Supervised Methods for Credit Card Fraud Detection

Tyler Scott
CSCI 4502-001B (Distance)
Student ID: 103289678
tysc7237@colorado.edu

Christina Nguyen
CSCI 5502-001B (Distance)
Student ID: 830302322
nguyencq@colorado.edu

Nicholas Clement
CSCI 4502-001B (Distance)
Student ID: 103151912
nicl7004@colorado.edu

Christopher Struckman
CSCI 4502-001B (Distance)
Student ID: 102391835
christopher.struckman@colorado.edu

ABSTRACT

With the recent surge in online transactions and e-commerce, credit card fraud, which is the illegal and unauthorized use of credit cards or debit cards, has risen. This is a major problem for consumers, credit card companies, and banks, who accept the cost burden associated with the transactions. Prior research has utilized data mining and machine learning techniques for automatically detecting credit card fraud, but with the recent advances in machine learning—programming tools, ensemble methods, deep neural networks, etc.—we believe there is room for improvement. We have analyzed several supervised machine learning models, including naive Bayes, logistic regression, ensemble methods, and more, as well as several metrics for evaluating the models. With the goal of determining the best overall supervised model for the task, it was determined that the random forest and bagging models outperformed other models with an F-score and average precision of approximately 0.84 and 0.83, respectively. This is compared to the naive Bayes baseline model, which achieved an F-score and average precision of 0.13 and 0.11, respectively. By utilizing more sophisticated machine learning models, previously unused for credit card fraud detection, we believe our results will help advance fraud detection and highlight ways to improve current techniques used in industry.

1 INTRODUCTION

With the increase of online transactions, the opportunities for massive credit card fraud also increase. As techniques for stealing credit cards shift towards quick, massive transactions, the cost burden rises for card companies and consumers. Being able to detect fraud early on, and rapidly adapting to new fraudulent methods, can save millions of dollars per year. Since most existing research in this area comes before the surge in online shopping and the recent wave of machine learning, we believe that our project will help guide new work with respect to using technology to detect ever-changing approaches to credit card fraud.

The goal of this research is to apply supervised data mining techniques to the problem of credit card fraud, specifically behavioral fraud, which is defined as a fraudulent transaction where a credit card number was stolen. We aim to go beyond the existing work with binary classifiers and explore sophisticated deep learning models and ensemble methods, such as bagging and boosting, for detecting fraud. Prior work, as detailed in Section 2, has focused on simple classifiers such as logistic regression and perceptrons, which motivates the application of recent, more sophisticated models. However, to facilitate comparisons with previous work, we will also explore simpler binary classifiers such as logistic regression and naive Bayes. Credit card fraud detection datasets are inherently imbalanced, which is a challenge that both past researchers, as well as our team sought to overcome.

Section 4 contains the results from our work. We implemented several models, including naive Bayes, logistic regression, a deep neural network, a decision tree, a random forest, bagging, boosting, and a support vector machine, and compared the models using several metrics, including accuracy, area under the ROC curve, precision, recall, F-score, average precision, and training time. Due to the imbalance in the data, accuracy and area under the ROC curve were not satisfactory differentiators of classifier performance. Instead, F-score and average precision became the most important determinants of performance. We determined that the random forest and bagging models performed best, with an F-score and average precision score of approximately of 0.84 and 0.83, respectively. This was a huge improvement on our baseline naive Bayes model, which achieved an F-score and average precision of 0.13 and 0.11, respectively. Further analysis on the performance of the different classifiers is detailed in Section 4.

The remainder of the report is organized into the following sections:

- Related Work: past research including their challenges and results
- Methodology: description of the research we conducted including challenges and preliminary results
  - Data: description of the dataset including features and size
  - Initial Analysis: initial exploration of the dataset
  - Visualization: initial visualization of the dataset using a t-distributed stochastic neighbor embedding (t-SNE)
  - Supervised Methods: description of supervised models implemented, training procedures, and hyperparameter tuning
  - Tools: description of tools utilized
- Evaluation: analysis of the metrics computed and comparison of model performance
- Discussion: discussion of the lessons and limitations of the work including future work and improvements

- Conclusion: summary of credit card fraud detection and our results

2 RELATED WORK

Credit card fraud is not new, and the size of the issue in terms of costs and victims makes this an area previously explored. However, most studies took place before the rise in e-commerce and organized fraud crime rings [2]. We believe that these changes fundamentally alter the way credit card fraud is perpetrated. Thus, we aim to improve on credit card fraud detection by leveraging previous work in the field and recent advances in data mining. The prior literature we will use for reference and baseline model performance are:

- Data mining for credit card fraud: A comparative study, Bhattacharyya et al.
- Credit Card Fraud Detection Using Bayesian and Neural Networks, Maes et al.

Bhattacharyya et al. present three classification models: a logistic regression model, a support vector machine (SVM), and a random forest for automatically detecting credit card fraud. They note that almost all datasets for detecting fraudulent transactions are heavily skewed, leading to different training schemes and evaluation metrics. To adjust, they do random undersampling of the majority class (genuine transactions) [2]. They argue that this performs better than other approaches, such as oversampling the minority class or weighting predictions based on prior distributions for each of the two classes [2]. Table 1 below contains results from their research. The main limitations of this work is that they only explored relatively simple classifiers. For a problem that is so widespread, we believe that more models should be compared and analyzed, including sophisticated models, specifically ensemble methods and neural networks. Furthermore, this work was done in 2010, which was before many of the recent advances in neural networks were researched. As previously mentioned, we believe that recent advances in machine learning techniques will open a pathway for improvement in automatic credit card fraud detection.

Maes et al. focused on using Bayesian networks and neural networks for detecting credit card fraud. Notably, they use several different feed-forward multi-layer perceptrons trained using the Backpropagation algorithm [5]. Table 2 below contains receiver operating characteristic (ROC) curve results from the networks. One point to note from this research is that their dataset contains confidential features because of customer privacy [5]. This is very similar to our dataset, presented in Section 3.1, where the features are anonymized and transformed through Principal Components Analysis (PCA). The limitations in this work are that they used simple perceptron models, as opposed to deep neural networks, as seen in our work, however, Maes et al. conducted their research in 2002, which was much earlier than the adoption of deep neural networks. Also, their main metric was related to area under the ROC curve, which we found to be an inferior metric given imbalanced datasets. Using metrics such as F-score or average precision would be a better indicator of performance. This limitation made it challenging to compare our models with theirs.

A key claim in Maes et al. is that the loss from fraud outweighs the cost from investigating and stopping fraud. As such, we will adapt a similar stance and aim to minimize false negatives rather than false positives. To do this, a correlation analysis of attributes can help improve the results by indicating which highly correlated attributes might be unduly affecting results [5].

As these studies note, the training datasets for credit card fraud rely on correct categorization. There exists a non-zero probability in previous work that fraudulent transactions are misclassified as legitimate and legitimate transactions are misclassified as fraudulent [2, 5].

Our work, which builds on that of Bhattacharyya et al. and Maes et al., differs in that we explored a wide range of supervised classification models, which are evaluated using several metrics. This allows us to explore the best metrics for imbalanced datasets, as well as models that perform best for credit card fraud detection. As seen in Section 4, our results compare to those from Bhattacharyya et al.

3 METHODOLOGY

3.1 Data

The data we are using for our research is the *Credit Card Fraud Detection* (CCFD) dataset of European credit card transactions [1]. This dataset can be found at:

https://www.kaggle.com/dalpozz/creditcardfraud

The data contains $284,807$ examples, labeled into two classes: genuine transactions and fraudulent transactions, where the fraudulent transactions make up 0.172% of the total set. This results in $284,315$ genuine examples and 492 fraudulent examples. Since the data could potentially contain private information on the credit-card holders, the creators ran most of the features through the Principal Component Analysis (PCA) algorithm and renamed them to $V_1, V_2, \ldots, V_N$, where $N = 28$. The only features that are not renamed are the monetary amount of each transaction and the elapsed number of seconds since the first transaction. Thus, the proposed data mining research will refer to the features using their original notation, and lose context and interpretability of most dimensions of the data. We found that the data was clean and preprocessed, so data cleaning and preprocessing was not explicitly performed.

3.2 Initial Analysis

Before developing supervised classification models, we focused on analyzing and exploring the dataset from Section 3.1. As seen in Figure 1, plotting the amount of each fraud transaction over time showed that most transactions in the CCFD dataset were done at a very low value. This seems contrary to [5], which said that undetected fraudulent transactions cost more than investigating a fraudulent transaction. This could be because undetected fraudulent transactions can continue over a long period of time, or else are outweighed by the large purchases. The reasons are not central to our paper, but it is important for us to note that our methods will likely be trained to identify small amounts as fraudulent, rather than large transactions.

Table 1: Cross-validation performance of different techniques [2]

	Accuracy	Recall	Specificity	Precision	F	AUROC
LR	0.947	0.654	0.979	0.778	0.709	0.942
SVM	0.938	0.524	0.984	0.782	0.624	0.908
RF	0.962	0.727	0.987	0.86	0.787	0.953

Table 2: This table compares the results achieved with artificial neural networks for a false positive rate of respectively 10% and 15% [5]

Experiment	±10% false pos	±15% false pos
ANN-fig 2(a)	60% true pos	70% true pos
ANN-fig 2(b)	47% true pos	58% true pos
ANN-fig 2(c)	60% true pos	70% true pos

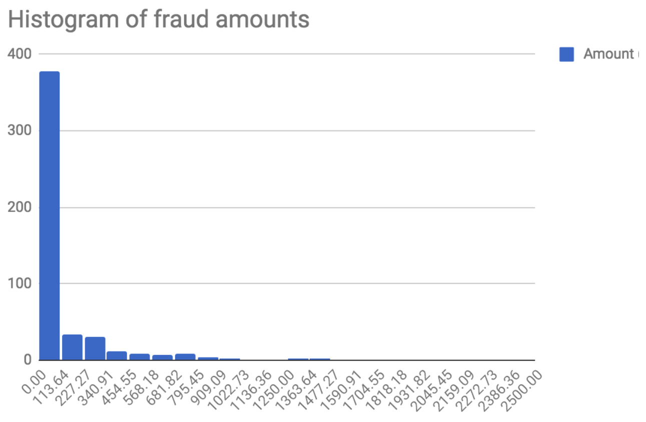


Figure 1: Histogram of the amounts for fraudulent credit card transactions in the CCFD dataset.

The creators of the dataset stated that the features $V_1, V_2, \dots, V_{28}$ resulted from PCA, so those features should be orthogonal. As a safety measure, we computed the dot product between pairs of feature vectors and ensured the resulting value was zero:

$$V_i \cdot V_j = 0 \quad \forall i = 1, \dots, 28, j = 1, \dots, 28, i \neq j$$

We found that this was indeed true. Furthermore, it wasn't explicitly stated whether dimensionality reduction had been conducted after finding an orthogonal basis of features from PCA, so we explored how well each feature separated genuine and fraudulent transactions. Figure 2 shows the distribution of each of the anonymous features, $V_1, \dots, V_{28}$, color-coded by class label. This allowed us to reduce the dimensions of our dataset by removing the features that didn't distinguish the two classes. We removed $V_{13}, V_{15}, V_{20}, V_{22}, V_{23}, V_{24}, V_{25}, V_{26}, V_{28}$ as a result of this analysis. The removal of these features may not significantly improve supervised classification performance using discriminative models such as logistic regression and neural networks because those models can set weight values to zero for uninformative features, but for generative models such as naive Bayes, removing these features can help model the posterior probability distribution of the class label given the data.

A simple attribute correlation across the entire dataset showed no correlation between any attributes. This makes sense since most of the features are from PCA, so they are pairwise-orthogonal. However, by splitting the dataset based on class, we found a marginal increase in correlation between attributes within a given class. Furthermore, for transactions classified as fraudulent, both the pairs of V_{16} and V_{17}, as well as V_{17} and V_{18} have correlation coefficients that are statistically significant with a significance level of 0.05 as seen in Figure 3. No attributes have strong negative correlations. Expanding the significance level to 0.1 revealed only a couple more highly correlated attributes, but going out to a significance level of 0.15 revealed a number more. One of the key attributes of our dataset is that the transactions are temporally linear. Interestingly, the attributes that are highly correlated for fraud also have the highest correlation with the time attribute. However, their correlation with the time attribute is not high (between 0.25 and 0.3), but they are the highest amongst all 28 attributes. In general, while not statistically significant, nearly all attributes were more highly correlated with time rather than with fraud amounts. Since the fraud amounts were overwhelmingly small, as we saw in Figure 1, any correlation between fraud transaction amount and attributes would be near 0, or pure chance.

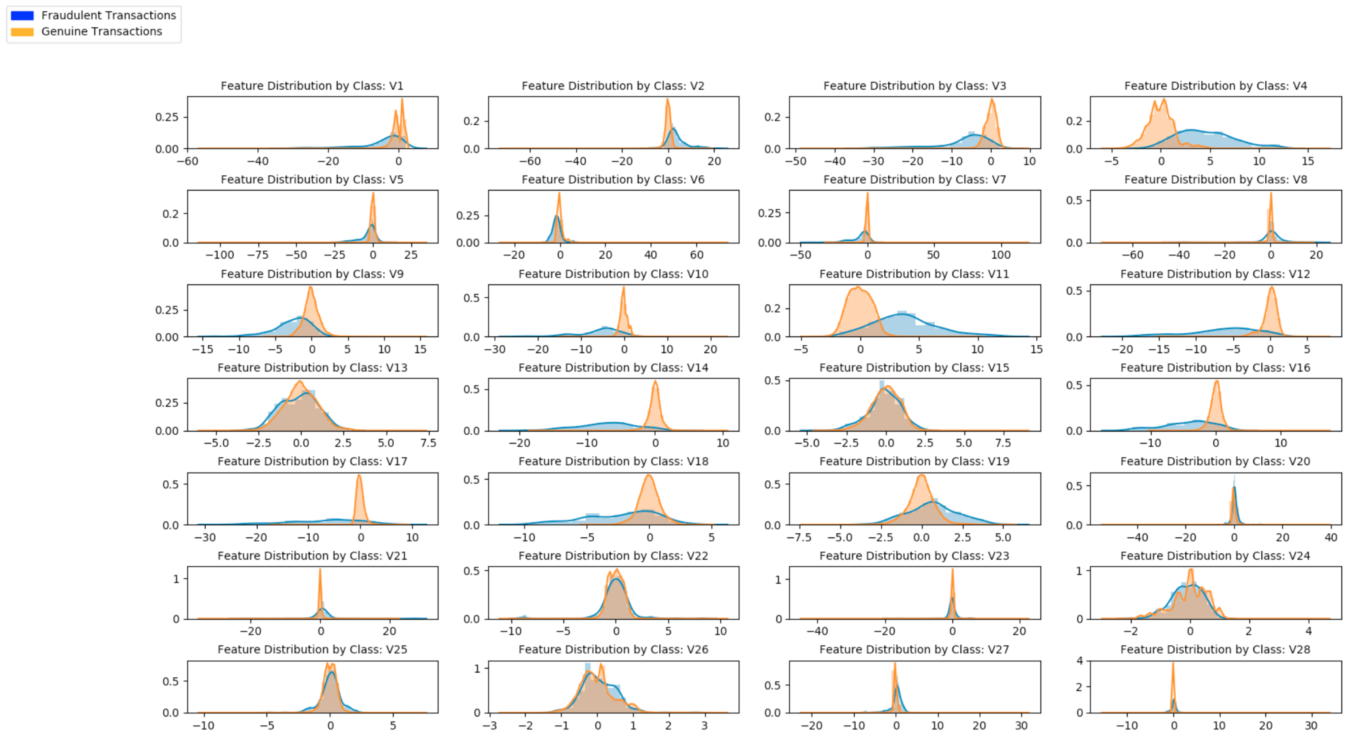


Figure 2: Distribution histograms for each of the anonymous features $V_1, \ldots, V_{28}$, color-coded by class label.

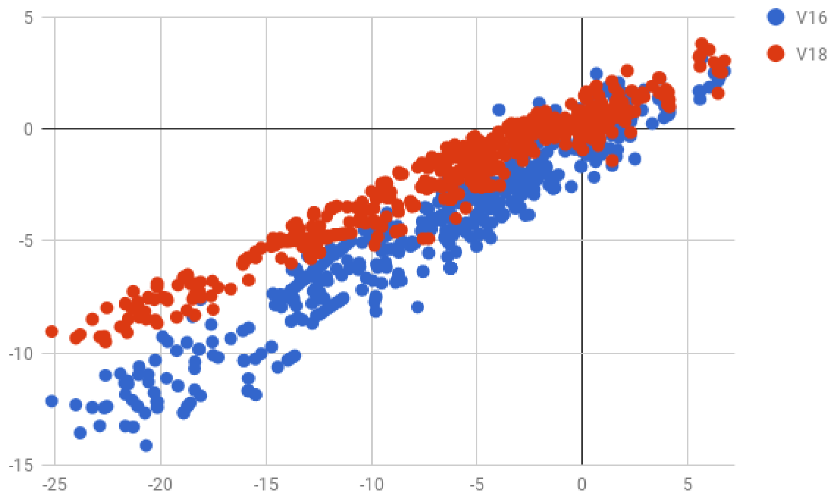
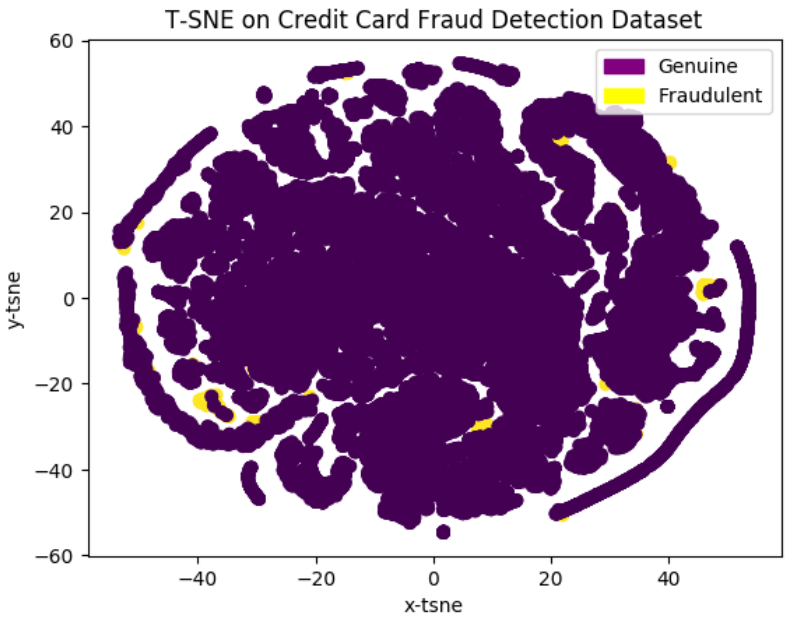


Figure 3: Correlation scatter plots with V_{17} as the x-axis, and V_{16} and V_{18} as points. This is for fraudulent transactions only.

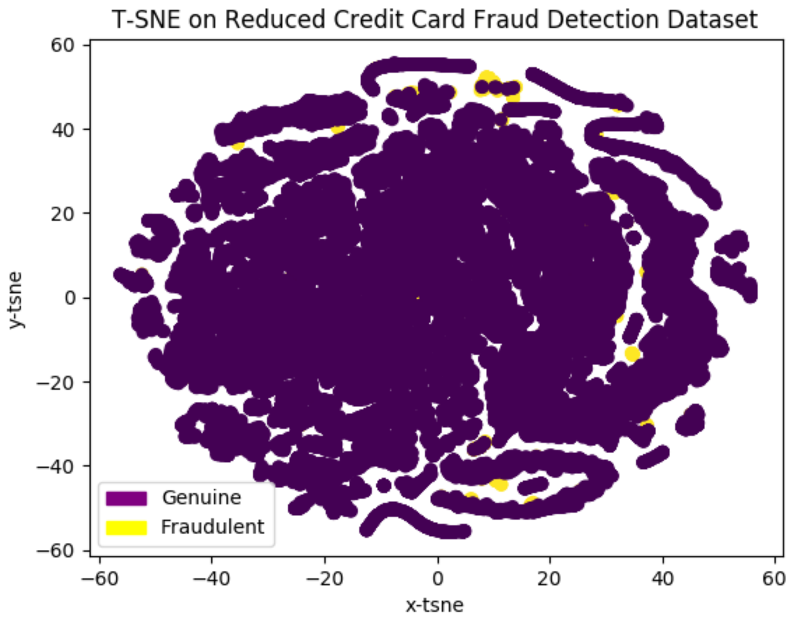
These correlation patterns do not hold for the genuine transactions, where there were no significant attribute-to-attribute correlations, even up to a significance level of .15. Unsurprisingly, the features that did not distinguish the two classes had the lowest correlation coefficients with all other attributes when comparing attribute-to-attribute.

3.3 Visualization

Credit card fraud detection is a domain that lacks human intuition regarding the applicability of machine learning. This is most likely because humans cannot visualize the many varying dimensions involved in a potential fraudulent transaction. As a result, we explored



(a) t-SNE on full dataset



(b) t-SNE on reduced dataset

Figure 4: t-SNE on 50,000 data points from the CCFD dataset

the t-distributed stochastic neighbor embedding (t-SNE) method for visualizing both classes of transactions. Figure 4 contains the results of t-SNE on 50,000 data points from both the full dataset and the reduced dataset. Both visualizations of the CCFD dataset are similar. This is slightly unexpected since the reduced dataset should, in theory, help separate fraudulent and genuine transactions. This is an indication that reducing the dataset may have a small impact on classification performance, but nonetheless, it is always useful from an efficiency and performance perspective to reduce redundancy in the feature space. The result of this visualization is also an indication that unsupervised approaches such as statistical clustering and anomaly detection may not be worth exploring. With respect to local proximity relationships between points, it doesn't appear to be that fraudulent transactions are anomalies compared to genuine transactions. Likewise, it doesn't appear that fraudulent transactions have distinct clusters compared to genuine. Also, since the fraudulent transactions appear to be very similar to some of the genuine transactions, this indicates that the supervised classifiers shouldn't necessarily get significantly high results. We should expect to see lower values for some of the evaluation metrics.

3.4 Supervised Methods

Since the CCFD dataset is labeled, supervised binary classification methods can be utilized to learn from the training examples and predict unseen, novel examples. As stated in Section 2, prior research on credit card fraud detection has focused on simple, binary classifiers such as logistic regression, random forests, and feed-forward multi-layer perceptrons. However, recent advances in classification may perform better than previous methods and improve on state-of-the-art results. We implemented several classification models including naive Bayes, logistic regression, a decision tree, a random forest, bagging, boosting, a support vector machine, and a deep neural network. For these models, we also removed uninformative features, as determined in Section 3.2. Also, all hyperparameters were tuned using a validation dataset, which consisted of 20% of the original data. The goal of implementing several different classification models was to compare them using a variety of evaluation metrics, as well as compare them to research conducted in the past. We thought it would be a useful contribution to explore several supervised models, highlighting metrics such as time complexity, accuracy, ability for online learning after training, and others. For more information on specific evaluation metrics, see Section 4.

3.4.1 Naive Bayes. As a baseline model, we implemented naive Bayes. The goal of the model was simply to provide a realistic measure of performance that we could compare to. Since naive Bayes is a generative model that relies heavily on the prior probability of each class, we expected the classifier to guess that each transaction was genuine. Also, the model doesn't rely on a loss function, like logistic regression and neural networks, so we couldn't weigh the fraudulent examples more heavily like in Sections 3.4.2 and 3.4.3. One extension we explored was manually setting the prior probabilities of each class. We tried various values of the priors between no weighing at all, and weighing fraudulent transactions equivalently to genuine transactions. It was determined that the original statistical priors performed best. The performance of the naive Bayes model can be found in Table 3.

3.4.2 Logistic Regression. We also implemented a logistic regression model to detect credit card fraud. Unlike naive Bayes, the logistic regression model uses a log-loss cost function, which allowed us to weight different examples. Thus, we empirically determined that weighing the fraudulent example eight times more than genuine transactions seemed to have the best performance. We also experimented with extreme cases, such as not using a weighing scheme and apply a weighing scheme that balanced the importance of each class (weigh fraudulent examples approximately 2000 times more). What we found is that when a weighting scheme is not applied, the classifier just guesses all transactions as genuine, however, when each class is weighted equally, the precision metric and F-score metric (which depends on precision) drop significantly. This intuitively makes sense because precision is defined as:

$$precision = \frac{TP}{TP+FP}$$

and since there are so many genuine transactions, the false-positive value is large relative to the true-positive value, driving precision to zero. Thus, using a value in the middle was desired, and a value of eight was determined to perform best based on validation set results.

3.4.3 Deep Feed-Forward Neural Network. A basic deep feed-forward neural network was implemented on the credit card fraud detection task. Several architectures, loss functions, and activation functions were used and evaluated using a held-out validation set. We tested both shallow and deep models with cross-entropy loss, logistic loss, and squared-error loss, as well as activation functions such as sigmoid, hyperbolic tangent, and rectified linear unit (ReLU). The network with best performance was the following:

- Input layer of 20 units (after removing features based on 3.2)
- 2 hidden layers of 64 units with hyperbolic tangent activation
- Output layer of 1 unit with sigmoid activation
- Weights and biases initialized using a random normal distribution with mean 0 and standard deviation of 1
- Weighted cross-entropy loss function where fraudulent transactions are heavily weighted (8 times more than genuine transactions)
- Adam optimizer trained for 500 epochs, learning rate of 0.01, and 256 examples per batch

3.4.4 Decision Tree. A simple decision tree model was implemented. Like several other models, a weighting scheme, where the fraudulent class was weighted 8 times more than the genuine class, was used. Also, since their are only 2 classes, splitting features using the Gini index performed better than information gain and gain ratio. Without setting any limit on the size of the tree, the model could get 100% accuracy, as well as an average precision and F-score of 1.0 on the training set. However, the model was heavily overfitting, so the performance on the validation set was very poor. To overcome this, we pruned any branches of the tree where the number of remaining samples was less than 125. This value was determined with the validation set and the results are based on this pruning technique.

3.4.5 Support Vector Machine. The support vector machine model was the most challenging to tune because the data is clearly not linearly separable, so a kernel was needed. We tested the three most popular kernels: linear, polynomial, and radial basis function (RBF). It was determined that the RBF kernel worked best. We initially found that the soft-margin SVM would heavily overfit, similar to the decision tree. Soft-margin support vector machines have a hyperparameter, C, which is used to determine the amount of slack allowed. The loss function is as follows:

$$L(\boldsymbol{w}, \boldsymbol{\epsilon}) = \boldsymbol{w}^T \boldsymbol{w} + C \sum_{i=1}^{n} \epsilon_i$$

where $\boldsymbol{w}$ is the weight vector of the decision boundary and $\boldsymbol{\epsilon}$ is the slack, distance on the wrong side of the decision boundary, for each of the i data points. If the goal is to minimize loss and C is large, then ϵ_i needs to be small, thus, the model will try to correctly classify each training example, leading to overfitting. If C is small, then there is room for slack and the term acts as regularization. Initially, C was large and the model was overfitting, but we lowered C to a value of 1.0, which ended up giving the best results.

3.4.6 Ensemble Methods. Three ensemble methods were implemented: a random forest, bagging, and boosting. We decided to implement all three because they each have slight differences and it was unclear which one would perform best given the highly imbalanced dataset.

The first ensemble method implemented was a random forest. Random forests not only bootstrap-sample the data, but each individual classifier in the ensemble only uses a subset of the total features. The idea is that only using a subset of features will reduce variance in the ensemble, which is a tactic for reducing overfitting. It is similar to adding a regularization term in logistic regression or a dropout layer in a neural network. Like several other classifiers, we determined that weighing fraudulent transactions 8 times more important than genuine transactions performed best. Also, like the decision tree, Gini index was the best method for splitting features.

The second ensemble method implemented was a bagging model. Like random forests, bagging bootstrap-samples the data for each individual classifier, but unlike random forests, it uses all of the features. So, each individual classifier has all possible information on each training example.

The final ensemble method implemented was a boosting model. Both random forests and bagging models use fully-developed classifiers for each individual model in the ensemble, such as a fully-developed decision tree. However, boosting converges under the assumption that each individual classifiers is weak, such as a "decision stump", or decision tree with a single decision. Random forests and bagging also employ an equal-vote scheme in which each classifier in the ensemble has an equal say in the final classification decision. Boosting, on the other hand, employs a weighted-vote scheme in which the classifiers of the ensemble that performed best have a heavier say in the final classification decision. Furthermore, as each classifier is trained, the training examples it classifies incorrectly get weighted heavier for the next classifier, so models, over time, will focus on the more challenging training examples. Boosting tends to perform better than random forests and bagging, however, one downfall is that boosting is more likely to cause overfitting, which can be inflated on imbalanced datasets.

3.5 Tools

For other researchers and peers who would like to explore credit card fraud detection, below is a description of the tools we used for our analysis. The goal is that others can build on our results and get started quickly by leveraging useful data mining tools. For many of the machine learning tasks, Scikit-learn [3] was the main library for development of the models, data preprocessing, validation, and associated evaluation. Tensorflow [4] was used in conjunction with Scikit-learn for development of the deep learning models. For simpler tasks such as data storage and transformation, Numpy [7] and Pandas [8] were used. Finally, for data analysis and visualization tasks, Matplotlib [6] and Seaborn [9] were used.

4 EVALUATION

4.1 Evaluation Metrics

As with many classification tasks, simple evaluation metrics such as accuracy are used to compare performance between models. This is most likely because accuracy is a simple measure to implement, it generalizes to more than just binary labels, and it is model-agnostic. However, one major drawback of accuracy is that it is assumed that there is an equal representation of examples from each class. In cases where the number of examples per class is skewed, accuracy is a poor and misleading measure. For example, if we consider a dataset with 99 examples from class A and 1 example from class B, then a classifier that simply guesses class A, regardless of the input features, will have an accuracy of 99%. It seems like the classifier is doing extremely well, when in fact, the classifier is learning nothing from the data. This is exactly the case we have with the CCFD dataset. For this dataset, 99.828% of the data is labeled as genuine transactions (see Section 3.1). To combat the misleading measure of accuracy, we determined that the following metrics were better indicators of classifier performance on imbalanced datasets:

- Recall
- Precision
- F-Score
- Area under the ROC curve (AUROC)
- Average precision (AP)

Due to the imbalance in the dataset, we chose to focus on metrics related to receiver operating characteristic (ROC) curves and precision-recall curves. These metrics are better indicators of model performance on skewed data, where higher values for all of the metrics are better. Furthermore, our main metric is average precision (AP), which is a better-calibrated numerical representation of area under the precision-recall curve, which is analogous to area under the ROC curve (AUROC). Precision-recall curves are very similar to ROC curves, however, precision-recall curves focus more on correct classification of the positive class (fraudulent class), whereas ROC curves focus on distinguishing between the two classes. In other words, ROC curves focus on both true positives and true negatives, however, we want to focus more on true positives, which aligns with precision-recall curves. For credit card fraud detection, fraudulent transactions are much more important to classify than genuine transactions, which naturally leads to using AP as the main metric. Both AUROC and AP are computed by sliding a decision threshold over either true positive rate and false positive rate for AUROC or precision and recall for average precision. By sliding a decision threshold across these spaces, it creates a curve which is known as ROC curves or precision-recall curves. Both AUROC and AP are measures of the area under these curves, which signify classifier performance, where a value of 1.0 is optimal. F-Score is another common metric for imbalanced data that computes a similar value to average precision:

$$F = \frac{2 \times precision \times recall}{precision + recall}$$
$$precision = \frac{truepos}{truepos + falsepos}$$
$$recall = \frac{truepos}{truepos + falseneg}$$

Both F-Score and AP can be substituted for one another and usually have similar values for the same classifier.

Table 3: Performance of different supervised classification models on the credit card fraud detection task. The dataset used in [2] is different, so relative comparisons should be used as opposed to absolute comparisons.

	Accuracy	Recall	Precision	F-Score	AUROC	Average Precision (AP)	Training Time (seconds)	Online Learning?
Logistic Regression [2]	0.947	0.654	0.778	0.709	0.942	N/A	N/A	Yes
SVM [2]	0.938	0.524	0.782	0.624	0.908	N/A	N/A	No
Random Forest [2]	0.962	0.727	0.86	0.787	0.953	N/A	N/A	No
Naive Bayes (Ours)	0.981	0.889	0.063	0.156	0.933	0.13	0.06	No
SVM (Ours)	0.9989	0.4167	0.8948	0.5676	0.7083	0.6484	118.25	No
Logistic Regression (Ours)	0.9993	0.808	0.782	0.795	0.904	0.74	1.88	Yes
Feed-forward Neural Net (Ours)	0.9994	0.8532	0.7812	0.8105	0.946	0.786	151.42	Yes
Decision Tree (Ours)	0.99893	0.8079	0.6576	0.7227	0.9036	0.6773	5.86	No
Boosting (Ours)	0.9994	0.7928	0.8929	0.8253	0.8845	0.8236	248.33	No
Random Forest (Ours)	0.9995	0.8048	0.8814	0.8408	0.9023	0.8313	5.32	No
Bagging (Ours)	0.9995	0.7665	0.9453	0.8457	0.8832	0.8265	30.80	No

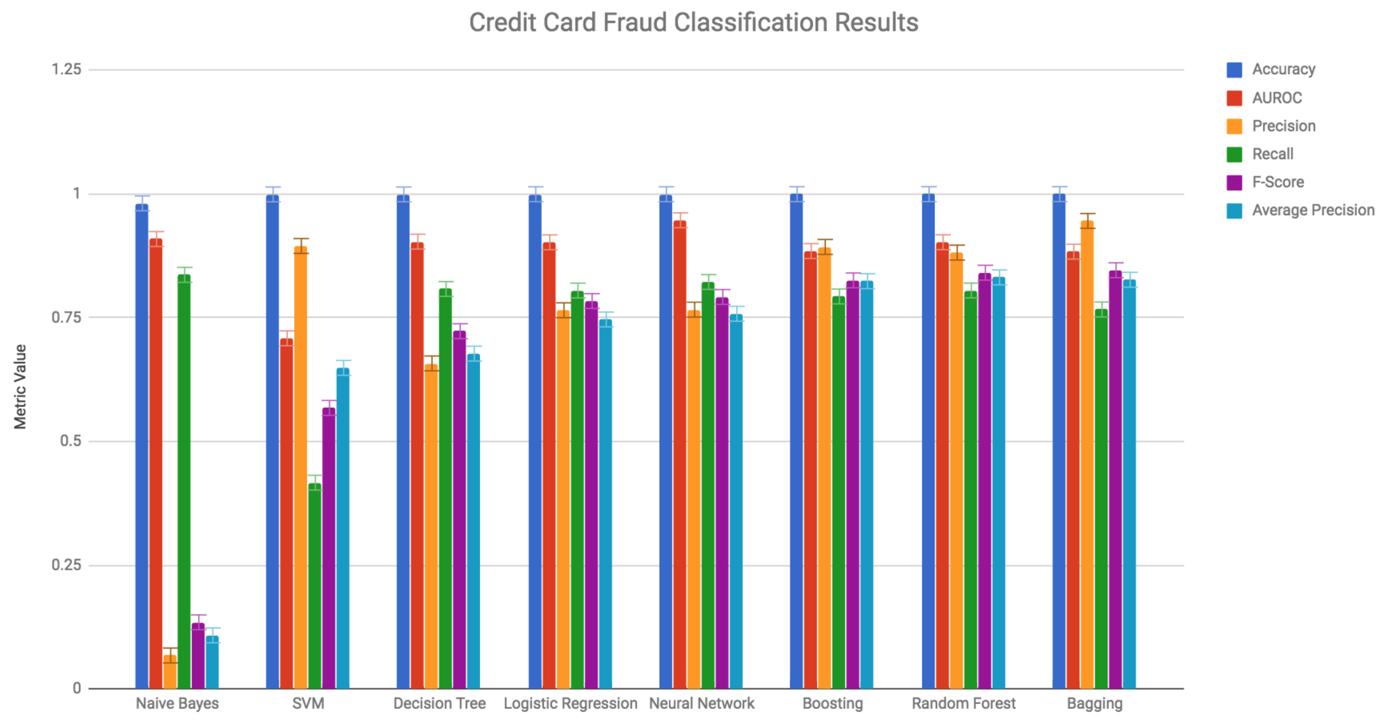


Figure 5: Bar plot showing performance for each classification model using all evaluation metrics.

4.2 Results

Table 3 shows a comparison of several supervised classification models from [2] and our team and Figure 5 shows a visual comparison of our models. For reference, SVM refers to support vector machine. Also, [2] does not compute average precision or training time as a metric, so that can only be used to compare our models and [2] used a different credit card fraud detection dataset, with a different class imbalance, so the comparison between the two groups of models shouldn't be absolute, but rather used as a relative comparison of our models to state-of-the-art performance on supervised fraud detection.

We used naive Bayes as a baseline model to gauge performance on the task. In terms of accuracy and AUROC, the model appears to perform well, with an accuracy of 0.981 and AUROC of 0.933, however, when using metrics more suitable for imbalanced data such as F-score and AP, we see that naive Bayes had a challenging time classifying fraudulent transactions. Since the prior probability on genuine transactions was so high, the classifier most likely guessed that each transaction was genuine, which accounts for the low precision value.

An interesting result is that logistic regression and the deep feed-forward neural network both perform very similarly. The neural network outperforms logistic regression slightly, but the improvements are modest. This is most likely an indication that logistic regression, which can be thought of as a neural network with no hidden layers and a sigmoid output activation, has enough capacity to distinguish genuine and fraudulent transactions. Thus, the 4-layer neural network didn't need the extra layers to do well on the task. Also, the logistic regression model had a significantly shorter training time, taking only 1.88 seconds compared to the neural network, which took 151.42 seconds. Since the results of the logistic regression model and neural network are so similar, taking into account the training time may signify that logistic regression is an overall better model.

When compared to naive Bayes, we can see that logistic regression and the neural network were able to classify fraudulent transactions and didn't guess genuine for all of the training examples. The two reasons for this are that the logistic regression and neural network classifiers were more flexible and could better distinguish the two classes, and we could apply a weighting scheme to these classifiers such that fraudulent training examples were eight times more important to classify correctly. In other words, if the classifier incorrectly labeled a fraudulent transaction (false negative), the loss function was penalized eight times more than incorrectly labeling a genuine transaction (false positive). This is the exact intuition that we had when conducting preliminary research on credit card fraud. It is much more important to classify fraudulent transactions correctly, at the expense of misclassifying genuine transactions, than misclassifying fraudulent transactions (i.e. false negatives are more important than false positives).

The support vector machine performed worse than expected. It had a relatively low F-Score and AP (values of 0.5676 and 0.6484, respectively) compared to logistic regression and the neural network. We hypothesize that this is due to the loss function associated with soft-margin support vector machines.

$$L(\boldsymbol{w}, \boldsymbol{\epsilon}) = \boldsymbol{w}^T \boldsymbol{w} + C \sum_{i=1}^{n} \epsilon_i$$

When C is large, the slack associated with each training example, ϵ_i, must be small, which forces the model to overfit the data, hurting generalization performance on unseen data. Likewise, when C is small, the classifier learns to classify everything as genuine because that will maximize the margin and thus, reduce the loss function the most. As a result of this, we believe that SVMs are inherently poor with imbalanced data, which is why the results are poor compared to other models.

The final models to compare are the decision tree and the ensemble methods. Intuitively, the decision tree will not perform as well as the ensemble methods because the ensemble methods leverage the decision of multiple decision trees as opposed to one. This is also represented in the results, where the decision tree performs slightly worse than the logistic regression model, with an F-Score and AP of 0.7227 and 0.6773, respectively. The ensemble methods ended up performing best out of all of the classifiers. The random forest and bagging models both perform almost identically, both slightly outperforming the boosting model, with an F-Score and AP of 0.84 and 0.83, respectively. We believe the reason for this is that the boosting model had a challenging time classifying some of the fraudulent transactions, forcing the model to heavily weigh those training examples, and as a result, the model overfit the training data, lowering performance on the test set. The random forest and bagging models don't weigh training examples, so they aren't as susceptible to overfitting. The ensemble methods, in general, perform better than other classifiers because of the ability to re-sample the data for each individual classifier. The ensembled classifiers will each have a different representation of fraudulent and genuine transactions, where some of the classifiers will have a larger sample of fraudulent than the original dataset. This helps to re-balance the classes, without heavily undersampling or oversampling the dataset, and without changing the loss function to re-weight the penalty of fraudulent transactions. When comparing just the random forest and bagging model, the random forest will most likely be preferred, since the training time is 5.32 seconds compared to bagging, which has a training time of 30.8 seconds. Thus, after designing, implementing, and analyzing these classifiers, it was determined that the random forest model outperformed all other models.

Finally, we can also compare our models to those implemented in [2]. Since the datasets are different, it may not be correct to do an absolute comparison of the two, but we can see that our models have very similar metric values, if not better, compared to [2]. This is an indication that our models compare to state-of-the-art research on credit card fraud detection.

5 DISCUSSION

After finishing the project, we believe that our results have been very positive. We have successfully applied machine learning and data mining techniques to a real-world problem, credit card fraud detection, and our results compare to those in previous state-of-the-art research. We carried out a full data mining pipeline from initial research, data analysis, design & implementation of machine learning models, and evaluation. We found that the models we explored worked very well. We didn't run across many problems with the design and implementation of those models. Also, we were fortunate in that the data was already preprocessed and the dimensionality had already been reduced, which made it much easier to dive right into the initial analysis and model implementation.

We initially had plans to explore unsupervised models, such as clustering and anomaly detection, as well as a supervised recurrent neural networks for sequential learning. However, we mistakenly misunderstood our data in that the dataset is one long sequence. To train a recurrent neural network, each training example needs to be its own independent sequence. Because of this, we were unable to utilize a recurrent neural network for sequential learning. Due to time constraints, we also did not explore unsupervised methods for detecting credit card fraud. However, based on the t-SNE visualization results, we hypothesize that the unsupervised methods wouldn't have been nearly as successful as the supervised methods.

Given more time, we would have liked to implement k-fold cross validation with our supervised models. We used a single held-out validation dataset for hyperparameter tuning, but since the data was imbalanced, it may have been better to use multiple validation sets, which would ensure that almost all of the fraudulent transactions were included in the training data. Furthermore, we didn't explore the possibility of undersampling the genuine transactions or oversampling the fraudulent transactions. It was determined that weighing the classes in the algorithms themselves was successful.

One limitation of this work is that a comparison between our approach, undersampling, and oversampling was not conducted. For others that seek to extend our work, possible future work would be:

- Explore unsupervised methods (clustering and anomaly detection)
- Compare our results with classifiers that undersample/oversample data
- Tune classifiers with k-fold cross validation instead of a single validation set

Since this project is so related to everyday people, it is interesting to formulate the problem as if we were working for a bank. That is, if we were performing this analysis on behalf of a bank, we would be exploring metrics much more general than average precision, F-score, etc. One metric that is much more important is revenue loss. When a fraudulent transaction occurs, the bank has to refund the customer with the amount of the transaction. Thus, this is a direct loss to the bank's revenue. As a result, banks are most likely much more concerned with classifying fraudulent transactions correctly. Also, by incorporating a revenue-loss-per-fraud-transaction value, such as $25 per transaction, for example, it would allow the bank to quantify the number of false positives and false negatives that are acceptable. Since we don't have this data, it is much harder to perform this analysis, but this is a further direction the project could go, if we were paired with a bank and had access to their data. This projects the task into a real-world domain, where not only is the classifier's performance important, but also how the performance is optimized for the real-world and the associated measures of performance such as revenue loss.

6 CONCLUSION

With the increase of online transactions, the opportunities for massive credit card fraud also increase. As techniques for stealing credit cards shift towards quick, massive transactions, the cost burden rises for card companies and consumers. Being able to detect fraud early on, and rapidly adapting to new fraudulent methods, can save millions of dollars per year. As such, we have explored automatic credit card fraud detection using supervised classification. We've implemented several supervised classification models including naive Bayes, logistic regression, a neural network, a decision tree, a random forest, bagging, boosting, and a support vector machine. Due to the nature of credit card fraud, most datasets, including ours from Section 3.1, are imbalanced, where fraudulent transactions are under-represented compared to genuine transactions. Due to this, it was determined that average precision and F-score were the best metrics for analyzing classification performance. As seen in Table 3, the random forest and bagging models performed best with F-score and average precision values at approximately 0.84 and 0.83, respectively. We hope that our results from applying recent, sophisticated supervised classification techniques helps to propagate the field of credit card fraud detection forward, and also highlights ways to improve current techniques used in industry.

REFERENCES

[1] Andrea. [n. d.]. Credit Card Fraud Detection. ([n. d.]). https://www.kaggle.com/dalpozz/creditcardfraud/kernels

[2] Siddhartha Bhattacharyya, Sanjeev Jha, Kurian Tharakunnel, and J. Christopher Westland. 2011. Data Mining for Credit Card Fraud: A Comparative Study. *Decis. Support Syst.* 50, 3 (Feb. 2011), 602–613. https://doi.org/10.1016/j.dss.2010.08.008

[3] Open Source Community. [n. d.]. scikit-learn. ([n. d.]). http://scikit-learn.org/stable/

[4] Google. [n. d.]. Tensorflow. ([n. d.]). https://www.tensorflow.org/

[5] Sam Maes, Karl Tuyls, Bram Vanschoenwinkel, and Bernard Manderick. 1993. Credit Card Fraud Detection Using Bayesian and Neural Networks. In *In: Maciunas RJ, editor. Interactive image-guided neurosurgery. American Association Neurological Surgeons.* 261–270.

[6] NUMFocus. [n. d.]. Matplotlib. ([n. d.]). https://matplotlib.org/

[7] NUMFocus. [n. d.]. Numpy. ([n. d.]). http://www.numpy.org/

[8] NUMFocus. [n. d.]. Pandas. ([n. d.]). https://pandas.pydata.org/

[9] Michael Waskom. [n. d.]. Seaborn. ([n. d.]). https://seaborn.pydata.org/

7 APPENDIX

7.1 Honor Code Pledge

On my honor, as a University of Colorado Boulder student, I have neither given nor received unauthorized assistance.

7.2 Group Member Contributions

Tyler Scott

- Helped implement naive Bayes, logistic regression, neural network, random forest, and support vector machine
- Tuned hyperparameters of all supervised models using validation data
- Researched and implemented computation of the evaluation metrics
- Wrote code to plot the class distributions for the initial analysis
- Helped implement t-SNE visualization on the dataset
- Wrote significant portions of the abstract, introduction, related work, methodology, evaluation, discussion, and conclusion of the report
- Helped create slides for the presentations

Nicholas Clement

- Helped implement naive Bayes, logistic regression, decision tree, bagging, and boosting
- Implemented t-SNE visualization on the dataset

- Wrote code to plot ROC curves and precision-recall curves for the classifiers
- Wrote portions of the methodology and evaluation of the report
- Helped create slides for the presentations

Christopher Struckman

- Helped implement naive Bayes and the support vector machine
- Wrote code for the initial analysis to check orthogonality of the dataset
- Wrote portions of the methodology and evaluation sections of the report
- Helped create slides for the presentations

Christina Nguyen

- Wrote code for the initial analysis to check correlation between attributes of the dataset
- Analyzed past research papers on credit card fraud
- Wrote significant portions of the introduction, related work, and initial analysis