ZEPTO ASSIGNMENT

Link: Colab File

INTRODUCTION:

In the rapidly evolving world of online advertising, the ability to accurately predict click-through rates (CTR) is crucial for optimizing ad performance and enhancing user engagement. This project focuses on developing a robust CTR prediction model using 90 days' worth of data from Zepto, a leading e-commerce platform. By leveraging a comprehensive dataset that includes search queries, product interactions, and various contextual features, we aim to build a model that can effectively rank products based on their likelihood of being clicked.

ABOUT THIS DATASET:

Attributes:

● search_term: Search query.
● city_id: Unique IDs for different cities (city identifier).
● product_variant_id: Unique IDs for unique products (product identifier).
● is_clicked: 0/1 indicating non-click/click (target variable).
● query_type: Head/tail (head means popular query, tail means unpopular queries).
● total_clicks: Total clicks on a product for a given query at the city level (QxPxC).
● session_views: Total views of a product for a given query at the city level (QxPxC).
● query_products_clicks_last_30_days: Total clicks on a product for a given query at the city level in the last 30 days (QxPxC).
● CTR_last_30_days: Click-through rate of a product for a given query at the city level in the last 30 days (QxPxC).
● CTR_last_7_days: Click-through rate of a product for a given query at the city level in the last 7 days (QxPxC).
● CTR_product_30_days: Click-through rate of a product at the city level in the last 30 days (PxC).
● query_product_plt_clicks_60_days: Total clicks on a product for a given query in the last 60 days across the platform (QxPxPlt).
● query_product_plt_ctr_60_days: Total click-through rate of a product for a given query in the last 60 days across the platform (QxPxPlt).
● CTR_plt_30_days: Click-through rate of a product in the last 30 days across the platform (PxPlt).
● predicted_category_name: Predicted category name for the query.
● predicted_subcategory_name: Predicted subcategory name for the query.

● query_product_plt_clicks_30_days: Total clicks on a product for a given query in the last 30 days across the platform (QxPxPlt).
● Product_name: Name of the product.
● Brand_name: Brand of the product.
● category_name: Category of the product.
● subcategory_name: Subcategory of the product.
● latest_margin: Profit margin of the product.
● savings: Savings on the product.
● savings_with_pass: Savings on the product with a Zepto pass.
● ad_revenue: Advertising revenue.
● total_unique_orders: Total unique orders (PxC).
● product_clicks_30_days: Total clicks on the product in the last 30 days at the city level (PxC).
● product_clicks_plt_30_days: Total clicks on the product in the last 30 days on the platform (PxPlt).
● total_unique_orders_plt_30_days: Total unique orders in the last 30 days on the platform (PxPlt).
● product_ctr_city_30_days: Product click-through rate in the last 30 days for the city (PxC).
● query_product_similarity: Cosine similarity of query and product embeddings.

Q -> Query
P -> Product
C -> City
Plt -> Platform

IMPORT LIBRARIES:

1. The NumPy and pandas libraries are essential for data manipulation and analysis in Python.
2. NumPy is used for numerical computing in Python. It provides support for arrays, matrices, and many mathematical functions.
3. Pandas is a powerful data manipulation library built on top of NumPy. It provides data structures like DataFrame and Series for efficient data manipulation.
4. Matplotlib and Seaborn is a plotting library used for creating static and interactive visualizations in Python.

DATA CLEANING:

Data cleaning is a crucial step in preparing the dataset for analysis and modeling. The following operations demonstrate the process of handling missing values in the 'product_name' and 'brand_name' columns.

1. Removing Rows with Missing 'product_name'

First, we remove any rows where the 'product_name' column has missing values. This ensures that all the entries in our dataset have valid product names, which is essential for accurate analysis. We achieve this by filtering out the rows with null values in the 'product_name' column.

2. Imputing Missing Values in 'brand_name'

Next, we address the missing values in the 'brand_name' column by replacing them with a constant value, 'others'. To accomplish this, we use the `SimpleImputer` class from the scikit-learn library. The `SimpleImputer` is configured to replace missing values with a specified constant value. In this case, we set the constant to 'others'.We cannot simply remove null values in this case as the null values account for more than 5% of the column data.

The process of imputation involves fitting the imputer on the 'brand_name' column to identify the missing values and then transforming the column by filling these missing values with 'others'. The transformed data is then reassigned to the 'brand_name' column in the dataframe.

By performing these data cleaning operations, we ensure that our dataset is free from missing values in the 'product_name' and 'brand_name' columns. This step is vital as it enhances the quality and reliability of our dataset, leading to more robust and accurate analysis and modeling results.

3. Detecting Duplicates

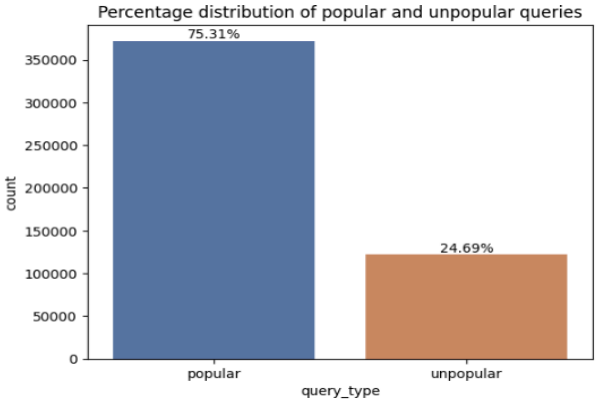
The df.duplicated()`.sum()` function accounts the number of `True` values, effectively giving the total number of duplicate rows in the dataframe.

By using this approach, we can quickly ascertain how many duplicate rows are present in our dataset, allowing us to decide on the appropriate action, such as removing these duplicates to maintain data quality.

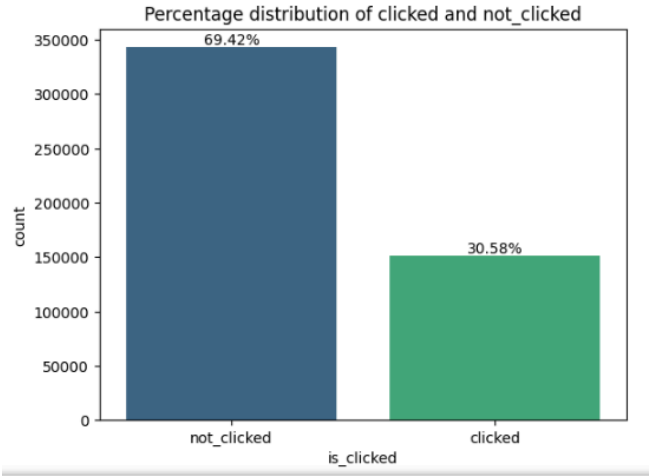
EXPLORATORY DATA ANALYSIS (EDA)

Key Findings:

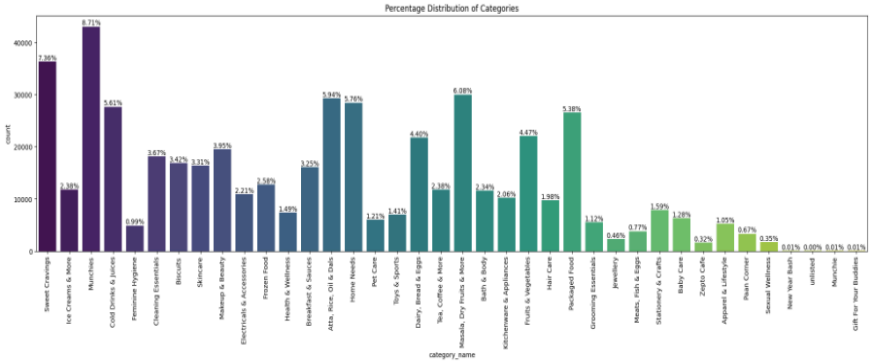
1. The majority of search queries are popular, accounting for 75.31% of the total queries, while the remaining 24.69% are unpopular.



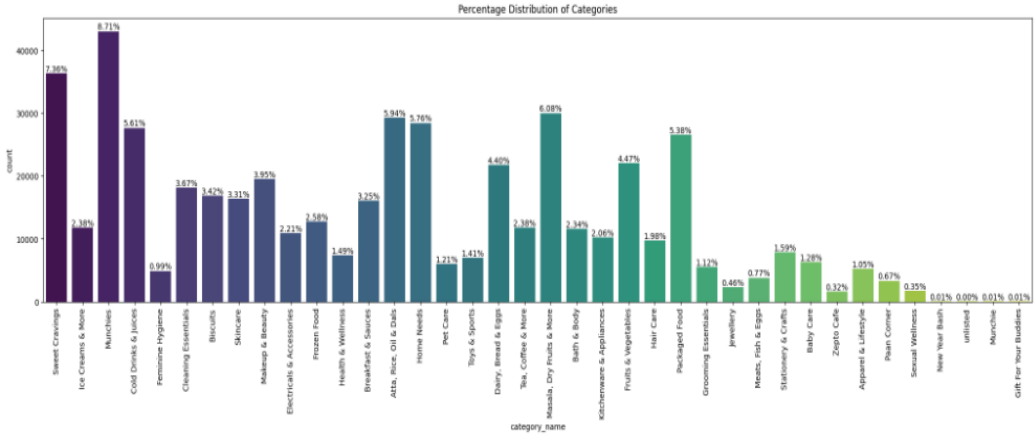
2. Approximately 30.58% of the queries result in clicks, while the majority, 69.42%, do not lead to clicks.



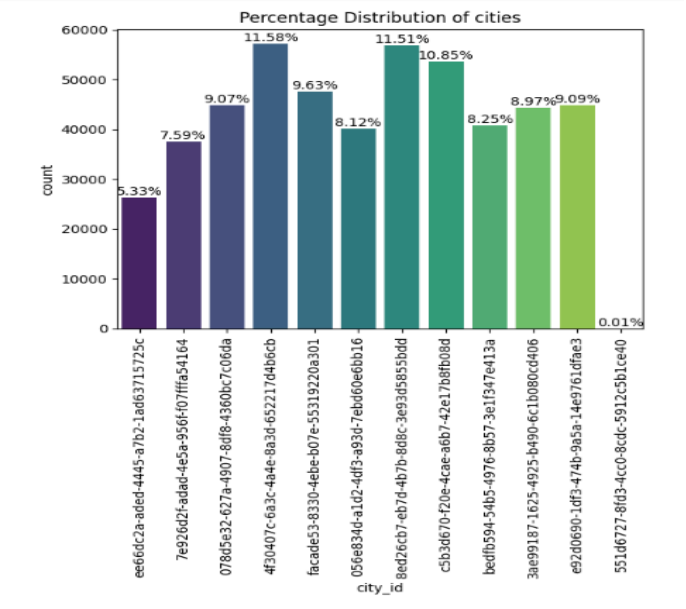
3. Munchies category accounts for maximum percentage distribution.



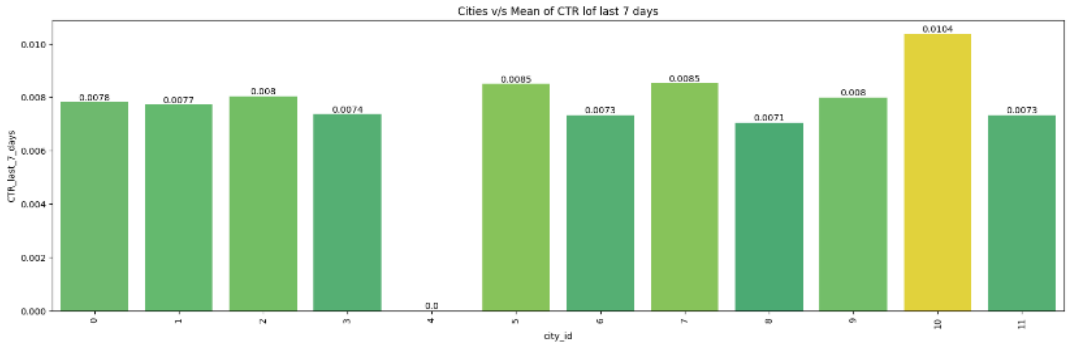
4. Munchies category accounts for maximum percentage distribution.



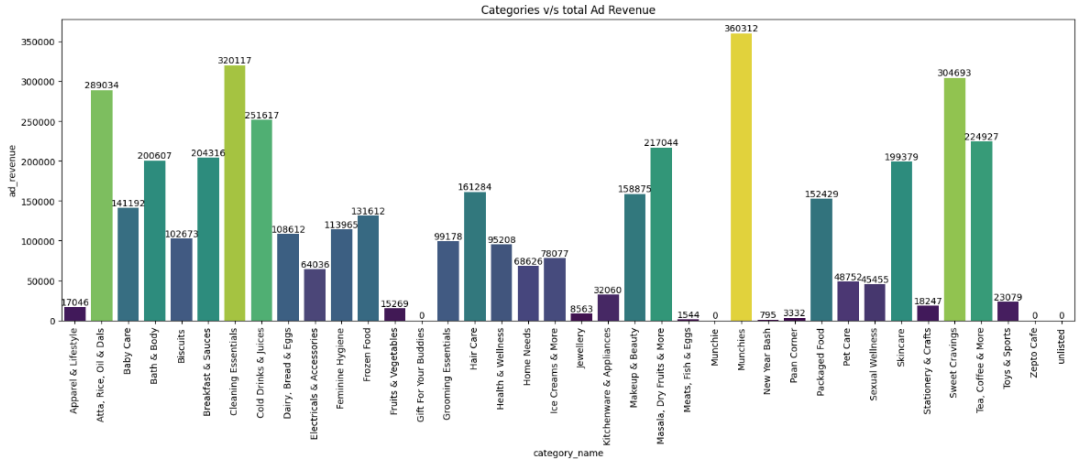
5. The highest percentage fraction of data from a city is 11.58% and lowest is 0.01%.



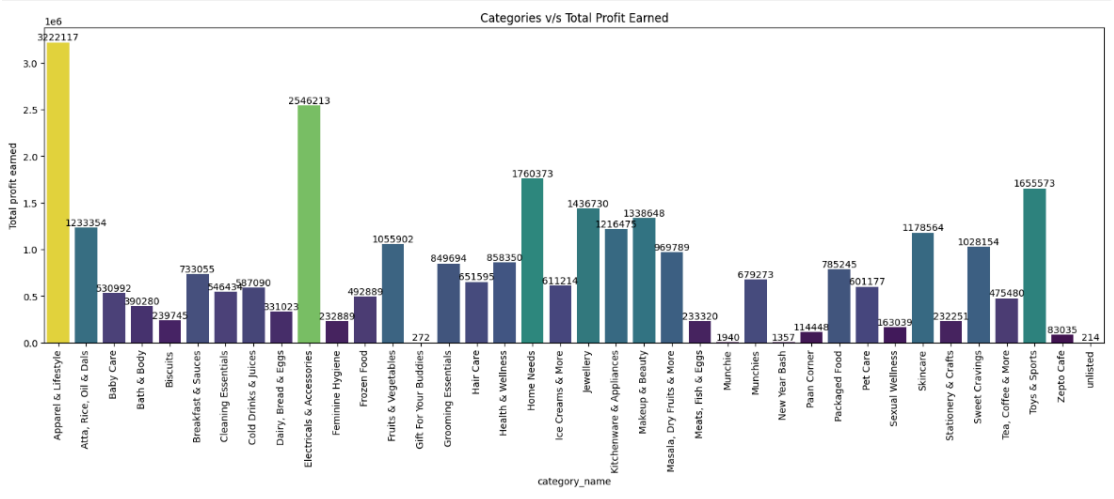
6. The average CTR for each city ranges from 0 to 0.0104.



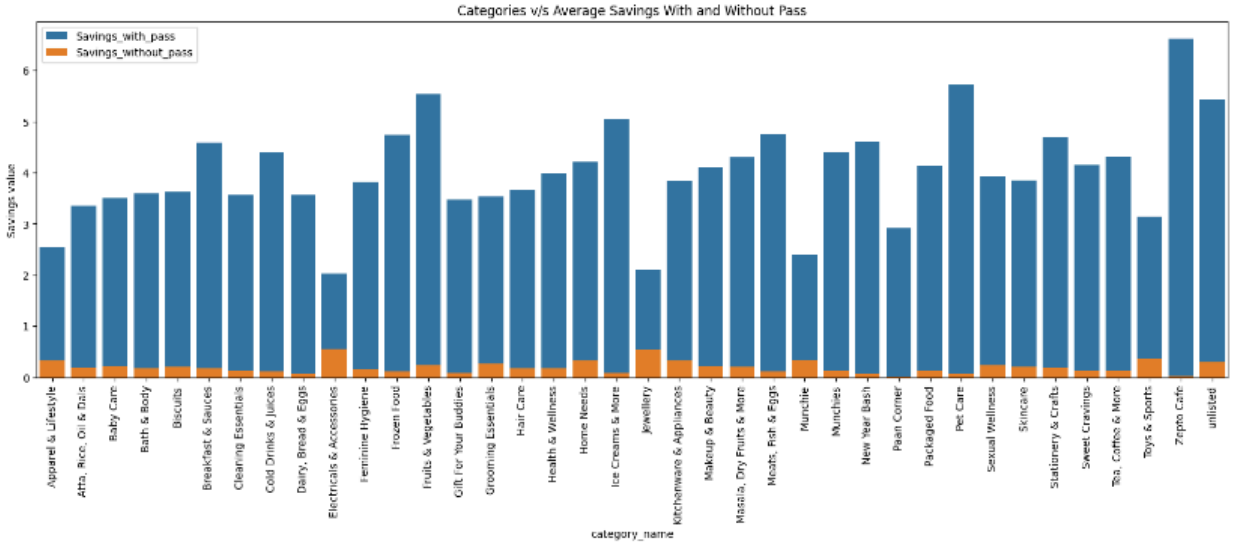
7. The top three categories with the highest Ad Revenue were Munchies, Cleaning Essentials and Sweet Cravings.



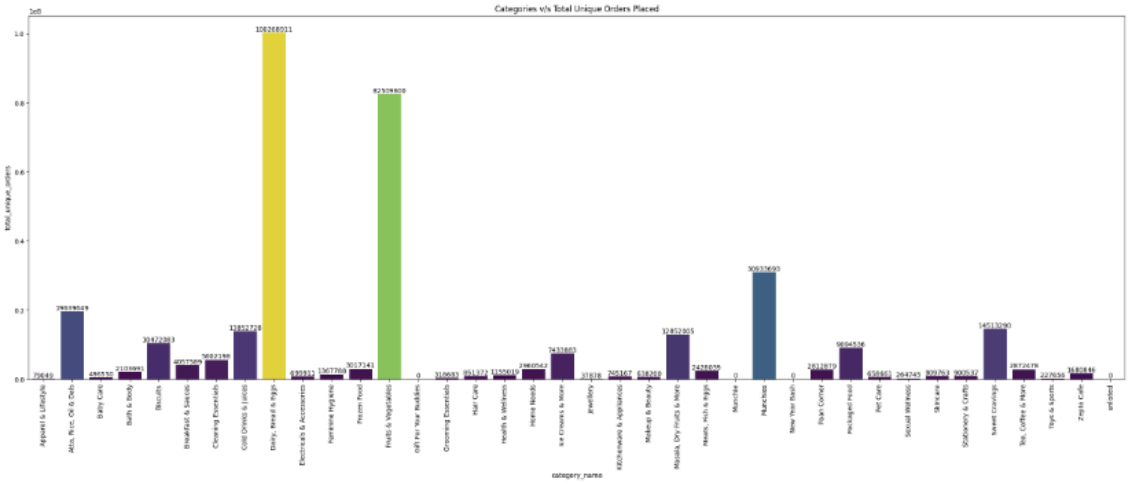
8. ‘Apparel and Lifestyle’ category accounts to maximum total profit earned



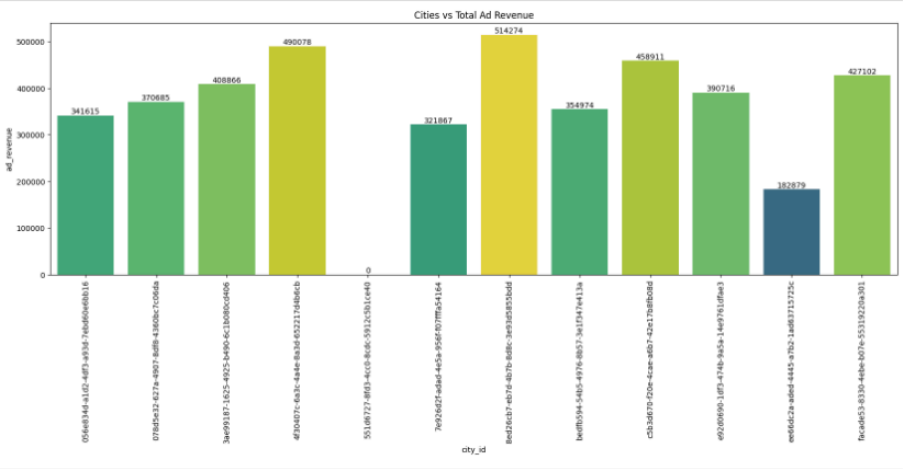
9. The following graph clearly shows the difference in savings with pass and without pass. And this difference is huge in the case of all categories.



10. Total unique orders placed for ‘Dairy bread and eggs’ and ‘fruits and vegetables’ categories are way higher than other categories.



11. Almost every city contributed equally to total as revenue.



FEATURE TRANSFORMATION:

To convert categorical variables into numerical representations suitable for machine learning algorithms, we applied Label Encoding. This technique transforms categorical text labels into numerical values, which is necessary because many algorithms require numerical input.

We used Label Encoding to iterate over each categorical column in the dataset. For each column, we replaced the original categorical values with numerical labels. Additionally, we stored the mappings of original categorical values to their encoded numerical values in a dictionary. This mapping allows us to understand the transformation and provides a reference for interpreting the model's predictions or for inverse transformations if needed.

FEATURE CONSTRUCTION:

We created two new features, '`latest_trends`' and '`city_wise_trends`', to capture recent and location-based trends. The '`latest_trends`' feature measures the ratio of click-through rates (CTR) over the last 7 days to the last 30 days, providing insight into recent performance. Similarly,'`city_wise_trends`' assesses the ratio of product CTR over the last 30 days to platform CTR in the same period, highlighting city-specific trends. Both features default to zero if the denominator is zero to avoid division errors. These transformations help enhance the model's ability to understand temporal and geographical patterns in the data.

FEATURE SELECTION:

To evaluate feature importance, we first split the dataset into training and testing sets. We then trained a RandomForestClassifier on the training data. Using the trained model, we extracted the feature importances, which indicate the contribution of each feature to the model's predictions. We created a DataFrame to list these features along with their importance scores, and sorted this DataFrame to highlight the most influential features. This process helps in identifying and focusing on the most significant features for model performance.

MODEL TRAINING:

The model training process involves preparing the dataset, encoding categorical variables, and fitting a Random Forest Classifier to predict click probabilities for different products based on given queries. Initially, we create a balanced dataset with a mix of queries and products, each accompanied by 16 random features and a target variable indicating whether a product was clicked. The features and the target variable are then split into training and testing sets to evaluate the model's performance.

A Random Forest Classifier with 400 trees and a maximum depth of 8 is trained on the training data, capturing complex patterns in the features to predict the likelihood of a product being clicked. The trained model's performance is assessed using metrics such as accuracy and ROC AUC score, ensuring its effectiveness in distinguishing between clicked and unclicked products. This robust model is then utilized to rank products based on their predicted click probabilities for each query, providing valuable insights for recommendation systems.

RANK PREDICTION:

The `predict_click_probabilities` function aims to predict and rank products based on their click probabilities for a given search query. The function first checks if the query exists in the dataset. If the query is found, it filters the dataset to obtain the relevant products and their features, excluding the target variable `is_clicked`. The model then predicts the click probabilities for these products using their feature values. The results are compiled into a DataFrame containing the product names and their respective click probabilities.

Additionally, the function calculates a new metric, `Click_Cosine_Product`, by multiplying the total clicks by the query-product similarity, and sorts the products based on this metric in descending order. This sorting helps in determining the relative importance of each product for the given query. Finally, the function iterates through the unique products and prints the product names with the highest click probabilities and cosine similarity, providing a ranked list of products tailored to the query.

```
query_name = 'comb'
results = predict_click_probabilities(query_name)
```

```
Vega Detangling Hair Comb - 1265 (Black)
Vega Tulip Grooming Hair Comb For Men And Women-Large, (Dc-1279)
Vega Double Sided Lice Hair Comb - Hmc-37
Vega Lilac Shampoo Hair Comb For Men And Women
Vega Graduated Dressing Hair Comb Handmade For Men And Women
```

```
query_name = 'chocolate'
results = predict_click_probabilities(query_name)
```

```
Cadbury Dairy Milk Chocolate Bar
Cadbury Dairy Milk Chocolate Home Treats
Nestle Nestle Munch Chocolate
Cadbury Dairy Milk Silk Oreo Chocolate Bar
Cadbury Dairy Milk Roast Almond Chocolate Bar
```