AWS Machine Learning Engineer Capstone Report

Hieu Nguyen Minh

August 2024

1 Problem Definition

This project stems from Starbucks' direct marketing system, which maintains customer engagement by sending personalized offers through various channels like email, social media, the web, or its app. These offers include buy-one-get-one (BOGO) deals, discounts, and informational messages, each designed to incentivize purchases or provide product information. However, in order for marketing campaigns to be successful, they must generate profits. Companies need to carefully target customers likely to respond to offers while attracting new consumers and rewarding loyal ones without unnecessarily cutting into profits. Some customers only respond to rewards, while others dislike marketing entirely, highlighting the complexity of marketing decisions. With the rise of machine learning and the availability of large datasets, intelligent systems can enhance marketing campaigns by analyzing consumer behavior patterns.

1.1 Problem Statement

Starbucks invests in marketing campaigns with the goal of generating profits, making it crucial to deliver the most relevant offers to the right customers. However, some customers never see the offers, which may indicate an issue with the chosen communication channel, while others see the offers but don't make purchases, suggesting a mismatch in offer type or targeting. On the other hand, some customers engage with the offer, trying new products or spending more, which is the desired outcome. This project aims to solve the problem of identifying the most appropriate offer for each customer, which is the one that leads to a purchase influenced by the offer. If a customer doesn't see the offer or doesn't act on it, or if they make a purchase without being influenced by the offer, it is not considered effective.

1.2 Solution Statement

To address the problem, this project proposes using machine learning techniques to analyze customer behavior based on their interactions with Starbucks. Specifically, a neural network will be trained to predict how customers will respond to different offers, determining whether they will complete the offer cycle. Considering that consumer behavior is influenced by past experiences, the project will employ a Recurrent Neural Network (RNN) to account for time-dependency in decision-making.

1.3 Evaluation Metrics

The accuracy, precision, and recall will be used to evaluate the performance of both the FNN and RNN.

$$\begin{aligned} accuracy &= \frac{TP}{N}, \\ precision &= \frac{TP}{TP+FP}, \\ recall &= \frac{TP}{TP+FN}, \end{aligned} \tag{1}$$

where TP (true positive) is the number of correctly classified samples, and N is the total number of samples; FP (false positive) is the number of wrongly classified sample when the true label is negative; FN (false negative) is the number of wrongly classified sample when the true label is positive.

1.4 Project Design

I follow the guideline to design the project workflow.

1. Exploratory Data Analysis (EDA) Read data files and produce data visualization to understand the data distribution and characteristics.
2. Data cleaning and engineering Process the data, like filling the N/A values, remove duplicates, and create new features from existing features; Prepare the data to be ready to feed the neural networks; Label the record as appropriate offer or not.
3. Split the dataset into training, validation, and test sets. The training set is for training the networks, while the validation set is for evaluating the models during the training phase. The test set contains data never seen before by the network, so that we can evaluate how well the model performs on new data.
4. Build and train the FNN and RNN.
5. Evaluate and compare model performances

2 Data Analysis and Data Preprocessing

The dataset used in this project is provided by Udacity and Starbucks, consisting of simulated data that replicates customer behavior on the Starbucks rewards mobile app. The data generation program models how individuals make purchasing decisions and how promotional offers affect those decisions. Each simulated person has hidden traits that impact their buying patterns and are linked to their visible characteristics. The simulation tracks events such as receiving, opening, and responding to offers, as well as making purchases. However, the dataset does not track specific products, only the transaction or offer amounts.

Data dictionary

The dataset used in this project is provided by Udacity and Starbucks, consisting of simulated data that replicates customer behavior on the Starbucks rewards mobile app. The data generation program models how individuals make purchasing decisions and how promotional offers affect those decisions. Each simulated person has hidden traits that impact their buying patterns and are linked to their visible characteristics. The simulation tracks events such as receiving, opening, and responding to offers, as well as making purchases. However, the dataset does not track specific products, only the transaction or offer amounts.

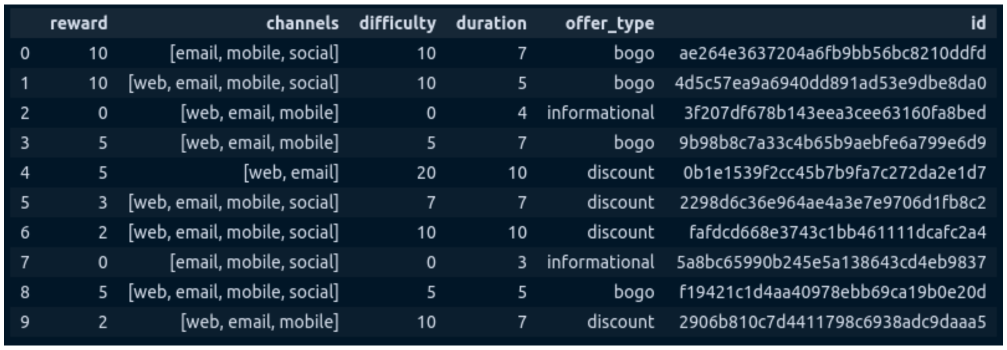
Data dictionary

1. `portfolio.json`: Offers sent during 30-day test period (10 offers x 6 fields)
   - reward: (numeric) money awarded for the amount spent
   - channels: (list) web, email, mobile, social
   - difficulty: (numeric) money required to be spent to receive reward
   - duration: (numeric) time for offer to be open, in days
   - offer_type: (string) bogo, discount, informational
   - id: (string/hash)
2. `profile.json`: Rewards program users (17000 users x 5 fields)
   - gender: (categorical) M, F, O, or null
   - age: (numeric) missing value encoded as 118
   - id: (string/hash)
   - became_member_on: (date) format YYYYMMDD
   - income: (numeric)
3. `transcript.json`: Event log (306648 events x 4 fields)
   - person: (string/hash)
   - event: (string) offer received, offer viewed, transaction, offer completed
   - value: (dictionary) different values depending on event type
   - offer id: (string/hash) not associated with any "transaction"
   - amount: (numeric) money spent in "transaction"
   - reward: (numeric) money gained from "offer completed"
   - time: (numeric) hours after start of test

2.1 portfolio

There are three types of offers: buy-one-get-one (BOGO), discount, and informational:

- BOGO: a user needs to spend a certain amount to get a reward equal to that threshold.
- Discount: a user gains a reward equal to a fraction of the amount spent.



	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed
3	5	[web, email, mobile]	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9
4	5	[web, email]	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7
5	3	[web, email, mobile, social]	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2
6	2	[web, email, mobile, social]	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4
7	0	[email, mobile, social]	0	3	informational	5a8bc65990b245e5a138643cd4eb9837
8	5	[web, email, mobile, social]	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d
9	2	[web, email, mobile]	10	7	discount	2906b810c7d4411798c6938adc9daaa5

Figure 1: `portfolio` dataframe

- Informational: no reward, no requisite amount that the user is expected to spend.

Offers can be delivered via multiple channels:

- email
- social media
- on the web
- via the Starbucks's app.

Every offer has a validity period (*duration*) before the offer expires. The dataframe does not have any missing value. Features `reward`, `difficulty`, and `duration` have values in different ranges, so I have to scale them to the same range. Figure 2 shows Porfolio data distribution before and after rescaling.

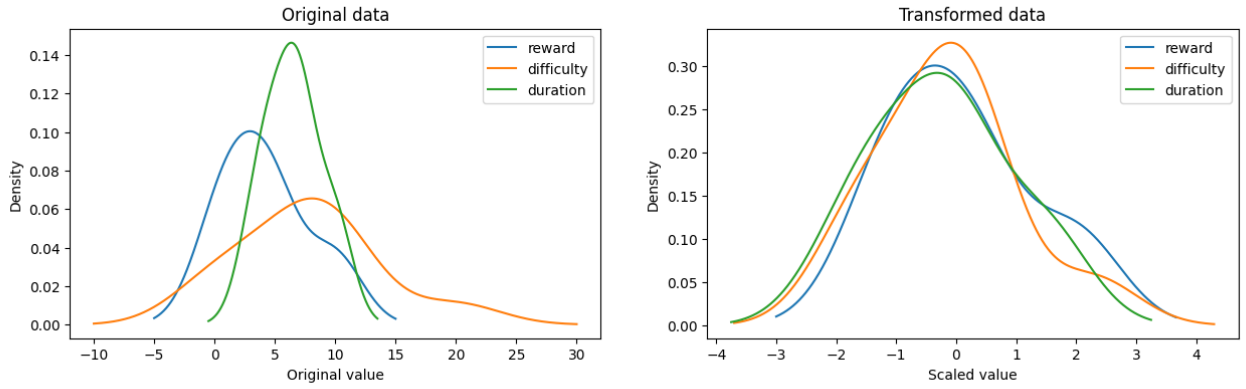
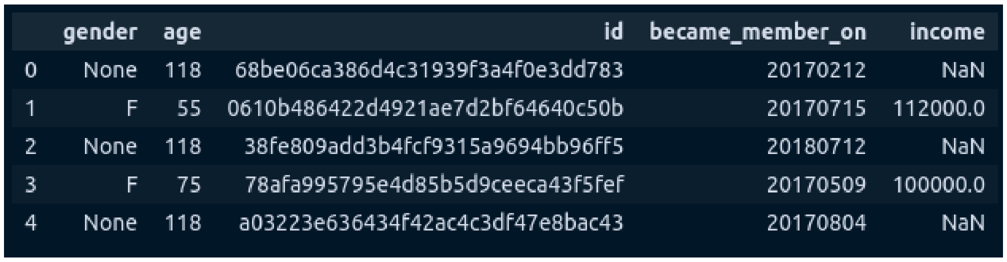


Figure 2: Porfolio data before and after rescaling

2.2 profile

There are empty values in columns `income` and `gender`, and abnormal value 118 in column `age`. It is necessary to verify if these issues are in the same row. The result is `Gender:`



	gender	age	id	became_member_on	income
0	None	118	68be06ca386d4c31939f3a4f0e3dd783	20170212	NaN
1	F	55	0610b486422d4921ae7d2bf64640c50b	20170715	112000.0
2	None	118	38fe809add3b4fcf9315a9694bb96ff5	20180712	NaN
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	20170509	100000.0
4	None	118	a03223e636434f42ac4c3df47e8bac43	20170804	NaN

Figure 3: Head of `profile` dataframe

```
number of NA: 2175
Income:  number of NA: 2175
Age:  missing values:  2175
Gender and income missing in the same rows?:  True
Gender and age missing in the same rows?:  True
Income and age missing in the same rows?:  True
```

So, all the missing values occur in the same rows.

Next, I plot some graphs (Figure 4, 5) to explore the `age` feature.

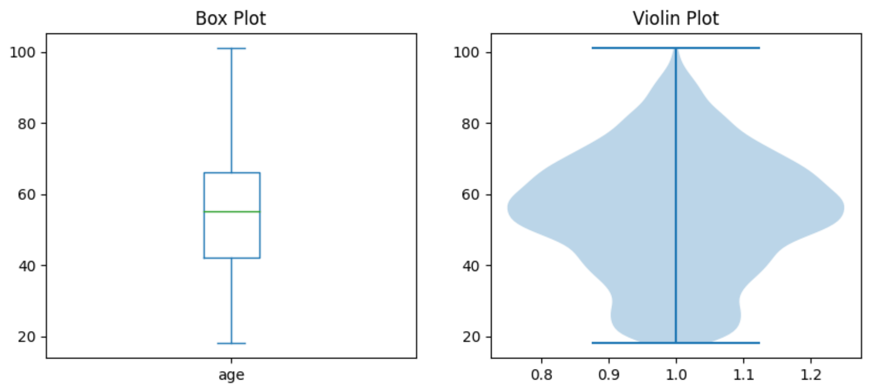


Figure 4: Box plot and Violin plot of `age` feature

The `age` feature is normally distributed among the population, so simple standardization is enough. Figure 6 shows the `age` feature distribution before and after rescaling.

Similarly, Figure 7, 8 show data exploration for the `income` feature. This feature is not well distributed and there is a strong correlation between some ranges of age and income. I also rescale `income` feature, as shown in Figure 9.

Finally, I plot distributions figures for the `became_member_on` feature (Figure 10, 11). This feature deistribution is left-skewed, and should be scaled to a normal distribution. Figure 12 shows the `became_member_on` feature distribution before and after rescaling.

2.3 transcript

This dataset does not have any missing value.

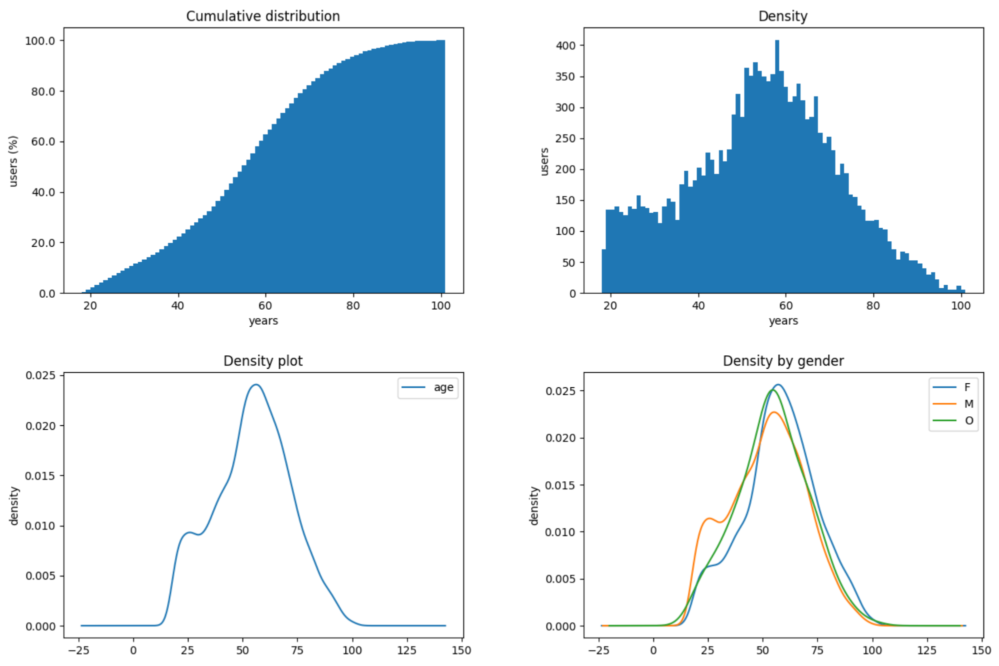


Figure 5: Distributions of the `age` feature

3 Implementation

3.1 Benchmark Model

A Feedforward Neural Network (FNN) will be trained on the same dataset as the RNN to enable comparison of their results. While an FNN analyzes static input without considering customer history, an RNN can make predictions based on past events. A conventional FNN model might repeatedly suggest an offer that once yielded good results, without recognizing when it becomes irrelevant, such as when a customer no longer wants to make the same purchase or is already bought without incentives. The RNN, on the other hand, is designed to capture the relationship between past experiences and future behavior.

3.2 Algorithms and Techniques

This project implements a RNN to analyze offers sent to customers over time. Once a customer receives an offer and makes a purchase, the same offer may no longer be relevant, as the customer might prefer to try a different product. Sending the same offer again might be less effective than offering something new. This temporal relationship makes RNNs ideal for the task, as they are designed to process sequential data by maintaining a hidden state that evolves over time, representing the internal customer state. The neural network models will be developed using the PyTorch framework.

The configuration for the models is as follows:

FNN

- Input: 16 nodes

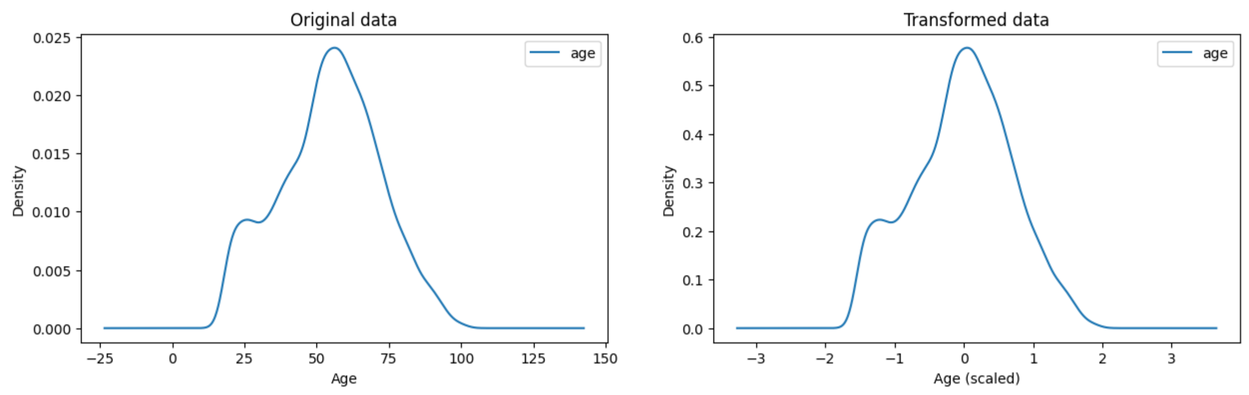


Figure 6: `age` feature before and after rescaling

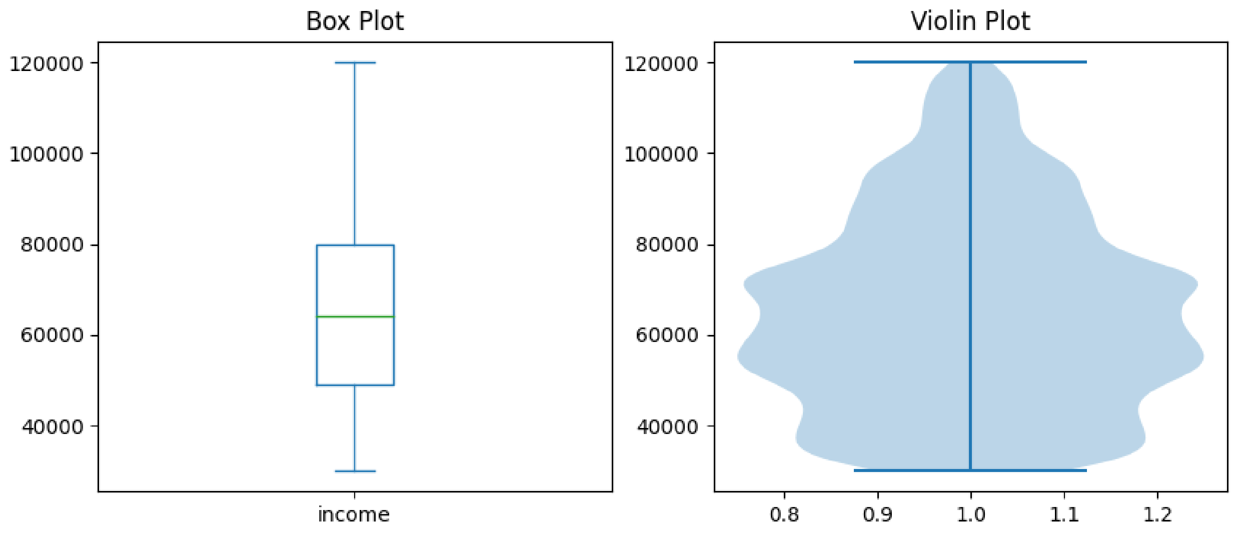


Figure 7: Box plot and Violin plot of `income` feature

- 2 hidden layers, 128 nodes each
- Output: 2 logits values

RNN

- Input: 16 features
- 2 GRU layers, 128 nodes each
- 1 fully-connected layer, 128 nodes
- Output: 2 logits values

The ReLU activation function is used in all hidden layerrs. The optimizer is Adam, with learning rate of 0.001 and weight decay of 0.001.

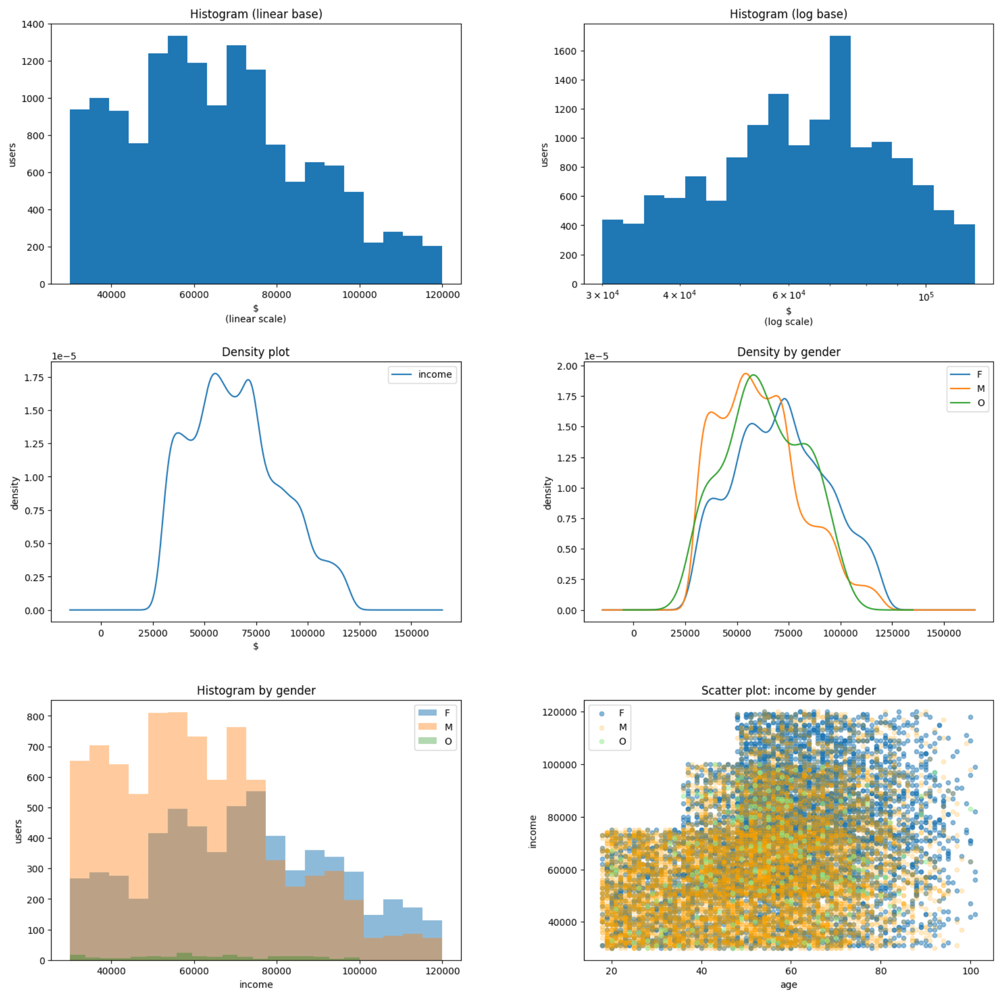


Figure 8: Distributions of the `income` feature

3.3 Refinement

I initially built a recurrent network using PyTorch's RNN module as it was the simplest model to construct and train. With this base model, I could then make adjustments to see if performance improved. The first modification was changing the activation function from TanH to ReLU, which improved accuracy by about 1%, but also caused the model to overfit the training data. Next, I swapped the vanilla RNN for a GRU (Gated Recurrent Unit) and later for an LSTM (Long Short-Term Memory). However, this change yielded similar accuracy across all three models. Both GRU and LSTM models showed a tendency to overfit, with GRU quickly memorizing the training set and diverging on the validation set. To address this, I applied L2-Regularization by adjusting the weight decay parameter in the Adam optimizer to 0.001, which improved generalization, as seen in the following graph.

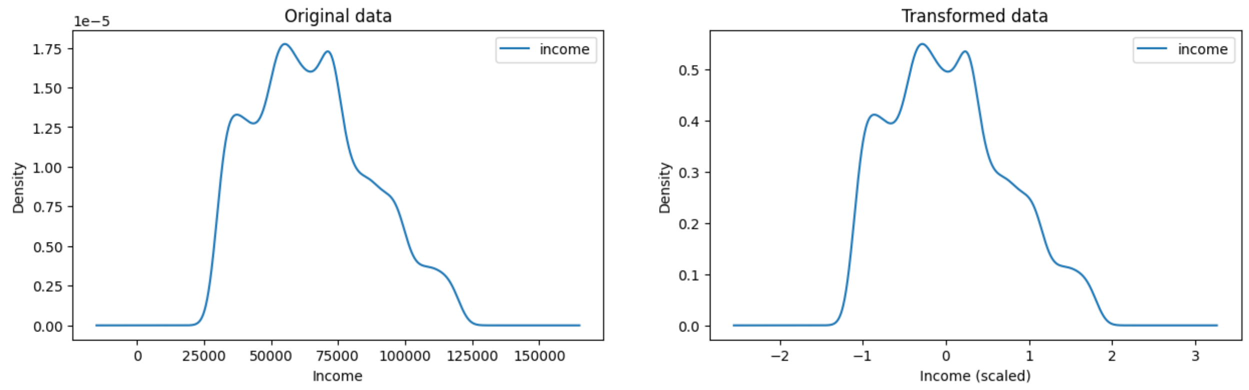


Figure 9: `income` feature before and after rescaling

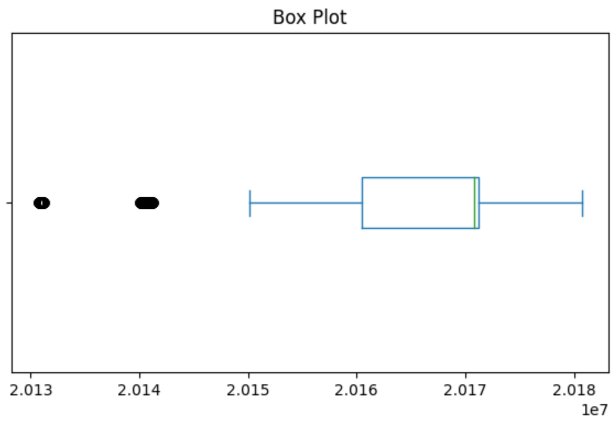


Figure 10: Box plot and Violin plot of `became_member_on` feature

4 Results

Training loss and validation loss of FNN and RNN are shown in Figure 14. Both models tend to converge after a few epochs. Figure 15 and 16 show the results, which were taken from an unseen dataset, proving that both models could generalize well. The final accuracy is about 87%, which is a good value for this project.

Justification

Two different neural networks were trained to recognize patterns in the dataset and predict the likelihood of a customer completing an offer. The benchmark model is a Linear Feed-Forward Network that analyzes static data, focusing on a single customer and offer without considering past events. The proposed model is a Recurrent Neural Network, which processes the same input data but accounts for the customer's history, making the environment dynamic. The results show that both models perform similarly when predicting offer completion, indicating that the Recurrent Network identifies hidden patterns as effectively as the Linear Network and confirming the accuracy of the RNN's predictions for future scenarios.

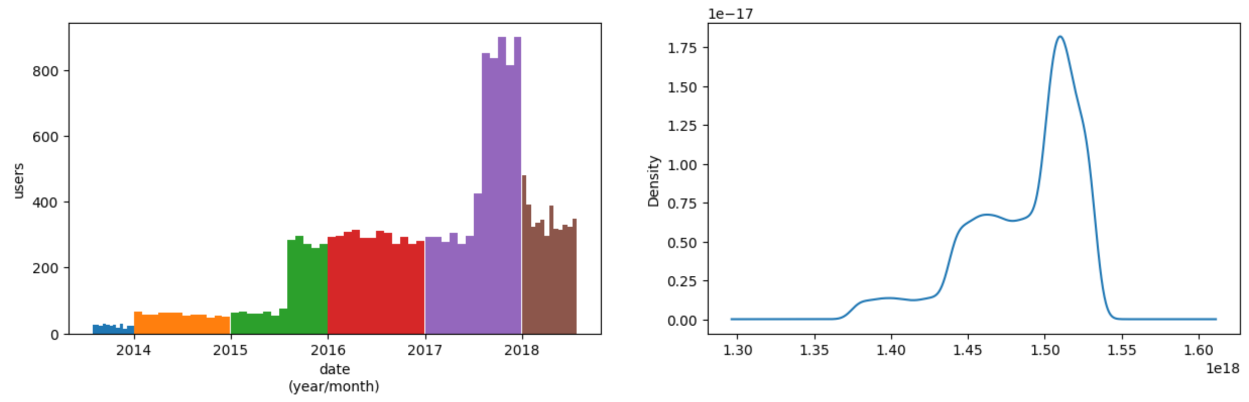


Figure 11: Distributions of the `became_member_on` feature

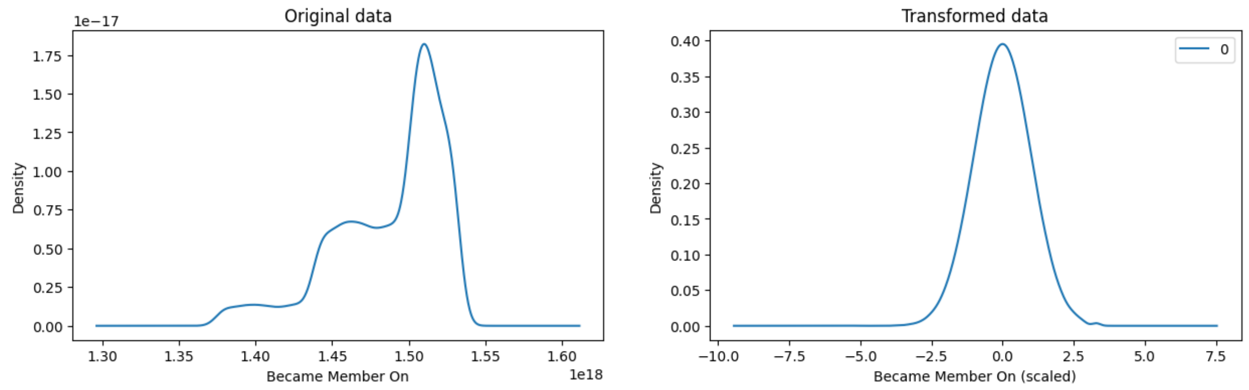


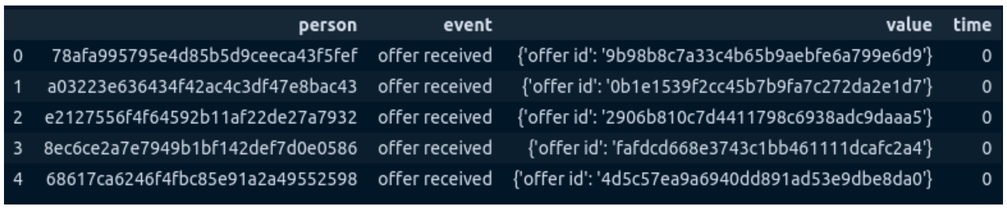
Figure 12: `became_member_on` feature before and after rescaling

5 Conclusion

In this project, I investigate the Starbuck dataset and build a system to predict the most appropriate offer that should be sent to customers. A a linear network would repeatedly suggest the same offer. In contrast, the recurrent model developed in this project takes into account both the unique traits of each customer and their past experiences to recommend the most appropriate offer for the current moment.

References

[1] https://github.com/silviomori/udacity-machine-learning-capstone-starbucks



	person	event	value	time
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0
1	a03223e636434f42ac4c3df47e8bac43	offer received	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	0
2	e2127556f4f64592b11af22de27a7932	offer received	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}	0
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}	0
4	68617ca6246f4fbc85e91a2a49552598	offer received	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}	0

Figure 13: Head of `transcript` dataframe

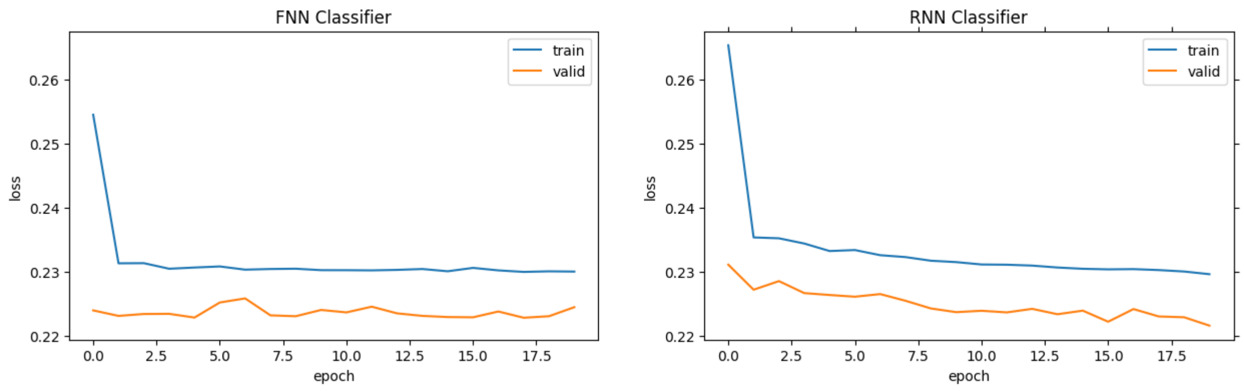
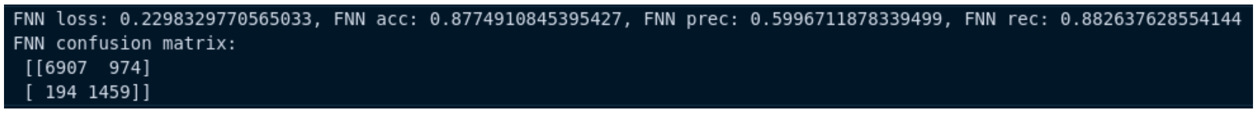
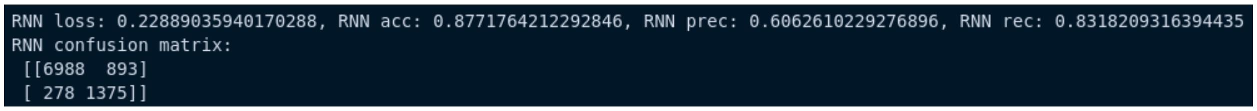


Figure 14: Training loss and validation loss of FNN and RNN.



```
FNN loss: 0.2298329770565033, FNN acc: 0.8774910845395427, FNN prec: 0.5996711878339499, FNN rec: 0.882637628554144
FNN confusion matrix:
 [[6907  974]
 [ 194 1459]]
```

Figure 15: FNN result.



```
RNN loss: 0.22889035940170288, RNN acc: 0.8771764212292846, RNN prec: 0.6062610229276896, RNN rec: 0.8318209316394435
RNN confusion matrix:
 [[6988  893]
 [ 278 1375]]
```

Figure 16: RNN result.