test

April 14, 2023

1 1. Problem 1: Evolution of Sales Volume

1.0.1 General notes for the exam:

You can access all the data via your usual import package (e.g PANDAS in python):

In Python:

```
import pandas as pd
df_product = pd.read_csv('product_data.csv')
```

In R:

```
df_product <- read.csv('product_data.csv')
```

All needed packages are available in the test: Non exhaustive list: pandas, numpy, sklearn, scipy, …

Do not hesitate to ***COMMENT*** on your code and explain your ideas.

Try and answer all questions fully. If running out of time, please note that questions 6, 8 and 9 deliver the most points in the scoring system.

Throughout this entire exam, your goal will be to help a grocery company to better use its marketing campaigns.

1.0.2 How to debug your code:

To see the result of any print statements, you should: 1. Choose the tab "Custom input" next to the "Test Results" tab. 2. Fill in the text box that appears with: 'BCG'. You do not have to put your code in this box.

1.0.3 Description of the data provided on Section 1:

You are provided two datasets containing data about: 1. customer_data: containing data about customers 2. orders_data: containing data about orders

You can access them with the following snippet:

```
customers_df = pd.read_csv('customer_data.csv')
orders_df = pd.read_csv('order_data.csv')
```

The table `customer_data` containing the following columns: - `customer_id`: id of the customer - `birth_date`: birth date of the customer - `acquisition_channel`: marketing channel through which the customer was acquired.

The table **orders_data** containing the following columns: - **order_id**: id of the order - **transaction_date**: date of the transaction - **product**: ordered product - **price**: price of the bought product in $ - **quantity**: quantity ordered of the product - **customer_id**: customer issuing the order

Before getting modeling, we would like to do some preliminary analysis to understand better the data we have.

[ ]:
```
# Import libraries
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import OrdinalEncoder
```

[ ]:
```
customers_df = pd.read_csv('customer_data.csv')
orders_df = pd.read_csv('order_data.csv')
orders_df.sample(frac = 0.2)
```

[ ]:
```
           transaction_date                          product  price  \
131086  2018-03-23 15:30:00       FLUTED ANTIQUE CANDLE HOLDER   1.63
246279  2016-07-01 11:33:00    CHERRY BLOSSOM  DECORATIVE FLASK   3.75
390670  2018-10-11 11:38:00         NOEL GARLAND PAINTED ZINC    0.39
488987  2018-11-22 11:28:00         BAKING SET SPACEBOY DESIGN   4.95
488630  2018-11-22 10:41:00   HOT WATER BOTTLE TEA AND SYMPATHY   9.13
…                         …                                 …      …
128571  2016-03-22 12:15:00                    RED   EGG  SPOON   0.12
28140   2018-12-13 12:59:00              HEART OF WICKER SMALL   1.65
246427  2017-07-01 12:33:00             MINI FUNKY DESIGN TAPES   0.85
60875   2016-01-17 17:54:00            METALIC LEAVES BAG CHARMS   2.46
111373  2017-03-07 10:22:00       CHILDS BREAKFAST SET SPACEBOY    9.95

        quantity                         orders_id  \
131086         1  ea371ad80ec9434c997b108b910b2ff0
246279         4  edbceb54cd0b4c5084badb59588ee41a
390670        24  943d76f5b2af4dd39d49ade6618adb82
488987         6  70de8d6044d545a7bdead690f8187f90
488630        54  78bb8e4e3f144c8f9e81730cf14a9507
…            …                                 …
128571        24  0f2daaa3caa1487d9726fd961d7bd6aa
28140          1  77488eb3eec24877b1af18c2e6ac19ac
246427         1  8300a2811d7f469a9af5a6b1423b3f71
60875          1  64aac215d914498eb05054c13099ae36
111373         2  c1307579a5b74cd78871f37b79da10ce

                             customer_id
131086  a44ae6c32327414892dd4533dfe6ce36
```

```
246279  932a65facad34b32a289eaee17a82c3b
390670  ad38ba7686024120b276b0b55c4f9871
488987  d0929e57a17546b2831b7947fad81ac3
488630  d3a39c3d89834f46952d9d462104ef3a
…                                    …
128571  948e3a7271c046549caca9269504f5ee
28140   b61a1cd362a749a2b4423f78884ba159
246427  a35f40416e814055b3a0170d57fe5dbb
60875   f3da694957184ff7a1cf194422be9a62
111373  51e1c830874f4aef8f95253844d121dd

[108382 rows x 6 columns]
```

[ ]:
```python
customers_df.head()
```

[ ]:
```
                        customer_id          birth_date acquisition_channel
0  213803050f1a4336b286e6781d4a7073   1964/12/14-8:14:50                 Web
1  77614bf0e41449d68586425edf550ef8  1964/10/23-19:32:14               Radio
2  c6c4fe164c09499db138fc9946b9d14b    1947/9/1-12:54:54            Billboard
3  29fd2cc211f941188d3254e4e9df378b    1987/7/24-3:11:23               Radio
4  db56add1ca8242cd90136337663cd6ec    1966/3/3-12:43:10                  TV
```

[ ]:
```python
orders_df.head()
```

[ ]:
```
      transaction_date                              product  price  quantity  \
0  2016-12-01 08:26:00   WHITE HANGING HEART T-LIGHT HOLDER   2.55         6
1  2016-12-01 08:26:00                  WHITE METAL LANTERN   3.39         6
2  2016-12-01 08:26:00       CREAM CUPID HEARTS COAT HANGER   2.75         8
3  2016-12-01 08:26:00  KNITTED UNION FLAG HOT WATER BOTTLE   3.39         6
4  2016-12-01 08:26:00       RED WOOLLY HOTTIE WHITE HEART.   3.39         6

                          orders_id                       customer_id
0  487421d8e2cc41ccb62ef3719b46510e  213803050f1a4336b286e6781d4a7073
1  c33967c39e594e78a76d99d25d0d6ff9  77614bf0e41449d68586425edf550ef8
2  32ef4d6ee91d4d49b5393dc9ec007cc6  c6c4fe164c09499db138fc9946b9d14b
3  1f3ecb8a0aeb4eed93527a8f1f09a471  c6c4fe164c09499db138fc9946b9d14b
4  78ba1e6269394163aae507389075acdd  29fd2cc211f941188d3254e4e9df378b
```

1.0.4 Problem 1:

What is the relative difference between the total sales on 2018 and 2017 ? The relative difference is defined as: (sales_2018/sales_2017 - 1) Sales are calculated as an ammount in $

[ ]:
```python
big_table = pd.merge(orders_df, customers_df, how="left", on="customer_id")
big_table['amount'] = big_table['price']*big_table['quantity']
big_table['transaction_date'] = pd.to_datetime(big_table['transaction_date'],
 ↪format='%Y/%m/%d %H:%M')
```

```
big_table['year'] = big_table['transaction_date'].dt.year
#big_table.head()
sales_by_year = big_table.groupby('year')['amount'].sum()
sales_2017 = sales_by_year.loc[2017]
sales_2018 = sales_by_year.loc[2018]
print("Sales in 2017: {}".format(sales_2017))
print("Sales in 2018: {}".format(sales_2018))
print("The relative difference between 2018 and 2017 is {}".format((sales_2018/
 ↪sales_2017)-1))
```

```
Sales in 2017: 3091888.722
Sales in 2018: 3397950.332
The relative difference between 2018 and 2017 is 0.09898855926549088
```

2 2. Problem 2: Population type per acquisition channel

2.0.1 Problem 2:

What is the median age per acquisition channel? Please return a pandas dataframe containing the following columns: - acquisition_channel: contains the channel name (radio, tv, …) - median_age: median age of the given channel

N.B: The median should be calculated on an integer 'age': convert the age to an integer before calculating the median.

[ ]:
```
big_table = pd.merge(orders_df, customers_df, how="left", on="customer_id")
big_table['birth_date'] = pd.to_datetime(big_table['birth_date'], format='%Y/%m/
 ↪%d-%H:%M:%S', errors='coerce')
big_table.dropna(inplace=True)
big_table['birth_year'] = big_table['birth_date'].dt.year
big_table['age'] = 2022 - big_table["birth_year"]
big_table["age"].head()

big_table.groupby('acquisition_channel')['age'].agg("mean")
```

[ ]:
```
acquisition_channel
Billboard    49.460909
Radio        49.619083
TV           49.607502
Web          49.540526
Name: age, dtype: float64
```

[ ]:
```
res = big_table.where(big_table.quantity <= 0)[['product', 'quantity']]
res.dropna(inplace=True)
res.head()
```

[ ]:
```
                                product  quantity
141                            Discount      -1.0
```

```
154   SET OF 3 COLOURED  FLYING DUCKS      -1.0
235    PLASTERS IN TIN CIRCUS PARADE      -12.0
236  PACK OF 12 PINK PAISLEY TISSUES      -24.0
237  PACK OF 12 BLUE PAISLEY TISSUES      -24.0
```

3 3. Problem 3: Popular product within millennials

3.0.1 Problem 3:

What is the most popular product (in terms of number of sold units) among the millennials (born between 1981 and 1996 incl.) ?

[ ]:
```
big_table.dropna(inplace=True)
t = big_table.where((big_table.birth_year>=1981) & (big_table.
 ↪birth_year<=1996))[["product", "quantity"]]
t.dropna(inplace=True)
# t.head(20)
t.groupby('product').agg("sum").sort_values(by=['quantity'], ascending=False).
 ↪head(5)
```

[ ]:
```
                                    quantity
product
WORLD WAR 2 GLIDERS ASSTD DESIGNS    14506.0
JUMBO BAG RED RETROSPOT              11714.0
PACK OF 72 RETROSPOT CAKE CASES       8814.0
WHITE HANGING HEART T-LIGHT HOLDER    8210.0
ASSORTED COLOUR BIRD ORNAMENT         7701.0
```

4 4. Linear regression (1/2)

4.1 Section 2: Linear Regression

We would like to understand which marketing channel is the most effective. For that, the marketing department of our client provided us with a dataset containing weekly spends on each channel and the revenue generated that week during the 3 last years.

The data in on a tabular format with the following columns: - week: a week identifier - spends_tv: spendings on tv marketing campaign that week - spends_radio: spendings on ads on the radio - spends_web: spendings on web ads - spends_billboard: spendings on phisical ads on billboards - revenue: revenue generated during this week

Your colleague had the idea of treating this problem as regression problem where he tries to estimate the revenue as linear function of spendings. He has performed a linear regression and obtained the following results:

5 OLS Results

Variable	Value
Dep. Variable	revenue
Model	OLS
Method	Least Squares
Date	Mon, 14 May 2018
Time	21:48:12
No. Observations	156
Df Model	3
Covariance type	nonrobust
R-squared	0.816
Adj. R-squared	0.712
F-statistic	6.646
Prob(F-statistic)	0.00157
Log-Likelihood	-12.974

==

	coef	str err	t	P>|t|
spends_tv	10454.7	197.2	53.02	<0.00001
spends_radio	5984.2	959.3	6.238	0.0041543
spends_web	8324.1	134.5	61.89_	<0.00001
spends_billboard	6278.5	434.1	14.46	<0.000359
const	30332.2	202.1	150.1	<0.00001

==

Variable	Value
Omnibus	0.176
Prob(Omnibus)	0.916
Skew	0.141
Kurtosis	2.786
Durbin-Watson	2.346
Jarque-Bera (JB)	0.167
Prob(JB)	0.920
Cond. No.	176.

==

Warnings: [1] Standard errors assume that the convariance matrix of the errors is correctly specified.

5.0.1 Problem 4:

Question 1:

With certainty, can you provide the MOST effective marketing channel?

Yes, it is TV. To arrive to this result we will define effectiveness of a channel as the revenue generated by unit money spent in this channel. And as we have a good R squared which is a good indicator of fit. So:

$$\frac{\text{Revenue net generated by channel}}{\text{Expenses on this channel}}$$

$$= \frac{\text{Revenue brut generated by channel} - \text{Expenses on this channel}}{\text{Expenses on this channel}}$$

$$= \frac{(\text{Expenses on this channel} * \text{coef of OLS}) - \text{Expenses on this channel}}{\text{Expenses on this channel}}$$

$$= \text{coef of OLS} - 1$$

So the variable that have the biggest coefficient is the most effective. In this case TV. N.B: We have a good p-value for spends_tv which is a good indicator for fiability for this variable.

5.0.2 Problem 5:

Question 2:

With certainty, can you provide the LEAST effective marketing channel? Pick **ONE** option - TV - Radio - Web - Billboard - Can't say

No, it could be the radio or billboard. Following the preceding idea. It should be radio the least effective, however we can see that standard error is high. It is the same with billboard, and that in practice may be billboard the least effective channel, but with our model we cannot guarantee the least effective.

6 6. Regression model to predict revenues

6.1 Section 2: Build a regression model

The marketing department with which we are working want to send personalised promotions to targeted customers. They need help from us to get the highest value customers: customers who will generate the most revenues.

For this, you are asked to create a **regression model** that predicts the demand for a given customer and year. The **target value** is **revenue** which is equal to the **price * quantity agregated at year level**

Important Note in this section:

Do not hesitate to write comments and to modularize your code. You will be evaluated both on the result and the quality of your code. If you get stuck in a question, do not hesitate to move on the next question. Points are assigned independently for each question.

Question 1:

Using the two tables from Question 1 (orders_data and customer_data), create an aggregated table of revenus by year and customer_id.

[ ]:
```
big_table = pd.merge(orders_df, customers_df, how="left", on="customer_id")
big_table['revenus'] = big_table['price']*big_table['quantity']
```

```
big_table['transaction_date'] = pd.to_datetime(big_table['transaction_date'],␣
↪format='%Y/%m/%d %H:%M')
big_table.head()
```

```
[ ]:     transaction_date                           product  price  quantity  \
0 2016-12-01 08:26:00   WHITE HANGING HEART T-LIGHT HOLDER   2.55         6
1 2016-12-01 08:26:00                  WHITE METAL LANTERN   3.39         6
2 2016-12-01 08:26:00       CREAM CUPID HEARTS COAT HANGER   2.75         8
3 2016-12-01 08:26:00  KNITTED UNION FLAG HOT WATER BOTTLE   3.39         6
4 2016-12-01 08:26:00       RED WOOLLY HOTTIE WHITE HEART.   3.39         6

                          orders_id                       customer_id  \
0  487421d8e2cc41ccb62ef3719b46510e  213803050f1a4336b286e6781d4a7073
1  c33967c39e594e78a76d99d25d0d6ff9  77614bf0e41449d68586425edf550ef8
2  32ef4d6ee91d4d49b5393dc9ec007cc6  c6c4fe164c09499db138fc9946b9d14b
3  1f3ecb8a0aeb4eed93527a8f1f09a471  c6c4fe164c09499db138fc9946b9d14b
4  78ba1e6269394163aae507389075acdd  29fd2cc211f941188d3254e4e9df378b

            birth_date acquisition_channel  revenus
0   1964/12/14-8:14:50                 Web    15.30
1  1964/10/23-19:32:14               Radio    20.34
2     1947/9/1-12:54:54           Billboard    22.00
3     1947/9/1-12:54:54           Billboard    20.34
4     1987/7/24-3:11:23               Radio    20.34
```

```
[ ]: # filtered_table = big_table[['transaction_date', 'customer_id', 'birth_date',␣
↪'price', 'quantity', 'acquisition_channel', 'revenus']]
filtered_table = big_table
```

```
[ ]: # We explicity wrote two forms to write an aggregate.
agg_revenues = \
    filtered_table.groupby(
        [filtered_table.transaction_date.dt.year, 'customer_id']
        )\
        .agg({"birth_date":'first',
                "price":'mean',
                "quantity":'sum',
                "acquisition_channel":'first',
                "revenus":lambda price: price.sum(),
                "orders_id": lambda orders: orders.nunique(),
                "product": lambda product: product.nunique(),
            }
        ).reset_index()

agg_revenues = agg_revenues.rename(columns={"price": "mean_price",␣
↪"transaction_date": "year_transaction", "orders_id": "total_orders",␣
↪"product": "purchased_products"})
```

```
agg_revenues.head()
```

```
[ ]:    year_transaction                       customer_id          birth_date  \
0              2016  00031990d7154c5d890d2dababf5225c     1974/8/8-9:10:18
1              2016  0003ce92d18d4fd995546c24728b44ff  1988/11/30-21:40:13
2              2016  0003fa288daa4e909add27ef3c219a27      1960/2/3-3:12:7
3              2016  000616d6d3fd435a9ca693f2f2d064fd     1995/2/2-7:45:11
4              2016  0006f73efa16466fa4cfac16dc26dcb9     1947/1/8-9:42:53

   mean_price  quantity acquisition_channel  revenus  total_orders  \
0    2.530000        28                  TV    48.10             4
1    1.650000         6                 Web     9.90             1
2    1.557500        80                  TV    32.33             4
3    3.200000         9                 Web    41.05             2
4   11.716667        27                  TV    80.85             3

   purchased_products
0                   4
1                   1
2                   4
3                   2
4                   3
```

Question 2: Create the following features on the aggregated table from question 1: - age: customer age - prev_year_revenue: revenue generated by the customer on the previous year - prev_year_nb_products: number of distinct products bought by the customer on the previous year

```
# Age
CURRENT_YEAR = 2022
agg_revenues['birth_date'] = pd.to_datetime(agg_revenues['birth_date'],␣
 ↪format='%Y/%m/%d-%H:%M:%S', errors='coerce')
agg_revenues.dropna(inplace=True)
agg_revenues['age'] = CURRENT_YEAR - agg_revenues["birth_date"].dt.year
agg_revenues.head()
```

```
[ ]:    year_transaction                       customer_id          birth_date  \
0              2016  00031990d7154c5d890d2dababf5225c 1974-08-08 09:10:18
1              2016  0003ce92d18d4fd995546c24728b44ff 1988-11-30 21:40:13
2              2016  0003fa288daa4e909add27ef3c219a27 1960-02-03 03:12:07
3              2016  000616d6d3fd435a9ca693f2f2d064fd 1995-02-02 07:45:11
4              2016  0006f73efa16466fa4cfac16dc26dcb9 1947-01-08 09:42:53

   mean_price  quantity acquisition_channel  revenus  total_orders  \
0    2.530000        28                  TV    48.10             4
1    1.650000         6                 Web     9.90             1
```

```
2    1.557500        80                 TV     32.33             4
3    3.200000         9                Web     41.05             2
4   11.716667        27                 TV     80.85             3

   purchased_products  age
0                   4   48
1                   1   34
2                   4   62
3                   2   27
4                   3   75
```

[ ]:
```
# previous year revenue
agg_revenues_index = agg_revenues.set_index(['year_transaction', 'customer_id'])
def get_revenue_year_before(row):
    try:
        revenue_year_before = agg_revenues_index.loc[(row['year_transaction'] -␣
 ↪1, row['customer_id']), 'revenus']
    except KeyError:
        revenue_year_before = None

    return revenue_year_before

agg_revenues['prev_year_revenue'] = agg_revenues.apply(lambda row:␣
 ↪get_revenue_year_before(row), axis=1)
```

[ ]:
```
agg_revenues_reduced = agg_revenues.dropna()
agg_revenues_reduced.head()
```

[ ]:
```
       year_transaction                       customer_id          birth_date  \
52732              2017  00031990d7154c5d890d2dababf5225c 1974-08-08 09:10:18
52733              2017  0003ce92d18d4fd995546c24728b44ff 1988-11-30 21:40:13
52734              2017  0003fa288daa4e909add27ef3c219a27 1960-02-03 03:12:07
52735              2017  000616d6d3fd435a9ca693f2f2d064fd 1995-02-02 07:45:11
52736              2017  0006f73efa16466fa4cfac16dc26dcb9 1947-01-08 09:42:53

       mean_price  quantity acquisition_channel  revenus  total_orders  \
52732       3.850         9                  TV    35.35             2
52733       2.045        26                 Web    49.06             4
52734       3.980        16                  TV    -8.56             4
52735       4.120        14                 Web    28.31             3
52736       3.105        20                  TV    54.68             2

       purchased_products  age  prev_year_revenue
52732                   2   48              48.10
52733                   4   34               9.90
52734                   4   62              32.33
52735                   3   27              41.05
```

```
52736                          2    75                 80.85
```

Question 3: Can you add some other features that may help the model? We expect you to add at least 2 features.

- Total number of purchased products.
- Total number of transactions.
- Average number of purchased products by order. = quantity / number of transactions
- Average spending by transaction = revenus / number of transactions.
- Time between two last purchased.
- Time between last purchased and current's day.

I've add two columns in the previous group by to obtain:

- Total number of purchased products by year is already in the table.
- Total number of transactions by year is already in the table.

Exactly I've added:

- "orders_id": lambda orders: orders.nunique(),
- "product": lambda product: product.nunique(),

[ ]:
```
# Average number of purchased products by order. = quantity / number of␣
↪transactions
agg_revenues_reduced["purchased_by_order"] = agg_revenues_reduced["quantity"] /␣
↪agg_revenues_reduced["total_orders"]
```

```
/home/frank/anaconda3/envs/BCG/lib/python3.7/site-
packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[ ]:
```
# Average spending by transaction = revenus / number of transactions.
agg_revenues_reduced["spending_by_order"] = agg_revenues_reduced["revenus"] /␣
↪agg_revenues_reduced["total_orders"]
agg_revenues_reduced.head()
```

```
/home/frank/anaconda3/envs/BCG/lib/python3.7/site-
packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[ ]:
```
       year_transaction                       customer_id          birth_date  \
52732              2017  00031990d7154c5d890d2dababf5225c 1974-08-08 09:10:18
52733              2017  0003ce92d18d4fd995546c24728b44ff 1988-11-30 21:40:13
52734              2017  0003fa288daa4e909add27ef3c219a27 1960-02-03 03:12:07
52735              2017  000616d6d3fd435a9ca693f2f2d064fd 1995-02-02 07:45:11
52736              2017  0006f73efa16466fa4cfac16dc26dcb9 1947-01-08 09:42:53

       mean_price  quantity acquisition_channel  revenus  total_orders  \
52732       3.850         9                  TV    35.35             2
52733       2.045        26                 Web    49.06             4
52734       3.980        16                  TV    -8.56             4
52735       4.120        14                 Web    28.31             3
52736       3.105        20                  TV    54.68             2

       purchased_products  age  prev_year_revenue  purchased_by_order  \
52732                   2   48              48.10            4.500000
52733                   4   34               9.90            6.500000
52734                   4   62              32.33            4.000000
52735                   3   27              41.05            4.666667
52736                   2   75              80.85           10.000000

       spending_by_order
52732          17.675000
52733          12.265000
52734          -2.140000
52735           9.436667
52736          27.340000
```

Question 4: We will split the data on a training and test sets. The test set should correspond to orders in the year 2018, and the other years are training set. Train your regression model on the training set and then use it to predict the outcome on the test set. Calculate the RMSE (Root Mean Square Error) on the test set and return it.

Models: - Linear regression - Ensemble methods: - Random forest - Neural Networks

[ ]:
```python
# Clean data

# Drop customer_id and birth_date
# Encode acquisition_channel

dataset = agg_revenues_reduced.drop(["customer_id", "birth_date"], axis=1)

enc = OrdinalEncoder()
dataset["acquisition_channel"] = enc.
 ↪fit_transform(dataset[["acquisition_channel"]])
dataset['acquisition_channel'] = dataset['acquisition_channel'].astype("int32")

dataset.head()
```

```
[ ]:        year_transaction  mean_price  quantity  acquisition_channel  revenus  \
52732               2017       3.850         9                    2    35.35
52733               2017       2.045        26                    3    49.06
52734               2017       3.980        16                    2    -8.56
52735               2017       4.120        14                    3    28.31
52736               2017       3.105        20                    2    54.68

       total_orders  purchased_products  age  prev_year_revenue  \
52732             2                   2   48              48.10
52733             4                   4   34               9.90
52734             4                   4   62              32.33
52735             3                   3   27              41.05
52736             2                   2   75              80.85

       purchased_by_order  spending_by_order
52732            4.500000          17.675000
52733            6.500000          12.265000
52734            4.000000          -2.140000
52735            4.666667           9.436667
52736           10.000000          27.340000
```

```
[ ]: dataset_train = dataset.loc[agg_revenues_reduced['year_transaction'] != 2018]
dataset_test = dataset.loc[agg_revenues_reduced['year_transaction'] == 2018]

X_train = dataset_train.drop(["revenus"], axis=1)
y_train = dataset_train['revenus']
X_test = dataset_test.drop(["revenus"], axis=1)
y_test = dataset_test['revenus']
```

7 Linear Regression

```
[ ]: LR = LinearRegression().fit(X_train, y_train)
y_pred = LR.predict(X_test)
rmse = mean_squared_error(y_test, y_pred)
print(f"The RMSE is :{rmse}")
r2 = r2_score(y_test, y_pred)
print(f"{r2*100:.2f}% of the predictions are explained by our data.")
```

```
The RMSE is :7153.247901956687
84.78% of the predictions are explained by our data.
```

8 Random Forest

[ ]:
```
RF = RandomForestRegressor(
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features = "auto",
    max_leaf_nodes = None,
    min_impurity_decrease = 0.0,
    bootstrap = True,
    oob_score = False,
    n_jobs = None,
    warm_start = False,
    ccp_alpha = 0.0,
    max_samples = None
).fit(X_train, y_train)
y_pred = RF.predict(X_test)
rmse = mean_squared_error(y_test, y_pred)
print(f"The RMSE is :{rmse}")
r2 = r2_score(y_test, y_pred)
print(f"{r2*100:.2f}% of the predictions are explained by our data.")
```

```
The RMSE is :4414.4720760097925
90.61% of the predictions are explained by our data.
```

9 MLP (Neural Networks)

[ ]:
```
# MLPRegressor
MLP = MLPRegressor(
    hidden_layer_sizes=(100,),
    activation="relu",
    solver="adam",
    alpha=0.0001,
    batch_size="auto",
    learning_rate="constant",
    learning_rate_init=0.001,
    power_t=0.5,
    max_iter=200,
    shuffle=True,
    tol=1e-4,
    warm_start=False,
    momentum=0.9,
    nesterovs_momentum=True,
    early_stopping=False,
    validation_fraction=0.1,
    beta_1=0.9,
```

```
    beta_2=0.999,
    epsilon=1e-8,
    n_iter_no_change=10,
    max_fun=15000
    ).fit(X_train, y_train)
y_pred = MLP.predict(X_test)
rmse = mean_squared_error(y_test, y_pred)
print(f"The RMSE is :{rmse}")
r2 = r2_score(y_test, y_pred)
print(f"{r2*100:.2f}% of the predictions are explained by our data.")
```

```
The RMSE is :8195.979999927045
82.56% of the predictions are explained by our data.
```

9.1 The best model is Random Forest

10 7. Data Assessment

10.1 Section 3: Build a regression model

Question:

On the provided data, we only have the quantity of sold products. What other important data is missing to estimate the real demand?

- Product segment
- Distribution Channel

11 8.

11.1 Section 4: Model interpretation

We would like now to focus on a particular and rare product: 'Chia seeds'.

This product represent 2% of the sales volume (in terms of quantity),

Your colleague has build a classification model to predict if a given customer will buy this product. The output of the algorithm is binary: - 1 when the model predicts that the customer will buy 'Chia seeds' - 0 when the model predicts that the customer will **NOT** buy 'Chia seeds'

The model your colleague made has a good accuracy: - For Chia seeds buyers, the model is correct 98% of the time. - For **NON** Chia seeds buyers, the model is correct 98% of the time.

You have run the model on a customer from your database, and the model predicted a positive answer meaning that he will buy chia seeds.

Question:

What is the probability of this customer to be a chia seeds buyer?

Pick **ONE** option: - 99% - 90% - 80% - 70% - **50%** <——— - 40% - 30% - 20% - 10% - 1%

Model:

Random variable X :

```
- 0: Consumer will not buy Chia.
- 1: Consumer will buy Chia.
```

Random variable Y :

```
- 0: Model predict that customer will not buy Chia.
- 1: Model predict that custumer will buy Chia.
```

Data from problem:

$$P(X = 0) = 0.98$$
$$P(X = 1) = 0.02$$
$$P(Y = 1|X = 1) = 0.98$$
$$\Rightarrow P(Y = 0|X = 1) = 0.02$$
$$P(Y = 0|X = 0) = 0.98$$
$$\Rightarrow P(Y = 1|X = 0) = 0.02$$

Question:

$$P(X = 1|Y = 1) = ?$$

Solution:

$$P(X = 1|Y = 1)$$
$$= \frac{P(X = 1, Y = 1)}{P(Y = 1)}$$
$$= \frac{P(Y = 1, X = 1)}{\sum_i P(Y = 1, X = i)}$$
$$= \frac{P(Y = 1, X = 1)}{P(Y = 1, X = 0) + P(Y = 1, X = 1)}$$
$$= \frac{1}{1 + \frac{P(Y = 1, X = 0)}{P(Y = 1, X = 1)}}$$
$$= \frac{1}{1 + \frac{P(Y = 1|X = 0)P(X = 0)}{P(Y = 1|X = 1)P(X = 1)}}$$
$$= \frac{1}{1 + \frac{0.02 * 0.98}{0.98 * 0.02}}$$
$$= \frac{1}{2}$$
$$= 0.5$$

12 9.

12.1 Section 5: Preprocessing Step

Imagine you have to develop a regression model. After gathering all the data you need on one table, and after you build your features, you ended up with a table having 7000 observations and 8000 features.

What is your next step? Can you provide 3 different techniques to do it? Can you also explain the main differences between them?

Please provide your answer in the following editor.

Dimensionality reduction (Projection Methods):

```
- PCA
    Maximize variance (eigenvalues)
- LDA
    Maximize separation between classes (maximize between-correlation)
```

Feature selection:

```
- Removing features with low variance
- Univariate feature selection
- Recursive feature elimination
- Select by correlation with target
```

Embedded Feature elimination for regression:

```
- L1-regularization (LASSO)
- LARS
- Random Forest
    Make trees with random columns then take the ones with better performance.
- Supervised principal components
    Same as PCA but with attention on labeled data
- Dantzig: https://hastie.su.domains/Papers/dantzig_annals.pdf
```