Udacity

Project: Capstone Project Starbucks

By: Ibrahim Bonzo

Contents

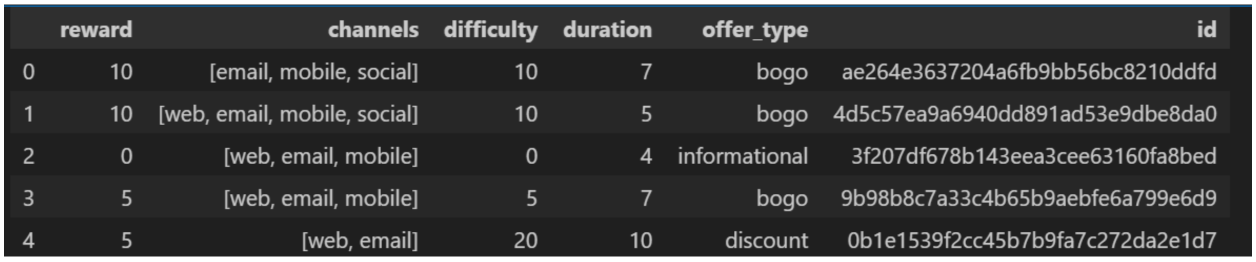
Introduction:

This project uses a Starbucks dataset to create personalized suggestions based on how well various offers work in an effort to increase sales. Businesses can optimize revenue production by effectively allocating resources and customizing promotions by identifying which offers provide the biggest gains. This is an important effort that will increase profitability and improve customer happiness by using targeted marketing strategies.

Description of Input Data:

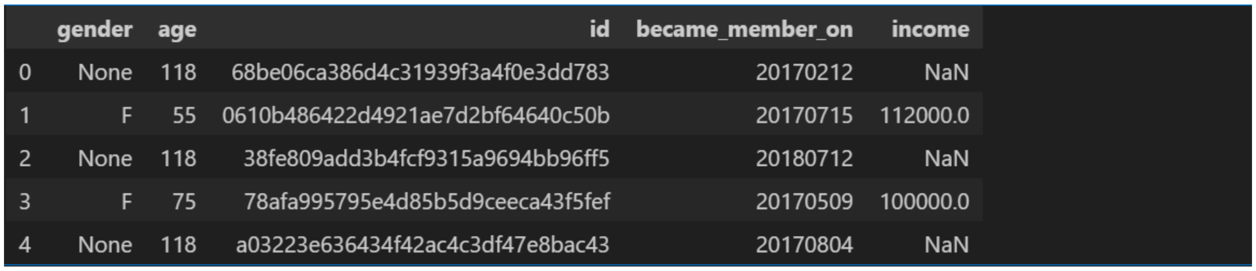
The portfolio.json file contains metadata about each offer, including offer IDs, offer types (e.g., BOGO, discount), minimum required spend (difficulty), reward, duration of the offer, and channels through which the offer is distributed. The profile.json file provides demographic data for each customer, such as age, gender, customer ID, income, and the date when the customer created an app account. The transcript.json file records various events, including transactions, offers received, offers viewed, and offers completed. Each record includes details such as event type, customer ID, time of the event, and value (either an offer ID or transaction amount).

Portfolio



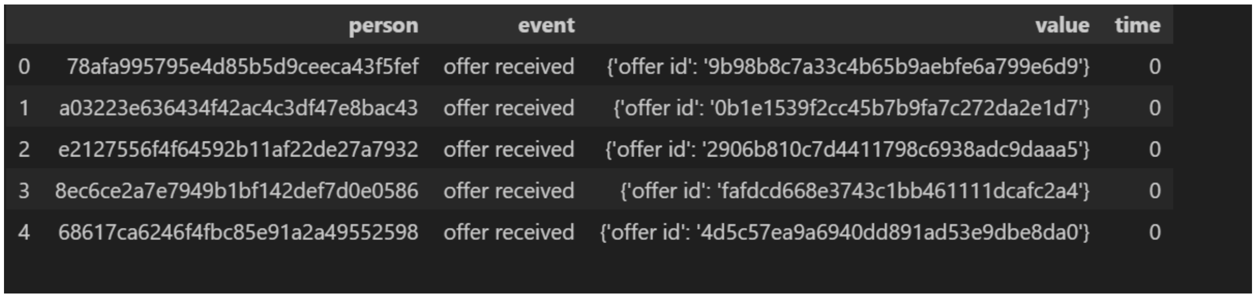
	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed
3	5	[web, email, mobile]	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9
4	5	[web, email]	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7

Profile



	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

Transcript



	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

Strategy for solving the problem:

Leverage the dataset to determine the most effective offers to send to individual customers by addressing two key questions:

1. Which offer prompts increased purchasing from a specific customer?
2. Which demographic segments are most responsive to offer types?

Through advanced analytics techniques, including segmentation analysis and predictive modeling, we will tailor recommendations based on customer purchasing history, demographics, and offer preferences. This engine seeks to optimize Starbucks' promotional efforts, driving sales and enhancing customer engagement.

Metrics with justification

Using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) as evaluation metrics is also appropriate, considering specific context of Starbucks project.

Mean Absolute Error (MAE):

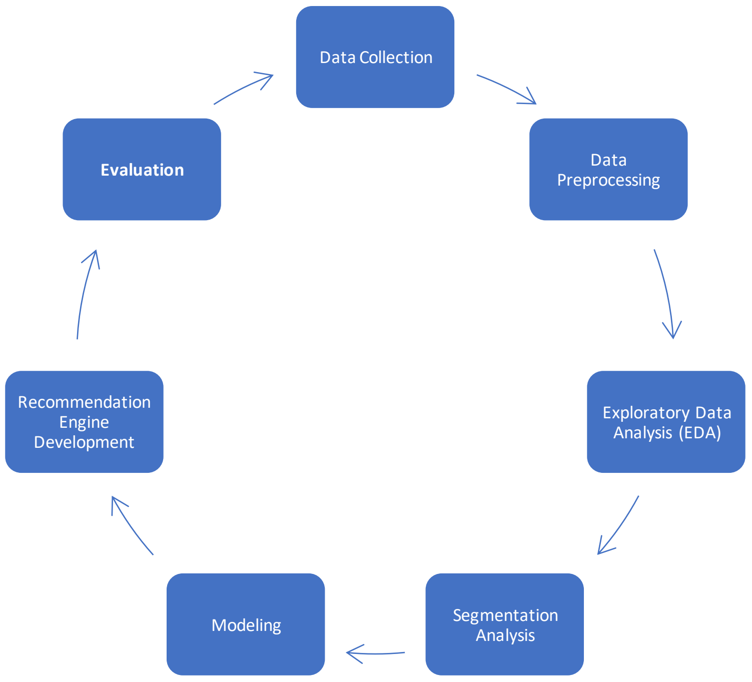
MAE measures the average magnitude of errors between predicted and actual values. In the context of recommendation engine in Starbucks project, MAE can quantify the average difference between the predicted response (purchase behavior after receiving an offer) and the actual response. MAE provides a straightforward interpretation of the magnitude of errors, making it easy to understand the average deviation of the model's predictions from the true values. By using MAE, you aim to minimize the absolute difference between predicted and actual responses, ensuring that the recommendation engine provides accurate estimates of customer behavior in response to offers.

Root Mean Squared Error (RMSE):

RMSE is a commonly used metric that measures the square root of the average of squared differences between predicted and actual values. In this recommendation engine, RMSE can quantify the overall accuracy of the model's predictions by penalizing larger errors more heavily than smaller ones. RMSE provides a comprehensive measure of prediction accuracy, considering both the bias and variance of the model's predictions. By minimizing RMSE, contribute to reduce the overall deviation of the recommendation engine's predictions from the true values, ensuring that the model provides reliable estimates of customer behavior in response to offers.

Expected solution.

Below figure shows steps to achieve results



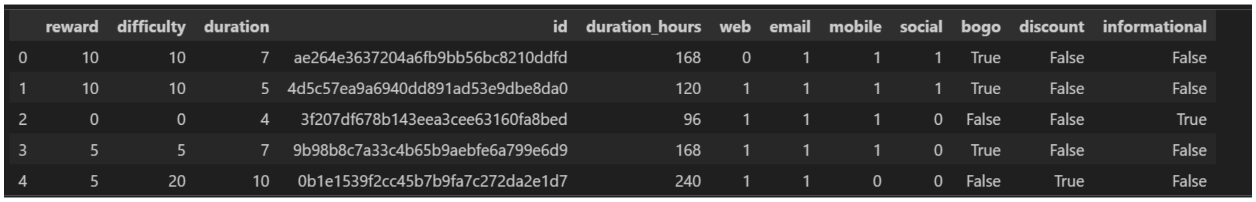
Exploratory Data Analysis (EDA):

During EDA, we will visualize and analyze the dataset to uncover patterns, trends, and insights that can inform the development of the recommendation engine. Key findings may include:

Data cleaning

Clean and preprocess the portfolio DataFrame. Args:portfolio: DataFrame containing portfolio data to be cleaned. Returns: portfolio: DataFrame with cleaned portfolio data.

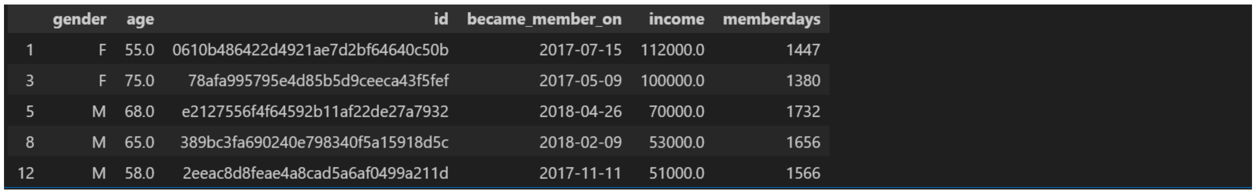
- Create a copy of the DataFrame
- Convert duration from days to hours
- Apply one-hot encoding to the channels column
- Apply one-hot encoding to the offer_type column
- Drop the channels and offer_type columns
- Merge the DataFrame with one-hot encoded offer types



	reward	difficulty	duration	id	duration_hours	web	email	mobile	social	bogo	discount	informational
0	10	10	7	ae264e3637204a6fb9bb56bc8210ddfd	168	0	1	1	1	True	False	False
1	10	10	5	4d5c57ea9a6940dd891ad53e9dbe8da0	120	1	1	1	1	True	False	False
2	0	0	4	3f207df678b143eea3cee63160fa8bed	96	1	1	1	0	False	False	True
3	5	5	7	9b98b8c7a33c4b65b9aebfe6a799e6d9	168	1	1	1	0	True	False	False
4	5	20	10	0b1e1539f2cc45b7b9fa7c272da2e1d7	240	1	1	0	0	False	True	False

Cleanse the profile DataFrame. Args:profile: DataFrame to be cleansed.Returns:cleaned_profile: The sanitized DataFrame.

- Replace missing values encoded as 118 with NaN
- Drop rows with missing values
- Convert 'became_member_on' column to datetime format
- Calculate the number of days since the user became a member of Starbucks

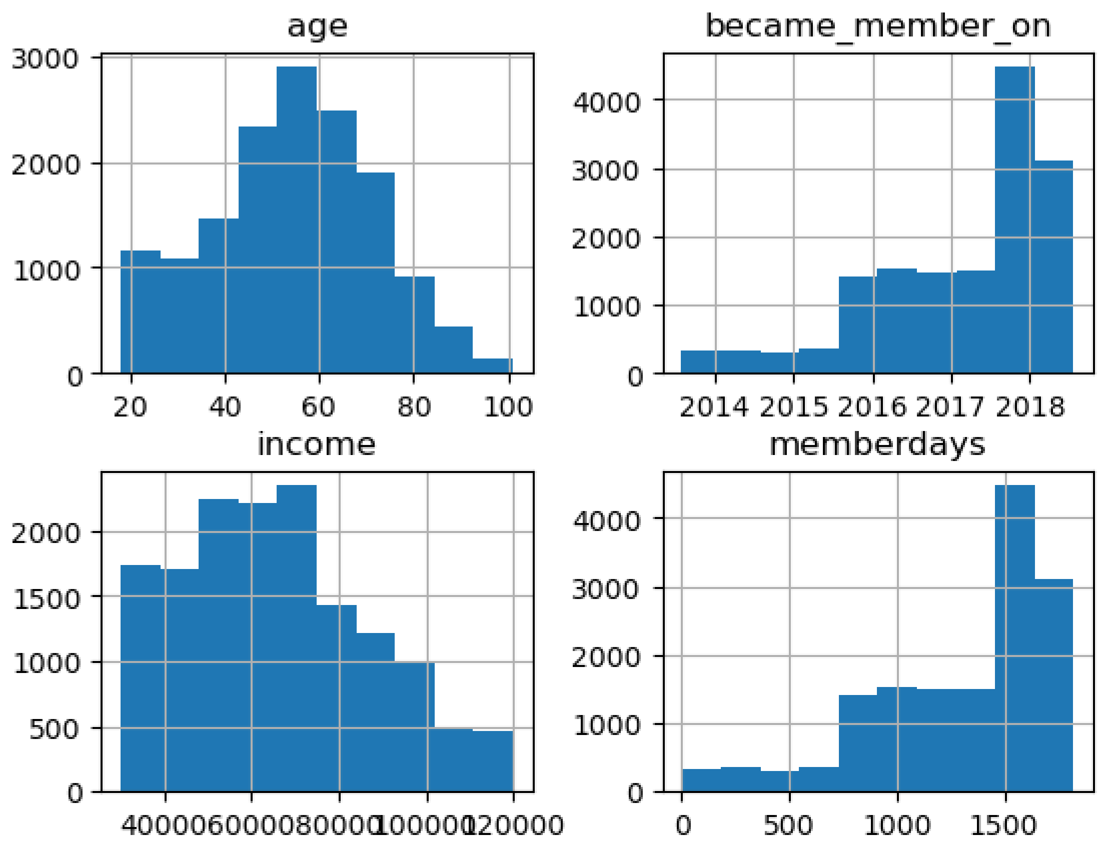


	gender	age	id	became_member_on	income	memberdays
1	F	55.0	0610b486422d4921ae7d2bf64640c50b	2017-07-15	112000.0	1447
3	F	75.0	78afa995795e4d85b5d9ceeca43f5fef	2017-05-09	100000.0	1380
5	M	68.0	e2127556f4f64592b11af22de27a7932	2018-04-26	70000.0	1732
8	M	65.0	389bc3fa690240e798340f5a15918d5c	2018-02-09	53000.0	1656
12	M	58.0	2eeac8d8feae4a8cad5a6af0499a211d	2017-11-11	51000.0	1566

Visualization

During EDA, we will visualize and analyze the dataset to uncover patterns, trends, and insights that can inform the development of the recommendation engine. Key findings may include:

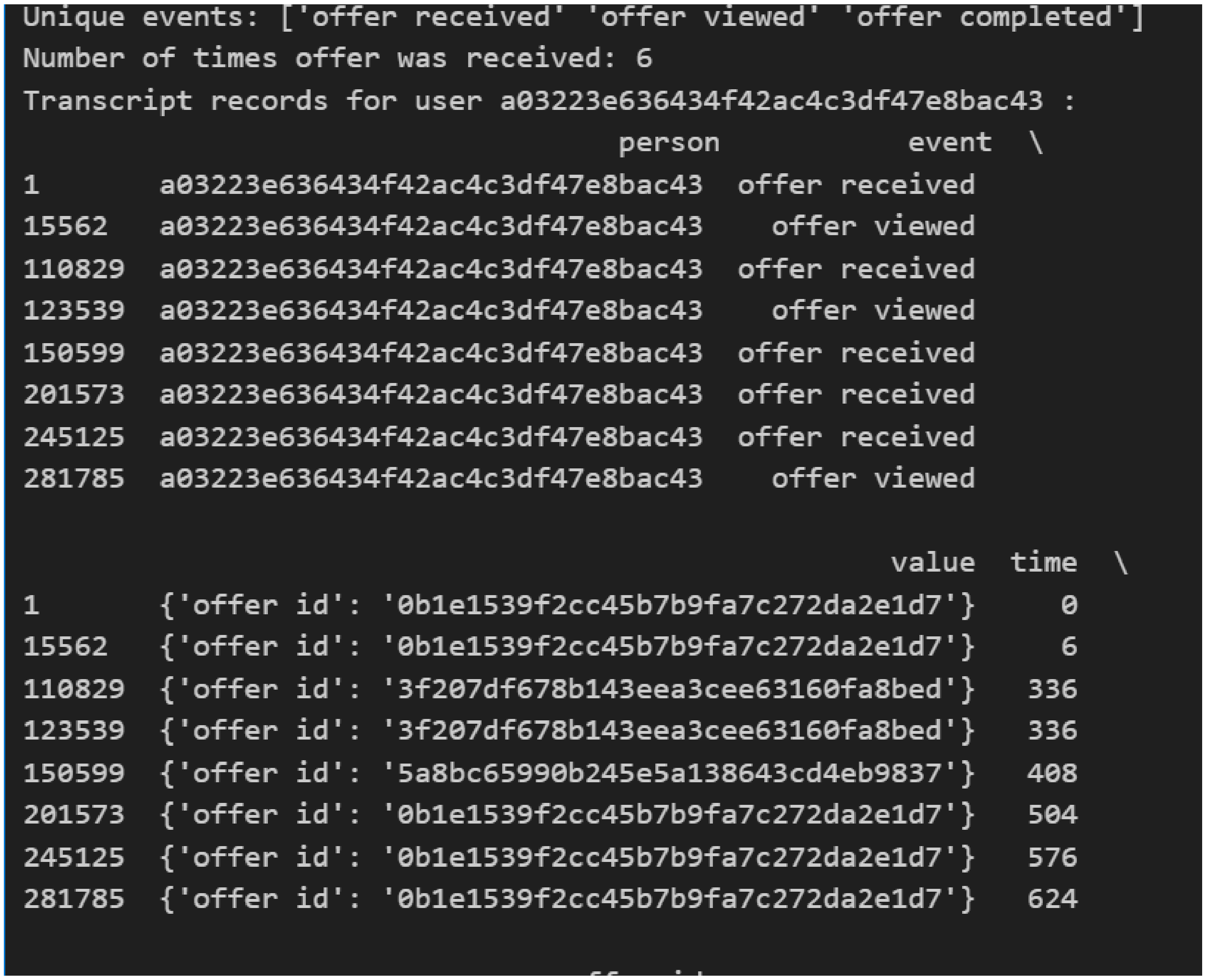
- Distribution of customer demographics (age, gender, income) and their impact on purchasing behavior.
- Frequency and effectiveness of different offer types (BOGO, discount, informational).
- Correlation between offer difficulty, reward, and customer responsiveness.
- Time trends in offer usage and customer engagement.
- Interaction patterns between offers received, viewed, and completed.



Explore More

example user: 'a03223e636434f42ac4c3df47e8bac43

- Check unique events in the 'event' column
- Check how many times the offer was sent (offer received)
- Filter transcript records for a specific user (example user: 'a03223e636434f42ac4c3df47e8bac43')
- Print the filtered transcript records for the user



```
Unique events: ['offer received' 'offer viewed' 'offer completed']
Number of times offer was received: 6
Transcript records for user a03223e636434f42ac4c3df47e8bac43 :
                                  person           event  \
1       a03223e636434f42ac4c3df47e8bac43  offer received
15562   a03223e636434f42ac4c3df47e8bac43    offer viewed
110829  a03223e636434f42ac4c3df47e8bac43  offer received
123539  a03223e636434f42ac4c3df47e8bac43    offer viewed
150599  a03223e636434f42ac4c3df47e8bac43  offer received
201573  a03223e636434f42ac4c3df47e8bac43  offer received
245125  a03223e636434f42ac4c3df47e8bac43  offer received
281785  a03223e636434f42ac4c3df47e8bac43    offer viewed

                                                   value  time  \
1       {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}     0
15562   {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}     6
110829  {'offer id': '3f207df678b143eea3cee63160fa8bed'}   336
123539  {'offer id': '3f207df678b143eea3cee63160fa8bed'}   336
150599  {'offer id': '5a8bc65990b245e5a138643cd4eb9837'}   408
201573  {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}   504
245125  {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}   576
281785  {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}   624
```

Data Processing

Return the user-item matrix indicating the number of offer completions by each user.

This function creates a user-item matrix based on the offer dataframe, where:

- Rows represent users.
- Columns represent offers.
- Values represent the number of offer completions by the user (1 if completed, 0 otherwise).

INPUT:

offer (DataFrame): A cleaned transcript dataframe.

filename (str): The filename to save the user-item matrix as a pickle file.

OUTPUT:

Creation of training datasets as 'train_df.p', Testing datasery as 'train_df.p' and full dataset as user_item_matrix.p

Modeling:

```
This function performs matrix factorization using a basic form of FunkSVD with no regularization
    INPUT:
    complete_mat - (numpy array) a matrix with users as rows, offers as columns, and offer completed as values
    latent_features - (int) the number of latent features used
    learning_rate - (float) the learning rate
    iters - (int) the number of iterations
    OUTPUT:
    user_mat - (numpy array) a user by latent feature matrix
    offer_mat - (numpy array) a latent feature by offer matrix
```

Steps

- Set up useful values to be used through the rest of the function.
- initialize the user and matrices with random values.
- initialize sse at 0 for first iteration.
- keep track of iteration and MSE
- for each iteration update our sse
- For each user-offer pair
- if the rating exists
- compute the error as the actual minus the dot product of the user and offer latent features.
- Keep track of the sum of squared errors for the matrix
- update the values in each matrix in the direction of the gradient.
- Create user-by-item matrix - nothing to do here

- Fit FunkSVD with the specified hyper parameters to the training data
- Test for the best number of latent feature. (with latent features 10)
- Test for the best number of latent feature. (with latent features 5)

Funk Singular Value Decomposition (FunkSVD):

Underlying Principles: FunkSVD is a matrix factorization technique used for collaborative filtering in recommendation systems. It decomposes the user-item interaction matrix into two lower-dimensional matrices: one representing users and their latent features, and the other representing items (offers) and their latent features. The dot product of these matrices approximates the original user-item interaction matrix, allowing for the prediction of missing values (unseen user-item interactions).

Assumptions:

FunkSVD assumes that users and items can be represented by a set of latent features, and their preferences or characteristics can be captured by the dot product of these latent feature vectors. It also assumes that the user-item interaction matrix is sparse, with many missing values.
Parameter Settings: Parameter settings for FunkSVD include:

latent_features:

The number of latent features to be learned by the model. This parameter controls the complexity of the model and the dimensionality of the latent feature space.

learning_rate:

The step size for updating the latent feature matrices during optimization. A smaller learning rate may lead to slower convergence but prevents overshooting the optimal solution.

Iterations:

The number of iterations or epochs for training the model. More iterations allow the model to learn better representations but may increase training time.
Choice in Relation to Data Characteristics: FunkSVD is suitable for recommendation tasks with implicit feedback data, such as user-item interactions without explicit ratings. The choice of parameters, such as the number of latent features, is made based on the sparsity of the user-item interaction matrix, the complexity of user preferences, and computational constraints.

Parameter Tuning:

Underlying Principles: Parameter tuning involves selecting the optimal values for hyperparameters (e.g., latent_features, learning_rate, iters) to maximize model performance.

Assumptions:

The assumption is that different combinations of hyperparameters can lead to variations in model performance. By systematically searching through the hyperparameter space, we aim to identify the configuration that yields the best results.
Parameter Settings: Parameter tuning techniques may include grid search, random search, or Bayesian optimization. Grid search exhaustively searches through a predefined set of hyperparameter combinations, while random search samples randomly from the hyperparameter space. Bayesian optimization uses probabilistic models to guide the search process more efficiently.
Choice in Relation to Data Characteristics: Parameter tuning is essential to find the optimal configuration that balances model complexity and performance. The choice of tuning technique depends on factors such as the size of the hyperparameter space, computational resources, and the desired level of optimization.

Model evaluation and Validation

The validation function calculates the Mean Squared Error (MSE) for the predictions made by a recommendation model on a test dataset. Let's break down the function and its parameters:

Parameters:
test_df: A pandas DataFrame representing the test dataset containing user-item interactions. The DataFrame has users as rows, offers as columns, and offer completions as values.
user_mat: A NumPy array representing the user by latent feature matrix learned by the recommendation model. It captures the latent features of users.
offer_mat: A NumPy array representing the latent feature by offer matrix learned by the recommendation model. It captures the latent features of offers.
Returns:
mse: A float representing the Mean Squared Error (MSE) for the predictions made by the recommendation model on the test dataset.
Functionality:
The function first calculates the total number of non-missing (non-NaN) values in the test dataset using np.count_nonzero(~np.isnan(test_df)).
It initializes an accumulator variable sse_accum to track the sum of squared errors.
It iterates over each user-item pair in the test dataset:
If the interaction value in the test dataset is not missing (i.e., not NaN), it predicts the user's reaction to the offer using the predict_reaction function.
If a prediction is made (i.e., predict_value is not None), it calculates the squared error between the actual interaction value and the predicted value.
It accumulates the squared error in sse_accum.
Finally, it calculates the Mean Squared Error (MSE) by dividing the sum of squared errors by the total number of non-missing values in the test dataset.
Use of predict_reaction function:
The function uses the predict_reaction function to predict user reactions to offers based on the learned user and offer matrices.
This function is assumed to be defined elsewhere and returns a prediction for the user's reaction to a given offer.
The validation function allows for evaluating the performance of the recommendation model by quantifying the squared errors between predicted and actual interactions on the test dataset. A lower MSE indicates better accuracy and performance of the recommendation model.

Results/Conclusion

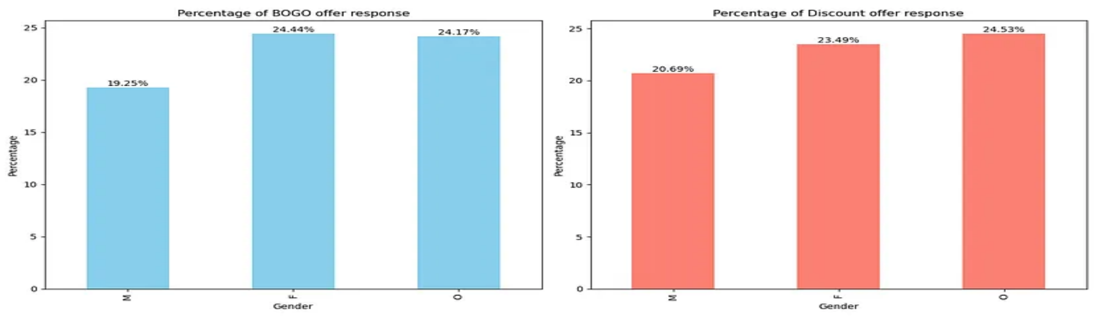
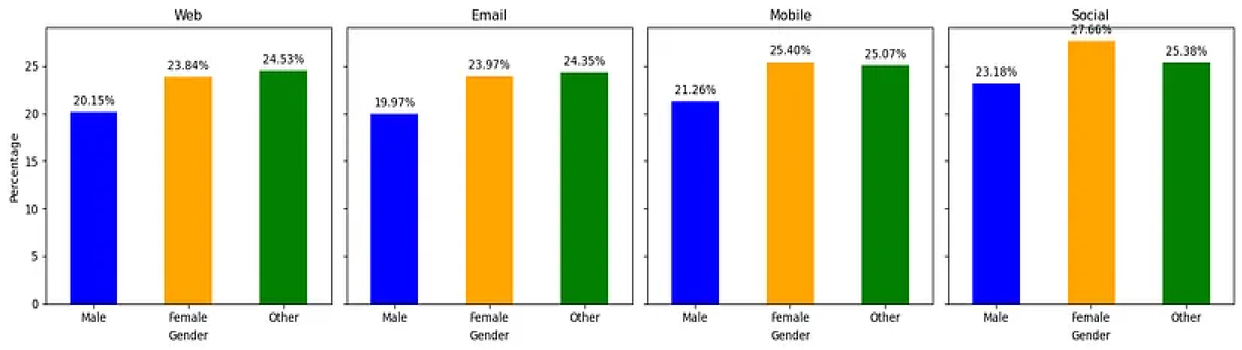


fig1.gender offer

Based on the above fig1.gender offer of offers sent and responses by gender, it’s evident that both BOGO (Buy One Get One) and discount offers were more frequently directed towards males, followed by females and individuals of other genders. Despite this distribution, males consistently exhibited higher response rates to both offer types compared to females and individuals of other genders. While females generally displayed slightly lower response rates than males, individuals of other genders consistently had the lowest response rates across both offer types. These findings suggest potential gender-based variations in response behaviors, with males showing greater engagement overall. Further analysis is required to discern the underlying factors driving these disparities and to optimize offer strategies accordingly.



f2.gender offer channel

The analysis of f2.gender offer channel of offers sent and responses by gender across different channels reveals consistent patterns of engagement. Regardless of the channel, males consistently exhibited higher response rates compared to females and individuals of other genders, with females generally displaying slightly lower response rates than males and individuals of other genders having the lowest response rates overall. While the distribution of offers across channels remained relatively consistent, with email being the most popular channel followed by web, mobile, and social media, the response patterns by gender persisted across all channels. These findings highlight the importance of considering gender-based differences in response behaviors when designing marketing strategies, suggesting potential avenues for further exploration to optimize engagement and enhance overall effectiveness across diverse demographic groups.

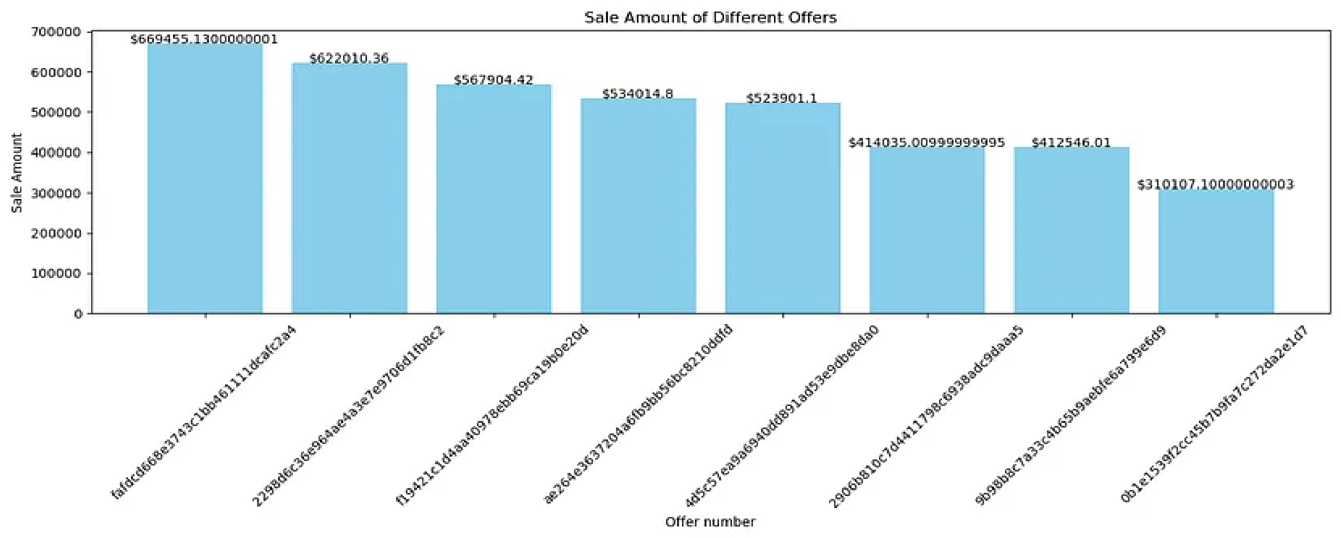


fig3.best sale offer

fig3.best sale offer presents offer IDs and their corresponding gains, indicating the financial performance associated with each offer. The offer with the ID “fafdcd668e3743c1bb461111dcafc2a4” yielded the highest gain of $669,455.13, followed by offer ID “2298d6c36e964ae4a3e7e9706d1fb8c2” with a gain of $622,010.36. The offers identified by IDs “f19421c1d4aa40978ebb69ca19b0e20d,” “ae264e3637204a6fb9bb56bc8210ddfd,” and “4d5c57ea9a6940dd891ad53e9dbe8da0” also contributed significantly to gains, amounting to $567,904.42, $534,014.80, and $523,901.10, respectively. The remaining offers, with IDs “2906b810c7d4411798c6938adc9daaa5,” “9b98b8c7a33c4b65b9aebfe6a799e6d9,” and “0b1e1539f2cc45b7b9fa7c272da2e1d7,” generated gains of $414,035.01, $412,546.01, and $310,107.10, respectively. This data provides insights into the relative effectiveness of different offers in generating revenue, which can inform future marketing strategies and investment decisions.

Reflection

Model Selection and Tuning:

Went well: Exploring different recommendation algorithms and techniques, such as collaborative filtering, content-based filtering, and hybrid approaches, provided valuable insights into their strengths and weaknesses. Tuning hyperparameters and evaluating model performance helped identify the most effective approach for the given dataset.

Challenges: Selecting the optimal algorithm and tuning hyperparameters posed challenges due to the trade-off between model complexity, prediction accuracy, and computational resources. Balancing these factors while ensuring the scalability and interpretability of the models required iterative experimentation and evaluation.

Evaluation Metrics and Validation:

Went well: Choosing appropriate evaluation metrics, such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), or Mean Squared Error (MSE), allowed for quantitative assessment of model performance. Implementing robust validation techniques, such as cross-validation or holdout validation, ensured reliable estimation of the models' generalization ability.

Challenges: Interpreting evaluation metrics in the context of the project goals and stakeholders' expectations required careful consideration. Addressing issues related to data sparsity, class imbalance, or overfitting during validation posed challenges in ensuring the reliability and fairness of model evaluations.

Acknowledgment

Udacity for training and data recommendations

Starbucks for dataset