X_train1.columns)
forest_importances = forest_importances.sort_values(ascending=False)
std = std[forest_importances.index[0:20]]
fig, ax = plt.subplots(figsize=(10,8))
ax.barh(forest_importances.index[0:20], forest_importances[0:20], xerr=std,␣
    ↪align='center')
ax.invert_yaxis()  # labels read top-to-bottom
ax.set_title("Feature importances using Roc-Auc")
ax.set_xlabel("Mean decrease in Roc-Auc")
plt.xlim(0,0.13)
fig.tight_layout()
```



NB- SMOTH

```
[ ]: random.seed(2022)
     mutual_filter = SelectKBest(mutual_info_classif, k=11)
     naiv_b = BernoulliNB(alpha=1)
     nb_smoth1 = make_pipeline(mutual_filter, StandardScaler(), naiv_b)
     nb_smoth1.fit(X_res1, y_res1)
     nb_smoth1_pred = nb_smoth1.predict(X_valid1)
     mutual_filter.feature_names_in_[mutual_filter.get_support(indices=True)]
```

```
[ ]: array(['WeekOfMonth', 'DayOfWeekClaimed', 'MonthClaimed', 'Age', 'Fault',
            'VehicleCategory_Sport', 'PastNumberOfClaims',
            'NumberOfSuppliments', 'NumberOfCars', 'Year',
            'BasePolicy_Liability'], dtype=object)
```

NB Smoth Compositon FS

```
[ ]: comp_names1 = decomp_fs_names(X_train1, y_train1 , k = 16, m = 11)
     naiv_b = BernoulliNB(alpha=1)
     nb_comp_smoth1 = make_pipeline( StandardScaler(), naiv_b)
```

```
nb_comp_smoth1.fit(X_res1[comp_names1], y_res1)
nb_comp_smoth1_pred = nb_comp_smoth1.predict(X_valid1[comp_names1])
```

NB- NearMiss

[ ]:
```
np.random.seed(2022)
mutual_filter = SelectKBest(mutual_info_classif, k=11)
naiv_b = BernoulliNB(alpha=1)
nb_nearmiss1 = make_pipeline(mutual_filter, StandardScaler(), naiv_b)
nb_nearmiss1.fit(X_under1, y_under1)
nb_nearmiss1_pred = nb_nearmiss1.predict(X_valid1)
mutual_filter.feature_names_in_[mutual_filter.get_support(indices=True)]
```

[ ]:
```
array(['DayOfWeek', 'Age', 'Fault', 'VehicleCategory_Sedan',
       'VehiclePrice', 'Deductible', 'PastNumberOfClaims', 'AgeOfVehicle',
       'NumberOfSuppliments', 'AddressChange_Claim',
       'BasePolicy_Liability'], dtype=object)
```

NB NearMiss Compositon FS

[ ]:
```
np.random.seed(2022)
naiv_b = BernoulliNB(alpha=1 , )
nb_comp_nearmiss1 = make_pipeline( StandardScaler(), naiv_b)
nb_comp_nearmiss1.fit(X_under1[comp_names1], y_under1)
nb_comp_nearmiss1_pred = nb_comp_nearmiss1.predict(X_valid1[comp_names1])
```

NB- CS

[ ]:
```
train_weights1 = sklearn.utils.compute_sample_weight({0: 1, 1: 16}, y_train1)
```

[ ]:
```
np.random.seed(2022)
mutual_filter = SelectKBest(mutual_info_classif, k=11)
naiv_b = BernoulliNB(alpha=1 )
nb_cs1 = make_pipeline(mutual_filter, StandardScaler(),naiv_b)
kwargs1 = {nb_cs1.steps[-1][0] + '__sample_weight': train_weights1}
nb_cs1.fit(X_train1, y_train1, **kwargs1)
nb_cs1_pred = nb_cs1.predict(X_valid1)
mutual_filter.feature_names_in_[mutual_filter.get_support(indices=True)]
```

[ ]:
```
array(['WeekOfMonthClaimed', 'Sex', 'Fault', 'VehicleCategory_Sport',
       'VehicleCategory_Sedan', 'Deductible', 'Days_Policy_Claim',
       'PastNumberOfClaims', 'WitnessPresent', 'BasePolicy_Liability',
       'BasePolicy_All Perils'], dtype=object)
```

NB CS with compostion

[ ]:
```
np.random.seed(2022)
comp_names1 = decomp_fs_names(X_train1, y_train1 , k = 16, m = 11)
naiv_b = BernoulliNB(alpha=1)
```

```
nb_comp_cs1 = make_pipeline(StandardScaler(),naiv_b)
kwargs1 = {nb_comp_cs1.steps[-1][0] + '__sample_weight': train_weights1}
nb_comp_cs1.fit(X_train1[comp_names1], y_train1, **kwargs1)
nb_comp_cs1_pred = nb_comp_cs1.predict(X_valid1[comp_names1])
```

SVM - SMOTH

[ ]:
```
np.random.seed(2022)
mutual_filter = SelectKBest(mutual_info_classif, k=12)
svm = SVC(gamma='auto', random_state= 2022)
svm_smoth1 = make_pipeline(mutual_filter, StandardScaler(), svm)
svm_smoth1.fit(X_res1, y_res1)
svm_smoth1_pred = svm_smoth1.predict(X_valid1)
mutual_filter.feature_names_in_[mutual_filter.get_support(indices=True)]
```

[ ]:
```
array(['WeekOfMonth', 'DayOfWeekClaimed', 'MonthClaimed',
       'WeekOfMonthClaimed', 'Age', 'Fault', 'VehicleCategory_Sport',
       'PastNumberOfClaims', 'NumberOfSuppliments', 'AddressChange_Claim',
       'NumberOfCars', 'BasePolicy_Liability'], dtype=object)
```

SVM - SMOTH COMPOSITION

[ ]:
```
np.random.seed(2022)
comp_names1 = decomp_fs_names(X_train1, y_train1 , k = 16, m = 12)
svm = SVC(gamma='auto', random_state= 2022)
svm_comp_smoth1 = make_pipeline(StandardScaler(), svm)
svm_comp_smoth1.fit(X_res1[comp_names1], y_res1)
svm_comp_smoth1_pred = svm_comp_smoth1.predict(X_valid1[comp_names1])
```

SVM - NearMiss

[ ]:
```
np.random.seed(2022)
mutual_filter = SelectKBest(mutual_info_classif, k=14)
svm = SVC(gamma='auto', random_state= 2022)
svm_nearmiss1 = make_pipeline(mutual_filter, StandardScaler(), svm)
svm_nearmiss1.fit(X_under1, y_under1)
svm_nearmiss1_pred = svm_nearmiss1.predict(X_valid1)
mutual_filter.feature_names_in_[mutual_filter.get_support(indices=True)]
```

[ ]:
```
array(['WeekOfMonth', 'DayOfWeek', 'MaritalStatus_Single', 'Age', 'Fault',
       'VehicleCategory_Sedan', 'VehiclePrice', 'Deductible',
       'DriverRating', 'PastNumberOfClaims', 'AgeOfVehicle',
       'NumberOfSuppliments', 'AddressChange_Claim',
       'BasePolicy_Liability'], dtype=object)
```

SVM NEARMISS COMPOSITION

[ ]:
```
comp_names1 = decomp_fs_names(X_train1, y_train1 , k = 16, m = 14)
svm = SVC(gamma='auto', random_state= 2022)
```

```
svm_comp_nearmiss1 = make_pipeline( StandardScaler(), svm)
svm_comp_nearmiss1.fit(X_under1[comp_names1], y_under1)
svm_comp_nearmiss1_pred = svm_comp_nearmiss1.predict(X_valid1[comp_names1])
```

6.0.1 SVM - CS

```
[ ]: np.random.seed(2022)
mutual_filter = SelectKBest(mutual_info_classif, k=14)
svm = SVC(gamma='auto', class_weight = {0:1 , 1:16}, random_state= 2022)
svm_cs1 = make_pipeline(mutual_filter, StandardScaler(), svm)
svm_cs1.fit(X_train1, y_train1)
svm_cs1_pred = svm_cs1.predict(X_valid1)
mutual_filter.feature_names_in_[mutual_filter.get_support(indices=True)]
```

```
[ ]: array(['WeekOfMonthClaimed', 'Sex', 'Age', 'Fault',
       'VehicleCategory_Sport', 'VehicleCategory_Sedan', 'Deductible',
       'Days_Policy_Accident', 'Days_Policy_Claim', 'PastNumberOfClaims',
       'WitnessPresent', 'BasePolicy_Liability', 'BasePolicy_Collision',
       'BasePolicy_All Perils'], dtype=object)
```

SVM - CS COMPOSITION

```
[ ]: comp_names1 = decomp_fs_names(X_train1, y_train1 , k = 16, m = 14)
svm = SVC(gamma='auto',  class_weight = {0:1 , 1:16}, random_state= 2022)
svm_comp_cs1 = make_pipeline(StandardScaler(), svm)
svm_comp_cs1.fit(X_res1[comp_names1], y_res1)
svm_comp_cs1_pred = svm_comp_cs1.predict(X_valid1[comp_names1])
```

7 Part 5 - Evaluating preformance, model selection

```
[ ]: ## define function for evaluating by few metrics and CM
def eval_pref(pred, y, classifier, model):
    precision, recall, thresholds = precision_recall_curve(y, pred)
    auc1 = auc(recall, precision)
    print(model,':')
    print('Accuarcy: ', round(accuracy_score(y,pred),4)*100, '%')
    print('Roc_Auc: ',round(roc_auc_score(y,pred),4)*100, '%')
    print('G-mean: ', round(geometric_mean_score(y,pred),4)*100, '%')
    print('F1-score: ',round(f1_score(y,pred),4)*100, '%')
    print('F2-score: ', round(fbeta_score(y, pred, beta=2),4)*100, '%')
    print('AUC-PR: ', round(auc1, 4)*100, '%')
    cm = confusion_matrix(y, pred, labels=classifier.classes_)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                                  display_labels=['Not Fraud', 'Fraud'])
    disp.plot()
```

```
[ ]: eval_pref(rf_pred, y_valid1, rf, 'RF')
```

```
RF :
Accuarcy:  94.49 %
Roc_Auc:  50.0 %
G-mean:  0.0 %
F1-score:  0.0 %
F2-score:  0.0 %
AUC-PR:  52.76 %
```


```
[ ]: eval_pref(rf_smoth1_pred, y_valid1, rf_smoth1, 'RF-SMOTH1')
```

```
RF-SMOTH1 :
Accuarcy:  75.11 %
Roc_Auc:  67.44 %
G-mean:  66.89 %
F1-score:  20.66 %
F2-score:  33.83 %
AUC-PR:  36.809999999999995 %
```


```
[ ]: eval_pref(rf_nearmiss1_pred, y_valid1, rf_nearmiss1, 'RF-NearMiss1')
```

```
RF-NearMiss1 :
Accuarcy:  52.22 %
Roc_Auc:  66.13 %
G-mean:  64.25999999999999 %
F1-score:  15.870000000000001 %
F2-score:  30.73 %
AUC-PR:  45.78 %
```

[ ]:
```
eval_pref(rf_cs1_pred, y_valid1, rf_cs1, 'RF-CS')
```

```
RF-CS :
Accuarcy:  59.64 %
Roc_Auc:  75.6 %
G-mean:  73.44000000000001 %
F1-score:  20.349999999999998 %
F2-score:  38.35 %
AUC-PR:  52.65 %
```


```
[ ]: eval_pref(svm_smoth1_pred, y_valid1, svm_smoth1, 'SVM-SMOTH1')
```

```
SVM-SMOTH1 :
Accuarcy:  72.92999999999999 %
Roc_Auc:  67.4 %
G-mean:  67.11 %
F1-score:  19.939999999999998 %
F2-score:  33.48 %
AUC-PR:  37.61 %
```


```
[ ]: eval_pref(svm_comp_smoth1_pred, y_valid1, svm_comp_smoth1, 'SVM-COMP-SMOTH1')
```

```
SVM-COMP-SMOTH1 :
Accuarcy:  78.35 %
Roc_Auc:  55.31 %
G-mean:  48.870000000000005 %
F1-score:  13.020000000000001 %
F2-score:  19.56 %
AUC-PR:  20.830000000000002 %
```

[ ]:
```
eval_pref(svm_nearmiss1_pred, y_valid1, svm_nearmiss1, 'SVM-NearMiss1')
```

```
SVM-NearMiss1 :
Accuarcy:  45.96 %
Roc_Auc:  60.33 %
G-mean:  58.13 %
F1-score:  13.489999999999998 %
F2-score:  26.669999999999998 %
AUC-PR:  42.58 %
```


```
[ ]: eval_pref(svm_comp_nearmiss1_pred, y_valid1, svm_comp_nearmiss1,␣
      ↪'SVM-COMP-NearMiss1')
```

```
SVM-COMP-NearMiss1 :
Accuarcy:  53.74 %
Roc_Auc:  53.92 %
G-mean:  53.92 %
F1-score:  11.42 %
F2-score:  21.69 %
AUC-PR:  31.52 %
```

[ ]:
```
eval_pref(svm_cs1_pred, y_valid1, svm_cs1, 'SVM-CS1')
```

```
SVM-CS1 :
Accuarcy:  60.39 %
Roc_Auc:  74.61 %
G-mean:  72.88 %
F1-score:  20.13 %
F2-score:  37.75 %
AUC-PR:  51.22 %
```

[ ]:
```
eval_pref(svm_comp_cs1_pred, y_valid1, svm_comp_cs1, 'SVM-Comp-CS1')
```

```
SVM-Comp-CS1 :
Accuarcy:  41.75 %
Roc_Auc:  63.36000000000001 %
G-mean:  58.52 %
F1-score:  14.219999999999999 %
F2-score:  28.599999999999998 %
AUC-PR:  48.03 %
```


```
[ ]: eval_pref(nb_smoth1_pred, y_valid1, nb_smoth1, 'NB-SMOTH1')
```

```
NB-SMOTH1 :
Accuarcy:  63.4 %
Roc_Auc:  65.12 %
G-mean:  65.10000000000001 %
F1-score:  16.8 %
F2-score:  30.53 %
AUC-PR:  39.24 %
```

[ ]:
```
eval_pref(nb_comp_smoth1_pred, y_valid1, nb_comp_smoth1, 'NB-COMP-SMOTH1')
```

```
NB-COMP-SMOTH1 :
Accuarcy:  40.42 %
Roc_Auc:  63.21 %
G-mean:  57.79 %
F1-score:  14.11 %
F2-score:  28.49 %
AUC-PR:  48.55 %
```


```
[ ]: eval_pref(nb_nearmiss1_pred, y_valid1, nb_nearmiss1, 'NB--nearmiss1')
```

```
NB--nearmiss1 :
Accuarcy:  56.730000000000004 %
Roc_Auc:  65.19 %
G-mean:  64.49000000000001 %
F1-score:  15.98 %
F2-score:  30.25 %
AUC-PR:  42.52 %
```

[ ]:
```
eval_pref(nb_comp_nearmiss1_pred, y_valid1, nb_comp_nearmiss1,␣
 ↪'NB-COMP-nearmiss1')
```

```
NB-COMP-nearmiss1 :
Accuarcy:  52.349999999999994 %
Roc_Auc:  61.77 %
G-mean:  60.85 %
F1-score:  14.34 %
F2-score:  27.63 %
AUC-PR:  40.92 %
```


```
[ ]: eval_pref(nb_cs1_pred, y_valid1, nb_cs1, 'NB-CS1')
```

```
NB-CS1 :
Accuarcy:  56.14 %
Roc_Auc:  72.36 %
G-mean:  70.03 %
F1-score:  18.54 %
F2-score:  35.47 %
AUC-PR:  50.72 %
```


```
[ ]: eval_pref(nb_comp_cs1_pred,y_valid1 , nb_comp_cs1, 'NB-Comp CS')
```

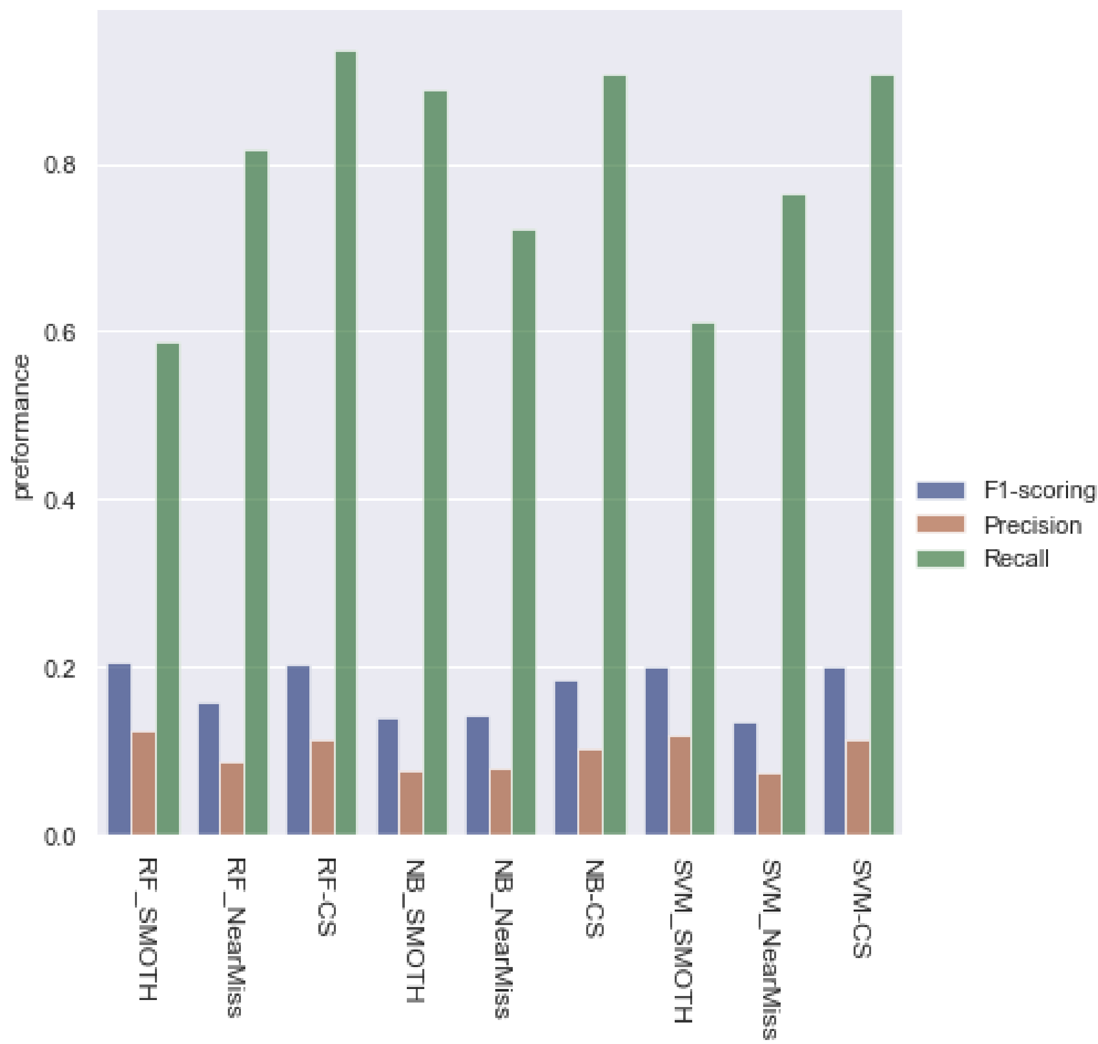
```
NB-Comp CS :
Accuarcy:  39.48 %
Roc_Auc:  63.82 %
G-mean:  57.66 %
F1-score:  14.24 %
F2-score:  28.84 %
AUC-PR:  49.69 %
```

comparison by bar plots

[ ]:
```
# presicion, recall, F1-score - without decompasition feature selection
dict_pre = {'Model':␣
↪['RF_SMOTH','RF_NearMiss','RF-CS','NB_SMOTH','NB_NearMiss','NB-CS','SVM_SMOTH','SVM_NearMis
↪'metric': np.repeat('Precision',9) ,'scoring': [] }
dict_rec = {'Model':␣
↪['RF_SMOTH','RF_NearMiss','RF-CS','NB_SMOTH','NB_NearMiss','NB-CS','SVM_SMOTH','SVM_NearMis
↪'metric': np.repeat('Recall',9), 'scoring': [] }
dict_f1 = {'Model':␣
↪['RF_SMOTH','RF_NearMiss','RF-CS','NB_SMOTH','NB_NearMiss','NB-CS','SVM_SMOTH','SVM_NearMis
↪'metric': np.repeat('F1-scoring',9), 'scoring': []  }
classifers_list = [rf_smoth1,rf_nearmiss1,rf_cs1,␣
↪nb_comp_smoth1,nb_comp_nearmiss1,nb_comp_cs1,svm_smoth1,svm_nearmiss1,svm_cs1]
pred_list = [rf_smoth1_pred,rf_nearmiss1_pred,rf_cs1_pred,␣
↪nb_comp_smoth1_pred,nb_comp_nearmiss1_pred,nb_cs1_pred,svm_smoth1_pred,svm_nearmiss1_pred,s
dict_f1['scoring'] =  [f1_score(y_valid1,pred) for pred in pred_list]
dict_pre['scoring'] = [precision_score(y_valid1,pred) for pred in pred_list]
dict_rec['scoring'] = [recall_score(y_valid1, pred) for pred in pred_list]

models_results = pd.concat([pd.DataFrame.from_dict(dict_f1),pd.DataFrame.
↪from_dict(dict_pre),pd.DataFrame.from_dict(dict_rec)])

sns.set_theme()
```

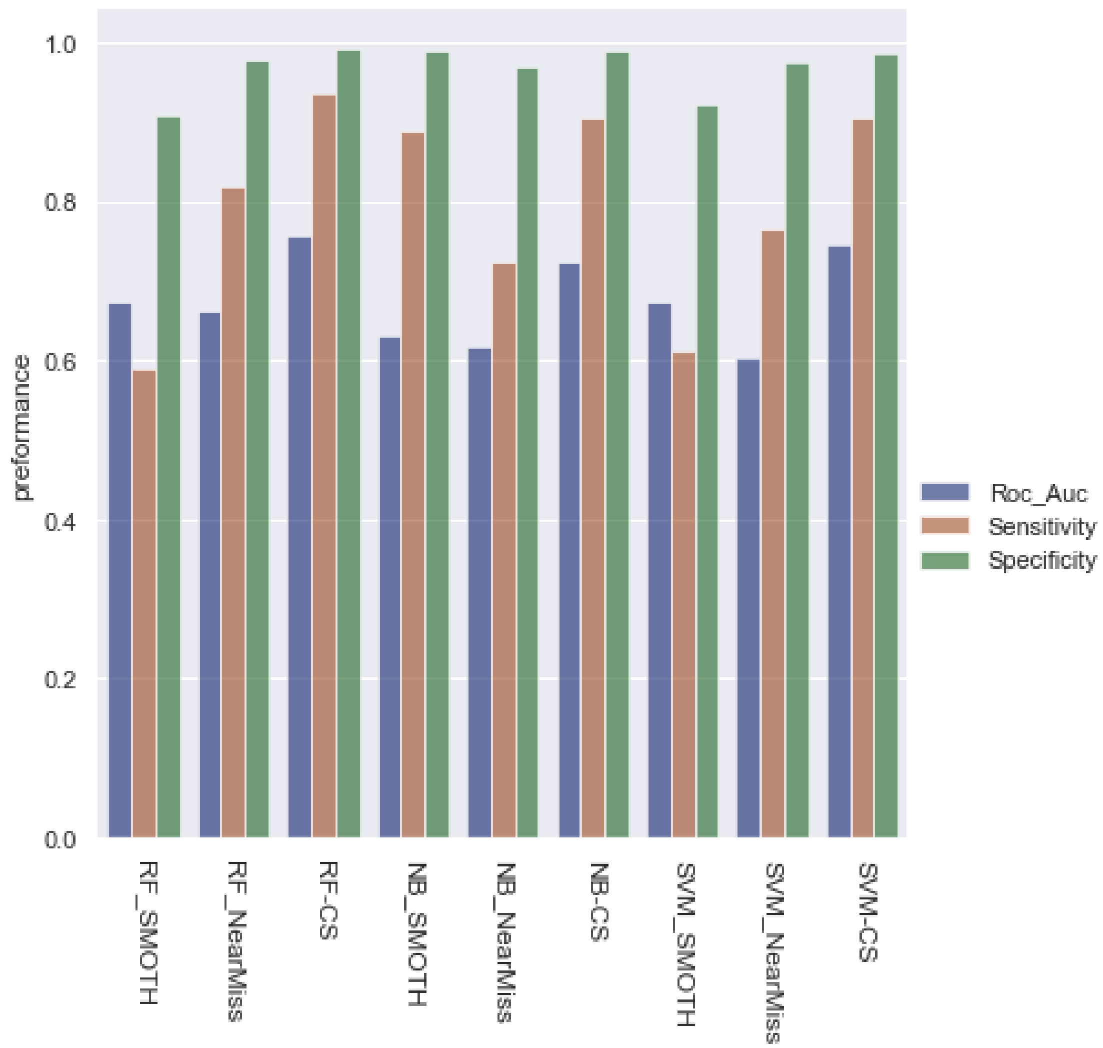
```
# Draw a nested barplot by models and scoring
g = sns.catplot(
    data=models_results, kind="bar",
    x="Model", y="scoring", hue="metric",
    ci="sd", palette="dark", alpha=.6, height=6
)
g.despine(left=True)
g.set_axis_labels("", "preformance")
g.legend.set_title("")
g.set_xticklabels(rotation = -90, size = 12)
```

[ ]: <seaborn.axisgrid.FacetGrid at 0x7f0a5c0f9810>



```
[ ]: # senstivity, specificity, Roc Auc
dict_roc = {'Model':␣
 ↪['RF_SMOTH','RF_NearMiss','RF-CS','NB_SMOTH','NB_NearMiss','NB-CS','SVM_SMOTH','SVM_NearMis
 ↪'metric': np.repeat('Roc_Auc',9) ,'scoring': [] }
dict_sens = {'Model':␣
 ↪['RF_SMOTH','RF_NearMiss','RF-CS','NB_SMOTH','NB_NearMiss','NB-CS','SVM_SMOTH','SVM_NearMis
 ↪'metric': np.repeat('Sensitivity',9), 'scoring': [] }
dict_spec = {'Model':␣
 ↪['RF_SMOTH','RF_NearMiss','RF-CS','NB_SMOTH','NB_NearMiss','NB-CS','SVM_SMOTH','SVM_NearMis
 ↪'metric': np.repeat('Specificity',9), 'scoring': []  }
pred_list = [rf_smoth1_pred,rf_nearmiss1_pred,rf_cs1_pred,␣
 ↪nb_comp_smoth1_pred,nb_comp_nearmiss1_pred,nb_cs1_pred,svm_smoth1_pred,svm_nearmiss1_pred,s


dict_roc['scoring'] =  [roc_auc_score(y_valid1,pred) for pred in pred_list]
dict_spec['scoring'] = [confusion_matrix(y_valid1, pred).ravel()[1]/
 ↪(confusion_matrix(y_valid1, pred).ravel()[1]+confusion_matrix(y_valid1,␣
 ↪pred).ravel()[2]) for pred in pred_list]
dict_sens['scoring'] = [recall_score(y_valid1, pred) for pred in pred_list]

models_results = pd.concat([pd.DataFrame.from_dict(dict_roc),pd.DataFrame.
 ↪from_dict(dict_sens),pd.DataFrame.from_dict(dict_spec)])

sns.set_theme()


# Draw a nested barplot by models and scoring
g = sns.catplot(
    data=models_results, kind="bar",
    x="Model", y="scoring", hue="metric",
    ci="sd", palette="dark", alpha=.6, height=6
)
g.despine(left=True)
g.set_axis_labels("", "preformance")
g.legend.set_title("")
g.set_xticklabels(rotation = -90, size = 12)
```

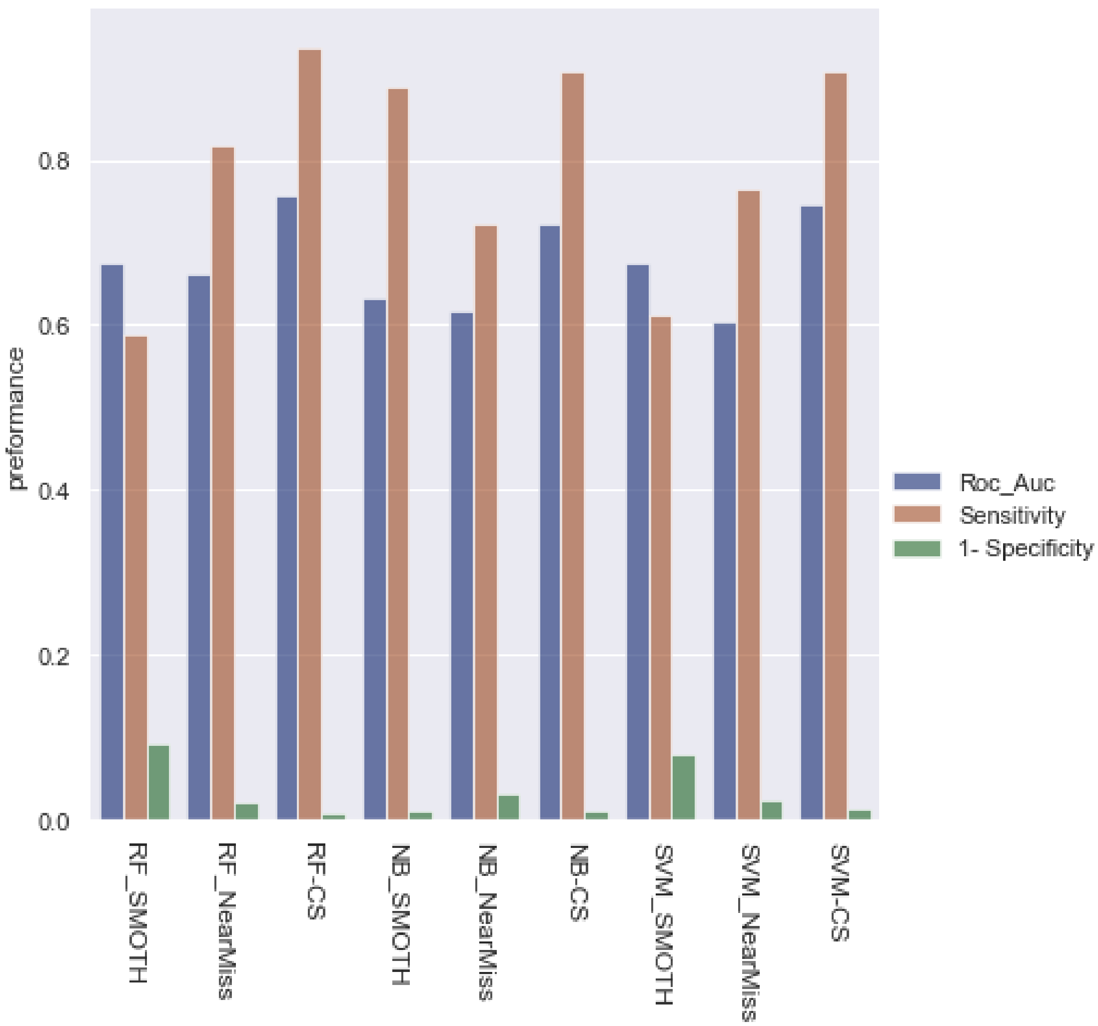
```
[ ]: <seaborn.axisgrid.FacetGrid at 0x7f0a5c1211e0>
```



[ ]:
```
# Sensitivity, 1- Specificity, Recall
dict_roc = {'Model':␣
↪['RF_SMOTH','RF_NearMiss','RF-CS','NB_SMOTH','NB_NearMiss','NB-CS','SVM_SMOTH','SVM_NearMis
↪'metric': np.repeat('Roc_Auc',9) ,'scoring': [] }
dict_sens = {'Model':␣
↪['RF_SMOTH','RF_NearMiss','RF-CS','NB_SMOTH','NB_NearMiss','NB-CS','SVM_SMOTH','SVM_NearMis
↪'metric': np.repeat('Sensitivity',9), 'scoring': [] }
dict_1spec = {'Model':␣
↪['RF_SMOTH','RF_NearMiss','RF-CS','NB_SMOTH','NB_NearMiss','NB-CS','SVM_SMOTH','SVM_NearMis
↪'metric': np.repeat('1- Specificity',9), 'scoring': []  }
pred_list = [rf_smoth1_pred,rf_nearmiss1_pred,rf_cs1_pred,␣
↪nb_comp_smoth1_pred,nb_comp_nearmiss1_pred,nb_cs1_pred,svm_smoth1_pred,svm_nearmiss1_pred,s
```

```
dict_roc['scoring'] =  [roc_auc_score(y_valid1,pred) for pred in pred_list]
dict_1spec['scoring'] = [1-(confusion_matrix(y_valid1, pred).ravel()[1]/
 ↪(confusion_matrix(y_valid1, pred).ravel()[1]+confusion_matrix(y_valid1,␣
 ↪pred).ravel()[2])) for pred in pred_list]
dict_sens['scoring'] = [recall_score(y_valid1, pred) for pred in pred_list]

models_results = pd.concat([pd.DataFrame.from_dict(dict_roc),pd.DataFrame.
 ↪from_dict(dict_sens),pd.DataFrame.from_dict(dict_1spec)])

sns.set_theme()


# Draw a nested barplot by models and scoring
g = sns.catplot(
    data=models_results, kind="bar",
    x="Model", y="scoring", hue="metric",
    ci="sd", palette="dark", alpha=.6, height=6
)
g.despine(left=True)
g.set_axis_labels("", "preformance")
g.legend.set_title("")
g.set_xticklabels(rotation = -90, size = 12)
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x7f0a611c7730>
```



we can see RF-CS has the best preformance

8 part 6 - Test preformance of the chosen model

[ ]:
```
# what Elad did before with train data only
# rf_cs1_pred_test = rf_cs1.predict(X_test1)
# eval_pref(rf_cs1_pred_test,y_test1 , rf_cs1, 'chosen model')
```

We chose our model, we can use the combined data from train and val to get better test results (a technique we saw in the introductory kaggle course)

[ ]:
```
X_full_train = pd.concat([X_train1,X_valid1])
y_full_train = pd.concat([y_train1, y_valid1])
rf_cs_final = RandomForestClassifier(n_estimators = 100, max_features ='sqrt',␣
 ↪random_state = 2022, max_depth =5,
```

```
                                          class_weight={0: 1 ,1:16})

rf_cs_final.fit(X_full_train, y_full_train)
rf_cs_final_pred = rf_cs_final.predict(X_test1)
eval_pref(rf_cs_final_pred,y_test1 , rf_cs_final, 'chosen full model')
```

```
chosen full model :
Accuarcy:  62.519999999999996 %
Roc_Auc:  79.57 %
G-mean:  77.18 %
F1-score:  23.75 %
F2-score:  43.65 %
AUC-PR:  56.230000000000004 %
```