Summary of Methods

I aimed to develop a dashboard that would facilitate modeling an imbalance classification task using credit card fraud data sourced from Kaggle.

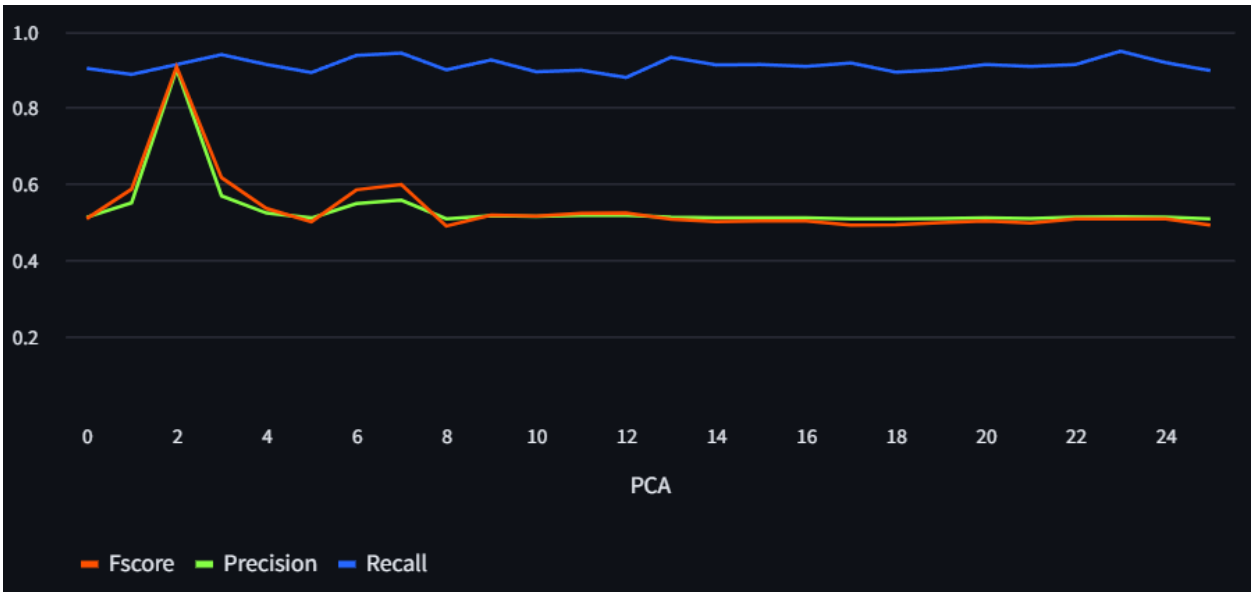
The imbalanced nature of the dataset also had to be accounted for. I tested multiple cost sensitive learning methods such as resampling, balancing algorithms, stratified splitting, and using different evaluation metrics. This development was done in a Jupyter notebook.

The finished dashboard used user-inputted hyperparameters to customize the finished model, which was then saved locally to log and analyze past models. I constructed a pipeline that performed a stratified train test split, followed by resampling of the user's choice, scaling, the number of principal components to be used, and lastly fitting to the algorithm of choice. Certain steps could be omitted or changed based on user input.

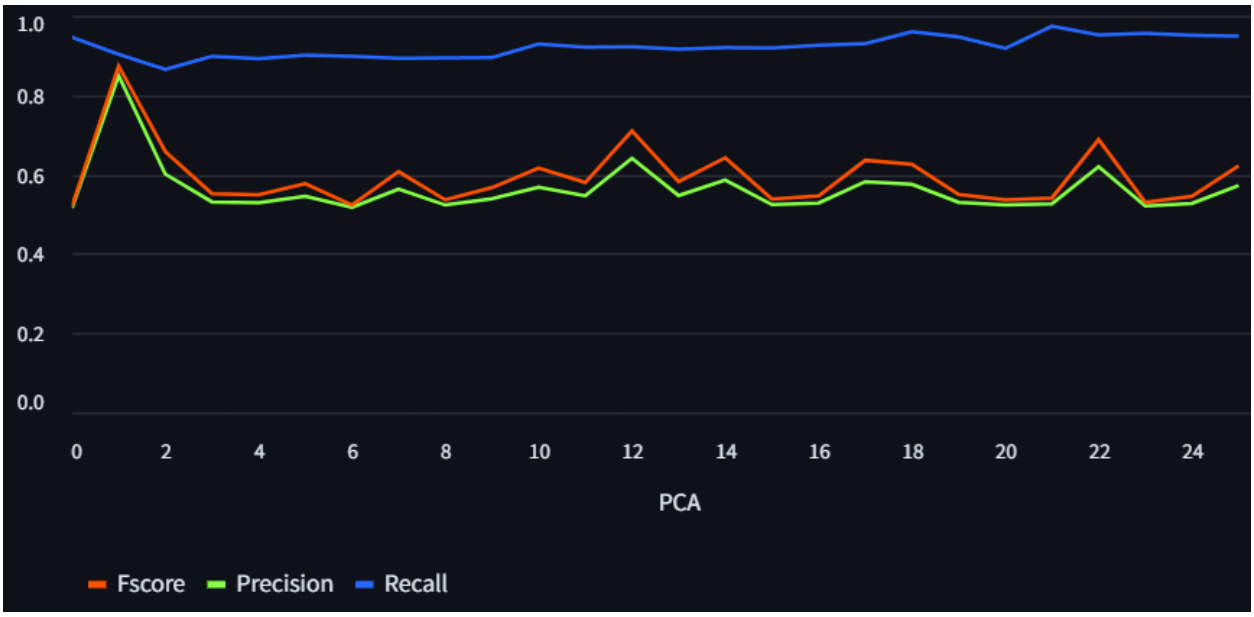
The undersampling and oversampling algorithms used were imblearn's NearMiss and Synthetic Minority Oversampling Technique (SMOTE), respectively.

Summary of Results

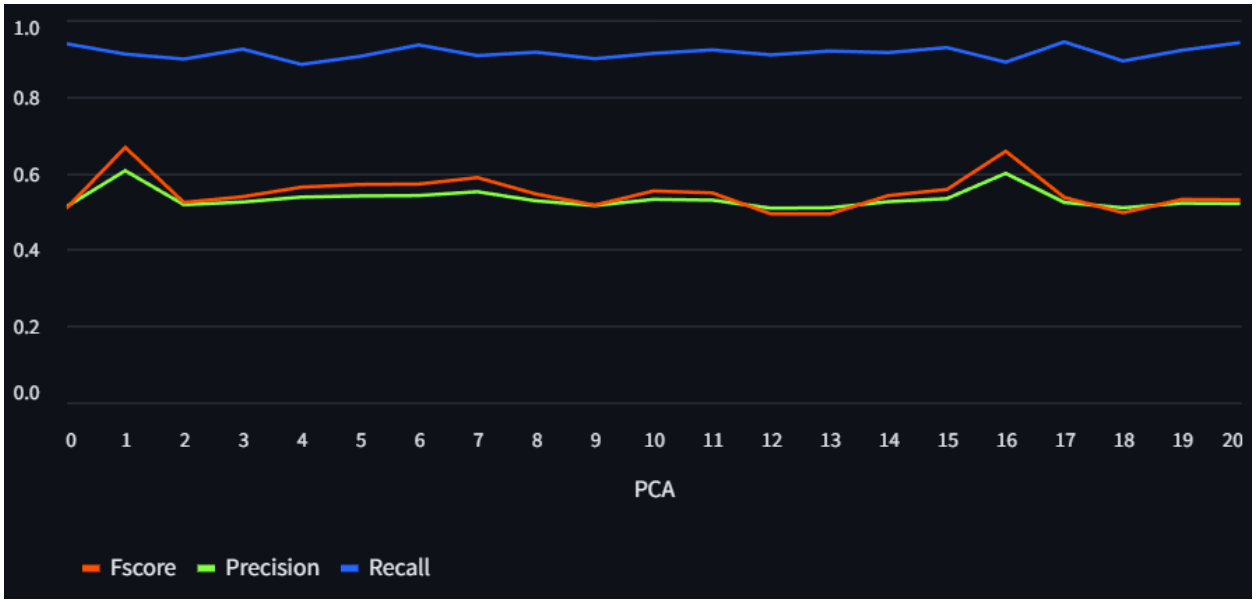
Models using logistic regression and undersampling:



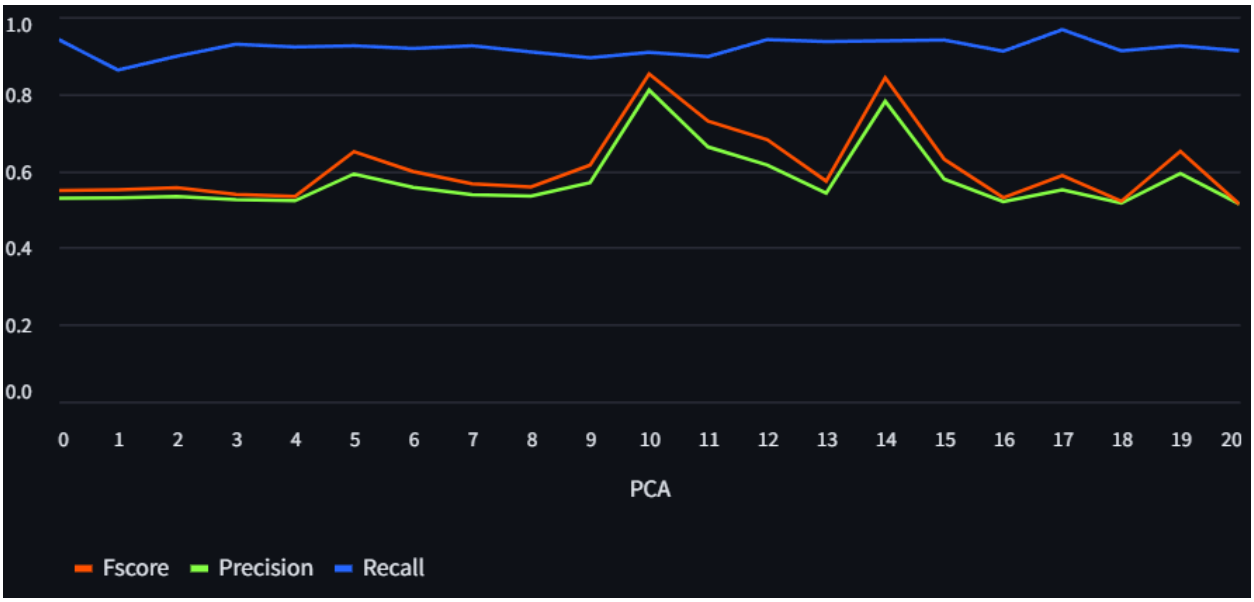
Models using logistic regression and oversampling:



Models using random forest classification and undersampling:



Models using random forest classification and oversampling:



In general, both logistic regression and random forest classification yielded consistently high scores for recall, but also generally suffered in precision. This indicates that models perform well in identifying fraud but are also eager to identify benign cases as fraud as well, leading to an influx of false positives.

This is not the worst case scenario where the model would have low recall score, which would indicate that a lot of fraudulent transactions aren't being properly identified as fraudulent. This would be far more disastrous for the credit card company.

However, having low precision is still undesirable from a workplace perspective; it could mean a lot of man-hours put into servicing customer complaints and double checking on transactions. Ideally, we want both precision and recall to be high.

The dashboard allowed me to test models using different combinations of algorithms, resampling methods, and numbers of principal components, as well as visualize them.

Notably, logistic regression performs very well with 1 to 2 principle components, yielding around 0.90 for precision, recall, and fscore. Using SMOTE tended to yield more spikes in performance. On the other hand, random forest classification performed poorly all around in precision score when using undersampling, but had spikes in performance when using oversampling at around 10 and 14 principle components, where scores were around 0.85.

Next Steps

Going forward, it will be worthwhile to implement support for other algorithms into the dashboard. Furthermore, I will perform additional test runs using the same parameters as described above to see if i can yield similar results.