Credit Card Fraud Detection

By: Jacob Avchen, Sal Russo, & Jacob Hollander

Project Proposal

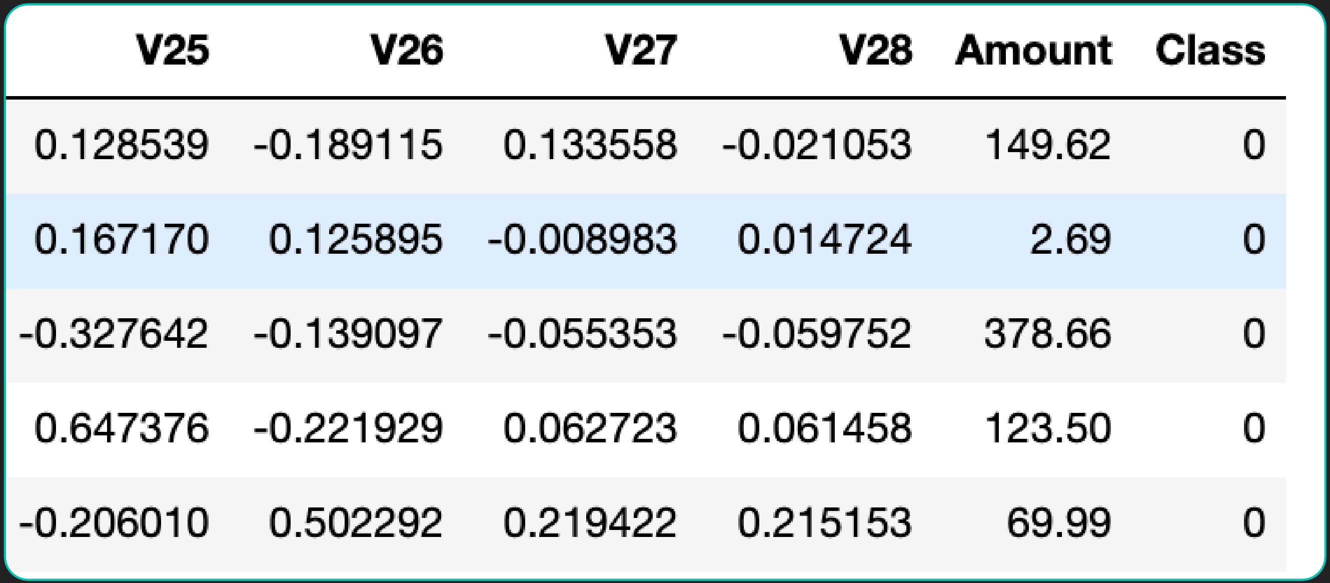
- Utilize multiple machine learning modeling techniques to identify whether a credit card purchase is fraudulent or not fraudulent

Tools

- Pandas for data acquisition & data cleaning
- Matplotlib & Seaborn for data visualization
- SciKit learn for machine learning modeling & scaling
- Keras for Neural Network

Our Dataset

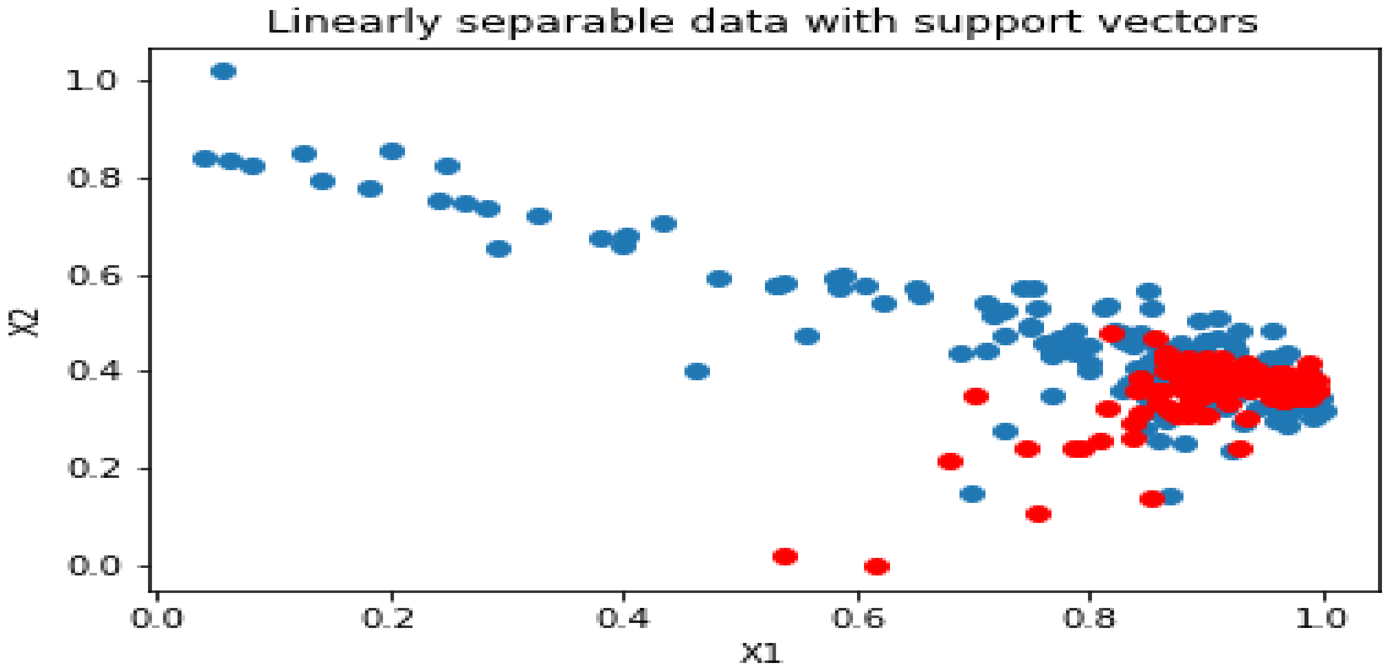
- Data is extracted from a csv file provided by Kaggle
- Roughly 280,000 transactions
- Cut into snippets of equal cases of fraud & not fraud transactions
- V1-V28 were produced through an obfuscation process. For security reasons, we cannot backtrack these numbers to any values that would make sense to us, so our model will only be specific to the non-sensitive data here

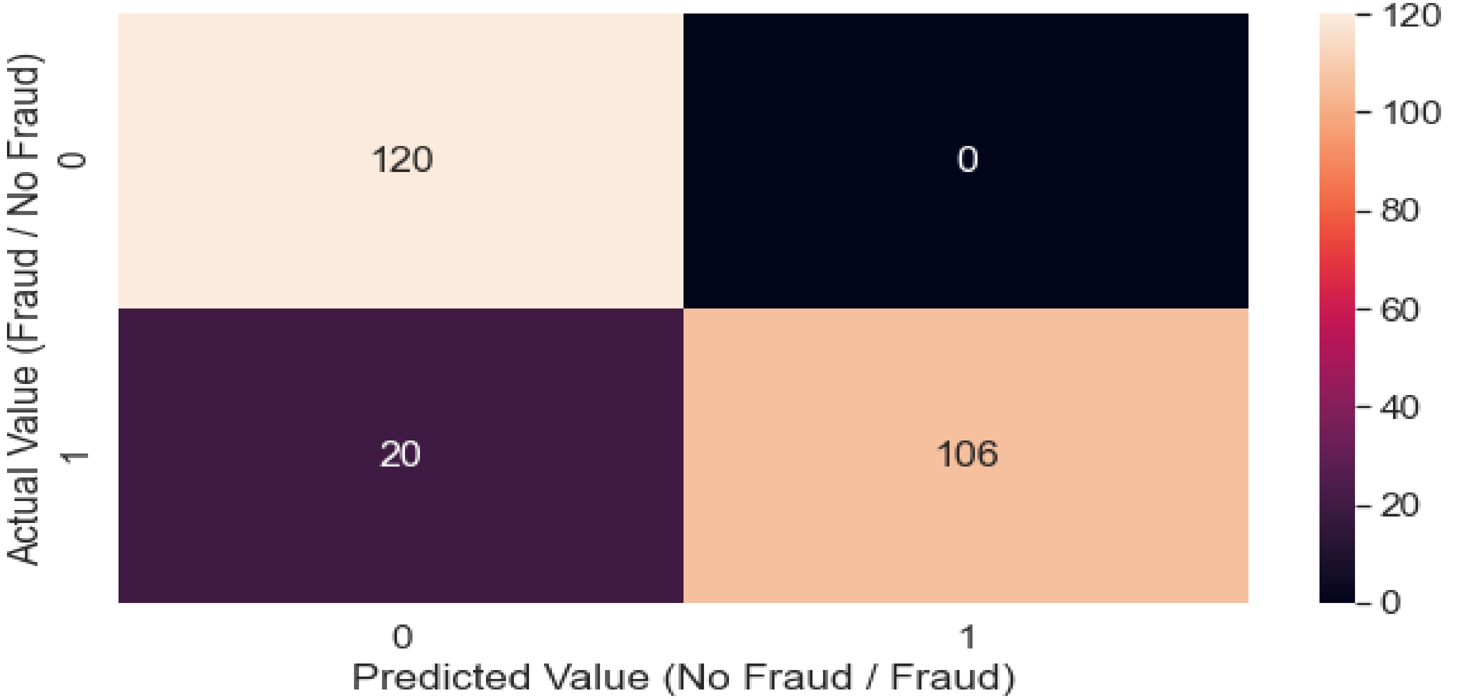


V25	V26	V27	V28	Amount	Class
0.128539	-0.189115	0.133558	-0.021053	149.62	0
0.167170	0.125895	-0.008983	0.014724	2.69	0
-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
0.647376	-0.221929	0.062723	0.061458	123.50	0
-0.206010	0.502292	0.219422	0.215153	69.99	0

SVM

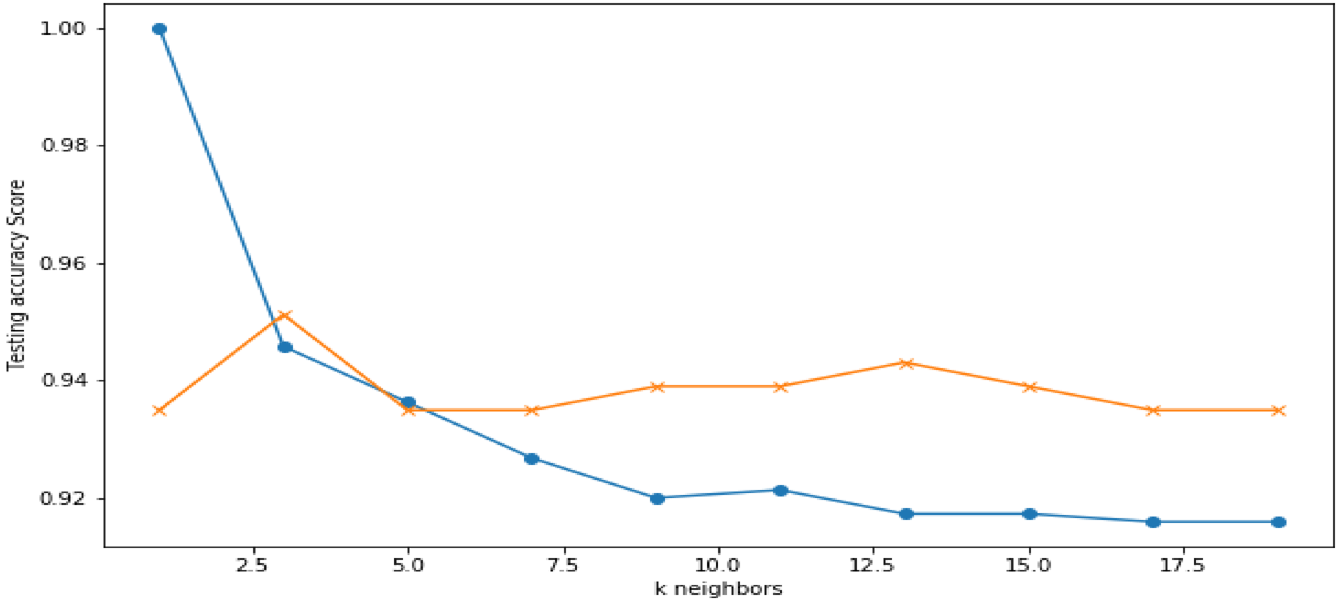
- Tested a 75/25 train/test split of balanced fraud to not fraud transactions
- Plotted X1 & X2 (2 features)
- Achieved a .919 Test Accuracy with the SVM Model
- All of our errors were "true negative" which means the model failed to catch 20 cases of fraud

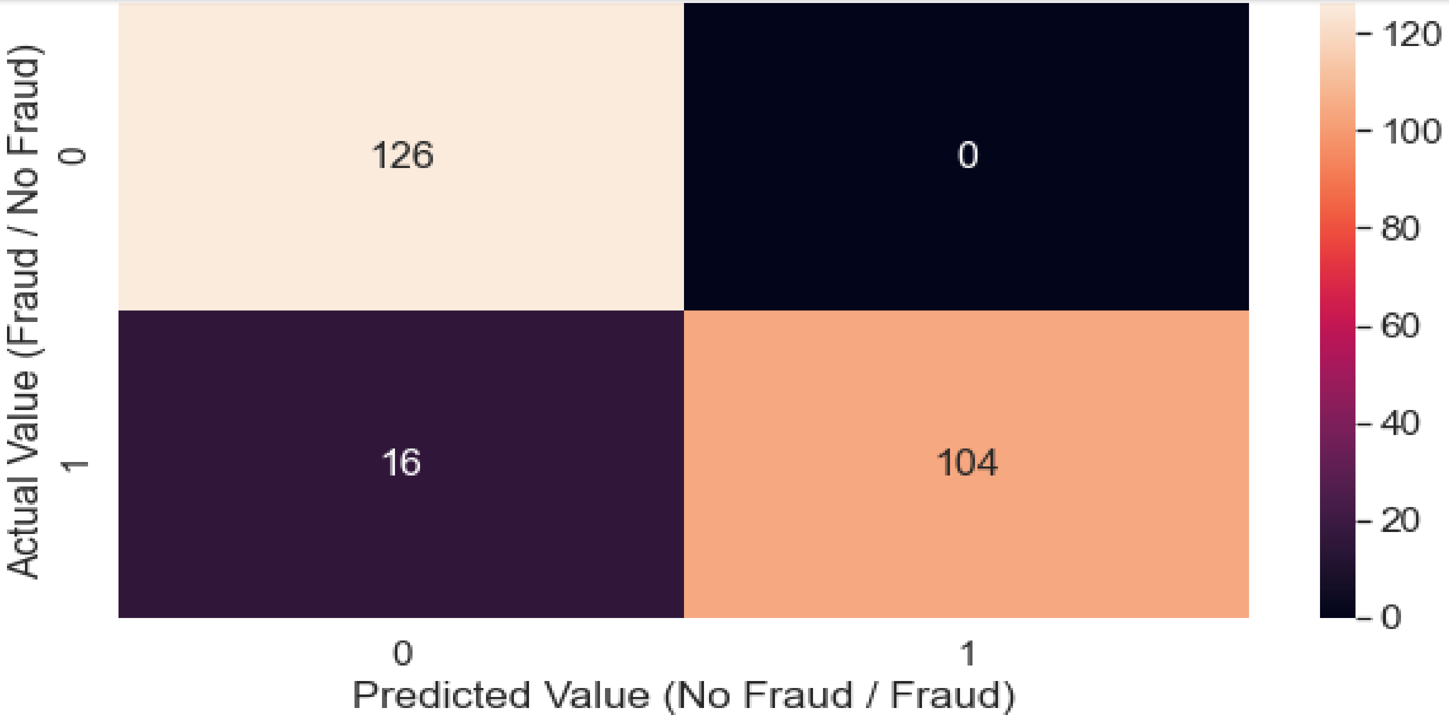




K Nearest Neighbors

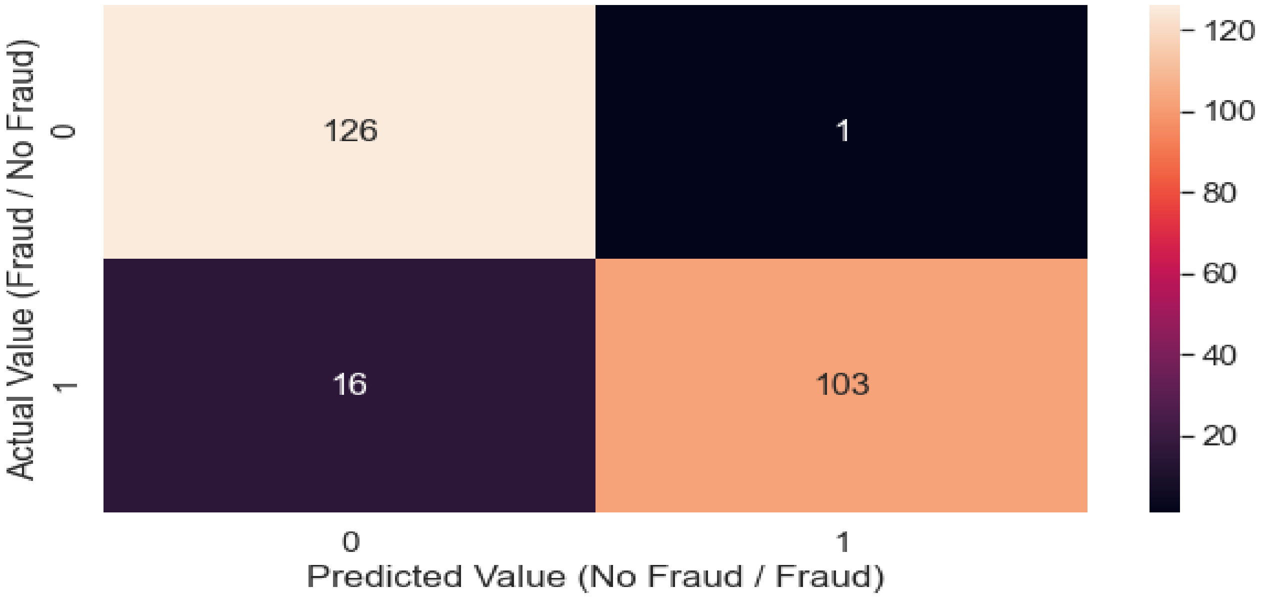
- Tested a 75/25 train/test split of balanced fraud to not fraud transactions
- Plotted K1-K19
- Closest accuracy was using K=3 at 0.944/0.947
- Highest test accuracy was at K=5 with an accuracy of 0.951

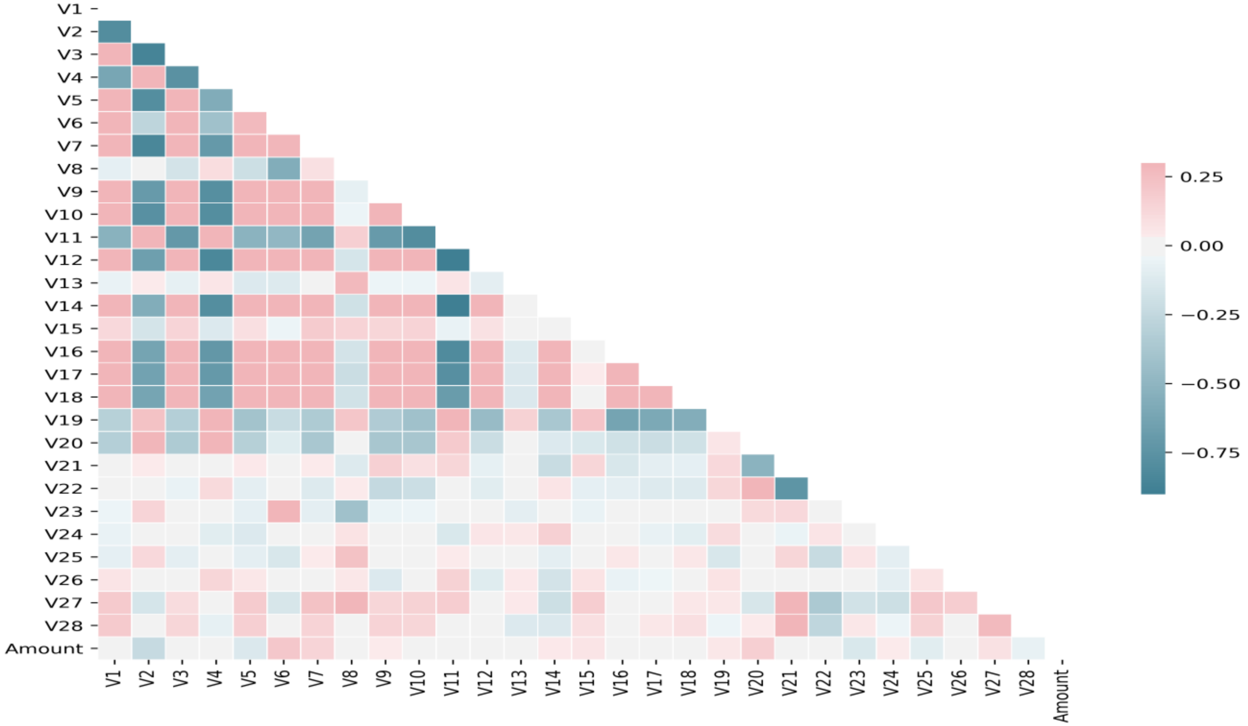




Logistic Regression

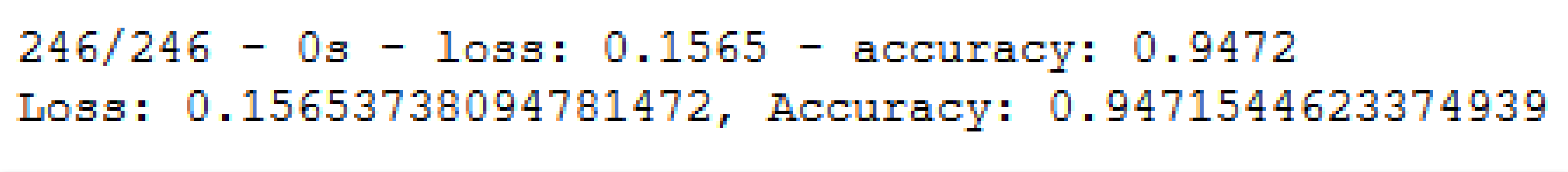
- Used a balanced snippet of our dataset
- Confusion Matrix shows 8 false positives
- Achieved an accuracy of 0.934 with a standard deviation of 0.012
- Heat map shows strong correlations between features 1 thru 20 but drops off after



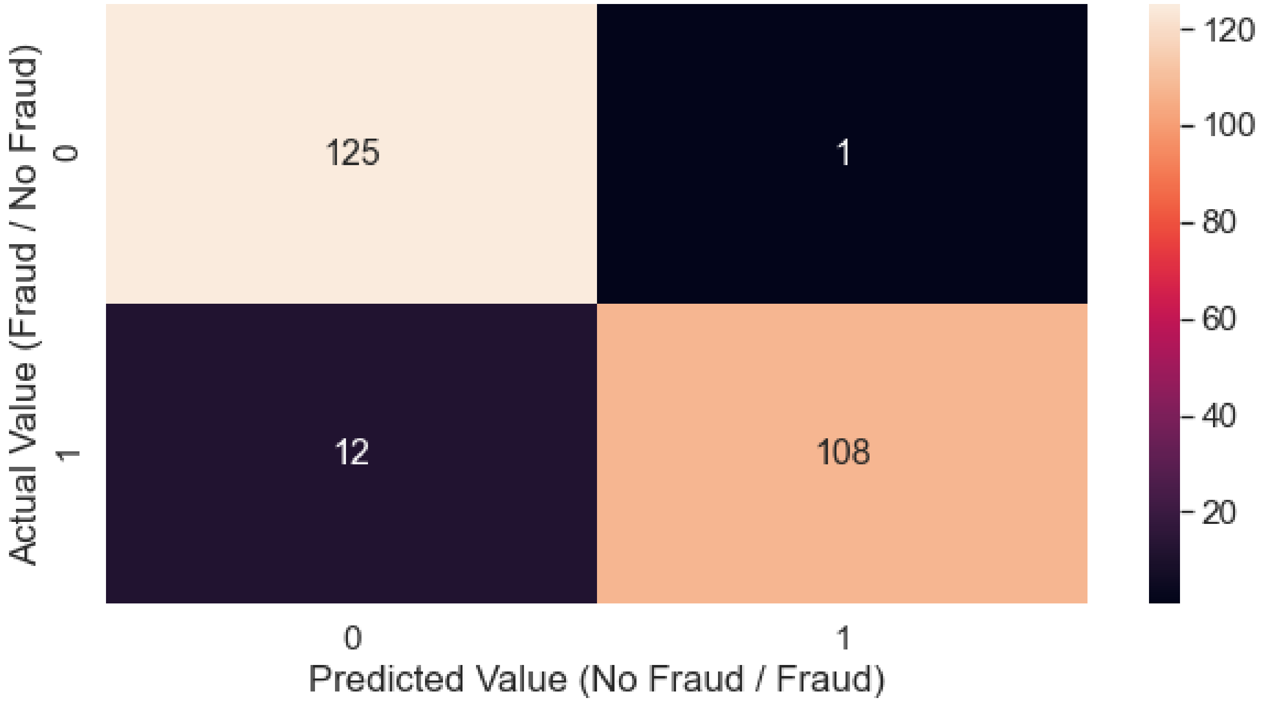


Neural Network

- Our first Neural Model has a single hidden layer
- We used all 28 V values in our data and the Amount column
- The model's predictive accuracy rounded out to 94.7%
- Compared to the actual values, the model returned 3 False Positive values and missed 10 cases of fraud out of 246 points of data

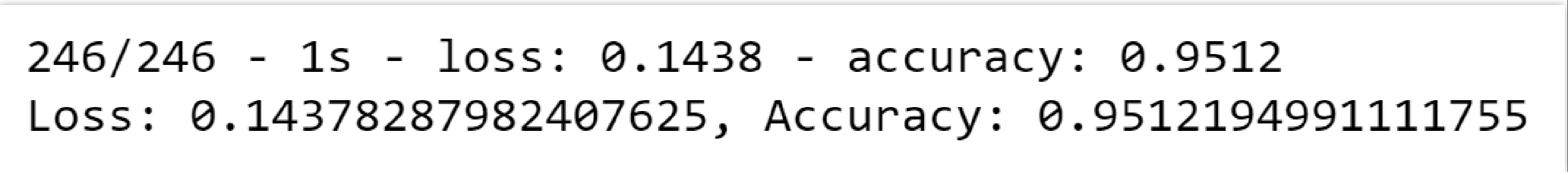


```
246/246 - 0s - loss: 0.1565 - accuracy: 0.9472
Loss: 0.15653738094781472, Accuracy: 0.9471544623374939
```

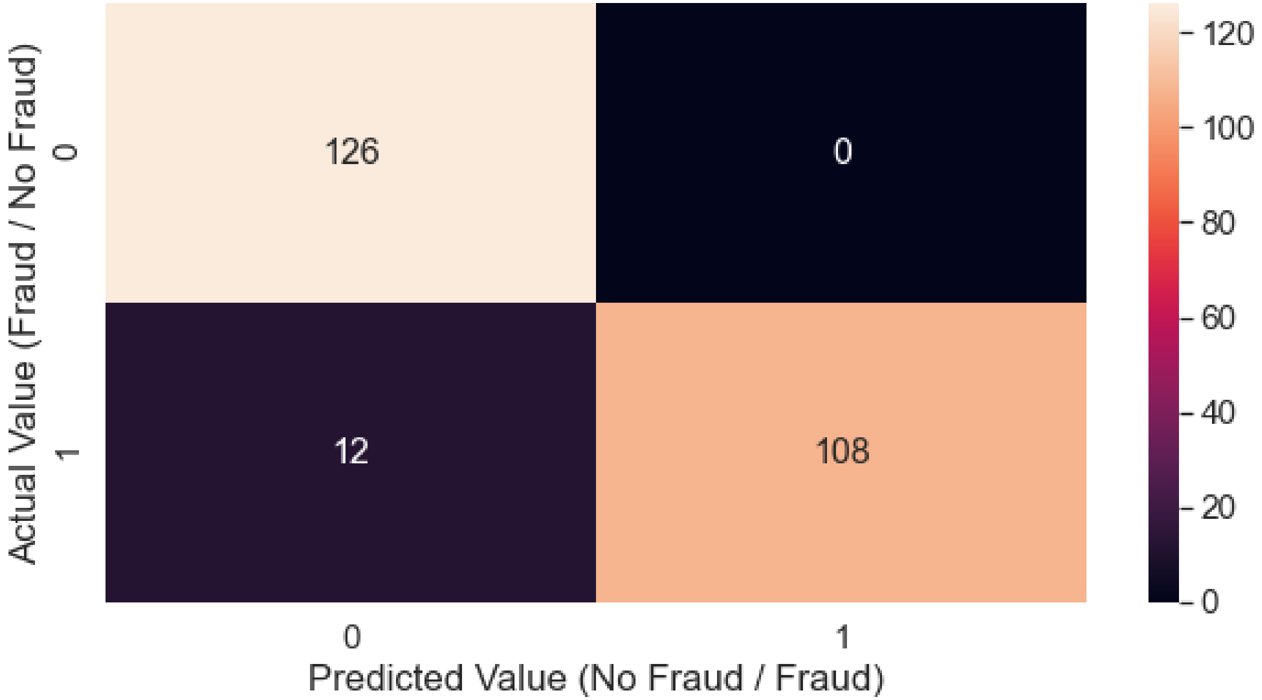


Deep Neural Network

- Our second Neural Model has 2 hidden layers
- We used the same distribution of data for both models
- The Deep Model's predictive accuracy rounded out to 95.1%
- Compared to the actual values, the Deep Model did not return any False Positives and missed 12 cases of fraud
- Despite higher accuracy than the first model, this model missed more cases of actual fraud



```
246/246 - 1s - loss: 0.1438 - accuracy: 0.9512
Loss: 0.14378287982407625, Accuracy: 0.9512194991111755
```



Comparison of Model Accuracies

	SVM	K Nearest Neighbors	Logistic Regression	Neural Network	Deep Neural Network
Accuracy	0.919	0.951	0.934	0.947	0.951

Improvements/Limitations

Improvements

- Create a user interactive webpage to demonstrate our model
- Better train our models to prioritize errors being false positives rather than true negatives
- More effectively deal with an unbalanced dataset

Limitations

- Our features (V’s) came from sensitive information and therefore could not be backtracked to any values that would make sense to us

Thank You!

Questions?