Credit Card Fraud Detection

Angie, Devika, Lesther

I. Introduction

What do we plan to do?

Credit card transactions are part of our everyday activities such as online payments, in-store purchases, transfers between accounts, etc. After COVID-19, the growth of these transactions has significantly increased. According to [4] Federal Trade Commission's (FTC) Annual Data Book of 2020 statistics report, there have been 66,090 cases of credit card fraud that have affected customers world wide. This causes the importance of offering banks and different entities the opportunity to quickly identify credit card fraud by creating a model that can classify these daily activities.

In this project, we train several binary classifiers using KNN (k-nearest neighbors), random forest, naive bayes with important techniques to handle heavily imbalanced data like under sampler, over sampler, SMOTE to evaluate the past transaction details from our dataset and understand customers' behavior and possible patterns.

What is the data?

We used a dataset of credit card transactions in Europe from September 2013, created by Worldline and the Machine Learning Group of ULB (Université Libre de Bruxelles). The positive class accounts for much less than 1% of the dataset observations.

What are the challenges from imbalanced data?

Class imbalances within datasets are a common problem when using models to predict outcomes. A class imbalance occurs when the number of observations in one class is higher than the number of observations in another. This is certainly the case in our dataset, as mentioned above. This is because imbalanced datasets can severely hamper model accuracy since algorithms work best when the number of observations in each class is similar to each other. In our case, we cannot afford to have wrong predictions in the case of a fraudulent transaction because the cost of it is higher, there is a lot at stake for individuals who possess those credit cards. Models with imbalanced datasets tend to have poor predictive performance, specifically for the minority class. This is the irony of such datasets, the minority class usually tends to be the more important one. For non-fraudulent transactions, you could have nearly a hundred percent accuracy, versus a 0% accuracy for fraudulent transactions. The model’s overall accuracy could seem to look exceptional not because the model is able to accurately identify fraudulent transactions which occur more rarely but because most transactions will be non-fraudulent.

What do we propose?

Simply testing the model based on an accuracy score isn't as helpful as testing the model after resampling the data. This is why we tested each model after removing samples from the majority class, and adding samples from the minority class. These two methods each come with their own cost. Undersampling, or to simply put it, removing records of the majority class can lead to loss of information. Oversampling on the other hand duplicates the minority class which results in overfishing and results in a more generalized output.

II. Data

As we mentioned before, we use a severely imbalanced dataset created by Worldline and the Machine Learning Group of ULB. It consists of 284,807 credit card transactions in Europe from 2013. The positive class indicates that the transaction is fraudulent; there are 492 positive observations, accounting for 0.172% of the total or a 1:578 ratio approximately.

The dataset has 31 variables, including the target, V1 to V28, Time, Amount, and Class. Bounded by the confidentiality of the data, the authors do not provide background information and the original features. But, they do indicate that Features V1 to V28 are principal components resulting from Principal Component Analysis (PCA). Also, only Amount and Time are not components of PCA. Now, what are those two features? The amount is the amount of the transaction carried out, whereas Time represents the seconds elapsed between the transaction and the first transaction in the dataset.

There are no entries with missing values, so there is no need for any data imputation strategy required to train that model.

III. Algorithms

What models do we use? What are their drawbacks? What are they good for?

Our algorithm to test dataset are KNN, naive bayes, decision trees, and random forest algorithms. KNN is a fairly simple algorithm to use since it requires us to tune only a few hyperparameters, however, the run time for this algorithm was quite high since it executes itself in real-time which contributes to its speed. Naive Bayes converges much faster than all our models selected, however, it makes the assumption that our features are strictly independent of each other. We cannot validate that they are not as consumer spending certainly tends to have patterns involved. Decision Trees and Random Forests both support automatic feature interaction, and implicit feature importance, however, Random Forest is computationally more complex and quite slow with a dataset like ours.

What techniques do we use to handle the class imbalance?

Our project uses the imbalanced-learn python module which offers sophisticated methods for producing a balanced dataset. The "RandomUnderSampler" module picks random subsects from the majority class. "RandomOverSampler," on the other hand, produces random new samples for the minority class instead of duplicating instances. We also utilize the Synthetic Minority Oversampling Technique (SMOTE) which adds 'synthetic' points within the minority class using its k-nearest neighbors to produce new data points.

What Metrics do we use to measure the models and why?

Since our target instances, i.e. the existence of fraud make up a very small slice of our overall sample data used to train our models, we have to evaluate the model differently than normal machine learning models. This is classified as an imbalanced classification problem, hence as explained earlier, a simple evaluation metric can mislead us by showing a high accuracy based on the majority class. Metrics that support our model and provide greater insight include Precision, Recall, and F1 scores.

- Precision tries to calculate the percentage of reported fraudulent transactions that were actually fraudulent. This is useful since the cost of classifying a non-fraudulent transaction as fraudulent is too high.

- Recall on the other hand gives us the percentage of the total number of fraudulent transactions that were actually captured by each model. This measure helps us since it is absolutely critical that we identify every single fraudulent transaction and can sometimes be okay with classifying a non-fraudulent transaction as fraudulent, as it is better to be safe than sorry in matters of credit card misuse.

- The F1 score gives us a combination of both precision and recall scores since the F1 score is the harmonic mean of the two and often maximizing both precision and recall is challenging.

Table 1: Model Performance Across Strategies

Model	Accuracy	Precision	Recall	F1
KNN	0.9995	0.9162	0.8100	0.8592
KNN_Oversampling	0.9991	0.7223	0.8358	0.7736
KNN_Undersampling	0.9832	0.1204	0.8858	0.2007
KNN_Bagging	0.9991	0.9095	0.5467	0.6819
Naïve Bayes	0.9780	0.0617	0.8246	0.1148
Naïve Bayes_Oversampling	0.9742	0.0545	0.8526	0.1025

Model	Accuracy	Precision	Recall	F1
Naïve Bayes_Undersampling	0.9658	0.0431	0.8628	0.0821
Naïve Bayes_Bagging	0.9797	0.0667	0.8246	0.1233
Decision Tree	0.9992	0.7886	0.7562	0.7707
Decision Tree_Oversampling	0.9117	0.0174	0.8983	0.0342
Decision Tree_Undersampling	0.9992	0.7796	0.7157	0.7452
Decision Tree_SMOTE	0.9981	0.4769	0.8044	0.5967
Random Forest	0.9995	0.9264	0.7475	0.8237
Random Forest_Oversampling	0.9995	0.9119	0.7600	0.8274
Random Forest_Undersampling	0.9748	0.0586	0.8984	0.1100
Random Forest_SMOTE	0.9995	0.8809	0.8225	0.8496
Hard Voting Classifier	0.9996	0.9000	0.8521	0.8731

What feature selection methods do we use?

Before getting ready to test a model its very important to perform feature selection to avoid over fitting on redundant data due to this we tried Recursive Feature Elimination with Cross-Validation (RFECV) and Sequential Forward Search (SFS). RFECV is a feature selection method that takes in feature weights, in our case we calculate them with Random Forest, and then start removing the weakest features recursively by considering fewer features using CV until it reaches an optimal number of features. On the other hand, Sequential Forward Search starts the model with no features, and on each loop will continuously add features that best improve the model's performance; it will stop once it reaches no additional performance increase in the model.

For the RFECV function to find an optimal set of features automatically, the CV process needs a scoring metric to evaluate the predictions, in our case we use the F1 score. In contrast, the SFS function needs an optimal number of features to select, so we aim to select half of the features and by using F1 as a performance measure as well. Given that Decision Trees and Random Forest handle the feature selection with entropy, we don't use RFECV or SFS with them. Also, It is important to mention that for the KNN model, we used a sample of 20% from the training Data to do the SFS and RFECV. As for the Naive Bayes model, RFECV uses a 10% sample from the training data to get the optimal features.

In table 2, we observe the performance of the feature selection functions applied to the KNN and naive bayes models. For our classifier naive bayes, we identified a significant increase

in our metric F1 when using SFS. As for KNN, based on the F1 score, we saw a worse performance using RFECV and SFS than without feature selection.

Table 2: Model Performance Using Feature Selection

Model	Accuracy	Precision	Recall	F1
KNN_RFECV	0.9995	0.9038	0.8100	0.8536
KNN_SFS	0.9995	0.8989	0.7850	0.8377
Naive Bayes_RFECV	0.9909	0.1419	0.8400	0.2427
Naive Bayes_SFS	0.9961	0.2840	0.8299	0.4228

From the different model iterations that we can see in tables 1 and 2, we picked the iteration with the best F1 performance to proceed with the evaluation of the test data. Hence, in table three, we have the final model selections and their respective performance across the different scores. The best KNN iteration is the model without any sampling technique, using k = 7 neighbors. For the Naive Bayes, we achieved the best F1 using SFS feature selection. Moreover, the best Decision Tree and Random Forest score is the model without any sampling strategy. Finally, we combine the KNN, Naive Bayes and Decision Tree models into a Voting Classifier and use a majority vote to predict the class labels.

Table 3: Final Model Selections Performance

Model	Accuracy	Precision	Recall	F1
KNN	0.9995	0.9162	0.8100	0.8592
Naïve Bayes_SFS	0.9961	0.2840	0.8299	0.4228
Decision Tree	0.9992	0.7886	0.7562	0.7707
Random Forest	0.9995	0.9264	0.7475	0.8274
Hard Voting Classifier	0.9996	0.9000	0.8521	0.8731

IV. Results

Our best model is random forest with an F1 score of 0.8586, this classifier constantly improves by using its building functionality to handle the amount of features this data set has, this logic is bagging because it decorrelates meaning at each split it validates the feature to only select the smallest subset instead of all of them. Therefore, we didn't have to add feature

selection to this model and at the same time we understand why its computational time is a bit higher. We use the F1 score to measure the performance of our model which indicates harmonic mean of Precision and Recall to evaluate the correct classification of the positive class having in consideration that our data is heavily imbalanced our main focus is that this model is able to correctly classify the minority which represent the fraud transactions. Closely following is the Hard Voting Classifier with an F1 score of 0.8367, which combines the best of the best KNN, naive bayes, and decision tree to make predictions. Finally, the worst performance corresponds to the naive bayes with an F1 of 0.3934, a precision of .25, and a recall of 0.85. The Naive Bayes precision means that it is correct only 25% of the time when it flags a transaction as fraudulent.

Table 4: Test Data Performance Results

Model	Accuracy	Precision	Recall	F1
Random Forest	0.9995	0.8500	0.8673	0.8586
Hard Voting Classifier	0.9994	0.8367	0.8367	0.8367
KNN	0.9994	0.8810	0.7551	0.8132
Decision Tree	0.9993	0.7788	0.8265	0.8020
Naive Bayes_SFS	0.9955	0.2553	0.8571	0.3934

V. Conclusion

Training and test data results summary?

Our overall testing phase consisted in selecting our classifiers KNN, Naive Bayes, Decision Trees, Random Forest using cross validation (CV) and running each one as a base algorithm to evaluate their performance. Next applied feature techniques like Random Under Sampler, Random Over Sampler, SMOTE, etc that are used to handle imbalance dataset to understand how each model reacted, here main expectation was to see a significant change in performance since based on [9]Machine Learning recommendation this helps standard models to avoid leaning towards the majority vote but in our case we didn't see any major improvements here. Our last phase on training was to run models through feature selection this step is very important to avoid our model over fitting on redundant data, making incorrect decision due to noise, reduces the training time and improves final metrics although this was applied specifically for KNN & Naive bayes since Decision Tree and Random Forest have this functionality buildedin. Due to the conditions of our imbalance dataset we saw the need of sampling our classifiers to obtain best features and avoid high computation. On another hand, once we trained our data and got best result out of each proceed to prepare steps to test our selected models using a voting classifier with hyperparameter set as hard to make the final decision based on predictions that appear the most, this result in to Random Forest with an F1 score of 0.8586.

Is there any margin for improvement? What can we do?

- Currently, our model only utilizes the best model to conduct feature selection. We made this choice due to time constraints within the scope of our project. Ideally, we would have liked to run feature selection on each one of our models with the multiple sampling techniques we tested.

- Due to resource constraints, our model currently ran the K-Nearest-Neighbor algorithm using 20% of the dataset. We have learned that KNN functions better with more data, however, due to limited time, we chose to run KNN on a fraction of our dataset. The results and outcome through the KNN model could have been different. The KNN model also currently trains on only that fraction of the data.

- In our model, we ran Random Forest on 10% of our data, however, this algorithm was trained on the entire dataset which can provide greater accuracy. We could have ran the full data set to evaluate the training performance to confirm if better results.
- We could certainly improve our feature selection methods, as of now, the only insight we had into the relationship of the features to each other was through the correlation matrix. Due to security reasons, the data source did not provide any details or background information on our feature list V1 - V28. Ideally, we would have loved to do feature selection based on some additional theoretical knowledge of the data.

Any recommendations?

- In a future iteration of our project, we recommend incorporating cost-sensitive learning. This would provide greater sensitivity to our minority class and take into account the cost of missing out on a fraudulent transaction. The key difference between cost-sensitive learning and cost-insensitive learning is that cost-sensitive learning treats the different misclassifications differently. The model would create a cost-matrix for a binary classification problem and either create cost-sensitive classifiers or take existing cost-blind classifiers and turn them into cost-sensitive ones.

- We would also recommend using other metrics to measure performance such as Area Under the Precision-Recall Curve (AUCPR), this kind of metric is better suited for imbalanced data since ROC's (receiver operating characteristic curve) tend to provide an excessively optimistic view of the performance.

VI. Reference

[1]Brownlee, J. (2020, January 14). *Why is imbalanced classification difficult?* Machine Learning Mastery. Retrieved May 7, 2022, from https://machinelearningmastery.com/imbalanced-classification-is-hard/

[2]Google. (n.d.). *Classification: Precision and recall | machine learning crash course | google developers*. Google. Retrieved May 6, 2022, from https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall?hl=es_419

[3]*Imbalanced classification: Handling imbalanced data using Python*. Analytics Vidhya. (2020, July 24). Retrieved May 6, 2022, from https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/

[4]Mint. (2021, June 21). *25 credit card fraud statistics to know in 2021 + 5 steps for reporting fraud*. MintLife Blog. Retrieved May 9, 2022, from https://mint.intuit.com/blog/planning/credit-card-fraud-statistics/

[5]Saxena, S. (2018, May 13). *Precision vs recall*. Medium. Retrieved May 6, 2022, from https://medium.com/@shrutisaxena0617/precision-vs-recall-386cf9f89488

[6]Ling, C. X., & Sheng, V. S. (2008). *Cost-Sensitive Learning and the Class Imbalance Problem* . Charles X. Ling's publications List (Selected). Retrieved May 9, 2022, from https://csd.uwo.ca/~xling/papers.html

[7]1.13. feature selection. scikit. (2007). Retrieved May 9, 2022, from https://scikit-learn.org/stable/modules/feature_selection.html

[8]Boyd, K., Eng, K.H., Page, C.D. (2013). Area under the Precision-Recall Curve: Point Estimates and Confidence Intervals. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds) Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2013. Lecture Notes in Computer Science(), vol 8190. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-40994-3_29

[9]Agarwal, R. (2022, April 26). *The 5 most useful techniques to handle imbalanced datasets*. Medium. Retrieved May 9, 2022, from https://towardsdatascience.com/the-5-most-useful-techniques-to-handle-imbalanced-datasets-6cdba096d55a#:~:text=1.,Random%20Undersampling%20and%20Oversampling&text=A%20widely%20adopted%20and%20perhaps,class%20(over%2Dsampling).

VII. Appendix

Figure A1: Feature Correlations Heatmap

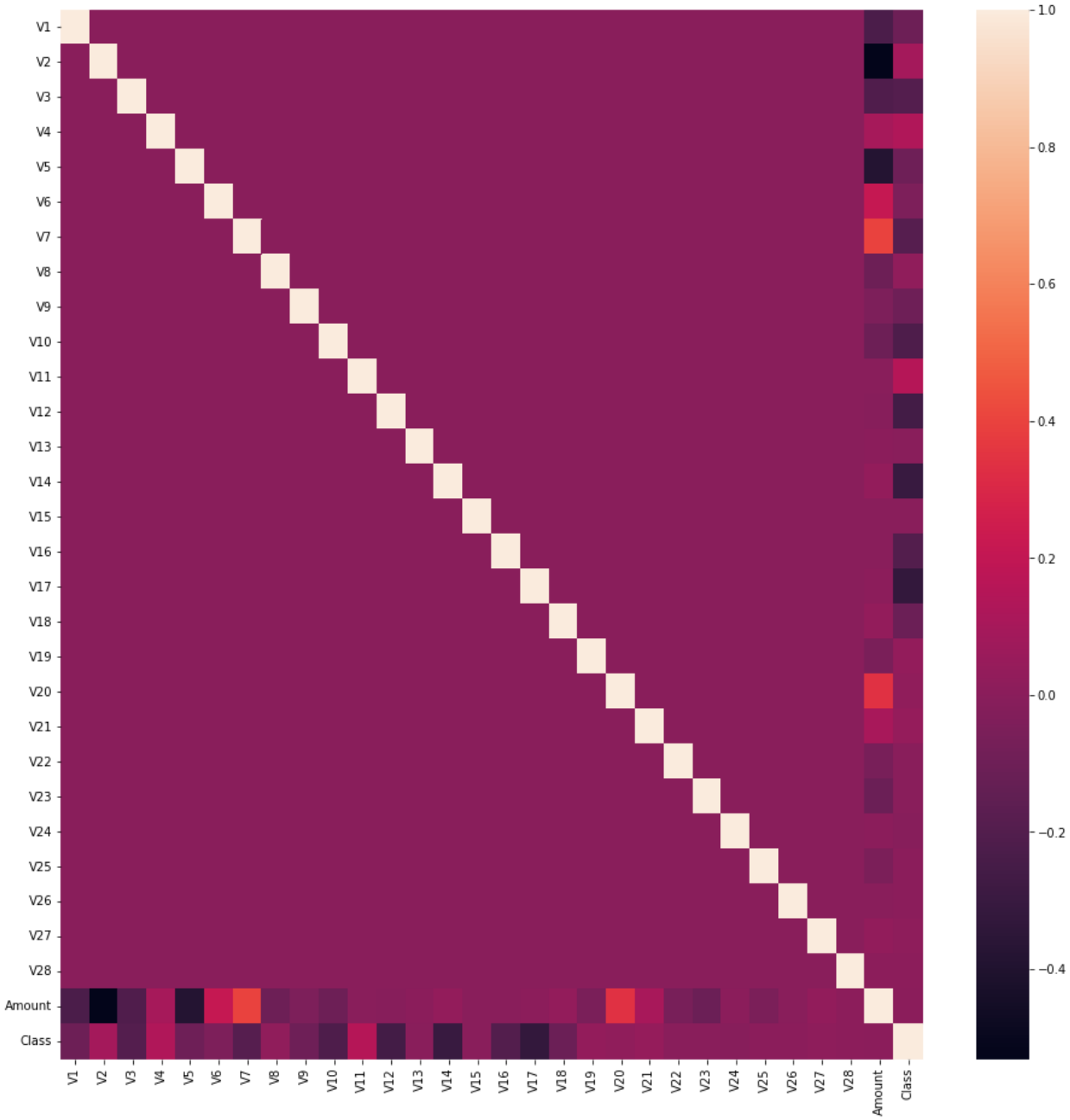


Figure A2: Features Histogram

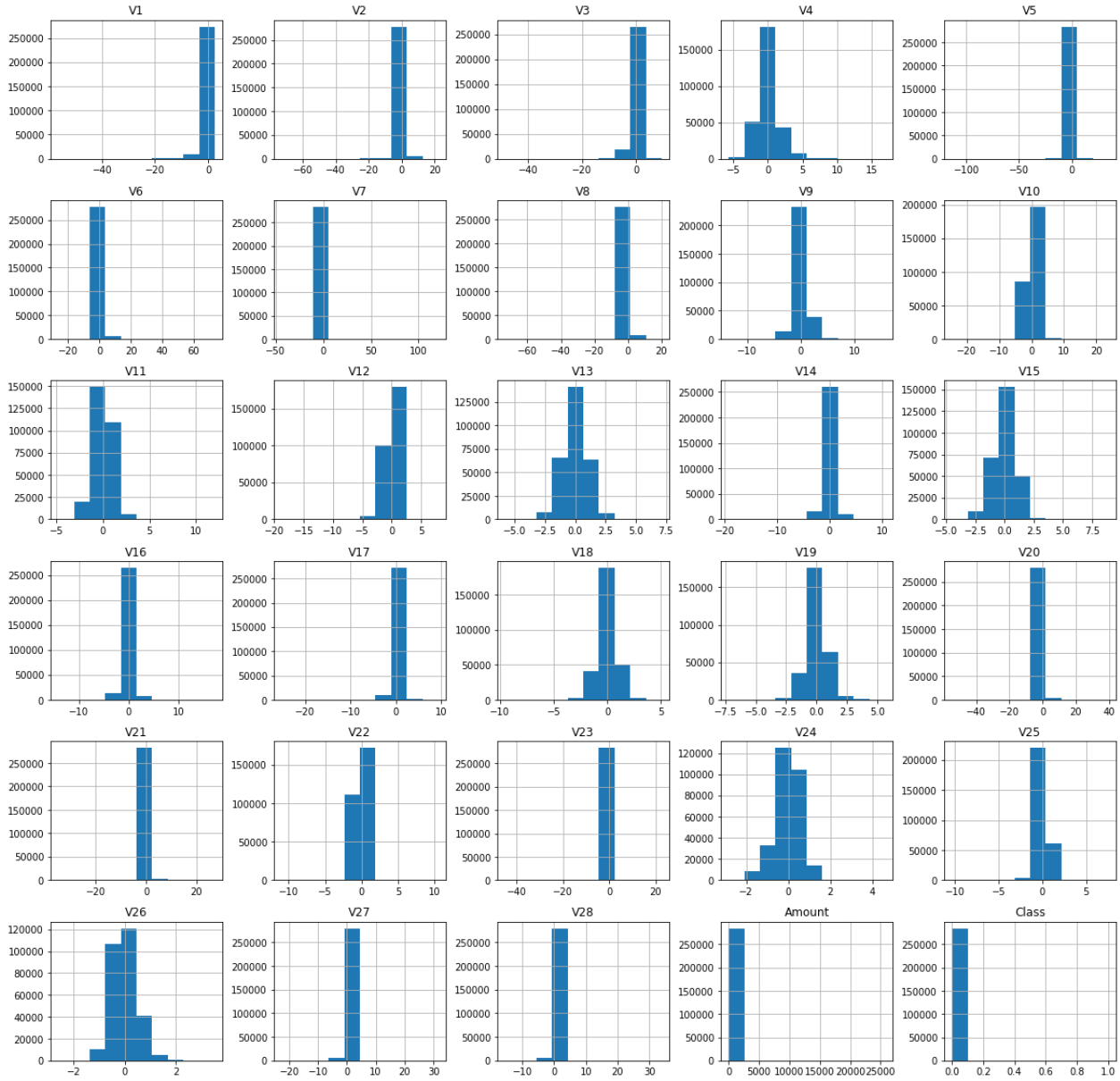
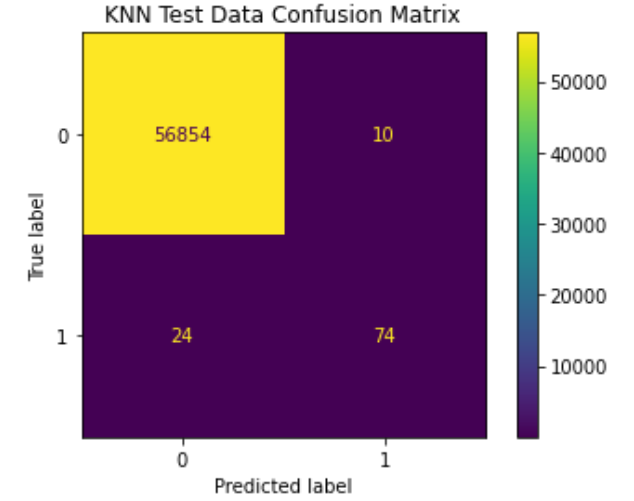
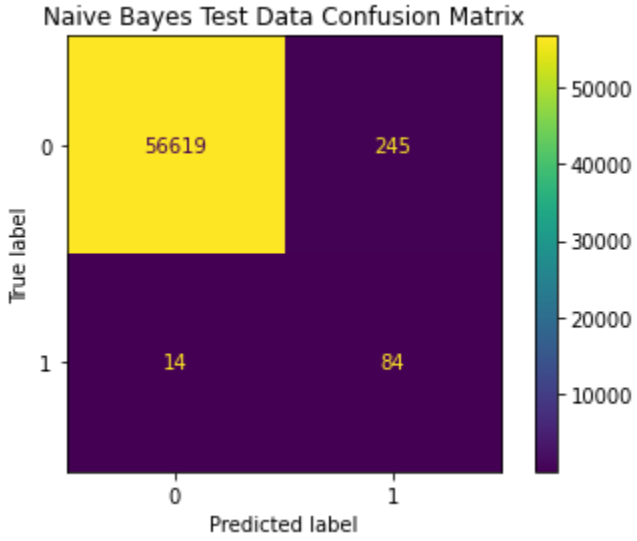


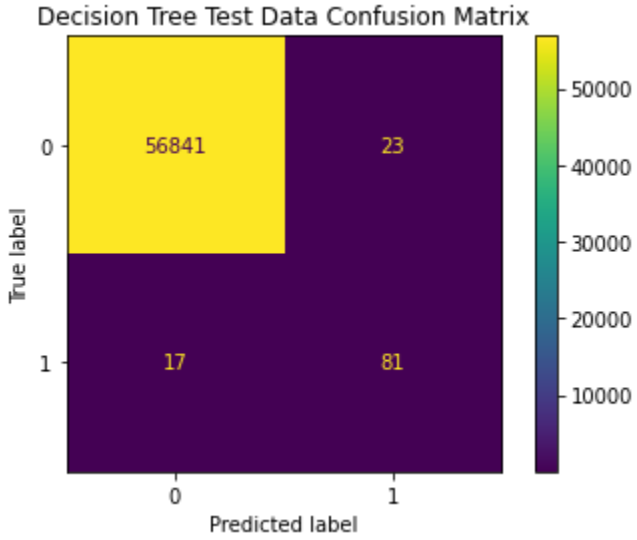
Table A1: Best SFS features with Naive Bayes

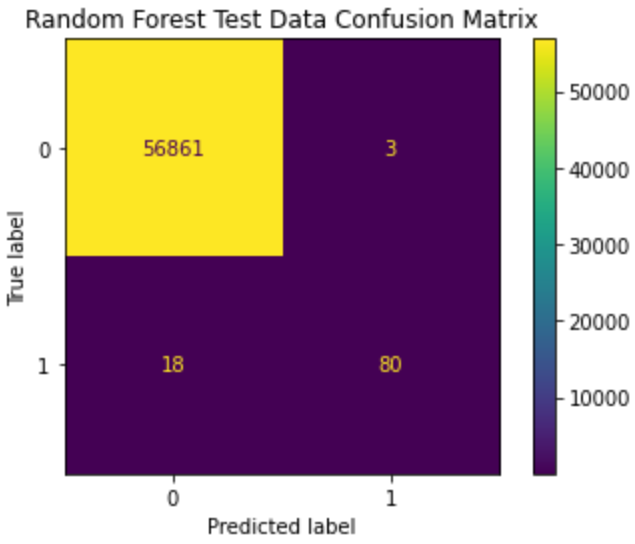
Features
V4
V6
V9
V11
V12
V13
V14
V15
V16
V18
V19
V22
V24
V26

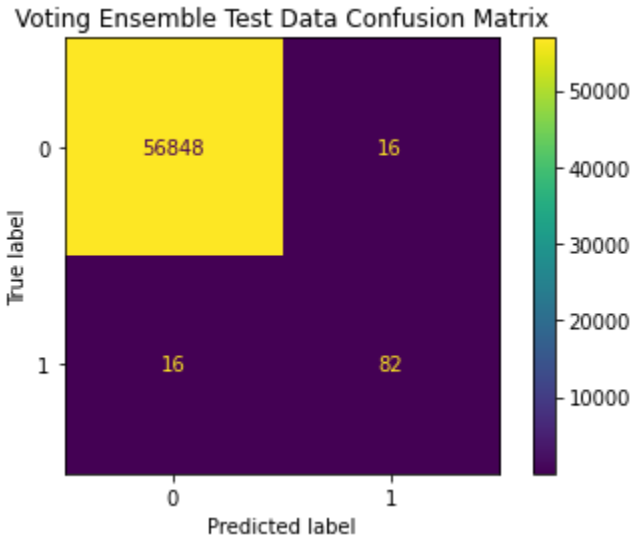
Figure A3: Best Models Confusion Matrix








Random Forest Test Data Confusion Matrix
56861
3
18
80
0
1
True label
0
1
Predicted label
50000
40000
30000
20000
10000


Voting Ensemble Test Data Confusion Matrix
56848
16
16
82
0
1
True label
0
1
Predicted label
50000
40000
30000
20000
10000