Using Big Data to Identify Insurance Fraud

Jefferson Van Buskirk

Lehigh University
CSE449 Final
jef@lehigh.edu

ABSTRACT

The world of insurance is riddled with fraud. A 2022 study by The Coalition Against Insurance Fraud (3) indicates that insurance fraud can cost U.S. consumers $308.6 billion yearly (1). A 2017 study by Verisk found that auto insurers lose $29 Billion annually (2). This loss is passed on to insurance consumers, representing 14% of their personal premiums. Auto Insurance fraud takes two forms: soft insurance fraud and hard insurance fraud (4). Auto insurance fraud is clearly an expensive problem for insurance companies, however each individual instance of fraud is costly to investigate, often more costly than simply paying the claim. In addition, investigations into individual customers damage customer loyalty and overall brand image, large issues in a highly competitive market (2).

To combat the problems of insurance fraud, many companies are looking to big data models to predict fraud without directly investigating customers. These models use data regarding policies, demographics, the vehicles in question, and customer demographics to make predictions about the legitimacy of policies and claims. These models can be used when pricing policies for customers, incorporating the risk of fraud into their premium correctly, and to detect when fraudulent claims are made. When a fraudulent claim is detected,the insurance company can then decide to investigate.

In the following paper I outline my exploration of auto insurance fraud in India. I used five files with data regarding policies, vehicles, claims, customer demographics, and whether or not they were reported fraudulent. I preprocessed this data to prepare it for different types of models, used different models to determine trends within the data and explore the legitimacy of fraud detection in data, and created software to determine how to accurately price a policy and find the expected value of an investigation. In conjunction, these tools can provide tremendous value to an insurance company by accurately pricing policies, detecting fraudulent claims, and reducing the number of investigations that lose money.

All data, Weka scripts, and PySpark scripts available at https://gitlab.com/lolitsjef/using-big-data-to-identify-insurance-fraud

0 TOOLS

For this project I used three tools: Microsoft Excel, Weka Knowledge Flow, and PySpark. Microsoft Excel is a spreadsheet editor that handles CSV files. This was helpful for preprocessing the data. Weka is an open source data mining software that I used to train and evaluate different classification models. Knowledge flow is a visual programming setup with modules for importing, manipulating, classifying, and evaluating data and models. PySpark is the Python API for Apache Spark: an open source data processing framework. I used PySpark for the “fraud fee” calculation of individual policies and expected value of investigation of individual claims.

1 DATA

The data for this project came in 5 csv files: Policy, Demographics, Claim, Vehicle, and Target with a total of about 41 columns. In total there were 28,837 claims within the data set. The first part of this project was understanding what information each file had, the types of information (string, int, nominal) and preparing the data for my first models. Below is a description of each file, for information on every column, see AttributeInformation.pdf in the data folder.

Policy Information

The first file in the set contains policy information. This information is created when a customer is purchasing insurance. Important information includes customer and policy ID’s, dates of coverage, the state, policy limit (single and umbrella), deductible premium, as well as information regarding payment method. Trends within this data can be used to create a “fraud fee” based on the likelihood of fraud.

Demographic Information

The second file in the set contains customer demographic information. This includes age, zip code, gender, education level, occupation, hobbies, insured hobbies, and capital gains/losses. This information is also available when creating a new policy and can thus be included in calculating a “fraud fee”. Ethics issues regarding demographic trends certainly arise when predicting a customer’s fraudulent behavior based on demographic information, these issues are discussed in section 3.

Claim Information
The third file in the set contains information regarding claims. Information regarding the accident include the date, time,location, type of accident, type of collision, severity of collision, number of vehicles, property damage, bodily injuries, witnesses, if a police report exists, and which authorities were contacted. In addition to the information regarding the incident, there is information regarding the amount of money claimed for injuries, property damage, vehicle damage, and the total amount. This data is known after a claim is filed; trends can be used to detect insurance fraud.

Vehicle Information
This file contains each customer's vehicle in the policy. The structure of the file is different from the other files, which caused issues during preprocessing, but contains the make, model, and year for each vehicle.

Target
The last file contains whether or not a claim was reported as fraudulent. This is used to train the models as well as evaluate the performance of each model.

2 PREPROCESSING

Before I could use the data, I needed to prepare it for both Weka and Pyspark. The first portion of this preprocessing was to combine the files into two files, one for information known when creating a policy, and one for all information available after a claim is made. To do this I first opened each file in Excel and sorted by customerID. Working with Excel is much easier than Weka as I could select elements more freely, however Excel gets very slow as the total amount of data increases. With 28,837 rows and 41 columns, the data set started with 1,182,317 elements. After sorting on customerID in each file I launched Weka.Weka allowed me to join each file on the CustomerID, ensuring that each record was properly merged. Weka creates a new column, customerID2, when doing this which I needed to remove (using Excel) before joining each next file.

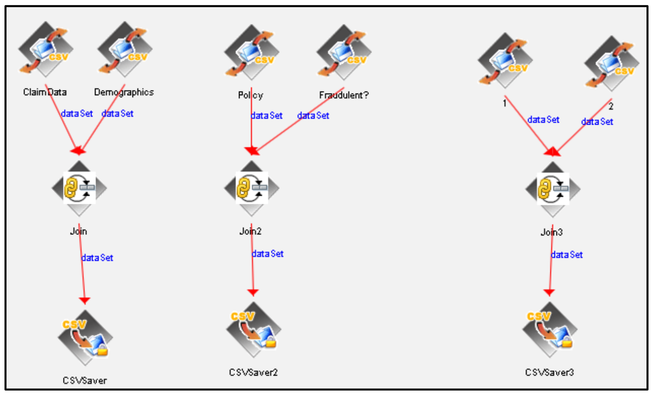
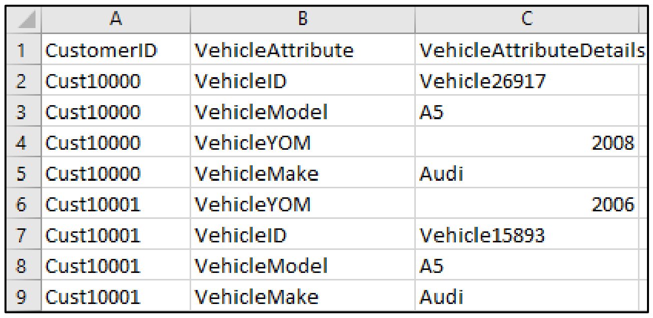


Figure 1: Weka flow for joining files into singular csv

One issue arising during this step was the organization of the vehicle information file. While every other file had only one attribute per column, the vehicle file put 4 attributes into one column "VehicleAttribute" and the corresponding value in "VehicleAttributeDetails" as shown below. The attributes were not sorted in any particular order, making it very difficult to read.



	A	B	C
1	CustomerID	VehicleAttribute	VehicleAttributeDetails
2	Cust10000	VehicleID	Vehicle26917
3	Cust10000	VehicleModel	A5
4	Cust10000	VehicleYOM	2008
5	Cust10000	VehicleMake	Audi
6	Cust10001	VehicleYOM	2006
7	Cust10001	VehicleID	Vehicle15893
8	Cust10001	VehicleModel	A5
9	Cust10001	VehicleMake	Audi

Figure 2: Vehichle Information data before preprocessing

To solve this I used a layered sort on Excel, sorting by VehichleAttribute and then CustomerID. This grouped all of the "VehicleAttribute" while retaining the "CustomerID" ordering. I then copy/pasted the "VehicleAttribute" into four separate columns representing each attribute. After fixing this issue, the vehicle information file could be properly joined with the rest of the data.

After aggregating the data to a single file, I removed string characters such as " and '. These characters cause nominal (categorical) values to be read as strings. I also grouped all null values under "?" so that Weka could properly read them. From here, the single file (FINALDATA.csv) could be loaded directly into Weka and used properly. All of the preprocessing steps occurred twice, once to create FINALDATA.csv which has all of the available information, and once to create FINALDATANOCLAIM.csv which has all of the information that would be available before a claim is made.

3 ETHICS

During the modeling portion of this project I performed my classification using the data with and without ethics. Including ethics raised the accuracy of the models by 4%, a substantial difference when the ZeroR baseline is 73%. The ethics behind using demographic information in big data models is a growing conversation as its relevance and impact are constantly growing. As more systems are run through AI systems, the importance of data ethics becomes more important as the consequences will be felt more often. The data I am using has found trends within the demographic portion of the data. These trends can be the reality of the situation, or the algorithm could be biased from biases in the data. Demographic data is seen as reasonable when zoomed out. According to the WEF, Finland is the world's safest country and Colombia is the most dangerous (4). Most people would not have a problem using this statistic in an algorithm to determine the price of auto insurance in these two countries. It makes sense that Finland auto insurance is cheaper as there is less violence and theft. It makes sense that Colombian auto insurance would be more expensive as there is more violence and theft. As the data zooms in, it feels more individual and becomes more of an ethical issue. If a health insurance company wants to charge a person more because they live in a "dangerous neighborhood", this is seen as discrimination and is an ethical issue. Somewhere between the zoomed out generalization and the zoomed in personal data, there is a line. Finding this line is different for every project, and important to consider when using the demographics of individuals.

The core ethical challenges faced by big data are the boundary of public and private good, privacy concerns, transparency of data collection, equity of access, and informed use of collected information (5). While all of these are pressing issues, my project mostly relates to transparency of data collection and informed usage of collected information.

Transparency of Data Collection

Transparency of data collection has two portions: informing individuals that they are being collected, and informing users of the data how the data was collected. While the algorithms used in data modeling are unbiased, as they do not understand the concept of human demographics, the data used to train these models often has underlying biases. If the data has an underlying bias, the model will read this bias as a trend, and thus generate biased results. This bias can come from human bias, but does not necessarily have to.

On November 27, 2022 I was in a car accident in which my car wsa hit from behind. There wasn't too much visible damage to my car, but after taking it to a mechanic the estimate for repair was around $2,500; a substantial amount. After discussing with the person who hit me, they decided they preferred to pay me out of pocket rather than getting insurance involved. Doing so prevented their insurance company from finding out about the accident, thus they received no data from the situation. Had the person who hit me decided to use insurance, their company would have known about the accident and marked the driver as a more risky person likely to get into an accident. Using insurance information to create a model to predict a driver's driving performance may lead to the model biasing towards wealthier drivers. As wealthier drivers are less likely to call their insurance, the insurance data lacks wealthy drivers' accidents. There is no human prejudice in this system, but the data would still be biased against poorer drivers, as there is more available data for them.

The data used in this project only contains data on customers who ended up filing a claim. All of the data is accurate for these customers as they filled it out themselves, except for the most important column: ReportedFraud. This column is the most important as it is what the model is trying to predict. How was this data collected? If only half of the customers were investigated, then there will certainly be fraudulent claims marked as legitimate. If the insurance company exhibited prejudice towards specific demographics (race, gender, financial) they likely would have investigated those people with those more, thus finding more fraud. This bias would manifest itself into a trend in the actual model, leading to more accurate results in the classification, but not in the real world.

The data provided no explanation of how it was collected. I am unsure if customers knew that their data was being collected. I am also unsure of the process of determining which claims should be investigated. This data is not very transparent and thus would not be a reliable source to determine the fraudulent behavior of real customers.

Informed Usage of Data Collection

In addition to informing customers that their data is being collected, it is also important to inform customers of how this data will be used. Customers will be more or less likely to share information when they know what the information is being used for and who will have access to the information. Customers who shop online understand that the shop they are purchasing from will remember their purchasing behavior, similar to how a real shop may have "regulars". Customers often enjoy when sites remember who they are as they feel more connected and welcome to the brand behind the website. The difference between online shopping and in store shopping is that all of the data is saved and often sold to other sites. Coffee shop customers enjoy being regulars, but would feel violated if another company advertised to them based on their coffee habits. In real life, this would be considered sharing private information, but online this practice is commonplace. Informing customers how their data will be used, is just as important as telling them their data is being collected. Many sites save information purely for the purpose of selling it, which most customers would not opt into. By transparently explaining what will happen with the data being collected, users feel safer. Recent regulations have required many sites to switch from 'opt in' to 'opt out' data collection, requiring sites to clearly explain what happens with collected data before giving users the option to 'opt in'. These regulations are the start of preserving internet privacy, and making the space safer for all users.

In addition to not knowing whether or not customers were informed of their data being collected, the data set does not explain the customers' knowledge of the usage of the data. This issue combined with the possibility of biases in the data set makes this data unsuitable for real life applications, however it is a fine proof of concept for my project. The systems built in my project are suitable for real life applications, given they are trained on a data set that is generated using proper ethical discretion. During classification I will explore the differences in accuracy when omitting demographic data, however I decided to include demographic data in my fraud investigation expected value and fraud fee calculation scripts as the increased accuracy led to better results.

4 CLASSIFICATION MODELS

After preparing the data I generated models for classifying whether or not a given claim/policy was fraud. I used two data sets, one with all of the data, which would be available after an accident occurs and a claim is made, and a set with only the data available at the time of creating a policy. To explore the effect of demographics, I also created these same sets without the demographic information. For classification I used five models: ZeroR, OneR, J48 Tree, Random Forest, and Naive Bayes.

ZeroR

ZeroR uses no rules in classification. When reading the data set it finds which classification is the most common and always guesses that a given record is that classification. This is helpful because it provides a baseline of how well other classification models need to perform. A model performing at 91% accuracy sounds very accurate, but if the ZeroR accuracy is 90% the model is not doing very much. The ZeroR for this data set always chose that a given record was not fraudulent. This had an accuracy of 73%.

OneR

Similar to ZeroR, OneR chooses one variable and judges all classification on this variable. OneR allowed me to both see the most important variable and provided another baseline for the larger models. When first running OneR it output an accuracy of 93% using zipcode as its one rule. Zipcode was interesting because I loaded it as a nominal variable (categorical) when it should be considered a geological variable. The locations of two zipcodes correspond to each other rather than being independent. Types of colors have nothing to do with each other and cannot be ordered, but two zipcodes can be close or far away from each other. Interpreting as a nominal variable created a rule with 993 different branches and had a 93% accuracy. This accuracy seemed rather unreasonable, performing at the same level as J48 and Random Forest (when using zipcode). I suspect that there was some level of bias in the reporting of data that had to do with zipcode. There are likely more fraud investigation agencies within these zipcodes leading to more investigations or prejudiced bias that caused the insurance company to investigate these areas more. Interpreting zipcode as a nominal variable also caused big issues when one-hot encoding the data for the scripts, so I decided to remove the variable from my models.

J48 Decision Tree

A J48 decision tree creates a flowchart using the data and ends at the classification. To classify a given record, one needs to simply follow the flowchart using the data within the record and they will end up at the model's classification. The model puts more important features at the top of the tree and less important features near the bottom. Inspecting the output decision tree allowed me to explore which variables had larger impacts on the classification. The J48 tree also had the highest accuracy after the one-hot encoding and was used for the scripts explained in section 6.

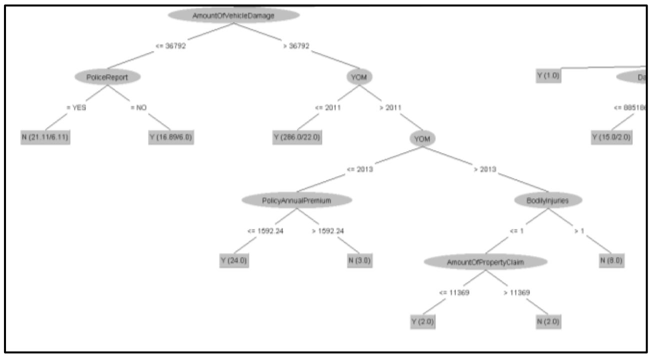


Figure 3: Lower portion of J48 decision tree generated in classification

Random Forest

The random forest classification model uses a set of decision trees to create a more accurate model of the data. The idea behind a set of trees is that some models will overestimate or underestimate which variables are important and split variables at different intervals, but averaging these mistakes together will create a more accurate model. The resulting model still takes the form of a decision tree, however the creation of this tree has many intermediate trees that are averaged together. This model was the most accurate in Wekaand I used it for attribute ranking, however it struggled in the fraud investigation expected value and fraud fee calculation scripts after the one-hot encoding was performed.

Naive Bayes

Naive Bayes is a classifier built around probability. For each variable it runs the bayes theorem, shown below, and calculates the probability of classification based on the value in that variable. This model is very fast compared to the decision tree models, however it works under the assumption that every variable is independent of every other variable. This assumption is generally incorrect, it is incorrect for this data set, but allows the model to be built much faster. For larger data sets, this speed increase can take precedence over accuracy.

5 CLASSIFICATION RESULTS

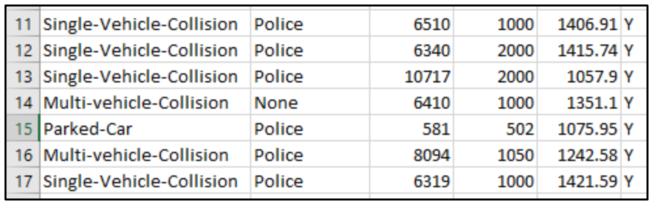
These five models provided both an understanding of the trends within the data and the most influential attributes.

Attribute Results

To understand the most important attributes I used an attribute selection module in Weka with a Random Forest classifier. Random Forest had the highest accuracy of all the models, and thus would have the best insight into the important attributes. In order, the most important attributes were

1. "TypeOfIncident"
2. "AuthoritiesContacted"
3. "AmountOfInjuryClaim"
4. "Policy_Deductible"
5. "PolicyAnnualPremium"

These are all attributes that directly relate to the incident or represent some amount of money. By importing the output from the attribute selection module into Excel (AttributeImportances.xlsx) I was able to group all of the fraudulent ranges together to understand what values were more likely to point towards fraud.



11	Single-Vehicle-Collision	Police	6510	1000	1406.91	Y
12	Single-Vehicle-Collision	Police	6340	2000	1415.74	Y
13	Single-Vehicle-Collision	Police	10717	2000	1057.9	Y
14	Multi-vehicle-Collision	None	6410	1000	1351.1	Y
15	Parked-Car	Police	581	502	1075.95	Y
16	Multi-vehicle-Collision	Police	8094	1050	1242.58	Y
17	Single-Vehicle-Collision	Police	6319	1000	1421.59	Y

Figure 4: Sorted attribute data in Excel

The first two variables, "TypeOfIncident" and "AuthoritiesContacted", fraudulent claims were almost always a "Single-Vehicle-Collision" with "Police" or "None" contacted, while non-fraudulent claims were "Multi-vehicle-Collision" with "Police", "Fire", or "Ambulance" contacted. Intuitively it makes sense that a collision with multiple vehicles is less likely to be fraud as multiple insurance companies will likely be involved as well as another party. Incidents with one vehicle, "Single-Vehicle-Collision", will only have the one insurance company which is easier to defraud than multiple. Assuming people who commit insurance fraud are reasonable people, they also probably do not want to involve other people and risk hurting someone else in a "Multi-vehicle-Collision". "Police" is the most common answer, representing about 29% of all records (8324/28837) and thus less significant as the other answers. It makes sense that claims with "None" contacted are more likely to be fraudulent as contacting the authorities requires a real accident to take place. If only the police are contacted, this usually means there was an accident, but not a severe enough one to call an ambulance and thus the fraud is probably exaggerating the costs of the incident. If an ambulance or fire truck is called, the accident is certainly more severe and less likely to involve fraud.

The next three variables; "AmountOfInjuryClaim", "Policy_Deductible", and "PolicyAnnualPremium"; relate to the amount of money in the transaction. Fraudulent "AmountOfInjuryClaim" ranges from 0 to 21,330 with an average of 7,917. Legitimate "AmountOfInjuryClaim" ranges from 0 to 21,450; a similar range; but with an average of 7,123. This $794 difference likely comes from inflating injury damages. In the aggregate, fraudulent claims have $7,123 of real injuries, with $794 fraudulent claims on top. Injuries are harder to outright fake, but inflating them is reasonably easy. "Policy_Deductible" has the most staggering difference with the fraudulent range from 0 to 2000 and an average of 1,122 while legitimate claims have a range of 0 to 21450 and an average of 7,122. This difference certainly points towards fraud as a fraudulent claim with a lower deductible will pay out far more than one with a higher deductible. Customers with higher deductibles likely have more coverage or lower premiums, however they will not be able to claim as much in an accident, the exact opposite goal of auto insurance fraud. "PolicyAnnualPremium" showed no significant differences between fraudulent and legitimate claims on their own, but likely are involved with trends in conjunction with the other attributes.

Model Results

Running each of the 5 classification models on the data before and after a claim, with and without demographics generated the results below.

Data Availabilities
A - Creating Policy
B - Creating Policy (No Demographics)
C - After Claim
D - After Claim (No Demographics)

	ZeroR	OneR	J48	Random Forest	Naïve Bayes
A	73.0025	79.803	86.1389	90.6887	77.8263
B	73.0025	74.7052	82.5635	85.3551	72.3332
C	73.0025	79.803	88.23	92.9324	80.3614
D	73.0025	74.7052	87.2174	91.7811	76.6542

Figure 5: Classifier accuracy table

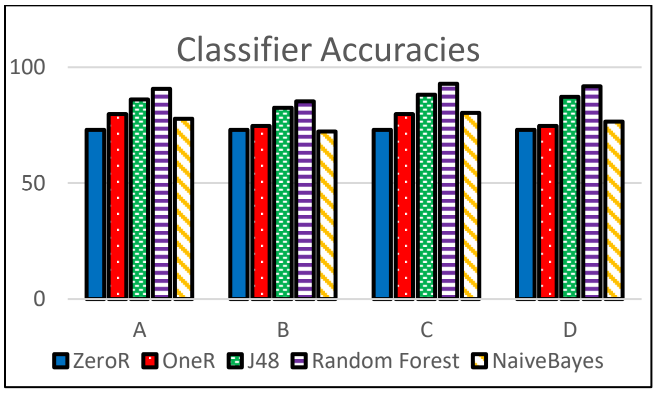
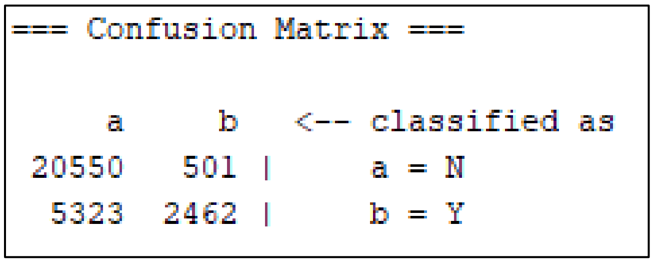


Figure 6: Classifier accuracy chart

ZeroR and OneR Results

The ZeroR baseline for the data set was 73%. Each classifier will be judged against this baseline, as doing no classification still provides an accuracy of 73%. OneR had an accuracy of 79.8% with demographic information and an accuracy of 74.7% without demographic information. The OneR chosen with demographic information was "CapitalLoss". There was not a simple boundary for "CapitalLoss". When seeing the rule was "CapitalLoss" I expected that there would be one number, and being greater than this number would lead to more fraud. This intuitively makes sense, however was not the reality. For OneR, there were many different ranges that were marked as fraudulent. This appears to be overfitting, as these ranges appear to be arbitrary artifacts of the data set, however the accuracy increased by 6% over 10 crossvalidation folds. When removing demographic data, the OneR classifier chooses the date the policy is created: “DateOfPolicyCoverage”. Dates are transformed into their UTC time and represented as an integer, this makes the ranges very difficult to interpret however I assume this points to more fraudulent policies opened at specific times of year. Interestingly, the data set containing both before and after information regarding the claim chooses the date of the policy being opened. Both "CapitalLoss" and “DateOfPolicyCoverage” are absent in the important features listed above. This is because the Random Forest classifier has access to all of the available attributes and finds connections between them. The “TypeOfIncident” is considered the most important attribute by Random Forest, but only in conjunction with the other top attributes. OneR chooses the attribute that can predict the most without the help of any other attributes.



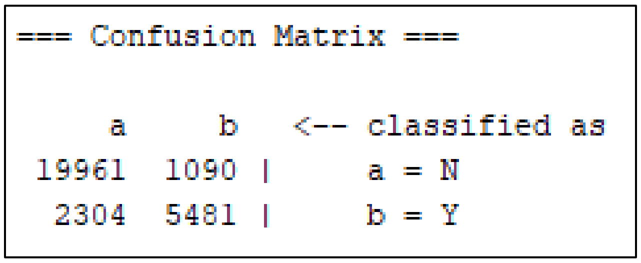
```
=== Confusion Matrix ===

     a     b   <-- classified as
 20550   501 |     a = N
  5323  2462 |     b = Y
```

Figure 7: OneR confusion matrix

J48 Results

J48 performs quite well. With all data it has an accuracy of 88% and with minimal data still performs at 82.6%. This 6% difference almost perfectly matches the OneR difference of 5% difference between these data sets. Looking into the J48 tree with all data, the first attribute the tree splits with is “SeverityOfIncident”. Similar to “TypeOfIncident”, “SeverityOfIncident” encapsulates many of the other attributes. More severe incidents are less likely to be fraud while “Trivial-Damage” leads to more fraud, the same trend I discussed regarding “TypeOfIncident”. Interestingly, the second attribute the J48 tree splits on is “InsuredHobbies”. This attribute is slightly misleading as it represents a hobby that the customer has, but does not necessarily mean the customer has insurance on this hobby. Other demographic attributes, such as “InsuredGender”, have a similar naming scheme but are less misleading. Looking at the tree, hobbies that are less intense usually point to less fraud. Across all severity levels, “movies”, “dancing”, “golf”, and “reading” generally point towards legitimate claims. Hobbies “cross-fit”, “exercise", and “basketball” are branch further down, indicating that they are more associated with fraud. Hobbies that are game-like such as “board-games”, “video-games”, and “chess” also branch further indicating that there may be some trend here as well. I’m not really sure what to make of this. The hobbies that lead to more nodes are similar and the hobbies that point to no fraud are also similar. This means that there is a trend within hobbies, but intuitively that seems like a big stretch. I think the J48 tree has an affinity for nominal variables with a large amount of categories because it splits the data up into much smaller categories that can be analyzed further. After hobbies, many different attributes show up. Attributes regarding money on both the claim and the policy are common, they follow the trends discussed in the attribute results section. Vehicle year of make, “YOM” is also prevalent in the tree. Almost every split has older “YOM” point to legitimate claims while newer vehicles are more likely to be fraudulent. This intuitively aligns with the the goal of getting the most amount of money from a fraudulent claim as payouts for newer cars are higher. For less severe incidents, higher “BodilyInjuries” almost always point to fraud, while lower “BodilyInjuries” points to less fraud at higher severities. These two variables tell a story about the accident, and when they do not align the story is less likely to be real and thus fraudulent. “InsuredGender”, the gender of the customer, shows up on the tree but without a common trend. I would expect males to be more likely to commit fraud than females, as they commit other crimes at a higher rate (6). “InsuredGender” generally shows up three or four levels deep into the tree, meaning the records that make it there are already very segmented, and sometimes points directly to males committing fraud and sometimes points directly to females committing fraud.



```
=== Confusion Matrix ===

     a     b   <-- classified as
 19961  1090 |     a = N
  2304  5481 |     b = Y
```

Figure 8: J48 Confusion Matrix

Random Forest Results

Random Forest does not provide the final decision tree as an output, however it probably uses the attributes found from the Random Forest attribute selection module. Random Forest had the highest accuracy of all classifiers with a 92.9% accuracy with all data and an 85% accuracy with the minimal data. Surprisingly, the Random Forest maintains a 90.7% accuracy with the data available when making a new policy when demographics are included. Weka has a model performance chart module that plots different aspects of the performance. By plotting the true positives against the true negatives and looking at the area under the curve, I was able to see how accurate the model was and where the mistakes were being made. In this model a “positive result” is a legitimate claim and a “negative result” is a fraudulent claim.

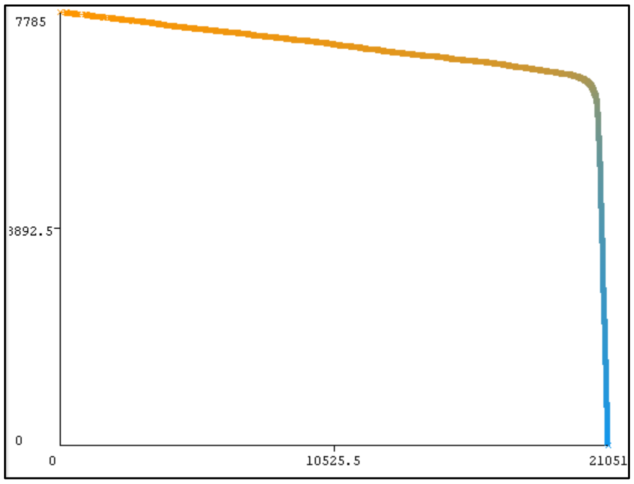


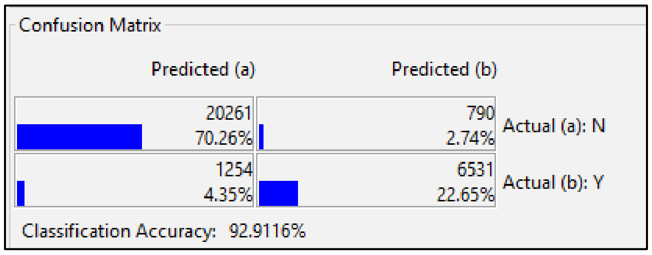
Figure 9: Random Forest accuracy, X axis: true positives, Y axis true negatives

The area under the curve represents the accuracy of the model. A perfect chart would have the full chart under the area of the curve. As the area decreases, the accuracy decreases. The area can be increased by moving the corner of the curve up or to the right. Moving the corner up lifts the horizontal portion, increasing the area, and moving the corner right widens the horizontal area. The graph for the Random Forest is slightly biased towards true positives. The true negative part of the graph has less area to move whereas the true positive side could be raised a bit.

The amount of true positives and true negatives can be artificially changed by changing the score threshold. The Random Forest classifier assigns each record a final score at the end of classification. The higher the score represents how likely it is to be fraudulent. A score of 0 means the claim is certainly fraudulent and a score of 1 means the claim is certainly legitimate. The color of the graph represents the final threshold score of a given record, blue for legitimate claims and yellow for fraudulent ones. The graph is somewhat misleading, as the X axis is three times longer than the Y axis, and it appears there are a similar number of yellow to blue points. In reality there are 6,500 classified fraudulent claims and 22,000 legitimate ones.

Weka has a cost-benefit analysis module in which this threshold can be changed. The Random Forest automatically picks the score threshold that provides the highest accuracy: 0.58. This means that any record with a score lower than 0.58 is considered fraud. In the module, the threshold can be manually changed. If an insurance company wanted to avoid falsely accusing anyone of fraud, they would want to lower the false negative (remember in this graph a “positive result” is a legitimate claim and a “negative result” is a fraudulent claim). Increasing the threshold means that more records will be below the threshold and thus marked fraudulent. Lowering this threshold will reduce the amount of claims, thus only flagging claims that are more certain to be fraudulent.

The default confusion matrix for Random Forest has a 2.09% false negative rate with 602 claims wrongly flagged fraudulent.

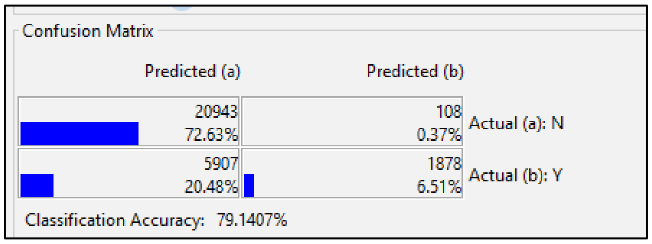

Confusion Matrix

	Predicted (a)	Predicted (b)	
	20261 70.26%	790 2.74%	Actual (a): N
	1254 4.35%	6531 22.65%	Actual (b): Y

Classification Accuracy: 92.9116%

Figure 10: Random Forest Default Confusion matrix,
Score Threshold = 0.58

While this matrix is the highest accuracy, and thus the best threshold for maximizing investigation return, there is more at stake than just money. Investigating customers is not a good look for an insurance company, especially when they are wrong. By changing the threshold, the number of false negatives can be reduced, at the expense of the overall accuracy. Lowering the threshold to 0.14 lowers the false negative rate to 0.37%, but lowers the overall accuracy to 79%.


Confusion Matrix

	Predicted (a)	Predicted (b)	
	20943 72.63%	108 0.37%	Actual (a): N
	5907 20.48%	1878 6.51%	Actual (b): Y

Classification Accuracy: 79.1407%

Figure 11: Random Forest Confusion matrix,
Score Threshold = 0.14

The tradeoff of false negatives to overall accuracy would need to be made at the discretion of the insurance company. To make this decision easier, I plotted number of false positives against the threshold, visually showing the impact of changing the threshold.

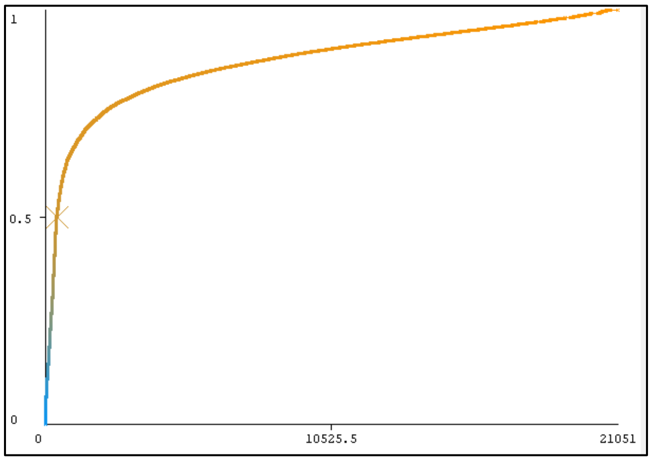


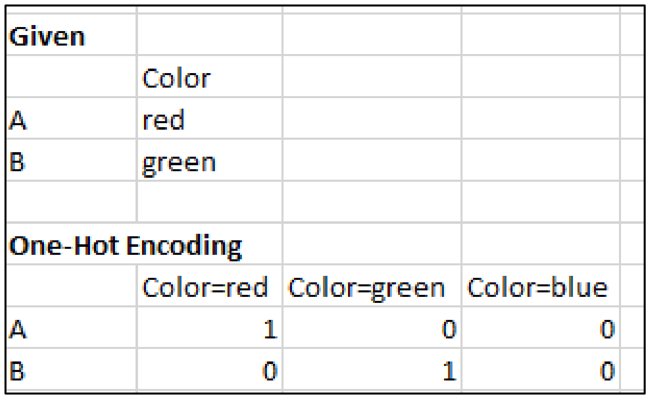
Figure 12: Threshold v. False Negatives

Naïve Bayes Results
Naive Bayes performed the worst out of the three main classifier models. The data is fairly intertwined and thus a model that considers this is needed. Naive Bayes did have the fastest running time, which could be important as the size of the data set increases. I did not explore this classifier as much due to the lower accuracy.

6 REAL WORLD SCRIPTS

The final part of my project was creating fraud investigation expected value and fraud fee scripts that could be used in the real world. As discussed in section 3, these scripts would need to be trained on ethically sourced data before being implemented for real.

One-Hot Encoding
Before writing the PySpark scripts, I needed to encode the data to remove categorical attributes. While Weka supports these categorical (nominal) attributes, PySpark does not. To fix this I used a technique called One-Hot encoding. One-Hot encoding turns a singular categorical variable into separate binary columns. If the original attribute was “color”, with values “red”, “green”, and “blue”; One-Hot encoding would remove the color column and create a “color=red”, “color=green”, and “color=blue” columns. For a record that originally had a “color” value of “green”, the values for “color=red” and “color=blue” columns would be 0 while the “color=green” column would be blue.
Weka has a nominal to binary module so this preprocessing was just a simple Weka Script. I converted nominals for both the data available when creating a policy, and after a claim is made (FINALDATACONVERTEDNOMINALSNOZIP.csv, FINALDATANOCLAIMCONVERTEDNOMINALSNOZIP).



Given				
	Color			
A	red			
B	green			
One-Hot Encoding				
	Color=red	Color=green	Color=blue	
A	1	0	0	
B	0	1	0	

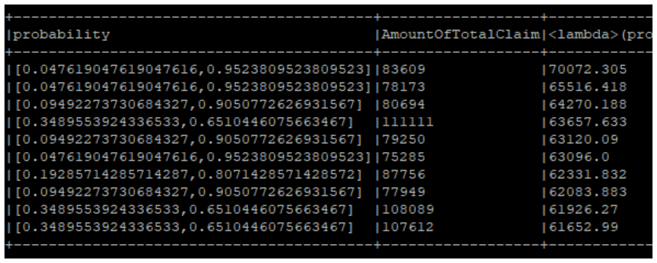
Figure 13: Example of One-Hot encoding on a color vairable

Fraud Investigation Expected Value
The first script I developed (FindInvestigateable.py) generates the expected value of an investigation into a given claim. For any claim there is some expected return on finding a fraudulent claim, however it is not always worth investigating. If a claim for $3000 is complete fraud, but an investigation is $5,000, the claim probably should just be paid out. An insurance company could use this script, knowing the cost of an investigation, to determine whether a given claim should be investigated. The fraud investigation expected value script takes a similar form to the Weka J48 flow. I import the csv file, randomly select 10% of the data as a test set and the other 90% as a training set (a real world application would use new claims as the test set), and assemble the data in a vector_assembler. The J48 classifier then takes the assembled data and creates a model. Using an evaluator on the “ReportedFraud” attribute I can then get a list of predictions and sort by the most likely to be fraudulent. Unlike Weka, the PySpark predictions give the exact probability, according to the model, of being fraudulent. Using a simple lambda function I can take this probability and multiply it by the size of the claim to get some expected value. The lambda function used is:

```
(v[1] * v2 * 0.88 * factor * successRate)
```

`v[1]` - Probability of being fraud
`v2` - “AmountOfTotalClaim”
`0.88` - The overall accuracy of the J48 model
`Factor` - a user specified factor
`successRate` - success rate of investigations

Multiplying the probability by the total amount of the claim returns the expected value. This number, however, is not necessarily accurate. According to Weka, the J48 has an accuracy of about 88%, and thus the probability of being fraud is 88% accurate. The final issue is that the entire claim may not be fraudulent. A claim of $50,000 with a 90% probability will have an expected investigation value of $39,600. (50,000*.9*.88). This claim is considered fraudulent if the actual damages are $45,000 and the customers inflated the costs to $50,000. I am not aware of how insurance law works. I'm not sure if finding this $5,000 discrepancy would void the entire claim or simply save the company $5,000. To combat this, I added a user specified factor. This factor represents the average inflation of costs. A value of 1 means that the entire claim is inflated and a value of 0.5 implies that half of the claim is inflated. If part of the claim being fraudulent voids the whole claim, then this factor can be set to 1 or removed entirely. For my tests I have the factor set to 1. Finally, investigations are not always correct. Even a claim that is fraudulent that is investigated may have no proof of being fraudulent. The success rate of investigation is difficult to find because the investigations that are wrong are not known to be wrong. This value should be put in by the investigation agency. For my tests I have set the factor to 1.Running the script and sorting by highest expected value returns the following:



```
+---------------------------------------+------------------+----------
|probability                            |AmountOfTotalClaim|<lambda>(pro
+---------------------------------------+------------------+----------
|[0.047619047619047616,0.9523809523809523]|83609           |70072.305
|[0.047619047619047616,0.9523809523809523]|78173           |65516.418
|[0.09492273730684327,0.9050772626931567] |80694           |64270.188
|[0.3489553924336533,0.6510446075663467]  |111111          |63657.633
|[0.09492273730684327,0.9050772626931567] |79250           |63120.09
|[0.047619047619047616,0.9523809523809523]|75285           |63096.0
|[0.19285714285714287,0.8071428571428572] |87756           |62331.832
|[0.09492273730684327,0.9050772626931567] |77949           |62083.883
|[0.3489553924336533,0.6510446075663467]  |108089          |61926.27
|[0.3489553924336533,0.6510446075663467]  |107612          |61652.99
+---------------------------------------+------------------+----------
```

Figure 14: Script results. “CustomerID”, “ReportedFraud”, and “Prediction” cut off to fit image size requirements. Lambda= EV

Customer 14401, the first row, has a 95% probability of being fraudulent. Given the “AmountOfTotalClaim” of $83,609; assuming the entire claim is invalid if it is fraudulent and an investigation has a 100% effectiveness, the expected value of investigation is $70,072. If the insurance company can conduct an investigation for less than 70,072; they should. Customer 14231, the bottom row, has a higher “AmountOfTotalClaim”, but a much lower probability of being fraudulent. The expected value of this investigation, under the same assumptions, is only $61,652.

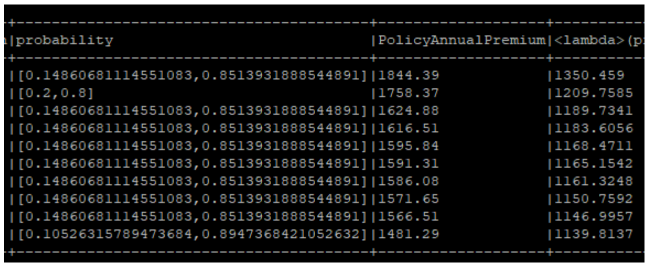
Fraud Fee

The fraud fee calculation script (FraudFee.py) generates a fraud fee at the time of creating a policy. This type of prediction would receive a lot of ethical pushback, as a company is predicting that you as an individual are going to commit crime, however there are non-trivial trends within the data that show this prediction is possible. If I were to implement a fraud fee into my insurance policies they would probably take the form of a deposit that would be returned if a customer’s claim was legitimate. In practice it is probably easier to make this fee standard for all customers, but I still wanted to make the tool to explore the data trends. This script takes almost identical form to the expected value script, the main differences being the data set used (no claims information), and the lambda function:

```
(v[1] * v2 * 0.86 * factor)
```

`v[1]` - probability
`v2` - "PolicyAnnualPremium"
`0.86` - The overall accuracy of the model
`factor` - user specified factor

In this function the factor probably shouldn't be 1. The factor can be adjusted to reduce the overall fraud fee, a factor of 1 keeps it the same while smaller numbers reduce it. Given a policy with a premium of $1,000 and a probability of 35%, the fraud fee calculated with a factor of 1 is $301 (1,000 * .35 *.86). A probability of 35% means there is a 65% probability that the person does not commit fraud, and thus the model predicts their claim will be legitimate. A $301 fee for fraud seems quite high when the model predicts that the person will not commit fraud. By reducing the the factor to 0.5 or something lower, the fees seem much more reasonable. This factor would be set by the insurance company, for my tests it is still set to 1.



```
-+---------------------------------------+--------------------+----------
a|probability                            |PolicyAnnualPremium|<lambda>(p
-+---------------------------------------+--------------------+----------
 |[0.14860681114551083,0.8513931888544891]|1844.39           |1350.459
 |[0.2,0.8]                               |1758.37           |1209.7585
 |[0.14860681114551083,0.8513931888544891]|1624.88           |1189.7341
 |[0.14860681114551083,0.8513931888544891]|1616.51           |1183.6056
 |[0.14860681114551083,0.8513931888544891]|1595.84           |1168.4711
 |[0.14860681114551083,0.8513931888544891]|1591.31           |1165.1542
 |[0.14860681114551083,0.8513931888544891]|1586.08           |1161.3248
 |[0.14860681114551083,0.8513931888544891]|1571.65           |1150.7592
 |[0.14860681114551083,0.8513931888544891]|1566.51           |1146.9957
 |[0.10526315789473684,0.8947368421052632]|1481.29           |1139.8137
-+---------------------------------------+--------------------+----------
```

Figure 15: Script results. “CustomerID”, “ReportedFraud”, and “Prediction” cut off to fit image size requirements. Lambda= Fee

By sorting the results by the highest fees, the issue regarding fee sizes becomes apparent. The first customer has an 85% probability of submitting a fraudulent claim and thus the fraud fee is $1350. While the factor can be used to lower this fee the overall questions of “should we accept this person as a customer?” still remains. It would probably be better for an insurance company to deny service and avoid the entire issue. An insurance company could also bundle all of the policies they create in a month, calculate the fraud fee for each of the policies, and average the fee across all policies. This would be more equitable and would raise less ethical issues.

7 CONCLUSION

This project exposed me to the amount of work required to make real applications using large data sets. Data needs to be ethically sourced, an expensive and difficult process. The data then requires extensive processing and preparation before any classification can be done. After preprocessing, choosing the correct classification models, parameters, and thresholds requires extensive work before producing any results. Even after making models and generating results, implementing these results into systems that are accepted by the general public is quite difficult as well. Although there is extensive work, there are rather wonderful results. In this project I showed that using readily available numbers and values with readily available models, an insurance company can predict which of their claims are fraudulent. Although very inhuman, the accuracy of the models is staggeringly high as the trends that are unable to be processed by humans are trivial for the computer. Regulations regarding the use of this type of prediction will certainly increase as the models improve and the amount of available data increases. There certainly is a line that defines what predictions are acceptable to act on, but making these predictions and understanding the trends of the world is most certainly the future.

REFERENCES

[1] https://www.iii.org/article/background-on-insurance-fraud

[2] https://www.verisk.com/siteassets/media/campaigns/gated/underwriting/verisk-the-challenge-of-auto-insurance-premium-leakage.pdf?__FormGuid=8c509869-699d-4698-9ac3-9ada3d271c97&__FormLanguage=en-US&__FormSubmissionId=65299f67-8c20-408c-97bd-221fc1cee1bf

[3] https://insurancefraud.org/wp-content/uploads/The-Impact-of-Insurance-Fraud-on-the-U.S.-Economy-Report-2022-8.26.2022.pdf

[4] https://www.weforum.org/reports/the-travel-tourism-competitiveness-report-2017/#series=TTCI.A.02

[5] https://content.iospress.com/articles/statistical-journal-of-the-iaos/sji170395

[6] 6https://ucr.fbi.gov/crime-in-the-u.s/2019/crime-in-the-u.s.-2019/topic-pages/tables/table-42