**Lucas**

**Title**

**Overview**

**Introduction**

**Motivation**

**Research Questions and Hypothesis**

**Challenges**

**Methodology Overview**

The diagram below shows a simplified version of our process. Our data was made up of two tables that had the transaction and identity information of some purchases that we merged along the transaction ID. Once the data was combined the memory needed for each column was reduced, missing values were filled by -999 and the categorical values where encoded. Once our data was ready we performed feature engineering by selecting the K best features, dropping high correlation columns and performing Principal Component Analysis. Due to our imbalanced data our next step was to sample the data so it would be more balanced before passing it to our models. We used a total of 5 classification models each of which was tested and tuned before being passed to our Ensemble Model. After being combined in the Ensemble model we are able to predict some more accurate results.

**Dataset Description**

**Exploratory Data Analysis**

For the Exploratory Data Analysis our work is somewhat limited as we decided to spend more time on building and testing our models. As we have just seen in the previous slides the data we were dealing with was very large with our combined set having roughly 430 different variables. As we had such a large number of variables the main focus for our Exploratory Data Analysis was looking at what was useful to us. To start we looked at the amount of data that we were missing as can be seen by our first figure showing the count of variables that are missing data somewhere in the range. From this we can see that although the largest count is on the 0-20% range we still have a lot of missing variables. Next we looked at the correlation matrix. Unfortunately we had too many variables here to plot the full matrix so we had to break it down but it gave us a good idea of how our data relates. After that we wanted to look at the heavy class imbalance of our data so we created the bar graph showing just how little of our data is labeled as fraudulent.

**Methodology: Feature Engineering**

**Methodology: Data Preprocessing**

**Handling Imbalanced Data**

As we can see from our earlier slides our data is heavily imbalanced with about 3% of the records being fraudulent. To deal with this imbalanced data we tried a number of different over and under sampling techniques. Oversampling is a technique that generates more cases of a minority class while undersampling reduces the amount of a majority class. From our results SMOTE was the best overall for oversampling. Smote works by generating new data from its k closest neighbors. Adasyn, which is also an oversampling technique that puts more weight on the decision boundary, was not quite as effective. All of the undersampling techniques apart from Near Miss where not effective at balancing out the data set as they would just get rid of data that would make the decision confusing instead of balancing the cases. Near miss was slightly more effective as it selected a select number of majority classes based on the distance to the minority class. Our final choice for our sampling algorithm a combination of Smote and Tomek Over and under sampling which first generates more minority classes before removing classes on the decision boundary. This method was close to SMOTE but gave us slightly better results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sampling Technique** | **Overall Accuracy** | **Accuracy of predicting fraud** | **Accuracy of predicting non-fraud** | **AUC Score** |
| SMOTE (Oversampling) | 80% | 72% | 81% | 0.7604 |
| ADASYN (Oversampling) | 72% | 79% | 72% | 0.7543 |
| Edited Nearest Neighbors (Undersampling) | 93% | 37% | 95% | 0.6582 |
| One Sided Selection (Undersampling) | 97% | 15% | 100% | 0.5749 |
| Near Miss (Undersampling) | 65% | 61% | 65% | 0.6322 |
| Combination of SMOTE and Tomek (Under and Over Sampling) | 80% | 72% | 80% | 0.7609 |

**Methodology: Model Construction**

* ANN Lucas
  + To Build the Artificial Neural Network we made use of the Keras python library which allows you to add layers with specific activation functions and neuron amounts
  + One of the issues we had with this model was the overfitting as when it was trained on the sampled training data it had a high accuracy of predicting fraud but once testing data was passed in it was suddenly a lot less accurate. To fix this we simplified the model which seemed to give us more uniform but less accurate results
  + Our final model has a first hidden layer with a size of about 700 a second hidden dropout layer with a dropout rate of 0.3 and a third hidden layer with about 300 nodes
* XGBoost Lucas
  + XGBoost is an ensemble learner which generates quick results by generating a number of decision trees.
  + XGBoost was an interesting algorithm to work with because in Python it had options for how you wanted it to run on your computer by allowing you to specify cores used or cpu vs gpu
  + XGBoost was a much faster algorithm that provided us with similar results to the ANN with much less time taken
  + We also Had an issue with XGBoost oversampling but by changing the learning rate, samples and weight we were able to get more uniform results

**Methodology: Model Construction Continued**

* Ensemble Classifier Lucas
  + After all of our models have been built we need a way to combine them to get a better result. Unfortunately because they are not very similar the only option available to us was using a majority voting ensemble classifier
  + There are pre built methods available to build an ensemble classifier but they where not working with our ANN so we built our own custom function to combine our results
  + Luckily for us all of our results are in simple binary and we have an odd number of classifiers so we where able to combine our results by taking the rounded average of all of our classifiers outputs

**Methodology: Evaluation 1**

**Methodology: Evaluation 2**

**Application Areas**

**Future Directions**

**Works Cited**