**DATA 300 FINAL PROJECT**

**Predicting Credit Card Fraud**

**with KNN Classifier, Logistic Regression, Naive Bayes and Random Forest**

**Team Members: Amanda Altamirano, Liam Riener, Abhik Shrestha, Charlene Vu**

1. **Data preprocessing**

For our final project, we are researching credit card fraud. A credit card transaction is considered fraudulent when a person purchases something with a credit card but doesn’t pay back the credit they were loaned. Credit card fraud can impact a person’s credit score and doesn’t work out in the long run for them, but the credit system is reliant on keeping fraud in check. It’s important to model credit card fraud accurately to predict who, when, and where fraud will occur. This way, financial institutions can deny credit cards to people who are likely to commit credit card fraud and prevent fraud from occurring.

The dataset we used was found on Kaggle. It originally contained 122 variables with 307,510 rows of data. However, the original dataset was flawed and so the final dataset was significantly different from the original.

First, the dataset contained many missing values. 67 variables contained missing values, and 298,909 rows contained missing values. We decided that a sample size of roughly 8,000 was still large enough and that it was more worth it to keep the variables than the rows. We also eliminated the 16 categorical variables as our modeling methods (KNN, Logistic Regression, Random Forest, and Naive Bayes) couldn’t include categorical variables. We also separated the target variable of fraud from the features. We then scaled the data so that all features would be put on the same scale and therefore would be prioritized evenly.

Next, we used SMOTE (oversampling) as the data was heavily imbalanced. Data imbalance is a common problem in credit card fraud datasets as fraudulent purchases are far less common than regular ones. Originally, the dataset was roughly 1% fraudulent. Even after trimming rows with missing value, it was only 6% fraudulent. SMOTE made the dataset half fraudulent, half not. We used oversampling rather than under sampling because the number of rows had been trimmed so much in the removal of missing values.

Next, we used PCA (principal component analysis) to trim down the dataset further to only the most important variables. 48 of the 105 variables accounted for 90% of the variance in the dataset, so we only kept those 48. Finally, the data was ready to go into modeling.

1. **KNN Modeling**

Our first model was KNN classification, which we tested with k=1, 2, 3, 5, 7, 9, and 11. With one neighbor, the KNN’s validation accuracy was 88.70% and the testing accuracy was 88.06%. With two neighbors, the validation accuracy was 91.15% and the testing accuracy was 90.72%. With three, validation accuracy was 84.12% and the testing accuracy was 83.23%. With five, validation was 81.08% and testing accuracy was 79.33%. Accuracy decreased further with more neighbors in the KNN model, ending at roughly 75% for testing and validation accuracy with 11 neighbors. Every single KNN model had a higher f1 score on cases of fraud than non-fraud by a range of 5-12%. Also, every KNN model had a perfect recall score for fraud cases and a perfect precision score for non-fraud cases, showing that fraud cases were all correctly identified. The imperfection was in false positives, as shown by the precision score for fraud and in the lower recall score for non-fraud cases. This is great for applicability, as it is better to predict fraud and deny a payment out of precaution than to let cases of fraud go undetected or unpredicted. However, the significant statistic on false negatives also poses a question to us. Our best guess so far is that this might be due to the oversampling technique we used. Overall, the 2-neighbor model was clearly the most accurate. For further improvements, techniques like handling class imbalance, using different sampling methods, or exploring more complex models could be considered.

|  |  |
| --- | --- |
| A graph of a graph showing the size of a number of training scores  Description automatically generated with medium confidence | A blue squares with white text  Description automatically generated |

*Figure 1: Learning curve for KNN*

*Figure 2: Confusion matrix for KNN*

**3. Logistic Regression**

Fraud detection is a binary classification task, where the target is to determine if a transaction is fraudulent (1) or not (0), and Logistic Regression is specifically designed for binary classification problems, making it a suitable choice for our project. Logistic regression is also computationally efficient, making it suitable for quick testing and baseline model development.

One of the key strengths of the logistic regression model in this case is its reasonable precision and recall. The model performs better in detecting fraudulent transactions (class 1) than non-fraudulent ones (class 0). Specifically, the recall for fraudulent transactions is 71% on the validation set and 73% on the test set. Additionally, logistic regression serves as a robust baseline model for comparison in this case. It provides a starting point against which the performance of more complex models, such as Random Forests, can be evaluated. This helps determine whether these advanced methods genuinely add value to the prediction task.

Despite its strengths, the model has several weaknesses. The model achieves a validation accuracy of 70.22% and a test accuracy of 70.42%, which suggests that it does not generalize well. Precision and recall scores for both positive and negative cases are roughly 70%, showing that error in the model was spread out evenly. However, the significant number of false negatives means fraudulent transactions are sometimes misclassified as non-fraudulent. In real-world fraud detection, this is particularly problematic, as failing to detect fraud can have severe consequences for financial institutions that implement this model, and institutions would much rather use something like the KNN model which may suffer from false positives but don’t miss actual fraud cases.

The main limitation of logistic regression is its reliance on linear assumptions. It assumes a linear relationship between the features and the log-odds of the target variable. However, real-world data, especially in fraud detection, often exhibit complex, non-linear patterns that this model cannot capture. Moreover, logistic regression may underperform compared to more sophisticated, non-linear models when dealing with larger or more complex datasets (in our case this dataset has nearly 50 features with thousands of observations). While oversampling techniques like SMOTE were used to address class imbalance, logistic regression still struggles to handle such imbalanced datasets effectively. This results in suboptimal performance, particularly in terms of recall and precision for the minority class.

|  |  |
| --- | --- |
| A graph of a graph showing the growth of logistic regression  Description automatically generated | A blue squares with white text  Description automatically generated |

*Figure 3: Learning curve for Logistic Regression*

*Figure 4: Confusion matrix for Logistic Regression*

**4. Naive Bayes (Bernoulli)**

The results from your Bernoulli Naive Bayes (BNB) model suggest a moderate level of performance, with some room for improvement. On both the validation and testing sets, the model achieved around 65% to 67.6% accuracy, indicating that it correctly predicted the class labels for over half of the instances. However, while these results are decent, they also suggest that the model has room for refinement, especially if higher accuracy is desired for more critical applications.

Looking at the classification report for both the validation and testing sets, we can see that precision and recall are fairly balanced for both classes, with precision for class 1 being slightly better than recall. This indicates that the model is more conservative when predicting class 1, making fewer false positive errors, but occasionally missing some true instances of class 1 (as reflected in the false negatives). The F1-scores for both classes are also fairly similar, around 0.65 for the validation set and 0.67 for the testing set, indicating a balanced trade-off between precision and recall. This balance is crucial in many applications where both false positives and false negatives can carry significant consequences.

The confusion matrices provide a clearer picture of the model's performance on each class. In both the validation and testing sets, the model tends to perform better with class 0, with higher recall, suggesting that it is more adept at identifying instances of class 0. However, the model also seems to struggle somewhat with class 1, as seen in the higher number of false negatives for class 1 in both the validation and testing sets. This could be a result of class imbalance or suboptimal model settings.

To improve performance, tuning hyperparameters such as smoothing parameters in the Naive Bayes algorithm could help enhance the model's accuracy. Additionally, exploring feature engineering techniques, or using methods to address any class imbalance, such as oversampling the minority class, may lead to better classification results. Furthermore, evaluating other models or combining models using techniques like ensemble methods might be beneficial for capturing patterns that Bernoulli Naive Bayes might not fully capture. Despite these opportunities for improvement, the model has shown consistent performance across validation and testing sets, which indicates that it is generalizing well to unseen data.

|  |  |
| --- | --- |
| A graph with a line graph and numbers  Description automatically generated with medium confidence | A blue squares with white text  Description automatically generated |

*Figure 5: Learning curve for Naïve Bayes*

*Figure 6: Confusion matrix for Naïve Bayes*

**5. Random Forest Classifier**

|  |  |
| --- | --- |
| A graph with a red line  Description automatically generated | A blue squares with white text  Description automatically generated |

*Figure 7: Learning curve for Random Forest*

*Figure 8: Confusion matrix for Random Forest*

The Random Forest Classifier was very accurate in both validation and testing. Accuracy scores were 95.82% and 95.30%, respectively. Both precision and recall for fraud and non-fraud cases was between 95% and 97% for both validation and testing datasets except for testing data non-fraud recall, which was 94%. This is exceptionally accurate and suggests that a model using multiple decision trees may be the most useful in predicting credit card fraud. A drawback of Random Forest is the complex computation, but the code ran very quickly for us on this dataset. It may be slower or more difficult on bigger datasets, however.

Overall, financial institutions seeking the most accurate model may want to use the Random Forest Classifier model. However, the KNN model with two neighbors also provides great accuracy while entirely avoiding false negative cases that are far more costly for the institutions using fraud detection models than false positives. However, legal issues may come up with a model that consistently denies non-fraudulent customers, or those customers may use a different financial institution. Overall, the undeniable accuracy suggests that the best model is the Random Forest Model.

**6. Conclusion**

**A graph of a model

Description automatically generated with medium confidence**

*Figure 9: ROC curves across all models*

*A graph of different colored bars

Description automatically generated*

*Figure 10: Evaluation metrics across all models*

In this project, we explored various machine learning models to predict credit card fraud, using a heavily preprocessed dataset to address issues such as missing values, data imbalance, and feature selection. Among the models tested, the Random Forest Classifier emerged as the most accurate, achieving over 95% accuracy in both validation and testing, with balanced precision and recall across fraud and non-fraud cases. While its computational complexity could pose challenges on larger datasets, it remains the most reliable model for detecting credit card fraud. The KNN model with k=2 also performed well, offering high accuracy and perfect recall for fraud cases, making it a practical alternative for applications where avoiding false negatives is critical. Logistic Regression and Naive Bayes, while less effective, provided valuable insights and served as solid starting points for comparison. Overall, the Random Forest Classifier is the best choice for financial institutions seeking a highly accurate and robust solution for fraud detection. Future work could focus on improving scalability and exploring advanced techniques, such as ensemble learning or deep learning, to further enhance predictive performance.

**GitHub repository:** <https://github.com/amandajalta/data300final24/tree/main>

**Team roles and contributions:**

Amanda: Naïve Bayes model, presentation slides

Liam: data preprocessing, edit & finalize report

Abhik: KNN model, Random Forest model

Charlene: LR model, metrics visualization, poster