Udacity Machine Learning Nanodegree
Fraud Detection on Insurance Claims

Venghatesh S
Nov 10th 2018

Introduction

Insurance fraud is any act committed to defraud an insurance process. False insurance claims are insurance claims filed with an intent to defraud the insurance provider. Insurance fraud is deliberately undetectable, unlike visible crimes such as robbery or murder.

The Coalition Against Insurance Fraud estimates that in 2006 a total of about $80 billion was lost in the United States due to insurance fraud. According to estimates by the Insurance Information Institute, insurance fraud accounts for about 10 percent of the property/casualty insurance industry's incurred losses and loss adjustment expenses. The National Health Care Anti-Fraud Association estimates that 3% of the health care industry's expenditures in the United States are due to fraudulent activities, amounting to a cost of about $51 billion. Other estimates attribute as much as 10% of the total healthcare spending in the United States to fraud—about $115 billion annually. Another study of all types of fraud committed in the United States insurance institutions (property-and-casualty, business liability, healthcare, social security, etc.)put the true cost at 33% to 38% of the total cash flow through the system. In the United Kingdom, the Insurance Fraud Bureau estimates that the loss due to insurance fraud in the United Kingdom is about £1.5 billion ($3.08 billion), causing a 5% increase in insurance premiums. The Insurance Bureau of Canada estimates that personal injury fraud in Canada costs about C$500 million annually. India forensic Center of Studies estimates that Insurance frauds in India costs about $6.25 billion annually.

Hence it is highly imperative for the insurance industry to develop solid capabilities that can help identify potential frauds with a high degree of accuracy, so that other claims can be cleared rapidly while identified cases can be scrutinized in detail.

Citations :

https://www.insurancenexus.com/fraud/role-data-and-analytics-insurance-fraud-detection

https://www.researchgate.net/publication/291833022_Analytics_for_Insurance_Fraud_Detection_An_Empirical_Study

https://acadpubl.eu/jsi/2017-116-13-22/articles/21/80.pdf

Dataset could be found in the below location
https://raw.githubusercontent.com/jodb/sparkTestData/master/insurance_claims.csv

In this capstone project we will look at creating a predictive model that predicts whether a auto insurance claim is fraudulent or not.

Problem Statement

The current approach for Auto insurance fraud detection is based on a Heuristic approach around fraud indicators. Various scenario rules would be framed and these rules would determine along with the value of the claim to understand whether a claim needs to be sent for investigation. The challenge with this approach is that it is heavily manual driven and has the following limitations

- Operates with a limited set of known parameters based on heuristic knowledge
- Not data driven

Venghatesh S
Nov 10th 2018

- Periodic manual re-calibration based on investigation outputs

This has led to auto insurance companies to explore ways of using machine learning algorithm which can look at the data patterns without judgement on relevance on data elements. Based on identified frauds, machine learning algorithms can develop a model through a variety of algorithmic techniques.

The below dataset is used for analysis.
https://raw.githubusercontent.com/jodb/sparkTestData/master/insurance_claims.csv

This is a **binary classification problem** since the end goal is to understand whether the claim is fraudulent or not.

We will run a handful of machine learning algorithms to understand which features are relevant for fraud detection after doing data processing and transformation.

Metrics

Accuracy is a common metric for binary classifiers; it takes into account both true positives and true negatives with equal weight.

$$\text{accuracy} = \frac{\text{true positives (TP)} + \text{true negatives (TN)}}{\text{Dataset Size}}$$

Where Dataset Size = true positive(TP) + true Negative (TN) + false Positive (FP) + False Negative (FN)

True Positive (TP) • Reality : Fraudulent Claim • ML Predicted : Fraudulent Claim	False Positive (FP) • Reality : Genuine Claim • ML Predicted : Fraudulent Claim
False Negative (FN) • Reality : Fraudulent Claim • ML Predicted : Genuine Claim	True Negative (TN) • Reality : Genuine Claim • ML Predicted : Genuine Claim

Accuracy alone doesn't tell the full story when you're working with a **class-imbalanced data set**, like this one, where there is a significant disparity between the number of positive and negative labels. Two better metrics for evaluating class-imbalanced problems: precision and recall.

Precision predicts the what proportion of positive identifications are identified correctly .

Precision = TP/(TP+FP)

Accuracy predicts what proportion of actual positives are identified correctly.

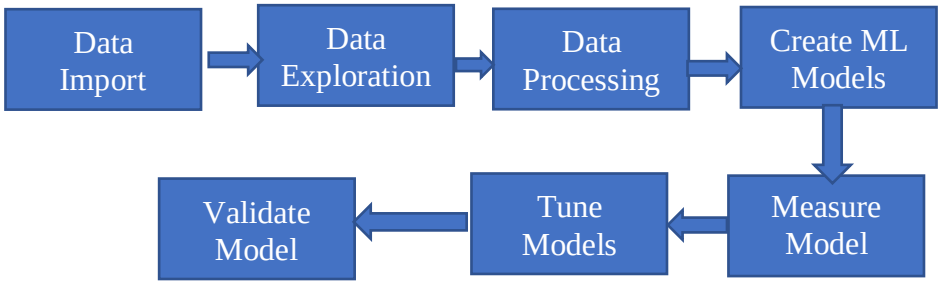
Recall = TP/(TP+FN)

To fully evaluate the effectiveness of a model, you must examine **both** precision and recall. Unfortunately, precision and recall are often in tension. That is, improving precision typically reduces recall and vice versa.

Venghatesh S
Nov 10th 2018

The best metric hence would be something which uses both recall and accuracy. That is called as the F1 Score.
The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

Problem Solving Steps



Analysis

Data Exploration

Introduction to Datasets

Number of Claims	1000
Number of attributes	400
Categorical attributes	21
Normal Claims	753
Fraudulent Claims	247
Fraudulent Incident Rate	32.8%

Detailed Description

The number of input variables is 39 and output variable 1 is

months_as_customer int64	How long the customer has been with the Insurance company ?
age int64	Age of the Policy holder
policy_number int64	Policy number
policy_bind_date string	Effective date of the policy
policy_state string	US State where the insurance policy is taken
policy_csl string	Combined Single limit of the policy

Venghatesh S
Nov 10th 2018

policy_deductable int64	Amount to be paid out by the policy holder from his pocket
policy_annual_premium float64	Annual premium amount to be borned by the policy holder
umbrella_limit int64	Limit for extra liability insurance.
insured_zip int64	Pincode of the policy holder
insured_sex string	Gender of the policy holder
insured_education_level string	Highest education level of the policy holder
insured_occupation string	Job profile
insured_hobbies string	Hobby of the policy holder
insured_relationship string	Relationship between policy holder and Beneficiary
capital-gains int64	Capital Gains
capital-loss int64	Capital Loss
incident_date string	Accident Date
incident_type string	Type of incident (Eg : Vehicle theft, single vehicle collison etc)
collision_type string	Type of Collison (Side, Rear, Front)
incident_severity string	Minor Damage, Total Loss, Major Damage
authorities_contacted string	Authorities approached to validate the claim
incident_state string	State where the incident occurred
incident_city string	City where the incident occurred
incident_location string	Exact spot where the incident occurrent
incident_hour_of_the_day int64	Time of the accident
number_of_vehicles_involved int64	Self Explantory
property_damage string	Any property damage (Yes or No or unknown)
bodily_injuries int64	No of people with body injury
witnesses int64	Number of people witness to the accident
police_report_available string	Avalability of police report (Yes or no)
total_claim_amount int64	Total Claim

Venghatesh S
Nov 10th 2018

injury_claim int64	Injury Claim Amount
property_claim int64	Property damage Claim Amount
vehicle_claim int64	Vehicle damage claim amount
auto_make string	Auto Manufacturer
auto_model string	Model name of the Auto
auto_year int64	Auto production year
fraud_reported string	Fraud (Yes or No). This is the output variable
_c39 float64	This variable should be dropped

- We have several categorical variables (String) datatypes in the dataset . Some Statistical models which uses calculation of distances (Euclidean or Mahlanobis or other measures) can handle only numerical attributes. So, these categorical attributes should be converted to numerical attributes through encoding.
- Some of the features in the dataset have got lot of distinct values and they wont be useful in prediction . Having them will increase dimensionality of the dataset with no considerable improvement in prediction due to them. This should be removed.
- Some of the data may be wrong . Eg : age of the driver may be marked as 2, which is not practically possible. But a analysis of this specific dataset shows no such anomalies.

The following are the distinct values found for each parameter (written under each column) .

months_as_customer 391	collision_type 4
age 46	incident_severity 4
policy_number 1000	authorities_contacted 5
policy_bind_date 951	incident_state 7
policy_state 3	incident_city 7
policy_csl 3	incident_location 1000
policy_deductable 3	incident_hour_of_the_day 24
policy_annual_premium 991	number_of_vehicles_involved 4
umbrella_limit 11	property_damage 3
insured_zip 995	bodily_injuries 3
insured_sex 2	witnesses 4
insured_education_level	police_report_available

Venghatesh S
Nov 10th 2018

7	3
insured_occupation 14	total_claim_amount 763
insured_hobbies 20	injury_claim 638
insured_relationship 6	property_claim 626
capital-gains 338	vehicle_claim 726
capital-loss 354	auto_make 14
incident_date 60	auto_model 39
incident_type 4	auto_year 21
	fraud_reported 2
	_c39 0

We will remove these below columns from our dataset in the Data Processing step to improve model accuracy.

- policy number (1000 distinct)
- policy bind date (951 distinct, Better year/month could be used)
- insured zip (995 distinct)
- insured location (1000 distinct)
- incident date (60 distinct, Better dayof the week could be used to understand any co-relation.)
- _c39

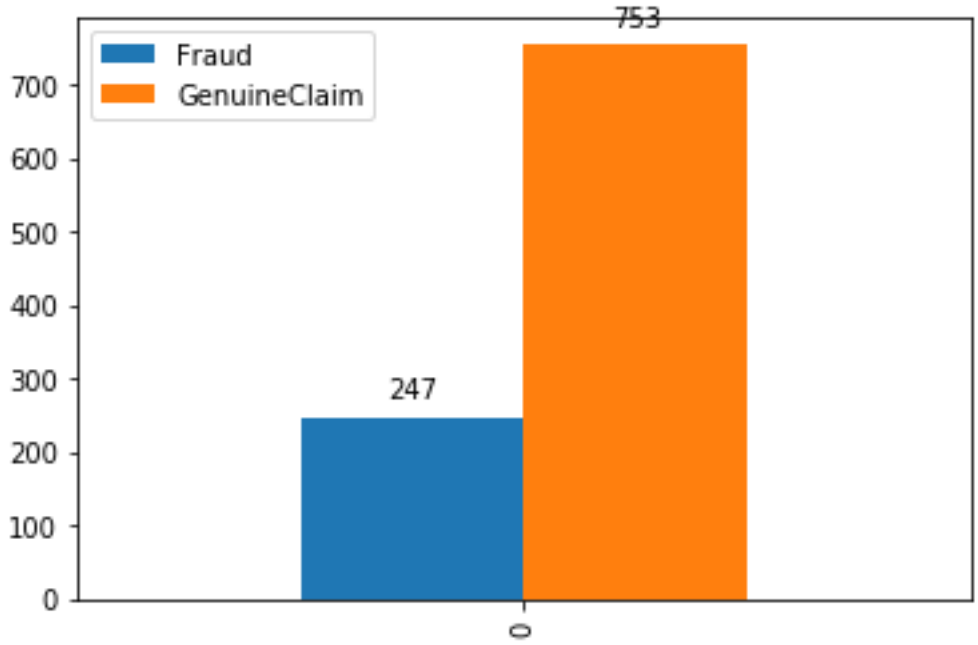
Like any other data distribution the data is skewed with genuine claims reported around 3 times more than fraudulent claims. This shows class imbalance.

Udacity Machine Learning Nanodegree

Fraud Detection on Insurance Claims

Venghatesh S

Nov 10th 2018



Data pre-processing

- Remove the columns that we have identified earlier having too many distinct categories or identifiers added like primary keys (eg : policy number) and won't be adding any value to the model.

  Columns to be dropped :

  policy_number, policy_bind_date, insured_zip, incident_location, Incident_date.

- Columns which are of datatype string needs to converted to numeric .
  Columns to be Converted to Numeric:

  'policy_state', 'policy_csl', 'insured_sex','insured_education_level', 'insured_occupation', 'insured_hobbies','insured_relationship', 'incident_date', 'week_day', 'incident_type','collision_type', 'incident_severity', 'authorities_contacted','incident_state', 'incident_city','property_damage', 'police_report_available', 'auto_make', 'auto_model','fraud_reported

- Columns to be looked as a range and a numerical value to be assigned . An example would be to look at columns like age , which can be grouped together in specific ranges which is more meaningful than taken as a single number. We can group ages beween 0 - 18, 18-30 etc so that meaningful analysis can be carried out

  Columns converted in to numeric values based on a range:

  Months_as_customer and age

Udacity Machine Learning Nanodegree

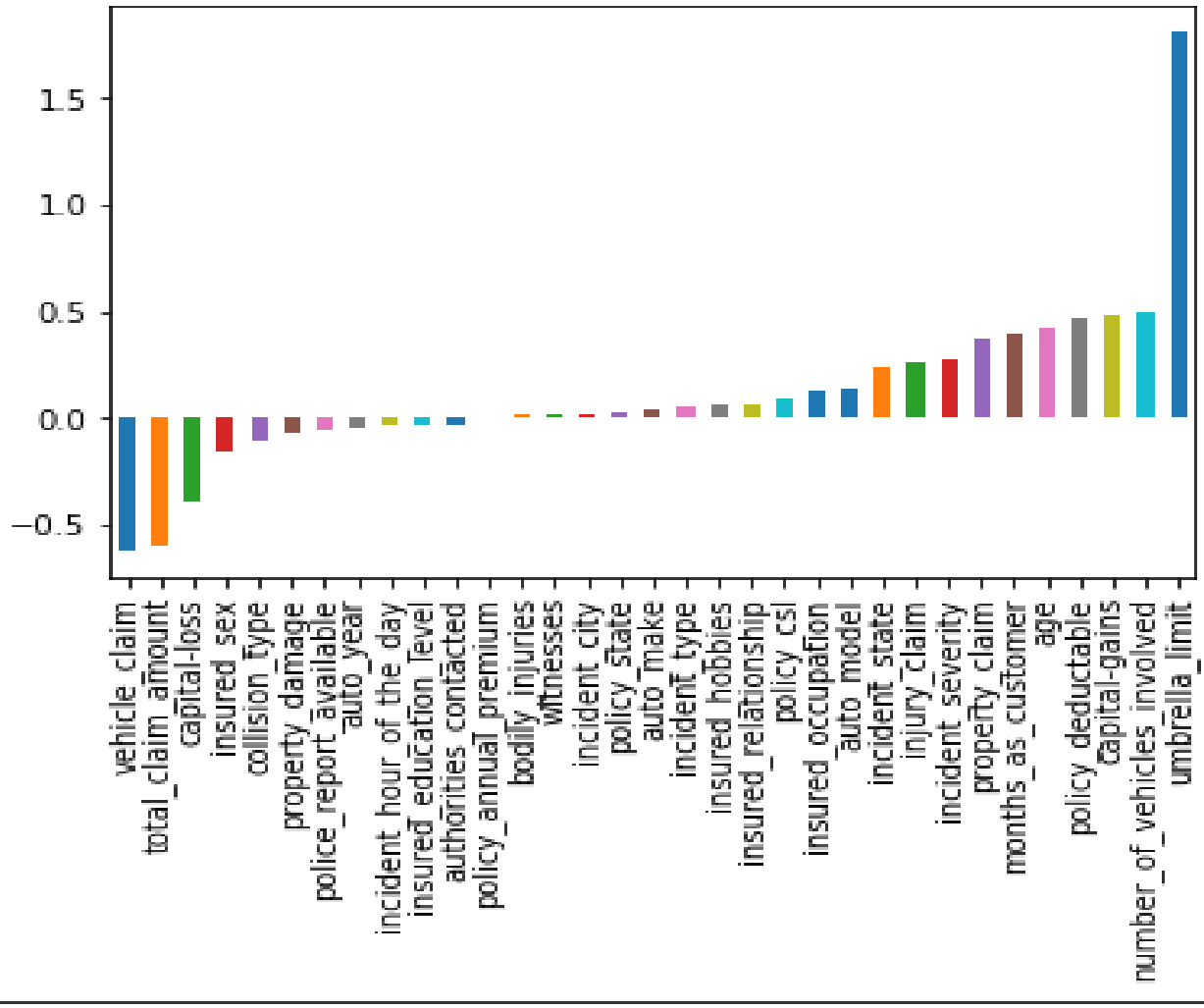
Fraud Detection on Insurance Claims

Venghatesh S

Nov 10th 2018

Exploratory Visualisation

Skewness Graph of features



In statistics, skewness is a measure of the asymmetry of the probability distribution of a random variable about its mean. In other words, skewness tells us the amount and direction of skew (departure from horizontal symmetry). The skewness value can be positive or negative, or even undefined. If skewness is 0, the data are perfectly symmetrical, although it is quite unlikely for a data like insurance claims. As a general rule of thumb:

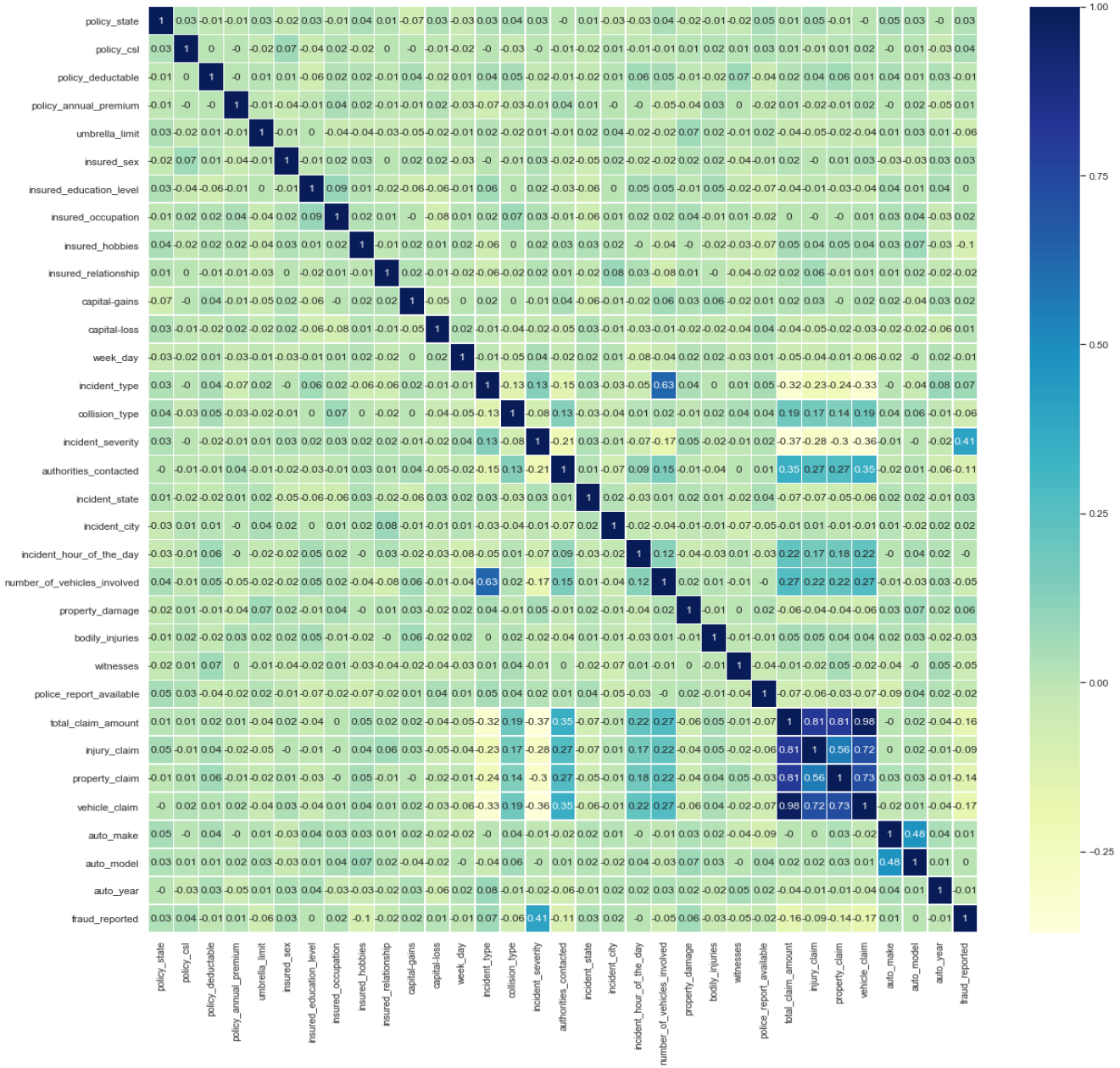
- If skewness is less than -1 or greater than 1, the distribution is highly skewed.
- If skewness is between -1 and -0.5 or between 0.5 and 1, the distribution is moderately skewed.
- If skewness is between -0.5 and 0.5, the distribution is approximately symmetric.

Seeing the above figure the only figure which seems to indicate high degree of skewness is the umbrella limit. The skewness can be reduced by applying logarithmic transformation. Logarithmic

Venghatesh S
Nov 10th 2018

transformation works for data which are numeric and positive. A look at the data for umbrella limit seems to indicate one negative value . Umbrella limit cannot be negative and its most likely a data entry error. After fixing this and applying logarithmic transformation on umbrella limit to reduce its skewness.

Co-relation is another exploratory technique we could use to understand the relationship between variables. Pearson's co-relation coeffiencient varies from -1 to +1 , with -1 indicating strong negative co-relation and + 1 indicating strong co-relation, 0 indicates no co-relation between the variables. A heatmap is the best way to understand the co-relation between variables.



Venghatesh S
Nov 10th 2018

As seen there exists a co-relation (neither too weak nor too strong) between fraudulent claims and incident severity. Incident severity will be a good feature to understand whether a given claim is fraudulent or not. Apart from that, all other features seems to be weakly co-related to fraudulent claims. Pearsons co-relation coefficient can give good results only with respect to linear relationships . Spearman and Kendall can be used for non-linear relationships . Spearman and Kendall also does not show much stronger relationships between various features.

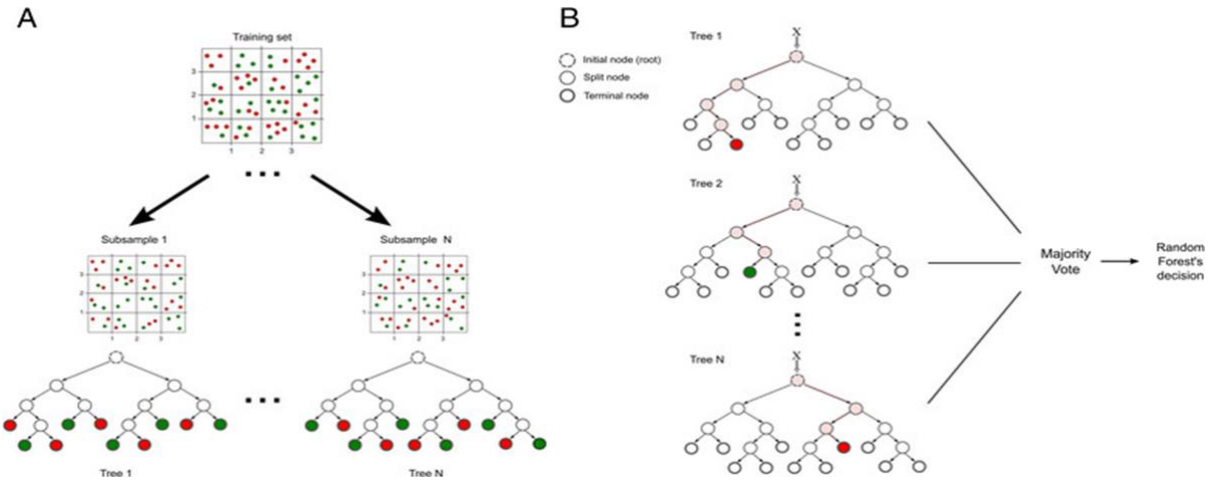
Algorithms and techniques

The given problem is a binary classification problem whether a auto insurance claim is genuine or fraudulent. The dataset contains adequate samples and number of features is not very large . A decision tree would be a best option here, but we would also evaluate SVM to understand how it compares to decision tree. One thing which we have observed using co-relation coefficients is that except for incident severity ,every other feature seems to be weekly co-related to the target variable.

We would look at 4 different models –Decision tree, Random Forest and Gradient boosting tree, SVM. By looking at their accuracy and f-score we can then decide on one model, fine tune their hyper parameters.

A decision tree is a graphical representation of all the possible solutions to a decision based on certain conditions. It's called a decision tree because it starts with a single box (or root), which then branches off into a number of solutions, just like a tree. Decision trees are helpful, not only because they are a visual representation that help you 'see' what you are thinking, but also because making a decision tree requires a systematic, documented thought process. Often, the biggest limitation of our decision making is that we can only select from the known alternatives. Decision trees help formalize the brainstorming process so we can identify more potential solutions. But Decision trees tend to overfit.

Random forests or random decision forests are an ensemble learning method
for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.Random decision forests correct for decision trees' habit of overfitting to their training set.
Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.



Udacity Machine Learning Nanodegree

Fraud Detection on Insurance Claims

Venghatesh S
Nov 10th 2018

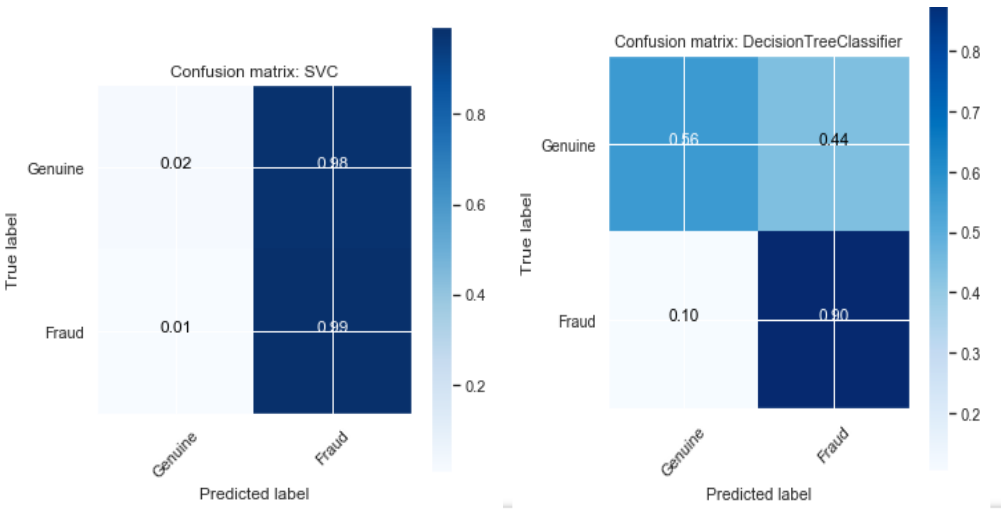
Boosting algorithms play a crucial role in dealing with bias variance trade-off. Unlike bagging algorithms, which only controls for high variance in a model, boosting controls both the aspects (bias & variance), and is considered to be more effective. Boosting is a sequential technique which works on the principle of **ensemble**. It combines a set of **weak learners** and delivers improved prediction accuracy. At any instant t, the model outcomes are weighed based on the outcomes of previous instant t-1. The outcomes predicted correctly are given a lower weight and the ones miss-classified are weighted higher. This technique is followed for a classification problem

Support Vector Machine (SVM) is a supervised machine learning algorithm and they perform classification by finding the hyper plane that differentiates between two classes well. SVM's really works in high dimensional spaces and generally useful when the number of dimensions is greater than the number of samples. As such in this dataset SVM may not be a right model compared to decision tree. But it would be prudent to run the model on the dataset to understand how the above algorithms perform.

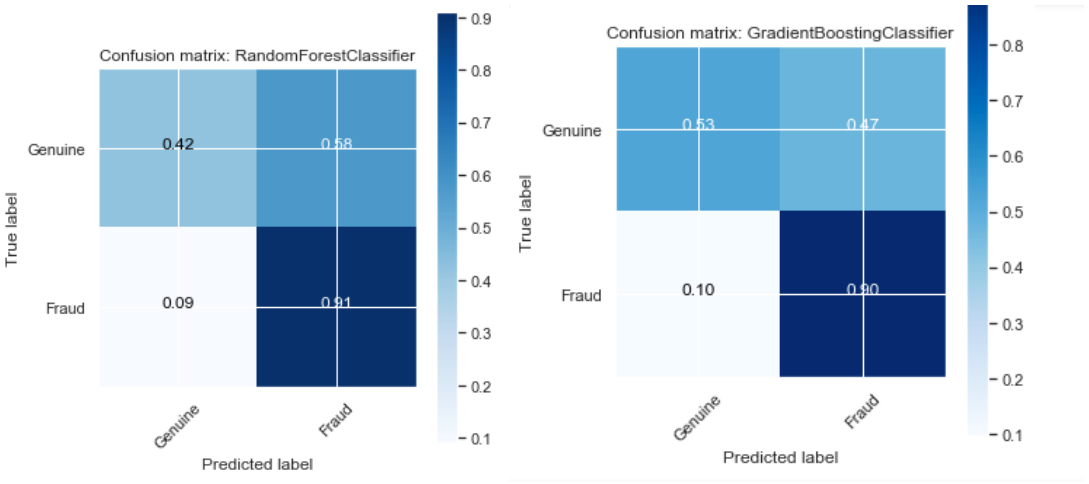
We split the dataset in to training and testing dataset in the ration 4:1. We fit different models using the training dataset and then do prediction on the testing dataset .

The below table shows the accuracy score for various models with default parameters. The data set is split in to training and testing dataset in the ratio 4: 1.

	SVC	DecisionTreeClassifier	RandomForestClassifier	GradientBoostingClassifier
acc_test	0.715	0.8	0.77	0.795
acc_train	0.7575	1	0.9875	0.9825
f_test	0.759358	0.847682	0.81761	0.840939
f_train	0.800106	1	0.989889	0.98475
pred_time	0:00:00.060109	0:00:00.019092	0:00:00.035995	0:00:00.030761
train_time	1:35:19.417864	0:00:00.100429	0:00:00.162568	0:00:01.232515



Venghatesh S
Nov 10th 2018



Training SVM took around 90 Minutes and this is expected since SVM is a poor trainer. Decision tree tends to overfit and we could see that accuracy and f- value on training dataset is 1. Gradient boosting seems to be having the maximum/same accuracy in the test dataset as decision tree. They also don't tend to overfit. It's a hard decision to chosen between these models based on the scores but Gradient boosting does not overfit and hence we could proceed with it. We would take this model and fine tune its parameters. Diagonal values in the confusion matrix should be higher for good performing models. Out of all the models, Gradient boosting performed well.

Refinement

Feature Selection

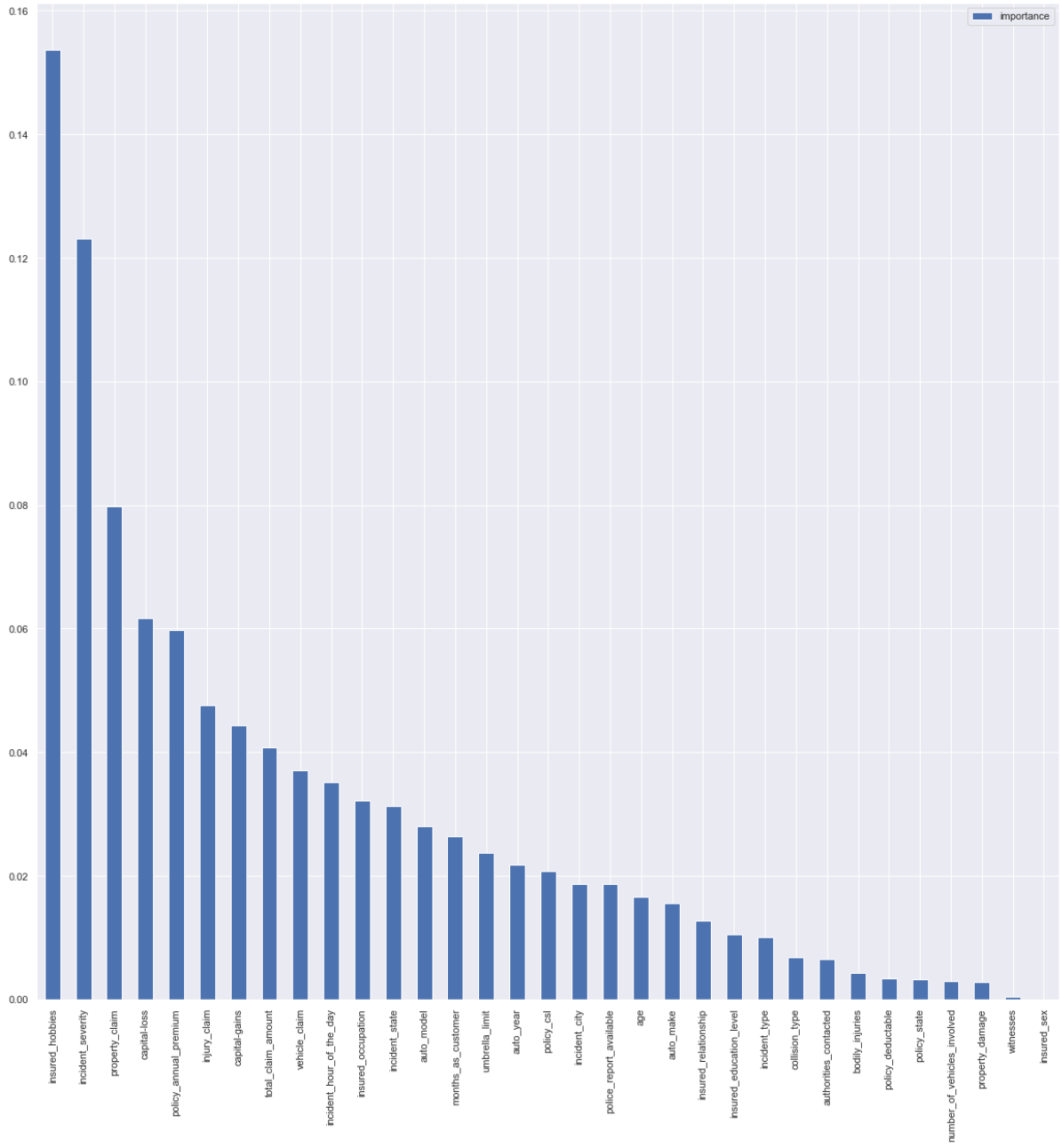
The best approach in machine learning is to choose attributes that are relevant to the domain and that would result in the boosting of model performance i.e. the attributes that result in the degradation of model performance are removed. This entire process is called Feature Selection or Feature Elimination.

Udacity Machine Learning Nanodegree

Fraud Detection on Insurance Claims

Venghatesh S

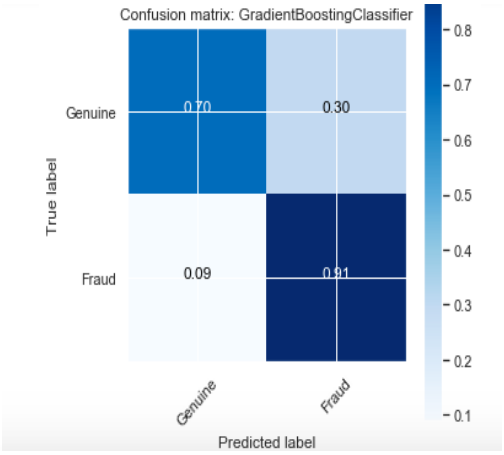
Nov 10th 2018



Features that seem to be more relevant are insured hobby , incident severity. Lets drop all other features and then train the model only on these features .

Venghatesh S
Nov 10th 2018

	GradientBoostingClassifier
acc_test	0.85
acc_train	0.86875
f_test	0.889193
f_train	0.923729
pred_time	0:00:00.003318
train_time	0:00:00.142735



Testing accuracy score and f_test accuracy score has improved compared to bench mark model. Also confusion matrix shows better results compared .

Now let us look at how various parameters of the tree could be further fine tuned.

min_samples_split

- Defines the minimum number of samples (or observations) which are required in a node to be considered for splitting.
- Used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.
- Too high values can lead to under-fitting hence, it should be tuned using CV.

2. **min_samples_leaf**
   - Defines the minimum samples (or observations) required in a terminal node or leaf.
   - Used to control over-fitting similar to min_samples_split.
   - Generally lower values should be chosen for imbalanced class problems because the regions in which the minority class will be in majority will be very small.
3. **min_weight_fraction_leaf**
   - Similar to min_samples_leaf but defined as a fraction of the total number of observations instead of an integer.
   - Only one of #2 and #3 should be defined.
4. **max_depth**
   - The maximum depth of a tree.
   - Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.
   - Should be tuned using CV.
5. **max_leaf_nodes**
   - The maximum number of terminal nodes or leaves in a tree.
   - Can be defined in place of max_depth. Since binary trees are created, a depth of ‘n’ would produce a maximum of 2^n leaves.
   - If this is defined, GBM will ignore max_depth.
6. **max_features**
   - The number of features to consider while searching for a best split. These will be randomly selected.

Venghatesh S
Nov 10th 2018

- As a thumb-rule, square root of the total number of features works great but we should check upto 30-40% of the total number of features.
- Higher values can lead to over-fitting but depends on case to case.

7. **loss**
   - loss function to be optimized. 'deviance' refers to deviance (= logistic regression) for classification with probabilistic outputs. For loss 'exponential' gradient boosting recovers the AdaBoost algorithm.
8. **n_samples**

   The number of boosting stages to perform. Gradient boosting is fairly robust to over-fitting so a large number usually results in better performance.

Out of the above parameters, I have chosen max depth ,min samples,loss,min_samples for fine tuning.

Parameter	Tested Value	Best Value
min_samples_leaf	[1,2,3,4,5,6,7,8,9,10,11,12]	2
max_depth	[2,3,4,5,10]	3
loss	[deviance,exponential]	deviance

Results

Model Evaluation and Validation

We evaluated different models and narrowed down to Gradient Boosting tree . We used sklearn metrics to understand the accuracy and F-scores fo the model. We fine tuned the GBT by using feature selection and fine tuning the parameters. The model accuracy improved by around 4%. The final confusion matrix after using relevant features and getting best fit using grid search looks like this.

Justification

The tuned model improved by around 7%/5% compared to benchmark model with respect to accuracy and F-scores. This improvement is primarily achieved by choosing relevant features. Further fine tuning of the parameters like max depth, number of samples did not improve the accuracy any further. This would need significant amount of time by doing trial and error method and is out of scope for this model.

	BenchMark Model	**Finetuned Model**
Accuracy Score	**0.81**	**0.84**
F-Score	**0.85**	**0.89**

.

Venghatesh S
Nov 10th 2018

Conclusion

Visualisation

Gradient Boosting

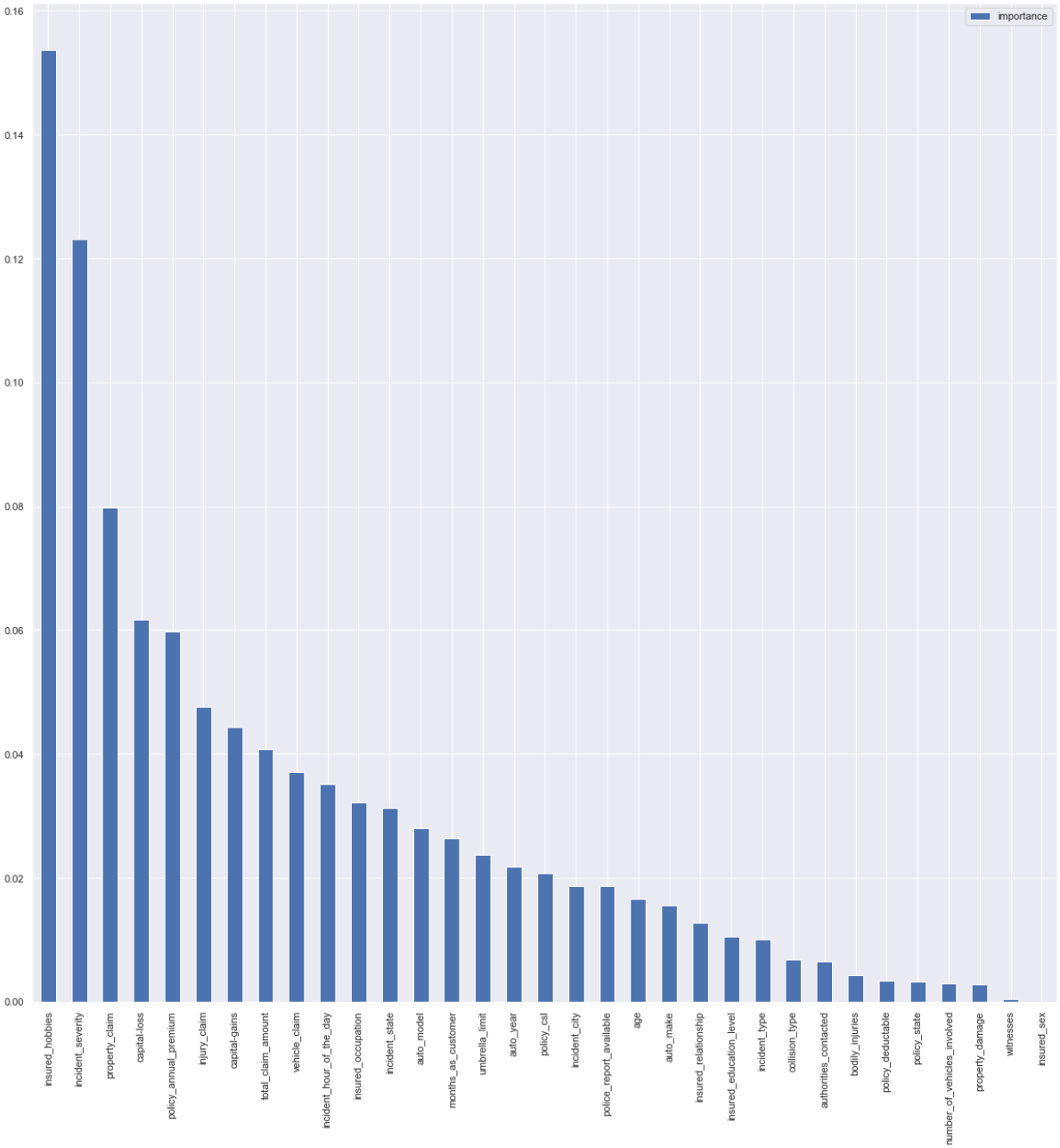
```
Feature Optimised model
------
Accuracy score on testing data: 0.8500
F-score on testing data: 0.8892

 Feature and Parameter Optimized Model
------
Final accuracy score on the testing data: 0.8400
Final F-score on the testing data: 0.8852
Classifier on non-optimised model
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=100,
              presort='auto', random_state=None, subsample=1.0, verbose=0,
              warm_start=False)
```

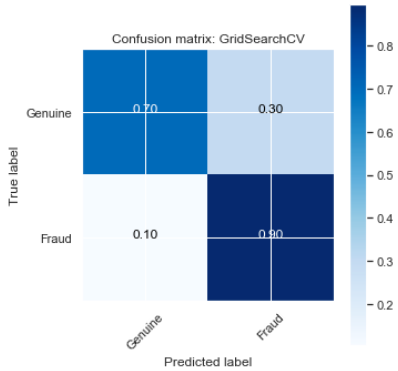
Udacity Machine Learning Nanodegree

Fraud Detection on Insurance Claims

Venghatesh S

Nov 10th 2018





Udacity Machine Learning Nanodegree
Fraud Detection on Insurance Claims

Venghatesh S
Nov 10th 2018

Reflection

The process used in the project could be summarised as follows

1. An initial problem to solved (Detecting fraud in insurance claim) was chosen and publicly available datasets were found.
2. The data was downloaded and pre-processed.
    a. Irrelevant features were removed
    b. Data was normalised.
    c. Non-numeric features are converted to numeric
3. Few models are chosen and analysed by running them through training and test dataset.
4. A bench mark was created for the classification problem.
5. Relevant feature extraction and fine tuning of parameters were done

The most difficult part is to fine tune the parameters to make the model more effective. The most interesting aspect found is that hobbies can play a important role in fraud detection. While initially looking at the dataset , I was thinking of dropping it since I thought this feature is ir-relevant.

Improvements

Further fine tuning is possible by looking at various other GBM Parameters. The dataset seems to be not having some critical information regarding demographic details of the insurer like his/her income, previous criminal history if any. The data is imbalanced since genuine claims outweigh the fraudulent claims, a balanced dataset might improve the accuracy of the model. The parameter level tuning did not yield any improvements compared to feature selection. Further fine tuning of the parameters could be done and model could be improved.

References :

https://en.wikipedia.org/wiki/Insurance_fraud

https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm

https://en.wikipedia.org/wiki/Support_vector_machine

https://en.wikipedia.org/wiki/Decision_tree