DATA ANALYTIC TECHNICAL REPORT

SIT717 – ASSIGNMENT 2

Abstract

Technical report on evaluation of Insurance claims data using classification

Rajeshkumar Mourya

218615876

Contents

Title

Title	Using decision trees to detect fraudulent claims from insurance data
Student name	Rajeshkumar R Mourya
Student id	218615876
Email	rmourya@deakin.edu.au

Abstract

Frauds are very costly for insurers resulting in losses from thousands to billions of dollars. This report is aimed at the classification of insurance claims to detect fraudulent claims using supervised machine learning algorithms. Machine learning has advanced over the years and its applications are widespread across different sectors. Digital transformation across industries has resulted in the generation of a vast amount of data. Businesses have changed their processes, capturing all data in digital form. The insurance industry has also transformed and has reduced processing times of claims using digital transformation. Various sectors are using insurance as a safeguard to reduce risks involved with accidents, losses, damages due to unforeseen events etc. While this has been a blessing for industries, some individuals and firms are trying to use these processes and its loopholes to benefits illegally from insurers. This technical report is targeting such claims that may impact the finances of insurers and damage their operations altogether. The report further demonstrates how the decision tree and random forest classifiers can be useful in classifying claims in different categories to help insurers.

Keywords: ***Data mining, Classification, Decision tree, Insurance claims, fraud, WEKA, J48, Random forest***

Introduction

Background

Groups and individuals are purposely misleading insurance companies to gain illegitimate financial benefits by providing forged documents and/or showing nonexistent expenses in insurance claims. Such claims are resulting in a huge amount of losses for the insurance industry. According to a recent survey, 15 per cent claims in the insurance industry are fraudulent. In the USA alone, insurers have suffered losses of over 30 billion USD every year due to health insurance frauds (Rawte & Anuradha 2015). Researchers have studied different machine learning techniques to tackle this issue. A mechanism for detecting auto insurance frauds was built using Naïve Bayes and decision algorithm (Bhowmik 2011). Bauder and Khoshgoftaar (2017) researched Medicare fraud using Gradient boosted machine, random forest and deep neural networks. A framework for detection of illegitimate insurance claims was proposed by Rayan (2019). This framework was further deployed as a fraud detection engine to Medi Assist group and in private companies where Medi Assist group was a health insurance provider. The engine utilized decision trees to flag fraud claims which were later processed by experts. The system provided excellent results and the rate of detection of fraudulent claims was increased by over 200%. Harjai, Khatri and Singh (2019) proposed a SMOTE-SVM model that resulted in the accuracy of 94% and recall of 99.9% in the detection of fraudulent cases. These researchers proved that machine learning can be a useful tool for detection of illegitimate claims by insurers, providing better accuracies and increased throughput.

Motivation

Frauds result in operational issues for the insurance industry. This may result in increased premium prices to purchase insurance, reduced rate of processing and slower operations. It impacts the reach of insurance and the benefits that can help individuals who cannot afford healthcare costs and repair expenses. The insurance industry is facing a hard time due to losses occurred annually due to these fraudsters. Insurers are dealing with very high claims ratio and it might become harder in future for them to find illegitimate claims within the stipulated processing time. This will increase the losses and damages to the insurance industry. The motivation behind this research was to reduce fraud claims using machine learning to detect and flag them. This will help in decreasing frauds, albeit not completely but it will reduce them significantly. Reduction in losses can also help in insurance policies to be affordable for everyone.

Aim

The aim of the data analysis in this report is to demonstrate the use of supervised learning techniques, decision tree (J48) and random forest classifiers for **binary classification** of insurance claims data. The insurance claims data is sourced from Kaggle repository (Kaggle, 2019). We will be analyzing the data to classify them in categories indicating if the claim was fraudulent or not.

This report is divided into following subsections: Dataset summary, details of data mining techniques, evaluation of models, a pseudo-code for improvised algorithm and conclusion.

Dataset summary

Insurance claims dataset is sourced from Kaggle repository (Kaggle, 2019) as a CSV file. WEKA 3.8.4 was utilized for an analysis hence we do not need to convert the file into ARFF file as it can be loaded directly into WEKA.

Data type

Dataset has 40 attributes, inclusive of numeric and nominal data types.

Relation name: datasets_45152_82501_insurance_claims

Attribute	Data type	Attribute	Data type
months_as_customer	Numeric	umbrella_limit	Numeric
age	Numeric	insured_zip	Numeric
policy_number	Numeric	insured_sex	Nominal
policy_bind_date	Nominal	insured_education_level	Nominal
policy_state	Nominal	insured_occupation	Nominal
policy_csl	Nominal	insured_hobbies	Nominal
policy_deductable	Numeric	insured_relationship	Nominal
policy_annual_premium	Numeric	capital-gains	Numeric

Attribute	Data type	Attribute	Data type
capital-loss	Numeric	incident_location	Nominal
incident_date	Nominal	incident_hour_of_the_day	Numeric
incident_type	Nominal	number_of_vehicles_involved	Numeric
collision_type	Nominal	property_damage	Nominal
incident_severity	Nominal	bodily_injuries	Numeric
authorities_contacted	Nominal	Witnesses	Numeric
incident_state	Nominal	police_report_available	Nominal
incident_city	Nominal	total_claim_amount	Numeric

Attribute	Data type
injury_claim	Numeric
property_claim	Numeric
vehicle_claim	Numeric
auto_make	Nominal
auto_model	Nominal
auto_year	Numeric
fraud_reported	Nominal
_c39	String

Data size

Size	261 KB
Attributes	40
Number of instances	1000
Number of fraud cases	247
Number of legitimate cases	753

Data quality

- The dataset includes labelled data for **target** class ‘**fraud_reported’** identifying claim to be reported as Yes (Y) or No (N), all 1000 instances are labelled without any missing value.
- **No class imbalance issue** as we have 24.7% of instances out of 1000 reported to be a fraud and are enough for the classification.

Attribute	fraud_reported	Distinct values	2

Label	Count
Y	247
N	753

- The attribute **_c39 is a String type and has 0 values**.
- Out of the remaining 39 attributes i.e. excluding _c39, only three attributes have missing values

Attribute	Number of Missing values
collision_type	178
property_damage	360
police_report_available	343

Data preprocessing

We have 40 attributes, and not all of them are useful for our data analysis. This is a high number of features and we need to remove irrelevant features/attributes from the dataset and select important features from the dataset to use them for classification.

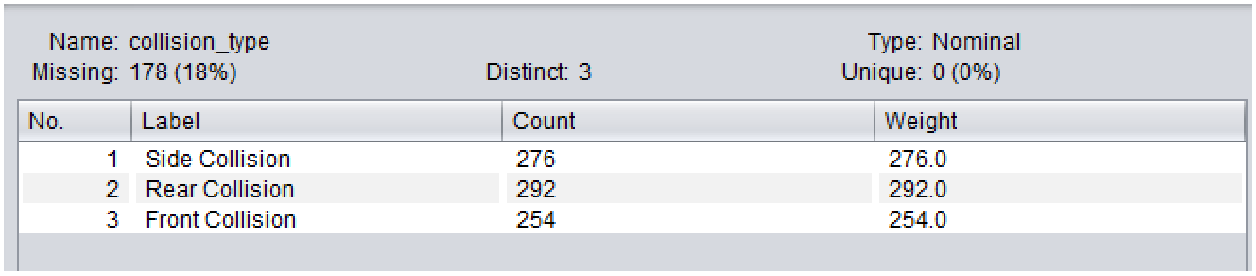
Step 1. Removing irrelevant features using '*remove' filter in Weka*

a. Removing **_c39 attribute** as it has no values
b. Removing the following attributes as they have a high number of unique values and cannot be distinctly associated with fraud detection
    i. **Incident_location** has 1000 unique values, equal to the number of instances, no information can be retrieved
    ii. **Policy_bind_date** has 995 unique value in a 1000 instance dataset, which is not relevant for classification
    iii. **Policy_number** as it is not helpful
    iv. **Incident_date and incident_city** attributes are similarly not relevant for data analysis

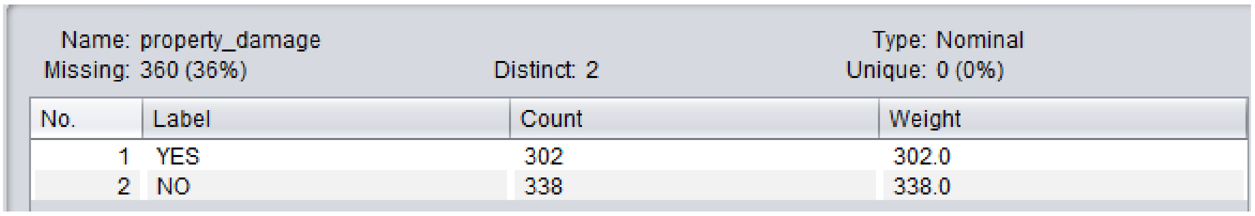
Step 2. Handling missing values using *ReplaceMissingValues* filter in preprocess tab of Weka to replace all missing values from attributes mentioned below using mean/modes from training data depending on the data type of attributes.

Attributes affected: **Collision_type, property_damage, police_report_available**

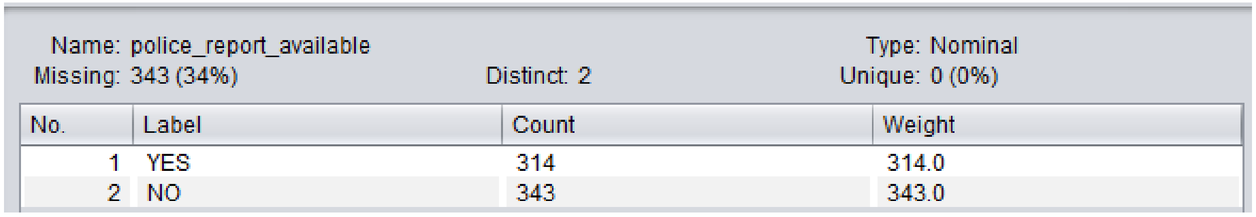
Before applying *ReplaceMissingValues* filter:


Name: collision_type
Missing: 178 (18%)　Distinct: 3
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	Side Collision	276	276.0
2	Rear Collision	292	292.0
3	Front Collision	254	254.0

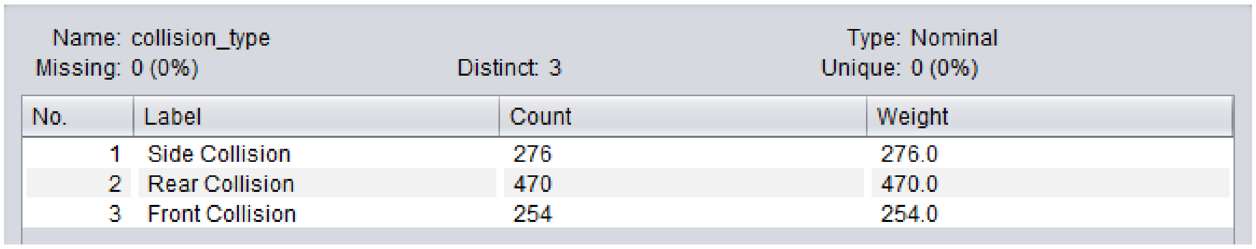

Name: property_damage
Missing: 360 (36%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	YES	302	302.0
2	NO	338	338.0

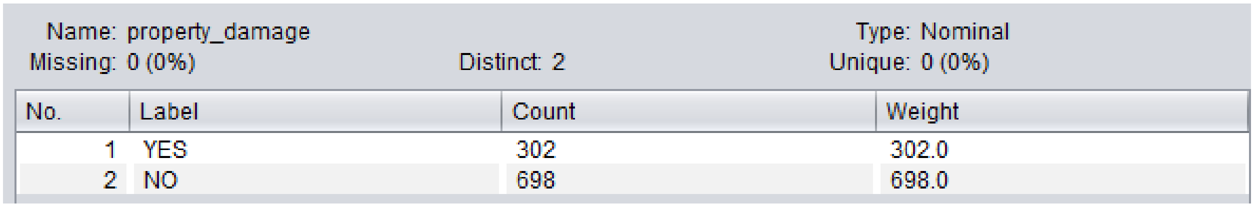

Name: police_report_available
Missing: 343 (34%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	YES	314	314.0
2	NO	343	343.0

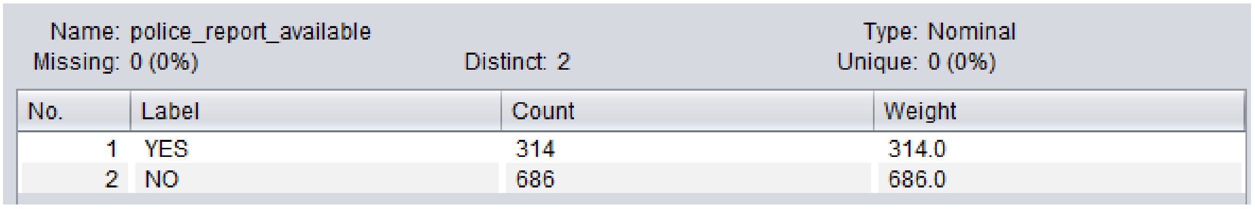
After applying *ReplaceMissingValues filter*:


Name: collision_type
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	Side Collision	276	276.0
2	Rear Collision	470	470.0
3	Front Collision	254	254.0


Name: property_damage
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	YES	302	302.0
2	NO	698	698.0


Name: police_report_available
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	YES	314	314.0
2	NO	686	686.0

This will help us with finding relevant features in the next step as deleting missing values can affect the dataset largely with most of the instances impacted.

Step 3. Saving an updated data file as a new ARFF file with 34 attributes and 1000 instances

Step 4. Selection of features

After removing unnecessary features and handling missing values, we still have 34 attributes. We will now use ***InfoGainAttributeEval*** filter with ***Ranker search method*** selecting top 10 features out of remaining features that are relevant to ***fraud_reported*** attribute. We will change **numToSelect** value in Ranker search method from -1 to 10, for retaining top 10 features.

Output dataset from feature selection:

We will use the output after applying the filter as the dataset for classification, saving this output to a new ARFF file to complete the data preprocessing. We will use this dataset hereafter for classification and evaluation. The dataset contains the top 10 attributes that provide maximum gain for output variable '**fraud_reported**', these attributes are as follows.

Attribute name	**Data type**
incident_severity	Nominal
insured_hobbies	Nominal
auto_model	Nominal
vehicle_claim	Numeric
total_claim_amount	Numeric
incident_type	Nominal
authorities_contacted	Nominal
property_claim	Numeric
injury_claim	Numeric
insured_occupation	Nominal

Data mining techniques

Data mining techniques are classified into supervised and unsupervised learning techniques. These techniques are highly effective in finding hindered patterns in the vast amount of data. These patterns can help us gain knowledge and insights from the data that will not be otherwise possible through human intervention or traditional techniques of analyzing and processing data (Rawte & Anuradha 2015). Supervised learning is the most used machine learning technique among the two. It is useful in training a model if we have labelled dataset available and the trained model can be used against new claims to classify them. While supervised learning is effective and has meaningful labels, it might be costly to label large input dataset. Support vector machine, K-nearest neighbours, decision tree, random forest etc. are a few examples of supervised learning/classification techniques. Unsupervised techniques or clustering are focused on instances finding anomalies in the data, this help in uncovering hidden patterns. They do not have any labels and are scalable to new claim patterns i.e. a new type of frauds. Since it is undirected, we might not always find patterns from selected features in a dataset. K-means clustering, DB Scan, hierarchical clustering are few clustering techniques used.

Nature of problem

We have insurance claims dataset which is already labelled for the target attribute or the outcome expected. As we have labelled dataset, we can easily apply the classifier to the dataset to build data models. This is classic **Binary Classification** problem where we need to classify the output attribute to **0 or 1** or **Yes or No** type of instance. The dataset has 10 attributes after preprocessing with 1000 instances and output attribute is '**fraud_reported**' attribute. We will use the cleaned dataset from data preprocessing step to build models on them and classify the 'fraud_reported' attribute in **Y or N** values.

Data mining technique adopted

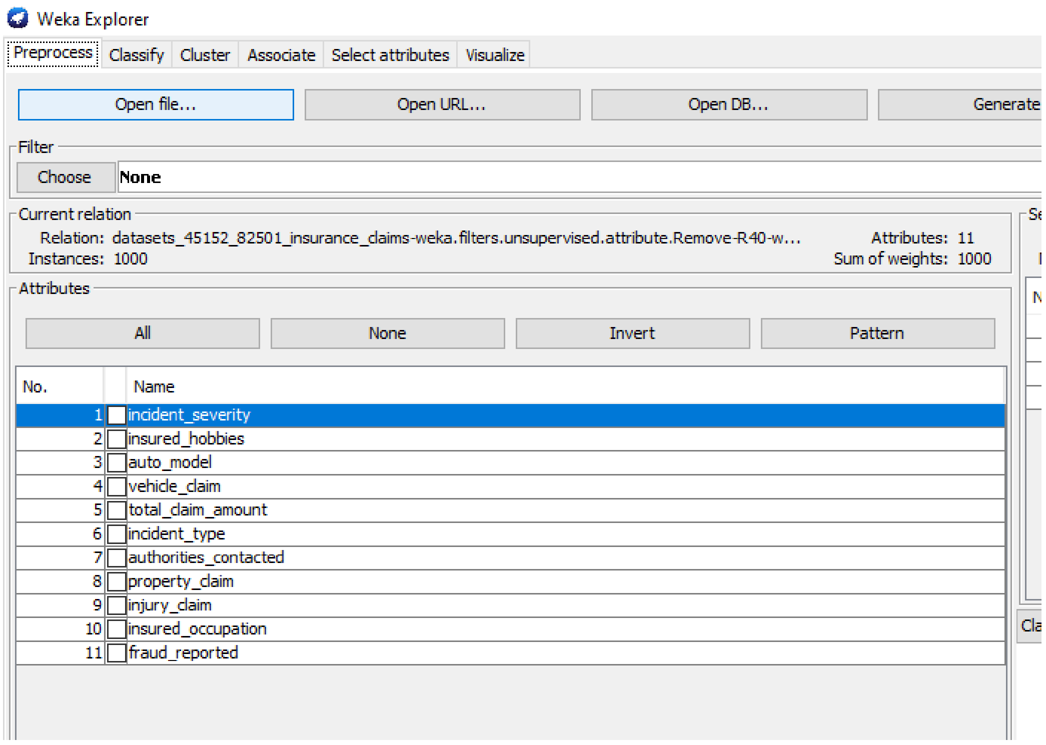
We will be using the **Decision tree (J48) classifier** on this dataset to classify fraudulent and normal claims. For comparison of performance, we will be using **Random Forest classifier** on the preprocessed dataset.

Decision Tree (J48) classifier:

A decision tree is a predictive modelling technique falling under supervised learning category. Decision trees can take discrete as well as continuous values, hence they are useful in classification and regression making them one of the most popular learning techniques (Shaikhina et al. 2019). The classifier is labelled as J48 under supervised learning in Weka.

Steps of a Decision tree classifier

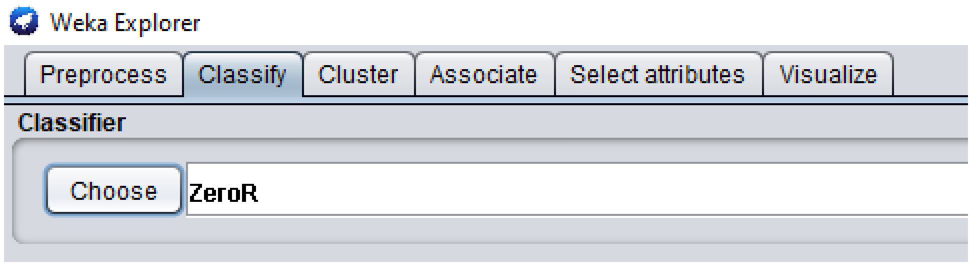
Step 1. Load preprocessed data in Weka



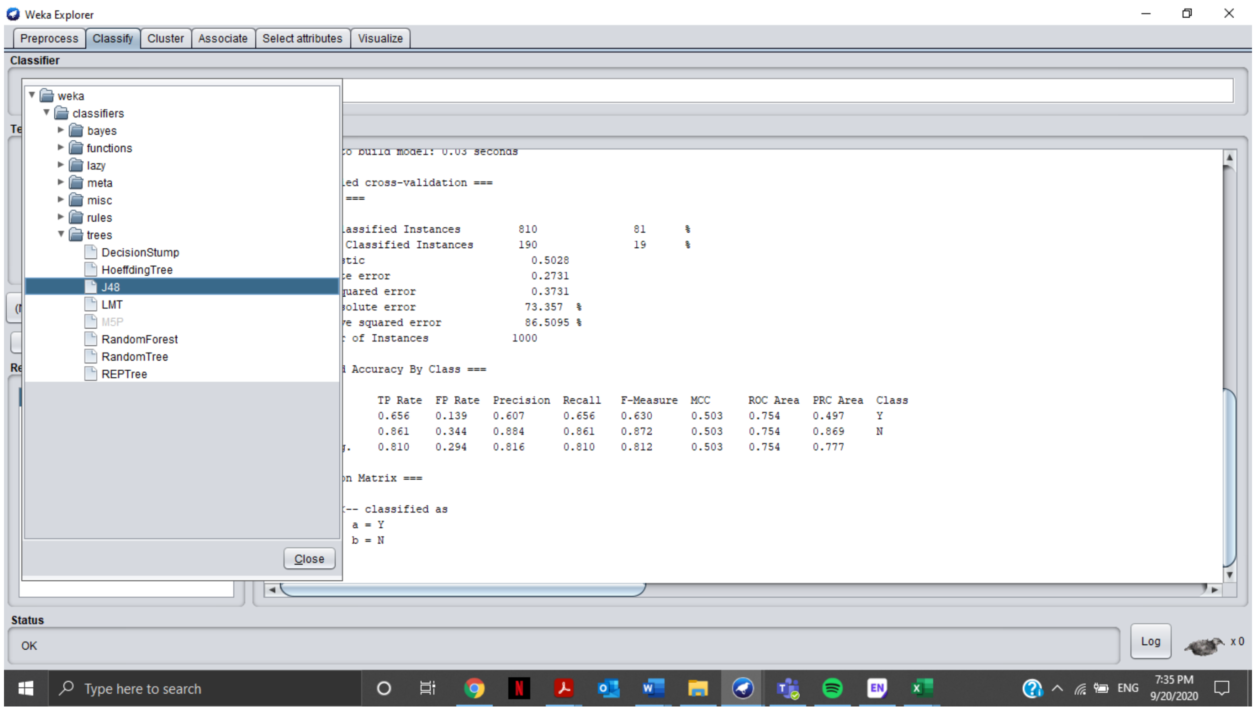
Relation: datasets_45152_82501_insurance_claims-weka.filters.unsupervised.attribute.Remove-R40-w...
Attributes: 11
Instances: 1000
Sum of weights: 1000

No.	Name
1	incident_severity
2	insured_hobbies
3	auto_model
4	vehicle_claim
5	total_claim_amount
6	incident_type
7	authorities_contacted
8	property_claim
9	injury_claim
10	insured_occupation
11	fraud_reported

Step 2. In Weka, next to preprocess tab, we have ‘classify’ tab available. We can choose the classifier of our choice from here. **J48 classifier** is located under ‘Trees’ folder in trees -> classifiers -> trees

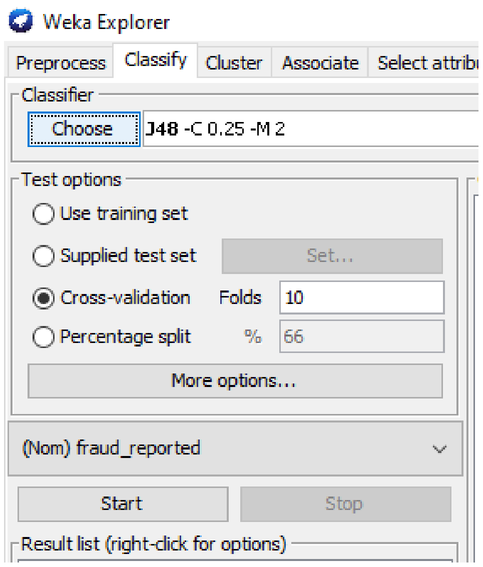


ZeroR



```
to build model: 0.03 seconds

ed cross-validation ===
===

lassified Instances         810               81      %
 Classified Instances       190               19      %
stic                          0.5028
e error                       0.2731
quared error                  0.3731
olute error                  73.357  %
ve squared error             86.5095 %
 of Instances              1000

d Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
      0.656    0.139    0.607      0.656    0.630      0.503    0.754     0.497     Y
      0.861    0.344    0.884      0.861    0.872      0.503    0.754     0.869     N
g.    0.810    0.294    0.816      0.810    0.812      0.503    0.754     0.777

n Matrix ===

<-- classified as
  a = Y
  b = N
```

Step 3. We select Cross-validation with default settings in ‘Test option’, selecting ‘**fraud_reported’** as output and then click on the ‘Start’ option to run the classifier.



J48 -C 0.25 -M 2

( ) Use training set
( ) Supplied test set
[✓ Cross-validation] Folds 10
( ) Percentage split % 66

(Nom) fraud_reported

Step 4. Analyze the result of classifier in classifier output window for class ‘Y’ and ‘N’

Pros and Cons

Pros of a decision tree classifier

- Decision tree classifier can work with continuous as well as discrete data

- Decision trees can map complex, non-linear heterogeneous dataset/input with output even if we cannot determine the correlation between attributes due to high complexity (Shaikhina et al. 2019)
- Highly transparent, hence easy to interpret and follow
- Data normalization is not required
- Less data preprocessing is required to use decision tree classifier

Cons of a decision tree classifier

- Decisions tree can be affected due to overfitting and pruning is required to reduce the impact
- The high dimensionality of data can affect the performance (Pal & Mather 2003)
- Minor changes in the dataset can affect the classifier performance
- Cost of building model may increase significantly in terms of time and money required for building a model for a dataset with a large number of attributes

Random forest classifier

Random forest classifier is used as the **counterpart algorithm** on our dataset to compare the performance of the decision tree against it. Random forest classifier is an advanced version of decision trees. Random forest classifier is an ensemble method. It involves the construction of multiple decision trees using bagging. Every time a new instance is passed, it is passed to each decision tree constructed. Each tree predicts the classification of the instance independently. These trees further vote for the corresponding class of that instance and a majority of votes decides prediction. Random forest outputs are less noisy and handle outliers better than single decision trees (Shaikhina et al. 2019).

Step 1. Load the preprocessed data in Weka if not loaded, and under classify tab, select 'Random Forest' classifier

Step 2. Next steps are similar to that of decision trees to apply the classifier and analyze the result. The random forest provides additional options that can be used to manipulate data further. Although we have cleaned the dataset, hence default settings will work in this case.

Step 3. Click start with 10-fold cross-validation selected in **Test options** window for output variable **'fraud_reported'**

Step 4. Analyze the results in the classifier output window

Evaluation of models

Decision tree and Random forest classifier

We are comparing the performance of J48/Decision tree classifier against random forest classifier. We have several metrics available to compare the classifiers, accuracy is a good measure although it is not always enough to compare and find better performing models. We have confusion matrix, classification report and different error measures that can help evaluate the performance for these two, that are discussed in detail further in this section.

We have used label **Y** as '**Fraudulent claim'** and **N as 'Legitimate claim'**

Performance metrics

Accuracy

Accuracy of a model is the total number of correct predictions divided by the total number of instances (1000 in our dataset). Accuracy can be the starting point to evaluate these models, but it is not the only thing we will be looking at. For our dataset, we got an accuracy of 81% for decision tree classifier, while almost similar accuracy of 80.9% was observed for the random forest as well. Both models can be said to be performing equally in terms of accuracy, depicting the difference of only 1 instance classified incorrectly.

Classifier			Accuracy
Decision tree	Correctly Classified Instances	810	81%
	Incorrectly Classified Instances	190	
Random forest classifier	Correctly Classified Instances	809	80.90%
	Incorrectly Classified Instances	191	

Error measures

Several error measures are also considered while evaluating the models, RMSE (root mean squared error) being used most. The error measure comparison is also not very conclusive, although random forest classifier is performing better in term of RMSE and root-relative error measure than decision tree classifier.

Error measure	Decision Tree	Random Forest
Mean absolute error	0.2731	0.2737
Root mean squared error	0.3731	0.3546
Relative absolute error	73.36%	73.52%
Root relative squared error	86.51%	82.22%

Confusion matrix

A confusion matrix is another metric that is often used instead of accuracy. The confusion matrix shows true positive (fraud cases detected as fraud cases), false positives (legitimate cases are detected as fraud), true negatives (normal cases detected as normal cases) and false negatives (fraud cases classified as normal/legitimate cases). Comparing the performance, we can see that **decision tree classifier** performed better to detect fraudulent cases, although it **raised a higher number of false alarms** than the **random forest classifier**. A false alarm may be costly for the firm as they take a human as well as financial resources and increase the cost of the investigation.

Decision Tree	a	b	classified as
	162	85	a = Y
	105	648	b = N

Random Forest	a	b	classified as
	106	141	a = Y
	50	703	b = N

Precision, Recall and F-measure

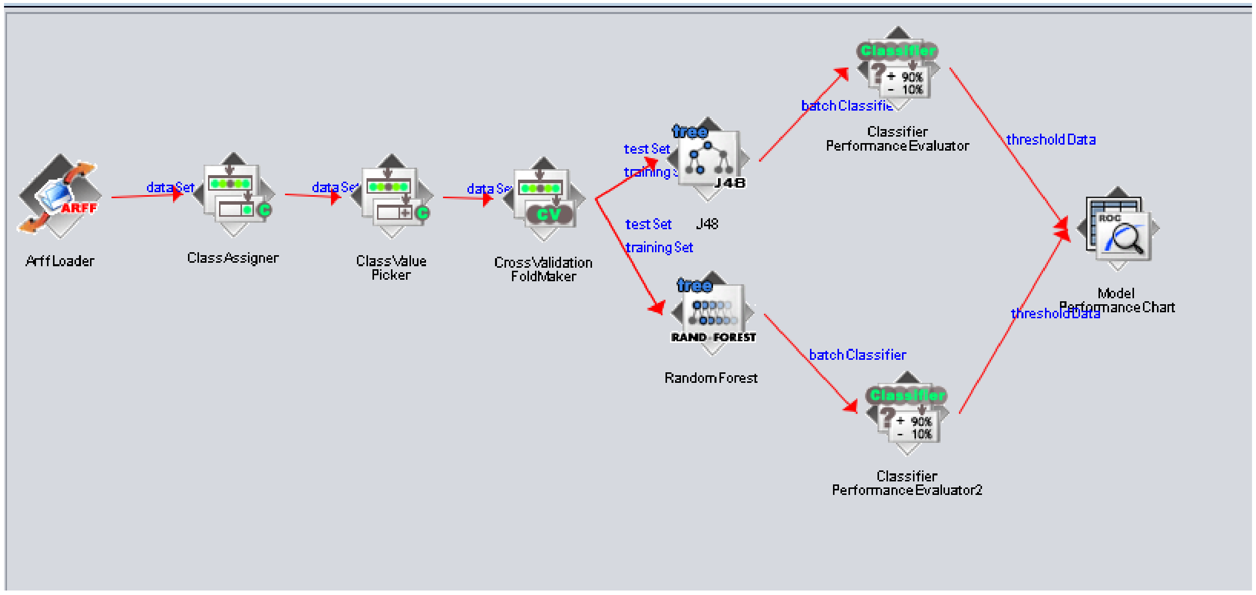
We will now compare precision, recall and f-measure metrics for our classifier along with True positive (TP) and False positive (TP) rates. Random forest shows better precision (0.679) for positive cases, i.e. fraudulent cases than decision tree classifier (0.607). When it comes to recall value, the decision tree has performed better with a recall of 0.656 while random forest shows a recall 0.429 for positive class. Although, the random forest is much better at predicting the normal cases, making them less prone to a false alarm that may incur extra cost insurers would not want to spend. FP rate for decision tree classifier is almost double than that of random forest classifier, making it inefficient for our purpose.

Decision Tree		TP Rate	FP Rate	Precision	Recall	F-Measure	Class
		0.656	0.139	0.607	0.656	0.63	Y
		0.861	0.344	0.884	0.861	0.872	N
	Weighted Avg	0.81	0.294	0.816	0.81	0.812	

Random forest		TP Rate	FP Rate	Precision	Recall	F-Measure	Class
		0.429	0.066	0.679	0.429	0.526	Y
		0.934	0.571	0.833	0.934	0.88	N
	Weighted avg	0.809	0.446	0.795	0.809	0.793	

Analyzing ROC using Weka Knowledge flow

Step 1. Building the estimator in Weka for decision tree classifier and random forest classifier



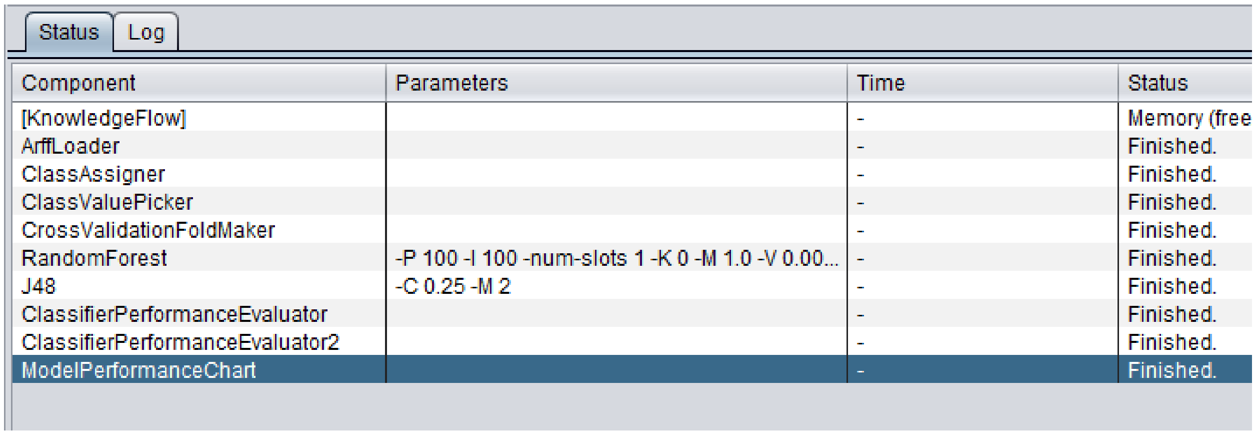
Step 2. Select ARFF loader and right-click on it, then select option 'configure' for loading filtered dataset with 10 attributes and output variable. Alternately, CSV loader can also be used for loading CSV files

Step 3. Select ClassAssigner, right-click on it to configure class attribute, i.e. prediction attribute 'fraudulent_reported'

Step 4. In class value picker, we configure to the positive output value, i.e. typing 'Y' for 'fraudulent_reported' attribute in the class value window

Step 5. In Cross-Validation fold maker, we select 10-fold cross-validation by changing value 'Number of folds' in the configuration window

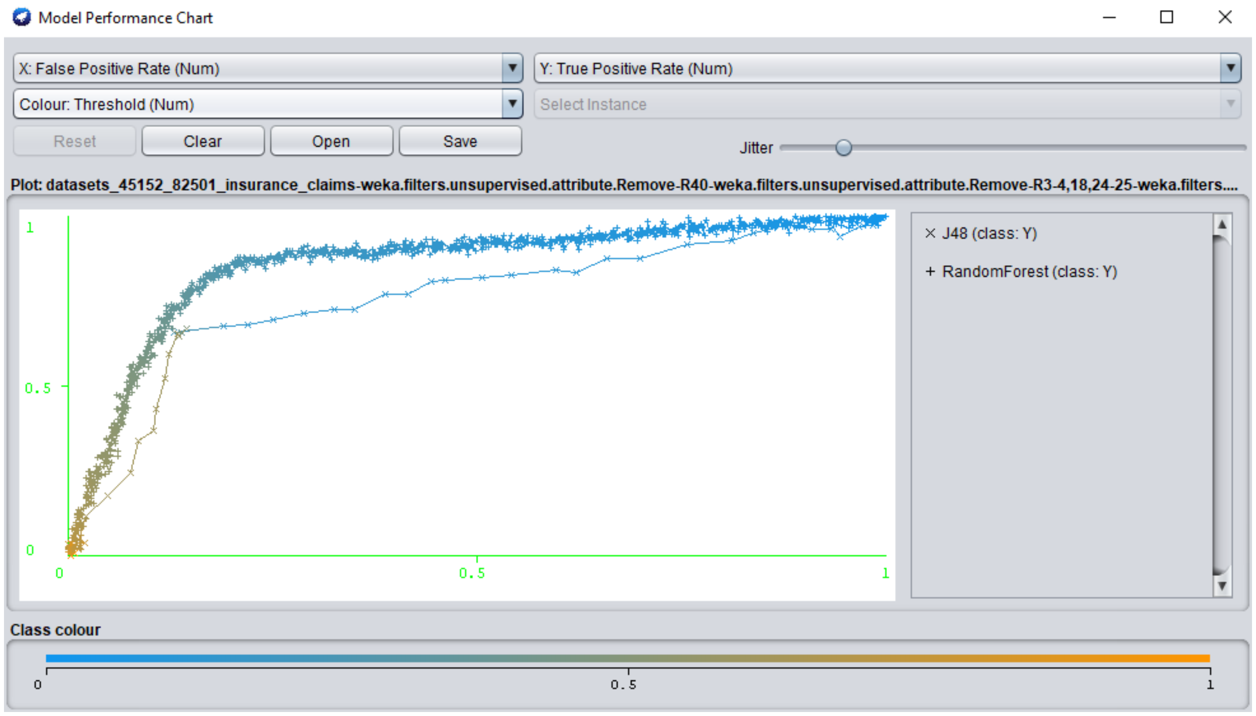
Step 6. Start the ARFF loading by selecting the play option in the left top corner of the Knowledgebase window and check if the status is finished



Component	Parameters	Time	Status
[KnowledgeFlow]		-	Memory (free
ArffLoader		-	Finished.
ClassAssigner		-	Finished.
ClassValuePicker		-	Finished.
CrossValidationFoldMaker		-	Finished.
RandomForest	-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.00...	-	Finished.
J48	-C 0.25 -M 2	-	Finished.
ClassifierPerformanceEvaluator		-	Finished.
ClassifierPerformanceEvaluator2		-	Finished.
ModelPerformanceChart		-	Finished.

Step 7. Visualizing the ROC curve for the classifiers

If we analyze the ROC curve, we can see that the random forest curve is well above decision tree (J48) classifier. True positive (fraud claims detected as fraud) values are on the Y-axis and False positive (legitimate claims detected as fraud) on X-axis. We can observe a lower false-positive rate for random forest classifier while it also performs comparatively better in detecting over cases.



Plot: datasets_45152_82501_insurance_claims-weka.filters.unsupervised.attribute.Remove-R40-weka.filters.unsupervised.attribute.Remove-R3-4,18,24-25-weka.filters....

Concluding the better Model

After checking different metrics and observing the ROC curve, we can conclude that the **Random forest classifier has outperformed decision tree (J48) classifier and showed better quality.** Even though we observed similar accuracies and error measure values. Although to detect positive/fraudulent cases at a lower cost, random classifier performed better than decision tree classifier for our insurance claim dataset.

Improvised algorithm

I am proposing to stack Adaboost classifier with a decision tree to improve the performance of the algorithm. Adaboost is a meta-estimator that fits a classifier on the given dataset and proceeds to fit more copies of the classifier to handle difficult cases in the dataset (Scikit learn, 2020). Adaboost uses a decision tree by default as base estimator if we do not specify any other classifier. In the proposed algorithm, we will be using decision stump (decision tree with max dept = 1) as the base estimator for Adaboost and then apply decision tree as meta estimator to improve the performance.

Algorithm: Stacking Adaboost with meta decision trees

Pseudo-code/Python

Step 1: From sklearn library, *import Adaboost classifier, Decision tree classifier and stacking classifier*

Step 2: Create a estimators list with *Adaboost and decision tree classifier*

```
estimators = [
    ('dt', DecisionTreeClassifier(random_state=0)),
    ('ada', AdaBoostClassifier(n_estimators=100, random_state=0))
]
```

Step 3: Pass the estimators list to Stacking classifier, using decision tree classifier as final estimator

```
clf = StackingClassifier(
    estimators=estimators, final_estimator=DecisionTreeClassifier()
)
```

Step 4: Fit the train and test to the *clf* created in step 3 using function *clf.fit(X_train, y_train)* and review the classification report to evaluate performance.

Results in Weka

We used the enhanced algorithm in Weka to review the performance of the algorithms and we found that the accuracy, precision, recall and f-measure were significantly improved showing the accuracy of 85.4% and reduced errors as well. The results were as follows

Classifier			Accuracy
Decision tree	Correctly Classified Instances	810	81%
	Incorrectly Classified Instances	190	
Stacked Adaboost with meta decision trees	Correctly Classified Instances	854	85.40%
	Incorrectly Classified Instances	146	

Error measure	Decision Tree	Stacked
Mean absolute error	0.2731	0.2074
Root mean squared error	0.3731	0.3226
Relative absolute error	73.36%	55.70%
Root relative squared error	86.51%	74.80%

Decision Tree		TP Rate	FP Rate	Precision	Recall	F-Measure	Class
		0.656	0.139	0.607	0.656	0.63	Y
		0.861	0.344	0.884	0.861	0.872	N
	Weighted Avg	0.81	0.294	0.816	0.81	0.812	

		TP Rate	FP Rate	Precision	Recall	F-Measure	Class
Stacked Adaboost with meta decision tree		0.887	0.157	0.65	0.887	0.75	Y
		0.843	0.113	0.958	0.843	0.897	N
	Weighted avg	0.854	0.124	0.882	0.854	0.861	

Conclusion

We reviewed the performance of decision tree classifier and random forest classifier using WEKA on insurance claims dataset in this report. The dataset had enough data points for fraudulent claims (24.7%) which resulted in balanced class and ensured classifier will not be biased towards legitimate claims. In a practical scenario, even though the losses the are in billions of dollars, the total number of illegitimate claims might be far lower than the legitimate claims, that can cause the classifier to be aligned towards legitimate cases, thus reducing the efficiency of classifiers. This issue can be handled using data sampling techniques e.g. SMOTE. Evaluation of the classifiers concluded that the random forest classifier has shown better quality than the decision tree classifier, raising a lower number of false alarms. The decision tree classifier can further be improvised to increase the accuracy using boosting algorithms. Classifiers are very useful as they learn historical data and predict the fraud pattern. However, insurers are introducing new policies and fraudsters keep finding new ways to gain illegitimate benefits. For the classifier to maintain the quality, we need to retrain the models periodically to identify such patterns. Future of fraud detection systems might also involve the use of advanced deep learning algorithms that will run much faster and have a greater impact.

References

Kaggle 2019*, Insurance claim*, retrieved 11 September 2020, <https://www.kaggle.com/roshansharma/insurance-claim>

Bauder, RA & Khoshgoftaar, TM 2017, 'Medicare Fraud Detection Using Machine Learning Methods', in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 858-65.

Bhowmik, R 2011, 'Detecting auto insurance fraud by data mining techniques', *Journal of Emerging Trends in Computing and Information Sciences*, vol. 2, no. 4, pp. 156-62.

Harjai, S, Khatri, SK & Singh, G 2019, 'Detecting Fraudulent Insurance Claims Using Random Forests and Synthetic Minority Oversampling Technique', in *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, pp. 123-8.

Pal, M & Mather, PM 2003, 'An assessment of the effectiveness of decision tree methods for land cover classification', *Remote Sensing of Environment*, vol. 86, no. 4, pp. 554-65.

Rawte, V & Anuradha, G 2015, 'Fraud detection in health insurance using data mining techniques', in *2015 International Conference on Communication, Information & Computing Technology (ICCICT)*, pp. 1-5.

Rayan, N 2019, 'Framework for Analysis and Detection of Fraud in Health Insurance', in *2019 IEEE 6th International Conference on Cloud Computing and Intelligence Systems (CCIS)*, pp. 47-56.

Shaikhina, T, Lowe, D, Daga, S, Briggs, D, Higgins, R & Khovanova, N 2019, 'Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation', *Biomedical Signal Processing and Control*, vol. 52, pp. 456-62.

Scikit 2020, *AdaBoost Classifier,* retrieved 15 September 2020, <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html>