# CIA I Research Notes

**Note:** These are notes I took down throughout the project, along with some explanation added where I saw it was needed. Im submitting this for your for your reference, so you can understand how I went about this project

Title - Motor Insurance Fraud Detection System


**Summary**

I took a labeled dataset consisting of around 1700 Motor Insurance Claims, roughly comprising a 50/50 split between Fraudulent and Legitimate claims.

As I went through the process of modeling the data, I realized that creating a model that completely eliminated the human element was not feasible. Therefore I aimed at creating a model that would both identify as many fraudulent cases and eliminate as many non-fraudulent cases as possible - the priority being the former.

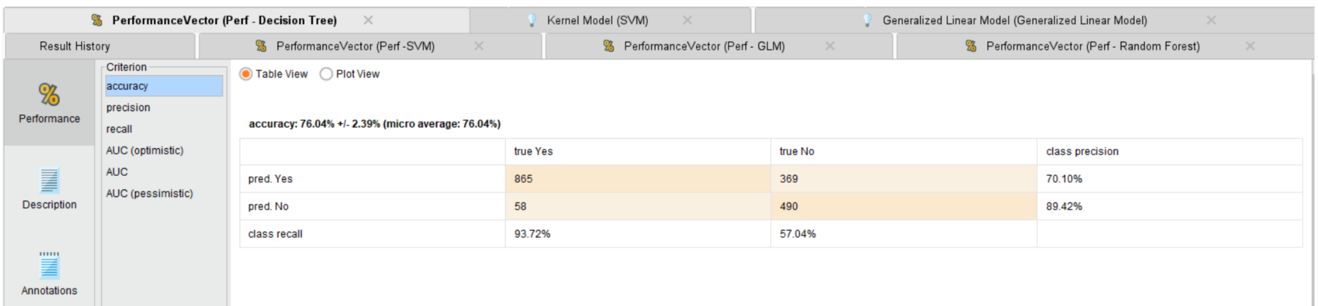
This way, the model would automatically flag a vast majority of Fraudulent Claims, and provided a reduced set of claims to the human experts, who can then comb through the same and pick out the True Positive cases of fraud. This ensures a high fraud detection rate at a lower workload.


**Plan of Action**

- Review Oracle Base Article
    - They started with Insurance Claim data
    - Used Unsupervised Clustering to ID suspicious Claims → the Experts then labeled said data points as Fraud or Legitimate
    - Used Decision Tree, Random Forest, GLM & SVM to create Supervised Fraud Detection models
        - Results not given

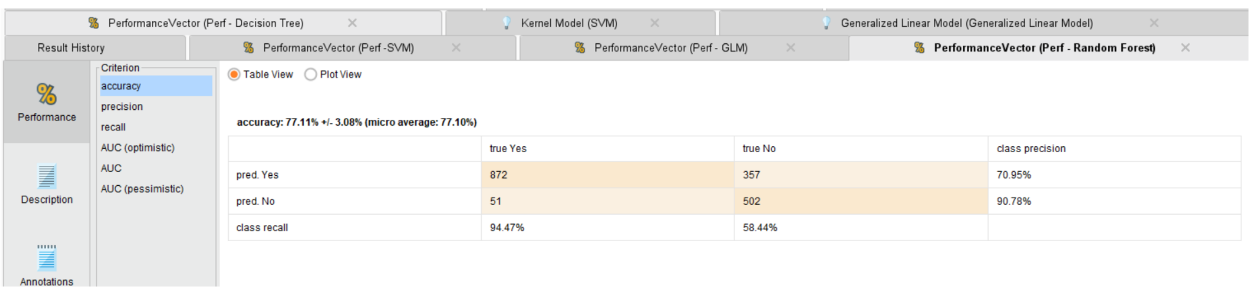
- Understand Dataset
    - Motor Car Insurance Claims - Details + Fraud Label

- Test Oracle Models



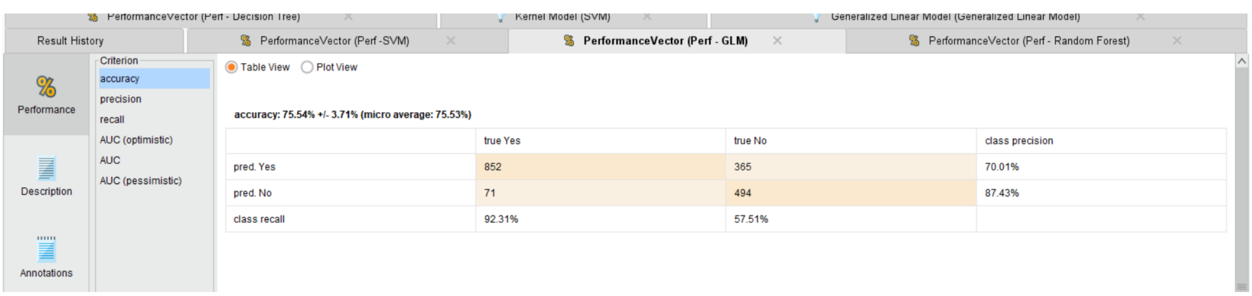

accuracy: 76.04% +/- 2.39% (micro average: 76.04%)

| | true Yes | true No | class precision |
|---|---|---|---|
| pred. Yes | 865 | 369 | 70.10% |
| pred. No | 58 | 490 | 89.42% |
| class recall | 93.72% | 57.04% | |



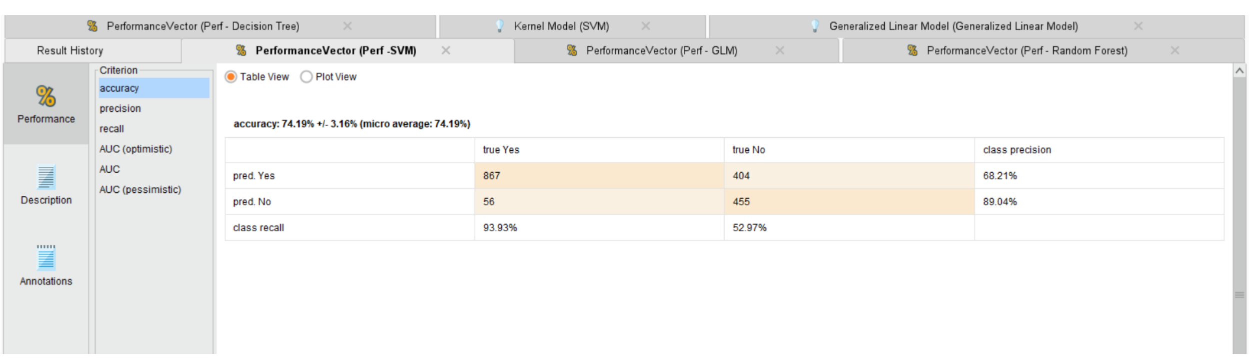

accuracy: 77.11% +/- 3.08% (micro average: 77.10%)

| | true Yes | true No | class precision |
|---|---|---|---|
| pred. Yes | 872 | 357 | 70.95% |
| pred. No | 51 | 502 | 90.78% |
| class recall | 94.47% | 58.44% | |




accuracy: 75.54% +/- 3.71% (micro average: 75.53%)

| | true Yes | true No | class precision |
|---|---|---|---|
| pred. Yes | 852 | 365 | 70.01% |
| pred. No | 71 | 494 | 87.43% |
| class recall | 92.31% | 57.51% | |

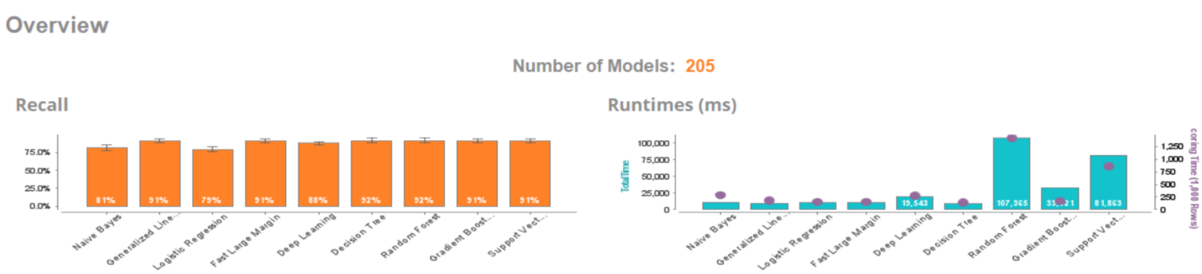



accuracy: 74.19% +/- 3.16% (micro average: 74.19%)

| | true Yes | true No | class precision |
|---|---|---|---|
| pred. Yes | 867 | 404 | 68.21% |
| pred. No | 56 | 455 | 89.04% |
| class recall | 93.93% | 52.97% | |

- How to Interpret Results - Positive Case - Fraud is Found
    - Recall = Sensitivity % of Positive Cases detected = % of Fraud cases caught
    - Specificity = % of Negative Cases Detected = % of Legitimate cases eliminated = Reduction in workload
    - Performance Metrics for fraud detection problem -
        - Recall = Fraud Detection rate
        - Workload Reduction level = Specificity
        - Total Workload = % of Claims needed to be investigated = [ (Recall * % Fraud cases) + (1 - Specificity) * Non-Fraud Cases ] / All Cases

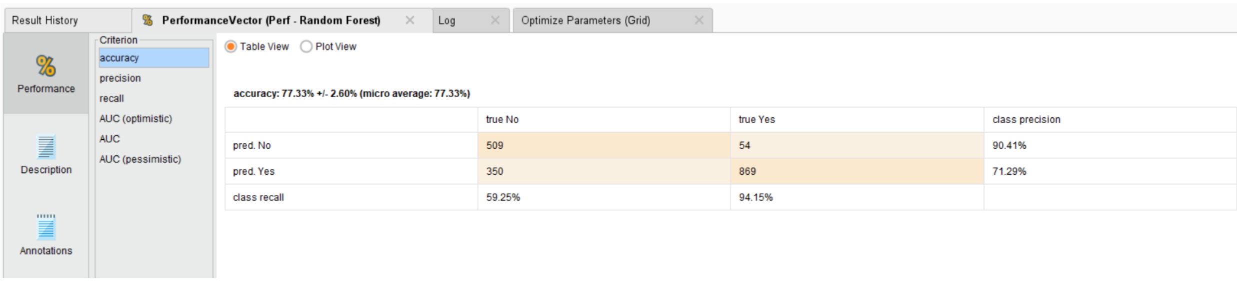
- Auto Modeling




| Model | Recall ↓ | Standard Deviation | Gains | Total Time | Training Time (1,000 ... | Scoring T |
|---|---|---|---|---|---|---|
| Random Forest | 91.7% | ± 3.4% | 254 | 1 min 47 s | 40 ms | 1 s |
| Decision Tree | 91.6% | ± 3.4% | 244 | 9 s | 16 ms | 144 ms |
| Generalized Linear Model | 91.2% | ± 2.7% | 230 | 10 s | 35 ms | 182 ms |
| Fast Large Margin | 91.2% | ± 2.7% | 230 | 11 s | 42 ms | 154 ms |
| Gradient Boosted Trees | 91.2% | ± 2.7% | 230 | 33 s | 68 ms | 170 ms |

- Final 6 Models Selected are
    - Random Forest (Best Performance)
    - Decision tree
    - GLM
    - SVM
    - Naive Bayes
    - Logistic Regression (last two chosen for high Specificity)

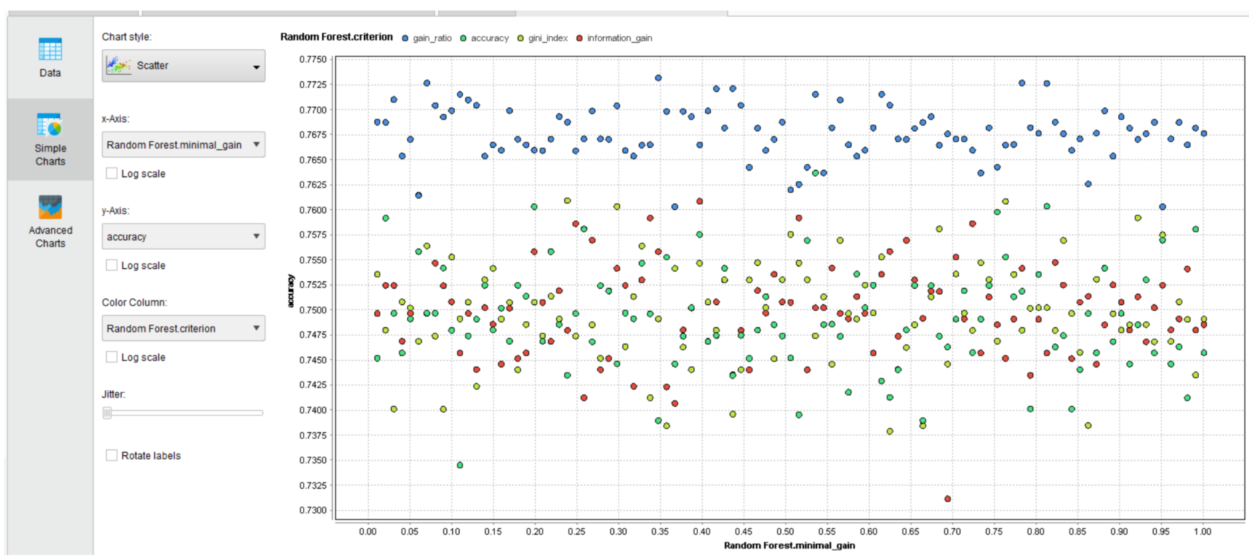
- Optimizing Random Forest model using [Optimize Parameters] operator in Rapidminer




accuracy: 77.33% +/- 2.60% (micro average: 77.33%)

| | true No | true Yes | class precision |
|---|---|---|---|
| pred. No | 509 | 54 | 90.41% |
| pred. Yes | 350 | 869 | 71.29% |
| class recall | 59.25% | 94.15% | |

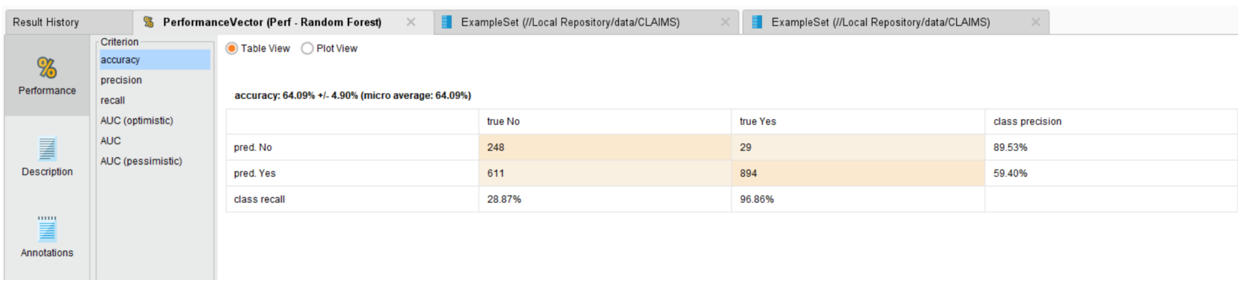
Used selected parameters and iterated to find model with best accuracy (a .22% gain over non-optimized model). Minimal gain ratio was most important parameter for optimization.




Note: Optimized Random Forest model had lower Recall (lower % of Fraud cases detected). So I has forgone this model and method of optimization and used Stacking instead (explained below).

- Maximizing Recall

By varying the confidence threshold (here it is 0.4), we can increase recall at the cost of specificity:




  - 96% fraud cases detected
  - Workload reduced by 28%
  - This is not an efficient way of improving the model, the trade-off between Fraud detection and workload reduction is too expensive

- Maximizing Precision

The reverse (increasing threshold) does not improve precision significantly. It in some cases decreases it.

- Correct Method of Testing

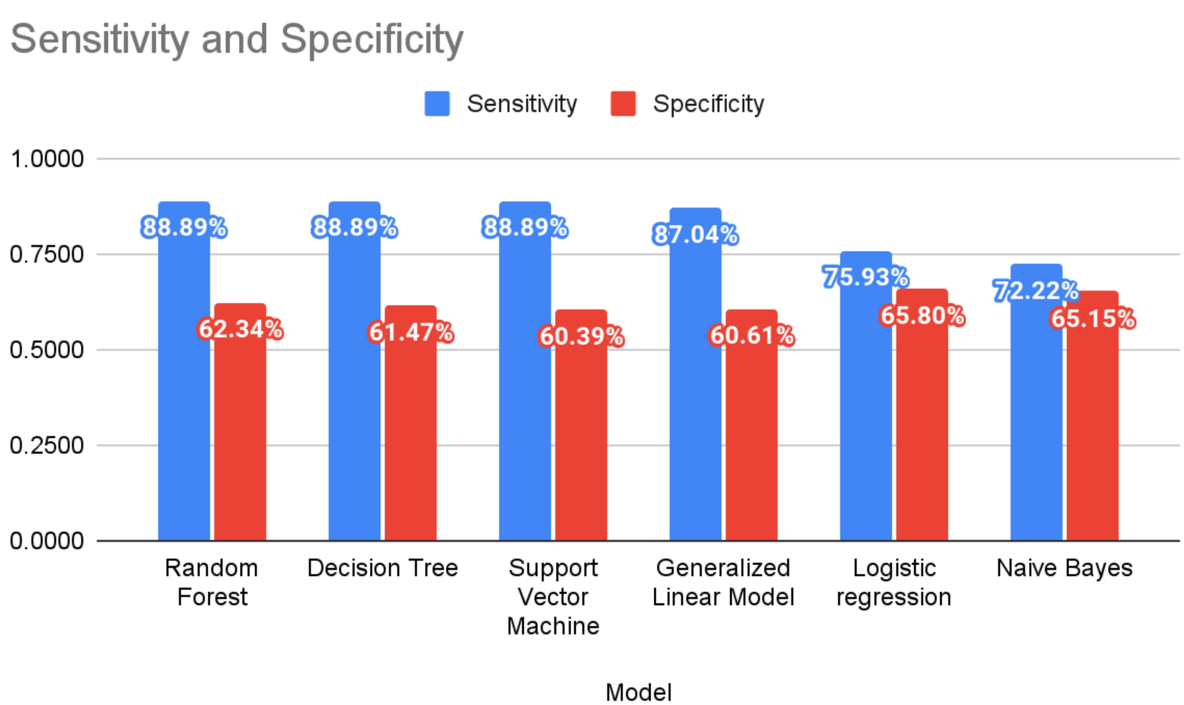
I made the following errors and while initially testing the models and solved these errors as follows

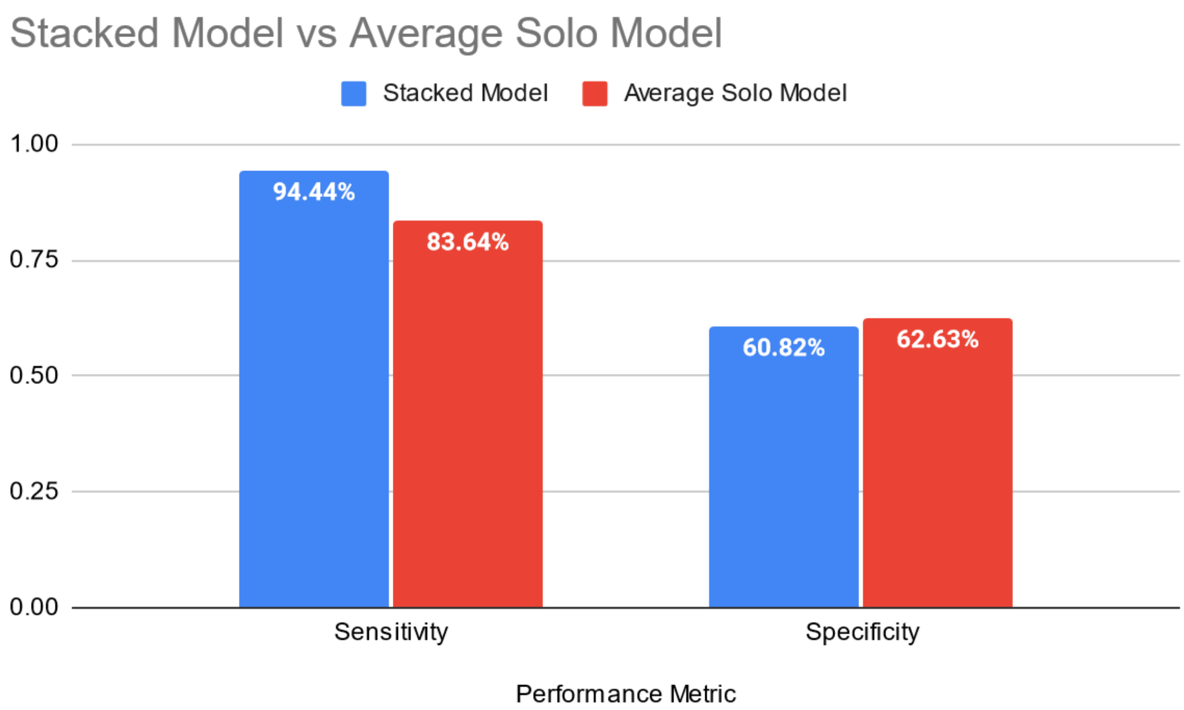
  - **Problem One - Improperly balanced Testing Data.** 10% of claims are fraudulent. The dataset has a 50/50 split. I tested with this ratio, and not the real world ratio of 10/90, inflating the models' performance
  - **Solution -** Sampling. Used sampling to create a dataset with a 10/90 split, which provided more accurate results
    - This yielded very similar results to testing on 50/50 ratio data, apart from a much lower Precision rate. Fraud and Non-Fraud detection rates remained similar in both cases
  - **Problem Two - Testing using Training Data.** I used cross validation to train and test models. CV trains and tests models on the same data, which leads to ingflated performance results.

    - **Solution - Train-Validation-Test Splitting.** By splitting the data at the beginning into 80/20 ratio, 80 for training and validation (testing conducting during the model training process) & 20 for final testing (this data is never seen by the model till the final test), the training and testing occurred on two mutually exclusive datasets
- Model Evaluation
- Solo Models - As seen in the workflow
    - The Claims dataset was split into 80/20 Train-Test ratio
    - The Training Data was fed into the 6 models tested
        - The models were trained using cross validation, to minimize the risk of overfitting
    - The Models were then tested on the 20% Testing data
    - Testing data was rebalanced to meet the 10/90 → Fraud/Legit ratio that the model would face in a real world scenario
    - The Models performance was outputted by rapidminer and recorded in a Google Sheet
    - This process was repeated thrice using different random splits and performance was averaged out to ID the realistic performance of each model
- Stacked Model
    - 4 models from the precious Solo Models were included in the Model, along with a Deep Learning Algorithm and Gradient Boosting Tree, were incorporated into a ensemble Stacked Model
        - SVM was left out as it required special configuration that I did not have the time for, and Decision Tree was left out as I already had Random Forest in the Stacked model
        - Deep Learning and GBT models were added to see if they would improve the Stacked Model's performance (as the Stacked Model would use information provided by the models incorporated into it as it saw fit, and adding models into the stacked model normally increases performance)
    - The rest of the workflow was similar to the Solo Models' Workflow
        - Performance testing 9 times using different random splits to find average performance

- Final Peformance
  - Split = 80/20

| Model | Sensitivity - 1 | 2 | 3 | AVG | Specificity - 1 | 2 | 3 | AVG |
|---|---|---|---|---|---|---|---|---|
| DT | 94.44 | 83.33 | 88.89 | 88.89 | 60.39 | 59.09 | 64.94 | 61.47 |
| RF | 94.44 | 83.33 | 88.89 | 88.89 | 59.74 | 61.04 | 66.23 | 62.34 |
| GLM | 94.44 | 77.78 | 88.89 | 87.04 | 59.74 | 58.44 | 63.64 | 60.61 |
| SVM | 100 | 83.33 | 83.33 | 88.89 | 59.09 | 57.79 | 64.29 | 60.39 |
| NB | 77.78 | 66.67 | 72.22 | 72.22 | 64.94 | 61.69 | 68.83 | 65.15 |
| Logistic regression | 88.89 | 61.11 | 77.78 | 75.93 | 64.94 | 63.64 | 68.83 | 65.80 |
| Stacked (average of 9 tests) | | | | 94.44 | | | | 60.82 |




Stacked Model vs Average Solo Model
Stacked Model
Average Solo Model
1.00
0.75
0.50
0.25
0.00
94.44%
83.64%
60.82%
62.63%
Sensitivity
Specificity
Performance Metric

## Project Idea

### Background

Insurance firms process vast amounts of insurance claims as a part of their routine business operations. Most claims are valid, and are approved by the insurers after being scrutinized by agents and adjusters. However, a portion of claims are invalid, largely due to 2 reasons - the loss not being covered by the insurance policy or intentional fraud committed by the insured. This project focuses on the detection of cases that fall under the latter category - intentional fraudulent claims.

### Research Problem

The objective of this project is to apply relevant statistical & machine learning algorithms to devise a ML model that will provide value to the insurance fraud detection process in two ways -

- automated detection of fraudulent claims with a reasonable level of Recall
- High precision of the detector

The critical success factor for this project is the successful identification of claims that are fraudulent.

### Dataset

**Name :** Motor Insurance Fraud Data Set

**Source :** Analytics and Data Oracle User Community Github

**Type of Data:** Quantitative & Qualitative data about Motor Insurance Claims, with fraudulent claims clearly labeled

**Source Article :** A Two-Step Process for Detecting Fraud using Oracle Machine Learning

**References**

[0] RapidMiner Academy Tutorials

[1] A Two-Step Process for Detecting Fraud using Oracle Machine Learning

[2] Insurance Frauds Control Act; an urgent need in India - BusinessToday

[3] IRDA - Indian Insurance Market