import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset (Replace 'data.csv' with your actual file path)

# df = pd.read\_csv('data.csv')

# Example: Uncomment and load the dataset appropriately

# df = pd.read\_excel('data.xlsx')

# Check the structure of the dataset

print("Shape of the dataset:", df.shape)

print("\nColumns and data types:\n", df.dtypes)

# Preview the first few rows

print("\nFirst few rows:\n", df.head())

### 1. Missing Value Analysis ###

missing\_data = df.isnull().sum().sort\_values(ascending=False)

missing\_percentage = (missing\_data / len(df)) \* 100

missing\_summary = pd.DataFrame({"Missing Count": missing\_data, "Missing Percentage": missing\_percentage})

print("\nMissing Value Summary:\n", missing\_summary)

# Visualize missing data

plt.figure(figsize=(10, 6))

sns.heatmap(df.isnull(), cbar=False, cmap='viridis')

plt.title("Heatmap of Missing Values")

plt.show()

### 2. Descriptive Statistics ###

# Numerical Columns

numerical\_columns = df.select\_dtypes(include=['int64', 'float64']).columns

print("\nSummary Statistics for Numerical Columns:\n", df[numerical\_columns].describe())

# Categorical Columns

categorical\_columns = df.select\_dtypes(include=['object']).columns

for col in categorical\_columns:

print(f"\nColumn: {col}")

print(df[col].value\_counts().head(10)) # Show top 10 categories

### 3. Distribution Analysis ###

# Numerical Distributions

for col in numerical\_columns:

plt.figure(figsize=(6, 4))

sns.histplot(df[col].dropna(), kde=True, bins=30)

plt.title(f"Distribution of {col}")

plt.xlabel(col)

plt.ylabel("Frequency")

plt.show()

# Categorical Distributions

for col in categorical\_columns[:5]: # Limit to 5 categories for demonstration

plt.figure(figsize=(8, 4))

sns.countplot(y=col, data=df, order=df[col].value\_counts().index[:10])

plt.title(f"Top 10 Categories in {col}")

plt.xlabel("Count")

plt.ylabel(col)

plt.show()

### 4. Correlation Analysis ###

# Correlation Matrix for Numerical Features

correlation\_matrix = df[numerical\_columns].corr()

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, fmt=".2f", cmap="coolwarm", cbar=True)

plt.title("Correlation Matrix")

plt.show()

### 5. Temporal Analysis ###

# Assuming `snapshot\_date` is a datetime column

if 'snapshot\_date' in df.columns:

df['snapshot\_date'] = pd.to\_datetime(df['snapshot\_date'])

df['YearMonth'] = df['snapshot\_date'].dt.to\_period('M')

time\_series = df.groupby('YearMonth').size()

plt.figure(figsize=(10, 6))

time\_series.plot(kind='line', marker='o')

plt.title("Trend Over Time")

plt.xlabel("Time")

plt.ylabel("Frequency")

plt.show()

### 6. Segmentation Analysis ###

# Grouping by categorical fields

if 'brand' in df.columns and 'addon\_rev' in df.columns:

brand\_revenue = df.groupby('brand')['addon\_rev'].mean().sort\_values(ascending=False)

plt.figure(figsize=(8, 4))

sns.barplot(x=brand\_revenue.index, y=brand\_revenue.values)

plt.title("Average Revenue by Brand")

plt.xlabel("Brand")

plt.ylabel("Average Revenue")

plt.show()

# Example Group Analysis

if 'region' in df.columns:

print("\nCounts by Region:\n", df['region'].value\_counts())

### 7. Telecom-Specific Insights ###

# Churn Indicators (if columns exist)

if 'contract\_tenure' in df.columns:

plt.figure(figsize=(8, 4))

sns.histplot(df['contract\_tenure'], kde=False, bins=30)

plt.title("Distribution of Contract Tenure")

plt.xlabel("Contract Tenure")

plt.ylabel("Frequency")

plt.show()

if 'service\_product\_revenue\_net\_inc\_vat' in df.columns:

plt.figure(figsize=(8, 4))

sns.boxplot(x='region', y='service\_product\_revenue\_net\_inc\_vat', data=df)

plt.title("Revenue Distribution by Region")

plt.xlabel("Region")

plt.ylabel("Revenue")

plt.xticks(rotation=45)

plt.show()

### 8. Save Results ###

# Save missing summary

missing\_summary.to\_csv("missing\_summary.csv", index=False)

print("\nMissing Summary saved as missing\_summary.csv")

### Instructions:

1. Replace data.csv or data.xlsx with your actual dataset file.
2. Ensure required libraries (pandas, numpy, matplotlib, seaborn) are installed.
3. Review results and adjust column names or analysis sections to match your dataset specifics.

4o

##### You said:

Write the code for a detailed Telecom-Specific Metrics

###### ChatGPT said:

ChatGPT

Here’s the Python code for detailed **telecom-specific metrics** analysis based on the schema provided:

python

Copy code

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset

# df = pd.read\_csv('telecom\_data.csv') # Replace with your actual file

# Ensure datetime parsing if required

# df['snapshot\_date'] = pd.to\_datetime(df['snapshot\_date'])

### 1. Average Revenue Per User (ARPU) ###

if {'addon\_rev', 'service\_revenue'}.issubset(df.columns):

df['total\_revenue'] = df['addon\_rev'] + df['service\_revenue']

arpu = df['total\_revenue'].mean()

print(f"\nAverage Revenue Per User (ARPU): {arpu:.2f}")

# Plotting ARPU distribution

plt.figure(figsize=(8, 4))

sns.histplot(df['total\_revenue'], kde=True, bins=30, color="blue")

plt.title("Distribution of Total Revenue (ARPU)")

plt.xlabel("Revenue")

plt.ylabel("Frequency")

plt.show()

### 2. Churn Analysis ###

if 'ee\_account\_ported\_out\_cnt' in df.columns:

churn\_rate = (df['ee\_account\_ported\_out\_cnt'] > 0).mean() \* 100

print(f"\nChurn Rate: {churn\_rate:.2f}%")

# Visualizing churn across regions if 'region' exists

if 'region' in df.columns:

churn\_by\_region = df.groupby('region')['ee\_account\_ported\_out\_cnt'].sum().sort\_values()

plt.figure(figsize=(10, 5))

churn\_by\_region.plot(kind='bar', color='orange')

plt.title("Churn by Region")

plt.xlabel("Region")

plt.ylabel("Churn Count")

plt.xticks(rotation=45)

plt.show()

### 3. Tenure Analysis ###

if 'contract\_tenure' in df.columns:

plt.figure(figsize=(8, 4))

sns.histplot(df['contract\_tenure'], kde=False, bins=30, color="green")

plt.title("Distribution of Contract Tenure")

plt.xlabel("Contract Tenure (Months)")

plt.ylabel("Frequency")

plt.show()

tenure\_mean = df['contract\_tenure'].mean()

print(f"\nAverage Contract Tenure: {tenure\_mean:.2f} months")

### 4. Data Usage Analysis ###

if 'data\_allowance\_gb' in df.columns:

plt.figure(figsize=(8, 4))

sns.histplot(df['data\_allowance\_gb'], kde=True, bins=30, color="purple")

plt.title("Distribution of Data Allowance (GB)")

plt.xlabel("Data Allowance (GB)")

plt.ylabel("Frequency")

plt.show()

data\_usage\_mean = df['data\_allowance\_gb'].mean()

print(f"\nAverage Data Allowance: {data\_usage\_mean:.2f} GB")

### 5. Revenue by Service Type ###

if {'service', 'service\_revenue'}.issubset(df.columns):

revenue\_by\_service = df.groupby('service')['service\_revenue'].sum().sort\_values(ascending=False)

print("\nRevenue by Service Type:\n", revenue\_by\_service)

plt.figure(figsize=(10, 5))

sns.barplot(x=revenue\_by\_service.index, y=revenue\_by\_service.values, palette="Blues\_d")

plt.title("Revenue by Service Type")

plt.xlabel("Service Type")

plt.ylabel("Total Revenue")

plt.xticks(rotation=45)

plt.show()

### 6. Monthly Recurring Revenue (MRR) ###

if 'curr\_monthly\_recurring\_margin' in df.columns:

df['curr\_monthly\_recurring\_margin'] = pd.to\_numeric(df['curr\_monthly\_recurring\_margin'], errors='coerce')

mrr = df['curr\_monthly\_recurring\_margin'].sum()

print(f"\nTotal Monthly Recurring Revenue (MRR): {mrr:.2f}")

# Visualizing MRR by Region

if 'region' in df.columns:

mrr\_by\_region = df.groupby('region')['curr\_monthly\_recurring\_margin'].sum().sort\_values()

plt.figure(figsize=(10, 5))

sns.barplot(x=mrr\_by\_region.index, y=mrr\_by\_region.values, palette="viridis")

plt.title("Monthly Recurring Revenue (MRR) by Region")

plt.xlabel("Region")

plt.ylabel("MRR")

plt.xticks(rotation=45)

plt.show()

### 7. Product/Service Adoption ###

if 'service\_product\_group' in df.columns:

product\_adoption = df['service\_product\_group'].value\_counts()

print("\nProduct/Service Adoption:\n", product\_adoption)

plt.figure(figsize=(8, 4))

sns.barplot(x=product\_adoption.index, y=product\_adoption.values, palette="magma")

plt.title("Product/Service Adoption")

plt.xlabel("Product Group")

plt.ylabel("Adoption Count")

plt.xticks(rotation=45)

plt.show()

### 8. Flag-Based Insights ###

# Analyzing 'add\_line\_flag' or other binary flags

if 'add\_line\_flag' in df.columns:

add\_line\_distribution = df['add\_line\_flag'].value\_counts(normalize=True) \* 100

print("\nAdd Line Flag Distribution (%):\n", add\_line\_distribution)

# Plotting distribution

add\_line\_distribution.plot(kind='pie', autopct='%1.1f%%', labels=['No', 'Yes'], colors=["red", "green"])

plt.title("Add Line Flag Distribution")

plt.ylabel("")

plt.show()

### 9. Revenue Trends by Snapshot Date ###

if {'snapshot\_date', 'service\_revenue'}.issubset(df.columns):

df['snapshot\_date'] = pd.to\_datetime(df['snapshot\_date'])

revenue\_trend = df.groupby(df['snapshot\_date'].dt.to\_period('M'))['service\_revenue'].sum()

plt.figure(figsize=(10, 5))

revenue\_trend.plot(kind='line', marker='o', color="blue")

plt.title("Revenue Trend Over Time")

plt.xlabel("Time")

plt.ylabel("Service Revenue")

plt.grid(True)

plt.show()

### 10. Cross-Analysis of Key Metrics ###

if {'contract\_tenure', 'service\_revenue'}.issubset(df.columns):

plt.figure(figsize=(8, 6))

sns.scatterplot(x='contract\_tenure', y='service\_revenue', data=df, alpha=0.6)

plt.title("Contract Tenure vs Service Revenue")

plt.xlabel("Contract Tenure (Months)")

plt.ylabel("Service Revenue")

plt.show()

# Save outputs if needed

df[['total\_revenue', 'contract\_tenure', 'curr\_monthly\_recurring\_margin']].to\_csv("telecom\_metrics.csv", index=False)

print("\nKey telecom metrics saved to telecom\_metrics.csv")

### Explanation of Key Metrics:

1. **ARPU (Average Revenue Per User)**:
   * Calculates total revenue (addon\_rev + service\_revenue) and finds the average.
   * Visualizes ARPU distribution.
2. **Churn Analysis**:
   * Identifies customers who ported out (ee\_account\_ported\_out\_cnt).
   * Analyzes churn trends by region or other categories.
3. **Tenure Analysis**:
   * Evaluates the distribution of customer contract tenures to identify patterns.
4. **Data Usage Analysis**:
   * Analyzes data\_allowance\_gb to understand customer data consumption.
5. **Revenue Insights by Service**:
   * Groups revenue by service type for product profitability insights.
6. **Monthly Recurring Revenue (MRR)**:
   * Calculates recurring margin to track steady revenue inflow.
7. **Product/Service Adoption**:
   * Analyzes the popularity of services/products (service\_product\_group).
8. **Flag-Based Insights**:
   * Examines binary flags like add\_line\_flag for feature adoption trends.
9. **Time-Based Revenue Trends**:
   * Tracks service\_revenue across months to identify growth or decline trends.
10. **Cross-Analysis**:

* Correlates contract tenure and revenue to discover patterns or anomalies.

This code is designed to adapt easily to telecom datasets. You can replace column names to suit your schema or expand further based on your business needs!

4o

##### You said:

Show metrics like revenue per model and things like postcode wise model,age group wise model distribution,no of connection wise model distribution,service aquatision wise model distribution

###### ChatGPT said:

ChatGPT

Here's Python code to calculate and visualize metrics like revenue per model, model distribution by postcode, age group, number of connections, and service acquisition:

python

Copy code

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load dataset

# df = pd.read\_csv('telecom\_data.csv') # Replace with your dataset

### 1. Revenue Per Model ###

if {'model\_name', 'service\_revenue'}.issubset(df.columns):

revenue\_per\_model = df.groupby('model\_name')['service\_revenue'].sum().sort\_values(ascending=False)

print("\nRevenue Per Model:\n", revenue\_per\_model)

# Visualize

plt.figure(figsize=(12, 6))

revenue\_per\_model.head(10).plot(kind='bar', color='blue')

plt.title("Top 10 Models by Revenue")

plt.xlabel("Model Name")

plt.ylabel("Total Revenue")

plt.xticks(rotation=45)

plt.show()

### 2. Postcode-wise Model Distribution ###

if {'postcode', 'model\_name'}.issubset(df.columns):

model\_distribution\_by\_postcode = df.groupby(['postcode', 'model\_name']).size().reset\_index(name='count')

print("\nPostcode-wise Model Distribution:\n", model\_distribution\_by\_postcode)

# Visualize (Top 10 Postcodes with Highest Model Counts)

top\_postcodes = model\_distribution\_by\_postcode.groupby('postcode')['count'].sum().sort\_values(ascending=False).head(10)

plt.figure(figsize=(10, 6))

top\_postcodes.plot(kind='bar', color='purple')

plt.title("Top 10 Postcodes by Model Distribution")

plt.xlabel("Postcode")

plt.ylabel("Model Count")

plt.xticks(rotation=45)

plt.show()

### 3. Age Group-wise Model Distribution ###

# Assuming there's a column 'age' and age groups can be created

if 'age' in df.columns:

bins = [0, 18, 30, 45, 60, 100]

labels = ['0-18', '19-30', '31-45', '46-60', '60+']

df['age\_group'] = pd.cut(df['age'], bins=bins, labels=labels, right=False)

age\_model\_distribution = df.groupby(['age\_group', 'model\_name']).size().reset\_index(name='count')

print("\nAge Group-wise Model Distribution:\n", age\_model\_distribution)

# Visualize Age Group Distribution

plt.figure(figsize=(12, 6))

sns.countplot(data=df, x='age\_group', order=labels, palette="coolwarm")

plt.title("Model Distribution Across Age Groups")

plt.xlabel("Age Group")

plt.ylabel("Count")

plt.show()

### 4. Number of Connections-wise Model Distribution ###

if 'account\_num' in df.columns:

connections\_per\_model = df.groupby(['model\_name', 'account\_num']).size().reset\_index(name='connection\_count')

connections\_distribution = connections\_per\_model.groupby('model\_name')['connection\_count'].sum().sort\_values(ascending=False)

print("\nConnections Per Model:\n", connections\_distribution)

# Visualize Top 10 Models with Highest Connections

plt.figure(figsize=(12, 6))

connections\_distribution.head(10).plot(kind='bar', color='green')

plt.title("Top 10 Models by Number of Connections")

plt.xlabel("Model Name")

plt.ylabel("Number of Connections")

plt.xticks(rotation=45)

plt.show()

### 5. Service Acquisition-wise Model Distribution ###

if {'service\_acquisition\_date', 'model\_name'}.issubset(df.columns):

df['service\_acquisition\_date'] = pd.to\_datetime(df['service\_acquisition\_date'])

df['acquisition\_year'] = df['service\_acquisition\_date'].dt.year

acquisition\_model\_distribution = df.groupby(['acquisition\_year', 'model\_name']).size().reset\_index(name='count')

print("\nService Acquisition-wise Model Distribution:\n", acquisition\_model\_distribution)

# Visualize Model Distribution by Year

plt.figure(figsize=(12, 6))

sns.countplot(data=df, x='acquisition\_year', order=df['acquisition\_year'].sort\_values().unique(), palette="viridis")

plt.title("Model Distribution by Service Acquisition Year")

plt.xlabel("Acquisition Year")

plt.ylabel("Count")

plt.show()

### Metrics Explained:

1. **Revenue Per Model**:
   * Calculates total service revenue generated by each model (model\_name).
   * Bar plot for top 10 models by revenue.
2. **Postcode-wise Model Distribution**:
   * Groups models by postcode and counts their occurrences.
   * Visualizes top 10 postcodes by total model counts.
3. **Age Group-wise Model Distribution**:
   * Categorizes customers into age groups and analyzes model distribution.
   * Visualizes age group distribution.
4. **Number of Connections-wise Model Distribution**:
   * Counts the number of connections (account\_num) per model.
   * Bar plot for models with the highest connections.
5. **Service Acquisition-wise Model Distribution**:
   * Tracks how model adoption varies by service acquisition year.
   * Visualizes the number of models acquired each year.

### Customizations:

* Update column names like postcode, age, and service\_acquisition\_date as per your schema.
* If you have additional filters (e.g., region, product group), add them to these aggregations.

Let me know if you need further customization or explanations!

4o

##### You said:

Household member wise model distribution

###### ChatGPT said:

ChatGPT

To analyze **household member-wise model distribution**, we can group the data by the number of household members and model names, then visualize the results. Here's the Python code for this analysis:

python

Copy code

# Check if the necessary columns exist

if {'hh\_adult\_cnt', 'model\_name'}.issubset(df.columns):

### 1. Household Member-wise Model Distribution ###

# Group data by household member count and model

household\_model\_distribution = df.groupby(['hh\_adult\_cnt', 'model\_name']).size().reset\_index(name='count')

print("\nHousehold Member-wise Model Distribution:\n", household\_model\_distribution)

# Aggregate total models by household member count

household\_total\_distribution = df.groupby('hh\_adult\_cnt')['model\_name'].count().sort\_values(ascending=False)

print("\nTotal Models by Household Members:\n", household\_total\_distribution)

# Visualize: Top Household Member Counts by Model Distribution

plt.figure(figsize=(10, 6))

sns.barplot(x=household\_total\_distribution.index, y=household\_total\_distribution.values, palette="coolwarm")

plt.title("Household Member-wise Model Distribution")

plt.xlabel("Number of Household Members")

plt.ylabel("Total Models Count")

plt.show()

### 2. Most Popular Models per Household Size ###

top\_models\_per\_household = household\_model\_distribution.groupby('hh\_adult\_cnt').apply(

lambda x: x.sort\_values('count', ascending=False).head(1)

).reset\_index(drop=True)

print("\nMost Popular Models Per Household Size:\n", top\_models\_per\_household)

# Visualize Most Popular Models Per Household Size

plt.figure(figsize=(12, 6))

sns.barplot(

data=top\_models\_per\_household,

x='hh\_adult\_cnt',

y='count',

hue='model\_name',

palette='Set2'

)

plt.title("Most Popular Model Per Household Size")

plt.xlabel("Number of Household Members")

plt.ylabel("Model Count")

plt.legend(title="Model Name", bbox\_to\_anchor=(1.05, 1), loc='upper left')

plt.show()

else:

print("The columns 'hh\_adult\_cnt' and/or 'model\_name' are missing from the dataset.")

### Explanation:

1. **Household Member-wise Model Distribution**:
   * Groups data by hh\_adult\_cnt (number of adults in the household) and model\_name to calculate the count of models per household size.
   * Aggregates total models per household size to identify how many models are associated with each household size.
   * Bar plot shows total model counts for various household sizes.
2. **Most Popular Models per Household Size**:
   * Identifies the most popular model for each household size by selecting the model with the highest count within each group.
   * Bar plot visualizes the most popular models for each household size, with a legend to distinguish models.

### Assumptions:

* hh\_adult\_cnt: Represents the number of adults in the household (or replace with hh\_member\_cnt if the dataset tracks total household members).
* model\_name: Represents the name of the device or model associated with the household.

### Output:

1. Tabular data showing the count of models for each household size and model.
2. Visualization of model distributions across different household sizes.
3. Visualization of the most popular model for each household size.

Let me know if you have specific variations or need further analysis!

4o

##### You said:

Perform analysis such is there any bias in model distribution and many more data science casual analysis and inference

###### ChatGPT said:

ChatGPT

Here’s an extended analysis to explore potential **biases** in model distribution, along with casual data science analyses and inferences that can reveal underlying patterns or insights from the telecom dataset:

### ****Bias Analysis in Model Distribution****

1. **Check Model Distribution by Key Categories (e.g., Gender, Age Group, Region)**: Biases can arise if a model is disproportionately adopted by specific demographic groups or geographic regions.

python

Copy code

# Model Distribution by Gender

if {'gender', 'model\_name'}.issubset(df.columns):

gender\_model\_distribution = df.groupby(['gender', 'model\_name']).size().reset\_index(name='count')

print("\nModel Distribution by Gender:\n", gender\_model\_distribution)

# Visualization

plt.figure(figsize=(10, 6))

sns.countplot(data=df, x='model\_name', hue='gender', order=df['model\_name'].value\_counts().index[:10])

plt.title("Top 10 Models by Gender")

plt.xlabel("Model Name")

plt.ylabel("Count")

plt.legend(title="Gender")

plt.xticks(rotation=45)

plt.show()

# Model Distribution by Region

if {'region', 'model\_name'}.issubset(df.columns):

region\_model\_distribution = df.groupby(['region', 'model\_name']).size().reset\_index(name='count')

print("\nModel Distribution by Region:\n", region\_model\_distribution)

# Visualization

plt.figure(figsize=(12, 6))

sns.countplot(data=df, x='model\_name', hue='region', order=df['model\_name'].value\_counts().index[:10])

plt.title("Top 10 Models by Region")

plt.xlabel("Model Name")

plt.ylabel("Count")

plt.legend(title="Region")

plt.xticks(rotation=45)

plt.show()

# Model Distribution by Age Group

if {'age\_group', 'model\_name'}.issubset(df.columns):

age\_model\_distribution = df.groupby(['age\_group', 'model\_name']).size().reset\_index(name='count')

print("\nModel Distribution by Age Group:\n", age\_model\_distribution)

plt.figure(figsize=(10, 6))

sns.countplot(data=df, x='age\_group', hue='model\_name', order=df['age\_group'].unique())

plt.title("Model Distribution by Age Group")

plt.xlabel("Age Group")

plt.ylabel("Count")

plt.xticks(rotation=45)

plt.show()

**Inferences**:

* Disproportionate adoption of certain models by a specific gender, region, or age group may indicate marketing or accessibility bias.

### ****Market Segmentation Analysis****

1. **Segment Users by Monthly Revenue and Model**: Identify if certain user groups generate more revenue for specific models.

python

Copy code

if {'model\_name', 'service\_revenue'}.issubset(df.columns):

revenue\_segments = pd.cut(df['service\_revenue'], bins=[0, 50, 100, 200, 500, np.inf],

labels=['Low', 'Mid-Low', 'Mid', 'High', 'Very High'])

df['revenue\_segment'] = revenue\_segments

revenue\_model\_distribution = df.groupby(['revenue\_segment', 'model\_name']).size().reset\_index(name='count')

print("\nRevenue Segment-wise Model Distribution:\n", revenue\_model\_distribution)

plt.figure(figsize=(12, 6))

sns.barplot(data=revenue\_model\_distribution, x='revenue\_segment', y='count', hue='model\_name')

plt.title("Revenue Segments vs Model Distribution")

plt.xlabel("Revenue Segment")

plt.ylabel("Count")

plt.legend(title="Model Name", bbox\_to\_anchor=(1.05, 1), loc='upper left')

plt.show()

**Inferences**:

* Certain models may dominate high-revenue segments, which could indicate premium positioning or customer preferences for high-value users.

### ****Temporal Trends in Model Adoption****

1. **Analyze Model Trends Over Time**: Understand if some models were adopted more in certain time periods (e.g., launch bias or seasonality).

python

Copy code

if {'model\_name', 'snapshot\_date'}.issubset(df.columns):

df['snapshot\_date'] = pd.to\_datetime(df['snapshot\_date'])

df['year\_month'] = df['snapshot\_date'].dt.to\_period('M')

model\_trends = df.groupby(['year\_month', 'model\_name']).size().reset\_index(name='count')

print("\nModel Trends Over Time:\n", model\_trends)

# Visualize trends for the top 5 models

top\_models = df['model\_name'].value\_counts().head(5).index

plt.figure(figsize=(14, 8))

for model in top\_models:

trend = model\_trends[model\_trends['model\_name'] == model]

plt.plot(trend['year\_month'].astype(str), trend['count'], marker='o', label=model)

plt.title("Top 5 Models - Adoption Trends Over Time")

plt.xlabel("Time (Year-Month)")

plt.ylabel("Count")

plt.legend(title="Model Name")

plt.xticks(rotation=45)

plt.grid(True)

plt.show()

**Inferences**:

* Models with spikes in specific months may indicate promotional events or holiday sales.

### ****Casual Analysis****

1. **Customer Behavior by Model and Tenure**: Check if models are linked to longer contract tenures or frequent recontracting.

python

Copy code

if {'contract\_tenure', 'model\_name'}.issubset(df.columns):

tenure\_by\_model = df.groupby('model\_name')['contract\_tenure'].mean().sort\_values(ascending=False)

print("\nAverage Contract Tenure by Model:\n", tenure\_by\_model)

# Visualize

plt.figure(figsize=(12, 6))

tenure\_by\_model.head(10).plot(kind='bar', color='teal')

plt.title("Average Contract Tenure by Model (Top 10)")

plt.xlabel("Model Name")

plt.ylabel("Average Tenure (Months)")

plt.xticks(rotation=45)

plt.show()

**Inferences**:

* Longer tenure models may indicate customer satisfaction or suitability for long-term contracts.

### ****Predictive Indicators****

1. **Model Adoption Prediction Based on Demographics**: Use logistic regression or decision trees to predict model adoption using features like age\_group, region, gender, etc.

python

Copy code

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report

if {'age\_group', 'region', 'gender', 'model\_name'}.issubset(df.columns):

# Encode categorical features

df\_encoded = pd.get\_dummies(df, columns=['age\_group', 'region', 'gender'], drop\_first=True)

# Simplify to a binary classification for one popular model (e.g., "Model A")

df\_encoded['model\_target'] = (df['model\_name'] == 'Model A').astype(int)

X = df\_encoded.drop(['model\_name', 'model\_target'], axis=1)

y = df\_encoded['model\_target']

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Train a Random Forest Classifier

model = RandomForestClassifier(random\_state=42)

model.fit(X\_train, y\_train)

# Evaluate

y\_pred = model.predict(X\_test)

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

**Inferences**:

* Model adoption can be predicted using customer demographics and other features, which can inform marketing strategies.

### Additional Analyses:

1. **Revenue Clustering**:
   * Use KMeans to segment users into revenue clusters based on their service\_revenue and other revenue-related metrics.
2. **Correlation with Churn**:
   * Explore correlations between churn (ee\_account\_ported\_out\_cnt) and other features like contract\_tenure or data\_allowance\_gb.

Let me know if you’d like to expand any part or dive deeper into specific insights!

4o

##### You said:

perform the aditional analysis

###### ChatGPT said:

ChatGPT

It seems the dataset (df) is not currently loaded or defined. To perform this analysis, I would need the actual dataset in a machine-readable format (CSV, Excel, or similar). If you have the dataset, please upload it, and I can rerun the analysis for you.

If the dataset is unavailable, you can replace df with your dataset and ensure columns like service\_revenue, addon\_rev, ee\_account\_ported\_out\_cnt, contract\_tenure, and data\_allowance\_gb exist for the code to work correctly. Let me know how you'd like to proceed! ​​

4o

##### You said:

Write the code for this: Additional Analyses: Revenue Clustering: Use KMeans to segment users into revenue clusters based on their service\_revenue and other revenue-related metrics. Correlation with Churn: Explore correlations between churn (ee

###### ChatGPT said:

ChatGPT

Here’s the Python code to perform the **Revenue Clustering** using KMeans and **Correlation with Churn** analysis:

python

Copy code

import pandas as pd

import numpy as np

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

import seaborn as sns

# Placeholder: Load your dataset

# df = pd.read\_csv('telecom\_data.csv') # Replace with your dataset

### 1. Revenue Clustering ###

if {'service\_revenue', 'addon\_rev'}.issubset(df.columns):

# Select revenue-related features

revenue\_data = df[['service\_revenue', 'addon\_rev']].dropna()

# Standardize the data

scaler = StandardScaler()

revenue\_scaled = scaler.fit\_transform(revenue\_data)

# Apply KMeans clustering

kmeans = KMeans(n\_clusters=4, random\_state=42)

revenue\_data['Cluster'] = kmeans.fit\_predict(revenue\_scaled)

# Add clusters back to the original dataframe

df.loc[revenue\_data.index, 'Revenue\_Cluster'] = revenue\_data['Cluster']

# Display cluster centroids

cluster\_centroids = pd.DataFrame(scaler.inverse\_transform(kmeans.cluster\_centers\_),

columns=['service\_revenue', 'addon\_rev'])

print("\nCluster Centroids:\n", cluster\_centroids)

# Visualize the clusters

plt.figure(figsize=(10, 6))

sns.scatterplot(data=revenue\_data, x='service\_revenue', y='addon\_rev', hue='Cluster', palette='Set2', s=50)

plt.title("Revenue Clustering")

plt.xlabel("Service Revenue")

plt.ylabel("Addon Revenue")

plt.legend(title="Cluster")

plt.show()

### 2. Correlation with Churn ###

if {'ee\_account\_ported\_out\_cnt', 'contract\_tenure', 'data\_allowance\_gb'}.issubset(df.columns):

# Check the correlation of churn with key features

correlation\_data = df[['ee\_account\_ported\_out\_cnt', 'contract\_tenure', 'data\_allowance\_gb']]

churn\_corr = correlation\_data.corr()

print("\nCorrelation with Churn:\n", churn\_corr)

# Visualize correlation heatmap

plt.figure(figsize=(8, 6))

sns.heatmap(churn\_corr, annot=True, fmt=".2f", cmap='coolwarm')

plt.title("Correlation with Churn")

plt.show()

### Additional: Add Revenue Segment Analysis ###

if {'service\_revenue', 'addon\_rev'}.issubset(df.columns):

# Create revenue segments

df['Total\_Revenue'] = df['service\_revenue'] + df['addon\_rev']

revenue\_bins = [0, 50, 100, 200, 500, np.inf]

revenue\_labels = ['Very Low', 'Low', 'Medium', 'High', 'Very High']

df['Revenue\_Segment'] = pd.cut(df['Total\_Revenue'], bins=revenue\_bins, labels=revenue\_labels, right=False)

# Analyze revenue segment distribution

revenue\_segment\_distribution = df['Revenue\_Segment'].value\_counts()

print("\nRevenue Segment Distribution:\n", revenue\_segment\_distribution)

# Visualize revenue segments

plt.figure(figsize=(10, 6))

revenue\_segment\_distribution.sort\_index().plot(kind='bar', color='orange')

plt.title("Revenue Segment Distribution")

plt.xlabel("Revenue Segment")

plt.ylabel("Number of Users")

plt.xticks(rotation=45)

plt.show()

### Explanation:

#### ****1. Revenue Clustering****:

* **Purpose**: Segment users into clusters based on their service\_revenue and addon\_rev.
* **Steps**:
  + Standardize the data using StandardScaler.
  + Use KMeans clustering to group users into 4 revenue clusters.
  + Visualize the clusters in a scatter plot with service\_revenue and addon\_rev.

#### ****2. Correlation with Churn****:

* **Purpose**: Explore correlations between churn (e.g., ee\_account\_ported\_out\_cnt) and other key features like contract\_tenure and data\_allowance\_gb.
* **Steps**:
  + Calculate the correlation matrix for the selected features.
  + Visualize the correlations using a heatmap.

#### ****3. Revenue Segmentation****:

* **Purpose**: Categorize users into revenue tiers based on total revenue (service\_revenue + addon\_rev).
* **Steps**:
  + Create bins (e.g., Very Low, Low, etc.) to segment revenue data.
  + Visualize the distribution of users across revenue tiers.

### Output:

* **Cluster Centroids**: Displays the average values of service\_revenue and addon\_rev for each cluster.
* **Scatter Plot**: Shows the revenue clusters visually.
* **Correlation Matrix**: Displays the strength of relationships between churn and other features.
* **Bar Chart**: Shows the distribution of users across revenue segments.

Replace placeholders with your dataset to perform this analysis. Let me know if you need any modifications or deeper analysis!

### 1. ****Demographic Analysis****

python

Copy code

# Demographics: Socio-Economic Class vs Income Band

if {'hh\_socio\_eco\_class\_details', 'hh\_income\_band\_details'}.issubset(df.columns):

demographic\_dist = df.groupby(['hh\_socio\_eco\_class\_details', 'hh\_income\_band\_details']).size().reset\_index(name='count')

print("\nDemographic Distribution:\n", demographic\_dist)

# Heatmap for Socio-Economic Class vs Income Band

demographic\_pivot = demographic\_dist.pivot(index='hh\_socio\_eco\_class\_details', columns='hh\_income\_band\_details', values='count')

plt.figure(figsize=(10, 6))

sns.heatmap(demographic\_pivot, annot=True, fmt="d", cmap="Blues")

plt.title("Socio-Economic Class vs Income Band Distribution")

plt.xlabel("Income Band")

plt.ylabel("Socio-Economic Class")

plt.show()

### 2. ****Model Popularity and Trends****

python

Copy code

# Model Popularity by Manufacturer and Channel Group

if {'model\_name', 'device\_manufacturer', 'channel\_group'}.issubset(df.columns):

model\_popularity = df.groupby(['device\_manufacturer', 'model\_name', 'channel\_group']).size().reset\_index(name='count')

print("\nModel Popularity by Manufacturer and Channel Group:\n", model\_popularity)

# Top 10 Models by Popularity

top\_models = df['model\_name'].value\_counts().head(10)

plt.figure(figsize=(12, 6))

sns.barplot(x=top\_models.index, y=top\_models.values, palette="viridis")

plt.title("Top 10 Models by Popularity")

plt.xlabel("Model Name")

plt.ylabel("Count")

plt.xticks(rotation=45)

plt.show()

### 3. ****Correlation and Churn Risk Analysis****

python

Copy code

# Correlation Matrix for Risk Factors

if {'section\_card\_limit', 'service\_maturity', 'month\_in\_contract\_remaining', 'hh\_socio\_eco\_class\_details'}.issubset(df.columns):

risk\_factors = df[['section\_card\_limit', 'service\_maturity', 'month\_in\_contract\_remaining']]

corr\_matrix = risk\_factors.corr()

print("\nCorrelation Matrix:\n", corr\_matrix)

# Heatmap for Correlations

plt.figure(figsize=(8, 6))

sns.heatmap(corr\_matrix, annot=True, fmt=".2f", cmap="coolwarm")

plt.title("Correlation Heatmap for Risk Factors")

plt.show()

### 4. ****Service Utilization Analysis****

python

Copy code

# Service Maturity vs Smart Type

if {'service\_maturity', 'mob\_smart\_type'}.issubset(df.columns):

service\_utilization = df.groupby(['mob\_smart\_type', 'service\_maturity']).size().reset\_index(name='count')

print("\nService Maturity by Smart Type:\n", service\_utilization)

# Visualize Service Utilization

plt.figure(figsize=(12, 6))

sns.barplot(data=service\_utilization, x='mob\_smart\_type', y='count', hue='service\_maturity', palette="Set2")

plt.title("Service Utilization by Smart Type")

plt.xlabel("Mobile Smart Type")

plt.ylabel("Count")

plt.legend(title="Service Maturity")

plt.xticks(rotation=45)

plt.show()

### Business Insights and Recommendations:

1. **Demographics**:
   * A skewed distribution in socio-economic classes or income bands might suggest focusing on underserved groups to expand reach.
2. **Model Popularity**:
   * Certain models dominate specific manufacturers or channels, indicating potential marketing successes or dependencies.
3. **Churn Risk**:
   * High correlations between section\_card\_limit and month\_in\_contract\_remaining could indicate financial or service constraints driving churn.
4. **Service Utilization**:
   * Variations in mob\_smart\_type and service\_maturity could point to underperforming services for specific device types, signaling areas for improvement.

#### ****1. Advanced Statistical Analysis****

from scipy.stats import chi2\_contingency, ttest\_ind

# Hypothesis Testing: Is there a significant difference in income bands across socio-economic classes?

if {'hh\_socio\_eco\_class\_details', 'hh\_income\_band\_details'}.issubset(df.columns):

contingency\_table = pd.crosstab(df['hh\_socio\_eco\_class\_details'], df['hh\_income\_band\_details'])

chi2, p, dof, ex = chi2\_contingency(contingency\_table)

print(f"\nChi-Square Test Results for Socio-Economic Class and Income Band:\nChi2 = {chi2}, p-value = {p}")

if p < 0.05:

print("Result: Significant association between socio-economic class and income band.")

else:

print("Result: No significant association.")

# Hypothesis Testing: Do card limits differ by service maturity levels?

if {'section\_card\_limit', 'service\_maturity'}.issubset(df.columns):

mature = df[df['service\_maturity'] > df['service\_maturity'].median()]['section\_card\_limit']

non\_mature = df[df['service\_maturity'] <= df['service\_maturity'].median()]['section\_card\_limit']

t\_stat, p\_value = ttest\_ind(mature, non\_mature, nan\_policy='omit')

print(f"\nT-Test Results for Card Limits by Service Maturity:\nT-Statistic = {t\_stat}, p-value = {p\_value}")

if p\_value < 0.05:

print("Result: Significant difference in card limits by service maturity levels.")

else:

print("Result: No significant difference.")

2. Segmentation and Clustering

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

# Segmentation using KMeans

if {'service\_revenue', 'addon\_rev', 'hh\_income\_band\_details'}.issubset(df.columns):

clustering\_data = df[['service\_revenue', 'addon\_rev']].dropna()

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(clustering\_data)

# Apply PCA for visualization

pca = PCA(n\_components=2)

pca\_data = pca.fit\_transform(scaled\_data)

# KMeans Clustering

kmeans = KMeans(n\_clusters=4, random\_state=42)

clusters = kmeans.fit\_predict(scaled\_data)

clustering\_data['Cluster'] = clusters

# Visualize Clusters

plt.figure(figsize=(10, 6))

sns.scatterplot(x=pca\_data[:, 0], y=pca\_data[:, 1], hue=clusters, palette='viridis', s=50)

plt.title("Customer Segmentation: Revenue Clustering")

plt.xlabel("PCA Component 1")

plt.ylabel("PCA Component 2")

plt.legend(title="Cluster")

plt.show()

WITH

-- Step 1: Mobile Uncovered Base (from the first query)

mobile\_uncovered\_base AS (

SELECT \*

FROM `bt-bvp-dataprod-dp-prod.cons\_bvp\_dp\_hhvm\_ro.vw\_hhv\_ibro\_subscription\_base`

WHERE snapshot\_date = DATE '2024-10-01'

AND household\_segment LIKE '%Unconverged%'

AND service = 'Mobile'

ORDER BY acx\_household\_key

),

-- Step 2: Converged Base (from the first query)

converged\_base AS (

SELECT

mobile\_uncovered\_base.\*,

CASE

WHEN converged\_base.single\_id IS NULL THEN 0

ELSE 1

END AS converged\_flag

FROM mobile\_uncovered\_base

LEFT JOIN `bt-bvp-dataprod-dp-prod.cons\_bvp\_dp\_hhvm\_ro.vw\_hhv\_ibro\_subscription\_base` converged\_base

ON mobile\_uncovered\_base.single\_id = converged\_base.single\_id

AND converged\_base.snapshot\_date = DATE '2024-10-08'

AND converged\_base.household\_segment LIKE '%Converged%'

AND converged\_base.service = 'Broadband'

),

-- Step 3: Pages Viewed (from the second query)

pages\_viewed AS (

SELECT \*

FROM bt.con.digital.dp-data.digital\_touchpoints\_rw.dp\_024\_ee\_pages\_viewed

WHERE hit\_date\_time >= '2024-11-12' AND hit\_date\_time < '2024-11-13'

AND platform = 'WEB'

),

-- Step 4: Checkout (from the second query)

checkout AS (

SELECT channel\_visit\_id, journey\_step, journey\_action

FROM bt.con.digital.dp-data.digital\_touchpoints\_rw.dp\_008\_ee\_checkout

WHERE hit\_date\_time >= '2024-11-12' AND hit\_date\_time < '2024-11-13'

),

-- Step 5: Qualified Journeys (from the second query)

qualified\_journeys AS (

SELECT DISTINCT pv.channel\_visit\_id

FROM pages\_viewed pv

WHERE pv.page\_number = 1 AND pv.page IN ('Springboard')

),

-- Step 6: Filtered Journeys (from the second query)

filtered\_journeys AS (

SELECT

pv.\*,

c.journey\_action,

c.journey\_step AS checkout\_step

FROM pages\_viewed pv

LEFT JOIN checkout c ON pv.channel\_visit\_id = c.channel\_visit\_id

JOIN qualified\_journeys qj ON pv.channel\_visit\_id = qj.channel\_visit\_id

),

-- Step 7: Combined Data (Join Mobile Data with Filtered Journeys)

combined\_data AS (

SELECT

fj.\*,

cb.\*

FROM filtered\_journeys fj

LEFT JOIN converged\_base cb

ON fj.channel\_visit\_id = cb.acx\_household\_key

)

-- Final Step: Select Data with Specific Conditions

SELECT \*

FROM combined\_data

WHERE converged\_flag = 1;