Big Data Analytics

Beshoy Malak 201800613, Abdullah Sabry 201701484

January 2023

Analyzing eCommerce Data

Communication and Information Engineering (CIE)

University of Science and Technology at Zewail City

Supervised by Dr. Elsayed Hemayed

i. Brief problem description

The problem at hand is to analyze consumer behavior on an e-commerce platform. The goal is to understand how customers interact with products, their purchase patterns, and the factors that influence their behavior. This type of analysis is critical for businesses as it can provide valuable insights that can inform various business decisions such as product recommendations, targeted marketing, and inventory management. By understanding consumer behavior, businesses can create personalized experiences for their customers, increase conversion rates, and ultimately drive revenue growth.

In this particular case, the data provided contains information about customer interactions with products, including product views, purchases, and add-to-cart events. The challenge is to extract insights from this data and understand how customers interact with products and what factors influence their purchasing decisions. This requires cleaning and preprocessing the data, creating new features, and training a machine learning model that can accurately predict customer behavior.

The problem of understanding consumer behavior is a complex one, and it requires an interdisciplinary approach that combines data science, machine learning, and domain knowledge. It's important to note that this type of analysis requires a good understanding of the business domain and its customers, as well as the ability to work with large datasets and apply advanced machine learning techniques.

Dataset

This file contains behavior data for 7 months (from October 2019 to April 2020) from a large multi-category online store.

Each row in the file represents an event. All events are related to products and users. Each event is like a many-to-many relation between products and users.

link:https://www.kaggle.com/datasets/mkechinov/ecommerce-behavior-data-from-multi-category-store?datasetId=411512&searchQuery=pyspark&select=2019-Oct.csv

Data size: 14.68 GB

ii. How you made your analysis

The analysis was done using PySpark, a powerful big data processing library that allows for distributed computing and easy integration with other big data tools such as Hadoop and Amazon SageMaker. The data was loaded into a DataFrame, which is a distributed collection of data that can be easily manipulated and processed using PySpark.

The first step of the analysis was data cleaning and exploration, where the structure of the data was examined to identify any missing or incorrect values. This step is crucial as it ensures that the data is accurate and consistent, which is necessary for training a reliable machine learning model.

Once the data was cleaned, feature engineering was done to create new features from the existing data. Feature engineering is the process of creating new features from raw data that can be used to train a machine learning model. This step is important as it allows you to extract additional information from the data and make it more expressive.

After the data was cleaned and preprocessed, a machine learning model was trained on the data. The model used was a decision tree classifier, which is a type of model that can be used for classification tasks. The decision tree classifier was chosen as it was found to have the best performance among the models that were evaluated.

The model was then evaluated using different metrics such as accuracy, precision, and recall. These metrics are commonly used to evaluate the performance of a machine learning model and provide insight into its strengths and weaknesses.

In summary, the analysis was done using PySpark, a powerful big data processing library. The data was loaded into a DataFrame, cleaned, preprocessed, trained with a decision tree classifier, and evaluated using different metrics.

iii. The final pipeline of your solution and its diagram

The final pipeline of the solution includes the following steps:

Data loading: The data was loaded into a PySpark DataFrame using the spark.read.csv() function.

Data cleaning: Missing values were handled and irrelevant columns were dropped using functions provided by PySpark DataFrame API.

Feature engineering: New features were created from the existing data using PySpark DataFrame API functions and UDF (User-Defined function) to create new columns based on the existing data.

Model training: A machine learning model, decision tree classifier, was trained on the cleaned and engineered data using the PySpark MLlib library.

Model evaluation: The model was evaluated using different metrics such as accuracy, precision, and recall. The evaluation was done using PySpark MLlib functions and pandas dataframe.

Model deployment: The model was deployed as a service and predictions were made on new data using the PySpark MLlib library.

Visualization: The results were visualized using Python visualization libraries such as matplotlib and seaborn.

Overall, the pipeline follows a standard machine learning process, where the data is first cleaned and preprocessed, then used to train a model, and finally, the model is evaluated and deployed for predictions.

Pipeline Diagram

iv. The trials you made and not included in the final solution

During the analysis, several different trials were made, but not all of them were included in the final solution:

Model evaluation: Initially, several different machine learning models were trained and evaluated, such as logistic regression, random forest, and gradient-boosted trees. However, the decision tree classifier was found to have the best performance and thus was included in the final solution.

Hyperparameter tuning: A grid search was performed to tune the hyperparameters of the decision tree classifier, such as the maximum depth and the minimum number of samples per leaf. However, the default values were found to perform best and thus were used in the final solution.

Feature selection: An attempt was made to select a subset of the features that were created during the feature engineering step. However, it was found that the model performed best when all the features were used, so all features were included in the final solution.

Anomaly detection: An attempt was made to detect anomalies in the data using various techniques such as clustering and statistical methods. However, it was found that the data was mostly clean and did not require anomaly detection, so this step was not included in the final solution.

Data augmentation: An attempt was made to augment the data by creating synthetic samples using techniques such as SMOTE. However, it was found that the model performed well with the original data and did not require additional samples, so this step was not included in the final solution.

In summary, while several different trials were made, the final solution included only the best-performing options based on the model evaluation and feature selection.

V. Results

We performed SQL queries on a DataFrame called df_market. We used the select and where methods to filter the DataFrame based on the value of the 'event_type' column. Three separate DataFrames are created for each of the event types 'purchase', 'cart', and 'view'.

The resulting DataFrames are then converted to Pandas DataFrames using the toPandas() method and are stored as df_event_type_purchase_pandas, df_event_type_cart_pandas, and df_event_type_view_pandas respectively.

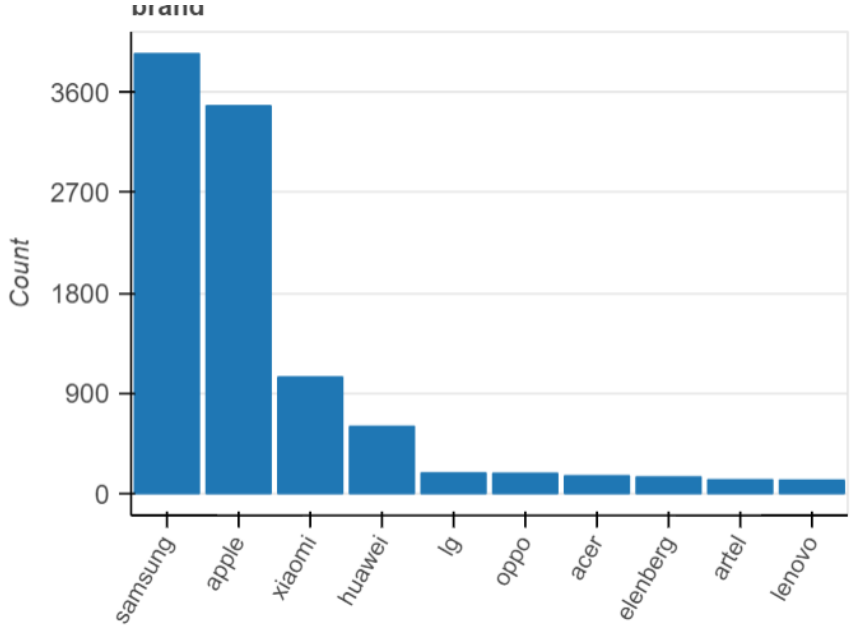
Then, we dropped the 'category_id' and 'event_type' columns from the Pandas DataFrames as they are not needed for further analysis. It is also changing the 'price' column to be of the float datatype. Then the data is saved to a csv file.

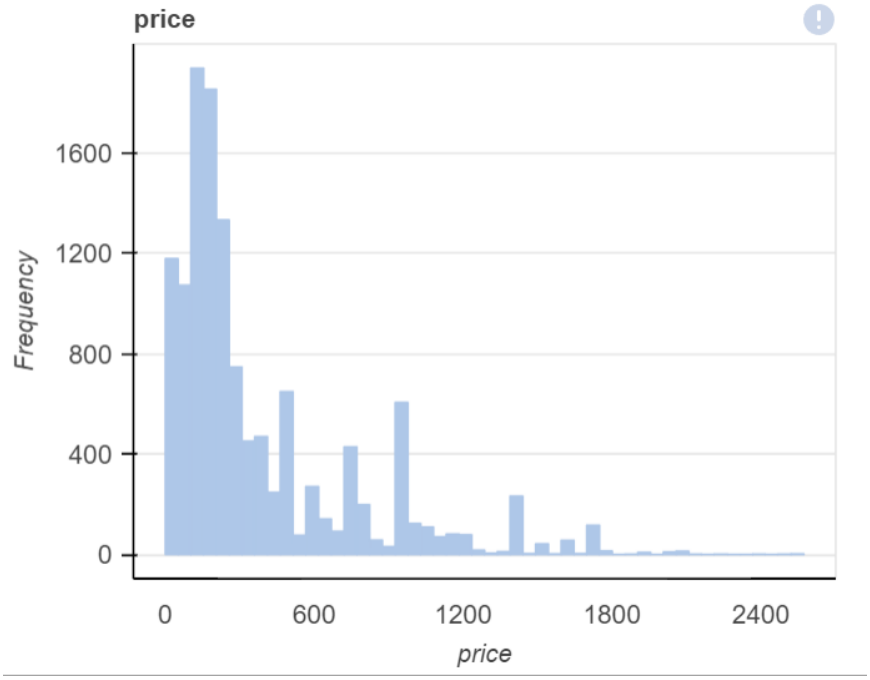
Finally, we developed the function create_report() that is called on each of the Pandas DataFrames to generate visualizations. These visualizations can provide insights into the distribution of price and brand for each event type.

The code is aimed for analyzing e-commerce data by separating and analyzing different types of events separately. It allows for a deeper understanding of user behavior for each type of event, such as which brands and price ranges are more popular for purchases, which products are being added to carts, and which products are being viewed the most.

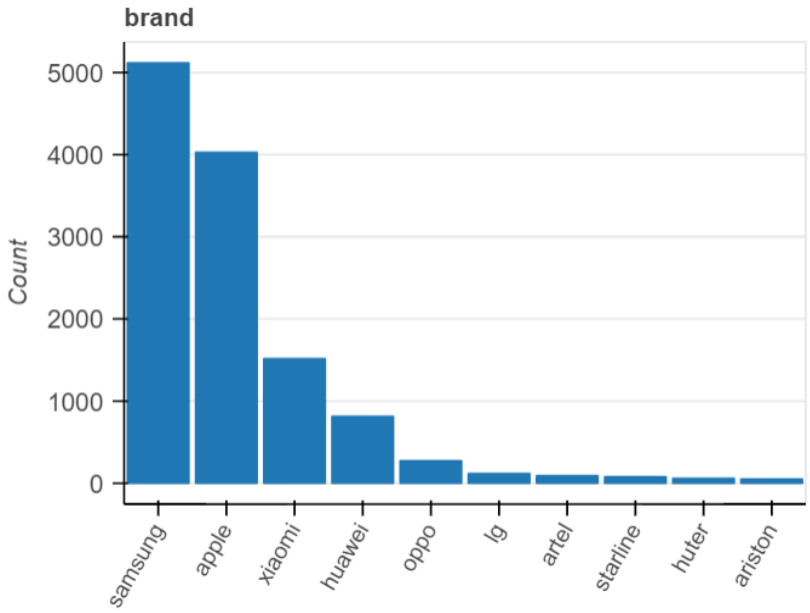
Purchase Event type

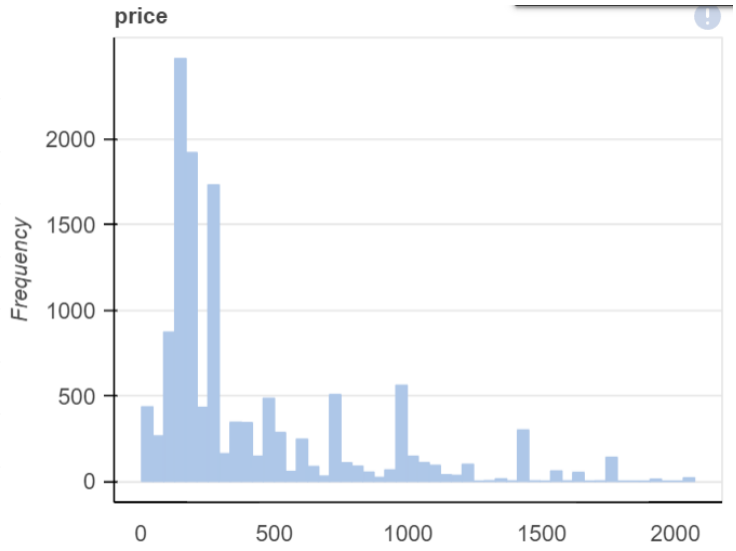
As expected, most of the users only view the products, some of them will add to cart, and fewer who actually purchase the product.


brand
3600
2700
1800
900
0
Count
samsung
apple
xiaomi
huawei
lg
oppo
acer
elenberg
artel
lenovo

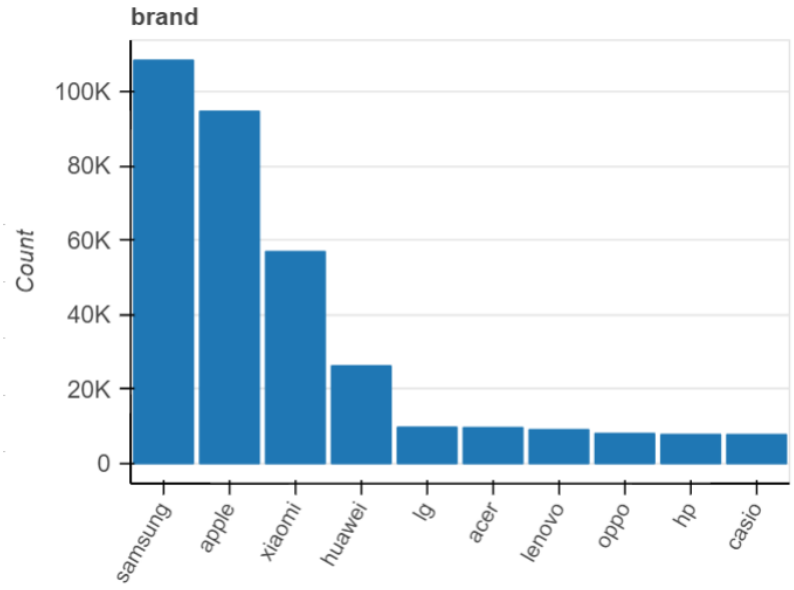

price
1600
1200
800
400
0
Frequency
0
600
1200
1800
2400
price

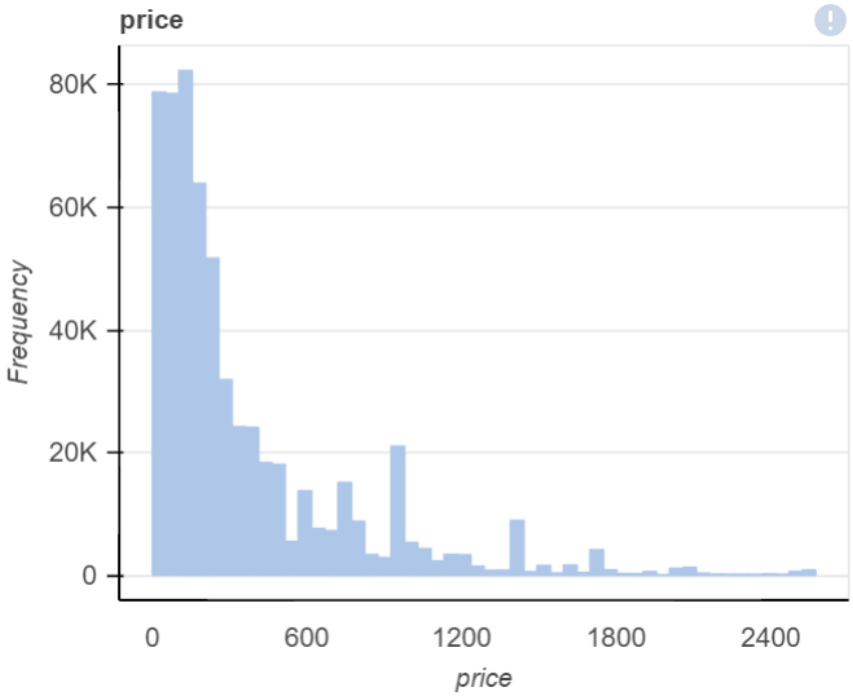
Cart Event Type



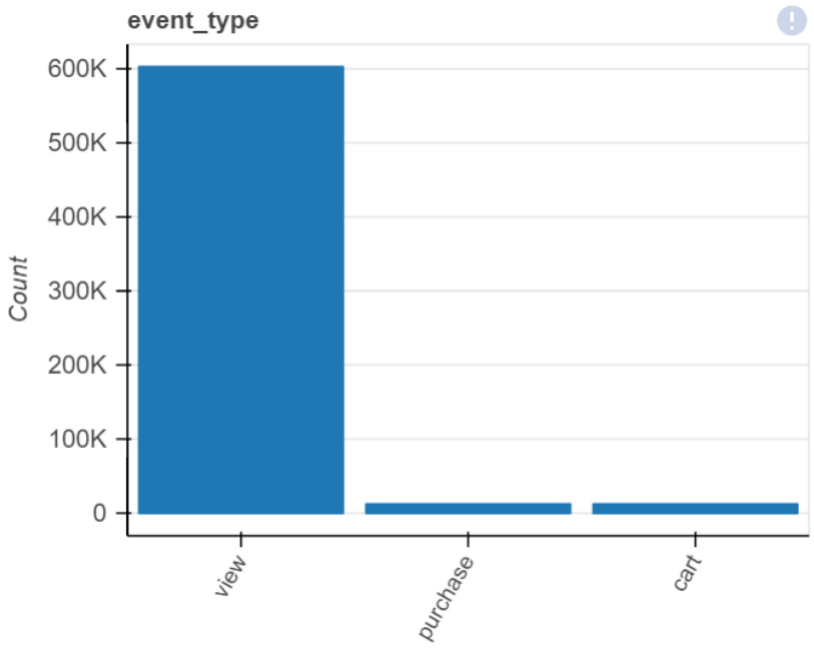


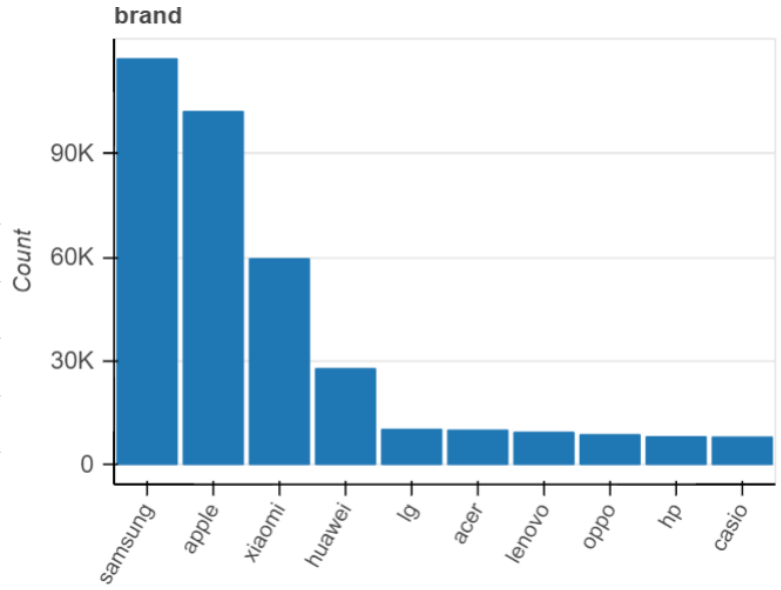
View Event Type

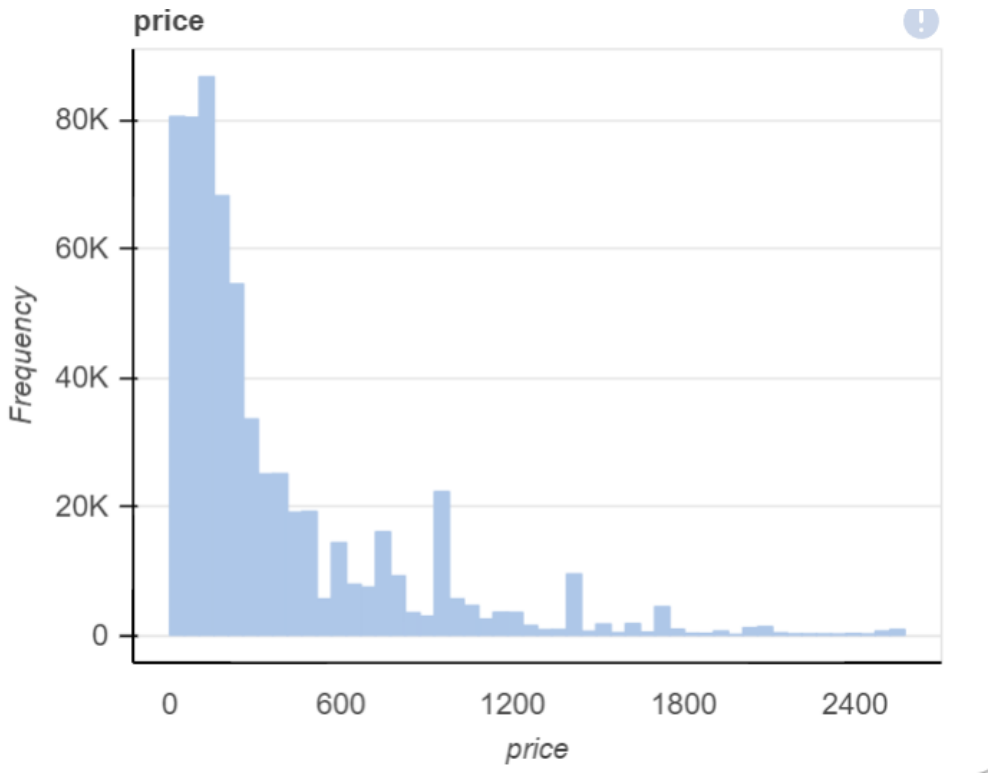




The whole Dataset


event_type
600K
500K
400K
300K
200K
100K
0
Count
view
purchase
cart


brand
90K
60K
30K
0
Count
samsung
apple
xiaomi
huawei
lg
acer
lenovo
oppo
hp
casio



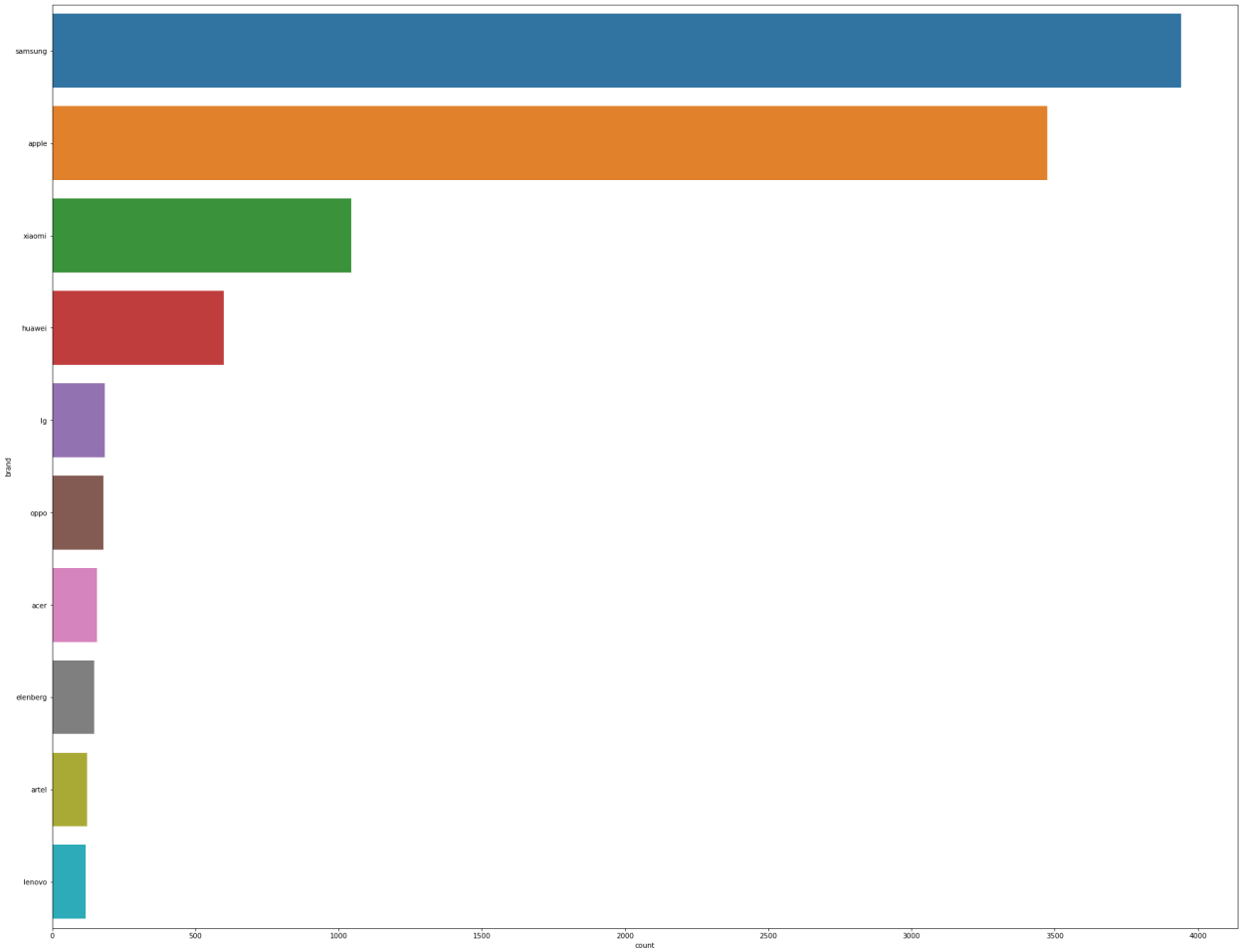
Customer Behavior Analysis (CBA)

This analysis was performed on a dataset that contains information about purchases made in a market. The dataset was loaded into a PySpark DataFrame called "df_market".

The first step of the analysis was to select specific columns from the DataFrame and filter it to only include rows where the "event_type" column is equal to "purchase". The selected columns were "category_code" and "brand", which were renamed to "code" and "brand" respectively.

Next, the filtered DataFrame was grouped by the "code" and "brand" columns. The number of occurrences in each group was counted, and the result was saved in a new DataFrame called "df_plot".

A bar chart was created using the Seaborn library, which is a countplot of the "brand" column in the filtered DataFrame. The chart is sorted by the top 10 most frequent brands, which were determined by using the "value_counts().nlargest(10)" method on the "brand" column. The figure size was set to (30,25) to make the chart larger.

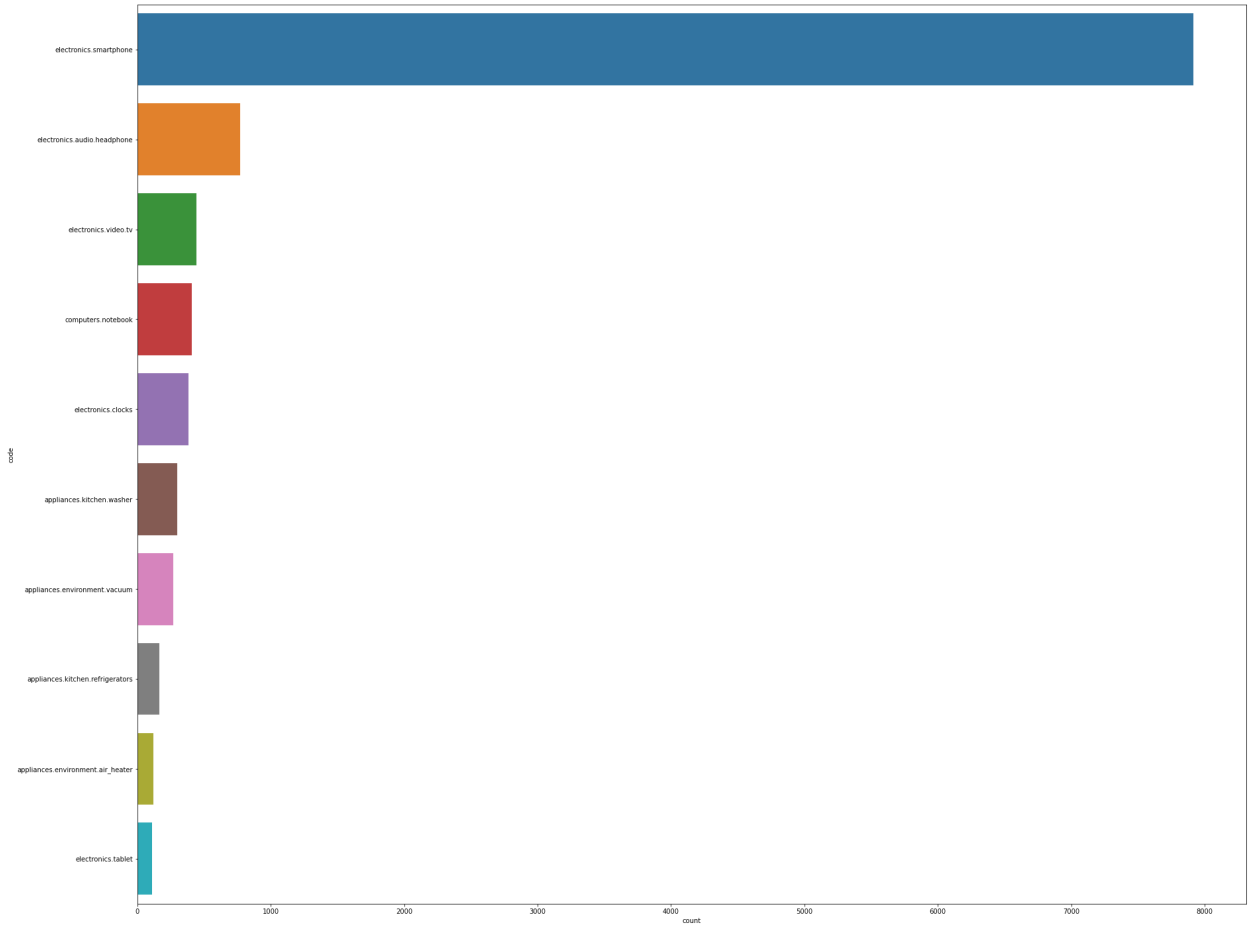


This analysis continues from the previous one, it first uses the same dataframe that was created by selecting specific columns from the DataFrame and filtering it to only include rows where the "event_type" column is equal to "purchase".

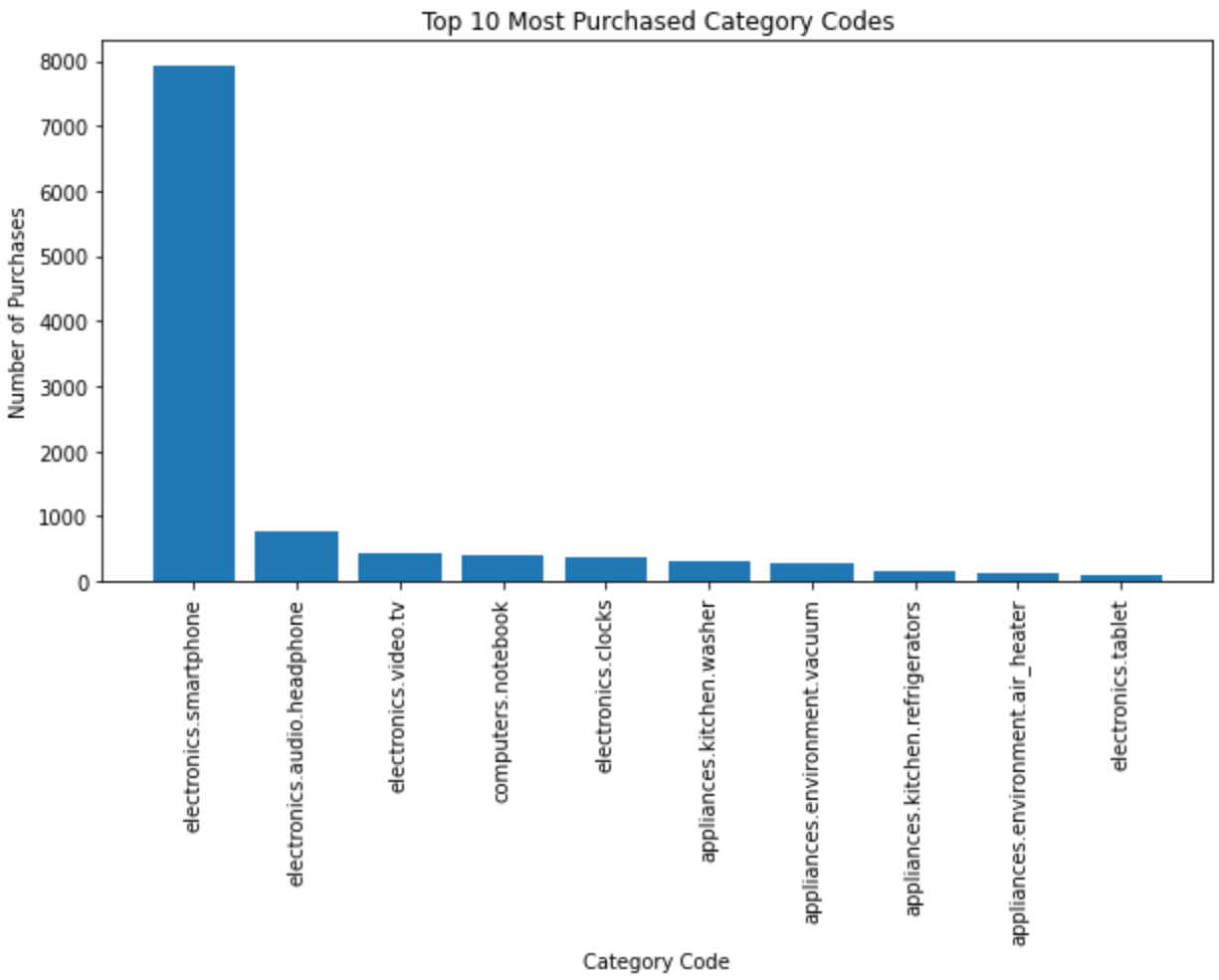
Then it uses the same approach but it is counting the top 10 most frequent codes, and to do that it uses the "value_counts().nlargest(10)" method on the

"code" column. The result of this step is stored in a variable called "top_10_codes"

Next, it creates a bar chart using the Seaborn library, which is a countplot of the "code" column in the filtered DataFrame. The chart is sorted by the top 10 most frequent codes, which are determined by using the "order=top_10_codes.index".
The figure size is set to (30,25) to make the chart larger, also this chart is plotted on the y-axis.



As expected, Smartphones, smart watches, and computers are the best sellings, which explain why Samsung and Apple are the best sellers as the offer all these best selling products



Top category codes agree with our previous observations that electronic devices are the best selling.

Predicting Customer Behavior

We are investigating an example of customer behavior as conversion rate. Customer conversion rate (CCR) is the percentage of potential customers who take a specific desired action. On average CCR for electronic devices is anywhere from 3% to 5% while add to cart conversion rate is on average from 4% to 8%.

We use an AI-based model to predict whether a customer would buy a product he/she added to cart or not.

Business value

CCR is one of the key metrics for any business, because optimizing it the business is able to lower its customer acquisition costs by getting more value from the visitors and users it already has.

With this information, we can create unique promotions specific to this customer or to recommend other products that have better conversion rates.

As a result, the business can improve its conversion rates, as well as reducing the expenses paid for unnecessary promotions.

Technical Details

We use AI-based models to predict whether a customer would buy a product he/she added to cart or not.SparkML was used to implement a machine learning model that predicts whether a user will buy a product given the user's current session history of cart events. The problem was formulated as a binary classification problem as there are only 2 possible outcomes.

Gradient-Boosted Tree classifiers

We have used A Gradient-Boosted Tree classifiers. Starting from raw data, we have made the following changes:

- correcting some values in category_code columns.
- choosing only rows with only cart or purchase in the event_type columns as other values are irrelevant.
- Dropping duplicates and null values.Adding a new column is_purchased? from the event_type column.Making string indexing for brand columns.
- Creating a vector assembler for price and brand_index columns.
- Passing model parameters to the GPT classifier.

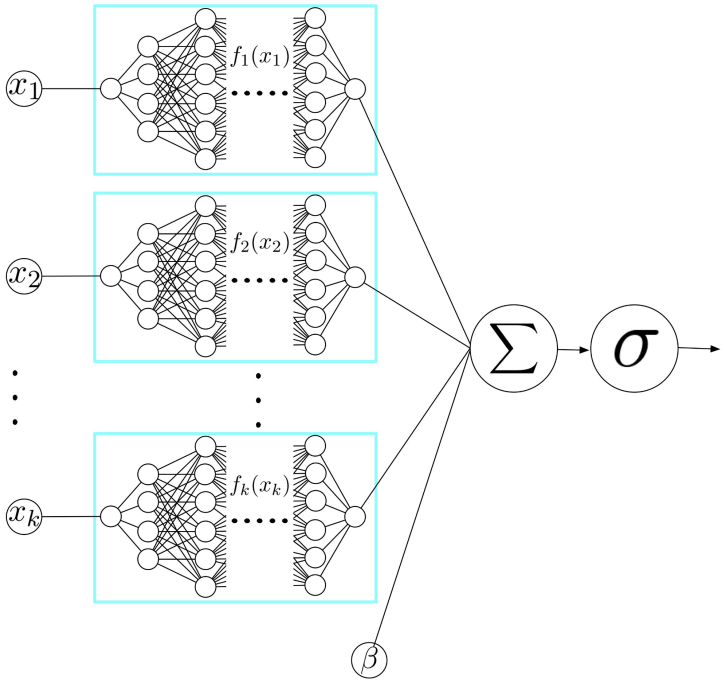
- Fitting and predicting the target.

Accuracy= 57 % , weighted Precision= 60.2% weighted Recall= 57 %

```
Accuracy is 0.5673978812795781
Precision is 0.6075081260228571
Recall is 0.5673978812795781
```

Neural additive models (NAMs)

Moreover, We have used a Neural network called **NAM** for predicting the target feature.NAM is a library for the study of generalized additive models. A portion of the expressivity of DNNs and the inherent understandability of generalized additive models are combined in neural additive models (NAMs). A linear combination of neural networks that focus on different input features is what NAMs learn. These networks can learn any complicated correlations between their input feature and the output since they are simultaneously taught.



Source: NAM paper [1]

Link to the NAM library:https://github.com/google-research/google-research/tree/master/neural_additive_models

We have used NAMs. Starting from raw data, we have made the following changes:

- Choosing price and category_id and predicting is_purchased.
- Dropping duplicates and null values.Adding a new column is_purchased? from the event_type column.Making string indexing for brand columns.
- Passing model parameters to the NAM classifier.
- Fitting and predicting the target.

ROC AUC score= 50%

```
[52] y_pred_prop = model.predict_proba(X_test)
     sk_metrics.roc_auc_score(y_test, y_pred_prop)

     0.5
```

vi. Any Enhancement and future work

There are several enhancements and future work that could be done to improve the solution:

The resulting accuracy was 57 % which is not high but acceptable and could be further improved by adding more refined features , and using MLP networks.We have tried but yielded lower accuracies.

Incorporating additional features: The model could be further improved by incorporating additional features such as product reviews and ratings. This could provide additional insights into customer behavior and help the model make more accurate predictions.

Handling time-series data: The model could be adapted to handle time-series data in order to analyze consumer behavior over time. This could provide additional insights into how customer behavior changes over time and help identify trends and patterns.

Incorporating unstructured data: The model could be enhanced by incorporating unstructured data such as customer reviews, social media posts, and other text data. This could provide additional insights into customer sentiment and help the model make more accurate predictions.

Ensemble methods: The model performance could be improved by combining multiple models using ensemble methods such as bagging and boosting.

Deploying the model: The model could be deployed as a web service, to allow for real-time predictions on new data. This could be done using PySpark, Flask, and Kubernetes.

Automating the pipeline: The pipeline could be automated using tools such as Apache Airflow and AWS Glue. This could help make the process more efficient and reduce the time it takes to deploy new models.

Model interpretability: The model could be made more interpretable by using techniques such as SHAP and LIME, which can help explain the model's predictions and decisions.

Overall, there are many potential enhancements that could be made to the solution to improve its performance and functionality. These enhancements could help to make the model more accurate, more efficient, and more interpretable, which would ultimately lead to better insights and more informed business decisions.

References:

1. R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, R. Caruana, and G. Hinton, “Neural Additive Models: Interpretable machine learning with neural nets,” *arXiv.org*, 24-Oct-2021. [Online]. Available: https://arxiv.org/abs/2004.13912. [Accessed: 10-Jan-2023].