Insurance Fraud Prediction

Report

Ronak Ghatalia
Computer Science
Ryerson University
Toronto, Canada

Sourena Khanzadeh
Computer Science
Ryerson University
Toronto, Canada

I. Introduction

Insurance fraud is one of the largest areas of white-collar crime, and continues to be a growing problem for the insurance industry. This results in increased losses from fraudulent claims, and higher premiums for customers.

It is often difficult to determine if a claim is legitimate or fraudulent needing further investigation. The purpose of our project is to use machine learning to see if we are able to predict a fraudulent claim, which would allow the insurance company to avoid making the payment.

Our research it going to compare the use of different common machine learning algorithms to see how they preform to solve this problem.

II. Related research

There has been previous research done in predicting insurance fraud, that involve the use of machine learning and deep learning algorithms.

The machine learning techniques seem to be mainly around clustering to determine between fraudulent or legitimate claim. GA-Kmeans and MPSO-Kmeans was used on a set of 5000 car insurance claims. EvoDM algorithms were more accurate than K-Means (Liu, et al. 2012). Also using a random forest method was found to be a suitable method for large unbalanced data sets (Li, Yan, et al. 2016).

Deep learning has also been used to try to detect insurance fraud. A random rough subspace based neural network can be more accurate at detection suspicious claims (Xu, Want, et al. 2011). A Bayesian learning neural network was used to detect insurance fraud (Viaene, Dedene, et al. 2005).

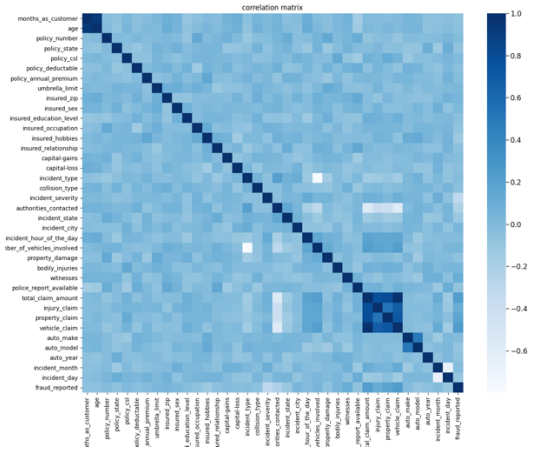
III. Methiods and Models

A. Data Processing

We used an insurance fraud data set from Kaggle. It consists of 1000 insurance claims, which have been determined to be fraudulent or non-fraudulent claims. There are a total of 39 columns and with 38 dependent variables, and 1 independent variable for fraud reported. The data is not balanced there are 247 fraud (24.7%) and 753 non-fraud (75.3%) claims.

We removed some columns that had a weak correlation with our output variable, and did not help for our experiments. We removed policy number, and street address. We also divided date into day and month. This gave us 37 input features for our experiments. Then we replaced all fields that had question marks with null. We then encoded all the string values by their corresponding fraud relativity, we grouped all the values and for each we found the mean of the fraud cases, the independent column is encoded to 1, 0. 1 is Y and 0 is N, which indicates whether it is fraud or not fraud. For the relevancy we add up all the ones for the specific string value and then take its mean. These are the values encoded for the string values (Appendix A).

Below is a correlation heat map matrix of each column. The darker the colour the higher the correlation between the two columns.



B. Expriments

Since out data is imbalanced with only 247 fraud cases out of the 1001 insurance claims. We used 460 insurance claims from the data set with 230 non-fraud and 230 fraud to allow for proper training and testing of the algorithms. From this set we used 184 fraud and 184 non-fraud for training, and tested on 46 fraud and 46 non-fraud.

With our limited data we also used k-fold cross validation with 5 folds, using the 460 claims from our balanced data set, which provides a testing set of 20%. This will test on 4 of the sets and test on the last one, and repeat 5 times allowing all sets to be used as a testing set.

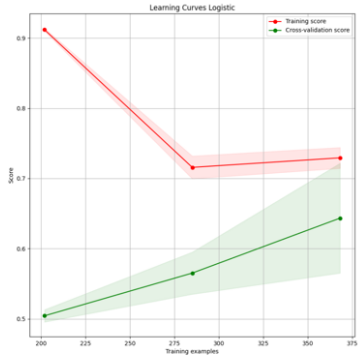
We used Logistic Classifier as our starting base case, SVM classifier, K-NN classifier, Decision Tree Classifier, Random Forest Classifier, and Neural Network MLP Classifier (Appendix B). The focus of our experiments is to compare the different algorithms, and see which one works better to accurately classify fraud cases.

The use of our algorithm should be to identify cases of potential fraud for review. Since only identified cases will be reviewed, and the non-fraud cases will not be reviewed, we want to prevent false negatives, and are focusing on high recall and accuracy.

IV. Analysis

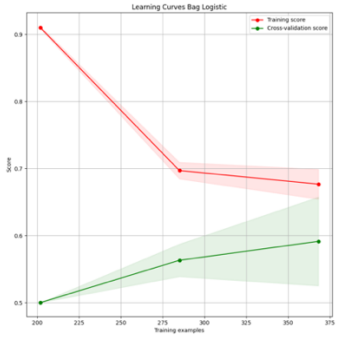
A. *Logistic Regression*

This is our base case that we used to start with. We used Newton's Method since it is an efficient way to solve for theta, and below is the graph of the learning curve.

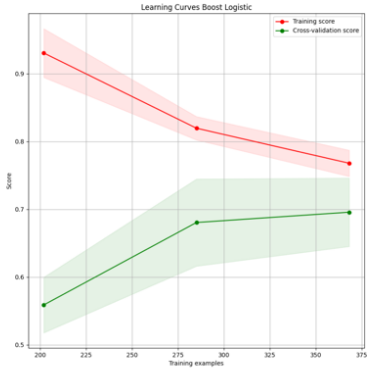


Based on the graph there is some bias and variance. To see if we could reduce this we tried bagging and boosting to see what impact this has on our results.

Bagging



AdaBoost

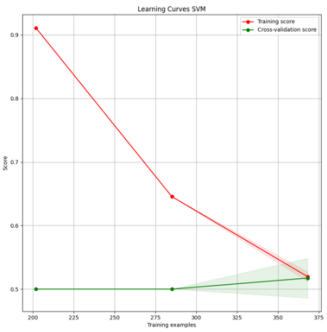


Bagging reduced the variance but increased the bias. With boosting there is a reduction in bias with a simair variance as bagging. AdaBoost helpes improve our logistic calssifier using a Newton solver, and provides the highest testing accuracy.

B. *SVM*

Since we do not know if your data is linearly seperable, we decided to use an RBF kernal for our SVM. This kernal will be able to sperate our classes if they are linearly seperable, and also if they are not linearly seperable.

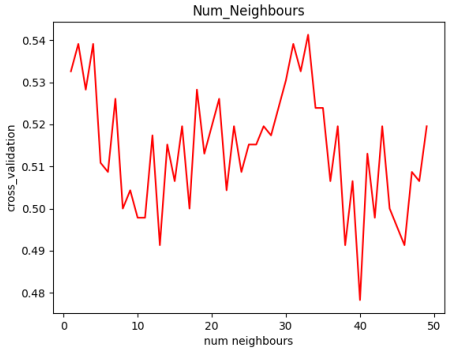
This did not perform well. There is a high bias, even though it has a low variance. Since the bias is low we did not use regularization which would reduce the bias more. This is not able to separate our data accuratly.



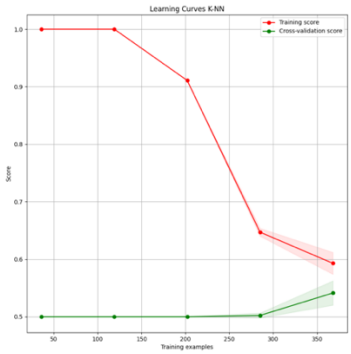
C. *KNN*

Since we have high dimensional data we decided to use KNN with Manhattan distance, which uses a city block measurement method. Based on the research by (Aggarwal, Hinneburg, Keim)[7] it is better to use Manhattan distance instead of Euclidean distance. High dimensional data is sensitive in Lk norm to values of k.

To determine the optimal neighbour size we created a graph with different values of neighbours and cross validation values. We found the optimal neighbour size is 33. We used equal weighting.



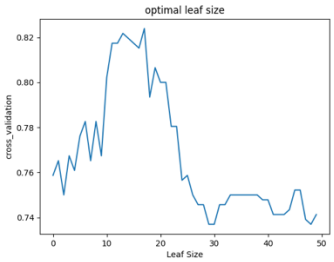
Based on the graph below there is a high bias and low variance using the optimal neighbour size. Since the bias is high we did not use regularization.



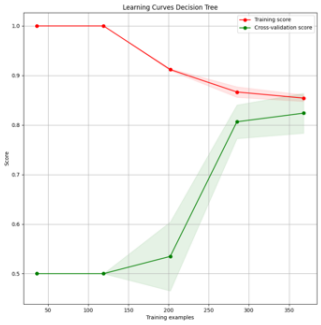
D. Decision Tree

Since we used SVM we decided to try this method also, since we have a classification problem. This takes SVM further by recursively dividing the subsection until it finds the right fit.

To determine the optimal leaf size, we compared the output from 1 to 50 leaves with the cross validation result. We found the optimal leaf size to be 18.



This provided a low amount of variance and low bias. It was not necessary to use any boosting or bagging with this result. It performed well.



E. Random Forest

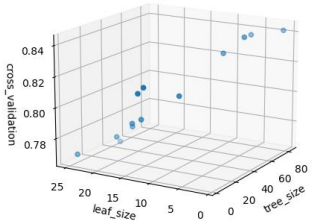
This was used because of the limitations of using decision trees. Decision trees are good and classifying training data, but are not good at generalizing to other data. Since we had 37 features we used features size of 25 with random bootstrapping, which would allow good variance between trees.

To determine the optimal leaf size and tree size to use, we compared a range of 1 to 25 for leaf size with a range of 1 to 100 tree sizes with the associated cross validation results. Below is the graph of our results.

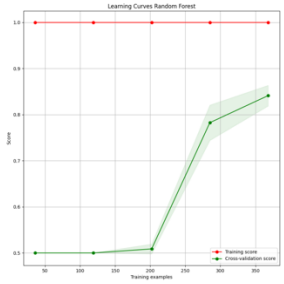
We obtained 2 optimal values:

tree_size = 83, leaf_size = 1

tree_size = 58, leaf_size = 4

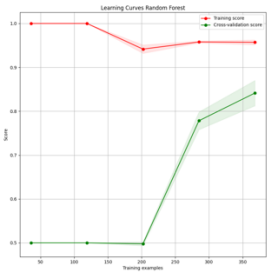


tree_size = 83, leaf_size = 1

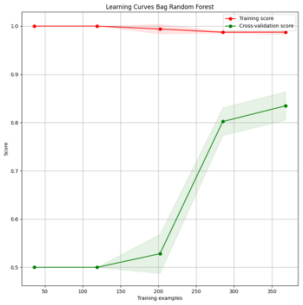


With a leaf size of 1 this model is over fitting our data based on the perfect fit on our training data. This would not be the optimal value.

tree_size = 58, leaf_size = 4



This does not over fit our data. Based on the results the bias is low but there is some variance. So we tried to use bagging to reduce the variance.



The bagging does not help improve the variance and our model does better without the bagging.

F. Neural Network

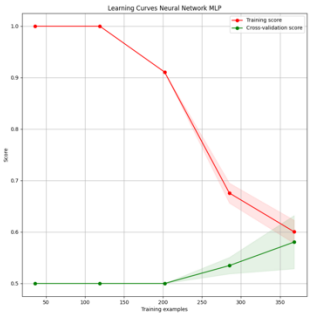
We created a MLP network as a starting point for further research which could be expanded to a deep learning network. Since we have a small data set we decided to use 2 layers. Layer sizes should be the same in a neural network[8].

To determine which activation function and solver would be optimal, we compared using different combinations with different sizes of 2 layers equal size.

The 4 graphs below show our results using different activation functions, and the lines are the solvers. We compared layers sizes from 37 to 100. The first graph is using identity, the second is using logistic, the third is using TanH, and the fourth is using relu. The blue is adam, yellow is sgd, and blue is lbfgs.

https://images-ext-1.discordapp.net/external/XoFgmsmRcKqwofA9X_Gnh7xWwCgc_5ivIYSID0gzQSs/https/raw.githubusercontent.com/sourenaKhanzadeh/insurance_fraud/master/plots/optimal_NN.png?width=1920&height=475

The results from our graph shows the optimal is a TanH activation function, using an adam solver, with 2 layers of size 92.

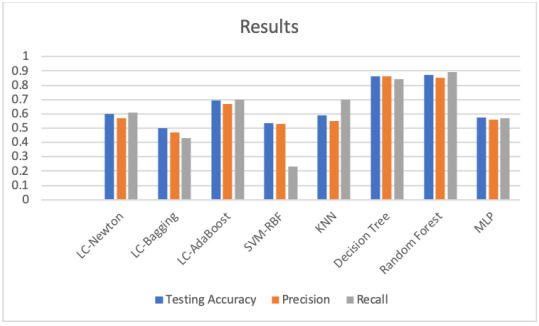


Using these parameters, we have low variance but a high bias. This is because of our small data size, and would probably be improved with a larger data size, and using additional hidden layers.

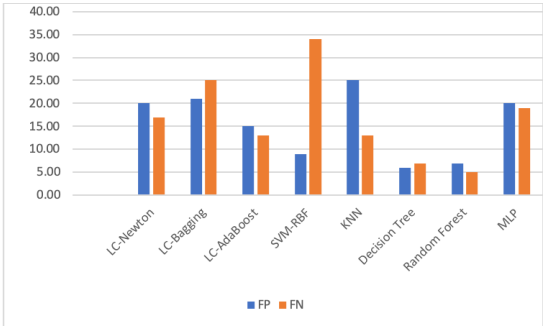
V. Conclusion

For our experimentation we tried to compare some common algorithms and see how they worked with our data, to predict insurance fraud. We wanted them to have good accuracy, but keep the false positives to a minimum.

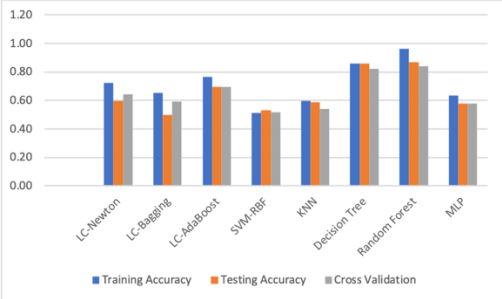
We looked at the training accuracy and cross validation results to make changes to model parameters, and determine if we should use regularization, boosting, or bagging. Once our models we optimized then we looked at the testing accuracy.



Our random forest model performed the best, with the highest testing accuracy and recall values.



Our random forest model has the lowest number of false negative values, which is better for our data for minimizing false negatives values.



From our experiments our random forest had the highest training, cross validation, and testing accuracy, even though it had a higher variance than some of the other models.

This analysis is a starting point to build from. There are additional things that could be done to try to improve these results. On further experimentation we would try feature selection using mutual information. This would change the input features we use and remove features that are not as useful to determine fraud. We could also use feature scaling during the data processing stage to see if this would help our models perform better.

There is further optimization that could be done with our model parameters. For logistic regression we could try to further optimize for learning rate, and starting points. We might be in a local minimum and using different starting points and learning rates might improve the results. For SVM we could try further experimentation with different kernels, and for KNN using different distance measurement techniques and weightings. Our decision tree and random forest preformed the best. We could try to further optimize using a different number of features. Our MLP was a base starting point that did not do well. We could try to use deep learning with a larger number of hidden layers and further optimize for hidden layers of larger sizes.

A. *Packages*

We used an object oriented programming style for our code (Appendix C). We have used packages such as sklearn, numpy, matplotlib, seaborn and numpy. Sklearn is used to fit different models such as logistic, KNN, SVM, Decision Trees, Random forest and Neural Networks. Sklearn is also used to encode the y label which is the fraud reported column.using LabelEncoder. Also Sklearn is responsible for splitting test and train sets, confusion matrix,cross_validation, classification_report and accuracy_score. From mpl_toolkits.mplot3d we imported Axes3D to implement 3D graphs. Numpy is used to manipulate arrays and mathematical data. Seaborn and Matplotlib are used to draw graphs and figures such as pie charts, bar graphs, scatter plots, 3D plots, and heat maps.

B. *Implementation*

The First thing we did was to do all the preprocessing such as replacing all ? with null, dropping the columns that were not necessary,then we split the dataset to make a more balanced dataset. We did that by choosing a specific number of rows that the fraud reported was true and then we did the same number of none frauds so now we have a balanced dataset with 230 frauds and 230 none frauds. Then the string columns were encoded, to encode the column we grouped the column by the name of the string and took the average of the frauds reported, before this we transferred frauds reported to 1, 0 by using the LabelEncoder from Sklearn module, Y is 1 and N is 0. As said we encoded the string value with the relevance of it being fraud for example, the auto make column has a column name Mercedes, we group all the Mercedes and got all the fraud reports, we added the 1, 0 and took their average, then replace Mercedes with the percentage of the relevancy.

For running test for optimal parameters for KNN, decision tree, and random forest algorithms we use a loop to go through a range of numbers, and the output is the cross validation values from our models. K-NN we iterated from a range of 1 to 50 arbitrary neighbours being the x-axis and cross_validation result being the y-axis. We did the same thing for the Decision Tree with min_num_leaf parameter. For Random Forest we needed to test two parameters, the number of trees and leaf size. To accomplish this efficiently we chose a number we thought fitting for the tree size which was 100 and 25 for leaf size. In other words we iterate over two loops one from 1 to 100 and the other from 1 to 25. But since it is computationally very heavy we decided to partition num_leafs by 2 so we used three for loops, first one is a k number for range(1, 101, 2), second range(1, 101) and third is range(k, k+2), every batch we found the optimal result for the cross_validation, which was a total of 12 optimal results and from those 12 we chose the max value received from the 12 maximas. Then we chose to draw a 3D graph from those 12 maximas.

VI. References

[1] https://fortunly.com/statistics/insurance-fraud-statistics#gref

[2] https://www.wnins.com/resources/personal/features/insurancefraud.shtml

[3] http://www.ijmlc.org/papers/136-L0021.pdf

[4] https://ieeexplore.ieee.org/abstract/document/7603443/citations#citations

[5] https://ieeexplore.ieee.org/abstract/document/5957885

[6] https://www.sciencedirect.com/science/article/abs/pii/S0957417405000825

[7] https://bib.dbvis.de/uploadedFiles/155.pdf

[8] https://www.youtube.com/watch?v=cObOAIImeVQ

Appendix A

Auto Model	Fraud Reported
Ultima	0.142857
ML350	0.142857
CRV	0.333333
Grand Cherokee	0.333333
TL	0.4
Camry	0.428571
Tahoe	0.5
RAM	0.538462
X6	0.571429
Passat	0.583333
Silverado	0.6
A5	0.6
Accord	0.6
Forrestor	0.6
Highlander	0.6
X5	0.615385
Malibu	0.625
3 Series	0.625
Pathfinder	0.636364
92x	0.666667
A3	0.666667
Maxima	0.666667
E400	0.7
Impreza	0.714286
Legacy	0.733333
Fusion	0.75
Civic	0.75
MDX	0.769231
Escape	0.777778
Wrangler	0.777778
C300	0.8
Corolla	0.8
F150	0.8
M5	0.857143
93	0.875
Neon	0.888889
95	0.888889
RSX	1
Jetta	1

Auto Make	Fraud Reported
Nissan	0.518519
Chevrolet	0.576923
Honda	0.578947
Toyota	0.590909
Mercedes	0.592593
Jeep	0.6
Audi	0.642857
BMW	0.657143
Accura	0.666667
Dodge	0.681818
Suburu	0.6875
Volkswagen	0.722222
Ford	0.777778
Saab	0.807692

Incident State	Fraud Reported
PA	0.428571
OH	0.545455
SC	0.574468
NC	0.634146
NY	0.674157
VA	0.75
WV	0.757576

Police Report	Fraud Rported
Yes	0.619048
No	0.743363

Incident City	Fraud Repoted
Northbend	0.584906
Arlington	0.603774
Riverwood	0.608696
Columbus	0.660377
Hillsdale	0.7
Springfield	0.706897
Northbrook	0.710526

Authorities Contacted	Fraud Reported
Other	0.617647
Ambulance	0.636364
Police	0.638095
Fire	0.666667
None	0.756757

Incident Severity	Fraud Reported
Major Damage	0.555556
Trivial Damage	0.612903
Minor Damage	0.713043
Total Loss	0.715909

Collision Type	Fraud Reported
Side Collision	0.585106
Rear Collision	0.616822
Front Collision	0.73494

Incident Type	Fraud Reported
Multi-vehicle Collision	0.60625
Parked Car	0.625
Single Vehicle Collision	0.685484
Vehicle Theft	0.771429

Insured Relationship	Fraud Reported
wife	0.538462
other-relative	0.591549
own-child	0.622951
unmarried	0.679245
husband	0.705882
not-in-family	0.777778

Insured Hobbies	Fraud Reported
base-jumping	0.315789
cross-fit	0.384615
golf	0.409091
dancing	0.5
camping	0.588235
skydiving	0.588235
paintball	0.588235
chess	0.615385
sleeping	0.625
reading	0.64
movies	0.6875
polo	0.714286
board-games	0.722222
yachting	0.73913
hiking	0.777778
bungie-jumping	0.782609
kayaking	0.782609
basketball	0.8125
video-games	0.8125
exercise	0.842105

Insured Occupation	Fraud Reportd
armed-forces	0.5
transport-moving	0.5
handlers-cleaners	0.526316
farming-fishing	0.590909
protective-serv	0.6
adm-clerical	0.611111
priv-house-serv	0.611111
machine-op-inspct	0.621622
craft-repair	0.64
sales	0.655172
exec-managerial	0.733333
other-service	0.769231
tech-support	0.785714
prof-specialty	0.888889

Insured Occupation	Fraud Reported
Associate	0.537037
JD	0.591837
College	0.6
Masters	0.7
High School	0.703704
PhD	0.714286
MD	0.72

Insured Sex	Fraud Reported
FEMALE	0.628866
MALE	0.681529

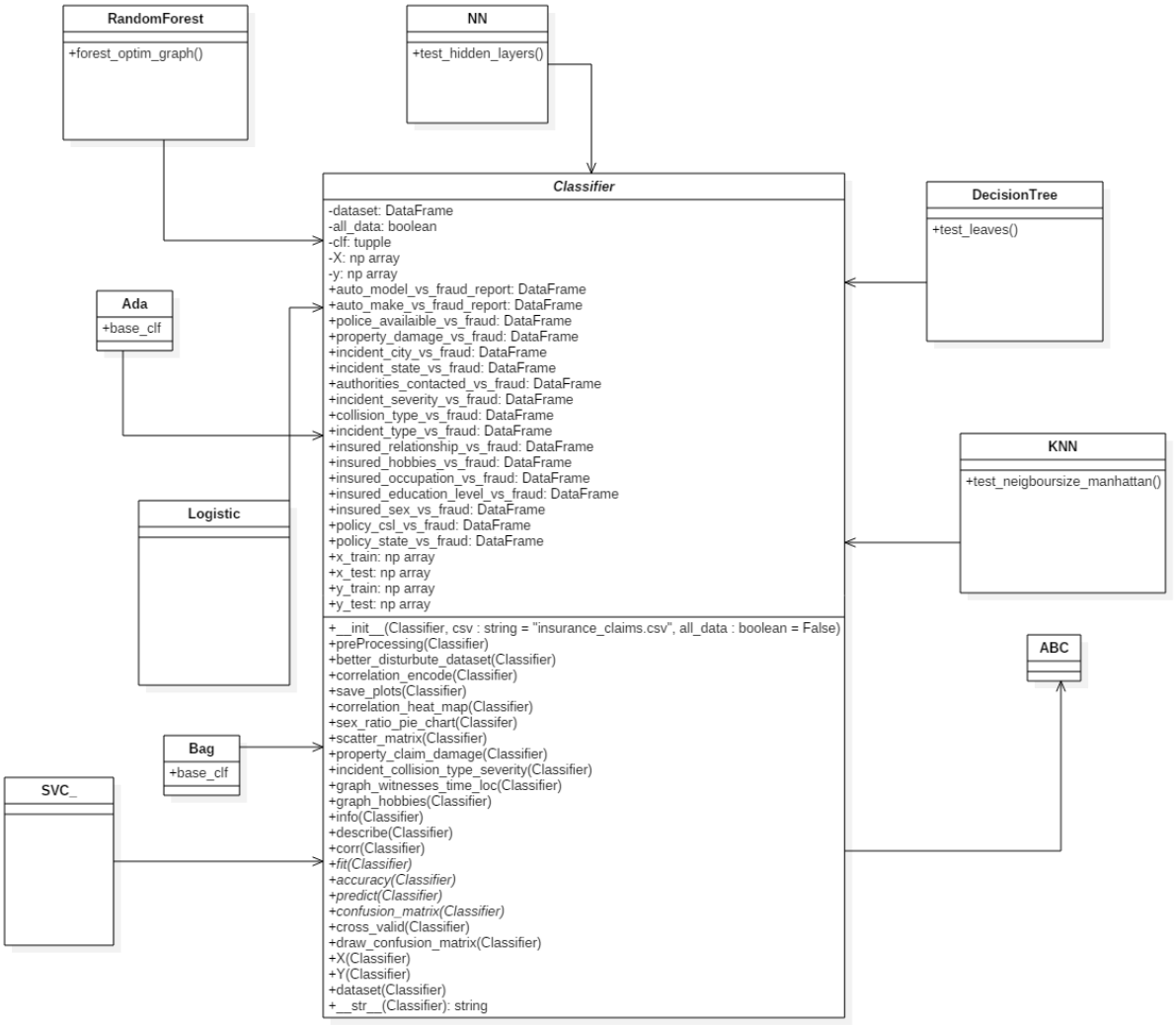
Policy CSL	Fraud Reported
250/500	0.632
100/300	0.65625
500/1000	0.673469

Policy State	Fraud Reported
IL	0.637795
IN	0.647059
OH	0.672131

Appendix B

Model	Training Accuracy	Testing Accuracy	Cross Validation	TP	FP	FN	TN	Precision	Recall	F1 Score
LC-Newton	0.723	0.598	0.643	28	20	17	27	0.57	0.61	0.59
LC-Bagging	0.655	0.500	0.593	27	21	25	21	0.47	0.43	0.45
LC-AdaBoost	0.766	0.696	0.696	33	15	13	31	0.67	0.70	0.69
SVM-RBF	0.511	0.533	0.517	39	9	34	10	0.53	0.23	0.32
KNN	0.598	0.587	0.541	23	25	13	31	0.55	0.70	0.62
Decision Tree	0.859	0.861	0.824	42	6	7	37	0.86	0.84	0.85
Random Forest	0.962	0.870	0.841	41	7	5	39	0.85	0.89	0.87
RF - Bagging	0.932	0.837	0.802	41	7	8	36	0.84	0.82	0.83
MLP	0.636	0.576	0.580	28	20	19	25	0.56	0.57	0.56

Appendix C


RandomForest
+forest_optim_graph()
NN
+test_hidden_layers()
Classifier
-dataset: DataFrame
-all_data: boolean
-clf: tupple
-X: np array
-y: np array
+auto_model_vs_fraud_report: DataFrame
+auto_make_vs_fraud_report: DataFrame
+police_availaible_vs_fraud: DataFrame
+property_damage_vs_fraud: DataFrame
+incident_city_vs_fraud: DataFrame
+incident_state_vs_fraud: DataFrame
+authorities_contacted_vs_fraud: DataFrame
+incident_severity_vs_fraud: DataFrame
+collision_type_vs_fraud: DataFrame
+incident_type_vs_fraud: DataFrame
+insured_relationship_vs_fraud: DataFrame
+insured_hobbies_vs_fraud: DataFrame
+insured_occupation_vs_fraud: DataFrame
+insured_education_level_vs_fraud: DataFrame
+insured_sex_vs_fraud: DataFrame
+policy_csl_vs_fraud: DataFrame
+policy_state_vs_fraud: DataFrame
+x_train: np array
+x_test: np array
+y_train: np array
+y_test: np array
+__init__(Classifier, csv : string = "insurance_claims.csv", all_data : boolean = False)
+preProcessing(Classifier)
+better_disturbute_dataset(Classifier)
+correlation_encode(Classifier)
+save_plots(Classifier)
+correlation_heat_map(Classifier)
+sex_ratio_pie_chart(Classifer)
+scatter_matrix(Classifier)
+property_claim_damage(Classifier)
+incident_collision_type_severity(Classifier)
+graph_witnesses_time_loc(Classifier)
+graph_hobbies(Classifier)
+info(Classifier)
+describe(Classifier)
+corr(Classifier)
+fit(Classifier)
+accuracy(Classifier)
+predict(Classifier)
+confusion_matrix(Classifier)
+cross_valid(Classifier)
+draw_confusion_matrix(Classifier)
+X(Classifier)
+Y(Classifier)
+dataset(Classifier)
+__str__(Classifier): string
DecisionTree
+test_leaves()
Ada
+base_clf
KNN
+test_neigboursize_manhattan()
Logistic
ABC
Bag
+base_clf
SVC_