Contents

[1.0 Overview 3](#_Toc101222650)

[2.0 Final Methodology 3](#_Toc101222651)

[3.0 Final Solutions 6](#_Toc101222652)

[4.0 Findings and results 13](#_Toc101222653)

[5.0 References 18](#_Toc101222654)

# Overview

In this digital era, the common issue faced by insurance industries is that such industries are fraud claims risk where insurers are not able to predict fraud claims and such risks account for a significant portion that would cost billions of dollars annually. The solution suggested is building a fraud claims detection Model using decision tree, which is a supervised learning algorithm that is widely used for classification. For this research, decision tree was chosen as it is a powerful and attractive method that can interpret and organize results easily and in a well-organized manner (Gaikwad et al., 2014). Decision trees are used to create descriptive models that will be used to describe the characteristics of fault claims. For example, if the incident\_severity\_minor\_damage <= 0.5, those that score less than 0.5 for that feature will be separated to a different decision point, and those that scored above 0.5 would be separated to a different decision point and so on (Bhowmik, 2011). Decision trees can display independent and dependent attributes in a tree-shaped structure. The motive of the clustering allows insures to identify complex fraudulent claims as insurance industries consists of large amount of sensitive data and variables that are non-observable that are not captured in traditional insurance rating dimensions.

# 2.0 Final Methodology

For the fraud claims detection model, the analysis was conducted to detect fraud claims to prevent losses by classifying the claims as ‘fraud’ and ‘genuine’ which is determined using features and decision rules. The raw data obtained was in the form of a CSV and consist of structured and unstructured data. For insurance industries, these data can be easily obtained from customer records, policy records and incident records that are usually stored in databases. The claims dataset consists of 1000 instances and 39 columns were extracted for this research. The figures below show the relationship between 3 tables and sample dataset that we combined for the detection model:

Table

Description automatically generated

Figure Relationship between tables for Benchmarking

A picture containing graphical user interface

Description automatically generated

Figure customer\_details

Table

Description automatically generated

Figure policy\_details

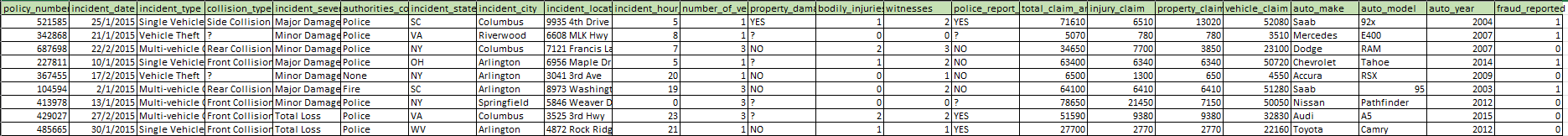


Figure incident\_details

Massaging and analysis

As the data obtained is in a raw manner, it would have to go through levels of analysis to before being used to train the model. The unstructured data would be converted to structured form before the quality of the data is processed. Such large data with several fields would have the possibility of containing null values that would affect the analysis. For the classification process, duplicates, unwanted characters, and data format are changed to ensure that the feature and target variables does not contain imperfections which will affect the accuracy of the model. The effectiveness of the fraud detection model depends on the quality of the data, imbalanced dataset can cause variance (Kumar & Garg, 2018). In this phase, massaging the data means converting raw data that is used for analytics.

Data Exploration

After the main dataset has been cleaned and processed, it will be observed in order to pursue with the hypothesis formation and testing. The dataset's redundant and outlier data would be cleaned during this cycle. For our classification, we would be using the decision tree method to build our model, decision trees are robust to outliers since divisions happens based on the segmentation of samples within the split ranges and not on absolute values (Gulati et al., 2016). Furthermore, correlation between variables is explored to investigate irrelevant features to be eliminated that would affect the effectiveness of the model. The data can be explored deeper to gain more insights on it by observing the features which would be used for the fraud detection model and ensuring that the model has a balanced training set (Nisbet et al., 2018).

Build Models

After data exploration, the model would be build based on the processed data. Various algorithms and experiments are conducted to ensure that the models reached their optimal performance. For the classification model, the dataset would be split into training and testing dataset and various variables and factors would be ruled out and included. Decision tree uses a branching method that would display all the possible outcomes of a problem and can handle conditional information by dividing the dataset into subgroups that are further considered as an individual dataset for detailed processing (Gulati et al., 2016). Decision tree algorithm is used on the test model to see if the model is able to make accurate predictions which would determine if the model were fit thus proving the hypothesis (Kumar & L., 2018). If the training set provided is oversampled, it increases the risk of overfitting by biasing the models, thus the complexity needs to be reduced to improve predictive accuracy and reduce error rate. In order to set a limit to spot the nodes to split further when the specified tree depth has been reached, tree depth is specified to customize the binary decision tree.

Visualization

Visualization is a central part of data analysis. It is widely used to examine and validate the analysis before decision-making. Visualization can be made using Matplotlib and Seaborn, which are used to plot graphs, histograms, box plots and tables etc (Making Sense of Data II: A Practical Guide to Data Visualization, Advanced ... - Glenn J. Myatt, Wayne P. Johnson - Google Books, n.d.). For the detection model, the confusion matrix would assist users in visualizing the performance of the classification model which contains information about the actual and predicted classifications (Visa et al., 2011). The decision tree will be plotted with plot\_tree which will map out all the possible outcomes of a series of related choices which would assist in evaluation of decisions.

# 3.0 Final Solutions

Jupyter Notebook was used for data cleaning and transformation, numerical simulation, exploratory data analysis and data visualization. Python, a high-level interpreted programming, was used for these processes as it provides the use of low-level libraries and clean high-level APIs (Raschka et al., n.d.).

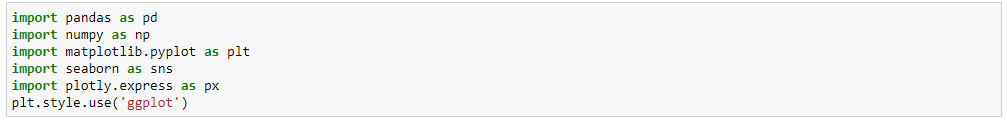


Figure Import libraries

For this analysis, Python libraries such as Pandas,Numpy,Matplotlib,seaborn, and plotly express will be used. Pandas is a library of rich data structures and tools to be used on structured data sets as it provides integrated, intuitive routines for performing common data manipulations and analysis on such data sets. Numpy (Numerical Python) , a foundational package for scientific computing was imported as it includes basic linear algebra routines, Fourier transform capabilities and random number generator. Numpy arrays are efficient in storing and manipulating the data and libraries in a lower level language could operate on the data. Matplotlib is a plotting library for Python and its numerical extension Numpy. It contributes an object-oriented API for embedding plots into applications and is used by SciPy. Seaborn package was developed based on the Matplotlib library to create captivating and informative statistical graphics. It can also can be used to develop the attractiveness of matplotlib graphics.

Graphical user interface, text

Description automatically generated with medium confidence

Figure Joining of tables

Graphical user interface, text

Description automatically generated

Figure df table

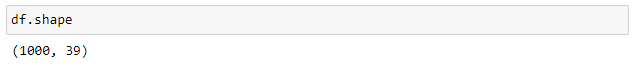


Figure DataFrame Shape

Here the three raw datasets are imported and labelled as ‘a’, ‘b’, and ‘d’. The pd.read\_csv() method to read the datasets that are stored in CSV. Next, we would have to merge the first two datasets. In this section, merge() is used to join the two data frames in pandas python. The key for joining the tables are specified using ‘on=[‘column\_name] and to determine on how the tables should be joined, we used the ‘how=’inner/left/right’) to perform natural joins. To export the tables that were joined into a CSV format, we use the ‘merged.to\_csv(‘filepath/filename’)’. Here, three tables were joined, and the main dataset that will be used is combined.csv, which is also labelled as df.

In figure 7, the df data frame is displayed to show the top 5 rows of the data. This would assist us in quickly testing if the objects contains the right data in it.



Figure Replacement of special characters in fields

As the dataframe consists of raw data that would have to go through data cleansing, the special characters (?) in the fields of various columns represent missing values. To make it easier for analyzing the data, we replaced the special character with ‘np.nan’ so that these nan values can be replaced with a valid value during data processing by using the df.replace() function.

Text

Description automatically generated with medium confidence

Figure df Summary

To have a quick summary on the data, the df.info() function is used. As shown in figure 10, this method prints out a summary on the DataFrame where the number of cells in each column and also the data types of the columns. The info() method is used to print the information and does not return any value.

Table

Description automatically generated with medium confidence

Figure Sum of null values

The df.isna() function is used to detect missing values for an array-like object in the pandas dataframe. In the dataset used, NaN stands for missing values in object arrays. As shown in the result in figure 11, the ‘collision\_type’ consists of 178 null values in that column.



Figure Handling Missing Values

To handle the missing values, df.fillna() method from Pandas is used. For this example, the ‘collision\_type’ column in the data frame is used for imputing missing values with the central tendency measures such as mode with the df.fillna() method, followed by (df[‘column\_name’].mode() to replace it with the mode value of the dataframe. If the data is skewed, it is recommended to use mode to replace the values as this method works for both numerical and categorical data which exists in our dataframe.

Graphical user interface, text

Description automatically generated with medium confidence

Figure Plotting a Heatmap

Chart

Description automatically generated

Figure Heatmap Result

In this section, we would plot a heatmap to display a 2D correlation matrix between two discrete dimensions by using coloured cells to represent data from a monochromatic scale. The values of the first dimension would be listed as the row and the second dimension would be listed as a column. This would assist in analysing the data as it makes patterns easily readable and displays the differences and variation of a data. To plot the heatmap, Seaborn, a Python library based on matplotlib is used for data visualization by using the sns.heatmap() function. The plt.figure() function in pyplot module is used to create a new figure and the ‘figsize’ is used to display the size of the plot figure as a tuple of the width and the height of the figure in inches. The Pandas corr=df.corr() function would compute pairwise correlation of columns without considering the null values. The sns.heatmap() function would plot the data ‘corr’ and display it as shown in figure 14. The correlation is a statistical method that is used to test relationships between quantitative variables or categorical variables. If the dataset contains of highly correlated features, the variance of the weights is high and the model would be sensitive to data and would not perform well. However, for this decision tree, it is immune to correlation by nature. As the attributes is split, the tree would choose only one of the perfectly correlated features and the correlation method can be used for our own understanding on the data.

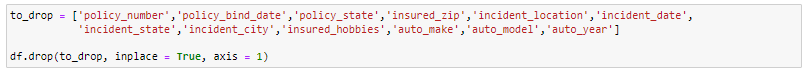


Figure Dropping unnecessary columns

Graphical user interface, text, application, email

Description automatically generated



Figure Feature and target columns

Text

Description automatically generated with medium confidence

Figure Coversion of categorical columns

Graphical user interface, text, application

Description automatically generated

Figure Joining of numerical and categorical indicator columns

To build the detection model, since the target and feature columns are used, it is important to eliminate unnecessary features that would not contribute to the training or testing of the detection model as it is considered a low predictive column and based on the heatmap. Next, the dataframes were separated into feature and target columns and the categorical columns and numerical columns were extracted as shown in Figure 16. Fraud reported consists of binary values which would be used as target columns. As shown in Figure 17, the categorical columns which consists of grouped data has to be converted into indicator variables as shown in figure 17 using the pd.get\_dummies() function from Pandas to ease the training and testing of the models. In figure 18, the dataset that would be used for training and testing is a combination of numerical and categorical indicator columns which are concatenated using the pd.concat() function.

Graphical user interface, text

Description automatically generated

Figure Splitting training and testing data

Before training the model, the dataset obtained would be split into training and testing data set. Here, the train\_test\_split function from Sklearn model selection is used to split the data arrays into two subsets. The test\_size would determine the proportion of the dataset that would be included in the train split.For this analysis, it is set to 0.25, which is 25%, because decision trees perform better on smaller training sets and various test\_size were tested and proven that test\_size = 0.25 was the best fit and was able to provide a higher accuracy.

Graphical user interface, text, application, Word

Description automatically generated

Figure Decision Tree

For this section, we would be training the decision tree by importing the DecisionTreeClassifier module from the Sklearn.tree library as shown in Figure 20. We would input the training values, the x and y values, that would be used to build the decision tree model. We declared the parameters (max\_depth=3) as it was able to give a better accuracy and would not display the tree in a complicated manner. Max\_depth is widely used to preprune a decision tree and the dataset has been trained.

A picture containing text

Description automatically generated

Figure Prediction

Graphical user interface, text, application

Description automatically generated with medium confidence

Figure Probability of classes

In this step, the clf, an estimator instance, which is used to store trained model values are further used to predict values based on the previously stored weights. As shown in the results of the prediction, it displays the class number that it is predicting, so for the first number, it is a genuine claim as it is ‘0’. However, for the 9th claim, it is predicted as a fraud claim as it displays the value ‘1’ for that class number as shown in Figure 21. Furthermore, the sklearn estimator also implements the predict\_proba() method that would return the class probabilities for each data point and would give the only probability of 1 as shown in figure 22.

# 4.0 Findings and results

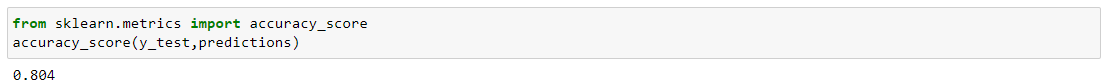


Figure Accuracy Score on 25-75 dataset

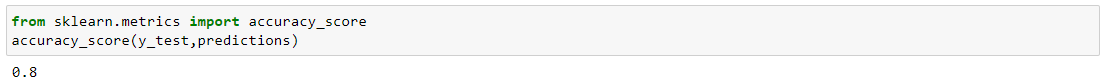


Figure Accuracy Score on 20-80 dataset

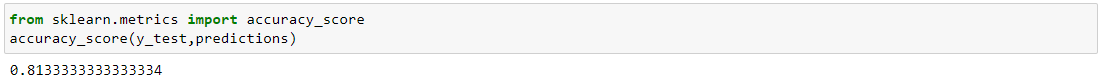


Figure Accuracy Score on 30-70 dataset

For this decision tree, the accuracy of the model is calculated using sklearn.metrics.accuracy\_score, which is an accuracy classification score as shown in Figure 23,24, and 25. The method of the accuracy score being calculated is using Accuracy Score = (TP+TN)/(TP+FN++TN+FP). True Positive is the outcome where the model correctly predicts the positive class and true negative is the outcome where the model was able to predict the negative class correctly. False positive is an outcome where the model incorrectly predicts the positive class, and the false negative is when the model incorrectly predicts the negative class. Accuracy\_score works well with multilabel classification in which the accuracy\_score function calculates the subset accuracy. In this model, max depth = 3 with a training data of 30% was able to give the best accuracy score and it is the best fit for the model. Max Depth =4 and 2 with the same proportion of training data were tried on the training model, and the accuracy of the model ranges between 0.72-0.77.



Figure Confusion Matrix on 25-75 dataset



Figure Confusion matrix on 20-80 dataset



Figure Confusion matrix on 30-70 dataset

Table

Description automatically generated

Figure Understanding confusion matrix

The confusion\_matrix module from sklearn.metrics was imported to measure the performance of the machine learning classification problem where the output can consist of 2 or more classes. In figure 29, it is shown what each value in the array represents, which is the TP and FP etc, which was explained on how such values were obtained in the section above. In our results, 192 is our TP, 31 as FP, 25 as FN, and 52 as TN, which is a good performance, that was done on 30-70 dataset.

Text

Description automatically generated

Figure Classification report on 25-75 dataset

Graphical user interface, text, application

Description automatically generated

Figure Classification report on 20-70 dataset

Graphical user interface, text, application

Description automatically generated

Figure Classification report on 30-70 dataset

A picture containing text

Description automatically generated

Figure Recall Formula

A picture containing text

Description automatically generated

Figure Precision Formula

Text

Description automatically generated with medium confidence

Figure F1 Score Calculation

The classification\_report module from sklearn.metrics is also imported to view the precision, recall and f1 score of the model. Based on our results both the genuine and fraud’s precision are above 0.63, which is about 63-88% correct of the time when tested on 30-70 dataset, which displays a good score. Recall is the ratio of correctly predicted positive observations to the observations in actual class. In our results, our classifier does not have a high number of false negatives and was able to score above 0.68-0.86, which is the score on the 30-70 dataset and provides the best score among all other training and testing dataset. Based on our results on 30-70 dataset, our classifier can identify real threats and is not disturbed by false alarms as the f1 score is between 0.65 – 0.87. Overall, the classifier model which is trained on 30-70 dataset performs well in all aspects compared to the other training and testing datasets.

Diagram

Description automatically generated

Figure Decision Tree of the Fraud Detection Model

In figure 36, the decision tree is displayed where it shows a branched flowchart showing multiple pathways for potential decisions and outcomes. The tree would begin with a decision node which signifies that a decision must be made. Based on the detection model, insurers can efficiently detect false and falsified claims in the insurance industries and prevent fraud by detecting patterns and achieving superior predictive performance.

# 5.0 References

Bhowmik, R. (2011). Detecting Auto Insurance Fraud by Data Mining Techniques. *Journal of Emerging Trends in Computing and Information Sciences*, *2*(4), 156–162.

Gaikwad, J. R., Deshmane, A. B., Somavanshi, H. V, Patil, S. V, & Badgujar, R. A. (2014). Credit Card Fraud Detection using Decision Tree Induction Algorithm. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, *6*, 2278–3075.

Gulati, P., Sharma, A., & Gupta, M. (2016). Theoretical Study of Decision Tree Algorithms to Identify Pivotal Factors for Performance Improvement: A Review. *International Journal of Computer Applications*, *141*(14), 19–25. https://doi.org/10.5120/ijca2016909926

Kumar, V., & Garg, M. L. (2018). Predictive Analytics: A Review of Trends and Techniques. *Article in International Journal of Computer Applications*, *182*(1), 975–8887. https://doi.org/10.5120/ijca2018917434

Kumar, V., & L., M. (2018). Predictive Analytics: A Review of Trends and Techniques. *International Journal of Computer Applications*, *182*(1), 31–37. https://doi.org/10.5120/ijca2018917434

*Making Sense of Data II: A Practical Guide to Data Visualization, Advanced ... - Glenn J. Myatt, Wayne P. Johnson - Google Books*. (n.d.). Retrieved April 10, 2022, from https://books.google.com.my/books?hl=en&lr=&id=lFBqIpM-vuQC&oi=fnd&pg=PR5&dq=data+visualization+clustering&ots=VPxXC2O4mS&sig=rPIm9b4xLC6JSB72C-wu\_WRCc\_E&redir\_esc=y#v=onepage&q=data visualization clustering&f=false

Nisbet, R., Miner, G., & Yale, K. (2018). Fraud Detection. *Handbook of Statistical Analysis and Data Mining Applications*, 289–302. https://doi.org/10.1016/B978-0-12-416632-5.00015-3

Raschka, S., Patterson, J., & Nolet, C. (n.d.). *information Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence*. https://doi.org/10.3390/info11040193

Visa, S., Ramsay, B., Ralescu, A., & Van der Knaap, E. (2011). Edited by Sofia Visa, Atsushi Inoue, and Anca Ralescu. *Maics*, *710*, 120–127.

# 6.0 Appendix

A picture containing table

Description automatically generated



Table

Description automatically generated

Graphical user interface

Description automatically generated

Chart

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface, application, table

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Table

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated

Diagram

Description automatically generated