Travis Mark

COMP 4334

Assignment 2

Abstract

Credit card fraud poses a significant challenge for retailers and banks, necessitating effective detection and prevention measures. In this assignment, I focused on developing a cost-effective, reliable, and scalable solution for real-time credit card fraud detection, crucial for minimizing financial losses. Utilizing credit card transaction data, including amount, merchant details, location, and timestamp, my objective is to identify potential fraudulent transactions and prevent their approval, saving both the customer and the company money. Given the rarity of fraudulent instances compared to legitimate transactions, accurate identification poses a significant challenge. In the real world, transactions as presented to a bank in a constant stream, with sometimes hundreds or thousands of transactions occurring nearly simultaneously depending upon the size of the institution.

Using a dataset obtained from Kaggle, I implemented and evaluated both random forest and extreme gradient boosted trees supervised machine learning models. After selecting the best model, I mimicked a streaming operation to simulate real-time fraud detection.

# Data

The data I used in the model was obtained from Kaggle (https://www.kaggle.com/datasets/kartik2112/fraud-detection/data?select=fraudTrain.csv). Due to the confidential nature of financial data, obtaining useable data which is both understandable and informative is challenging. While there are some other data set options available on Kaggle, those have either masked data or have had principal components analysis performed on them ahead of time, making the explainability and interpretation of the features in the model impossible. The reason I chose this data set in particular, is because it has interpretable fields, like merchant name, transaction amount, location, date of birth. In order to meet privacy laws, the trade off for having these fields is that this data is completely simulated. However, I felt that in practice of building and explaining the model and the applicability of translating this example to real world data would be more beneficial with interpretable features.

The simulated data contained transactions over the course of 2 years and had over 1.8 million records in total. The fields contained in this data set are: unique id for each row, transaction date time, unix time, transaction number, credit card number, merchant name, merchant category, merchant latitude and longitude, and transaction amount. The following data points for the cardholder are included: first and last name, gender, street address, city, state, and zip code, as well as latitude and longitude, city population, occupation, and date of birth. For each record the target fraud indicator label is provided.

One of the challenges of this use case is the rarity with which fraudulent transactions occur. In this data set, of the 1.8 million transactions, roughly 0.5% of the transactions are true fraud.

# Preparation

The original data is already split into training and test sets. In Spark, I combined the data back together and then performed my own split on the data. After splitting, I ensured that the instances of fraud in both the training and test sets were evenly distributed. For the building of my models, in attempting to keep as true to real world as I could, I dropped the city population field from the data. In my experience, this information would not be present in transaction information and with the goal of making this a real-time fraud detection solution, attempting to add in the city population from an external source could present issues with slowing down the processing when categorizing potentially thousands of transactions per second. Additionally, I also removed the unique id, unix time fields, and street address fields. The unique id is a similar field to the transaction number, which would be a unique identifier itself, and the unix time did not provide additional information above the transaction date time which is also present in the data and more interpretable by humans. I dropped these fields because, again keeping them in the data set and passing this extraneous data to and through the pipeline and model would only present issues with movement of data and scalability. Lastly, I performed some cleaning on the merchant names. Each merchant name had a prefix of “fraud\_”. I removed this prefix in the data to make it more realistic.

I also engineered additional fields for use in the models. From the transaction date time field I created a transaction hour and transaction minute fields. The purpose of this was to attempt to identify if fraudulent transactions are more likely to happen at certain times of day, fore instance in the middle of the night when cardholders may be unaware. I engineered an age feature from the date of birth and transaction date time fields. While an individual’s birth day may not be a relevant feature for the model, an individual’s age could prove useful in identifying pockets of cardholder’s who may be more susceptible to fraud.

For the age and transaction minute fields I used the bucketizer method in Spark to create bins of data for these features to use in the model training. For all the categorical features, I used the string indexer function to create categorical indexes to be used in the model. I assembled the features into a vectorizer and created stages in a pipeline to run each each and train the modes.

# Model Training

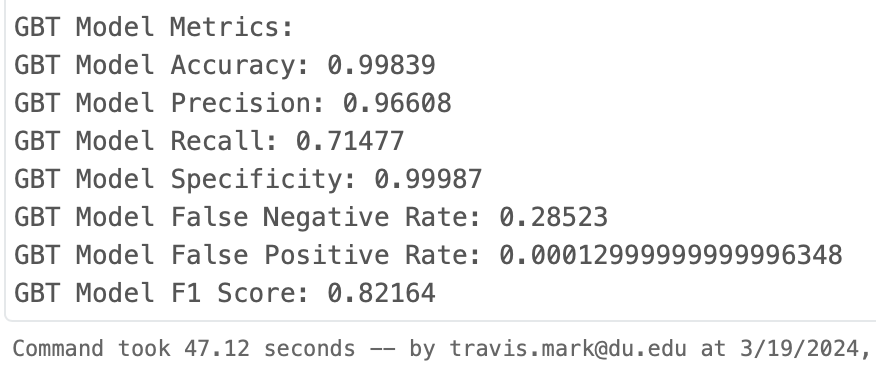
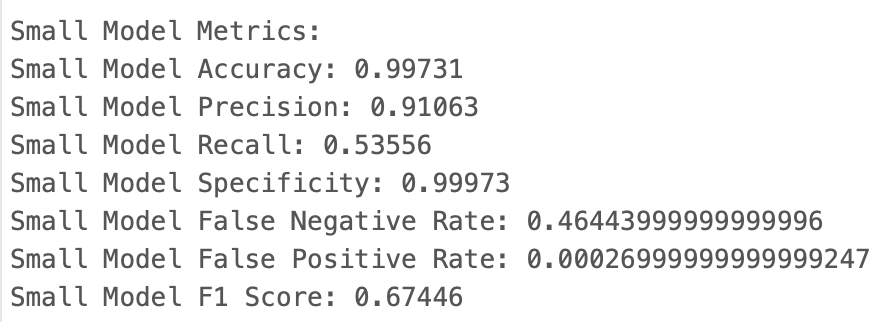
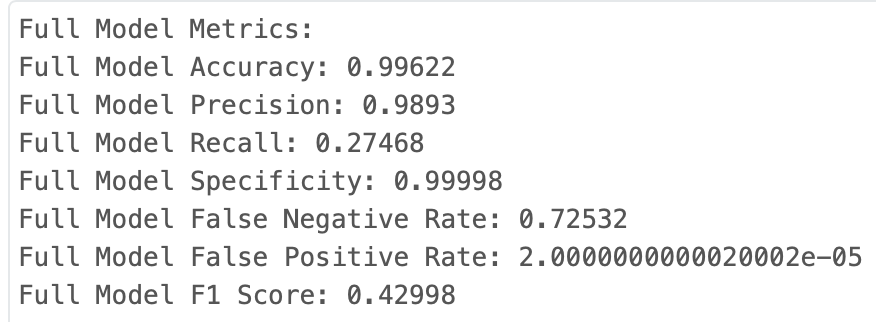
Initially, I trained 3 models, a full random forest model using all the available feature set and training a forest with 51 trees. Given the larger number of features in the model, I limited the maximum feature subset size able to be randomly drawn at each cut point to be the square root of the number of features. Second, I trained a smaller random forest model with the same number of trees, but limited to transaction amount, age, gender, transaction hour, merchant name, and merchant category. In this model, with fewer features, I allowed for all features to potentially be selected at each cut point. Lastly, I used the smaller feature set to train an extreme gradient boosted trees model. This method is a more complex version of random forest. Initially a weak tree is trained on the first iteration and then XGBoost builds each tree sequentially, with each new tree trying to correct for the errors made by the previous. At each iteration the model uses a method called gradient decent, which analyzes where the most error is coming from in the previous tree and aims to minimize its loss function in terms of the sum of the squared residuals. This method is iterative and allows the XGBoost model to improve upon the mistakes of prior trees, reducing the overall error.

# Model Selection

After training each model, I ran the training data back through the model to create predictions and validate against the true outcomes. For metrics, the accuracy of these models is not particularly helpful for evaluation because of the overwhelming number of non-fraud instances. By properly identifying nearly all of these non-fraud, it is possible to achieve a very high accuracy. In fact, by simply predicting all instances as non-fraud, would produce an accuracy of nearly 99.5%. Precision and recall are more informative in the model performance. Since F1 score is the harmonic mean of the precision and recall, it is a good metric for evaluating the models. Additionally, understanding the context of this problem, the objective to identify all (or as many as possible) fraudulent transactions, we want to limit the false negative rate. False negative occurs when the model predicts no fraud on a transaction which is truly fraud. This is a model failure which impacts the cardholder and company monetarily. We also want to limit the false positive rate, when the model predicts a fraudulent transaction that is truly legitimate. While it is important to limit these, a higher tolerance of false positives would be acceptable by consumers than false negative. False positives result in potentially impact consumers being contacted unnecessarily or potentially having their legitimate transaction blocked. This causes inconvenience for the consumer, but is understandable in an effort to protect them from fraudulent activity.

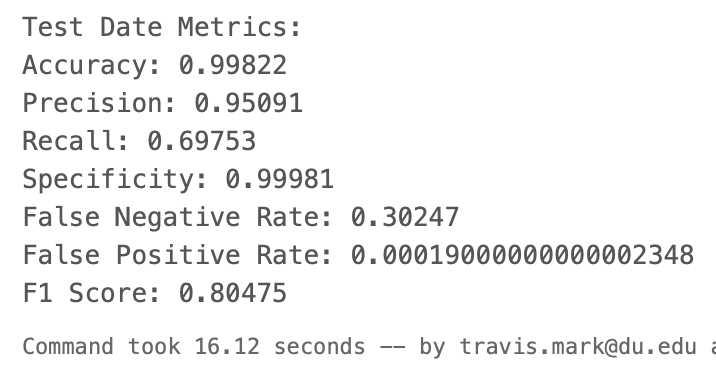
Validating the models using F1 score and false negative rate, the large random forest model performed worst, with a F1 score of 0.43. The full model also has a false negative rate 0.73. This is not necessarily surprising given the number of categorical features and high cardinality of those which are perhaps less deterministic in identifying fraud.

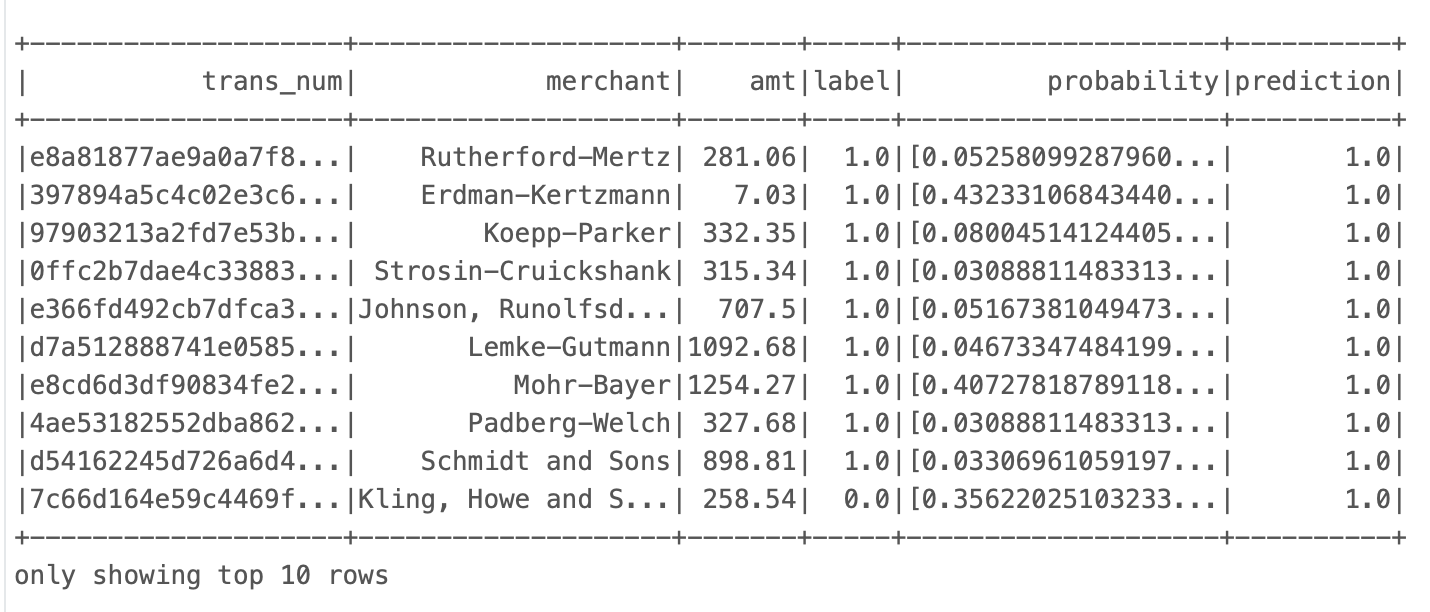
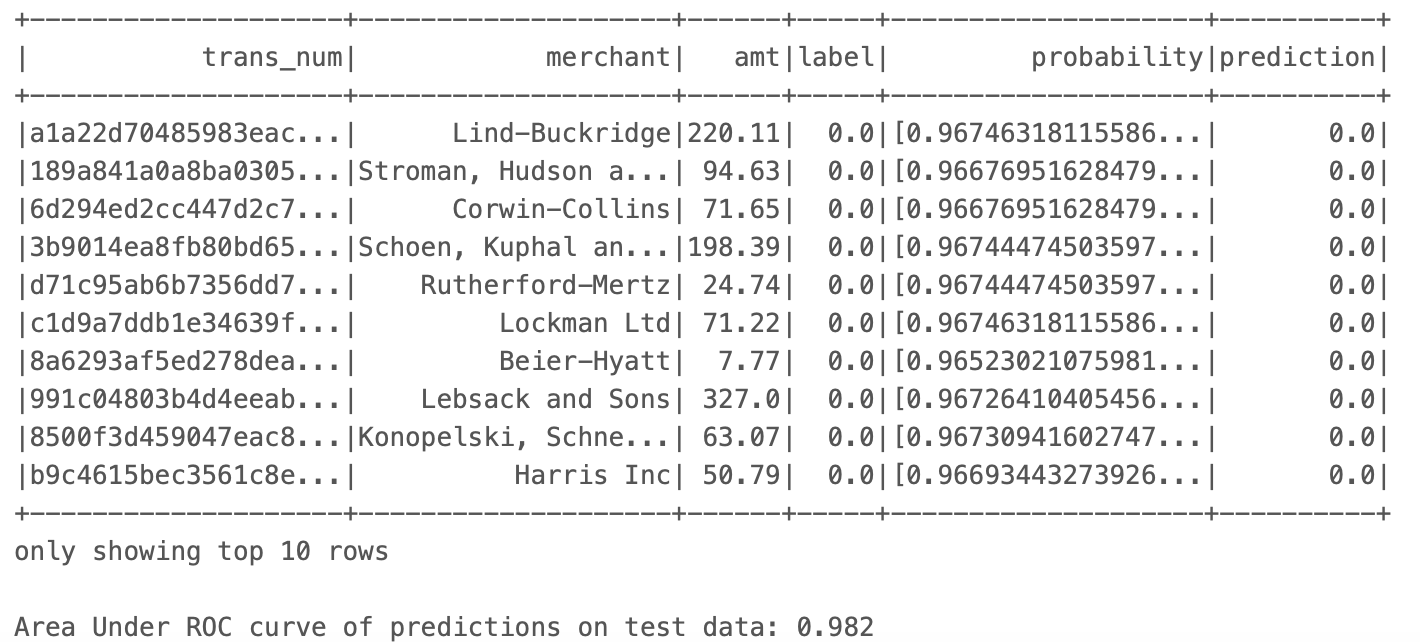
This is why I limited the smaller models down to a select few features which I believed may be most informative. The small random forest performed much better in validation. It achieved an F1 score of 0.67 and lowered the false negative rate to 0.46. Finally, the XGBoosted trees model performed the best of the three, achieving an F1 score of 0.82, while limiting the false negative rate to 0.29. Both significant improvements over the small random forest model.

I also looked at the area under the ROC curve to compare the models. This metric looks at not only the predicted outcome of the model but is the performance of predicted probabilities. In F1 or false negative rate, each instance is treated equally, but in the area under ROC the model is punished for incorrectly predicting the outcome proportional to the confidence of the prediction. Meaning a borderline prediction near the threshold of 50% being incorrect is not as impactful to the score as being incorrect on an instance where the outcome was predicted at 90%. Although this does reward the model for properly selecting non-fraudulent transactions with a high probability. The score is between 0 and 1, where .5 is essentially a random chance. The models showed the same trend in the area under ROC. The full model scored 0.9568, the small random forest scored 0.9768, and the XGBoost scored 0.9908. The value of 0.9908 is pretty performant overall.

Based on the validation of the models on the training data, I selected the XGBoost model as my final model to use on the test data and streaming data.

# Model Evaluation

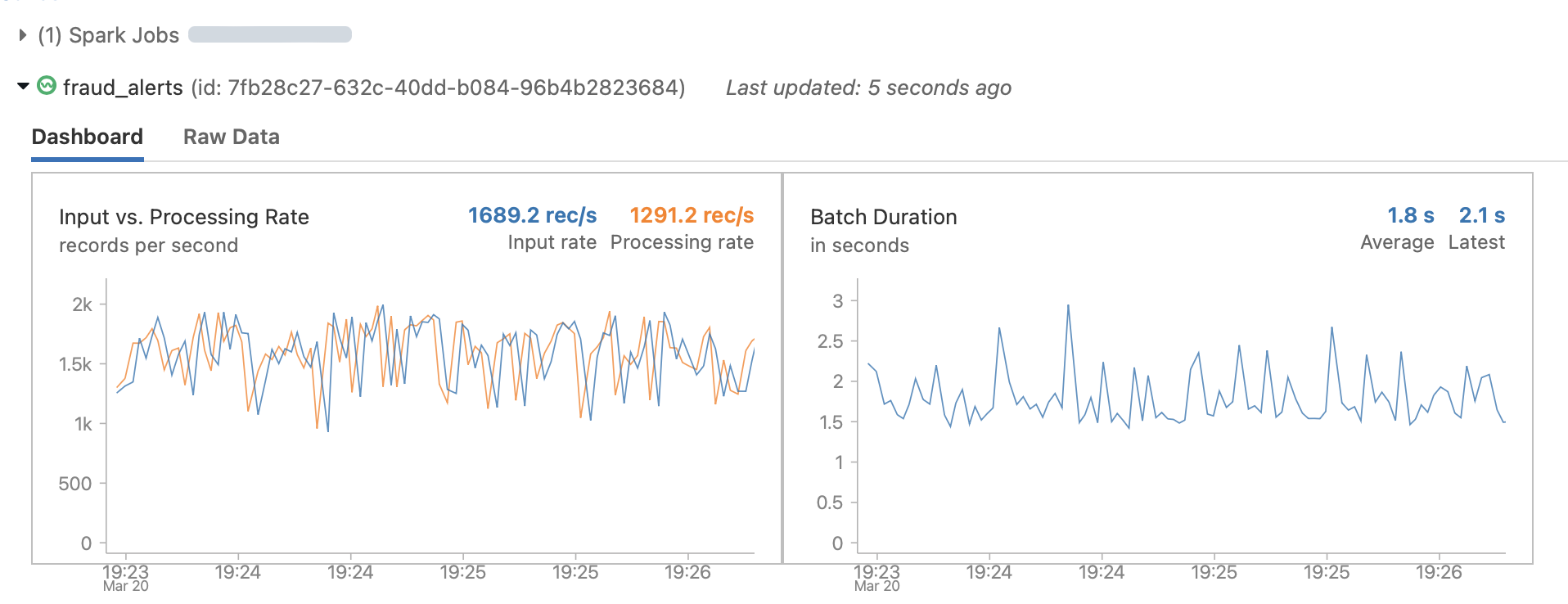
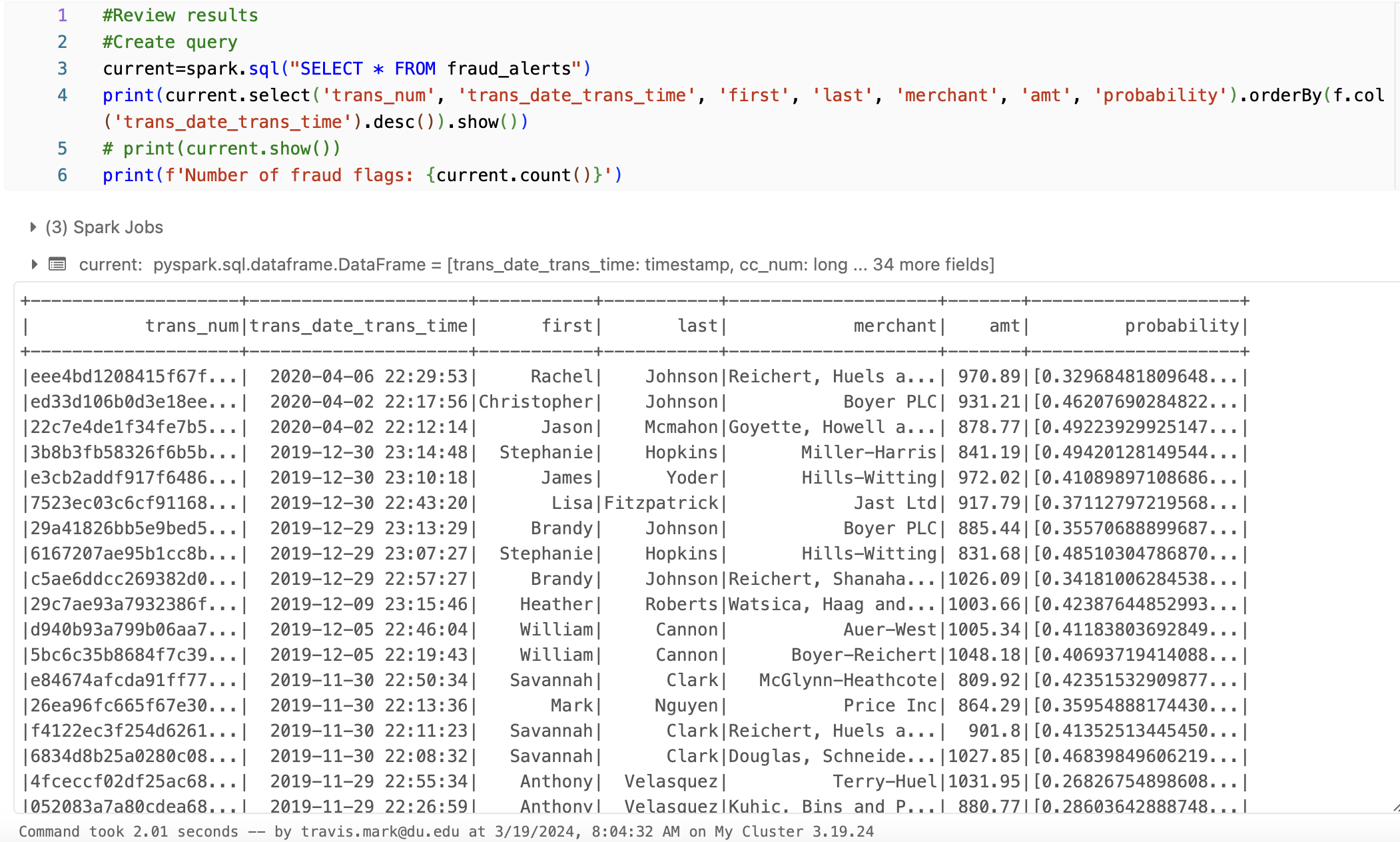
Using the XGBoost model on the test data, we can see how well the model will generalize to unseen data. The selected model performed well and very similar to how it did in validation. The model achieved a F1 score of 0.80 and a false negative rate of 0.30. The area under ROC curve for the test data is 0.982. While it is to be expected that the model will not perform as well on the unseen data, this indicates that the model is robust and not overfit.

A sample of the both non-fraud and fraud predictions on the test data:

# Streaming

The objective of this scenario is to be able to run live real-time transactions through the model and provide fraud detection to prevent flagged transactions from being processed. To simulate this using streaming in Spark, I broke the test data into multiple files. The test data contained over 550,000 transactions over the course of 24 months. I partitioned the data by date into 500 separate files, each containing a little over 1,000 transactions. I then created a stream to read in these files one at a time. To be as realistic as possible, as each file came into the stream, I performed the feature engineering for age and transaction hour. I then passed the data to the pipeline for transformation into predictions. Finally, I created a query to pick out transactions which were flagged as fraud and displayed them in descending order of dates, simulating new transactions coming into the stream and being predicted as fraudulent, as well as a count of the number of flagged transactions.

In the real world, this stream would output the fraudulently flagged transactions to the processing system to prevent them from being finalized.

Sample of query output:

# Future Iterations

While completing this model building and evaluation provided me with ideas for how to continue this in the future. Even in the best performing model, we are missing nearly 30% of fraudulent transactions. This is a decent outcome for the proof of concept. In the real world, there are features which provide information on whether the card was swiped or not present for the transaction (as in entered online) which may be useful in identifying fraudulent behavior. Additionally, creating a calculation to check for the distance a merchant is from a cardholder’s address. This may also be helpful in identifying behavior which is out of the ordinary for individuals or outside of a radius near the cardholder. I would look to analyze the feature importance to inform which features to select in the model based upon which provide the most useable information and eliminate those which may not be beneficial. I did do some of this in the assignment by comparing the use of bucketed minutes or not including them and by comparing the use of age buckets versus the raw age value. I also did do some hyperparameter tuning in my model selection, outside of the 3 discussed here. I explored which subset size performed well for random forest and how many iterations to run of the XGBoost, I ran sizes of 10, 20, 25, and 30, before landing on my model to showcase. I would like to perform further hyperparameter tuning using grid search to try to continue to improve the model.