Customer Segmentation Platform Analysis

Sauti.

BY: KYLE KEHOE, ANOUSHKA NAYAK, PRIYANSHA RASTOGI, JUSTIN CHOW

ANALYTICS TEAM

KYLE
OMSBA
Spring 2023

ANOUSHKA
OMSBA
Spring 2023

PRIYANSHA
OMSBA
Spring 2023

JUSTIN
OMSBA
Fall 2022

TABLE OF CONTENTS

01

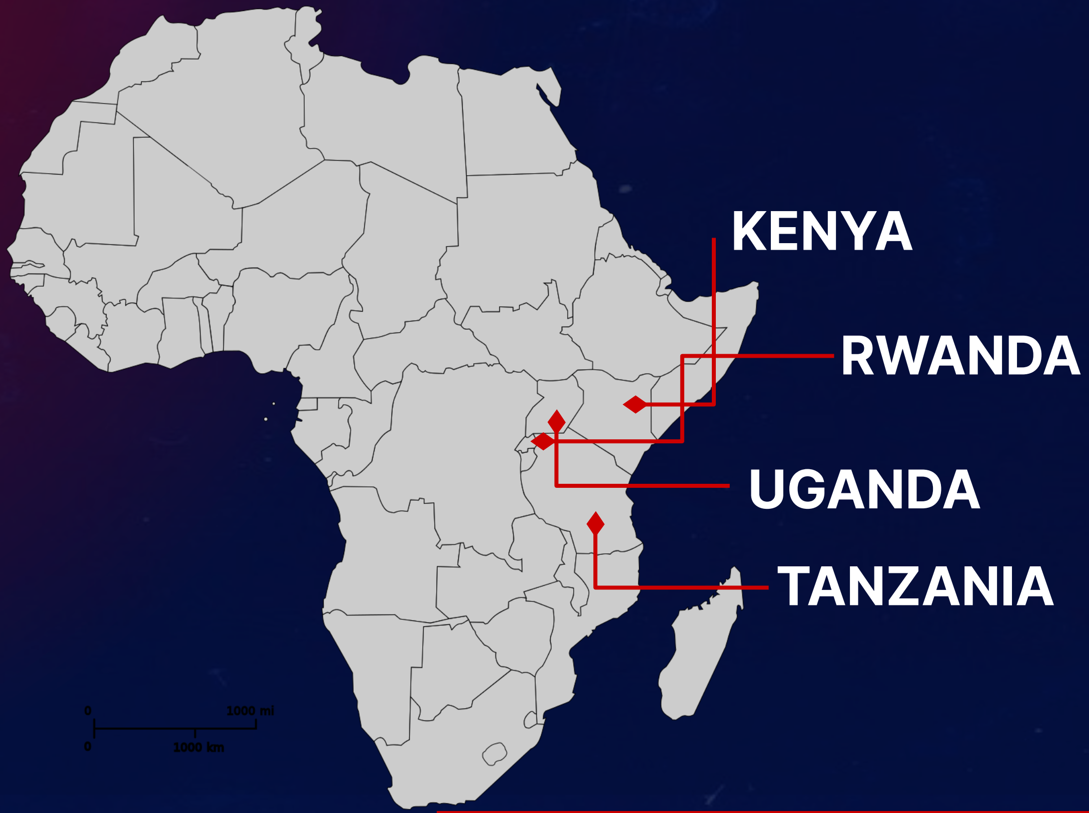
Objective

TEAM OBJECTIVE

Better understanding the analytical context of Sauti, our team objective is to better describe and understand the past and current customer behavior across their user base. Key demographics include the following:

- Gender
- Age
- Occupation
- Education
- Etc.

SAUTI EAST AFRICA REGIONS



02

Data Architecture

DATA PACKAGE CONTENTS I

PLATFORM_SESSIONS

- **sess_id** : Unique field that tracks user sessions
- **cell_num_id** : User phone number
- **created_date** : the date the session was initialized
- **udate** : the last session interaction
- **data** : session data collected in php serialized format
- **platform_id** : info platform accessed by the user.

PLATFORM_REQUESTS

- **request_id** : Unique ID
- **request_text** : user submitted input to Sauti
- **udate** : time the record was submitted
- **sess_id** : session request is associated with

DATA PACKAGE CONTENTS II

PLATFORM_REPLIES

- **reply_id**: unique ID
- **request_id**: associated request for which this record is a reply
- **response_id**: record for which the content of the reply
- **udate**: time this record was submitted
- **sess_id**: session this request is associated with

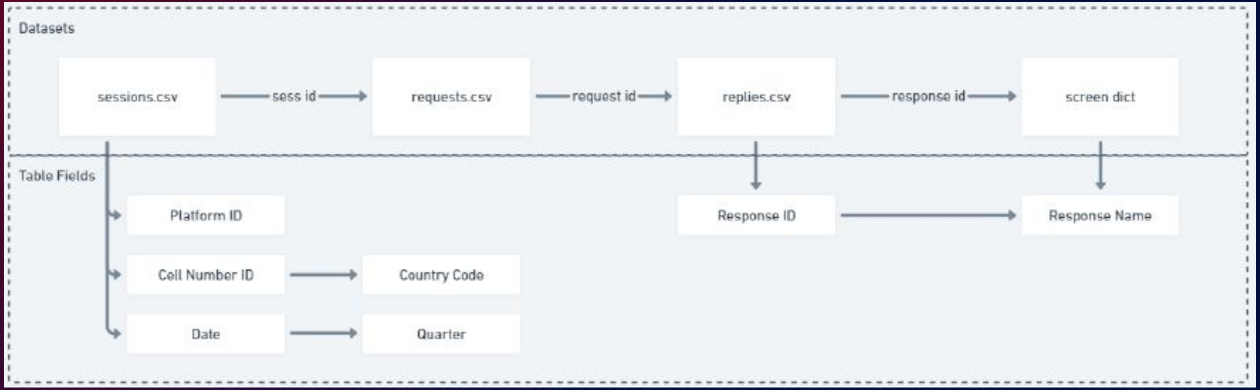
PLATFORM_SCREEN_DICT

- **platform_id**: platform that this response is associated with
- **response_id**
- **response_theme**: theme of our users' response
- **level**: organizational level of this response
- **parent**: parent to this response or top level

PLATFORM_DATABANK

- **cell_num_id**: user phone number
- **sess_id**: session ID from platform sessions where data is recorded from the user inputs
- **key_name**: collection of user-based and session-based key values (e.g., gender, education, crossingfreq, occupation, etc.)
- **value_name**: translated value
- **created_date**: time the value was recorded

DATA ARCHITECTURE WORKFLOW



- **session.csv:** if USER does not have an active Session ID, create new Session ID in table `platform_sessions`
- **request.csv**: Record what the user sent to Sauti in the table `platform_requests`
- **replies.csv**: Calculates the appropriate response to send to user from the table `platform_responses`. Insert into `platform_replies` table.
- **screen_dict.csv:** Response IDs are thematically categorized in the table `platform_screen_dict`.

Data collected in `platform_session_data` is processed, cleaned, and saved into `platform_databank.`

03
Exploratory Data Analysis

EXPLORATORY DATA ANALYSIS

Complete the definition of the following concepts

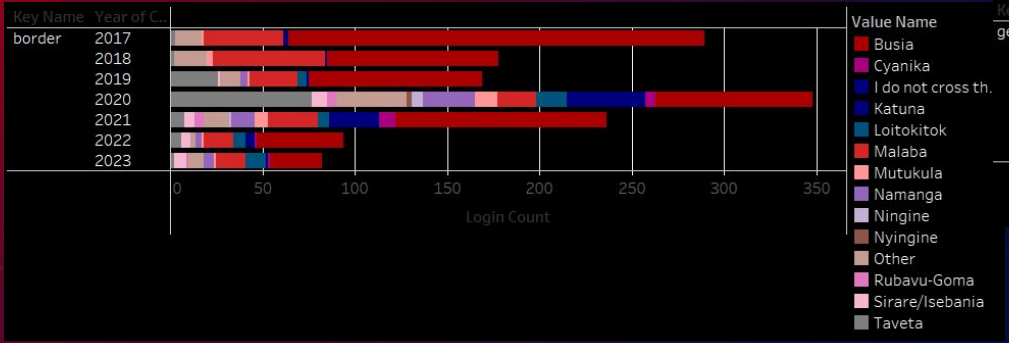
STAGE 1	STAGE 2	STAGE 3	STAGE 4
DATA COLLECTION	**DATA CLEANING**	**ANALYZE**	**VISUALIZE RESULTS**
Collect platform user and session data from Sauti	Cleaning data to gather exact data required for analysis	Analyze and explore aggregate data for conclusive findings	Visualize results in graphic form (e.g., bar, line, pie chart)

User vs Session Key Analysis

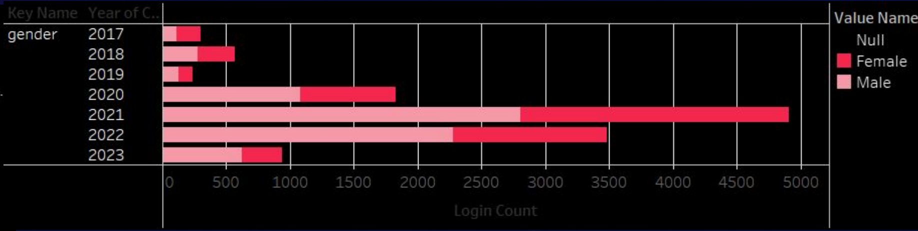
User-based analysis is distribution of the users attributes, while session-based analysis is the user participating in a session per year.

USER-BASED ANALYSIS

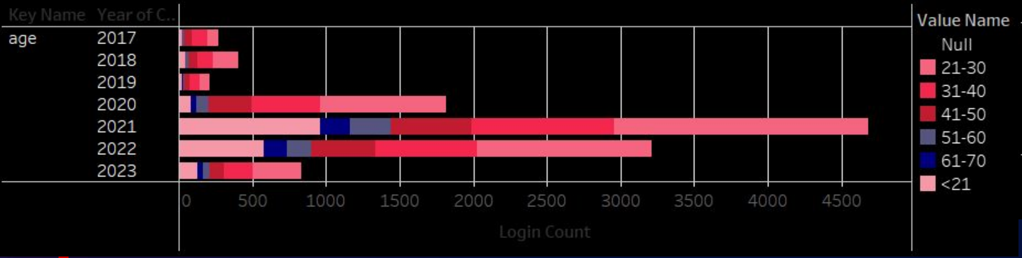
USER SESSIONS VS BORDER



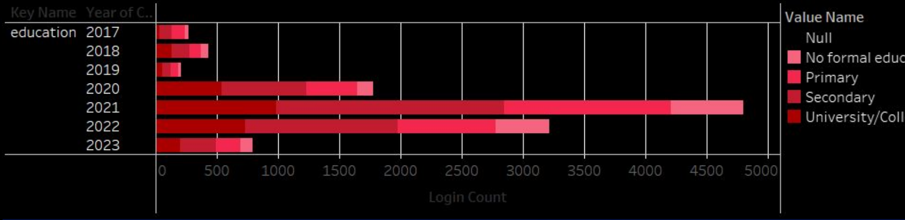
USER SESSIONS VS GENDER (M/F)



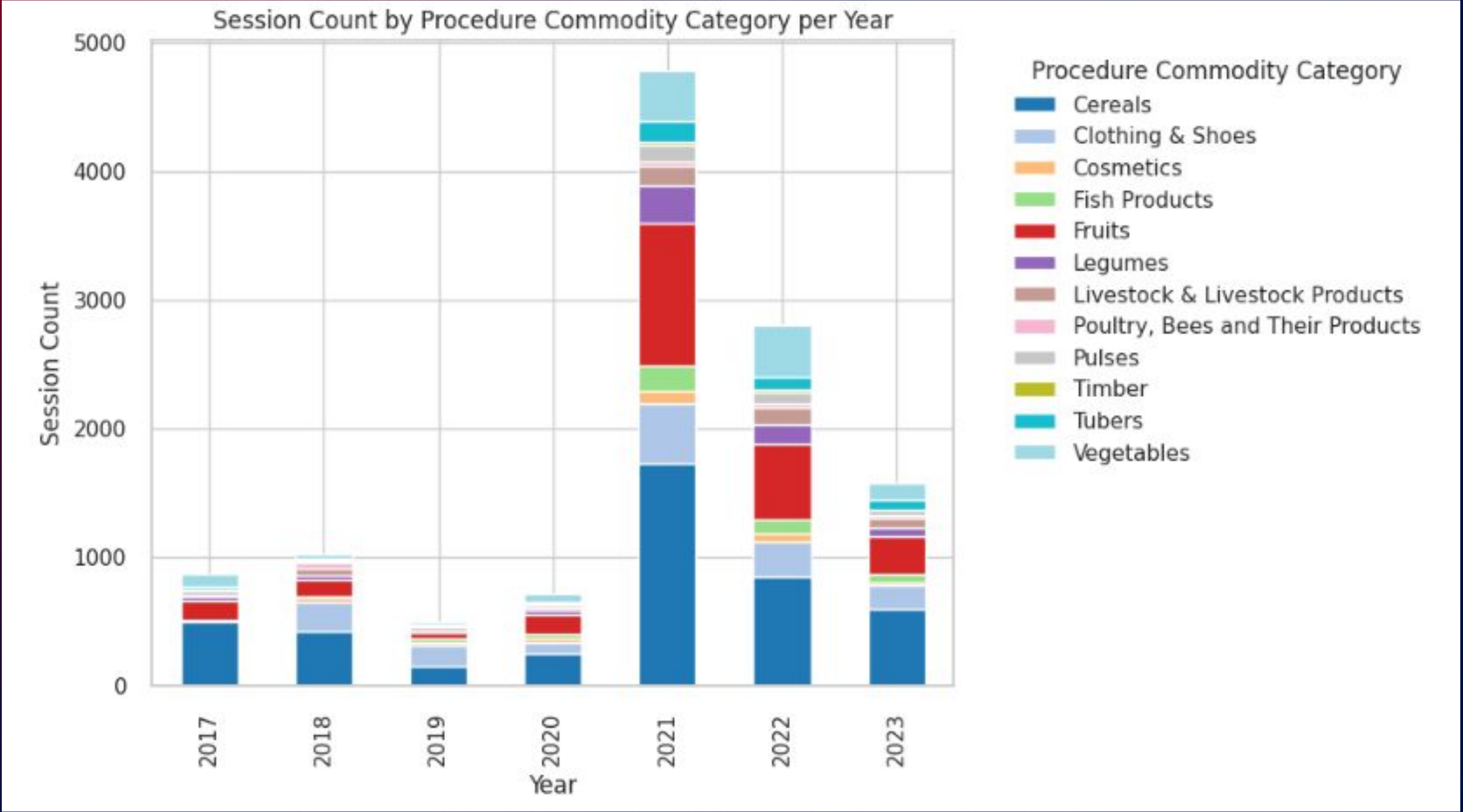
USER SESSION VS AGE RANGE



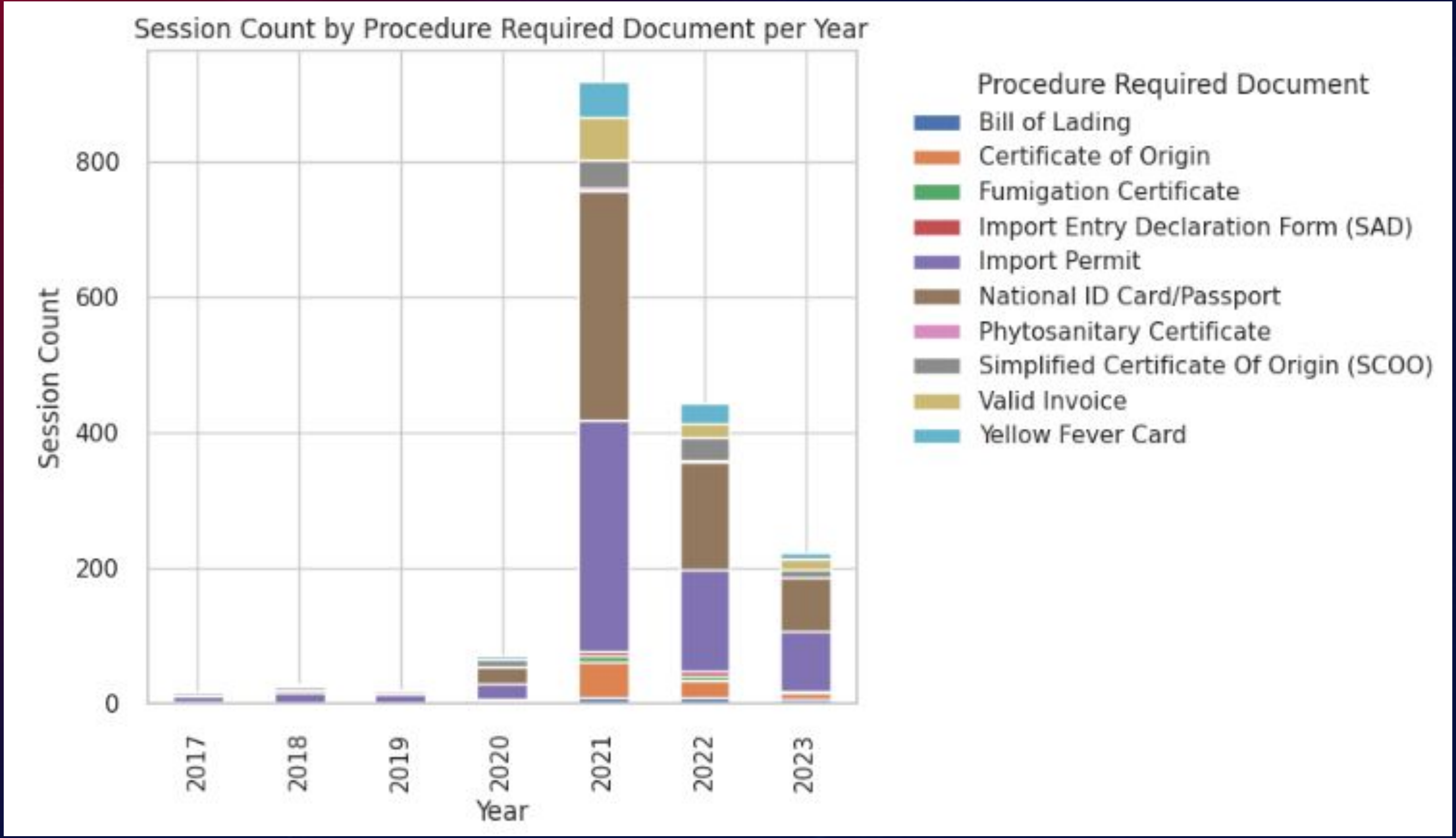
USER SESSION VS EDUCATION LEVEL



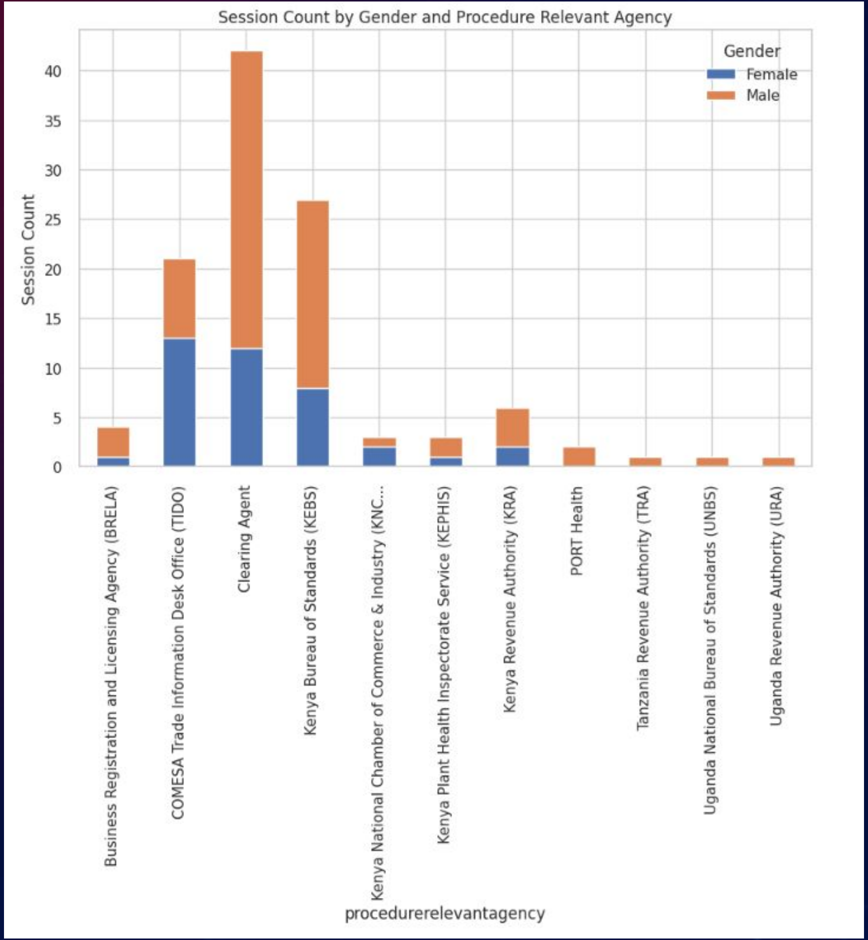
SESSION-BASED ANALYSIS



SESSION-BASED ANALYSIS

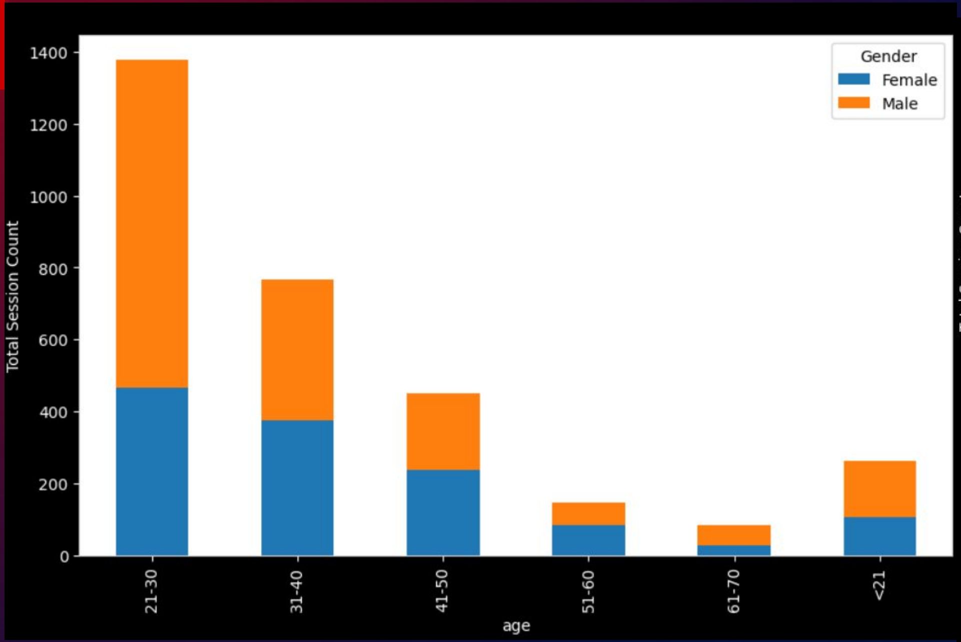


SESSION-BASED ANALYSIS

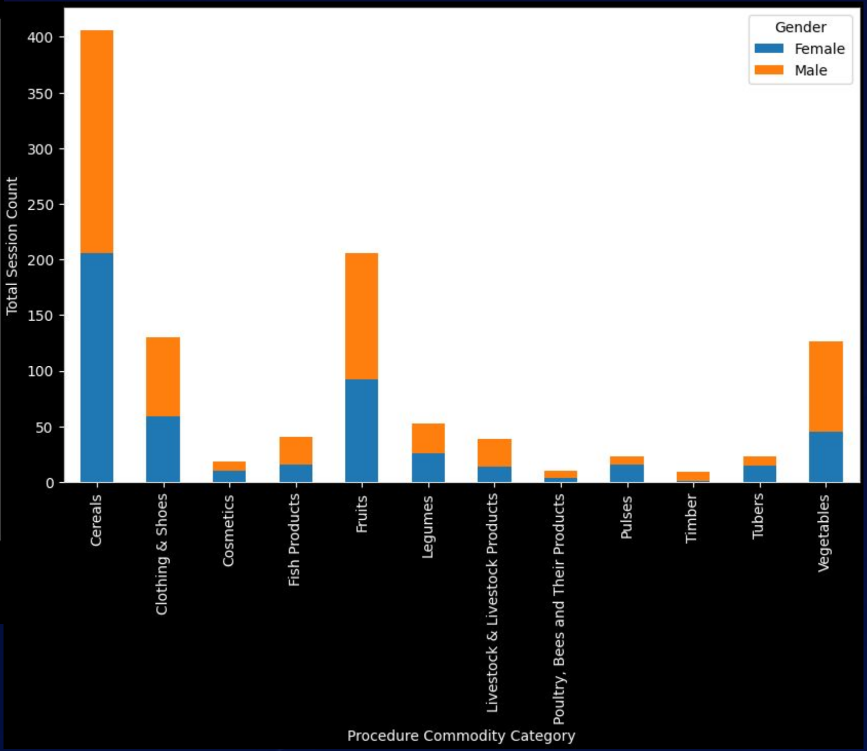


SOCIAL DETERMINANT ANALYSIS

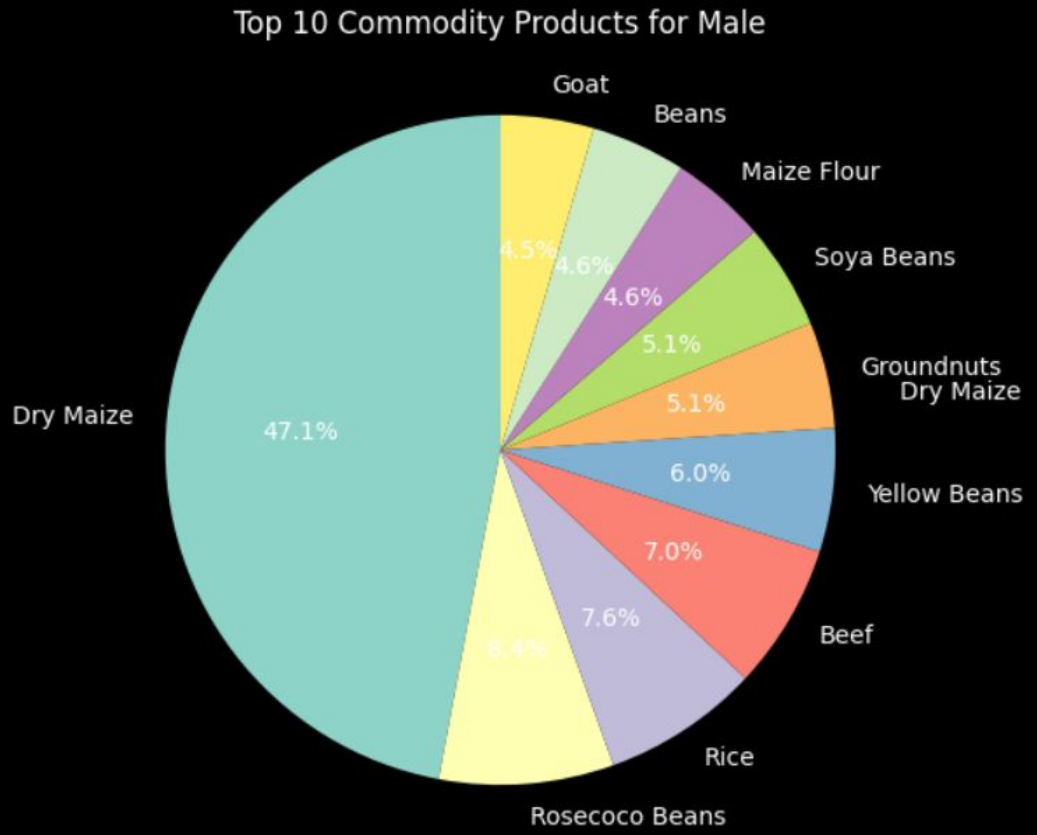
TOTAL SESSIONS VS AGE GROUP

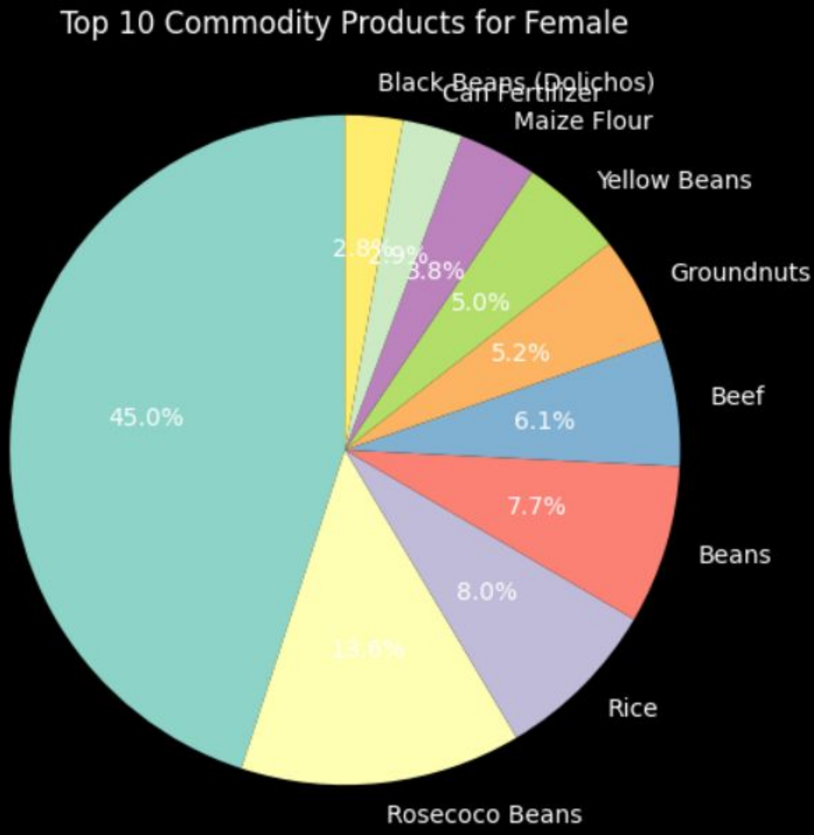


TOTAL SESSIONS VS COMMODITY CATEGORY

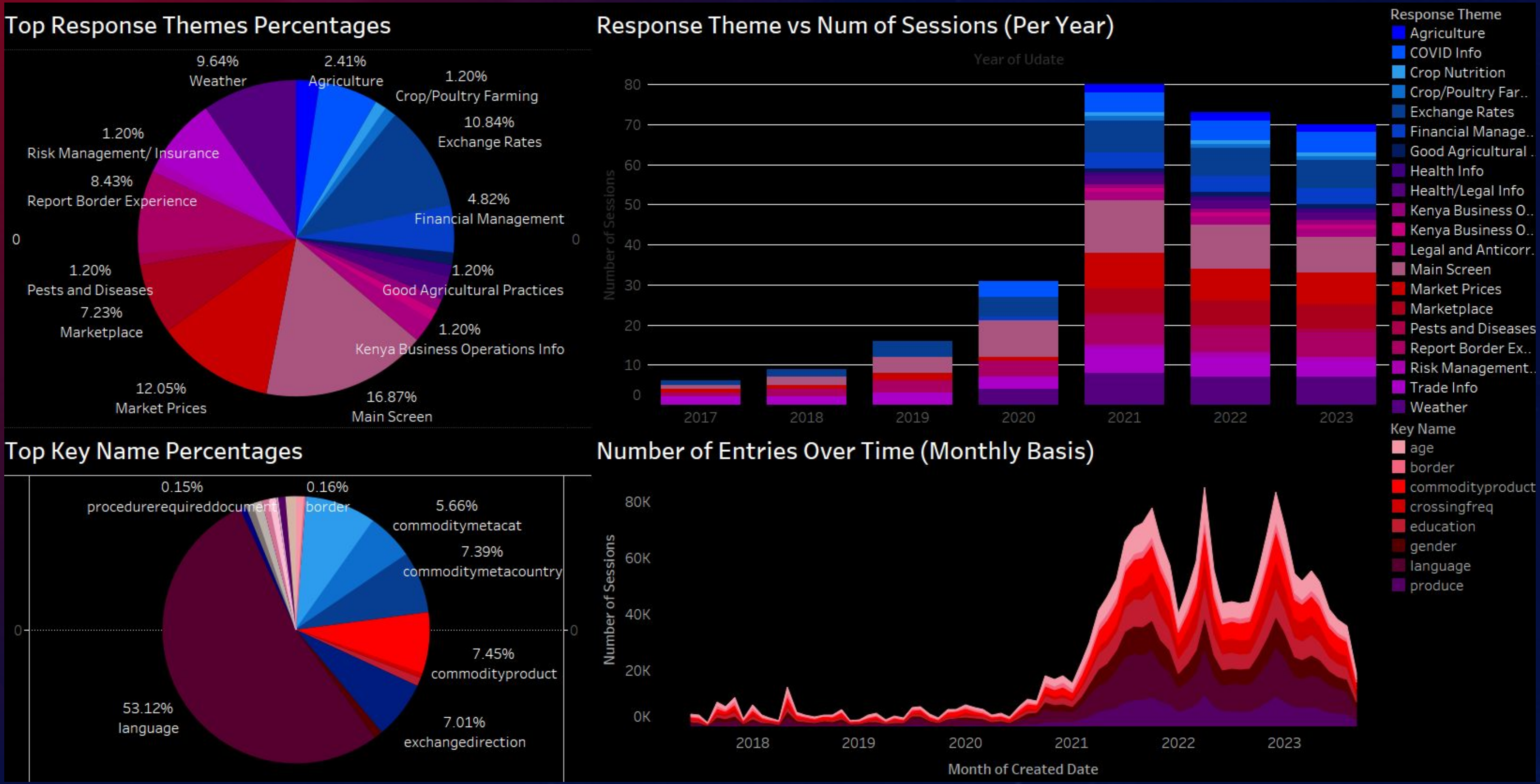


SOCIAL DETERMINANT ANALYSIS





COMPREHENSIVE DASHBOARD



04

Consumer Behavior Forecast: Commodity Meta Categories

Probability Prediction Algorithm for Commodity Meta Category using Random Forest Classifier ML model.

PRE-PROCESSING DATA

Step 1:

Create Pivot Tables to reshape the data based on specific columns, creating a new Dataframe with index as Sess_id,the key name column as columns and value_name column as the values.

Step 2:

Remove duplicate entries based on sess_id and key_name, using the agg function, keeping the first non null value

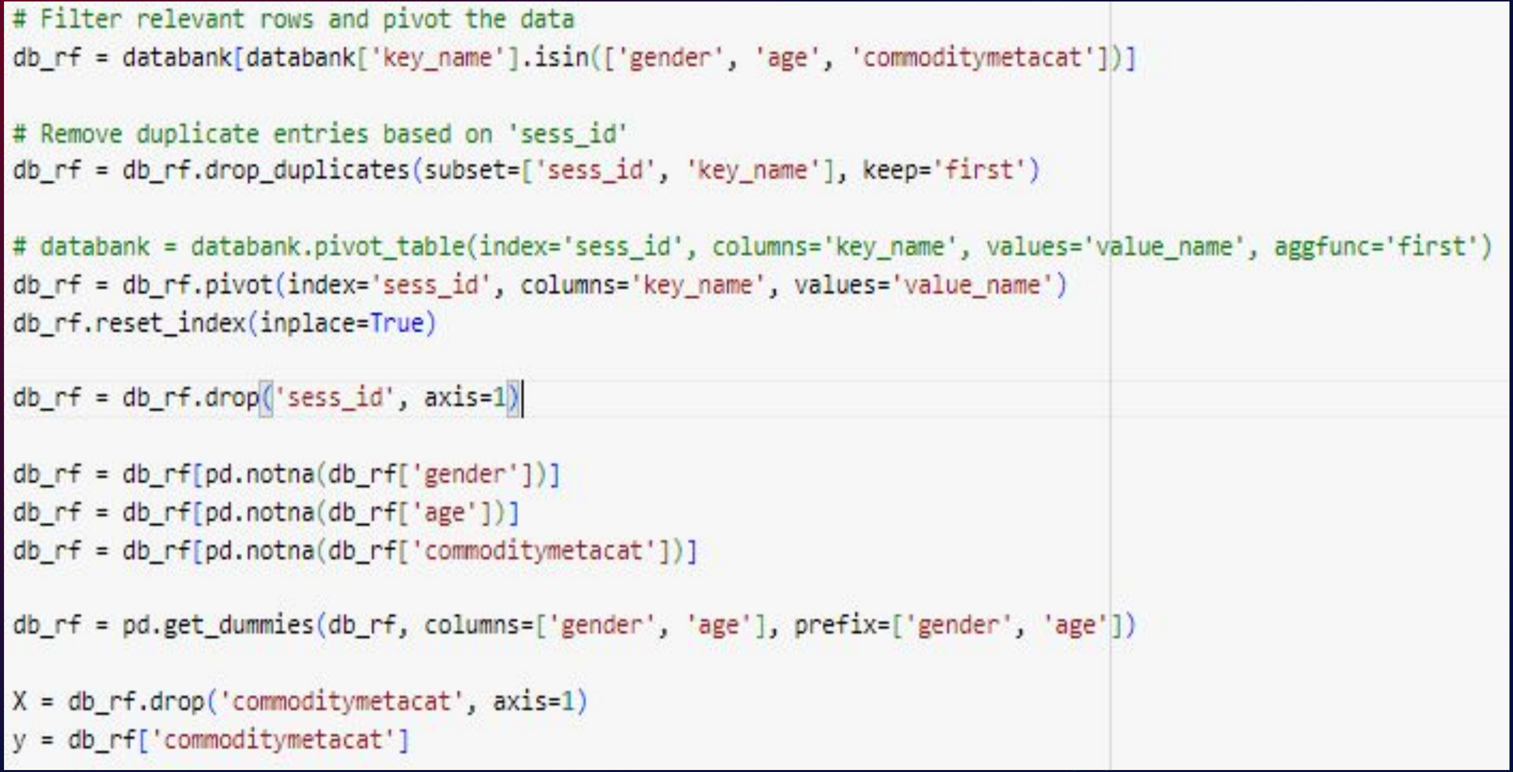
Step 3:

Drop Sess_id column, not needed for analysis for this part.

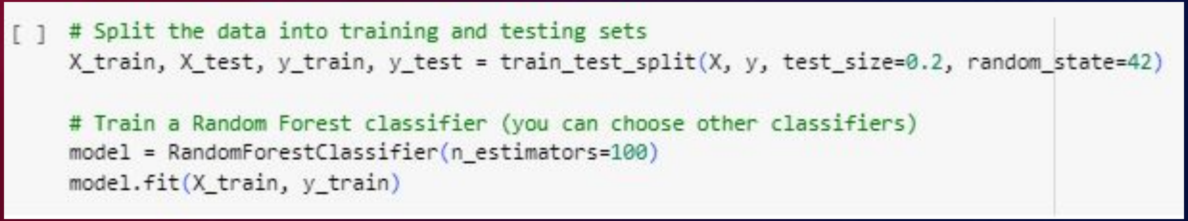
Step 4:

Remove null value rows for gender , age and commodity meta category columns and one hot encode gender and afge colu

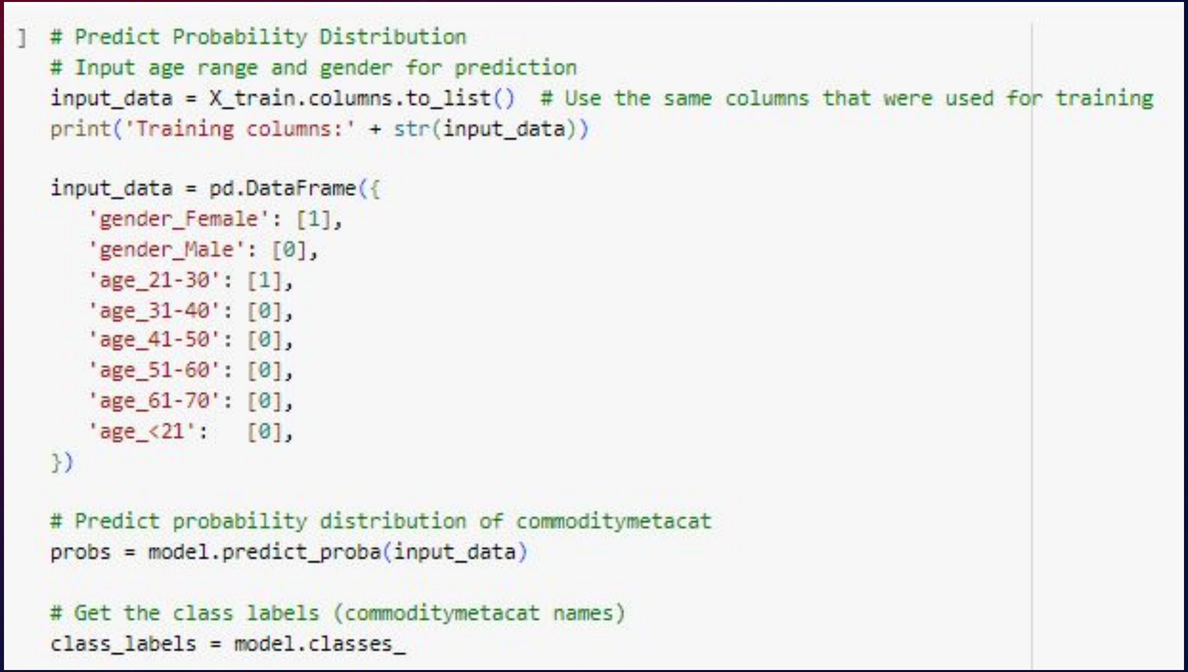
PROBABILITY DISTRIBUTION



```
# Filter relevant rows and pivot the data
db_rf = databank[databank['key_name'].isin(['gender', 'age', 'commoditymetacat'])]

# Remove duplicate entries based on 'sess_id'
db_rf = db_rf.drop_duplicates(subset=['sess_id', 'key_name'], keep='first')

# databank = databank.pivot_table(index='sess_id', columns='key_name', values='value_name', aggfunc='first')
db_rf = db_rf.pivot(index='sess_id', columns='key_name', values='value_name')
db_rf.reset_index(inplace=True)

db_rf = db_rf.drop('sess_id', axis=1)

db_rf = db_rf[pd.notna(db_rf['gender'])]
db_rf = db_rf[pd.notna(db_rf['age'])]
db_rf = db_rf[pd.notna(db_rf['commoditymetacat'])]

db_rf = pd.get_dummies(db_rf, columns=['gender', 'age'], prefix=['gender', 'age'])

X = db_rf.drop('commoditymetacat', axis=1)
y = db_rf['commoditymetacat']
```

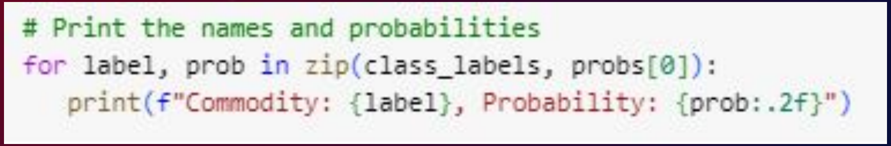


```
[ ] # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Train a Random Forest classifier (you can choose other classifiers)
    model = RandomForestClassifier(n_estimators=100)
    model.fit(X_train, y_train)
```

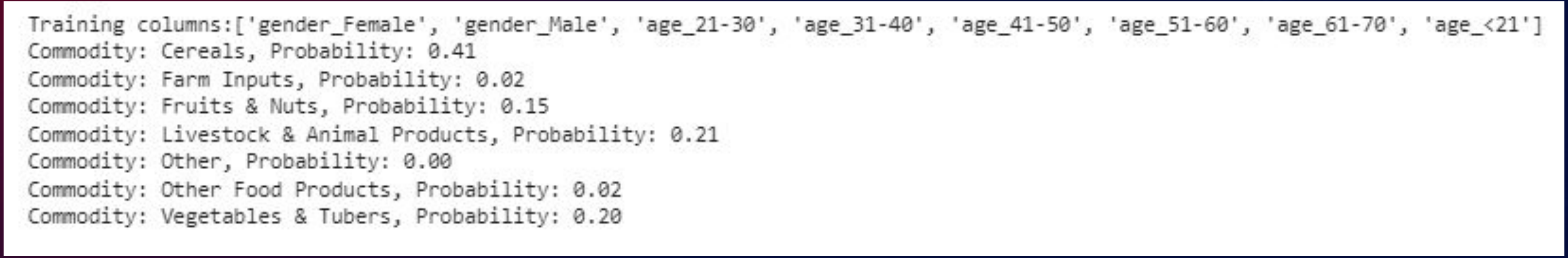


```
] # Predict Probability Distribution
  # Input age range and gender for prediction
  input_data = X_train.columns.to_list()  # Use the same columns that were used for training
  print('Training columns:' + str(input_data))

  input_data = pd.DataFrame({
     'gender_Female': [1],
     'gender_Male': [0],
     'age_21-30': [1],
     'age_31-40': [0],
     'age_41-50': [0],
     'age_51-60': [0],
     'age_61-70': [0],
     'age_<21':   [0],
  })

  # Predict probability distribution of commoditymetacat
  probs = model.predict_proba(input_data)

  # Get the class labels (commoditymetacat names)
  class_labels = model.classes_
```

PREDICTION RESULTS



```
# Print the names and probabilities
for label, prob in zip(class_labels, probs[0]):
   print(f"Commodity: {label}, Probability: {prob:.2f}")
```



```
Training columns:['gender_Female', 'gender_Male', 'age_21-30', 'age_31-40', 'age_41-50', 'age_51-60', 'age_61-70', 'age_<21']
Commodity: Cereals, Probability: 0.41
Commodity: Farm Inputs, Probability: 0.02
Commodity: Fruits & Nuts, Probability: 0.15
Commodity: Livestock & Animal Products, Probability: 0.21
Commodity: Other, Probability: 0.00
Commodity: Other Food Products, Probability: 0.02
Commodity: Vegetables & Tubers, Probability: 0.20
```

05 Demographic Clustering for Customer Segmentation

Preprocess the data with an LLM, run the embeddings through K-means, finally use Light GBM to explain each cluster

PREPROCESSING DATA

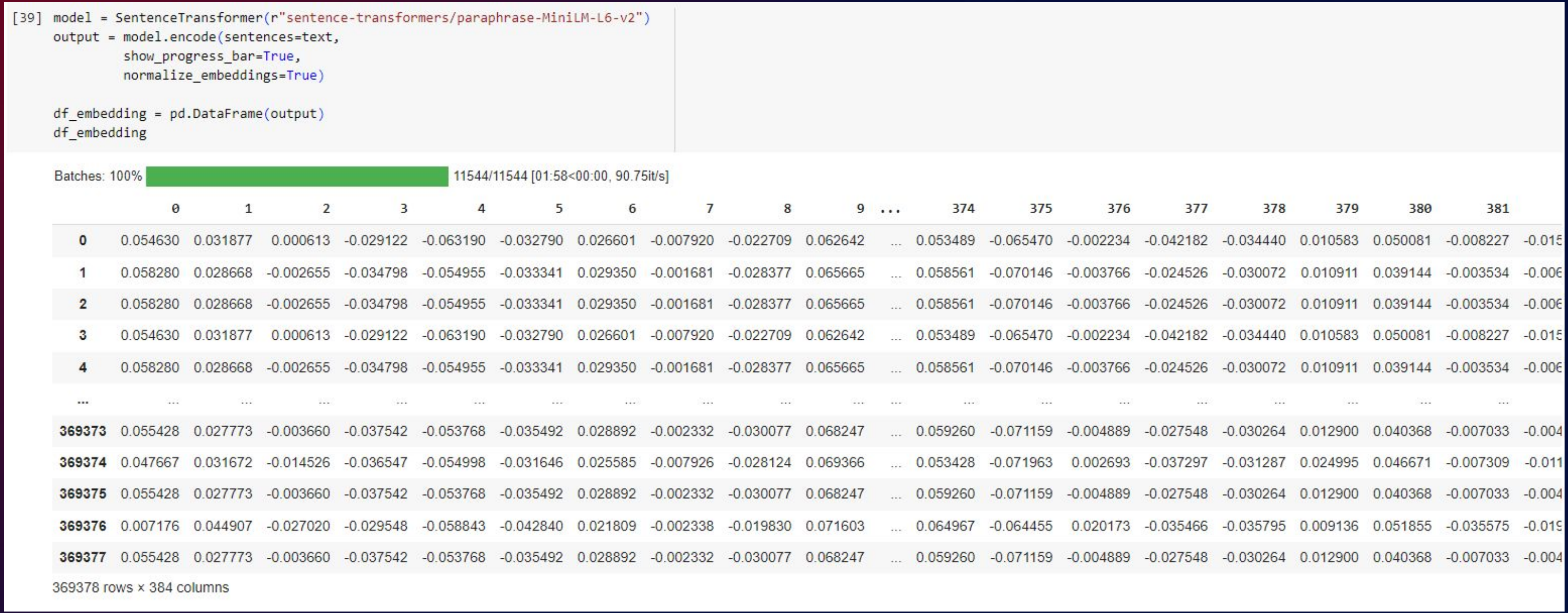
1. Split Key/Value Pairs

- Pivot tables suggests that the data may need to be organized in a tabular format
- The process involves splitting key-value pairs
- Restructure or normalize the data.

2. Convert Text to Embedded

- Reference to the process of representing text data as numerical vectors
- Common natural al language processing tasks such as sentiment analysis, text classification, and language modeling require this step.

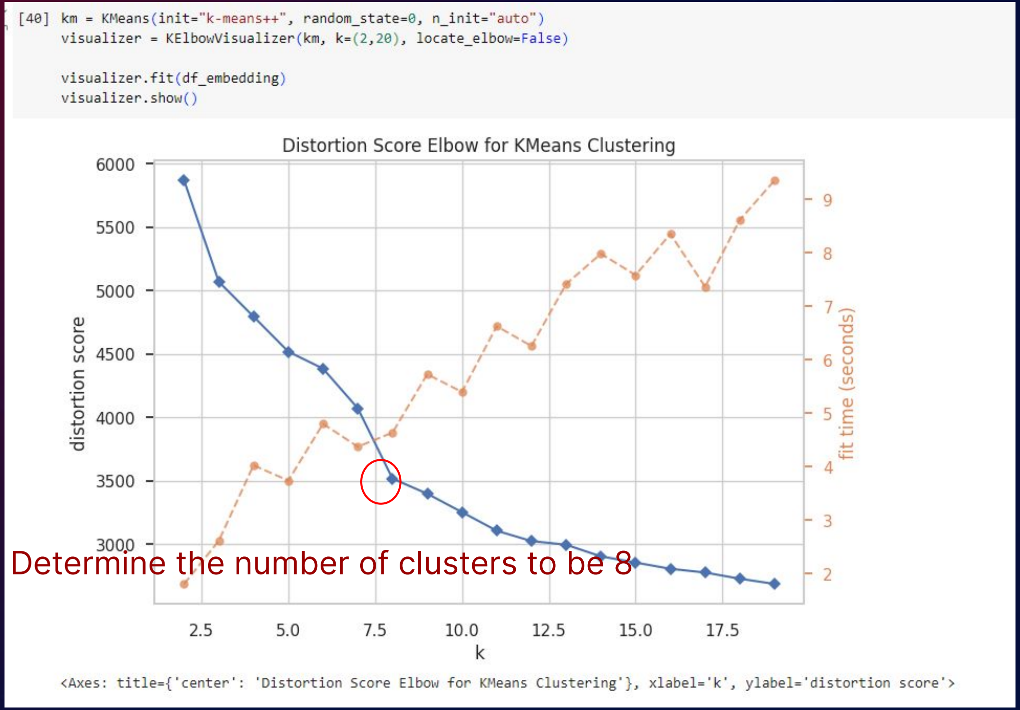
PREPROCESSING DATA



```
[39] model = SentenceTransformer(r"sentence-transformers/paraphrase-MiniLM-L6-v2")
     output = model.encode(sentences=text,
                 show_progress_bar=True,
                 normalize_embeddings=True)

     df_embedding = pd.DataFrame(output)
     df_embedding
```

```
Batches: 100% 11544/11544 [01:58<00:00, 90.75it/s]

               0         1         2         3         4         5         6         7         8         9  ...       374       375       376       377       378       379       380       381
     0  0.054630  0.031877  0.000613 -0.029122 -0.063190 -0.032790  0.026601 -0.007920 -0.022709  0.062642  ...  0.053489 -0.065470 -0.002234 -0.042182 -0.034440  0.010583  0.050081 -0.008227 -0.015
     1  0.058280  0.028668 -0.002655 -0.034798 -0.054955 -0.033341  0.029350 -0.001681 -0.028377  0.065665  ...  0.058561 -0.070146 -0.003766 -0.024526 -0.030072  0.010911  0.039144 -0.003534 -0.006
     2  0.058280  0.028668 -0.002655 -0.034798 -0.054955 -0.033341  0.029350 -0.001681 -0.028377  0.065665  ...  0.058561 -0.070146 -0.003766 -0.024526 -0.030072  0.010911  0.039144 -0.003534 -0.006
     3  0.054630  0.031877  0.000613 -0.029122 -0.063190 -0.032790  0.026601 -0.007920 -0.022709  0.062642  ...  0.053489 -0.065470 -0.002234 -0.042182 -0.034440  0.010583  0.050081 -0.008227 -0.015
     4  0.058280  0.028668 -0.002655 -0.034798 -0.054955 -0.033341  0.029350 -0.001681 -0.028377  0.065665  ...  0.058561 -0.070146 -0.003766 -0.024526 -0.030072  0.010911  0.039144 -0.003534 -0.006
   ...       ...       ...       ...       ...       ...       ...       ...       ...       ...       ...  ...       ...       ...       ...       ...       ...       ...       ...       ...
369373  0.055428  0.027773 -0.003660 -0.037542 -0.053768 -0.035492  0.028892 -0.002332 -0.030077  0.068247  ...  0.059260 -0.071159 -0.004889 -0.027548 -0.030264  0.012900  0.040368 -0.007033 -0.004
369374  0.047667  0.031672 -0.014526 -0.036547 -0.054998 -0.031646  0.025585 -0.007926 -0.028124  0.069366  ...  0.053428 -0.071963  0.002693 -0.037297 -0.031287  0.024995  0.046671 -0.007309 -0.011
369375  0.055428  0.027773 -0.003660 -0.037542 -0.053768 -0.035492  0.028892 -0.002332 -0.030077  0.068247  ...  0.059260 -0.071159 -0.004889 -0.027548 -0.030264  0.012900  0.040368 -0.007033 -0.004
369376  0.007176  0.044907 -0.027020 -0.029548 -0.058843 -0.042840  0.021809 -0.002338 -0.019830  0.071603  ...  0.064967 -0.064455  0.020173 -0.035466 -0.035795  0.009136  0.051855 -0.035575 -0.019
369377  0.055428  0.027773 -0.003660 -0.037542 -0.053768 -0.035492  0.028892 -0.002332 -0.030077  0.068247  ...  0.059260 -0.071159 -0.004889 -0.027548 -0.030264  0.012900  0.040368 -0.007033 -0.004

369378 rows × 384 columns
```

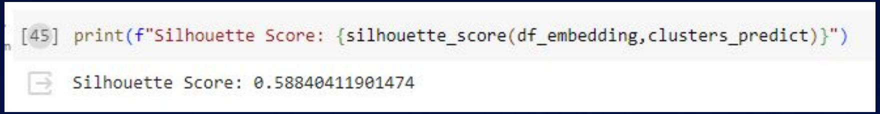
K-MEANS CLUSTERING



```
[40] km = KMeans(init="k-means++", random_state=0, n_init="auto")
     visualizer = KElbowVisualizer(km, k=(2,20), locate_elbow=False)

     visualizer.fit(df_embedding)
     visualizer.show()
```

Determine the number of clusters to be 8

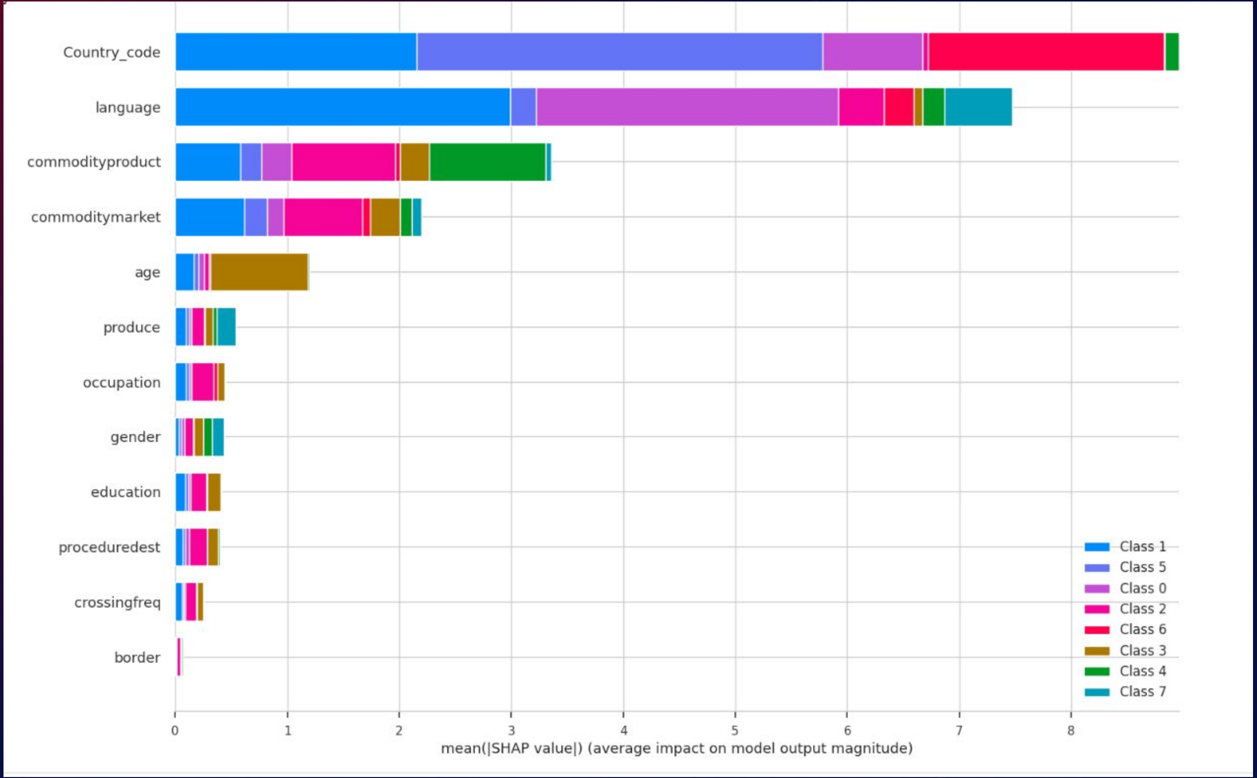
```
<Axes: title={'center': 'Distortion Score Elbow for KMeans Clustering'}, xlabel='k', ylabel='distortion score'>
```



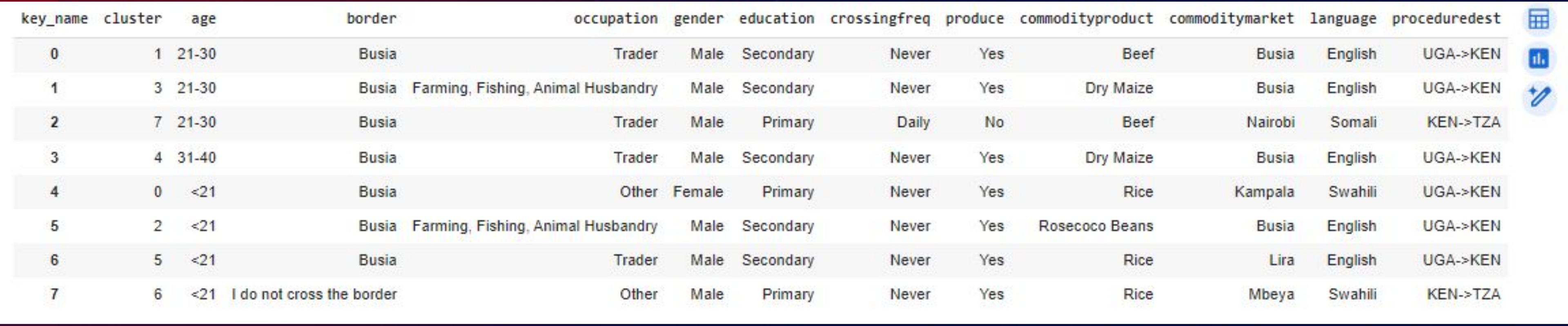
```
[45] print(f"Silhouette Score: {silhouette_score(df_embedding,clusters_predict)}")

Silhouette Score: 0.58840411901474
```

INTERPRETATION

Utilize Light GBM to show the most important variables for determining cluster, and predict the “average” profile in each one



INTERPRETATION (CONT.)

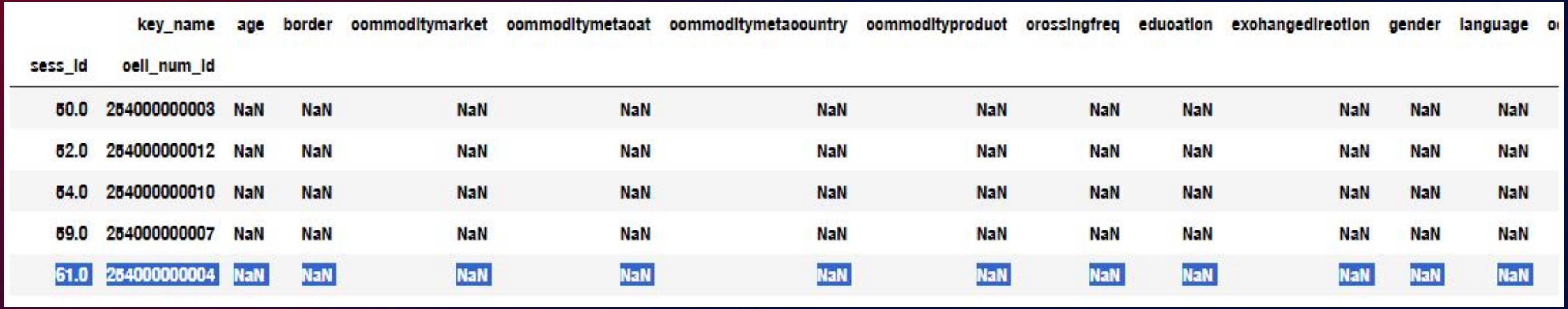


key_name	cluster	age	border	occupation	gender	education	crossingfreq	produce	commodityproduct	commoditymarket	language	proceduredest
0	1	21-30	Busia	Trader	Male	Secondary	Never	Yes	Beef	Busia	English	UGA->KEN
1	3	21-30	Busia	Farming, Fishing, Animal Husbandry	Male	Secondary	Never	Yes	Dry Maize	Busia	English	UGA->KEN
2	7	21-30	Busia	Trader	Male	Primary	Daily	No	Beef	Nairobi	Somali	KEN->TZA
3	4	31-40	Busia	Trader	Male	Secondary	Never	Yes	Dry Maize	Busia	English	UGA->KEN
4	0	<21	Busia	Other	Female	Primary	Never	Yes	Rice	Kampala	Swahili	UGA->KEN
5	2	<21	Busia	Farming, Fishing, Animal Husbandry	Male	Secondary	Never	Yes	Rosecoco Beans	Busia	English	UGA->KEN
6	5	<21	Busia	Trader	Male	Secondary	Never	Yes	Rice	Lira	English	UGA->KEN
7	6	<21	I do not cross the border	Other	Male	Primary	Never	Yes	Rice	Mbeya	Swahili	KEN->TZA

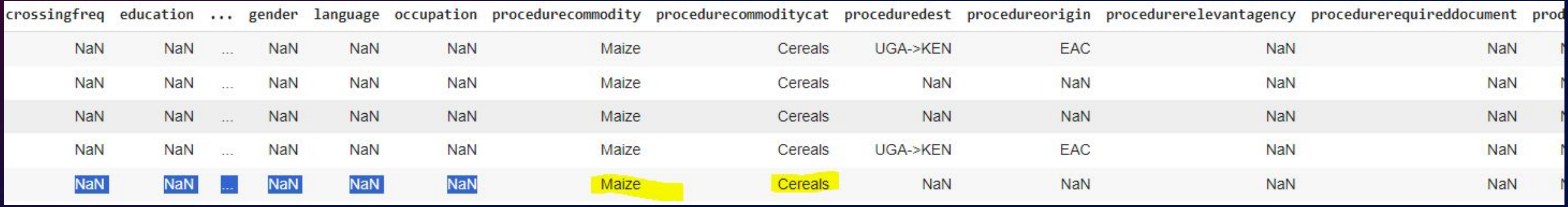
06

Problems Encountered

PROBLEM: MISSING VALUES



	key_name	age	border	oommodltymarket	oommodltymetaoat	oommodltymetaoountry	oommodltyproduot	orossingfreq	eduoatlon	exohangedlreotlon	gender	language	o
sess_Id	oell_num_Id												
50.0	254000000003	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
52.0	254000000012	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
54.0	254000000010	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
59.0	254000000007	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
61.0	254000000004	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	



crossingfreq	education	...	gender	language	occupation	procedurecommodity	procedurecommoditycat	proceduredest	procedureorigin	procedurerelevantagency	procedurerequireddocument	prod
NaN	NaN	...	NaN	NaN	NaN	Maize	Cereals	UGA->KEN	EAC	NaN	NaN	N
NaN	NaN	...	NaN	NaN	NaN	Maize	Cereals	NaN	NaN	NaN	NaN	N
NaN	NaN	...	NaN	NaN	NaN	Maize	Cereals	NaN	NaN	NaN	NaN	N
NaN	NaN	...	NaN	NaN	NaN	Maize	Cereals	UGA->KEN	EAC	NaN	NaN	N
NaN	NaN	...	NaN	NaN	NaN	Maize	Cereals	NaN	NaN	NaN	NaN	N

PROBLEM EXPLANATION

Data Inconsistency

There is inconsistency in the data; for example, the highlighted cells under procedurecommodity and procedurecommoditycat show "Maize" and "Cereals" which might indicate a need for standardization if they are meant to represent similar categories.

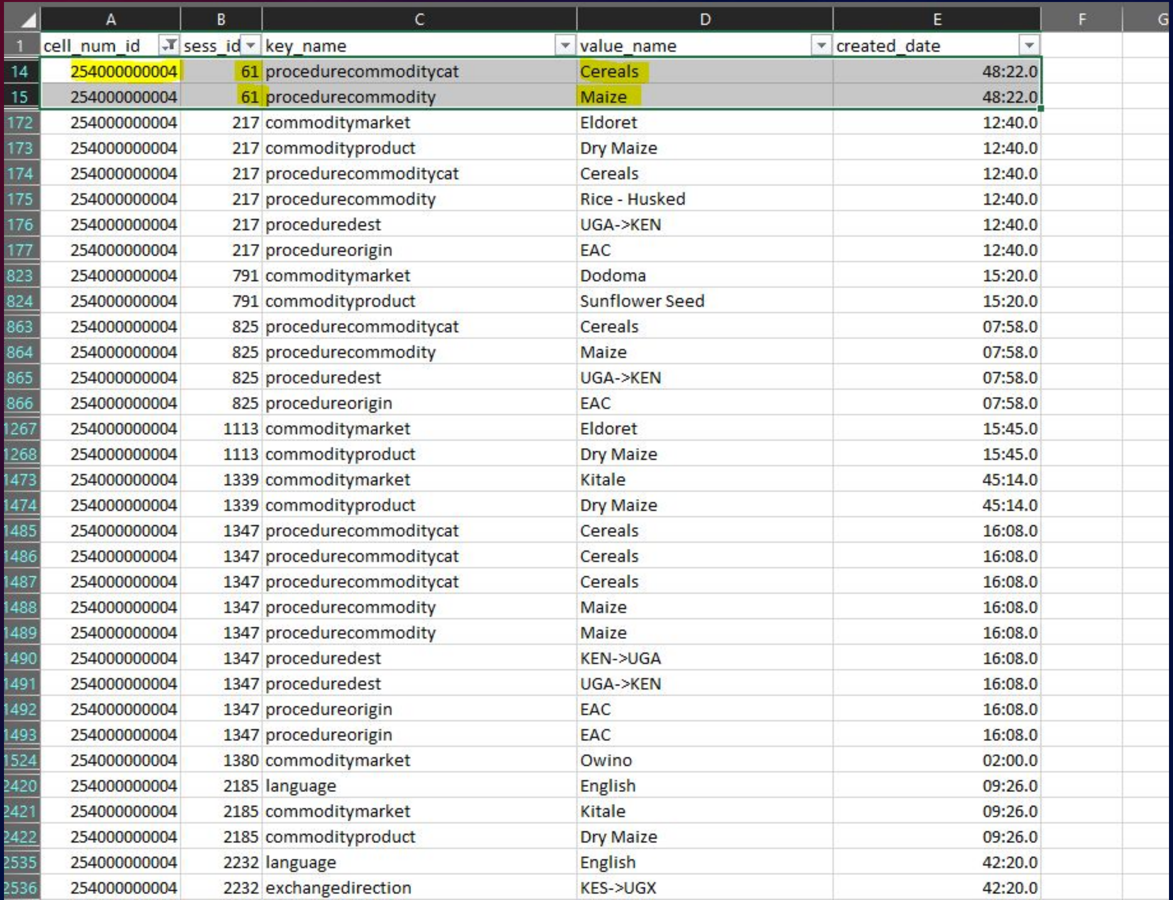
Lack of Identifier Uniqueness

If the sess_id or oell_num_id columns are meant to be unique identifiers, the presence of NaN could be problematic for tracking individual sessions or items.

Categorization Issue

If 'Maize' is a specific type of 'Cereal', then it may be correctly categorized under a broader 'Cereals' category. However, if there are other specific commodities listed that should also fall under 'Cereals' but do not, this could indicate inconsistent categorization.

PROBLEM INVESTIGATION



	cell_num_id	sess_id	key_name	value_name	created_date
14	254000000004	61	procedurecommoditycat	Cereals	48:22.0
15	254000000004	61	procedurecommodity	Maize	48:22.0
172	254000000004	217	commoditymarket	Eldoret	12:40.0
173	254000000004	217	commodityproduct	Dry Maize	12:40.0
174	254000000004	217	procedurecommoditycat	Cereals	12:40.0
175	254000000004	217	procedurecommodity	Rice - Husked	12:40.0
176	254000000004	217	proceduredest	UGA->KEN	12:40.0
177	254000000004	217	procedureorigin	EAC	12:40.0
823	254000000004	791	commoditymarket	Dodoma	15:20.0
824	254000000004	791	commodityproduct	Sunflower Seed	15:20.0
863	254000000004	825	procedurecommoditycat	Cereals	07:58.0
864	254000000004	825	procedurecommodity	Maize	07:58.0
865	254000000004	825	proceduredest	UGA->KEN	07:58.0
866	254000000004	825	procedureorigin	EAC	07:58.0
1267	254000000004	1113	commoditymarket	Eldoret	15:45.0
1268	254000000004	1113	commodityproduct	Dry Maize	15:45.0
1473	254000000004	1339	commoditymarket	Kitale	45:14.0
1474	254000000004	1339	commodityproduct	Dry Maize	45:14.0
1485	254000000004	1347	procedurecommoditycat	Cereals	16:08.0
1486	254000000004	1347	procedurecommoditycat	Cereals	16:08.0
1487	254000000004	1347	procedurecommoditycat	Cereals	16:08.0
1488	254000000004	1347	procedurecommodity	Maize	16:08.0
1489	254000000004	1347	procedurecommodity	Maize	16:08.0
1490	254000000004	1347	proceduredest	KEN->UGA	16:08.0
1491	254000000004	1347	proceduredest	UGA->KEN	16:08.0
1492	254000000004	1347	procedureorigin	EAC	16:08.0
1493	254000000004	1347	procedureorigin	EAC	16:08.0
1524	254000000004	1380	commoditymarket	Owino	02:00.0
2420	254000000004	2185	language	English	09:26.0
2421	254000000004	2185	commoditymarket	Kitale	09:26.0
2422	254000000004	2185	commodityproduct	Dry Maize	09:26.0
2535	254000000004	2232	language	English	42:20.0
2536	254000000004	2232	exchangedirection	KES->UGX	42:20.0

INVESTIGATION EXPLANATION

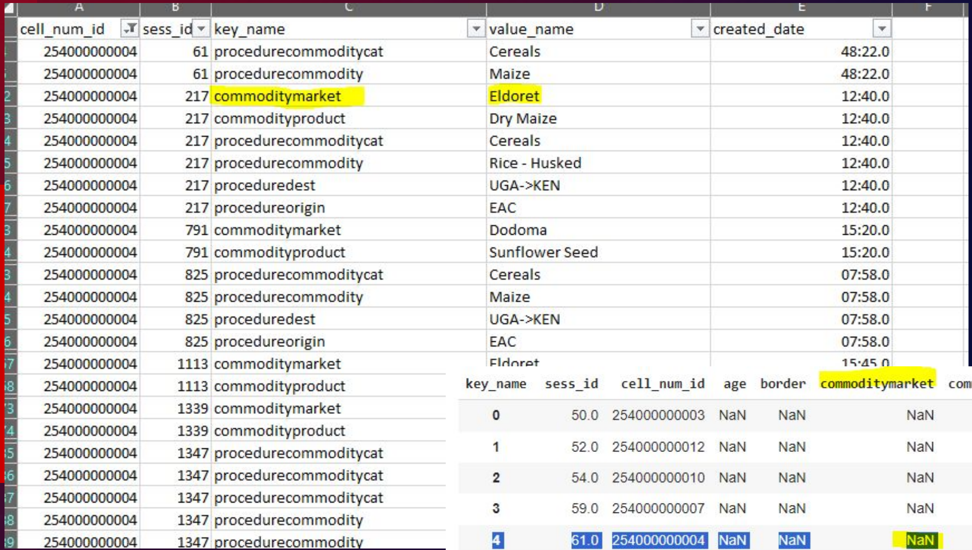
Hierarchy and Classification

'Cereals' appears in the column labeled 'procedurecommoditycat', suggesting it's a category for commodities. 'Maize' appears under 'procedurecommodity', indicating that it is a specific commodity within that category.

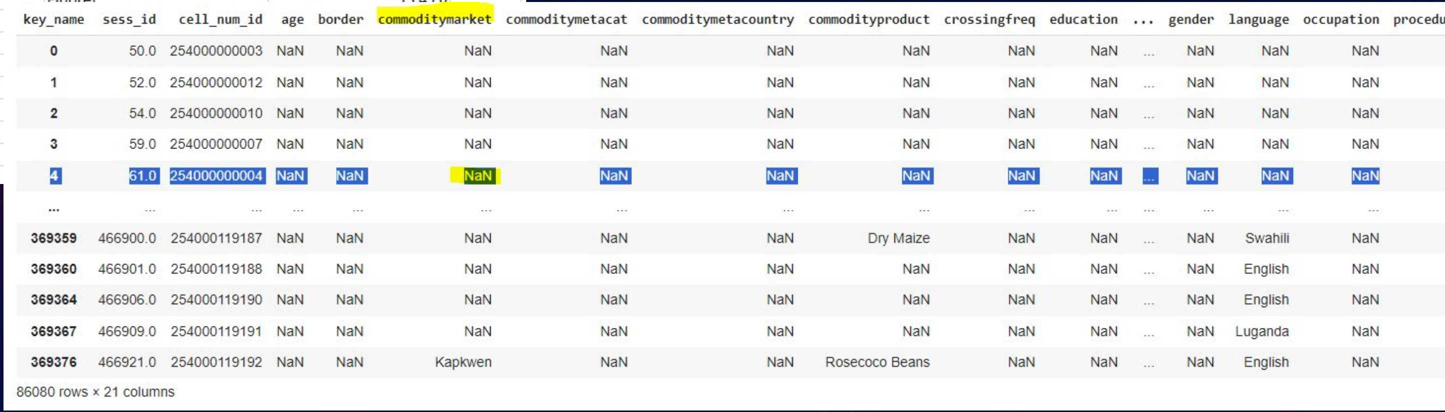
Data Duplication

The repeated appearance of '254000000004' under 'cell_num_id' with different 'procedurecommodity' and 'procedurecommoditycat' entries could indicate that the dataset contains multiple transactions or records for the same entity. This is explained by multiple users with different `sess_id` using the same cell phone.

PROBLEM COMPARISON



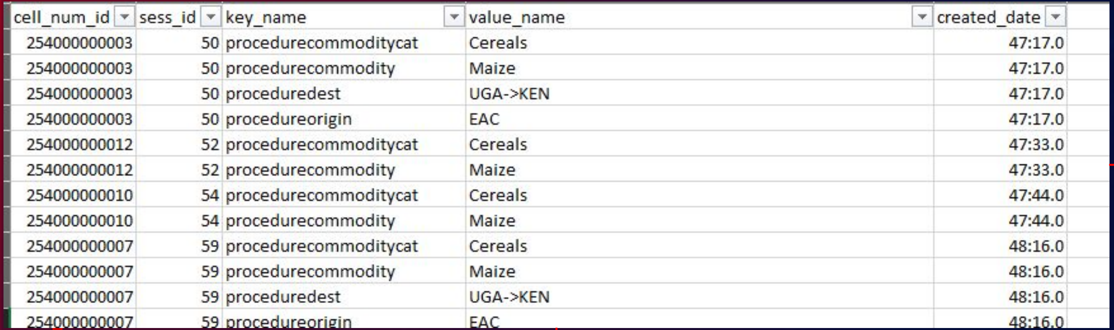
cell_num_id	sess_id	key_name	value_name	created_date
254000000004	61	procedurecommoditycat	Cereals	48:22.0
254000000004	61	procedurecommodity	Maize	48:22.0
254000000004	217	commoditymarket	Eldoret	12:40.0
254000000004	217	commodityproduct	Dry Maize	12:40.0
254000000004	217	procedurecommoditycat	Cereals	12:40.0
254000000004	217	procedurecommodity	Rice - Husked	12:40.0
254000000004	217	proceduredest	UGA->KEN	12:40.0
254000000004	217	procedureorigin	EAC	12:40.0
254000000004	791	commoditymarket	Dodoma	15:20.0
254000000004	791	commodityproduct	Sunflower Seed	15:20.0
254000000004	825	procedurecommoditycat	Cereals	07:58.0
254000000004	825	procedurecommodity	Maize	07:58.0
254000000004	825	proceduredest	UGA->KEN	07:58.0
254000000004	825	procedureorigin	EAC	07:58.0
254000000004	1113	commoditymarket	Eldoret	15:45.0
254000000004	1113	commodityproduct		
254000000004	1339	commoditymarket		
254000000004	1339	commodityproduct		
254000000004	1347	procedurecommoditycat		
254000000004	1347	procedurecommoditycat		
254000000004	1347	procedurecommoditycat		
254000000004	1347	procedurecommodity		
254000000004	1347	procedurecommodity		



key_name	sess_id	cell_num_id	age	border	commoditymarket	commoditymetacat	commoditymetacountry	commodityproduct	crossingfreq	education	...	gender	language	occupation	procedu
0	50.0	254000000003	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
1	52.0	254000000012	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
2	54.0	254000000010	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
3	59.0	254000000007	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
4	61.0	254000000004	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
369359	466900.0	254000119187	NaN	NaN	NaN	NaN	NaN	Dry Maize	NaN	NaN	...	NaN	Swahili	NaN	
369360	466901.0	254000119188	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	English	NaN	
369364	466906.0	254000119190	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	English	NaN	
369367	466909.0	254000119191	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Luganda	NaN	
369376	466921.0	254000119192	NaN	NaN	Kapkwen	NaN	NaN	Rosecoco Beans	NaN	NaN	...	NaN	English	NaN	

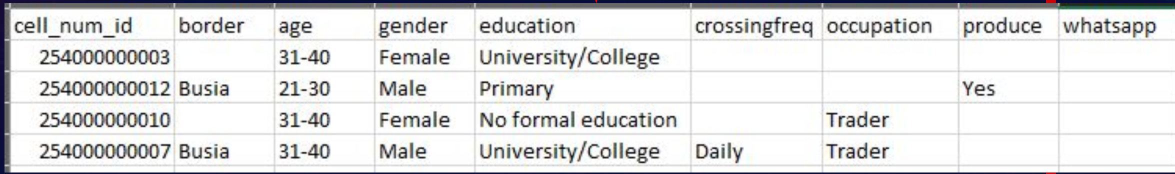
86080 rows × 21 columns

SUGGESTIONS



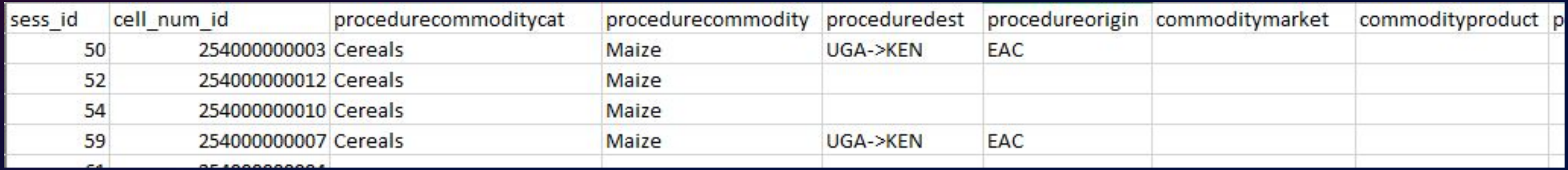
cell_num_id	sess_id	key_name	value_name	created_date
254000000003	50	procedurecommoditycat	Cereals	47:17.0
254000000003	50	procedurecommodity	Maize	47:17.0
254000000003	50	proceduredest	UGA->KEN	47:17.0
254000000003	50	procedureorigin	EAC	47:17.0
254000000012	52	procedurecommoditycat	Cereals	47:33.0
254000000012	52	procedurecommodity	Maize	47:33.0
254000000010	54	procedurecommoditycat	Cereals	47:44.0
254000000010	54	procedurecommodity	Maize	47:44.0
254000000007	59	procedurecommoditycat	Cereals	48:16.0
254000000007	59	procedurecommodity	Maize	48:16.0
254000000007	59	proceduredest	UGA->KEN	48:16.0
254000000007	59	procedureorigin	EAC	48:16.0

Demographics



cell_num_id	border	age	gender	education	crossingfreq	occupation	produce	whatsapp
254000000003		31-40	Female	University/College				
254000000012	Busia	21-30	Male	Primary			Yes	
254000000010		31-40	Female	No formal education		Trader		
254000000007	Busia	31-40	Male	University/College	Daily	Trader		

Session Based



sess_id	cell_num_id	procedurecommoditycat	procedurecommodity	proceduredest	procedureorigin	commoditymarket	commodityproduct	p
50	254000000003	Cereals	Maize	UGA->KEN	EAC			
52	254000000012	Cereals	Maize					
54	254000000010	Cereals	Maize					
59	254000000007	Cereals	Maize	UGA->KEN	EAC			

SUGGESTION EXPLANATION

Session-Based Segmentation

Propose organizing data by `sess_id`, which likely represents a session identifier. We want customer interactions to be tracked in discrete sessions, and have each session associated with specific actions or transactions (e.g., choosing a commodity like 'Maize'). Segmenting customers by sessions can provide insights into customer behavior within individual interactions with the system or service.

Linking Demographic to Sessions

Connect demographic information to the `cell_num_id` to serve as an unique identifier for each customer or transaction. By linking demographics such as age, gender, education, border (which might indicate the crossing point for a transaction), crossingfreq (how often they cross the border), occupation, produce (the type of goods being handled), and usage of whatsapp, a richer profile of each session is established.

07 Enhancing Customer Segmentation: Demographic and Session Data

Predicting the response_theme based on the input features from the cleaned demographic and session data

PREPROCESSING DATA

1. Split Key/Value Pairs

- Pivot tables suggests that the data may need to be organized in a tabular format
- The process involves splitting key-value pairs
- Restructure or normalize the data.

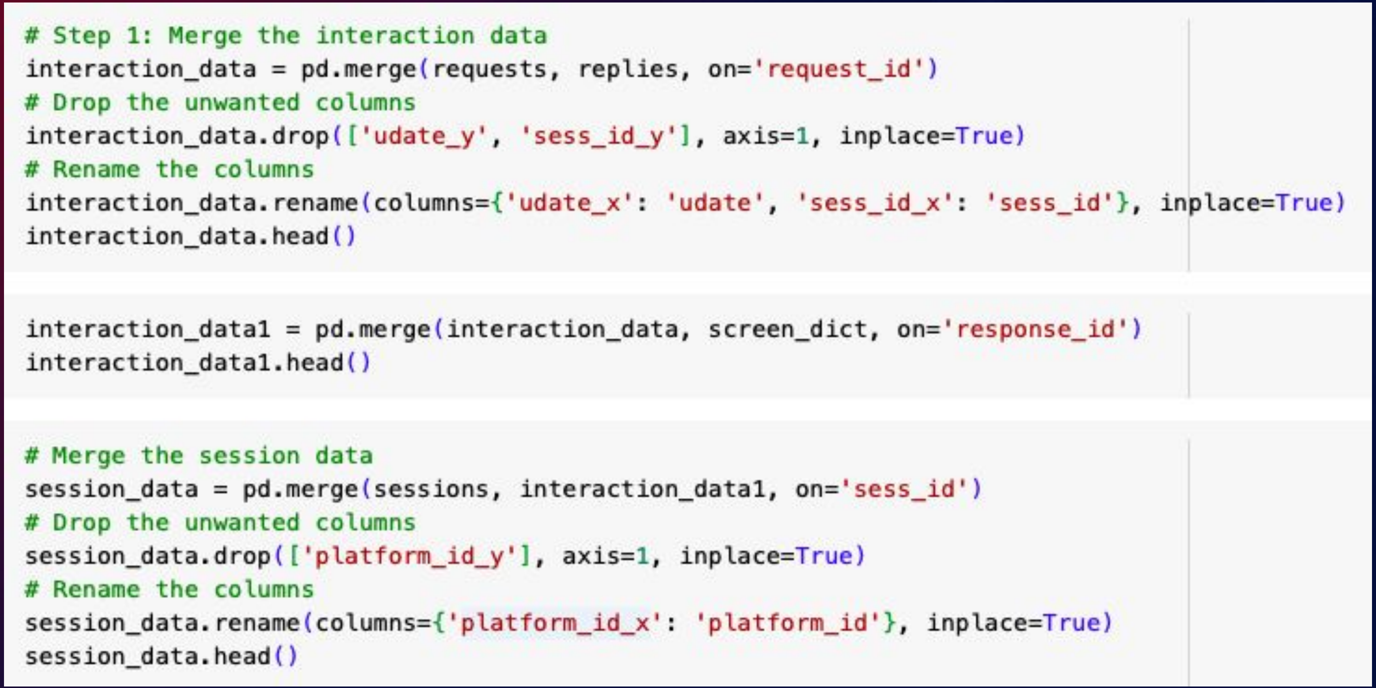
2. Redundancy Eliminated

- Merging multiple tables results in duplicate columns that can confuse models or skew results.
- This cleaning process involves removing these redundant columns to streamline the dataset.
- Prevent any potential issues that could arise from having multiple columns with the same data.

3. Utilize Distinct Entries

- Filtering or processing the data to ensure that each value used for making predictions is distinct and non-repetitive.
- Reducing noise to improve the quality of the predictions
- Focus on the most relevant and unique information available.

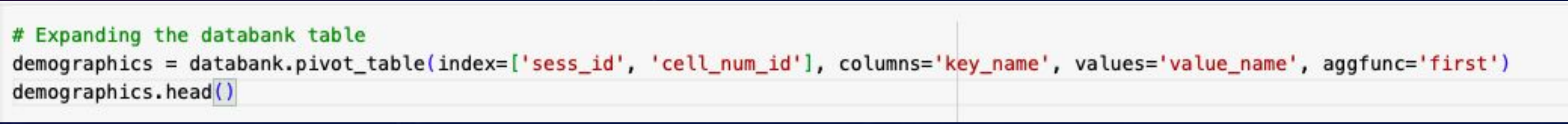
PRE-PROCESSING DATA



```
# Step 1: Merge the interaction data
interaction_data = pd.merge(requests, replies, on='request_id')
# Drop the unwanted columns
interaction_data.drop(['udate_y', 'sess_id_y'], axis=1, inplace=True)
# Rename the columns
interaction_data.rename(columns={'udate_x': 'udate', 'sess_id_x': 'sess_id'}, inplace=True)
interaction_data.head()
```

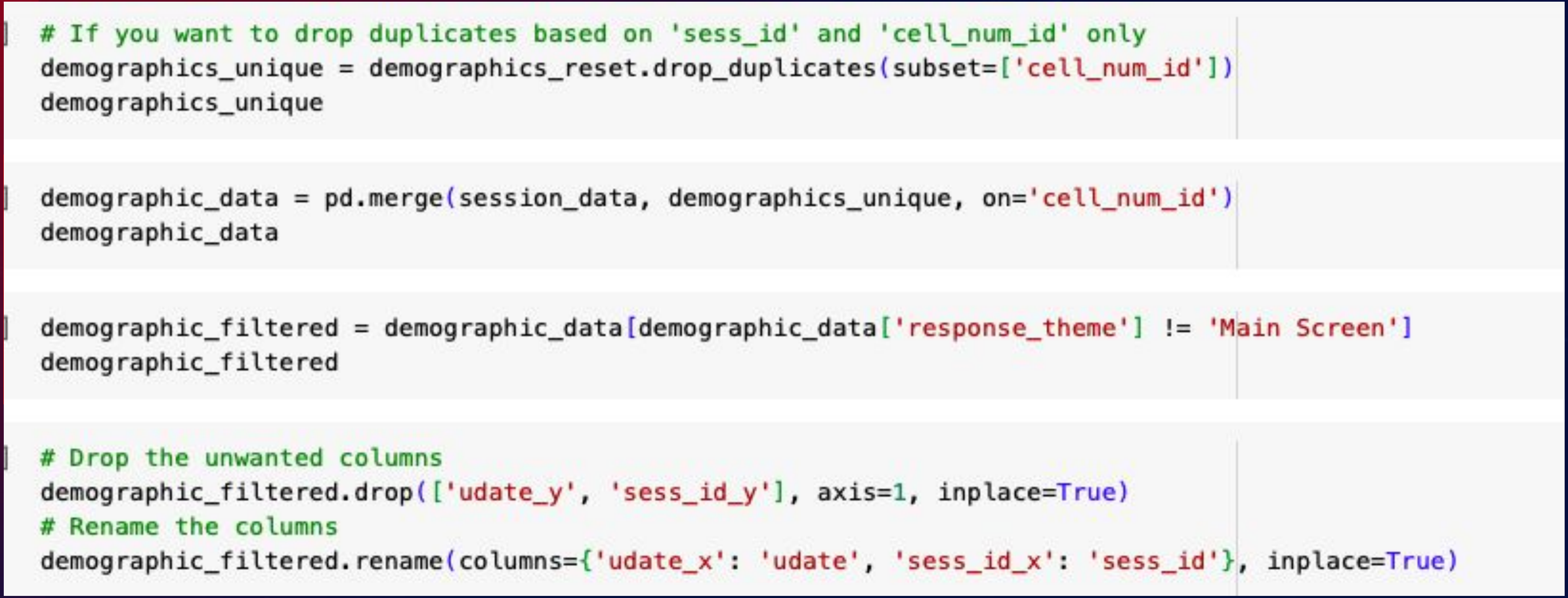
```
interaction_data1 = pd.merge(interaction_data, screen_dict, on='response_id')
interaction_data1.head()
```

```
# Merge the session data
session_data = pd.merge(sessions, interaction_data1, on='sess_id')
# Drop the unwanted columns
session_data.drop(['platform_id_y'], axis=1, inplace=True)
# Rename the columns
session_data.rename(columns={'platform_id_x': 'platform_id'}, inplace=True)
session_data.head()
```



```
# Expanding the databank table
demographics = databank.pivot_table(index=['sess_id', 'cell_num_id'], columns='key_name', values='value_name', aggfunc='first')
demographics.head()
```

PRE-PROCESSING DATA



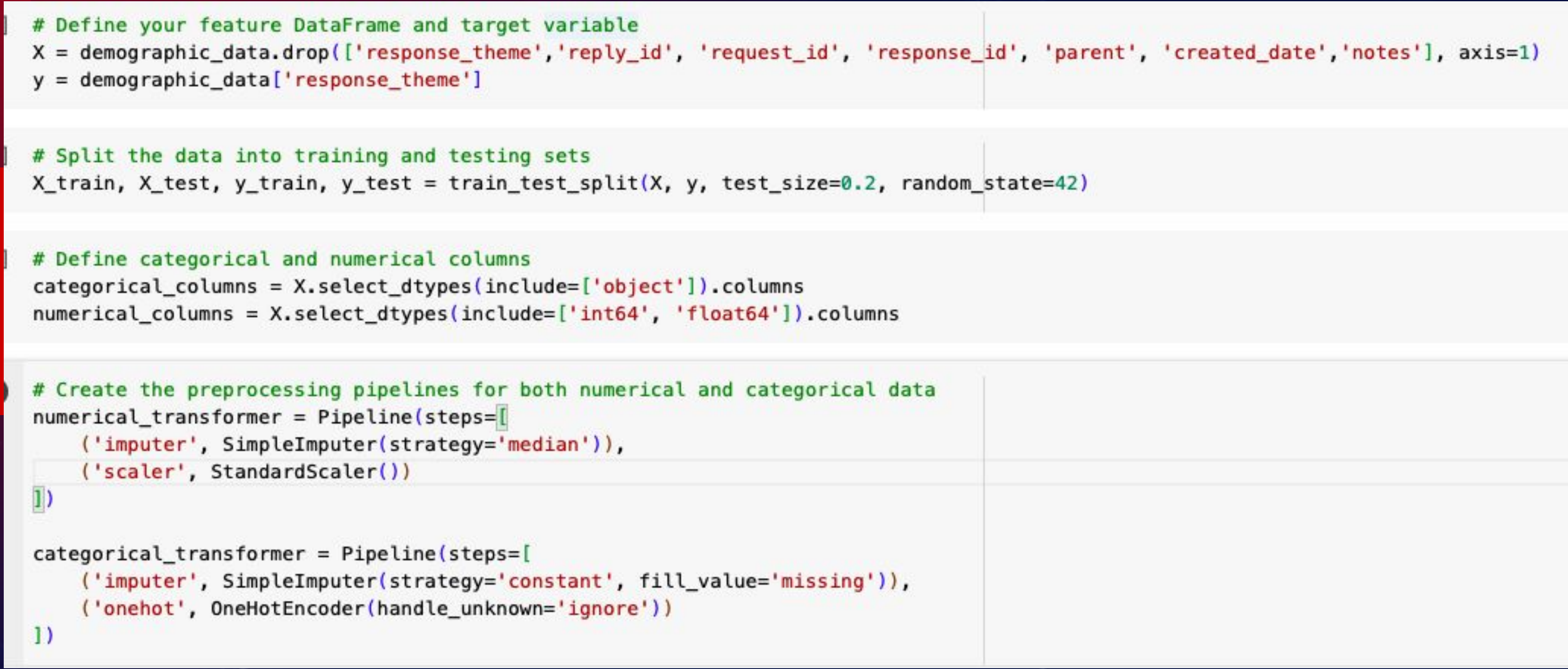
```
# If you want to drop duplicates based on 'sess_id' and 'cell_num_id' only
demographics_unique = demographics_reset.drop_duplicates(subset=['cell_num_id'])
demographics_unique
```

```
demographic_data = pd.merge(session_data, demographics_unique, on='cell_num_id')
demographic_data
```

```
demographic_filtered = demographic_data[demographic_data['response_theme'] != 'Main Screen']
demographic_filtered
```

```
# Drop the unwanted columns
demographic_filtered.drop(['udate_y', 'sess_id_y'], axis=1, inplace=True)
# Rename the columns
demographic_filtered.rename(columns={'udate_x': 'udate', 'sess_id_x': 'sess_id'}, inplace=True)
```

BUILDING ML PIPELINE



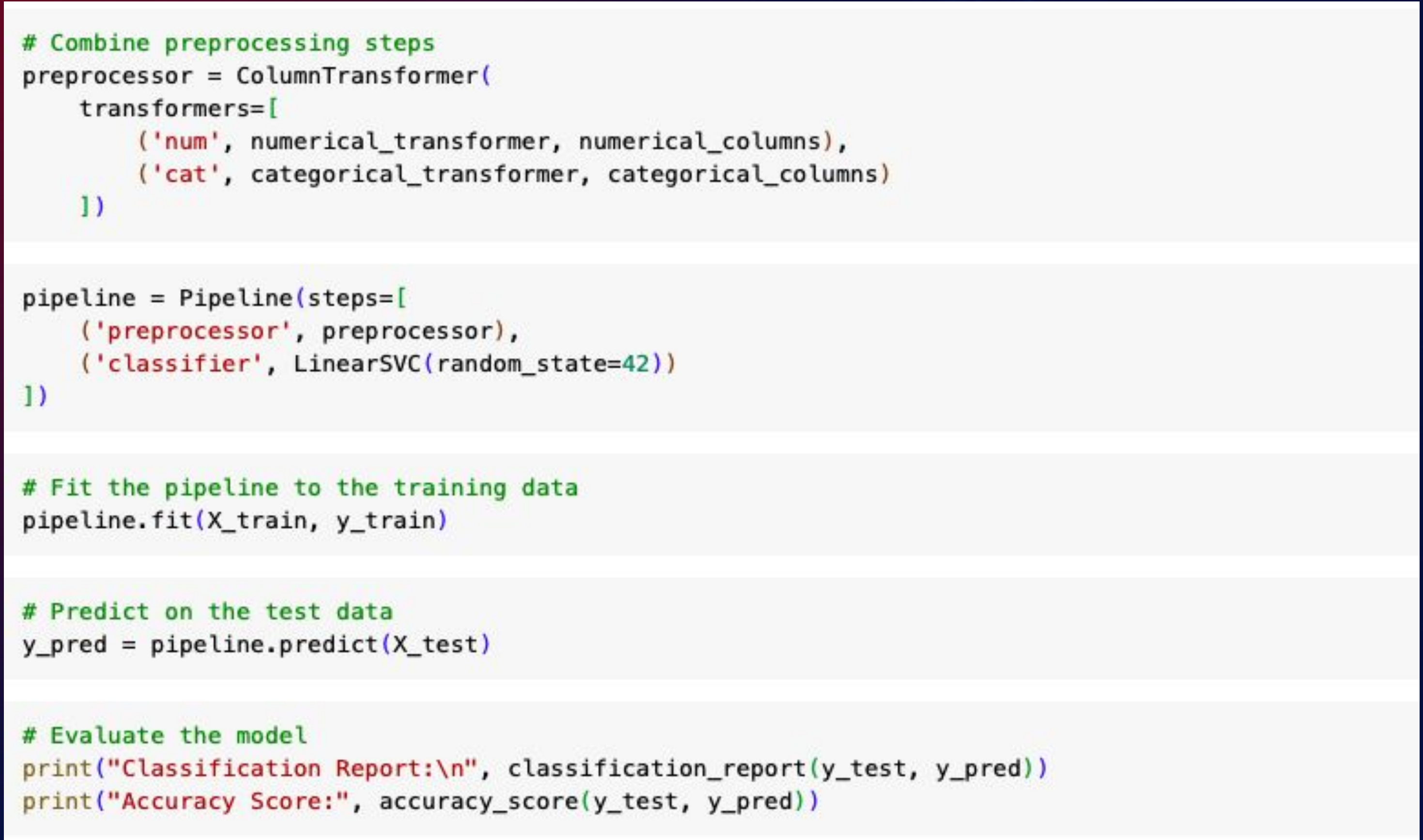
```
# Define your feature DataFrame and target variable
X = demographic_data.drop(['response_theme','reply_id', 'request_id', 'response_id', 'parent', 'created_date','notes'], axis=1)
y = demographic_data['response_theme']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Define categorical and numerical columns
categorical_columns = X.select_dtypes(include=['object']).columns
numerical_columns = X.select_dtypes(include=['int64', 'float64']).columns
```

```
# Create the preprocessing pipelines for both numerical and categorical data
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])
```

RUNNING MODEL



```
# Combine preprocessing steps
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_columns),
        ('cat', categorical_transformer, categorical_columns)
    ])
```

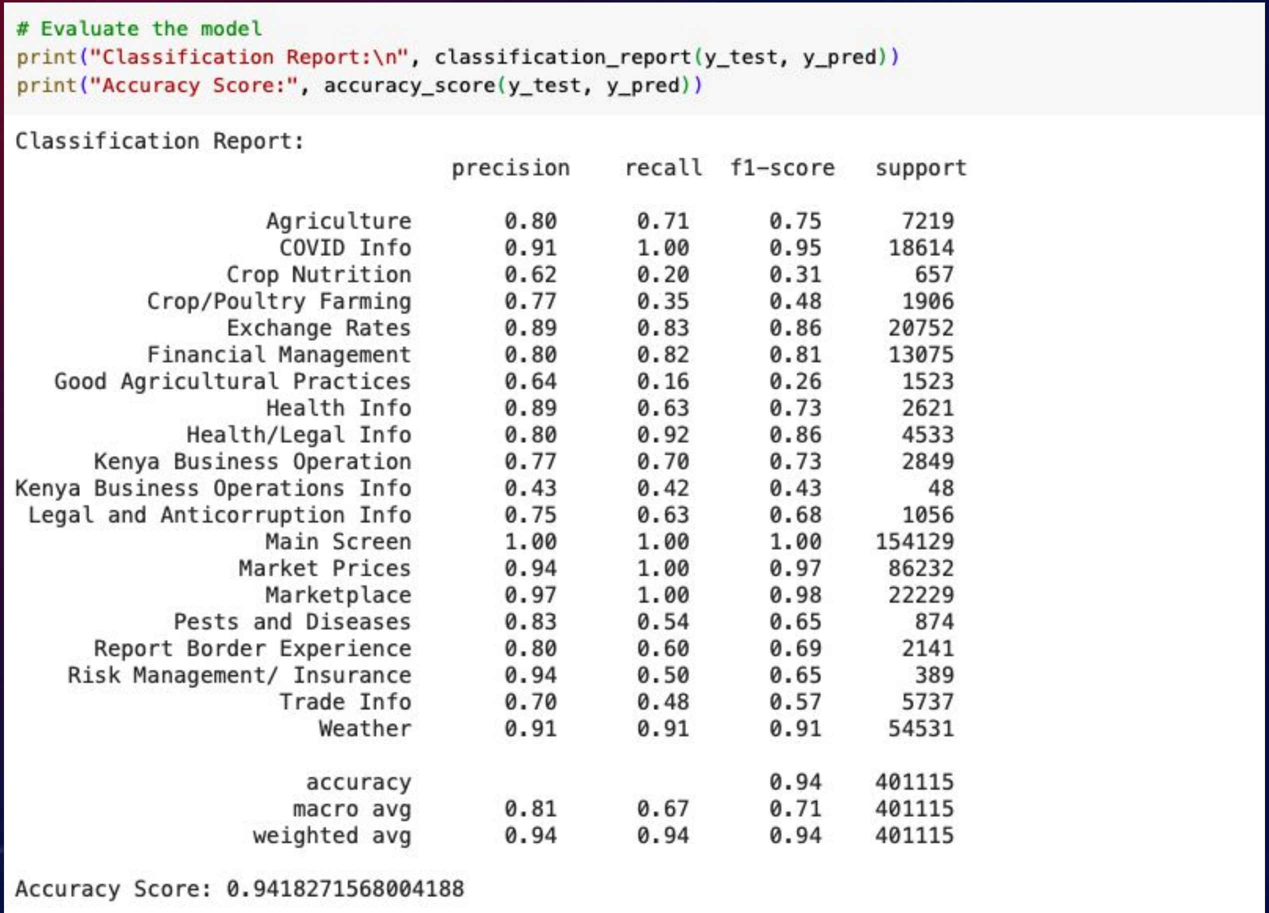
```
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LinearSVC(random_state=42))
])
```

```
# Fit the pipeline to the training data
pipeline.fit(X_train, y_train)
```

```
# Predict on the test data
y_pred = pipeline.predict(X_test)
```

```
# Evaluate the model
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))
```

RESULTS



```
# Evaluate the model
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))
```

```
Classification Report:
                                  precision    recall  f1-score   support

                     Agriculture       0.80      0.71      0.75      7219
                      COVID Info       0.91      1.00      0.95     18614
                  Crop Nutrition       0.62      0.20      0.31       657
           Crop/Poultry Farming       0.77      0.35      0.48      1906
                  Exchange Rates       0.89      0.83      0.86     20752
            Financial Management       0.80      0.82      0.81     13075
     Good Agricultural Practices       0.64      0.16      0.26      1523
                     Health Info       0.89      0.63      0.73      2621
               Health/Legal Info       0.80      0.92      0.86      4533
        Kenya Business Operation       0.77      0.70      0.73      2849
  Kenya Business Operations Info       0.43      0.42      0.43        48
   Legal and Anticorruption Info       0.75      0.63      0.68      1056
                     Main Screen       1.00      1.00      1.00    154129
                   Market Prices       0.94      1.00      0.97     86232
                     Marketplace       0.97      1.00      0.98     22229
              Pests and Diseases       0.83      0.54      0.65       874
        Report Border Experience       0.80      0.60      0.69      2141
      Risk Management/ Insurance       0.94      0.50      0.65       389
                      Trade Info       0.70      0.48      0.57      5737
                         Weather       0.91      0.91      0.91     54531

                        accuracy                           0.94    401115
                       macro avg       0.81      0.67      0.71    401115
                    weighted avg       0.94      0.94      0.94    401115

Accuracy Score: 0.9418271568004188
```

CLASSIFICATION RESULTS

- Accuracy: High at 94.18%, indicating strong overall model performance.
- COVID Info & Market Prices: Exceptional precision and recall, almost perfect classification.
- Agriculture: Good performance with a balance between precision (80%) and recall (71%).
- Crop Nutrition: Suboptimal, with low recall (20%), indicating many actual instances were missed.
- Trade Info: Moderate precision (70%) but lower recall (48%), suggesting challenges in identifying all relevant instances.
- Macro Average: Fair performance with equal weighting across categories (Precision: 0.81, Recall: 0.67, F1-Score: 0.71).
- Weighted Average: Excellent, considering class imbalance (Precision: 0.94, Recall: 0.94, F1-Score: 0.94).