

**Dineshchandar Ravichandran**

dravich@g.clemson.edu

**Archana Lalji Saroj**

[asaroj@g.clemson.edu](mailto:asaroj@g.clemson.edu)

**Prashanth Reddy Kadire**

[pkadire@g.clemson.edu](mailto:pkadire@g.clemson.edu)

**CPSC-6300**

**Insurance Fraud Detection**

**Checkpoint 3**

Contents

[Model choice justification 3](#_Toc116838367)

[XG-Boost:………………………………………………………………………………………………………...3](#_Toc116838368)

[SVM:……………………………………………………………………………………………………………….3](#_Toc116838369)

[Logistic regression: 4](#_Toc116838370)

[Chosen model's test error rate 5](#_Toc116838371)

[Performance of model based on the test error rate 7](#_Toc116838372)

[Predictions of model 9](#_Toc116838373)

[Goals for Checkpoint-3 9](#_Toc116838374)

# Alternative model choice justification

Our fraudulent claim data set comprises 1000 data points where only 24.70% of data are fraudulent, making the data set imbalanced. Furthermore, we also have labels stating which claims are fraudulent. As stated in checkpoint1, our project aims to create a classification model to classify genuine and fraudulent claims. Post going through a few of the existing works pertaining to statistical analysis and prediction on tabular data, as well as machine learning-based fraud detection models. We have narrowed our initial model implementation and prediction to the following:

1. XGBoost.
2. SVM.
3. Logistic regression.

Ridge Classifier:

The Ridge Classifier, based on Ridge regression method, converts the label data into [-1, 1] and solves the problem with regression method. The highest value in prediction is accepted as a target class and for multiclass data muilti-output regression is applied. In machine learning, ridge classification is a technique used to analyze linear discriminant models. It is a form of regularization that penalizes model coefficients to prevent overfitting. Overfitting is a common issue in machine learning that occurs when a model is too complex and captures noise in the data instead of the underlying signal. This can lead to poor generalization performance on new data. Ridge classification addresses this problem by adding a penalty term to the cost function that discourage complexity. This results in a model that is better able to generalize to new data.

Ridge classification works by adding a penalty term to the cost function that discourages complexity. The penalty term is typically the sum of the squared coefficients of the features in the model. This forces the coefficients to remain small, which prevents overfitting. The amount of regularization can be controlled by changing the penalty term. A larger penalty results in more regularization and a smaller coefficient values. This can be beneficial when there is little training data available. However, if the penalty term is too large, it can result in underfitting.

The loss function of Ridge classifier is not cross-entropy loss as like Logistic Regression. Rather the loss function is mean square loss with L2 penalty. It works in the following manner for the binary classification problems by making use of Ridge regression algorithm:

Converts the target variable into +1 and -1 appropriately

Train a Ridge model with loss function as mean square loss with L2 regularization (ridge) as penalty term

During prediction, if the predicted value is less than 0, it predicted class label is -1 otherwise the predicted class label is +1.

The cost function for ridge:

Min(||Y – X(theta)||^2 + λ||theta||^2)

Lambda is the penalty term. λ given here is denoted by an alpha parameter in the ridge function. So, by changing the values of alpha, we are controlling the penalty term. The higher the values of alpha, the bigger is the penalty and therefore the magnitude of coefficients is reduced.

It shrinks the parameters. Therefore, it is used to prevent multicollinearity

It reduces the model complexity by coefficient shrinkage.

Bagging Classifier:

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregates their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples(or data) from the original training dataset – where N is the size of the original training set. Training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out.

Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.

Algorithm for Bagging Classifier:

Classifier generation:

Let N be the size of the training set.

For each of t iterations:

Sample N instances with replacement from the original training set.

apply the learning algorithm to the sample.

store the resulting classifier.

Classification:

for each of the t classifiers:predict class of instance using classifier.

return class that was predicted most often.

# Chosen model's test error rate

As illustrated below XG-boost has better scores for training data, hence we will perform further training and testing using XG-Boost:

Text

Description automatically generated

Result:



The following graph illustrates the performance of Logistic regression, XG-Boost, and SVM model on train data:

Chart, box and whisker chart

Description automatically generated

# Performance of model based on the test error rate

Following the initial accuracy score, we can determine that XG-boost performs better than SVM and Logistic regression based on STD scores. So we proceeded to use XG-boost classification for our predictions and to improve the model's performance further, we performed hyperparameter tunning with Random Grid Search.

Text

Description automatically generated

Based on this following best parameter were received, and the XG-boost was further trained with these hyperparameters to improve its overall performance, with a train accuracy increase of 11.67%.

Text, letter

Description automatically generated

Text

Description automatically generated with low confidence

Following is the performance of tuned XG-boost on test data, where it is generating an accuracy of 81.5% and recall of 76.47%:

Graphical user interface, text, application

Description automatically generated

The model is achieving an AUC score of 79.85% with the following ROC:

Chart, line chart

Description automatically generated

# Predictions of model

We will be checking the performance of predictions of the XG-Boost based on: F1-score, recall, precision, F-beta, and confusion matrix:

Table

Description automatically generated

As illustrated above, XG-boost has a precision of 61% and an F-1 score of 68% for fraudulent claims.

Additionally, from the confusion matrix, we can observe that the model categorizes 7% of fraudulent claims as genuine claims and has an f-beta score of 72.7 % on recall. These scores show that the model can detect most fraudulent cases.