Start Dt - 10 Feb 24

Pandas documentation original cook book
< pandas.pydata.org > Ref
o Read on txt data & categorical data

Applied Data Science with Python

Course-End Project Problem Statement

Background Study → How marketing campaigns & surveys are designed. Data collected. Obj of survey.

This is core stats

Phase-End Project: Marketing Campaigns

Problem Scenario: 'Marketing mix' is a popular concept used in implementing marketing strategies. A marketing mix includes multiple areas of focus as part of a comprehensive marketing plan. This all revolves around the four Ps of marketing - product, price, place, and promotion.

Buzz words: Exploratory Data Analysis

Problem Objective: As a data scientist, you should perform exploratory data analysis and hypothesis testing. The goal is to gain a better understanding of the various factors that contribute to customer acquisition.

EDA

Data Description:
The variables birth-year, education, income, and so on are related to the first 'P' or 'People' in the tabular data provided to the user. The amount spent on wine, fruits, gold, etc., is related to 'Product'. The information pertinent to sales channels, like websites, stores, etc., is related to 'Place', and the fields which talk about promotions and results of different campaigns are related to 'Promotion'.

Steps to Perform:

1. Once data is imported, investigate variables like Dt_Customer and Income, etc., and check if they are imported correctly.

2. Income values for a few customers are missing. Perform missing value imputation. Assume that the customers with similar education and marital status make the same yearly income, on average. You may have to clean the data before performing this. For data cleaning, look into the categories of education and marital status.

3. Create variables to populate the total number of children, age, and total spending.

   Hint: From the number of purchases through the three channels, people can derive the total purchases.

   I didn't get this 😕

4. Create box plots and histograms to understand the distributions and outliers. Perform outlier treatment.

5. Use ordinal encoding and one hot encoding according to different types of categorical variables.

6. Create a heatmap to showcase the correlation between different pairs of variables.

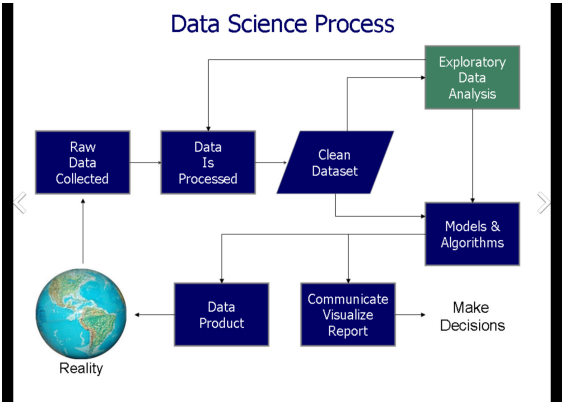
7. Test the following hypotheses:
   - Older people are not as tech-savvy and probably prefer shopping in-store.
   - Customers with kids probably have less time to visit a store and would prefer to shop online.
   - Other distribution channels may cannibalize sales at the store.
   - Does the US fare significantly better than the rest of the world in terms of total purchases?

8

- Use appropriate visualization to help analyze the following:
  - Which products are performing the best, and which are performing the least in terms of revenue?
  - Is there any pattern between the age of customers and the last campaign acceptance rate?
  - Which Country has the greatest number of customers who accepted the last campaign?
  - Do you see any pattern in the no. of children at home and total spend?
  - Education background of the customers who complained in the last 2 years.

Notes on EPA

- This is main obj & learning value of proj.
- Key features



⇒ Study
⇒ Find traits
⇒ Discover pattern
⇒ outliers iden
⇒ Relationship Model
⇒ Take decision

The data cly & descriptiv stats & visualistion are basic mechanical.

- Feture Engg → Tweak variables new functin

Correlation & Relationship.

Iden dependencies of variables
direction of relationship

Generate a hypothesis. Hypothesis is generally made at stage of preliminary data exploration based on a pilot study.

Data Quality analysis – EDA checks integrity and reliability of data.

Exec of Proj.

- Data is in a csv file. It is intentionally unclean.
- The (8) steps are documented

(1) **Import data**

This time again I prefered cleaning it in excel Power Query. However I have written python code for cleaning in code blocks to understand Command

Python proces for colab

→ Import colab snippet for loading from Google drive. This is most convinient & versatile

→ convert dtypes

→ Convert Dt Customer to date field
→ County, Marital status Edu to str

← There will be some issue in date conversions → iden error and chang item manually.

Unused code snippet in notebook.

Upload cleaned csv to google drive else run whole code again.

(2) Blanks in income value.

There are 20 odd blanks in income
These are to be filled with mean of grouping 'Education & Marital Status'.
Logic is sound & practical.

③ Addition of variables.

Seemed pretty straight

- No of kids = Kids at home + Teens
- Age = Current year − year of birth
- Total spending = Amt on (Wines, Fruit, Meat, Fish, sweet, gold)

(60) Hint is confusing. CCP proj from America context. Oversight likely.

② Categories

The Marital Status cat has many American acronyms like YOLO, Together, Absurd ⇒ Clean them Again linked to step 6.

What's relevance of Widow, Single YOLO on shopping pattern ⇒ They are all single & Lonely!

Marital Status into only ⇒ Together
Single

Reduces degree of freedom
normalised data easier to interpret

→ Atleast for first proj will make it less complex.

Similarly Education reduced to
SSC-Plus < Graduate < Masters
III categories.
This is ordinal data.

④ Find outliers →

df.info() get descriptive stats
examine visually
Only Income has a outlier. 66666
Clean manually or run algo.
→ I have a algo made for future projects.
→ Box plot is standard
→ Winsorize to set outliers to lower/upper bound

Options: Winsorize or clip
Winsorize needs to be carefully used
I iterated on my binary data - Promotion campaign & it got set to zero.
Exclude colums that are binary.

Examine data on histogram & box charts for every variable

⑤ Ordinal Encoding & Heat Map

Ordinal data - Education.
I have clubbed it into 3 categories
SSC-Plus, Graduation, Master
to Indian context. Reduced the df to make model simpler.

Non Ordinal → Used one hot encoding
→ Splits colums makes data huge.

Reduced dimensionality of Marital Status to Together, Single
Rest all subsumed

For correlation Matrix →

Reduce variables to make Heat map readable

(6) Hypothesis testing

(A) Older ppl like stores

Used Chisq simple to apply

It is not ideal but easy to implement. State Null Hypothesis

Find p value Reject/Accept.

(B) Customers with kids spend more online

- Club Kids in a binary column

Apply Chisq test against NuWeb Purchases

↓

Tried Logistic Regression for same hypothesis

tested has kids, Store Category & all numeric cols against tgt "Nuweb"

Tough to write code for regression labs / ran trouble shoot.

use iloc to list parameters in model fit

C] Distr channels cannibalise store sales

3 x t tests
→ Store Vs Online + Cat Vs
⇒ Store Vs both
⇒ Online Cat Vs both

Refer next PS

Divide α/3

Null hypothesis rejected

∴

Logistic Regression also applied by incr variables on tgt

D] US sales Better

Std T test

Visualisations

Simple matplot implementation.

What to use where most imp

Tweak & make your scripts for other projects. Age Vs last campaign interesting

Bonferroni Correction

1 Multiple Hypothesis Tests:
◦Imagine you're comparing means across three different groups (let's call them A, B, and C) using a t-test.
◦You want to determine if there are significant differences in some variable (e.g., average age) among these groups.
2 The Issue:
◦When you perform multiple t-tests, the chance of making a Type I error (i.e., rejecting a null hypothesis when it's actually true) increases.
◦Essentially, the more tests you run, the more likely you are to find a significant result purely by chance.
3 Bonferroni Correction:
◦The Bonferroni correction addresses this issue by adjusting the significance level for each individual test.
◦Instead of using the usual alpha (e.g., 0.05), you divide it by the number of tests you're conducting.
◦For three tests, the adjusted alpha becomes: (\frac{{0.05}}{{3}} = 0.0167).
4 Interpretation:
◦If the p-value from an individual t-test is less than or equal to 0.0167, you can consider it statistically significant after applying the Bonferroni correction.
◦This correction helps control the familywise error rate (the probability of making at least one Type I error across all tests).

Used in multiple t-sets

Machine Learning Modules

o ML modules make out of the box application of algos feasible with few lines of code ⇒ Focus on stats

o Focus on What algo to apply.

o All algos require data to meet certain condition: like normalisation

Done & Dusted - 11 Feb.

To do

- Check list for application of tests for hypothesis testing.
- Revisit Stats.
  - The maths I've to perfect test is all tools.

Good project plenty of learning.

Used Google Colab with data in G Drive

```
# -*- coding: utf-8 -*-
"""Marketing Campaign.ipynb
Automatically generated by Colaboratory.
Original file is located at
   https://colab.research.google.com/drive/11FEZlZ_5FRKmphBugYNZngVggGcmNgHt
"""
# @title Default title text
# prompt: import file from google drive into pandas data frame
import pandas as pd
from google.colab import drive
drive.mount('/content/drive')
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# Get the file path from Google Drive
file_path = '/content/drive/My Drive/Dataset/marketing_data.csv'
# Read the file into a Pandas DataFrame
df = pd.read_csv(file_path)
print(df.head())
print(df.info())
# Describe numerical variables to check for missing values, outliers, etc.
print(df.describe())
#df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'], format='%d-%m-%Y')
print(df.info())
"""This block was for data cleaning run for a import error in Dt Cystomer fd
only once if required"""
condition = df['Income'] > 68655
df_sorted = df[condition].sort_values(by='Income')
# Display the sorted DataFrame
print(df_sorted['Income'])
invalid_rows = df.loc[df["Dt_Customer"]== '2013-04-05']
df.loc[0, 'Dt_Customer'] = '2013-04-05'
print(invalid_rows)
print(df.head())
print(df.info())
df['Education'] = df['Education'].astype(str)
df.convert_dtypes()
# Rename the column
#df.rename(columns={"Money": "Income"}, inplace=True)
df['Marital_Status'] = df['Marital_Status'].astype(str)
df['Country'] = df['Country'].astype(str)
print(df.info())
"""Convert the 'Dt_Customer field to datetiem64. This is necessary to make
computations on dates"""
df['Dt_Customer']=pd.to_datetime(df['Dt_Customer'])
print (df['Dt_Customer'].dtype)
# Check no nulls in date field
df[df.notna()]
"""Data cleaning block for converting Income to float"""
# This code block is for converting currency to float64.
df['Income'] = df['Income'].str.replace(',', '').str.replace('$',
'').str.replace('.', '.', regex=False).astype(str)
print(df['Income'].dtype)
df['Income'].astype(float)
```

```
"""One time upload of my clean csv files ready for analysis. To be run only
once."""
df.to_csv(file_path, index=False)
"""This is trouble shooting code block as after grouping my Dtype of group
changed to object and couldnt do mathematical operations of padding data.
Refer fillna statement at line 8 which is # commented now"""
#print(df['Income'].head())
print(df['Income'].dtype)
#print(df.groupby(['Education', 'Marital_Status'])['Income'].head())
print(df.groupby(['Education', 'Marital_Status'])['Income'].dtype)
#print(df.groupby(['Education', 'Marital_Status'])['Income'].mean())
"""Find the
Data cleaning. CHeck for unique values in each categotical column
"""
print(df['Education'].unique())
print(df['Marital_Status'].unique())
print(df['Country'].unique())
#df.columns
# Identified categoroes like YOLO Alone aand Absurd filled durind survey are
rationalised
df[df['Marital_Status']=='YOLO']
df.loc[df['Marital_Status']=='YOLO','Marital_Status']="Single"
df[df['Marital_Status']=='Alone']
df.loc[df['Marital_Status']=='Alone','Marital_Status']="Single"
df[df['Marital_Status']=='Absurd']
df.loc[df['Marital_Status']=='Absurd','Marital_Status']="Single"
# Check the Education field for scope to reduce the dimentionality
df[df['Education']=='2n Cycle']
df.nunique(axis=0)
# Count unique values in the 'Education' column
unique_education_count = len(pd.unique(df['Education']))
print("Number of unique height values:", unique_education_count)
df.loc[df['Education']=='Basic','Education']="SSC"
df.loc[df['Education']=='SSC','Education']="SSC_plus"
df.loc[df['Education']=='2nd Cycle','Education']="SSC_plus"
edn=df[df['Education']=='SSC_plus']
# Club the PHD and Masters into masters
df.loc[df['Education']=='PhD','Education']="Master"
# Now I have just three education categories for my model
df.loc[df['Marital_Status']=='Widow','Marital_Status']="Single"
df.loc[df['Marital_Status']=='Divorced','Marital_Status']="Single"
df.loc[df['Marital_Status']=='Married','Marital_Status']="Together"
"""Reduce the dimensions of the Marital status to single and Married. Single
widow divorced subsumed in single and Married and Together clubbed"""
# How to list counts of categories
education_counts = df['Education'].value_counts()
print("Education value counts:\n", education_counts)
# SSC has just 45 respondants. Can i club this category with Diploma as it
is just 2% of the
"""At this stage my csv is clean and I should upload it to my drive to
prevent running through the same operation"""
df.to_csv(file_path, index=False)
# Check for missing values and their patterns
print(df.isnull().sum())
print(df.columns.tolist())
# Get current year for age calculation
# import the datetime module
import datetime
# get the current date and time
now = datetime.datetime.now()
```

```
# get the current year as a string in YYYY format
year = now.strftime("%Y")
int(year)
df['Year_Birth'] = df['Year_Birth'].astype(np.int64)
df.info()
type(year)
# Check for missing values and their patterns
print(df.isnull().sum())
print(df.columns.tolist())
# Get current year for age calculation
# import the datetime module
import datetime
# get the current date and time
now = datetime.datetime.now()
# get the current year as a string in YYYY format
year = now.strftime("%Y")
year=int(year)
df['Year_Birth'] = df['Year_Birth'].astype(np.int64)
#df['Income'].fillna(df.groupby(['Education', 'Marital_Status'])
['Income'].mean(), inplace=True)
#Use transform method as there is mismatch between spae of data frame and
the group !!!
df['Income'].fillna(df.groupby(['Education', 'Marital_Status'])
['Income'].transform('mean'), inplace=True)
# Alternate to group by
#df['Income'] = df.groupby(['Education', 'Marital_Status'])
['Income'].transform(lambda x: x.fillna(x.mean()))
# Create derived variables (e.g., total children, age, total spending) as
needed
df['Total_Children'] = df['Kidhome'] + df['Teenhome']
df['Age'] = year-df['Year_Birth']
df['Total_Spending'] = df[['MntWines', 'MntFruits', 'MntMeatProducts',
'MntFishProducts', 'MntSweetProducts', 'MntGoldProds' ]].sum(axis=1)
#Check
print(df.columns.tolist())
print(df.shape)
print(df.info())
print(df)
print(data)
"""#Outliers
Understand distr and remove outliers. I will winsorize outliers as deleted
fields will create nulls in data
"""
import scipy.stats as stats
df.columns
# Identify numerical columns for analysis
#Exclude your binaries . they got reset to zero with automated Winsorization
exclude_list = ['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5',
'AcceptedCmp1',
      'AcceptedCmp2', 'Response', 'Complain']
numerical_columns = df.select_dtypes(include=['int64',
'float64']).columns.difference(exclude_list)
#numerical_columns = df.select_dtypes(include=['int64', 'float64']) ----
this didnt work
# Careful with outlier check on binary columns. It will set them to zero
# Create a function for outlier detection and treatment
def detect_and_treat_outliers(column):
   """
   Detects outliers in a numerical column using IQR and removes or
winsorizes them.
```

```
    Args:
        column: The numerical column to analyze.
    Returns:
        The cleaned column with outliers removed or winsorized.
    """
    # Calculate IQR and quantiles
    q1 = column.quantile(0.25)
    q3 = column.quantile(0.75)
    iqr = q3 - q1
    # Identify potential outliers (values beyond 1.5 IQR from quartiles)
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    outliers = column[(column < lower_bound) | (column > upper_bound)]
    # 1. Winsorize outliers (replace with nearest non-outlier value)
    # Choosing winsorize with clip method as I want the shape of my data
frame unchanged with no nulls.
    # Another method we can use the matstats library the command is slick!
    #from scipy.stats import mstats
    #cleaned_column = mstats.winsorize(df[column_to_winsorize], limits=
(lower_limit, upper_limit))
    cleaned_column = column.clip(lower_bound, upper_bound,axis=0)
    return cleaned_column
clean_df=pd.DataFrame()
# Iterate through numerical columns and visualize distributions
#Basic plot for every column descriptive anaysis.
#Eyeballing the data is much better I feel and faster. Gimmmick
for column in numerical_columns:
    print(f"\nAnalyzing column: {column}")
    # Create box plot
    plt.figure(figsize=(8, 6))
    df[column].plot(kind='box')
    plt.title(f"Box Plot for {column}")
    plt.show()
    # Create histogram
    plt.figure(figsize=(8, 6))
    df[column].hist(bins=20)
    plt.title(f"Histogram for {column}")
    plt.show()
    # Detect and treat outliers (use the chosen method from the function)
    clean_df[column] = detect_and_treat_outliers(df[column])
    df[column] = detect_and_treat_outliers(df[column])
clean_df.info()
clean_df.describe()
df.describe()
# Save the cleaned DataFrame if needed
# cleaned_df.to_csv('cleaned_data.csv', index=False)
condition = clean_df['Income'] > 68655
df_sorted = clean_df[condition].sort_values(by='Income')
# Display the sorted DataFrame
print(df_sorted['Income'])
"""write my cleanedd df to new file clean_marketing.csv"""
clean_df.to_csv('/content/drive/My Drive/Dataset/clean_marketing.csv',
index=False)
# Now load the new csv in the df to keep the nomenclature same
df = pd.read_csv(file_path)
df.describe()
list(df)
"""#Ordinal Encoding and Heat Map
```

- Use ordinal encoding and one hot encoding according to different

```
types of categorical variables.
The educational data is ordinal Masters>Graduate>SSC_Plus
*   Country and Marital status are changed to one hot to remove bias to
encoding. This is stored in data frame data
"""
# Define the mapping for education levels (ordinal encoding)
#Education is ordinal we rank it SSC<Grd<Master
education_mapping = {
   "SSC_Plus": 1,
   "Graduate": 2,
   "Masters": 3
}
# Apply the mapping to the 'Education' column
df['Education_Encoded'] = df['Education'].replace(education_mapping)
# Encode categorical variables appropriately (ordinal vs. one-hot)
df['Education_Encoded'] = df['Education'].astype('category').cat.codes
# Encoding for non ordinal data
encoded_df = pd.get_dummies(df, columns=['Country', 'Marital_Status'])  #
Example for one-hot encoding
encoded_df.head()
dummy_columns = [col for col in encoded_df.columns if
col.startswith('Country_') or col.startswith('Marital_Status_')]
filtered_df = encoded_df[dummy_columns]
# Combine original and encoded DataFrames
df_with_encoded = pd.concat([clean_df, filtered_df], axis=1)
# Select numerical columns for correlation
numerical_columns = [col for col in df_with_encoded.columns if col not in
("Education",'Country','Marital_Status')]
print(numerical_columns)
# Reduce the numerical columns in the plot to managable. Cloose from
numerical columns
list_col=['Income', 'Recency',
          'NumWebPurchases',  'NumStorePurchases',
          'Complain',
         'Total_Children', 'Age', 'Total_Spending', 'Country_US',
'Marital_Status_Single', 'Marital_Status_Together']
# Calculate correlation matrix
correlation = df_with_encoded[list_col].corr()
# Create the heatmap using Seaborn for better control and colorbar
plt.figure(figsize=(15, 10))  # Set the figure size (adjust as needed)
sns.heatmap(correlation, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap with One-Hot Encoded Education")
plt.show()
# Visualize data distributions and correlations to assess normality,
outliers, and relationships
#
print(df.columns.tolist())
print(df.shape)
print(df.info())
"""#Hypothesis: Older ppl like stores
Null hypotheis: There is no evidence of association between age and store
purchase. Using the Chi square test. As sample size is large and the test is
fastest. May not be ideal as this is interval and ratio data.
"""
from scipy.stats import chi2_contingency
# Test hypothesis 1: Older people prefer in-store shopping
# Use appropriate tests like Chi-squared or Kruskal-Wallis depending on data
distributions
contingency_table = pd.crosstab(df['Age'], df['NumStorePurchases'])
chi2_stat, pval, exp_freq, obs_freq = chi2_contingency(contingency_table)
```

```
print("Chi-squared test statistic:", chi2_stat)
print("p-value:", pval)  # Interpret the p-value to draw conclusions
if pval<0.05:
   print('reject the null hypothesis, meaning theres /n significant evidence
of an association between age and in-store purchases')
else: print('we fail to reject the null hypothesis /n suggesting no
statistically significant association')
"""#Hypothesis 2 testing customer with kids
Which is the best method? Cji Square or logistic regression.
Null hyothesis: There is no difference between people without kids making
online purchase and people with kids making online purchase
Chi square code
"""
import pandas as pd
from scipy.stats import chi2_contingency
# Create binary variable for having kids (combining Kidhome and Teenhome if
needed)
has_kids = df["Kidhome"] + df["Teenhome"] >= 1
# Contingency table (cross-tabulation)
contingency_table = pd.crosstab(has_kids, df['NumWebPurchases'])
# Chi-square test
chi2, pval, a, b = chi2_contingency(contingency_table)
print("Chi-square test statistics:")
print(f"Chi-square: {chi2:.2f}")
print(f"p-value: {pval:.4f}")
# Interpretation
if pval < 0.05:
   print("There is a statistically significant association between having
kids and online shopping.")
else:
   print("There is no statistically significant association between having
kids and online shopping.")
"""Proving Hypothesis 2 with Logistic regression"""
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import OneHotEncoder
# Create binary variable for having kids (combining Kidhome and Teenhome if
needed)
df["has_kids"] = df["Kidhome"] + df["Teenhome"] >= 1
# Encode categorical variables (Education and Country) using OneHotEncoder
#encoder = OneHotEncoder(sparse=False, handle_unknown='ignore')
#encoded_df = pd.concat([df[["NumWebPurchases",  "NumStorePurchase",
"NumCatlogPurchase"]],has_kids,
#pd.DataFrame(encoder.fit_transform(df[["Education", "Country"]]),
columns=encoder.get_feature_names(["Education", "Country"]))], axis=1)
encoded_df=pd.concat([df_with_encoded,df['has_kids']],axis=1)
# Define features and target variable
features = ["has_kids", "NumStorePurchase", "NumCatlogPurchase"] +
list(encoded_df.columns)[6:]
target = "NumWebPurchases"
#list(encoded_df)
#encoded_df.head()
#encoded_df.iloc[:,0]
# Create and fit the model
model = LogisticRegression()
model.fit(encoded_df.iloc[:, [37,15,14]], encoded_df.iloc[:, 13])
#model.fit(encoded_df.loc['has_kids','NumCatalogPurchases'],
encoded_df.loc['NumWebPurchases'])
# Print coefficients and interpret
print("Logistic regression coefficients:")
for feature, coef in zip(features, model.coef_):
```

```
   print(f"{feature}: {coef[0]:.2f}")
# Interpretation
positive_coefs = []
negative_coefs = []
has_kids_coef = model.coef_[0,0]
print('The coeff of the model is:  ',has_kids_coef)
if has_kids_coef > 0:
   print("Customers with kids tend to have higher online purchases.")
elif has_kids_coef < 0:
   print("Customers with kids tend to have lower online purchases.")
else:
   print("Having kids has no significant impact on online purchases.")
"""# Hypothesis Distr Channels cannibalise sales at store
##Multiple comparison
- Perform three t-tests:
   - Store vs. Online/Catalog
   - Store vs. Both
   - Online/Catalog vs. Both
- Apply Bonferroni correction to adjust p-values for multiple comparisons
and control for Type I error.
- This method highlights specific groups with significant differences but
might be conservative.
"""
from scipy.stats import ttest_ind
# Define groups
store_only = df[df["NumWebPurchases"] + df["NumCatalogPurchases"] == 0]
#debug
online_catalog = df[df["NumWebPurchases"] + df["NumCatalogPurchases"] >= 1]
both = df.query("NumWebPurchases + NumCatalogPurchases >= 1 and
NumStorePurchases >= 1")
#both = df[(df["NumWebPurchases"] + df["NumCatalogPurchases"]) >= 1 &
df["NumStorePurchases"] >= 1]
# Test store sales between groups with Bonferroni correction
corrected_alpha = 0.05 / 3  # Adjust alpha for 3 comparisons
store_only_sales_mean = store_only["Total_Spending"].mean()
_, p1 = ttest_ind(store_only["Total_Spending"],
online_catalog["Total_Spending"], equal_var=False)
if p1 < corrected_alpha:
   print("Store sales differ between store-only and online/catalog groups
(p1", p1, ")")
_, p2 = ttest_ind(store_only["Total_Spending"], both["Total_Spending"],
equal_var=False)
if p2 < corrected_alpha:
   print("Store sales differ between store-only and both groups (p2", p2,
")")
_, p3 = ttest_ind(online_catalog["Total_Spending"], both["Total_Spending"],
equal_var=False)
if p3 < corrected_alpha:
   print("Store sales differ between online/catalog and both groups (p3",
p3, ")")
"""##Multinominal Logistic Regression
Model store purchases as the dependent variable with three categories (only
store, any online/catalog, both).
- Include control variables like income, age, or product category.
- Analyze coefficients to interpret how online/catalog purchases affect
store purchase probability while controlling for other factors.
- Provides direction and strength of effects but assumes categorical store
purchase variable.
"""
try:
```

```
 from sklearn.linear_model import MultinomialLogit
 print("MultinomialLogit is already available!")
except ImportError:
 print("MultinomialLogit not found. We'll install it for you.")
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import OneHotEncoder
# Encode channel types (dummy variables)
#encoder = OneHotEncoder(sparse=False, handle_unknown='ignore')
#encoded_df = pd.concat([df[["Total_Spending"]],
pd.DataFrame(encoder.fit_transform(df[["NumStorePurchase",
"NumWebPurchases", "NumCatlogPurchase"]]),
columns=encoder.get_feature_names(["Store", "Web", "Catalog"]))], axis=1)
# Define features and target variable
features = list(encoded_df.columns)[1:]
target = "NumStorePurchases"  # Convert store purchases to categorical
(e.g., high, medium, low)
# Create and fit the model
model = LogisticRegression()
model.fit(encoded_df[features], encoded_df[target])
# Interpret coefficients for online/catalog purchase effects on store
purchase categories
# Print coefficients and interpret
print("Logistic regression coefficients:")
for feature, coef in zip(features, model.coef_):
   print(f"{feature}: {coef[0]:.2f}")
"""#Hypothesis US Sales better
##Two Sample T test
1. Choosing the right test:
Since I have two groups (US and Rest of World) and a continuous variable
(total purchases), a simple two-sample t-test is appropriate.
2. Check normality with histogram. I have already done that in step 4 for
all variables.
"""
import pandas as pd
from scipy.stats import ttest_ind
# Filter US and Rest of World groups
us_data = df[df["Country"] == "US"]
rest_of_world_data = df[df["Country"] != "US"]
# Check assumptions (optional)
# ... (Perform normality and homogeneity tests if needed)
# Perform t-test
t_statistic, p_value = ttest_ind(us_data["Total_Spending"],
rest_of_world_data["Total_Spending"], equal_var=False)
# Interpretation
print(f"T-statistic: {t_statistic:.2f}")
print(f"p-value: {p_value:.4f}")
if p_value < 0.05:
   print("There is a statistically significant difference in total purchases
between US and Rest of World.")
   if t_statistic > 0:
       print("US has higher total purchases on average.")
   else:
       print("Rest of World has higher total purchases on average.")
else:
   print("There is no statistically significant difference in total
purchases between US and Rest of World.")
"""#**Visualisation**

o      Which products are performing the best, and which are performing the
least in terms of revenue?
```

```
This has to be a pie chart
"""
# Get the sum of the Products in a dic
data = {'values': [df['MntFishProducts'].sum(),
df['MntGoldProds'].sum(),df['MntFruits'].sum(),df['MntMeatProducts'].sum(),d
f['MntWines'].sum()],
       'labels': ['Fish', 'Gold','Fruit','Meat','Wine']}
plt.figure(figsize=(8, 8))  # Adjust figure size as needed
plt.pie(data['values'], autopct='%1.1f%%', startangle=140,
labels=data['labels'])
plt.title('Product performance')
plt.axis('equal')  # Equal aspect ratio ensures a circular pie chart
plt.show()
"""##Age Vs Last campaign Visualisation"""
df[df['AcceptedCmp5']==1].count()
df.columns
# Filter accepted and rejected customers for the last campaign
accepted_customers = df[df["AcceptedCmp5"] == 1]
rejected_customers = df[df["AcceptedCmp5"] == 0]
rejected_customers.info()
# Create separate scatter plots for accepted and rejected customers
plt.figure(figsize=(10, 6))
# Accepted customers
plt.scatter(accepted_customers["Age"], accepted_customers["Total_Spending"],
color='green', label='Accepted', alpha=0.7)
# Rejected customers
plt.scatter(rejected_customers["Age"], rejected_customers["Total_Spending"],
color='red', label='Rejected', alpha=0.7)
# Customize the plot
plt.xlabel("Age")
plt.ylabel("Total Spending")
plt.title("Last Campaign Acceptance Rate by Age and Spending")
plt.legend()
plt.grid(True)
plt.tight_layout()
# Additional visualizations (optional):
# 1. Boxplot:
plt.figure(figsize=(8, 6))
plt.boxplot([accepted_customers["Age"], rejected_customers["Age"]], labels=
["Accepted", "Rejected"], notch=True)
plt.xlabel("Campaign Acceptance")
plt.ylabel("Age")
plt.title("Age Distribution - Accepted vs. Rejected")
plt.show()
# 2. Density plot:
plt.figure(figsize=(8, 6))
sns.kdeplot(accepted_customers["Age"], label="Accepted", color='green')
sns.kdeplot(rejected_customers["Age"], label="Rejected", color='red')
plt.xlabel("Age")
plt.ylabel("Density")
plt.title("Age Density - Accepted vs. Rejected")
plt.legend()
plt.show()
"""Country Vs Last Campaign acceptance"""
# Filter customers who accepted the last campaign
accepted_customers = df[df["AcceptedCmp5"] == 1]
# Count accepted customers by country
country_counts = accepted_customers["Country"].value_counts().reset_index()
country_counts.columns = ["Country", "Count"]
# Visualization options:
```

```
# 1. Bar chart:
plt.figure(figsize=(10, 6))
plt.bar(country_counts["Country"], country_counts["Count"], color="blue")
plt.xlabel("Country")
plt.ylabel("Number of Accepted Customers")
plt.title("Countries with the Most Accepted Customers (Last Campaign)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
# 2. Pie chart:
plt.figure(figsize=(8, 8))
plt.pie(country_counts["Count"], labels=country_counts["Country"],
autopct="%1.1f%%", startangle=140)
plt.title("Top Countries with Accepted Customers (Share)")
plt.axis("equal")
plt.show()
"""##Visualizing Number of Children and Total Spends"""
# Define number of children variable (adjust if needed)
num_children = df["Kidhome"] + df["Teenhome"]
# Visualization options:
# 1. Scatter plot:
plt.figure(figsize=(10, 6))
plt.scatter(num_children, df["Total_Spending"], alpha=0.7)
plt.xlabel("Number of Children")
plt.ylabel("Total Spending")
plt.title("Total Spending by Number of Children")
plt.grid(True)
plt.tight_layout()
plt.show()
# 2. Boxplot:
plt.figure(figsize=(8, 6))
plt.boxplot(df["Total_Spending"], notch=True)
plt.xlabel("Number of Children")
plt.ylabel("Total Spending")
plt.title("Total Spending Distribution by Number of Children")
plt.show()
# 3. Heatmap (if you have multiple numeric spending variables):
if "MntWines" in df.columns:
   spending_columns = ["Total_Spending", "MntWines", "MntFruits",
"MntMeatProducts", "MntFishProducts", "MntSweetProducts", "MntGoldProds"]
   correlation_matrix = df[spending_columns].corr()
   plt.matshow(correlation_matrix, cmap="coolwarm")
   plt.xticks(range(len(spending_columns)), spending_columns, rotation=45)
   plt.yticks(range(len(spending_columns)), spending_columns)
   plt.colorbar()
   plt.title("Correlation Matrix (Spending Variables and Number of
Children)")
   plt.show()
"""visualize the educational background of customers who complained in your
data"""
df_grouped = df.groupby('Education')['Complain'].count().reset_index()
# Create the bar chart
plt.figure(figsize=(8, 6))
plt.bar(df_grouped['Education'], df_grouped['Complain'], color=['blue',
'green', 'orange'])
plt.xlabel('Education Level')
plt.ylabel('Number of Complaints')
plt.title('Complaints by Education Level')
plt.xticks(rotation=0)  # Rotate x-axis labels for better readability
# Customize the chart (optional)
```

```
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
# Show the chart
plt.show()
# Create the pie chart
plt.figure(figsize=(8, 8))
plt.pie(df_grouped['Complain'], labels=df_grouped['Education'],
autopct='%1.1f%%', startangle=140)
plt.title('Complaints by Education Level (Pie Chart)')
plt.axis('equal')  # Equal aspect ratio ensures a circular pie chart
# Customize the chart (optional)
plt.tight_layout()
# Show the chart
plt.show()
```