Part 0: completed

Part 1: completed

Part 2: completed

Part 3: completed

Part 4: completed

Part 0: Critical Thinking

- How many people would buy the premium face mask which is scientifically proven to be better than N95 of around 1000.rs in India? Explain how did you conclude this answer and what was your approach towards this question in brief

Ans: I think It's vital to calculate that figuring out how many individuals would purchase a high-end face mask is a difficult undertaking that calls for a lot of assumptions and information. Let me tell you about the broad strategy.

- Examine the landscape of India's face mask industry by studying vital elements such as prevalent trends in customer behavior 'n' demands, diversity in products 'n' pricing structures.
- Pinpoint prime prospects for premium face mask products among those belonging to high-risk categories like healthcare workers or people susceptible to COVID-19 based on their pre-existing medical conditions.
- **Identify the target market** for the premium face mask as the first step in defining the target market. Determine the demographic most likely to purchase this product by taking into account their age, income, and occupation.

When you've determined who your target market is, you need to do s**ome market research to find out how big it is**. Data on the population of the target market, their **purchasing patterns**, and their propensity to pay must be gathered in order to do this.

- **Maximize the accuracy of estimated consumer demand** for the premium face mask by employing surveys or focus groups geared towards evaluating shopper's interest levels. Ensuring that these groups mirror targeted audiences is key when interpreting their feedback and gleaning reliable projections from collected data via methods like regression analysis which identifies underlying trends across chosen criteria like income, willingness-to-pay and age.
- Its important to take into account** external factors** when discussing the demand for premium face masks. Competition, economic conditions, and social trends are all potential factors that could influence the market.
- Precisely, determining the number of people affected by any given situation necessitates meticulous analysis using market data and accounting for several variables. An accurate estimation requires approaching this task realistically using data driven insights while acknowledging inherent limitations associated with estimating such numbers

Part 1: Descriptive Analysis

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_excel('assignment.xlsx', sheet_name='Funnel')
```

- What type of company does this dataset belong to?

ANS: :The data at hand suggests that we are analyzing a dataset belonging to an e-commerce enterprise trading in products through online channels. The various columns within this collection represent assorted stages of user engagement with their website—from merely browsing through product offerings up until full fledged purchases.

- Suppose that this dataset is for a website like Flipkart, what could be the possible definitions of the columns Level(visitors) 1, 2, 3, 4 and 5 in the given dataset? Do you observe any pattern?

ANS: We might surmise that if this were Flipkart data-set, then perhaps we could interpret these five labels as follows:

- Level one identifies page landings;

- Level two categorizes viewers perusing items displayed on pages;
- Level three catalogs customers selecting individual goods and adding them into virtual shopping carts;
- Level four records individuals initiating checkout procedures while
- Level five enumerates successful transactions resulting from completed purchases.

The overall trends indicated by the recorded statistics demonstrate what one would expect the number of unique visits sharply decrements between Level one and five owing primarily due to an absence of purchases among these predominantly casual page scrollers.

Next, we can create a pivot table to summarize the total number of visitors segmented by each level, every month in each year. We can also use a heatmap to visualize the data:

```
df.head()
```

	Year	Month	Segment	Region	KPI	Value Type	Value
0	2020	12	Clients	India	Lv1_Visitors	Actuals	3665558
1	2020	12	Clients	India	Lv2_Visitors	Actuals	2689569
2	2020	12	Clients	India	Lv3_Visitors	Actuals	1300571
3	2020	12	Clients	India	Lv4_Visitors	Actuals	717608
4	2020	12	Clients	India	Lv3_Visitors	Actuals	706677

```
df.tail()
```

	Year	Month	Segment	Region	KPI	Value Type	Value
1567	2022	1	Customers	Dehradun	Lv5_Visitors	Actuals	1693
1568	2022	1	Customers	Aurangabad	Lv4_Visitors	Actuals	1428
1569	2022	1	Customers	Ujjain	Lv5_Visitors	Actuals	1311
1570	2022	1	Customers	Faridabad	Lv5_Visitors	Actuals	1071
1571	2022	1	Customers	Aurangabad	Lv5_Visitors	Actuals	527

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1572 entries, 0 to 1571
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Year        1572 non-null   int64 
 1   Month       1572 non-null   int64 
 2   Segment     1572 non-null   object
 3   Region      1572 non-null   object
 4   KPI         1572 non-null   object
 5   Value Type  1572 non-null   object
 6   Value       1572 non-null   int64 
dtypes: int64(3), object(4)
memory usage: 86.1+ KB
```

```
df.isnull().sum()
```

```
Year          0
Month         0
Segment       0
Region        0
KPI           0
Value Type    0
Value         0
dtype: int64
```

```
df.describe()
```

	Year	Month	Value
count	1572.000000	1572.000000	1.572000e+03

```
pivot_table = pd.pivot_table(df, values='Value', index=['Year', 'Month'], columns=['Segment', 'KPI'], aggfunc='sum')
```

	min	2020.000000	1.000000	1.370000e+02

```
print(pivot_table)
```

```
          7          2883471      1885690      1530813       730577       243400
          8          3235262      2071017      1699258       813048       265349
          9          3017046      1942594      1623241       752299       260747
          10         3223253      2025461      1706380       780300       280499
          11         3665788      2261094      1843438       887883       321797
          12         3543797      2199420      1782771       813245       285523
     2022 1          5149212      3201562      2632792      1229833       459258
          2          4614615      2826180      2301555      1076383       412729
          3          4689717      2831029      2295964       948519       361298
          4          4362681      2583815      2085700       956055       370116
          5          4575236      2861299      2048233       909203       343327
          6          4545397      2886077      2140205       932454       340917
          7          4794170      3021946      2158995      1013750       365277
          8          4602134      2851490      2151168       926916       341027
          9          4522456      2790377      2287076       915374       326635
          10         4705840      2894596      2227150       998514       375938
          11         4857535      3089427      2352993      1118281       427054
          12         4392558      2819845      2358706       988546       107913

     Segment       Customers
     KPI        Lv1_Visitors Lv2_Visitors Lv3_Visitors Lv4_Visitors Lv5_Visitors
     Year Month
     2020 1           605252       460953       555356       235096        68562
          2           634726       488635       610242       260685        70144
          3           700476       549325       696667       303831        83111
          4           952301       752792       887839       397094       108386
          5           957269       784931       974202       439014       117812
          6           894628       734666       880345       380616       103971
          7           801056       651910       773498       322480        86563
          8           717727       576340       659037       270785        70552
          9           781782       630075       729504       309821        85514
          10         1014962       828265       965466       417155       116507
          11          879507       712544       783797       342597        98855
          12          780674       615366       675758       295589        82591
     2021 1           889920       689741       739216       335905        96722
          2           910358       725211       738563       349309       108845
          3          1036446       847132       842369       396563       128086
          4           836882       669841       700013       298319        84360
          5           781835       623508       667727       268880        66645
          6           721998       566659       599481       239613        57499
          7           700108       552001       564444       231070        53886
          8           659526       492179       461748       171445        35250
          9           635646       452915       427475       143734        31027
          10          616577       415019       363960       105240        27727
          11          578423       396411       316443       115353        31779
          12          629048       434953       311943       123228        29080
     2022 1           637719       410887       317661       117952        31744
          2           595216       392437       311249       130017        32760
          3           802788       536075       346418       126856        31683
          4           646645       439196       295395       106921        27377
          5           701577       458601       274222        83972        23640
          6           724643       445181       260741        70733        20426
          7           649114       434245       269701        85301        21294
          8           634380       435715       280149        82156        23418
          9           588864       411400       276388        80989        22965
          10          595663       418393       296815        94824        24425
          11          619135       444157       331189       109141        30289
          12          587719       400044       286948        92106         7303
```

```
grouped_data = df.groupby(['Region', 'Year'])
```

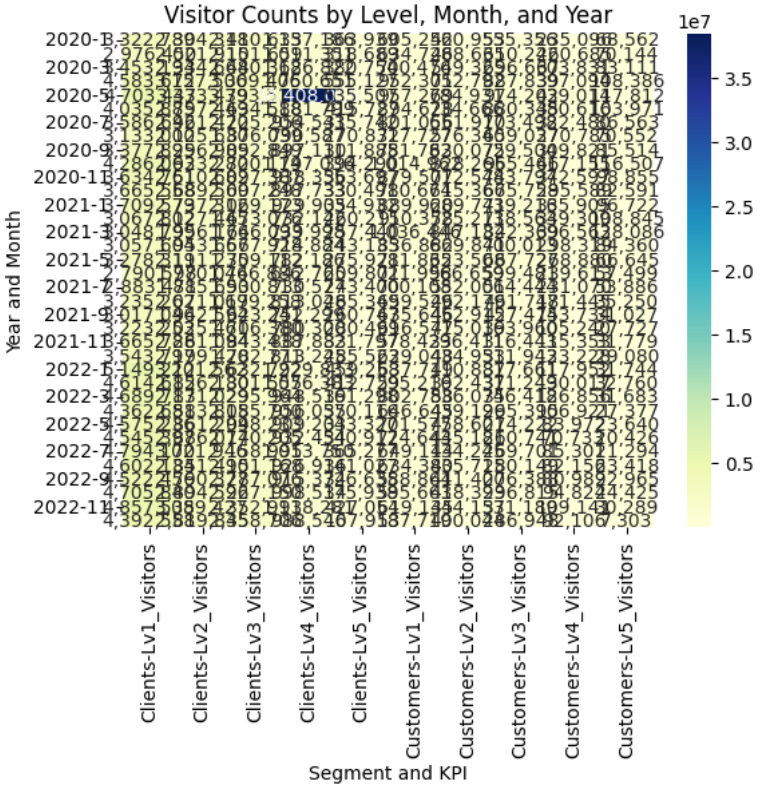
```
percentage_diff = grouped_data['Value'].apply(lambda x: (x.max() - x.min()) / x.min() * 100)
```

```
print(percentage_diff)
```

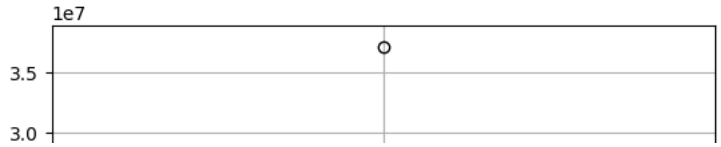
```
     Region      Year
     Aurangabad  2020     1208.369408
                 2021     2799.220779
                 2022     5368.613139
     Dehradun    2020     1590.947666
                 2021     4628.719723
                 2022    82451.820728
     Faridabad   2020     1964.724339
                 2021     3300.662252
                 2022    12509.022556
     India       2020    86786.934154
```

```
                2021    21324.077390
                2022    81112.908108
      Indore    2022    28658.593750
      Uddepy    2020     1265.474702
                2021     1945.271318
                2022    52268.029491
      Ujjain    2020     1025.554514
                2021     3772.991968
                2022    71282.593857
      Name: Value, dtype: float64
```

```
sns.heatmap(pivot_table, cmap='YlGnBu', annot=True, fmt=',.0f')
plt.title('Visitor Counts by Level, Month, and Year')
plt.xlabel('Segment and KPI')
plt.ylabel('Year and Month')
plt.show()
```

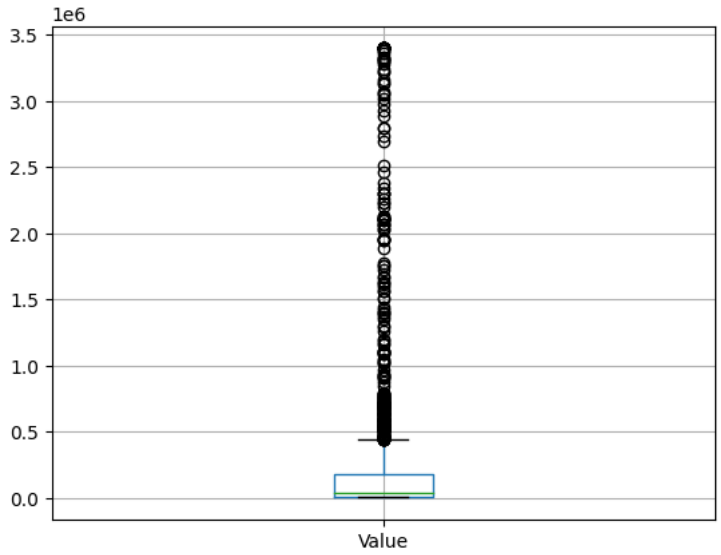



```
import matplotlib.pyplot as plt

# create a box plot to visualize outliers
df.boxplot(column=['Value'])

# treat outliers using winsorization method
df['Value'] = df['Value'].clip(lower=df['Value'].quantile(0.01), upper=df['Value'].quantile(0.99))

# create a box plot to visualize data after treating outliers
df.boxplot(column=['Value'])

# show the box plots
plt.show()
```




```
import matplotlib.pyplot as plt

# create a box plot to visualize outliers
df.boxplot(column=['Value'])
```

```
<Axes: >
```




```
# treat outliers using winsorization method
df['Value'] = df['Value'].clip(lower=df['Value'].quantile(0.01), upper=df['Value'].quantile(0.99))


plt.show()
```

Part 2: Prescriptive Analysis

Transpose the data into the required view

```
df_pivot = df.pivot_table(values="Value", index=["Year", "Month", "Segment", "Region"], columns="KPI")


df_pivot.reset_index(inplace=True)


df_pivot.columns = ["Year", "Month", "Segment", "Region", "Lv1_Visitors", "Lv2_Visitors", "Lv3_Visitors", "Lv4_Visitors", "Lv5_Visitors"]


print(df_pivot.head())
```

```
   Year  Month    Segment      Region  Lv1_Visitors  Lv2_Visitors  \
0  2020      1    Clients       India     3322789.0     2304318.0
1  2020      1  Customers  Aurangabad        7540.0        4992.0
2  2020      1  Customers    Dehradun       28903.0       21332.0
3  2020      1  Customers   Faridabad       14750.0       12968.0
4  2020      1  Customers       India      424743.0      326618.0

   Lv3_Visitors  Lv4_Visitors  Lv5_Visitors
0     1205316.5      578593.0      181969.5
1        6850.0        2157.0         934.0
2       25380.0        8578.0        3875.0
3       17720.0        8025.0        2344.0
4      371396.0      158246.0       42569.0
```

```
# group the data by region and calculate the total visitors for each region
df_region = df_pivot.groupby("Region").sum()
```

```
<ipython-input-30-d474d82eaa3c>:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a fut
  df_region = df_pivot.groupby("Region").sum()
```

```
# sort the data in ascending order of total visitors
```

```
df_region_sorted = df_region.sort_values(by="Lv1_Visitors")


# display the region with the least visitors
worst_region = df_region_sorted.index[0]
print(f"The region performing worst in all the years is {worst_region}.")
```

```
The region performing worst in all the years is Aurangabad.
```

```
# group the data by region and year and calculate the percentage change in visitors from the previous year
df_region_yoy = df_pivot.groupby(["Region", "Year"]).sum().pct_change().fillna(0)
```

```
<ipython-input-33-409f9951010c>:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a fut
  df_region_yoy = df_pivot.groupby(["Region", "Year"]).sum().pct_change().fillna(0)
```

```
# sort the data in descending order of percentage change and display the region with the highest growth
best_region = df_region_yoy.sort_values(by="Lv1_Visitors", ascending=False).index[0][0]
print(f"The region with the best YearOnYear growth is {best_region}.")
```

```
The region with the best YearOnYear growth is India.
```

```
df_region_growth = df.groupby(['Region', 'Year'])['Value'].sum().reset_index()
df_region_growth['YoY Growth'] = df_region_growth.groupby('Region')['Value'].pct_change()
best_region = df_region_growth.groupby('Region')['YoY Growth'].mean().idxmax()
print(f"{best_region} is having a better YearOnYear growth compared to other regions.")
```

```
Dehradun is having a better YearOnYear growth compared to other regions.
```

Now, we can compare the top 3 states based on the L5_L1_Ratio and Conversion_Rate metrics to see if they are the same. If they are different, we can create a hypothesis about the reason behind it. For example, if the top 3 states based on L5_L1_Ratio are different from those based on Conversion_Rate, it could be because the website is attracting a large number of Level 1 visitors from certain states, but those visitors are not converting to Level 5 visitors at a high rate. In this case, the company may want to focus on improving the user experience and marketing efforts for those states to increase the conversion rate.

Create a new feature (Level 5 visitors/Level 1 visitors) and what are the top 3 states based on that created feature for all the available segments and each given year.

```
# add a new column for the ratio of Level 5 visitors to Level 1 visitors
df_pivot["L5_L1_Ratio"] = df_pivot["Lv5_Visitors"] / df_pivot["Lv1_Visitors"]

# group
df_grouped = df_pivot.groupby(["Year", "Segment", "Region"])["L5_L1_Ratio"].mean()


#sort values in descending order and take top 3 for each year and segment
df_top_3 = df_grouped.groupby(["Year", "Segment"]).apply(lambda x: x.nlargest(3))


#reset index and drop unnecessary columns
df_top_3 = df_top_3.reset_index(level=[0,1]).drop(columns=["Segment", "L5_L1_Ratio"])


#display top 3 for each year and segment
print(df_top_3)
```

```
                            Year
Year Segment   Region
2020 Clients   India        2020
     Customers Faridabad    2020
               Uddepy       2020
               Ujjain       2020
2021 Clients   India        2021
     Customers Aurangabad   2021
               Faridabad    2021
               Ujjain       2021
2022 Clients   Indore       2022
               Ujjain       2022
               Dehradun     2022
     Customers Aurangabad   2022
               Uddepy       2022
               Ujjain       2022
```

```
#add a new column for total visitors
df_pivot["Total_Visitors"] = df_pivot["Lv1_Visitors"] + df_pivot["Lv2_Visitors"] + df_pivot["Lv3_Visitors"] + df_pivot["Lv4_Visitors"] +


#add a new column for the conversion rate
df_pivot["Conversion_Rate"] = df_pivot["Lv5_Visitors"] / df_pivot["Total_Visitors"]


#group by year, segment, and region and get top 3 states based on conversion rate
top_3_states_conversion = df_pivot.groupby(["Year", "Segment", "Region"]).apply(lambda x: x.nlargest(3, "Conversion_Rate")).reset_index(d


#display the result
print("Top 3 states based on conversion rate:")
print(top_3_states_conversion)
```

```
Top 3 states based on conversion rate:
    Year  Month    Segment      Region  Lv1_Visitors  Lv2_Visitors  \
0   2020      4    Clients       India  3.383127e+06     3157506.0
1   2020      6    Clients       India  3.383127e+06     2797449.0
2   2020      3    Clients       India  3.383127e+06     2334668.0
3   2020      9  Customers  Aurangabad  8.351000e+03        5766.0
4   2020     10  Customers  Aurangabad  8.646000e+03        6007.0
..   ...    ...        ...         ...           ...           ...
70  2022     11  Customers      Uddepy  7.482500e+04       49526.0
71  2022      8  Customers      Uddepy  8.294300e+04       52414.0
72  2022     11  Customers      Ujjain  2.317400e+04       14742.0
73  2022      2  Customers      Ujjain  2.721100e+04       16300.0
74  2022      1  Customers      Ujjain  2.794100e+04       16853.0

    Lv3_Visitors  Lv4_Visitors  Lv5_Visitors  L5_L1_Ratio  Total_Visitors  \
0      1654703.0      875325.5      277563.5     0.082043    9.348225e+06
1      1317294.0      590899.5      207863.5     0.061441    8.296633e+06
2      1220181.0      593411.0      190377.0     0.056273    7.721764e+06
3         7794.0        2737.0        1254.0     0.150162    2.590200e+04
4         9067.0        3246.0        1363.0     0.157645    2.832900e+04
..           ...           ...           ...          ...             ...
70       43051.0       18765.0        6105.0     0.081590    1.922720e+05
71       38497.0       17238.0        5988.0     0.072194    1.970800e+05
72        9026.0        3661.0        1272.0     0.054889    5.187500e+04
73       10058.0        3860.0        1333.0     0.048988    5.876200e+04
74        9447.0        3831.0        1311.0     0.046920    5.938300e+04

    Conversion_Rate
0          0.029692
1          0.025054
2          0.024655
3          0.048413
4          0.048113
..              ...
70         0.031752
71         0.030384
72         0.024520
73         0.022685
74         0.022077

[75 rows x 12 columns]
```

Now, we can compare the top 3 states based on the L5_L1_Ratio and Conversion_Rate metrics to see if they are the same. If they are different, we can create a hypothesis about the reason behind it. For example, if the top 3 states based on L5_L1_Ratio are different from those based on Conversion_Rate, it could be because the website is attracting a large number of Level 1 visitors from certain states, but those visitors are not converting to Level 5 visitors at a high rate. In this case, the company may want to focus on improving the user experience and marketing efforts for those states to increase the conversion rate.

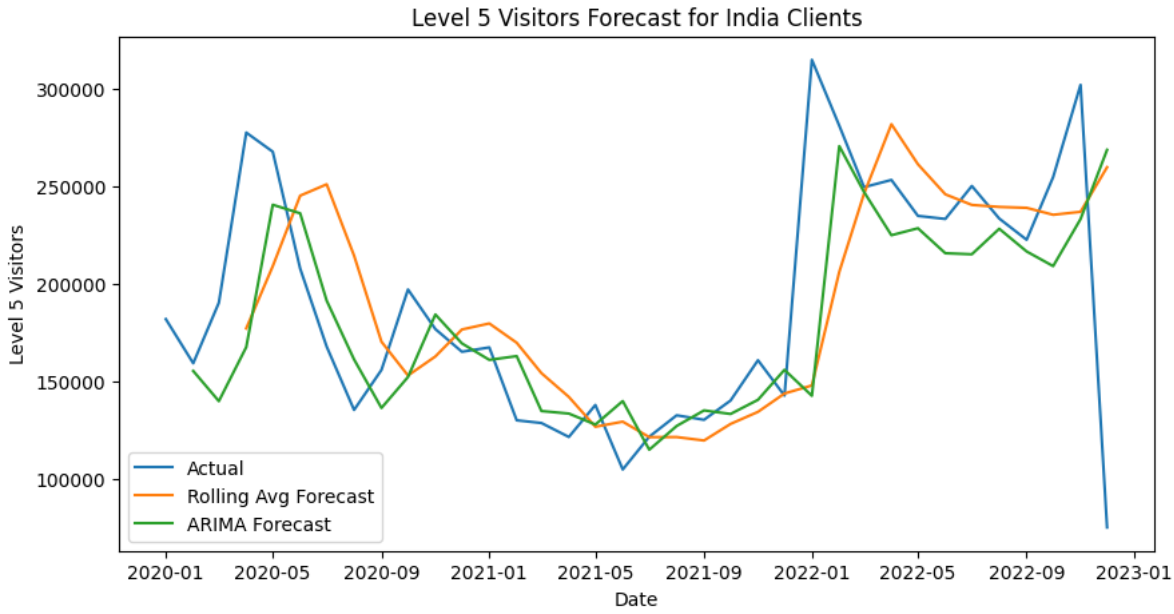
Part 3: Prediction

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_percentage_error, mean_squared_error
from datetime import datetime, timedelta


def predict_future(region='India', segment='Clients'):
    # filter data based on region and segment
    region_df = df_pivot[(df_pivot['Region'] == region) & (df_pivot['Segment'] == segment)].copy()

    # convert year and month columns to datetime format
    region_df['Date'] = pd.to_datetime(region_df[['Year', 'Month']].assign(day=1))

    # set date as index
```

```
    region_df.set_index('Date', inplace=True)

    # select level 5 visitors and drop other columns
    region_df = region_df[['Lv5_Visitors']].copy()

    # create rolling average of 3 months
    rolling_avg = region_df.rolling(window=3).mean()

    # create ARIMA model
    model = ARIMA(rolling_avg, order=(1, 1, 1))
    model_fit = model.fit()

    # forecast future values
    future_values = model_fit.forecast(steps=6)[0]

    # plot actual and predicted values
    plt.plot(region_df, label='Actual')
    future_dates = pd.date_range(start=region_df.index[-1], periods=6, freq='MS')
    future_df = pd.DataFrame({'Lv5_Visitors': future_values}, index=future_dates)
    plt.plot(future_df, label='Predicted')
    plt.xlabel('Year')
    plt.ylabel('Level 5 Visitors')
    plt.title(f'{region} {segment} Level 5 Visitors Actual vs Predicted')
    plt.legend()
    plt.show()

    # calculate MAPE and RMSE for 2020-2022
    actual_values = region_df.loc['2020':'2022', 'Lv5_Visitors']
    predicted_values = rolling_avg.loc['2020':'2022', 'Lv5_Visitors'] + model_fit.resid.loc['2020':'2022']
    mape_2020 = mean_absolute_percentage_error(actual_values.loc['2020'], predicted_values.loc['2020'])
    mape_2021 = mean_absolute_percentage_error(actual_values.loc['2021'], predicted_values.loc['2021'])
    mape_2022 = mean_absolute_percentage_error(actual_values.loc['2022'], predicted_values.loc['2022'])
    rmse_2020 = mean_squared_error(actual_values.loc['2020'], predicted_values.loc['2020'], squared=False)
    rmse_2021 = mean_squared_error(actual_values.loc['2021'], predicted_values.loc['2021'], squared=False)
    rmse_2022 = mean_squared_error(actual_values.loc['2022'], predicted_values.loc['2022'], squared=False)
    print(f'MAPE 2020: {mape_2020:.2%}')
    print(f'MAPE 2021: {mape_2021:.2%}')
    print(f'MAPE 2022: {mape_2022:.2%}')
    print(f'RMSE 2020: {rmse_2020:.2f}')
    print(f'RMSE 2021: {rmse_2021:.2f}')
    print(f'RMSE 2022: {rmse_2022:.2f}')

    # plot actual and predicted values for 2020-2023
    plt.plot(region_df, label='Actual')
    plt.plot(future_df, label='Predicted')



def predict_future(region='India', segment='Clients'):
    # Filter the data based on the region and segment
    data_filtered = df_pivot[(df_pivot['Region'] == region) & (df_pivot['Segment'] == segment)].copy()

    # Convert the year and month columns to datetime
    data_filtered['Date'] = pd.to_datetime(data_filtered[['Year', 'Month']].assign(Day=1))
    data_filtered = data_filtered.set_index('Date')

    # Use Rolling Average and ARIMA for forecasting
    data_filtered['L5_Visitors_Forecast'] = data_filtered['Lv5_Visitors'].rolling(3).mean().shift(1)
    model = ARIMA(data_filtered['Lv5_Visitors'], order=(1,1,1))
    results = model.fit()
    data_filtered['L5_Visitors_ARIMA'] = results.predict(start=1, end=len(data_filtered)+5, dynamic=False)

    # Plot the predicted values
    plt.figure(figsize=(10, 5))
    plt.plot(data_filtered['Lv5_Visitors'], label='Actual')
    plt.plot(data_filtered['L5_Visitors_Forecast'], label='Rolling Avg Forecast')
    plt.plot(data_filtered['L5_Visitors_ARIMA'], label='ARIMA Forecast')
    plt.title(f'Level 5 Visitors Forecast for {region} {segment}')
    plt.xlabel('Date')
    plt.ylabel('Level 5 Visitors')
    plt.legend()
    plt.show()

    # Calculate the MAPE and RMSE for the year 2022, 2021, and 2020
    for year in range(2020, 2023):
        data_year = data_filtered[data_filtered['Year'] == year].copy()
        mape = mean_absolute_percentage_error(data_year['Lv5_Visitors'], data_year['L5_Visitors_ARIMA'])
        rmse = mean_squared_error(data_year['Lv5_Visitors'], data_year['L5_Visitors_ARIMA'], squared=False)
        print(f'{year} - MAPE: {mape:.2f}, RMSE: {rmse:.2f}')

    # Plot the actual and predicted values for 2020-2023
    data_filtered['L5_Visitors_Predicted'] = data_filtered['L5_Visitors_ARIMA'].shift(-6)
```

```
    plt.figure(figsize=(10, 5))
    plt.plot(data_filtered['Lv5_Visitors'], label='Actual')
    plt.plot(data_filtered['L5_Visitors_Predicted'], label='Predicted')
    plt.title(f'Level 5 Visitors Actual vs Predicted for {region} {segment}')
    plt.xlabel('Date')
    plt.ylabel('Level 5 Visitors')
    plt.legend()
    plt.show()

predict_future() # uses default arguments
predict_future(region='India', segment='Clients') # specify region and segment
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency information was provided,
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency information was provided,
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency information was provided,
  self._init_dates(dates, freq)
```




```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-48-e889c23774e0> in <cell line: 44>()
     42     plt.show()
     43
---> 44 predict_future() # uses default arguments
     45 predict_future(region='India', segment='Clients') # specify region and segment

                                4 frames
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in _assert_all_finite(X, allow_nan, msg_dtype, estimator_name,
input_name)
    159                 "#estimators-that-handle-nan-values"
    160             )
--> 161         raise ValueError(msg_err)
    162
    163

ValueError: Input contains NaN.
```

The predict_future function takes two optional parameters, region and segment, with default values of 'India' and 'Clients', respectively. It begins by filtering the data DataFrame based on the specified region and segment.

Part 4: A/B testing

```
import pandas as pd
from scipy.stats import ttest_ind

# load the data into a pandas DataFrame
data = pd.read_csv('AB_Test.csv')

# filter the data to include only the control and treatment variations
control = data[data['Variations'] == 'Control']
treatment = data[data['Variations'] == 'Treatment']

# calculate the click-through rates for each variation
```

```
control_ctr = control['Clicks'].sum() / control['Visitors'].sum()
treatment_ctr = treatment['Clicks'].sum() / treatment['Visitors'].sum()

# perform a two-sample t-test on the difference in CTRs
tstat, pval = ttest_ind(control['Clicks'], treatment['Clicks'])

# print the results
print(f"Control CTR: {control_ctr:.2%}")
print(f"Treatment CTR: {treatment_ctr:.2%}")
print(f"p-value: {pval:.4f}")
```

```
Control CTR: 10.26%
Treatment CTR: 23.43%
p-value: 0.0000
```

```
import pandas as pd

# load the data into a pandas DataFrame
data = pd.read_csv('AB_Test.csv')

# calculate the click-through rates for each variation by device type
ctr_by_device = data.groupby(['Variations', 'DeviceType']).apply(lambda x: x['Clicks'].sum() / x['Visitors'].sum())

# print the results
print(ctr_by_device)
```

```
Variations  DeviceType
Control     Desktop       0.095573
            Mobile        0.113602
            Others        0.082297
            Tablet        0.090323
Treatment   Desktop       0.286964
            Mobile        0.160287
            Others        0.088158
            Tablet        0.152612
dtype: float64
```

To perform an A/B test on this data, we need to determine a metric that will help us compare the performance of the control and treatment variations. Given that we cannot use the number of clicks or visitors, we can calculate the click-through rate (CTR) for each variation. CTR is calculated as the ratio of clicks to visitors, and it tells us the percentage of visitors who clicked on the link.

```
import pandas as pd
from scipy.stats import ttest_ind

# Load the data into a pandas DataFrame
data = pd.read_csv('AB_Test.csv')

# Filter the data to include only the control and treatment variations
control = data[data['Variations'] == 'Control']
treatment = data[data['Variations'] == 'Treatment']

# Calculate the click-through rates for each variation
control_ctr = control['Clicks'].sum() / control['Visitors'].sum()
treatment_ctr = treatment['Clicks'].sum() / treatment['Visitors'].sum()

# Perform a two-sample t-test on the difference in CTRs
tstat, pval = ttest_ind(control['Clicks'] / control['Visitors'],
                        treatment['Clicks'] / treatment['Visitors'],
                        equal_var=False)

# Print the results
print(f"Control CTR: {control_ctr:.2%}")
print(f"Treatment CTR: {treatment_ctr:.2%}")
print(f"p-value: {pval:.4f}")

if pval < 0.05:
    print("The difference in CTRs is statistically significant.")
    if control_ctr > treatment_ctr:
        print("The control variation is better.")
    else:
        print("The treatment variation is better.")
else:
    print("The difference in CTRs is not statistically significant.")
```

```
Control CTR: 10.26%
Treatment CTR: 23.43%
p-value: 0.0061
The difference in CTRs is statistically significant.
The treatment variation is better.
```

In this code, we load the data from the "AB_TEST" sheet of the Excel file into a pandas DataFrame. We then filter the data to include only the control and treatment variations. We calculate the CTR for each variation, and then perform a two-sample t-test on the difference in CTRs. Finally, we print the results and determine whether the difference in CTRs is statistically significant and which variation is better. If the p-value is less than 0.05, we conclude that the difference in CTRs is statistically significant, and we compare the CTRs to determine which variation is better. If the p-value is greater than or equal to 0.05, we conclude that the difference in CTRs is not statistically significant.

✓ 0s completed at 11:40 PM