Capstone Project FEYZA ÇOTUK
Data Science Nanodegree AUGUST 24 st , 2024

Starbucks Capstone

Project Project Definition

Project Overview:

This dataset is a simulated representation of customer behavior within the Starbucks Rewards mobile app. Periodically, Starbucks sends various offers to app users, which may include simple advertisements or actual promotions such as discounts or BOGO (Buy One, Get One Free). It's important to note that not every user receives an offer during each promotional period.

A key challenge presented by this dataset is the fact that offers are not uniformly distributed among users, meaning different users may receive different offers. The objective of this project is to analyze the transaction data, along with demographic and offer information, to identify which demographic groups are most responsive to specific types of offers. The dataset is a simplified version of the real-world scenario, focusing on a single product, whereas Starbucks typically offers a wide range of products.

Each offer has a specific validity period before it expires. For instance, a BOGO offer might only be available for 5 days. Even informational offers, which are purely promotional without direct discounts, have a validity period during which the customer may be influenced by the advertisement.

Data Dictionary:

- **profile.json:** Contains data on 17,000 users in the rewards program, with the following fields:
  - **gender:** Categorical variable (M, F, O, or null).
  - **age:** Numeric variable, with missing values encoded as 118.
  - **id:** Unique identifier (string/hash).
- **portfolio.json:** Describes the offers sent during the 30-day test period, with 10 offers and 6 fields:
  - **reward:** Monetary reward for spending a certain amount.
  - **channels:** List of channels through which the offer was delivered (web, email, mobile, social).
  - **difficulty:** Amount that needs to be spent to receive the reward.
  - **duration:** Number of days the offer is valid.
  - **offer_type:** Type of offer (BOGO, discount, informational).
  - **id:** Unique identifier for the offer (string/hash).
- **transcript.json:** Logs the events related to the offers, containing 306,648 entries and 4 fields:
  - **person:** Unique identifier for the user (string/hash).
  - **event:** Type of event (offer received, offer viewed, transaction, offer completed).

    - **value:** A dictionary with various values depending on the event type, such as offer ID or amount spent.
    - **time:** Time elapsed since the start of the test period (in hours).

Offer Types:

There are three main types of offers:

- **Buy-One-Get-One (BOGO):** Requires the user to spend a certain amount to receive a reward equivalent to that amount.
- **Discount:** Provides a reward that is a fraction of the amount spent.
- **Informational:** These offers do not provide a monetary reward but aim to inform the user about a product or service. Despite having no spending requirement, they still have an influence period.

Offers can be delivered through multiple channels, including web, email, mobile, and social media.

Problem Statement:

The goal is to predict which purchase offer is most likely to elicit a higher level of user engagement, such as actions like ‘offer received’, ‘offer viewed’, ‘transaction’, and ‘offer completed’. This prediction is based on the demographic characteristics of customers and the specific attributes of the company’s purchase offers.

Evaluation Metrics:

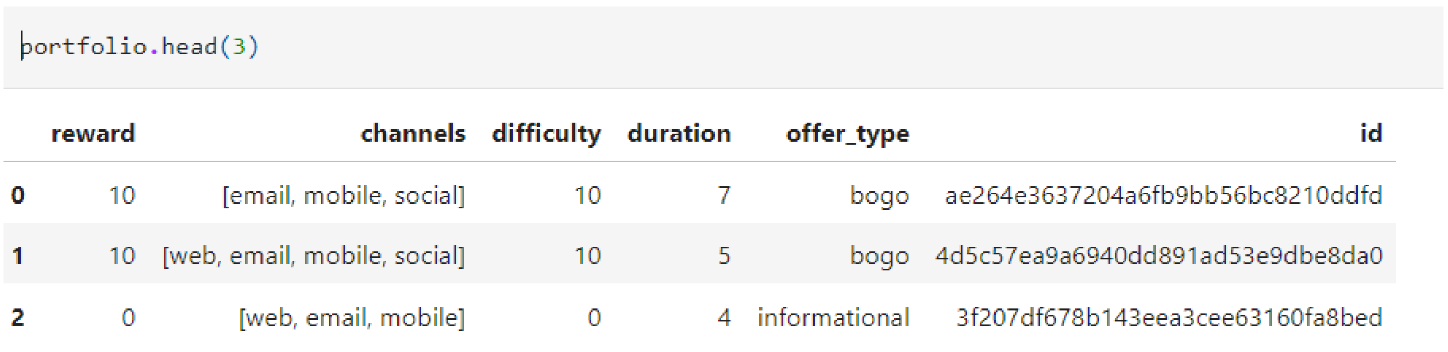
Accuracy is the primary metric for evaluating the performance of this classification model. It is applicable to both binary and multiclass classification problems and is calculated as follows:

Accuracy = (TP+TN)/(TP+FP+FN+TN)

where TP stands for True Positives, TN for True Negatives, FP for False Positives, and FN for False Negatives. Accuracy represents the proportion of correct predictions out of the total number of cases evaluated.

Data Access and Cleaning:

1. **Portfolio Data:**
    - Create a duplicate of the original DataFrame to preserve the original data for future use.
    - Transform the 'Channels' column into four distinct columns, each representing a different type of channel.
    - Rename the 'ID' column to 'offer_id' for clarity.

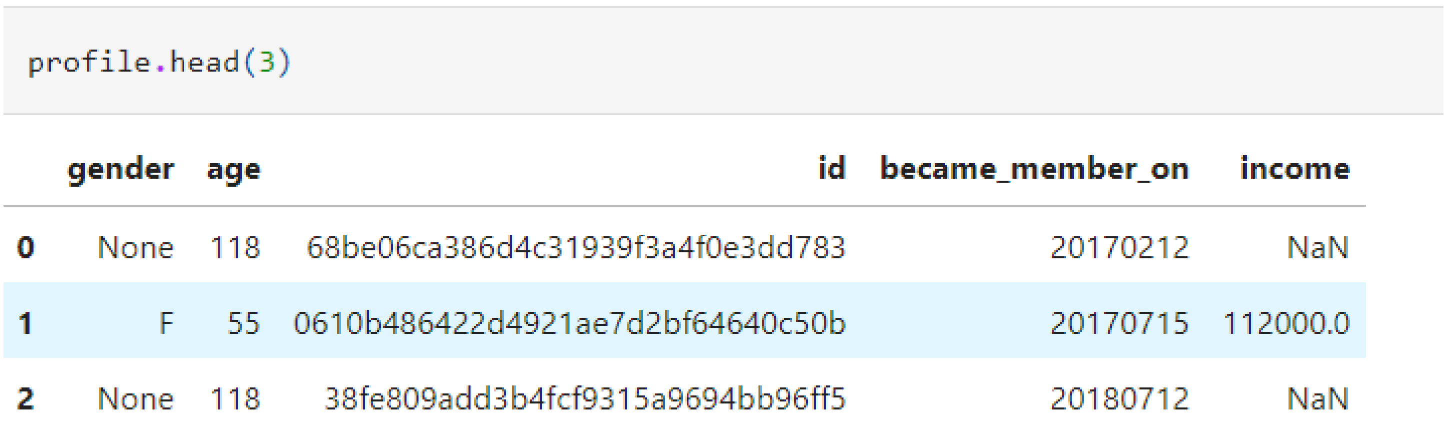


```
portfolio.head(3)
```

	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

Profile Data:

- Create a duplicate of the original DataFrame to ensure the integrity of the initial data.
- Convert the datatype of the 'became_member_on' column and format the dates correctly.
- Rename the 'ID' column to 'customer_id' for consistency and clarity.

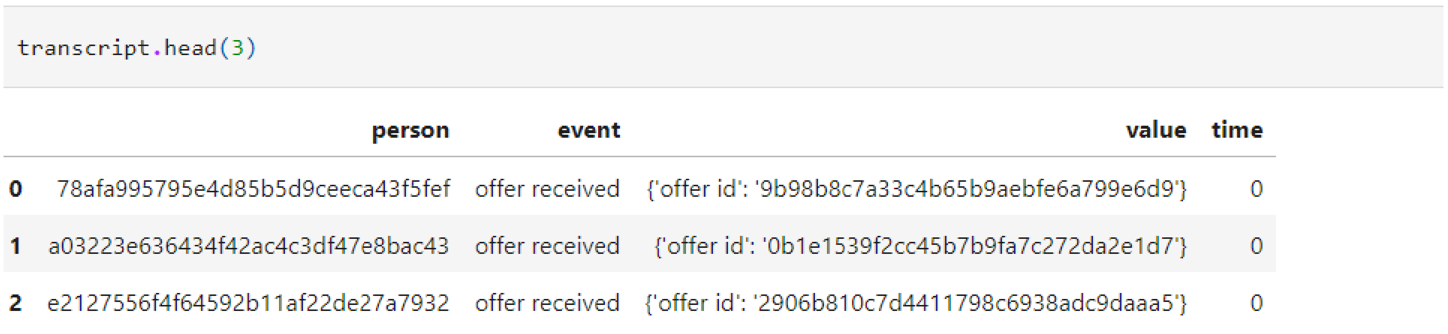


```
profile.head(3)
```

	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

Transcript Data:

- Create a duplicate of the original DataFrame to maintain the integrity of the initial data.
- Rename the 'person' column to 'customer_id' for consistency.
- Transform the 'Event' column into four separate columns, each representing a different type of event.
- Split the 'Values' column into two distinct columns based on its contents.



```
transcript.head(3)
```

	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

Data Cleaning:

- Combine all three datasets into a single cohesive dataset.

- Correct any discrepancies in the offer_id values.
- Resolve any issues related to event_id values.

	customer_id	event	time	offer-completed	offer-received	offer-viewed	transaction	offer_id	amount	gender	...	income
0	78afa995795e4d85b5d9ceeca43f5fef	offer-received	0	False	True	False	False	0.0	NaN	F	...	100000.0
1	a03223e636434f42ac4c3df47e8bac43	offer-received	0	False	True	False	False	1.0	NaN	None	...	NaN
2	e2127556f4f64592b11af22de27a7932	offer-received	0	False	True	False	False	2.0	NaN	M	...	70000.0
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer-received	0	False	True	False	False	3.0	NaN	None	...	NaN
4	68617ca6246f4fbc85e91a2a49552598	offer-received	0	False	True	False	False	4.0	NaN	None	...	NaN

5 rows × 22 columns

```
df_all.columns
```

```
Index(['customer_id', 'event', 'time', 'offer-completed', 'offer-received',
       'offer-viewed', 'transaction', 'offer_id', 'amount', 'gender', 'age',
       'became_member_on', 'income', 'reward', 'difficulty', 'duration',
       'offer_type', 'email', 'mobile', 'social', 'web', 'event_id'],
      dtype='object')
```

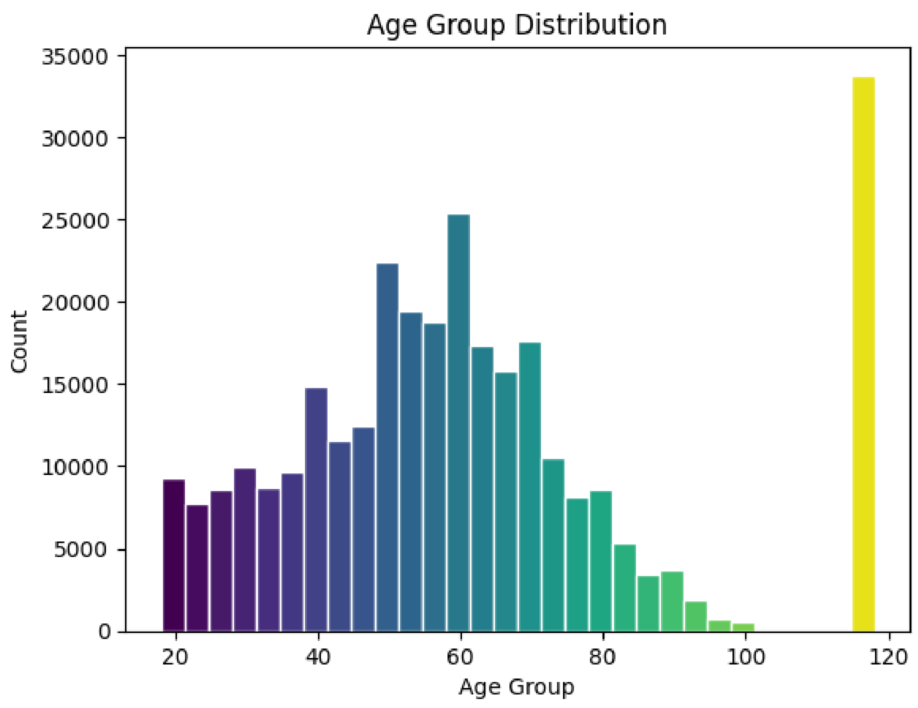
Data Analysis

Data Exploration and Visualization:

1. **Age Group:**
    - **Observation:**
        - There are outliers present, particularly users with an age greater than 115, which is unrealistic and indicates possible data entry errors.
        - The average age of users tends to be within the middle-aged range, approximately 50 to 62 years.

```
data.age.describe()
```

```
count    306534.000000
mean         60.909367
std          26.032030
min          18.000000
25%          43.000000
50%          57.000000
75%          72.000000
max         118.000000
Name: age, dtype: float64
```

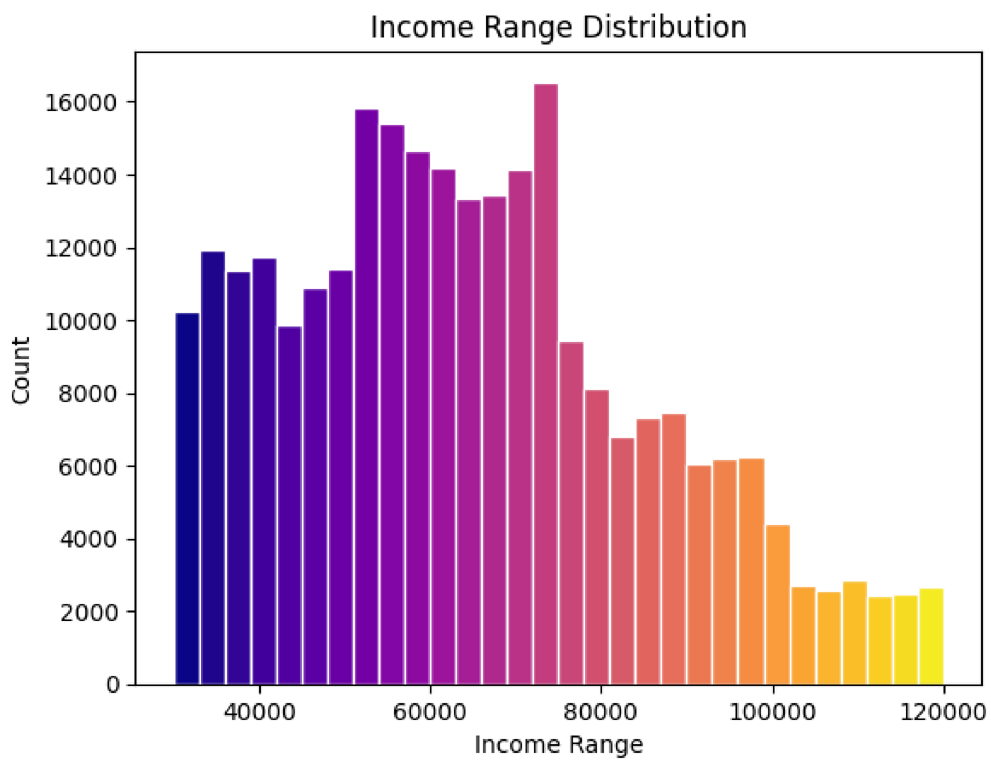


2.Income Range:

- **Observation:**
    - The average income of users falls within the middle-income group, typically ranging between $65,000 and $70,000.

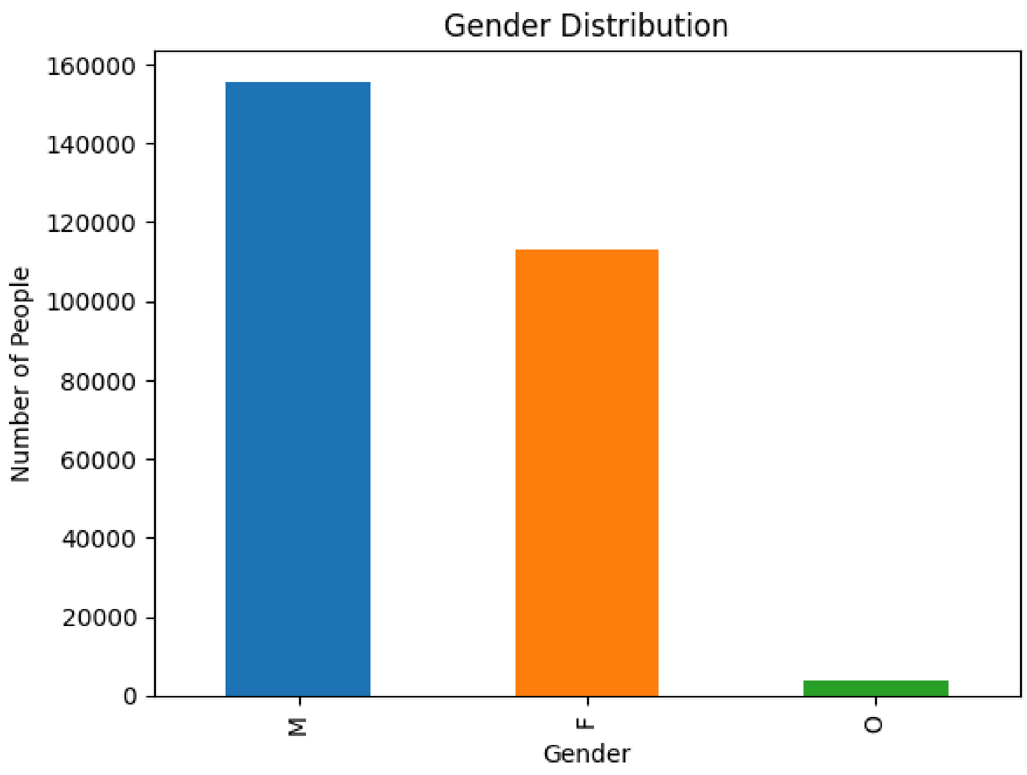
```
data.income.describe()
```

```
count    272762.000000
mean      64337.000755
std       21243.762941
min       30000.000000
25%       48000.000000
50%       62000.000000
75%       78000.000000
max      120000.000000
Name: income, dtype: float64
```



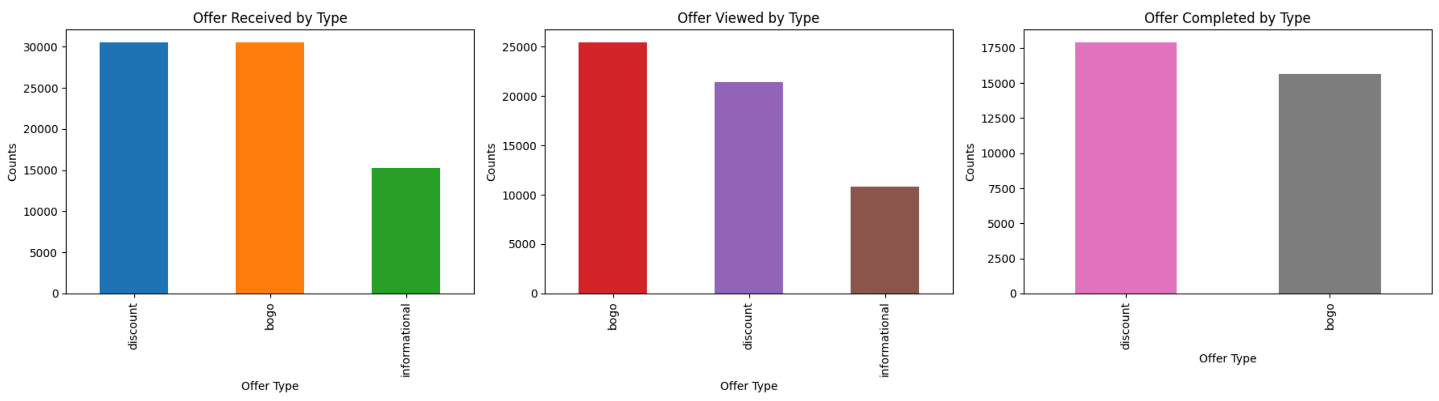
3.Gender:

- **Observation:**
  - Males constitute slightly more than 50% of the user base.
    - **Male proportion:** 50.79%
    - **Female proportion:** 36.90%
    - **Other:** 1.30%



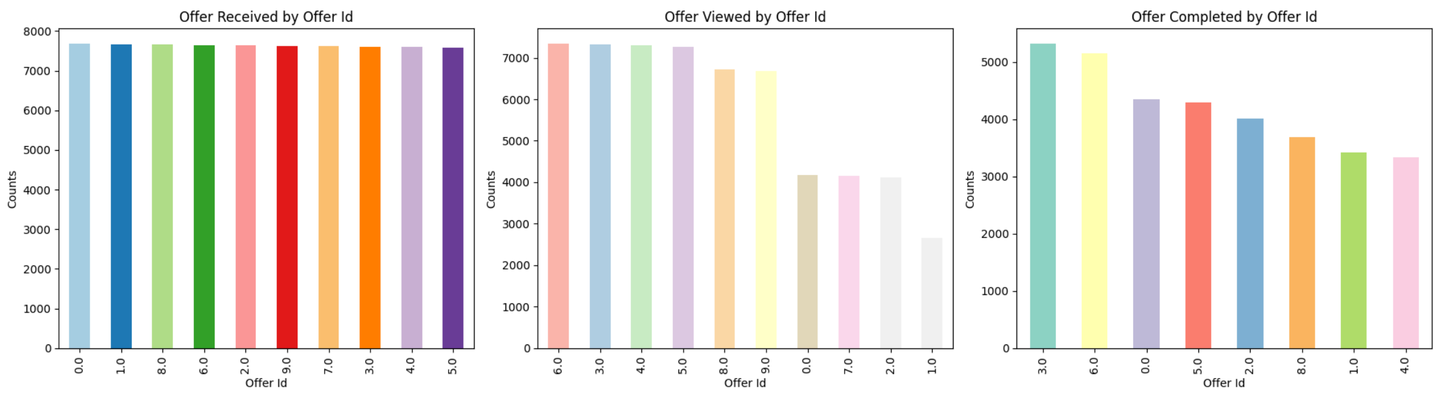
4.Number of Different Types of Offers Received by Users:

- **Observation:**
  - **BOGO Offers:**These are the most popular, with 30,499 users receiving them. Of those, 25,449 viewed the offer, and 15,669 completed it.
    - **Offer Viewed Proportion:** 83.44%
    - **Offer Completed Proportion:** 51.37%
  - **Discount Offers:** The percentage of users who viewed the offer is 70%.



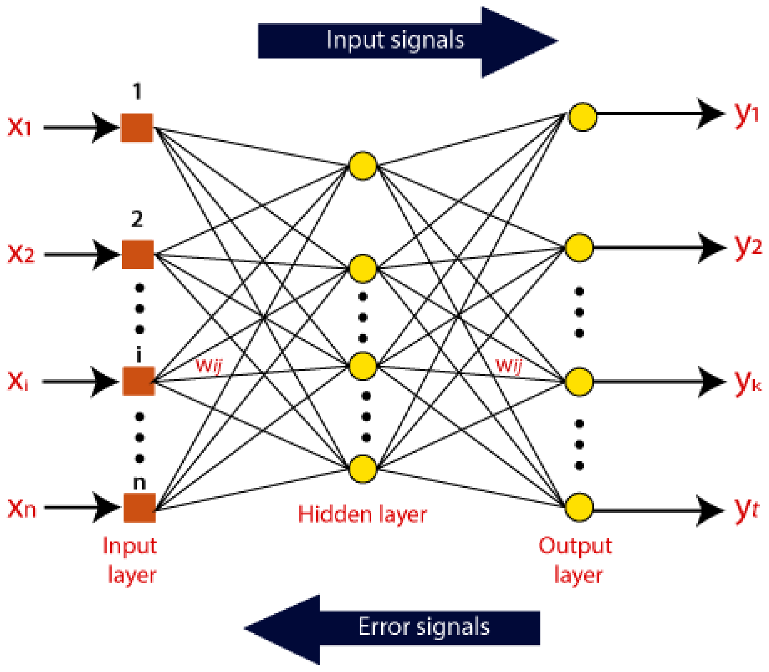
5.Number of Offer IDs Receiving Different Types of Offers:

- **Observation:**
  - Each offer ID received an approximately equal number of offers.
  - The viewing ratio decreased for certain offer IDs, specifically for IDs 0, 6, 7, and 5.
  - The offer completion rate is relatively satisfactory across the board.



Algorithms and Techniques:

The chosen classifier for this task is an Artificial Neural Network (ANN). An ANN is a collection of algorithms designed to identify and learn underlying patterns in data by emulating the functioning of the human brain. These networks consist of interconnected nodes, or "neurons," which can be either organic or artificial. One of the key strengths of neural networks is their adaptability to changing inputs, allowing the model to produce the most accurate results without requiring a redesign of the output criteria.



An Artificial Neural Network (ANN) is a component of a computing system created to replicate how the human brain analyzes and processes information. It serves as a cornerstone of Artificial Intelligence (AI) and is capable of solving complex problems that are challenging or infeasible by traditional human or statistical methods. ANNs possess self-learning capabilities, which allow them to improve their performance as they are exposed to more data over time.

Inputs:

Source data is fed into the neural network with the aim of making decisions or predictions based on the data. Typically, inputs to a neural network consist of a set of real values, each of which is fed into one of the neurons in the input layer.

Training Set:

A collection of inputs for which the correct outputs are known, used to train the neural network.

Outputs:

Neural networks produce predictions as a set of real values or boolean decisions. Each output value is generated by one of the neurons in the output layer.

Neuron/Perceptron:

The neuron, or perceptron, is the fundamental unit of a neural network. It receives input, processes it through an activation function, and generates a prediction. Common activation functions include Sigmoid, TanH, and ReLU. These functions help ensure that output values fall within an acceptable range, and their non-linear nature is essential for training the network effectively.

Error Function:

The error function measures how far the actual output of the current model is from the correct output. The goal during model training is to minimize the error function, bringing the output as close as possible to the correct value.

Hyperparameters:

A hyperparameter is a setting that influences the architecture or behavior of a neural network. In deep learning projects, tuning hyperparameters is crucial for developing a network that provides accurate predictions for a specific problem. Common hyperparameters include the number of hidden layers, the choice of activation function, and the number of epochs, or how many times the training process is repeated.

Methodology

Data Preprocessing:

One-Hot Encoding: One-hot encoding is applied to transform categorical columns like Gender and Offer_type into numerical format before model preparation. This involves creating binary vectors for each category, enabling the model to process these features effectively.

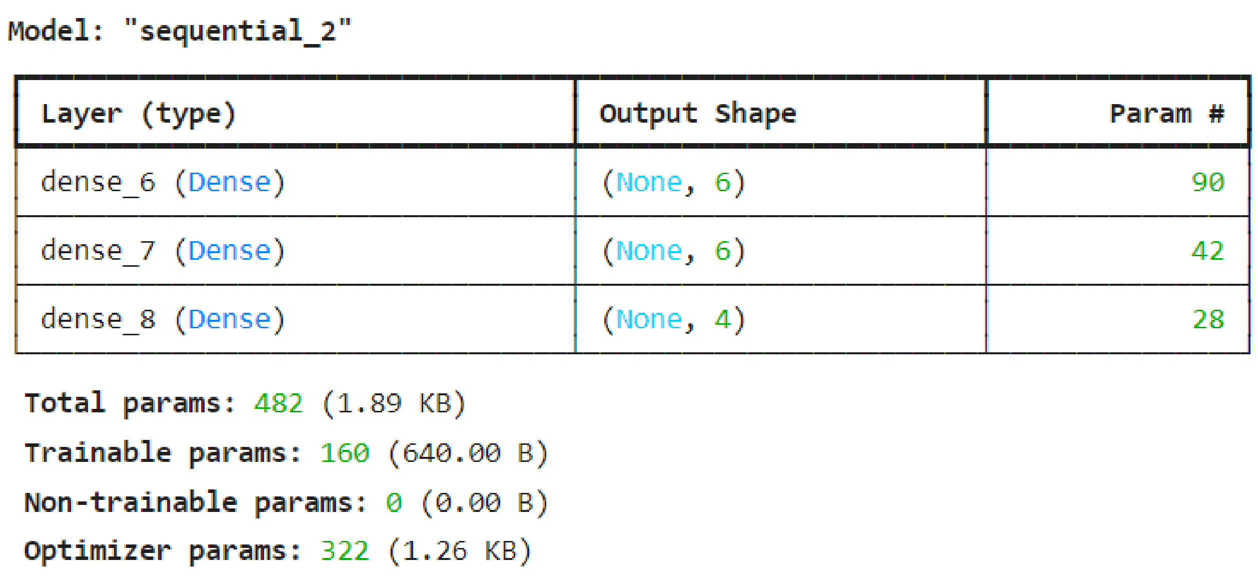
Dataset Splitting: The train_test_split function is used to divide the dataset into two parts: one for training the model and the other for testing it. The training portion is utilized to build the model, while the testing portion is reserved for evaluating the model's performance on unseen data, ensuring it can generalize well.

Feature Scaling:

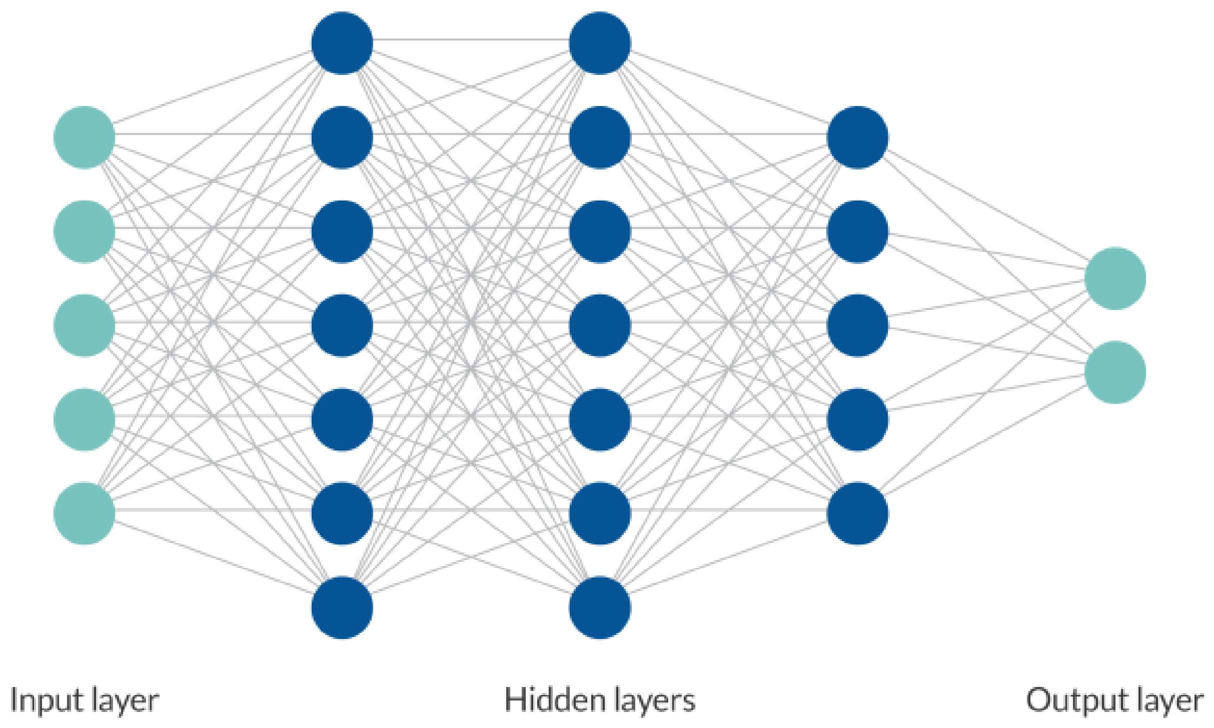
- **Normalization:** This technique adjusts the data so that the values range between 0 and 1. It is also known as Min-Max scaling and is useful when data does not follow a normal distribution.
- **Standardization:** This method involves rescaling the data such that it has a mean of 0 and a standard deviation of 1, which is particularly helpful when the data is normally distributed. This ensures that all features contribute equally to the model.

Converting Pandas DataFrame to NumPy Array: A Pandas DataFrame is a versatile two-dimensional data structure with labeled axes, which can be converted into a NumPy ndarray using the DataFrame.to_numpy() method. This conversion is beneficial for performing efficient numerical operations, especially when working with machine learning algorithms.

Implementation : Build a Model 1 :



```
Model: "sequential_2"
┌──────────────────────────────┬───────────────────────┬──────────────┐
│ Layer (type)                 │ Output Shape          │      Param # │
├──────────────────────────────┼───────────────────────┼──────────────┤
│ dense_6 (Dense)              │ (None, 6)             │           90 │
├──────────────────────────────┼───────────────────────┼──────────────┤
│ dense_7 (Dense)              │ (None, 6)             │           42 │
├──────────────────────────────┼───────────────────────┼──────────────┤
│ dense_8 (Dense)              │ (None, 4)             │           28 │
└──────────────────────────────┴───────────────────────┴──────────────┘
 Total params: 482 (1.89 KB)
 Trainable params: 160 (640.00 B)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 322 (1.26 KB)
```



Activation Function:

- Layer 1 and Layer 2 (Input Layer and Hidden Layer):
  The ReLU (Rectified Linear Unit) activation function is applied. ReLU is computationally efficient and commonly used in deep learning models. However, it has a limitation in that it cannot process inputs that are close to zero or negative values effectively.
- Layer 3 (Output Layer):
  The Softmax activation function is used in the output layer. Softmax normalizes the output values between 0 and 1 for each class, providing the probabilities that the input belongs to each specific class.

Optimizer:

- Adam:
  The Adam optimizer is utilized for training this model. Adam is an adaptive learning rate

optimization algorithm that combines the advantages of the AdaGrad and RMSProp algorithms, making it particularly effective in handling sparse gradients and noisy data.

Loss Function:

- Sparse Categorical Cross Entropy:
  The loss function used is sparse categorical cross-entropy, which is ideal for classification problems where each sample belongs to a single class. This loss function is chosen when the classes are mutually exclusive, ensuring that the model is penalized for incorrect predictions.

Refinement:

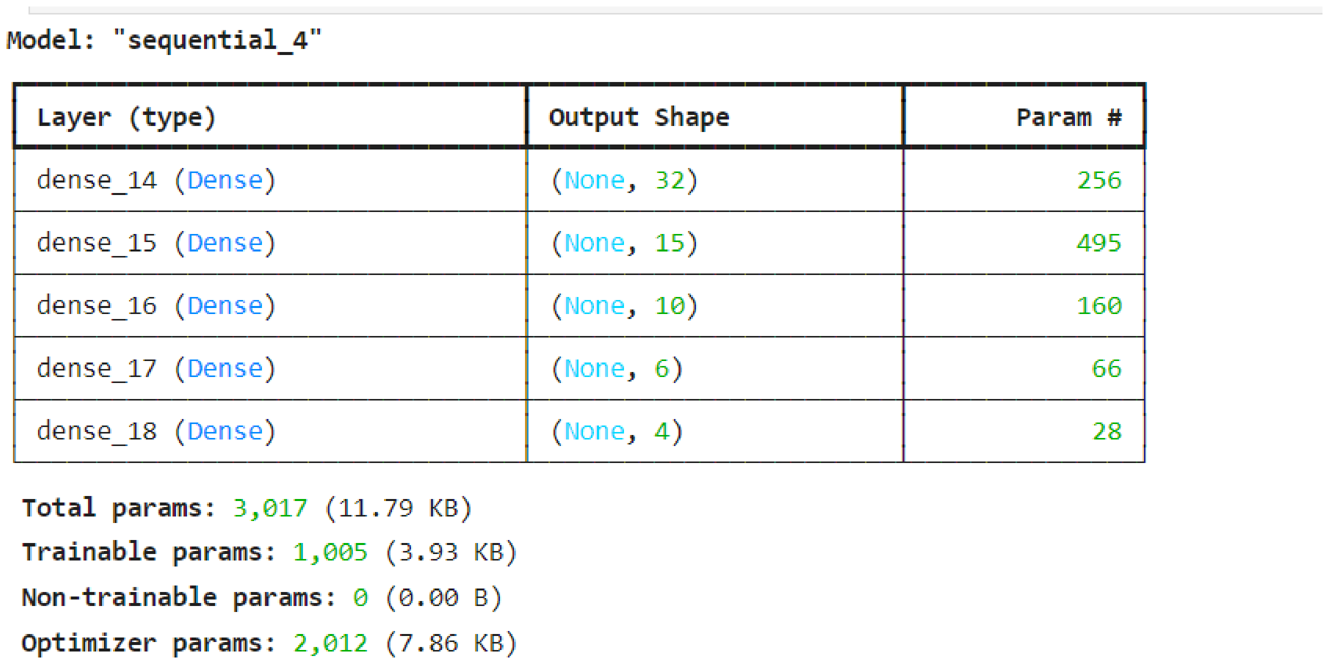
Improving the Prediction Model:

The current model is experiencing underfitting, indicated by high bias.

- **Bias:**
  Bias refers to how closely our predictive model's average predictions align with the training data. Algorithms with high bias tend to learn quickly and are easier to understand but lack flexibility. This limitation makes them less effective at predicting complex problems, leading to underfitting in the model.

How to Overcome Underfitting:

1. **Use a Larger Network:**
   - **More Hidden Layers:** Increasing the number of hidden layers can help the model capture more complex patterns in the data.
   - **More Hidden Units:** Adding more units within each hidden layer can provide the model with additional capacity to learn.
2. **Train the Model for a Longer Period:**
   - Extending the training duration allows the model more time to learn from the data, potentially improving its performance.
3. **Utilize Advanced Optimization Algorithms:**
   - Employing more sophisticated optimization algorithms can help the model converge to a better solution, reducing underfitting.



```
Model: "sequential_4"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ dense_14 (Dense)                │ (None, 32)             │           256 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_15 (Dense)                │ (None, 15)             │           495 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_16 (Dense)                │ (None, 10)             │           160 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_17 (Dense)                │ (None, 6)              │            66 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_18 (Dense)                │ (None, 4)              │            28 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 3,017 (11.79 KB)
 Trainable params: 1,005 (3.93 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 2,012 (7.86 KB)
```

Results

Model Evaluation and Validation:

During the development process, a validation set was utilized to continuously evaluate the model's performance. The final architecture and hyperparameters were selected based on their superior performance during this evaluation phase.

Justification:

In the refinement step, several techniques were implemented to address the issue of underfitting:

1. **Kernel Initializer:**
   - **Normal Initialization:** The neural network requires an initial set of weights, which are iteratively updated to improve the model. The kernel_initializer parameter specifies the statistical distribution or function used to initialize these weights. In this case, weights were initialized using a normal distribution. Other options, such as constants (like ones or zeros) or different distributions (e.g., uniform), could also be used.
2. **Increased the Number of Hidden Layers:**
   - Adding more hidden layers allows the network to capture more complex patterns in the data, thereby improving its ability to make accurate predictions.
3. **Increased the Number of Hidden Units:**
   - Increasing the number of units within each hidden layer provides the network with additional capacity to learn from the data, which can lead to better performance.

Training Process:

During training, the model underwent 15 epochs. Despite these efforts, the model's accuracy plateaued at 45.38%, with a validation accuracy of 45.22%. The loss remained consistent throughout the epochs, indicating that further adjustments might be necessary to improve performance.

Training and Validation Metrics Over 15 Epochs:

- **Epoch 1/15:**
  - Training Accuracy: 45.37%
  - Training Loss: 1.2777
  - Validation Accuracy: 45.22%
  - Validation Loss: 1.2629
- **Epoch 2/15 to 15/15:**
  - Training Accuracy: 45.38%
  - Training Loss: 1.2609
  - Validation Accuracy: 45.22%
  - Validation Loss: 1.2629

This consistency across epochs suggests that while the model is stable, it may require further refinement or different techniques to overcome its limitations and achieve better accuracy.

Conclusion

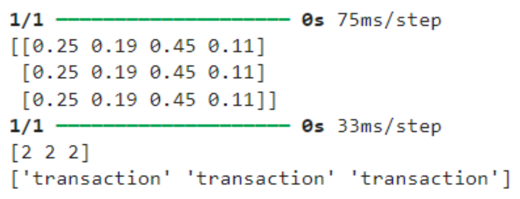
Reflection:

- I found this project to be particularly challenging, especially due to the complexity of the data structure in the transcript dataset.
- The model demonstrated better performance on the majority classes, but it struggled with the minority classes, highlighting an issue with imbalanced data.
- A significant number of events were incorrectly predicted as 'offer received', likely because it is the most frequent event in the dataset.
- My primary objective was to develop a model that could provide practical insights for the company, helping them make more informed decisions.
- However, the model's performance was not as strong as I had hoped. The accuracy remained relatively constant, with a final test accuracy of approximately 45.22% and a loss of 1.2612, indicating that further refinement is needed.

Model Evaluation:

- **Test Accuracy:** 45.43%
- **Test Loss:** 1.2612

When predicting on new data, the model consistently predicted the class 'transaction' for the examples provided, as shown by the following probabilities and predicted classes:



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 75ms/step
[[0.25 0.19 0.45 0.11]
 [0.25 0.19 0.45 0.11]
 [0.25 0.19 0.45 0.11]]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
[2 2 2]
['transaction' 'transaction' 'transaction']
```

These results indicate that all examples were classified as 'transaction', which may reflect the model's bias towards this class, further underscoring the challenge of imbalanced data.

Improvement

- **Dealing with Imbalanced Data:**
  Given that the dataset is imbalanced, achieving high accuracy was challenging. The model's performance could be improved by addressing this imbalance, potentially through techniques like oversampling the minority classes or employing more sophisticated methods like deep neural networks or recommendation engines.
- **Accuracy Enhancement:**
  The accuracy of the model could be significantly increased by implementing deeper neural networks, which have the capacity to capture more complex patterns, or by exploring recommendation systems that tailor predictions to individual users.
- **Model Development Challenges:**
  The main challenge was in analyzing the data and constructing the models. This aspect requires further refinement to improve both the model's predictive performance and its practical application.

References:

- Pandas DataFrame Replace Documentation
- Pandas Merge Documentation
- Splitting Columns in Pandas
- https://www.tensorflow.org/tutorials/keras/classification?hl=tr

The added layer must be an instance of class Layer...

I am new to machine learning. ...

stackoverflow.com

-

THANK YOU