Data Science for Venmo (DSV)

1

Project Overview

What is Venmo?

- Venmo is a mobile payment service owned by PayPal
- Account holders can transfer funds to others via a mobile phone app

$160,000,000,000

was the total payment volume processed by Venmo in 2020

What is this dataset?

- A Github user, sa7mon, uploaded a dataset of over 7,000,000 transactions scraped from the Venmo public API (July - September).
- He wanted to show how easy it is to get a lot of information without even an API key

What is our project (DSV)

Data Pre-Processing

- ➔ Anonymizing the data
  - ◆ Name
  - ◆ Photos
- ➔ Imputing/Filling any missing data
- ➔ Turning text data into categorical (numbers)

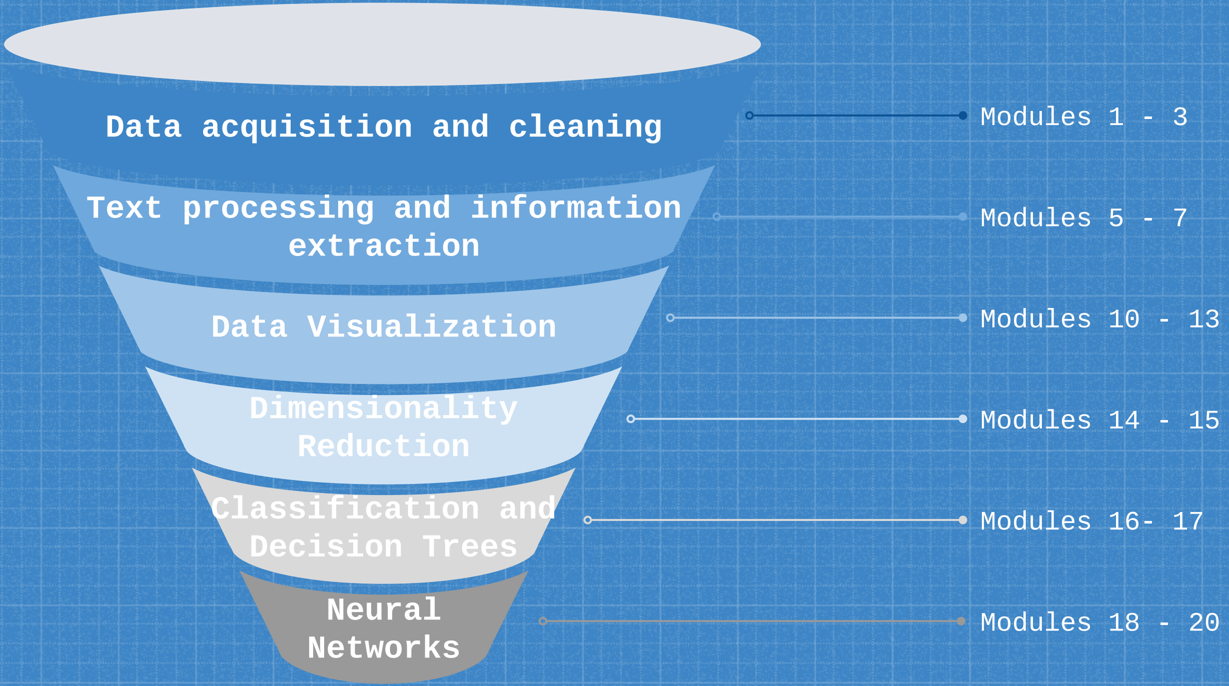
Data Processing

- ➔ Run classification models
  - ◆ Random Forest
  - ◆ SVM
- ➔ Run neural networks
  - ◆ Logistic regression
  - ◆ Feed-Forward NN

2

Application of Course Topics

Coure Topics and their use



Data acquisition and cleaning — Modules 1 - 3

Text processing and information extraction — Modules 5 - 7

Data Visualization — Modules 10 - 13

Dimensionality Reduction — Modules 14 - 15

Classification and Decision Trees — Modules 16- 17

Neural Networks — Modules 18 - 20

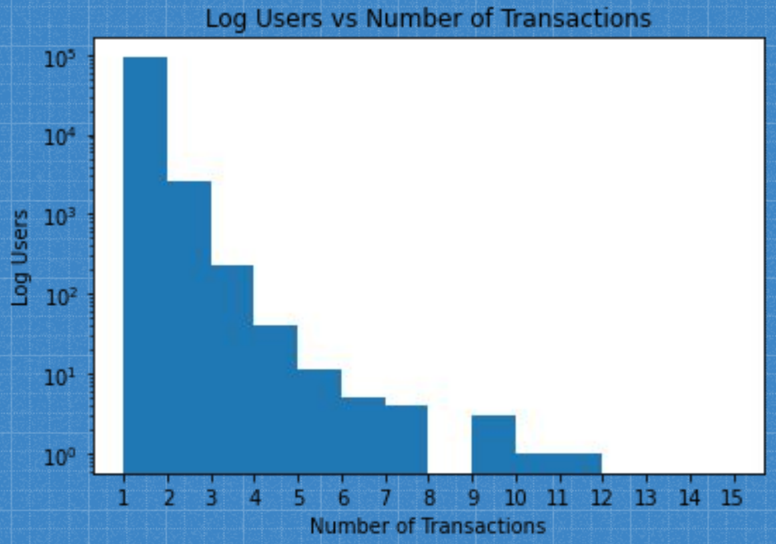
3

Exploratory Data Analysis (EDA)

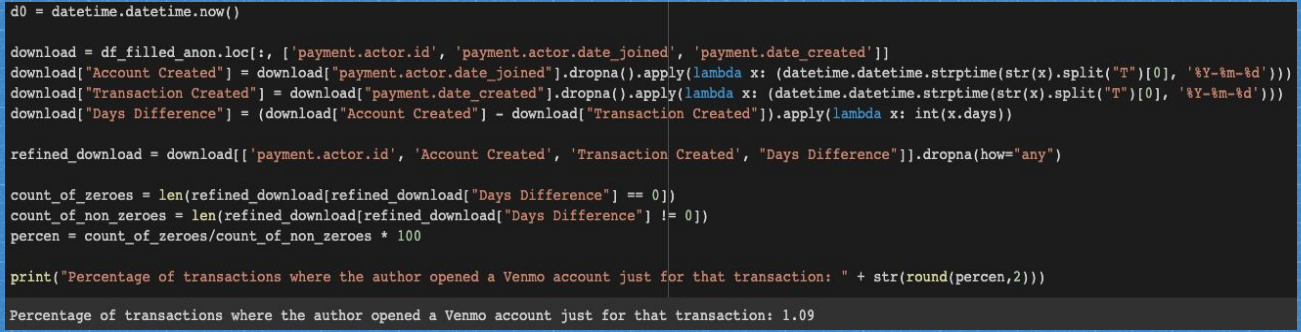
EDA 1: User IDs (log) Vs. Number of Transactions

Conclusions

- Large skew towards the number of transactions being 1.



EDA 2: Percentage of transactions where the author created a Venmo account just for that transaction



```
d0 = datetime.datetime.now()

download = df_filled_anon.loc[:, ['payment.actor.id', 'payment.actor.date_joined', 'payment.date_created']]
download["Account Created"] = download["payment.actor.date_joined"].dropna().apply(lambda x: (datetime.datetime.strptime(str(x).split("T")[0], '%Y-%m-%d')))
download["Transaction Created"] = download["payment.date_created"].dropna().apply(lambda x: (datetime.datetime.strptime(str(x).split("T")[0], '%Y-%m-%d')))
download["Days Difference"] = (download["Account Created"] - download["Transaction Created"]).apply(lambda x: int(x.days))

refined_download = download[['payment.actor.id', 'Account Created', 'Transaction Created', "Days Difference"]].dropna(how="any")

count_of_zeroes = len(refined_download[refined_download["Days Difference"] == 0])
count_of_non_zeroes = len(refined_download[refined_download["Days Difference"] != 0])
percen = count_of_zeroes/count_of_non_zeroes * 100

print("Percentage of transactions where the author opened a Venmo account just for that transaction: " + str(round(percen,2)))
```

```
Percentage of transactions where the author opened a Venmo account just for that transaction: 1.09
```

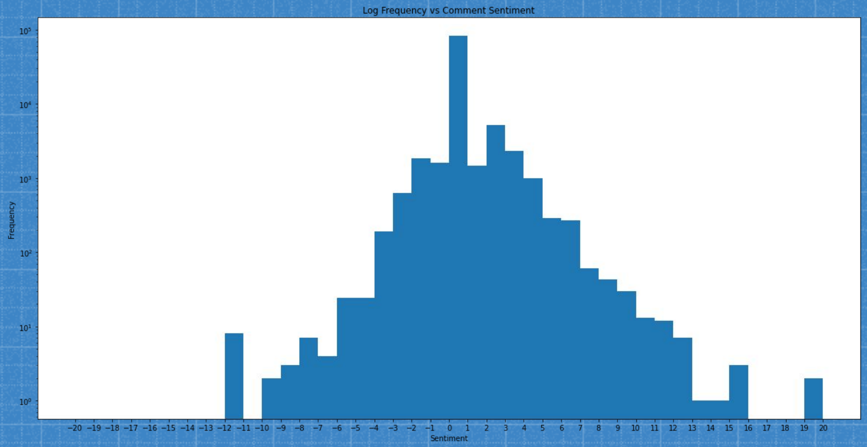
Conclusions

- Very few users make the account just to fulfil a request, most already have it.

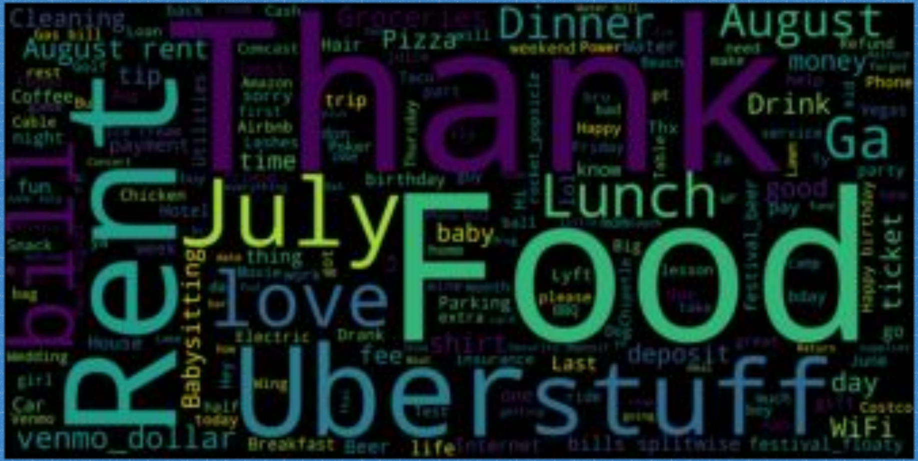
EDA 3: Frequency (log) Vs. Comment Sentiment

Conclusions

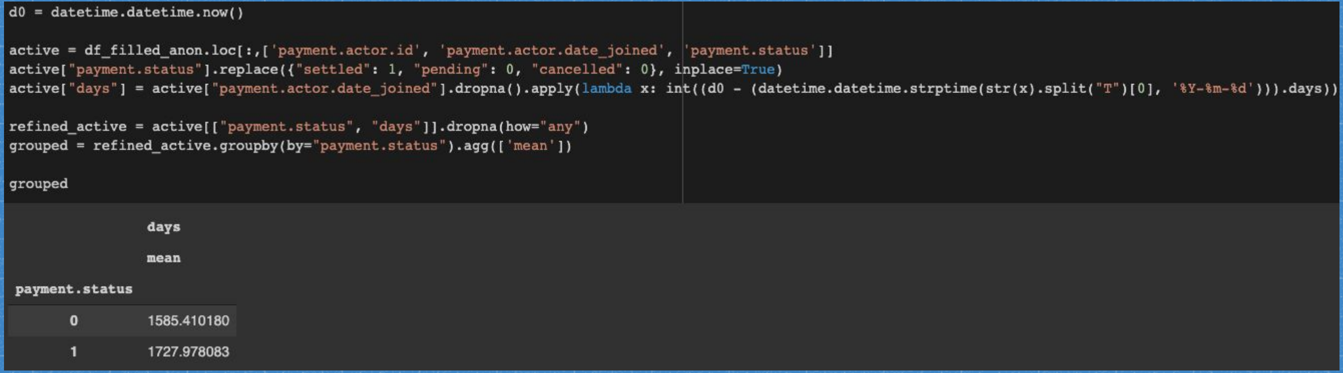
- The use of the app vs. the sentiment of the comment
- Usually comments are neutral, a bell curve represents the data and lower frequency of extremes



EDA 4: Word Cloud for comment titles



EDA 5: Correlation between the duration a user has been on Venmo for to their completion of a request



```
d0 = datetime.datetime.now()

active = df_filled_anon.loc[:,['payment.actor.id', 'payment.actor.date_joined', 'payment.status']]
active["payment.status"].replace({"settled": 1, "pending": 0, "cancelled": 0}, inplace=True)
active["days"] = active["payment.actor.date_joined"].dropna().apply(lambda x: int((d0 - (datetime.datetime.strptime(str(x).split("T")[0], '%Y-%m-%d'))).days))

refined_active = active[["payment.status", "days"]].dropna(how="any")
grouped = refined_active.groupby(by="payment.status").agg(['mean'])

grouped
```

```
                      days
                      mean
payment.status
             0        1585.410180
             1        1727.978083
```

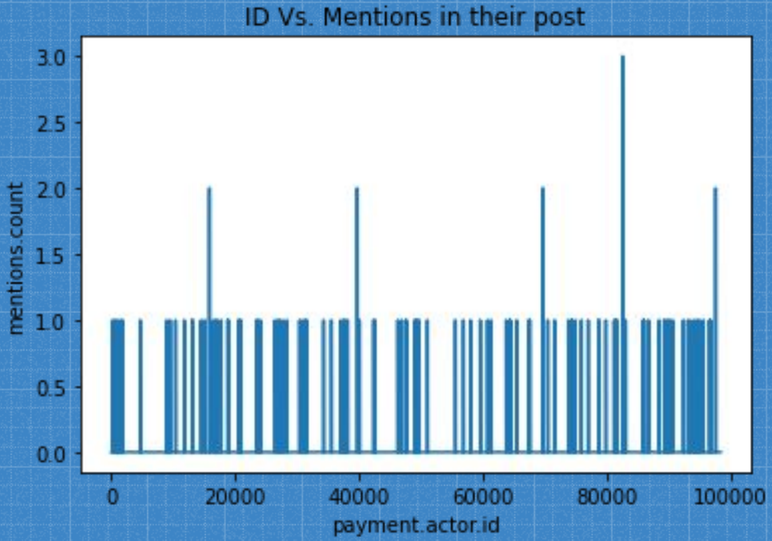
Conclusions

- The longer a user has been using venmo, the more likely they are to complete a given request

EDA 6: User IDs Vs. Number of mentions in their post

Conclusions

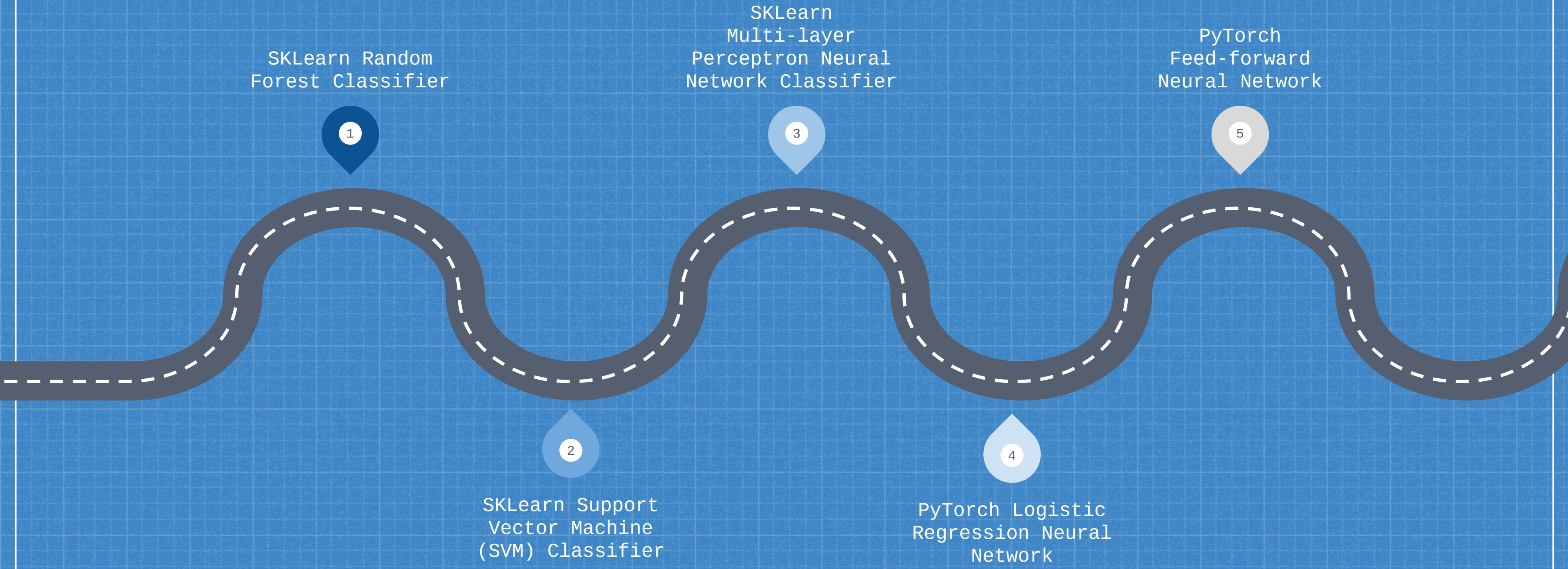
- Usually, no more than 2 individuals are mentioned or tagged in a transaction
- Useful for data cleaning



4

Modeling

Modelling techniques used



1 SKLearn Random Forest Classifier

2 SKLearn Support Vector Machine (SVM) Classifier

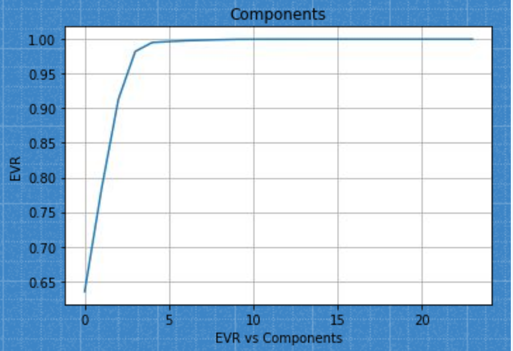
3 SKLearn Multi-layer Perceptron Neural Network Classifier

4 PyTorch Logistic Regression Neural Network

5 PyTorch Feed-forward Neural Network

Modeling Parameters

1. 80% train data set, 20% test data set
2. Labels (representing **frequency of transaction**) range: [1, 2, 3, 4, 5, 6, 7, 9, 10, 11]
3. PCA Components = 8 (0.99851582)
4. Input Dimensions = 8
5. Output Dimensions = 10

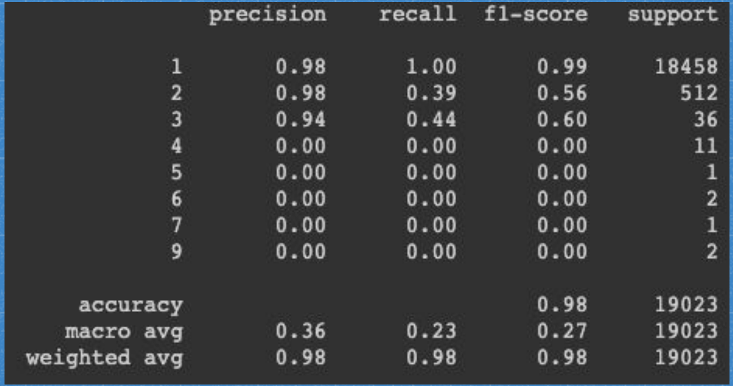


CLASSIFIERS

SKLearn Random Forest Classifier

- Label Prediction Precision for 1:**98%**
- Label Prediction Precision for 2:**98%**
- Label Prediction Precision for 3:**94%**
- Label Prediction Precision for 4:**0%**
- Label Prediction Precision for 5:**0%**
- Label Prediction Precision for 6:**0%**
- Label Prediction Precision for 7:**0%**
- Label Prediction Precision for 9:**0%**

Weighted Average Accuracy for SKLearn RFC : 98%

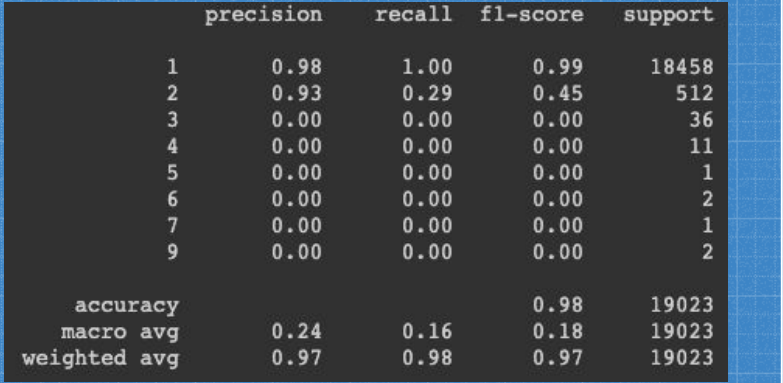


```
              precision    recall  f1-score   support

           1       0.98      1.00      0.99     18458
           2       0.98      0.39      0.56       512
           3       0.94      0.44      0.60        36
           4       0.00      0.00      0.00        11
           5       0.00      0.00      0.00         1
           6       0.00      0.00      0.00         2
           7       0.00      0.00      0.00         1
           9       0.00      0.00      0.00         2

    accuracy                           0.98     19023
   macro avg       0.36      0.23      0.27     19023
weighted avg       0.98      0.98      0.98     19023
```

SKLearn Support Vector Machine (SVM) Classifier

★ Label Prediction Precision for 1:**98%**
★ Label Prediction Precision for 2:**93%**
★ Label Prediction Precision for 3:**0%**
★ Label Prediction Precision for 4:**0%**
★ Label Prediction Precision for 5:**0%**
★ Label Prediction Precision for 6:**0%**
★ Label Prediction Precision for 7:**0%**
★ Label Prediction Precision for 9:**0%**

Weighted Average Accuracy for SKLearn SVM : 98%

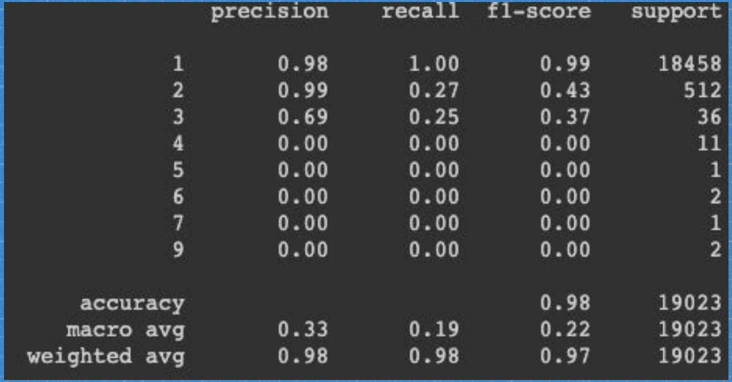


```
              precision    recall  f1-score   support

           1       0.98      1.00      0.99     18458
           2       0.93      0.29      0.45       512
           3       0.00      0.00      0.00        36
           4       0.00      0.00      0.00        11
           5       0.00      0.00      0.00         1
           6       0.00      0.00      0.00         2
           7       0.00      0.00      0.00         1
           9       0.00      0.00      0.00         2

    accuracy                           0.98     19023
   macro avg       0.24      0.16      0.18     19023
weighted avg       0.97      0.98      0.97     19023
```

SKLearn Multi-layer Perceptron Neural Network Classifier

- Label Prediction Precision for 1:**98%**
- Label Prediction Precision for 2:**99%**
- Label Prediction Precision for 3:**69%**
- Label Prediction Precision for 4:**0%**
- Label Prediction Precision for 5:**0%**
- Label Prediction Precision for 6:**0%**
- Label Prediction Precision for 7:**0%**
- Label Prediction Precision for 9:**0%**

Weighted Average Accuracy for SKLearn MPNNC : 98%



```
              precision    recall  f1-score   support

           1       0.98      1.00      0.99     18458
           2       0.99      0.27      0.43       512
           3       0.69      0.25      0.37        36
           4       0.00      0.00      0.00        11
           5       0.00      0.00      0.00         1
           6       0.00      0.00      0.00         2
           7       0.00      0.00      0.00         1
           9       0.00      0.00      0.00         2

    accuracy                           0.98     19023
   macro avg       0.33      0.19      0.22     19023
weighted avg       0.98      0.98      0.97     19023
```

NEURAL NETWORKS

PyTorch Logistic Regression Neural Network

Weighted Average Accuracy for Logistic Regression NN EPOCH 0 LOSS : 2%

Weighted Average Accuracy for Logistic Regression NN EPOCH 0 ACCURACY : 96%

Weighted Average Accuracy for Logistic Regression NN EPOCH 1 LOSS : 2%

Weighted Average Accuracy for Logistic Regression NN EPOCH 1 ACCURACY : 97%

Weighted Average Accuracy for Logistic Regression NN EPOCH 2 LOSS : 2%

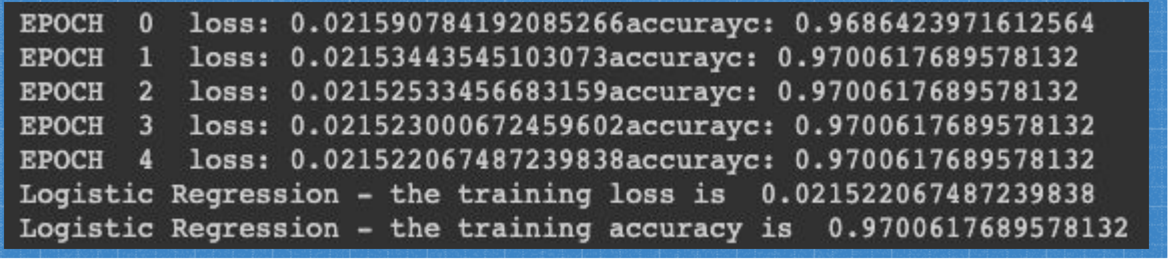
Weighted Average Accuracy for Logistic Regression NN EPOCH 2 ACCURACY : 97%

Weighted Average Accuracy for Logistic Regression NN EPOCH 3 LOSS : 2%

Weighted Average Accuracy for Logistic Regression NN EPOCH 3 ACCURACY : 97%

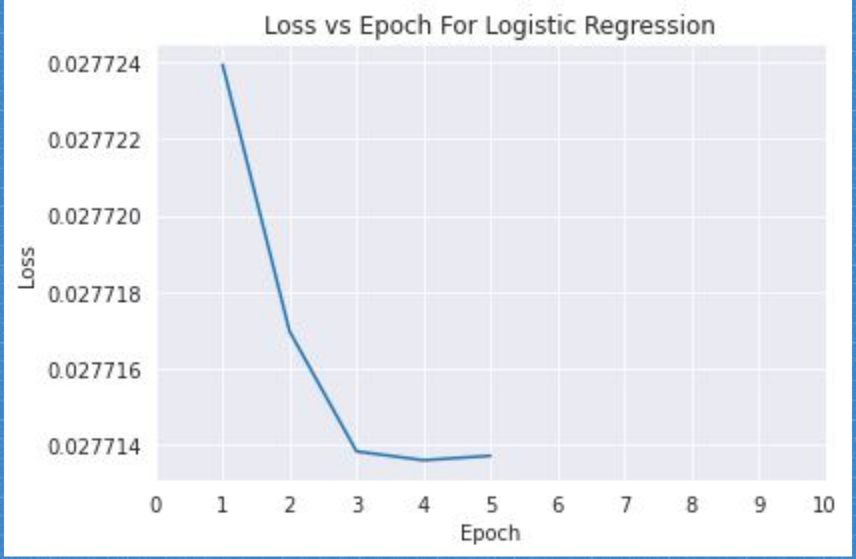
Weighted Average Accuracy for Logistic Regression NN EPOCH 4 LOSS : 2%

Weighted Average Accuracy for Logistic Regression NN EPOCH 4 ACCURACY : 97%

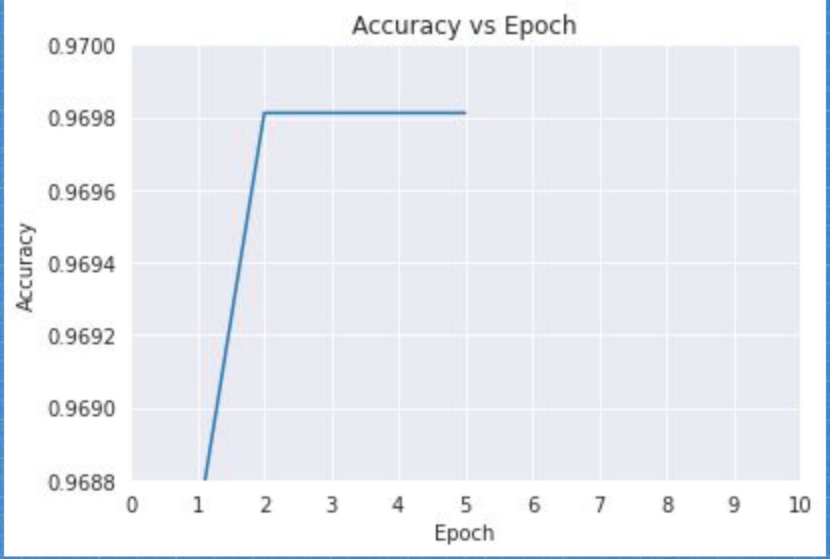


```
EPOCH  0  loss: 0.02159078419208526 6accurayc: 0.9686423971612564
EPOCH  1  loss: 0.0215344354510307 3accurayc: 0.9700617689578132
EPOCH  2  loss: 0.0215253345668315 9accurayc: 0.9700617689578132
EPOCH  3  loss: 0.02152300067245960 2accurayc: 0.9700617689578132
EPOCH  4  loss: 0.02152206748723983 8accurayc: 0.9700617689578132
Logistic Regression - the training loss is  0.02152206748723983 8
Logistic Regression - the training accuracy is  0.9700617689578132
```

PyTorch Logistic Regression Neural Network



Loss vs Epoch For Logistic Regression



Accuracy vs Epoch

PyTorch Feed-Forward Neural Network

Weighted Average Accuracy for Feed-Forward NN EPOCH 0 LOSS : 2%

Weighted Average Accuracy for Feed-Forward NN EPOCH 0 ACCURACY : 96%

Weighted Average Accuracy for Feed-Forward NN EPOCH 1 LOSS : 1%

Weighted Average Accuracy for Feed-Forward NN EPOCH 1 ACCURACY : 97%

Weighted Average Accuracy for Feed-Forward NN EPOCH 2 LOSS : 1%

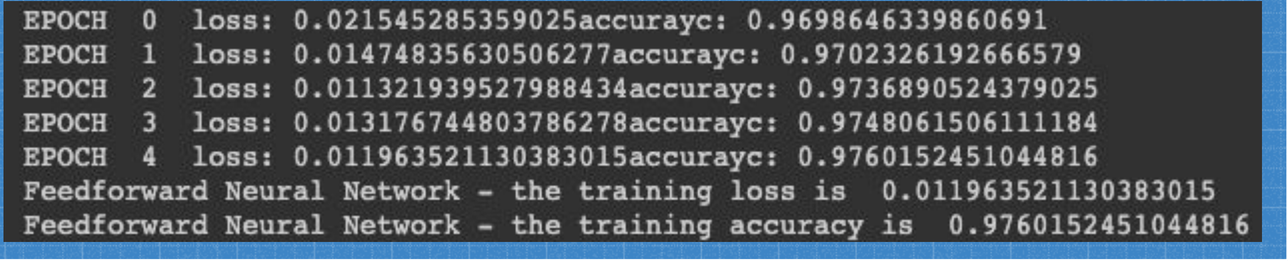
Weighted Average Accuracy for Feed-Forward NN EPOCH 2 ACCURACY : 97%

Weighted Average Accuracy for Feed-Forward NN EPOCH 3 LOSS : 1%

Weighted Average Accuracy for Feed-Forward NN EPOCH 3 ACCURACY : 97%

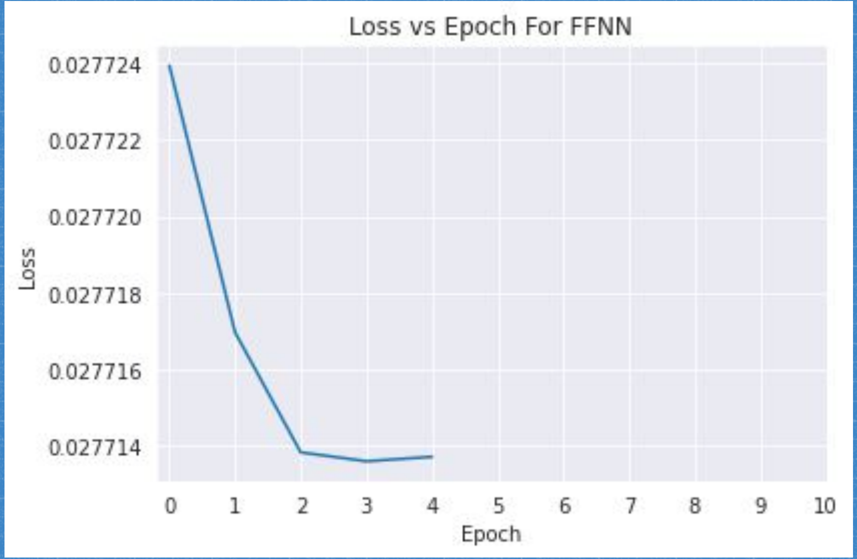
Weighted Average Accuracy for Feed-Forward NN EPOCH 4 LOSS : 1%

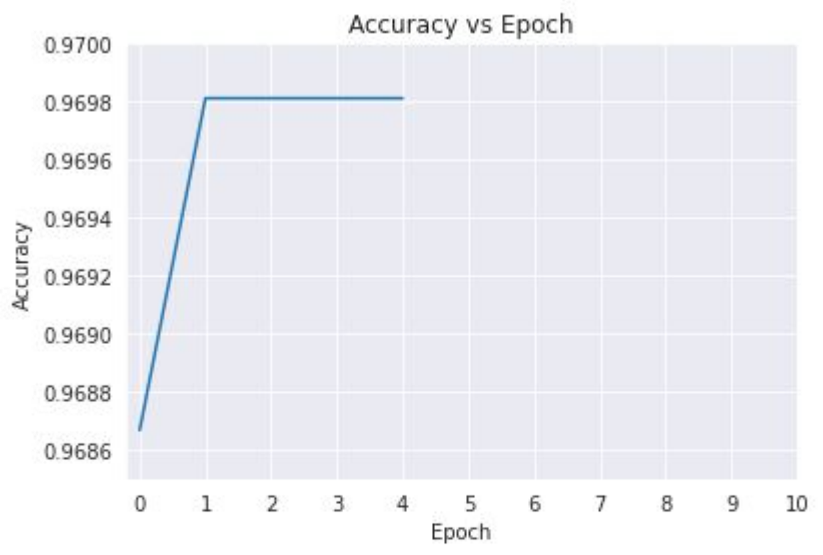
Weighted Average Accuracy for Feed-Forward NN EPOCH 4 ACCURACY : 97%



```
EPOCH  0  loss: 0.021545285359025accurayc: 0.9698646339860691
EPOCH  1  loss: 0.01474835630506277accurayc: 0.9702326192666579
EPOCH  2  loss: 0.011321939527988434accurayc: 0.9736890524379025
EPOCH  3  loss: 0.013176744803786278accurayc: 0.9748061506111184
EPOCH  4  loss: 0.011963521130383015accurayc: 0.9760152451044816
Feedforward Neural Network - the training loss is  0.011963521130383015
Feedforward Neural Network - the training accuracy is  0.9760152451044816
```

PyTorch Feed-Forward Neural Network





MODELLING SUMMARY

5

Challenges/Obstacles faced

Challenges Faced

Data reduction

Since the dataset of 7,000,000 was too big, we used a smaller dataset of 100k for testing. To overcome this challenge, we used the .pkl file format to conduct our final testing on the entire dataset

Data Visualization

We had to consider the best way to demonstrate what we were trying to show. The solution for this problem was to try a few different illustration techniques and pick the best one!

Sentiment Analysis

The sentiment analysis was challenging, especially with the abundant use of emojis. The solution to this was to use a Python library that would convert the emojis to text and then this analysis went a lot smoother.

Neural Network Modelling

The NN modelling was one of the most challenging parts of the project as we couldn’t really use much from HW5 and had to refresh our understanding to make the model in a format most appropriate for our use

6

Conclusions for Venmo

Conclusions

1. Skewed data (99% of data had label of 1)
2. Average model precision was 98% but this was due to the skew. Additional data would be required to train the models on more data with greater frequencies
3. Users that are predicted to be **premium** users (predicted frequency label > 3) can be rewarded with special perks
4. Users that are predicted to be **poor** users (predicted frequency label <= 1) can be given special perks to incentivise more use

7

Potential Next Steps

Beyond this project?

- Reach out to Venmo to ask them of their opinions
- Reach out to other companies (Ex. Square, Cash App, etc.) to conduct similar analysis for them
- Work with a consulting club at Penn to come up with real-life recommendations based on the results of our model.

Sample Recommendations

Users predicted to be PREMIUM **(transaction frequency > 3)** should be offered:

a. Discounts on the Venmo Credit Card
b. Moving up on the waitlist for the Venmo Credit Card
c. Allow these users to have more than the current daily limit of transactions (30)
d. Additional perks!

Thanks!

ANY QUESTIONS?