#Machine Learning Engineer Nanodegree Capstone Project Report

Chinmay Gopal
September 16th , 2020

Definition

Project Overview

Coffee is enjoyed almost religiously around the entire world, so much so that it is a 250 billion dollar industry. Arguably at the forefront of the coffee is Starbucks, a company which alone boasts a 40% market share in the US. Starbucks is a huge multinational customer conglomerate which could be found in 80 different countries. Besides, As of 2019 Starbucks is worth a staggering 100 million dollars and in that year alone sold over 2.3 billion glasses of the good stuff.

Thanks to starbucks and Udacity I can base my Capstone project using their impressive dataset. This dataset allows me to really stretch my analytical skills and really delve deep into data on a macro level. Not to mention that this Starbucks dataset includes a very real world experience in data science.

The 80-20 rule is prominently visible throughout many different businesses. The 80-20 rule in business economics refers to the phenomena that roughly 80% of revenue comes from about 20% of your customers. More specifically this refers to regular Starbucks customers that facilitate about %80 of the company's total revenue. Since frequency customers create so much revenue understanding these customers and their behaviour is very important for the future.

Consequently, it should come to no surprise that transforming infrequent customers into regulars is paramount to the company's future. Using data from infrequent customers such as, which offers compelled them to buy as well as which offers kept them engaged the longest, we can drastically improve the company’s future revenue.

In our dataset we find that over 84% of the revenue does indeed come from around 20% of the most frequent customers.

###Project Statement

The objective of this project is to increase Starbucks’s revenue by using user data as well as spending habits to create a model which targets infrequent customers at the Company. The model aims to predict whether or not the customer will use the offer or not. Understanding the customer’s spending habits will allow us to determine the type of offer to provide to customers to make them a regular at Starbucks. The motivation of this project is to help Starbucks efficiently use their offers to convert infrequent customers into regulars.

Problem Statement

The main purpose of this project is to transform infrequent customers into regular customers by using the most economical offers possible. This dynamic problem can be addressed in two major steps **A)**Using data to figure out which customers are irregulars

B) Using prior offer and customer data to predict which offers are needed to make that customer become a “regular”.

Due to the project’s nature I have decided to focus on the accuracy of the model in predicting the customer’s spending habits. This is mainly because I have no access to future data on how well our model is doing in creating more revenue.

###Solution Statement

Problem A- After data preprocessing and data visualisation. I created a threshold for customers who purchase Starbucks frequently by selecting 80% of the customer demographic who spends the least amount of money and has the least amount of transactions at the establishment. Uncoincidentally, this 80% also happens to just account for just about 23% of Starbucks revenue. The exact value I settled for is people with less than 25 transactions and who have spent less than $375 dollars at the establishment.

Problem B - The selected customer demographic was then run through the benchmark, Svm Classifier,MLP Classifier, another special neural net and a Logistic Regression. The model features were transactions, offers received, offers viewed, and offers completed gender, Money spent and Income. This model tries to predict the user's offer completion behaviour.

The purpose

###Evaluation Metrics

Due to the nature of this project, determining whether or not a customer becomes a more frequent customer at Starbucks would require real time data which I do not have access to. Therefore, after careful consideration, I decided that predicting customer behaviour is sufficient in analysing model performance.

In particular I chose to parse through to customer records and group each individual customer's offers into 2 main categories.

1-Viewed and Completed - This signifies that the user gave some thought into using the offer and decided it is preferable to use it.

2- Viewed and Not Completed - This signifies that the customer viewed the order and purposely chose not to use it for whatever reason.

	ViewAndComp	ViewAndNot	BogoCount	ICount	DisCount	Difficulty	Reward	Type	TotalTransactions	TotalSpent	UserID
0	1	4	1	0	2	10	2	3	8	127.60	0009655768c64bdeb2e877511632db8f
1	0	1	0	0	1	7	3	3	3	4.09	00116118485d4dfda04fdbaba9a87b5c
2	0	5	2	0	1	10	10	1	5	79.46	0011e0d4e6b944f998e987f904e8c1e5
3	1	2	1	0	2	5	5	1	8	196.86	0020c2b971eb4e9188eac86d93036a77
4	0	4	1	0	1	5	5	1	13	159.06	0020ccbbb6d84e358d3414a3ff76cffd
5	1	3	0	0	3	10	2	3	17	43.33	003d66b6608740288d6cc97a6903f4f0
6	0	2	0	0	1	10	2	3	17	68.51	00426fe3ffde4c6b9cb9ad6d077a13ea
7	2	0	1	0	1	5	5	1	7	173.50	004b041fbfe44859945daa2c7f79ee64
8	1	3	3	0	2	10	2	3	12	321.27	004c5799adbf42868b9cff0396190900
9	0	4	1	0	0	5	5	1	3	11.33	005500a7188546ff8a767329a2f7c76a

Using this data I created a supervised model to determine user behaviour.

Using these categories as features I used **F1** and **Accuracy** to evaluate and compare each model.

$$Accuracy = \frac{truepositives + truenegatives}{totalexamples}$$

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

Analysis

Datasets and Inputs

The dataset was divided into 3 parts, the profile, the transcript and the portfolio. In total the dataset has 1700 rows and about 15 columns.

In this project the final list of attributes I used are:

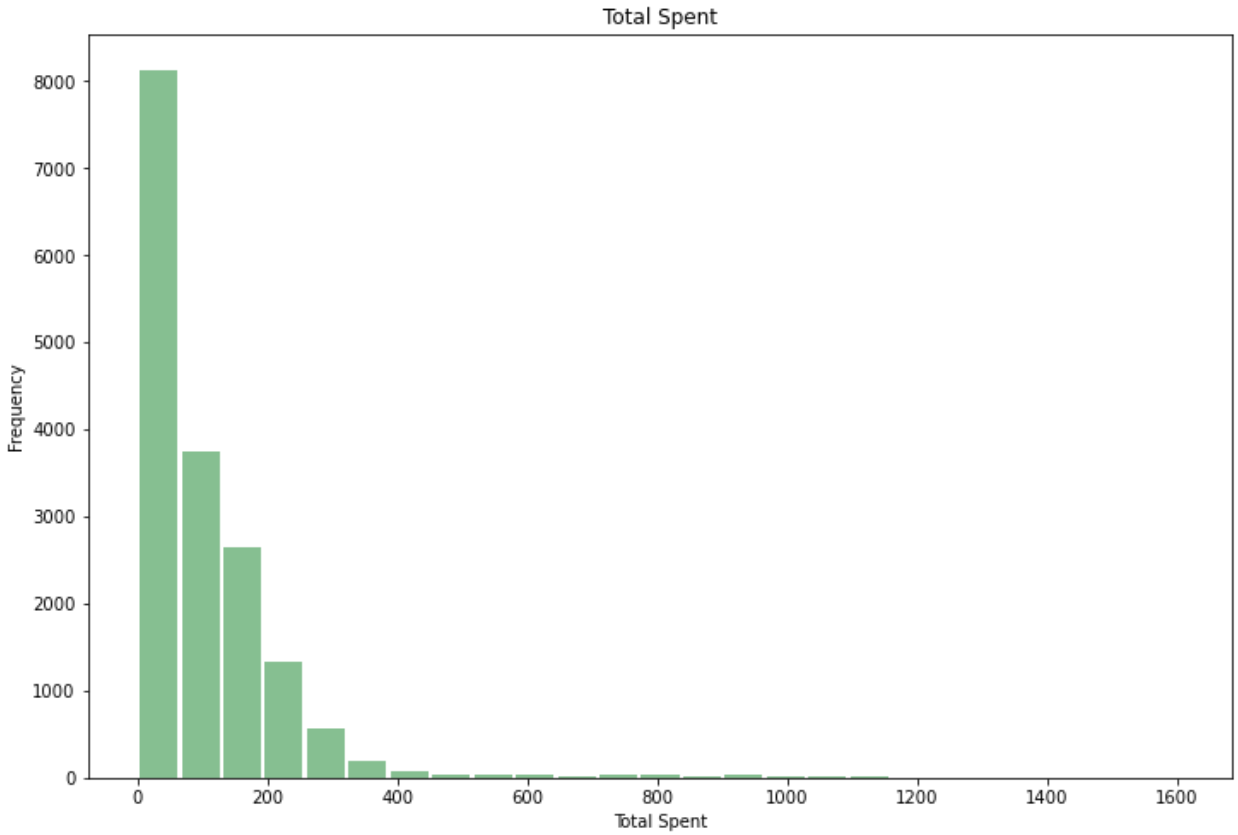
Offer details - offer type, reward, event, value
Customer details - Age, Gender, Income , Total spent, total transactions

Using pandas's merge function I joined all the datasets and eliminated the unwanted rows. All the Nan rows are eliminated because of their relatively small size. For columns such as the bogo count, total transaction count or the total spent I used a simple for loop to traverse a user id sorted dataset and added the values in a double array.
Offer details are very important in learning the attributes of the offer but I chose to not use duration because of its low significance. Offer information is key in determining which offers to give to our customers.
On the other hand, Customer details are also key in determining what kind of a person you are. For example, a customer who is female may like certain offers more than that of a male or a richer customer might like expensive items and require fewer offers.
In particular I used gender, income, total amount of viewed and completed offers prior and the total amount of viewed and not completed offers prior. It was the same methodology in creating these columns, using a for loop to traverse and fill a double array.

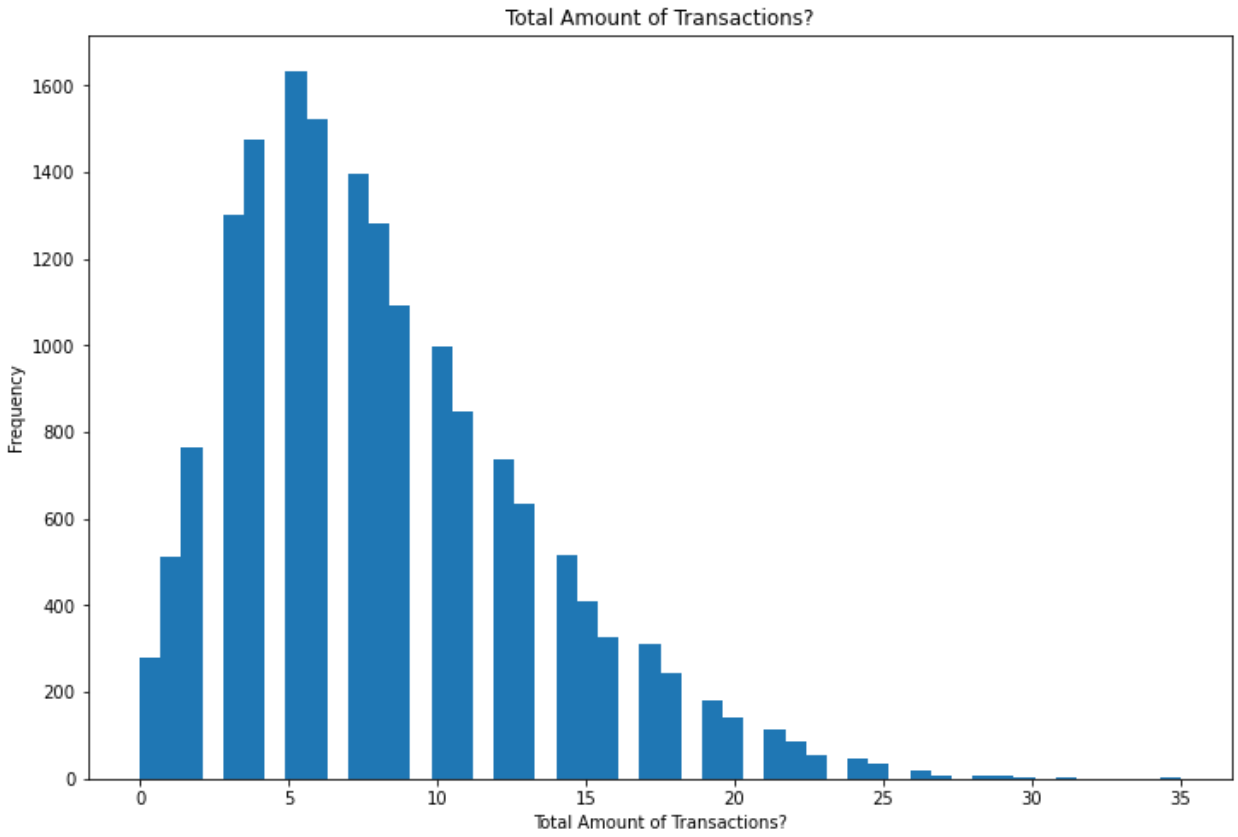
Because I do not have access to future data and cannot access the total amount of money lost/ gained or even the increase or decrease of customer frequency after Starbucks uses my model so I decided to focus my model on predicting customer behaviour. Therefore, I used the output as whether or not (1 or 0) a user viewed and completed a specific offer in the dataset. Specifically finding this offer involved me traversing the dataset looking for the very first viewed and completed/ not completed offer. I then decided to split the data into 70% for training and the remaining 30% for testing for all the models. The entirety of the data used in the project was provided by Starbucks through Udacity.

#Data Visualisation

Here is a histogram showing how Starbucks's demographic spends their money. On the x axis we have the total spent in dollars of customers and on the y axis we have the frequency. Immediately we can see the exponential nature of frequency as spending decreases.



Here is a very similar Histogram representing the total amount of transactions completed by customers. On the X axis we have total number of Transactions and on the y axis we have the frequency. We can see that the graph is skewed to the right with the majority of customers only completing less than 10 transactions.



These are the two main graphs from which I made the decision to split the data and differentiate regular and infrequent customers.

84% of customers have completed less than 25 transactions

82% of customers have spent less than $375 dollars at Starbucks.

Algorithms and techniques

After data preprocessing and wrangling the benchmark model was the random forest regression followed by logistic regression. These two models were chosen as a benchmark mainly because of their high interpretability, their computational efficiency and the binary nature of the data. The highest scoring benchmark model happened to be the logistic regression which ammsed a high test accuracy of ***81.98%*** Whereas the Random Forest regression had just a ***54.7%*** accuracy. It should be noted that the random Forest regression was especially flagged to being excessively overfitted. This can be seen in the high discrepancy between the training(93.8%) and testing accuracies.

Benchmark Models

Model Attributes for the Random Forest Classification

- Number of Trees was set to 100
- Loss function was MSE
- Min Number of sample splits was 2
- Min sample leaf was set to 1
- Tested on F1

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import  accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import f1_score,
from sklearn.utils.multiclass import unique_labels
empty = {}

rand = RandomForestClassifier(random_state=101) # RandomForestClassifier

gsc= GridSearchCV(estimator=rand, param_grid=empty, scoring='accuracy', cv=10)
print("Training {} :".format(rand.__class__.__name__))
gsc.fit(X, y)

print(rand.__class__.__name__)
print("Best accuracy Value attained: {}".format(round(grid.best_score_,4)))

grid = GridSearchCV(estimator=rand, param_grid=empty, scoring='f1', cv=10)
print("Training {} :".format(rand.__class__.__name__))
gsc.fit(X, y)

print(rand.__class__.__name__)
print("Best F1 score attained: {}".format(round(gsc.best_score_,4)))
```

Model Attributes for the logistic Regression

- Penalty was l2
- Max iteration was set to 1000
- Fit intercept was set to False
- Tested on F1

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score

logit = LogisticRegression(random_state=101, max_iter=1000).fit(X_train, y_train)


score = logit.score(X_test, y_test)
print("The accuracy of the baseline Logistic regression is " + str(score))

tester= logit.predict(X_test)
ans=f1_score(y_test, tester)

print("The f1 score of the Logistic regression is " + str(ans))
```

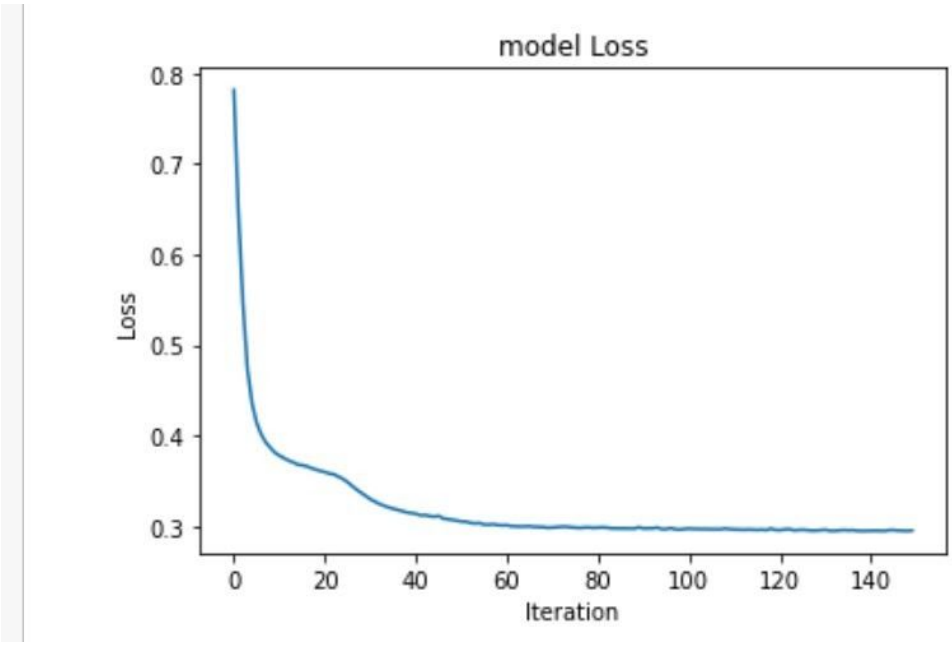
Advanced Model- Neural Nets

After the benchmark models I decided to try to predict user behaviour using a Neural network. A neural Network is a state of the art network of nodes in specific layers. Initially I decided to use a Multi layer perceptron to help classify the data because of the data's binary nature.

MLP Model Attributes

- 2 hidden layers with 20 and 3 notes respectively
- Input Layer with 14 nodes (same as number of data columns)
- Output Layer with 2 nodes (1 for predicted yes and 0 for predicted no)
- Tested on F1 score

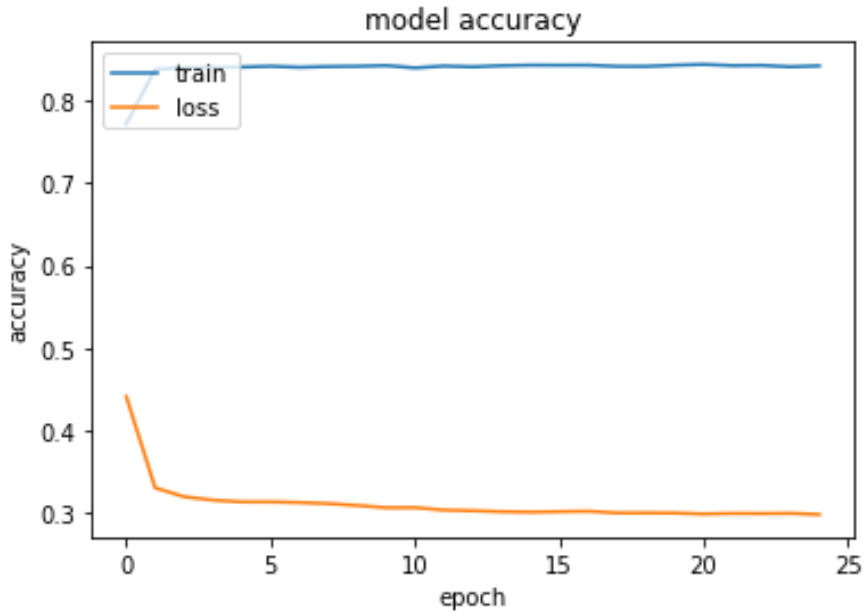
Below is the MLP's loss through testing. The MLP ended with testing accuracy of 58.8% which is below the benchmark value.



Keras Neural Network Attributes

- 5 Dense layers with sigmoid activation function at starting and output layer
- 3 Hidden layers with 15, 32, 12 nodes in each layer respectively
- The Hidden layers all used relu activation functions
- Final output layer is 2 nodes (1 for completed offer and 0 for not)
- Model was compiled with Adam optimizer, and used a loss function of sparse categorical cross entropy due to the nature of the data.
- Tested with F1 score

Below is the Neural Net's accuracy through the Epochs. The keras model performed much better with a model accuracy of 84% and a validation accuracy of 85% after just 25 epochs.



Methodology

1. **Data Cleaning** - Clean the data and make sure no missing values are present. If missing values are present eliminate them completely. I specifically chose to eliminate these rows due to their relatively insignificant size (about 4350 rows out of the total 17000 rows). Fortunately this only needed to be performed once for the profile dataset. The remaining datasets were perfect.

```
4350
17000
   gender  age                                id  became_member_on  income
0    None  118  68be06ca386d4c31939f3a4f0e3dd783          20170212     NaN
2    None  118  38fe809add3b4fcf9315a9694bb96ff5          20180712     NaN
4    None  118  a03223e636434f42ac4c3df47e8bac43          20170804     NaN
6    None  118  8ec6ce2a7e7949b1bf142def7d0e0586          20170925     NaN
7    None  118  68617ca6246f4fbc85e91a2a49552598          20171002     NaN
9    None  118  8974fc5686fe429db53ddde067b88302          20161122     NaN
10   None  118  c4863c7985cf408faee930f111475da3          20170824     NaN
11   None  118  148adfcaa27d485b82f323aaaad036bd          20150919     NaN
17   None  118  744d603ef08c4f33af5a61c8c7628d1c          20170801     NaN
23   None  118  2b826eba31074a059d63b0ae8f50b7d5          20170907     NaN
2175
```

```
print(len(profile['age']))

profile= profile.dropna()
print(len(profile['age']))
profile.head()# here we eliminated unwated rows
```

```
17000
14825
```

	gender	age	id	became_member_on	income
1	F	55	0610b486422d4921ae7d2bf64640c50b	20170715	112000.0
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	20170509	100000.0
5	M	68	e2127556f4f64592b11af22de27a7932	20180426	70000.0
8	M	65	389bc3fa690240e798340f5a15918d5c	20180209	53000.0
12	M	58	2eeac8d8feae4a8cad5a6af0499a211d	20171111	51000.0

2. **Data Segmentation** - Create some graphs emphasising the difference between frequent and infrequent customers. Create a reasonable threshold using features to differentiate types of customers. Remove frequent customers from the dataset. The graphs for these are shown above.

I decided to divide the data at the total number of transactions done at 25 and total spent at Starbucks 375 dollars.

- This is because approximately 84% of the users have transactions greator than 25 and have spent more than 375 dollars at starbucks.

n [17]:
```
#new Cut off Dataframe
from sklearn.preprocessing import MinMaxScaler

#Possible locations of cutoff
print("Length of dataset with People less than 25 transactions "+str((new_df2['TotalTransactions']<25).sum()))
print("Length of dataset with People less than 375 dollars in transactions "+str((new_df2['TotalSpent']<375).sum()))


df = new_df2[new_df2['TotalTransactions']<25]


#print(df.head(5))
df=df.drop(['UserID', 'TestId'],axis=1)




for x in df.index:
    if df['gender'][x]=='M':
        df['gender'][x]=1
    elif df['gender'][x]=='F':
        df['gender'][x]=2
    else:
        df['gender'][x]=3
        a

diff=  [0] * len(sorted_trans.person.unique())
reward=[0] * len(sorted_trans.person.unique())
type=  [0] * len(sorted_trans.person.unique())


scaler = MinMaxScaler()

df[df.columns.difference(['Yes/No'])] = scaler.fit_transform(df[df.columns.difference(['Yes/No'])])

df=df.sample(frac = 1)

print(df.head(10))
```

3. Split the remaining dataset 70-30 to training and testing.

Now that DataPreprocessign and Wrangling is done , Data splitting is Due

]:
```
import sklearn.model_selection as model_selection
#import sklearn.cross_validation as cross_validation1
from sklearn.model_selection import train_test_split

y=df['Yes/No']
#X=test_df.loc[:, test_df.columns != ['Yes/No','gender']]
X = df[df.columns.difference(['Yes/No'])]

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.75, random_state=101)
```

4. Using the mentioned data columns as features and whether or not the customer buys items we created a supervised model which trains off of prior data and learns the specifics of people's behaviour.

   Each data feature mentioned above should be input in a neural net of the classifier. This supervised method would be trained on data from previous years. The objective of the neural net is to learn certain patterns between customer type and offer details. Initially, I am hoping to create a model with a few hidden layers and a relatively high dropout rate.

5. Evaluate using accuracy.

```
Train on 13266 samples, validate on 1475 samples
Epoch 1/25
13266/13266 [==============================] - 2s 152us/step - loss: 0.4409 - accuracy: 0.7720 - val_loss: 0.3350 - val_accu
racy: 0.8380
Epoch 2/25
13266/13266 [==============================] - 2s 136us/step - loss: 0.3302 - accuracy: 0.8379 - val_loss: 0.3122 - val_accu
racy: 0.8495
Epoch 3/25
13266/13266 [==============================] - 2s 136us/step - loss: 0.3193 - accuracy: 0.8402 - val_loss: 0.3078 - val_accu
racy: 0.8522
Epoch 4/25
13266/13266 [==============================] - 2s 136us/step - loss: 0.3152 - accuracy: 0.8408 - val_loss: 0.3057 - val_accu
racy: 0.8508
Epoch 5/25
13266/13266 [==============================] - 2s 136us/step - loss: 0.3131 - accuracy: 0.8408 - val_loss: 0.3043 - val_accu
racy: 0.8522
Epoch 6/25
13266/13266 [==============================] - 2s 136us/step - loss: 0.3131 - accuracy: 0.8419 - val_loss: 0.3388 - val_accu
racy: 0.8129
Epoch 7/25
13266/13266 [==============================] - 2s 136us/step - loss: 0.3123 - accuracy: 0.8406 - val_loss: 0.3029 - val_accu
racy: 0.8549
Epoch 8/25
13266/13266 [==============================] - 2s 136us/step - loss: 0.3110 - accuracy: 0.8415 - val_loss: 0.3036 - val_accu
racy: 0.8461
Epoch 9/25
13266/13266 [==============================] - 2s 137us/step - loss: 0.3087 - accuracy: 0.8418 - val_loss: 0.3009 - val_accu
racy: 0.8495
Epoch 10/25
13266/13266 [==============================] - 2s 136us/step - loss: 0.3060 - accuracy: 0.8427 - val_loss: 0.2985 - val_accu
racy: 0.8468
Epoch 11/25
13266/13266 [==============================] - 2s 136us/step - loss: 0.3061 - accuracy: 0.8398 - val_loss: 0.3014 - val_accu
racy: 0.8393
Epoch 12/25
13266/13266 [==============================] - 2s 136us/step - loss: 0.3032 - accuracy: 0.8421 - val_loss: 0.2956 - val_accu
racy: 0.8475
Epoch 13/25
13266/13266 [==============================] - 2s 136us/step - loss: 0.3023 - accuracy: 0.8411 - val_loss: 0.2967 - val_accu
racy: 0.8495
Epoch 14/25
13266/13266 [==============================] - 2s 136us/step - loss: 0.3012 - accuracy: 0.8425 - val_loss: 0.2943 - val_accu
racy: 0.8488
Epoch 15/25
13266/13266 [==============================] - 2s 137us/step - loss: 0.3007 - accuracy: 0.8431 - val_loss: 0.3020 - val_accu
racy: 0.8393
```

Results

Due to the binary nature of the dataset the only suitable way to assess the performance of the model would be to measure the accuracy. My highest performing model being the keras Neural net gained an accuracy of 84% on the training and 85% on the validation data.

A	B	C
Models	F1 Score	Accuracy
RandomForestClassifier	0.8139	0.8335
Logistic Regression	0.7675	0.8174
MLP Classifier	0.8321	0.8437
Keras Neural Net	0.8478	0.8373

##Justification

Increasing the number of hidden layers or adjusting any more parameters in my neural net would only provide a negligible increase in performance in my opinion. Further increase in accuracy would be only possible with more data .

#Refinement

My Neural Net only had 2 hidden layers with 10 nodes each. In addition, due to my limited knowledge on Neural Nets I had been using Relu activation functions for all layers. After consulting with multiple mentors and searching through the internet I eventually landed on my current model which performs much better. I have also experimented with the batch size and the number of epochs during my refinement phase and concluded that a batch size of 10 and 25 epochs seems sufficient in this scenario.