IT Coding, Tools and Security Project

March 19, 2021

1.1 Introduction to Data Analysis and Visualization

The complex enviroment in which the companies are operating nowadays, require to sense when and how change happens in order to be ready repond to that change, preventing threats and leveraging opportunities.

Data analysis and visualization are processes that allow to clean, transform and visualize large set of raw data,to better understand their meaning. Through these techniques it is possible to transform large and complex datasets in visive information, making them easy to understand even by who is not familiar with data analysis. Its main purose is to simplify the methods to identify tendencies, data analisys models and data anomalies when we are dealing with big data.

In order to do so, companies collect an enormous amount of data from a lot of differente sources, both internal and external, that need to be stored and processed to extract useful information. The processes for store, analyze and distribute these data require an adequate IT infrastructes with a good combination of hardware and software solutions. This architecture should be designed considering the company's business objectives and should have the following charatestics:

- **Scalability**: to address the problem of always growin data.
- **Parallel processing**: in order to let data to be accessed in multiple ways from multiple places and at multiple rates of speed.
- **Low-latency resources**: reducing the time to store, process and deliver data at the minimum level.
- **Data optimization**: an optimal data architecture enables data independence and provides a simple, resilient, and agile environment to support its analysis.

Therefore, a good IT architecture for Data Analysis and Visualization should have the right balance between performance, availability and access to data. Performance are usually limited by the costs, the availability should be based on a good resiliency to failures and access should meet the business organization requirements.

In terms of software and cloud applications, there are numerous solutions that visualize data allowing to extract information in very easy, fast and intuitable way. Some examples are Tebleau, SAS, Splunk, and QlikView, or open-source tools such as Apache Hadoop, Spark, and Hive. But in order to design applications that better fit business intelligence requirements, the use of an object-oriented programming language like Python is very common to streamline large complex dataset. Python offers multiple data analysis and visualization libraries that are rich of different features, allowing to create highly customizable and interactive representations that go beyond the standard charts. The most used ones are: Numpy, Pandas, Plotly, SeaBorn, ggPlot and Matplotlib.

The combination of Python libraries allows Data Analyst to develop instant reports, interactive and real-time dashboards to better present their studies to company's stakeholders.

The typical data analysis process is made by: Collection, Cleaning, Exploratory Analysis and Visualization, and the building and deployment of the model for more complex project (Data Sceince, Machine Learning, …)

Through this project we want to give a simplified example of how powerful and useful the processes of data analysis and visualization through Python are when we analyze a set of structured data.

1.2 Project overview

The dataset that we used come from StockX, a startup company that built a resell marketplace for buying and selling sneakers, watches and clothing. The company is born 5 years ago and is expanding at an incredible speed, now counting more than 800 employees. Since the primary market of Stockx are sneakers, our dataset is based on two of the most famous brands on that market of recent years, Yeezy and "Nike x Off-White". The structured data provided by the company includes information on:

- **Order Date**
- **Sneaker Brand and Model**
- **Price of retail**
- **Price of sale**
- **Release date**
- **Country**

All these information are related to the time span that starts the first September 2017 and ends the 2019, for a total of 16 months.

The project is diveded in three sections: Data Cleaning, Exploratory Data Analysis, Data Visualization.

1.3 Step 1: Data cleaning

First of all we made some data cleaning in order to have a dataset that is uniform on its wholeness and that can be better managed in order to facilitate the following work on it. Data Cleaning is a fundamental part of analysis and analytics, especially if we have to visualize data. A tidy dataset allows limit mistakes and make the charts, plots, and graphs closer to reality and with an higher aesthetic quality.

In this step, we will:

- Remove null rows with null values (unless the values can be predicted or approximated)
- Ensure that each columns describe a variable and each rows an elements or observation
- Make the values be the proper type, to make operations easier
- Delete misspellings or other related error that can be an obstacle for subsequent activities
- Sort values by date and by brand

[28]:
```python
#Invoking the modules needed and reading the dataset
import pandas as pd
import numpy as np
import seaborn as sns
import os
import datetime
import plotly.express as px
import json
from matplotlib import pyplot as plt
from urllib.request import urlopen
```

[29]:
```python
#navigate to the directory where the file is saved
os.chdir("/Users/deianrajovic/Desktop/Further Learning/Data Science/Dataset")
os.getcwd()
```

[29]:
```
'/Users/deianrajovic/Desktop/Further Learning/Data Science/Dataset'
```

[30]:
```python
#reading csv
df = pd.read_csv("stockx.csv")
df.head()
```

[30]:
```
                                          Order Date   Brand  \
0  9/1/17, Yeezy,Adidas-Yeezy-Boost-350-Low-V2-Be…     NaN
1                                             9/1/17   Yeezy
2                                             9/1/17   Yeezy
3  9/1/17, Yeezy,Adidas-Yeezy-Boost-350-V2-Core-B…     NaN
4                                             9/1/17   Yeezy

                                Sneaker Name Sale Price Retail Price  \
0                                        NaN        NaN          NaN
1    Adidas-Yeezy-Boost-350-V2-Core-Black-Copper       $685         $220
2     Adidas-Yeezy-Boost-350-V2-Core-Black-Green       $690         $220
3                                        NaN        NaN          NaN
4  Adidas-Yeezy-Boost-350-V2-Core-Black-Red-2017       $828         $220

  Release Date  Shoe Size  Buyer Region
0          NaN        NaN           NaN
1     11/23/16       11.0    California
2     11/23/16       11.0    California
3          NaN        NaN           NaN
4      2/11/17       11.0  Rhode Island
```

As you can see, the dataset at the current state cannot be used for the analysis. For these reason in the following cells we're going to clean it. Messy rows have NaN values in every columns but the "Order Date", in which all the observations are reported.

[31]:
```python
#creating a temporary pdf to rearrange the biased columns
null_df = df[df.Brand.isnull() == True]
df = df[df.Brand.isnull() == False]

null_df.head()
```

[31]:
```
                                           Order Date Brand Sneaker Name  \
0   9/1/17, Yeezy,Adidas-Yeezy-Boost-350-Low-V2-Be…   NaN          NaN
3   9/1/17, Yeezy,Adidas-Yeezy-Boost-350-V2-Core-B…   NaN          NaN
38  9/3/17, Yeezy,Adidas-Yeezy-Boost-350-Low-V2-Be…   NaN          NaN
39  9/3/17, Yeezy,Adidas-Yeezy-Boost-350-V2-Core-B…   NaN          NaN
73  9/5/17, Yeezy,Adidas-Yeezy-Boost-350-V2-Core-B…   NaN          NaN

   Sale Price Retail Price Release Date  Shoe Size Buyer Region
0         NaN          NaN          NaN        NaN          NaN
3         NaN          NaN          NaN        NaN          NaN
38        NaN          NaN          NaN        NaN          NaN
39        NaN          NaN          NaN        NaN          NaN
73        NaN          NaN          NaN        NaN          NaN
```

[32]:
```python
#spliting and renaming
order_date = null_df["Order Date"]
null_df = order_date.str.split(",", expand=True).rename(columns={
    0 :'Order Date', 1 : 'Brand', 2 : 'Sneaker Name', 3: 'SalePrice1', 4 :␣
↪'SalePrice2', 5 : 'Retail Price', 6 : 'Release Date',
    7 : 'Shoe Size', 8 : 'Buyer Region'})

#casting
null_df["Shoe Size"] = null_df["Shoe Size"].apply(pd.to_numeric)
```

[33]:
```python
#merging the two sale price columns
saleprice = null_df['SalePrice1'].astype(str) + null_df['SalePrice2'].
↪astype(str)
null_df.insert(3, 'Sale Price', saleprice)
null_df.drop(['SalePrice1','SalePrice2'],axis=1, inplace=True)
null_df['Sale Price'] = null_df['Sale Price'].str.replace('"','', regex=True)

null_df.head()
```

[33]:
```
   Order Date   Brand                         Sneaker Name Sale Price  \
0      9/1/17   Yeezy      Adidas-Yeezy-Boost-350-Low-V2-Beluga      $1097
3      9/1/17   Yeezy  Adidas-Yeezy-Boost-350-V2-Core-Black-Red      $1075
38     9/3/17   Yeezy      Adidas-Yeezy-Boost-350-Low-V2-Beluga      $1068
39     9/3/17   Yeezy  Adidas-Yeezy-Boost-350-V2-Core-Black-Red      $1095
73     9/5/17   Yeezy  Adidas-Yeezy-Boost-350-V2-Core-Black-Red      $1162

   Retail Price Release Date  Shoe Size Buyer Region
```

```
0           $220      9/24/16       11.0   California
3           $220     11/23/16       11.5     Kentucky
38          $220      9/24/16       10.0     Kentucky
39          $220     11/23/16       13.0   New Jersey
73          $220     11/23/16        5.0   California
```

[34]:
```
#merging the initial df with the reorganized null_df
df1 = df.append(null_df)
```

[35]:
```
# date columns
df1["Order Date"] = pd.to_datetime(df1["Order Date"])
df1["Release Date"] = pd.to_datetime(df1["Release Date"])

# clean or change the name of the brands
df1["Brand"].replace({"Off-White": "Nike x OW", " Yeezy": "Yeezy"},␣
 ↪inplace=True)

# sneaker Name column
df1["Sneaker Name"] = df1["Sneaker Name"].str.replace("-", " ")

# price columns
df1["Sale Price"] = df1["Sale Price"].str.replace("$", "", regex=False)
df1["Retail Price"] = df1["Retail Price"].str.replace("$", "", regex=False)

# casting the price columns to numeric in order to make mathematical and␣
 ↪statistical operations
df1[["Sale Price", "Retail Price"]] = df1[["Sale Price", "Retail Price"]].
 ↪apply(pd.to_numeric)
```

[36]:
```
df1.sort_values(["Brand", "Order Date"], ascending=True, inplace=True)
df1.head()
```

[36]:
```
    Order Date      Brand                 Sneaker Name  Sale Price  \
128 2017-09-07  Nike x OW    Nike Air Max 90 Off White        1600
129 2017-09-07  Nike x OW    Nike Air Max 90 Off White        1090
130 2017-09-07  Nike x OW    Nike Air Presto Off White        1344
131 2017-09-07  Nike x OW    Nike Air Presto Off White        1325
132 2017-09-07  Nike x OW  Nike Air VaporMax Off White        1800

     Retail Price Release Date  Shoe Size   Buyer Region
128           160   2017-09-09        8.0     California
129           160   2017-09-09       11.5       New York
130           160   2017-09-09       10.0       New York
131           160   2017-09-09       10.0  Massachusetts
132           250   2017-09-09       12.0       Kentucky
```

This is the cleaned dataframe in which we are going to work in the next steps.

1.4 Step 2: Exploratory data analysis

In this step, through the use of numpy and standard numpy functions, we are going to create some variables representing common statistic values.

Exploratory Data Analysis (EDA) is an approach/philosophy for data analysis that employs a variety of techniques (mostly graphical) to: 1. Maximize insight into a data set; 2. uncover underlying structure. 3. Extract important variables. 4. Detect outliers and anomalies. 5. Test underlying assumptions. 6. Develop parsimonious models. 7. Determine optimal factor settings.

The particular graphical techniques employed in EDA are often quite simple, consisting of various techniques of: 1. Plotting the raw data: such as data traces, histograms, bihistograms, probability plots, lag plots, block plots, and Youden plots. 2. Plotting simple statistics: such as mean plots, standard deviation plots, box plots, and main effects plots of the raw data.

[37]:
```
yeezy = df1[df1.Brand == "Yeezy"]
nike = df1[df1.Brand == "Nike x OW"]
```

It is better to create two different dataframe for each brand, in order to make some exploratory data analysis. We will go through every individual column and find out relevant summary statistics.

[38]:
```
# ORDER DATE

# Yeezy
first_day_yeezy = yeezy["Order Date"].min().date()
last_day_yeezy = yeezy["Order Date"].max().date()
first_day_yeezy = datetime.datetime.strptime(str(first_day_yeezy), "%Y-%m-%d").
 ↪strftime("%d/%m/%Y")
last_day_yeezy = datetime.datetime.strptime(str(last_day_yeezy), "%Y-%m-%d").
 ↪strftime("%d/%m/%Y")
print("The first day available in the yeezy sub-dataset is " +␣
 ↪str(first_day_yeezy) + ". The last day, is " + str(last_day_yeezy) + ".")

# Nike
first_day_nike = nike["Order Date"].min().date()
last_day_nike = nike["Order Date"].max().date()
first_day_nike = datetime.datetime.strptime(str(first_day_nike), "%Y-%m-%d").
 ↪strftime("%d/%m/%Y")
last_day_nike = datetime.datetime.strptime(str(last_day_nike), "%Y-%m-%d").
 ↪strftime("%d/%m/%Y")
print("The first day available in the nike sub-dataset is " +␣
 ↪str(first_day_nike) + ". The last day, is " + str(last_day_nike) + ".")
```

```
The first day available in the yeezy sub-dataset is 01/09/2017. The last day, is
13/02/2019.
The first day available in the nike sub-dataset is 07/09/2017. The last day, is
13/02/2019.
```

[39]:
```
# BRAND

# Count
yeezy_percentage = len(yeezy) / len(df1)
nike_percentage = len(nike) / len(df1)

print()
print("In the period under consideration, the total sales of Yeezy have been "␣
↪+ str(f'{len(yeezy):,}') + ", that is the " + f'{yeezy_percentage:.2%}' + "␣
↪of the total number of sales. ")

print("Instead, Nike x Off-White sneakers have reached a total of " +␣
↪str(f'{len(nike):,}') + " sales, " + "counting only for the " +␣
↪f"{nike_percentage:.2%}"+".")
print()

# Percentage

# counting yeezy model sales and find percentage over the total
yeezy_models = yeezy["Sneaker Name"].value_counts().to_frame().reset_index()
yeezy_models.rename(columns={"index": "Sneaker Model",'Sneaker Name':'Count'},␣
↪inplace=True)
yeezy_models["Percentage"] = yeezy_models["Count"] / len(yeezy)
yeezy_models["Percentage"] = pd.Series([f'{val:.2%}' for val in␣
↪yeezy_models["Percentage"]])

# counting nike model sales and find percentage over the total
nike_models = nike["Sneaker Name"].value_counts().to_frame().reset_index()
nike_models.rename(columns={"index": "Sneaker Model", 'Sneaker Name':'Count'},␣
↪inplace=True)
nike_models["Percentage"] = nike_models["Count"] / len(nike)
nike_models["Percentage"] = pd.Series([f'{val:.2%}' for val in␣
↪nike_models["Percentage"]])
```

```
In the period under consideration, the total sales of Yeezy have been 72,162,
that is the 72.19% of the total number of sales.
Instead, Nike x Off-White sneakers have reached a total of 27,794 sales,
counting only for the 27.81%.
```

[40]:
```
# SNEAKER NAME

# selecting iconic model and calculating number of sales and percentage on␣
↪total brand sales

# yeezy 350 (first gen.)
```

```python
yeezy_350 = yeezy.loc[yeezy['Sneaker Name'].str.contains('V2') == False]


yeezy_350_count = len(yeezy_350)
yeezy_350_percentage = yeezy_350_count / len(yeezy)


# yeezy 350 V2
yeezy_350v2 = yeezy.loc[yeezy['Sneaker Name'].str.contains('V2')]


yeezy_350v2_count = len(yeezy_350v2)
yeezy_350v2_percentage = yeezy_350v2_count / len(yeezy)

print("YEEZY:\n\n",
      "Yeezy 350 V2 have been sold",yeezy_350v2_count,"sales during the entire␣
 ↪period considered,",f"{yeezy_350v2_percentage:.2%}","of total brand sales.
 ↪\n",
      "The iconic model of the first generation, instead, counts only for",␣
 ↪yeezy_350_count, "sales.\n\n This enormous gap is probably given by the fact␣
 ↪that the price is too high (since the lower stock number).")

# jordan 1
nike_jordan1 = nike.loc[nike['Sneaker Name'].str.contains('Air Jordan 1')]

nike_jordan1_count = len(nike_jordan1)
nike_jordan1_percentage = nike_jordan1_count / len(nike)


# air force
nike_airforce1 = nike.loc[nike['Sneaker Name'].str.contains('Nike Air Force 1')]

nike_airforce1_count = len(nike_airforce1)
nike_airforce1_percentage = nike_airforce1_count / len(nike)
print()
print ("NIKE:\n\n"
      " The total of Jordan 1 x Off White sold is",nike_jordan1_count,"pairs,␣
 ↪which is",f"{nike_jordan1_percentage:.2%}","of the total.",
       "While Nike Air Force 1 x Off White is",f"{nike_airforce1_percentage:.
 ↪2%}"+".")
```

```
YEEZY:

 Yeezy 350 V2 have been sold 71707 sales during the entire period considered,
99.37% of total brand sales.
 The iconic model of the first generation, instead, counts only for 455 sales.
```

This enormous gap is probably given by the fact that the price is too high (since the lower stock number).

NIKE:

The total of Jordan 1 x Off White sold is 5703 pairs, which is 20.52% of the total. While Nike Air Force 1 x Off White is 8.94%.

[41]:

```
#SALE PRICE

#mean
yeezy_avg_saleprice = round(yeezy['Sale Price'].mean(), ndigits=2)
nike_avg_saleprice = round(nike['Sale Price'].mean(), ndigits=2)

#q1
yeezy_q1_saleprice = np.quantile(yeezy["Sale Price"], 0.25)
nike_q1_saleprice = np.quantile(nike["Sale Price"], 0.25)



#median
yeezy_med_saleprice = round(yeezy['Sale Price'].median(), ndigits=2)
nike_med_saleprice = round(nike['Sale Price'].median(), ndigits=2)

#q3
yeezy_q3_saleprice = np.quantile(yeezy["Sale Price"], 0.75)
nike_q3_saleprice = np.quantile(nike["Sale Price"], 0.75)

#IQR
yeezy_iqr_saleprice = yeezy_q3_saleprice - yeezy_q1_saleprice
nike_iqr_saleprice = nike_q3_saleprice - nike_q1_saleprice

#Max and min
yeezy_min_saleprice = round(yeezy['Sale Price'].min(), ndigits=2)
nike_min_saleprice = round(nike['Sale Price'].min(), ndigits=2)

yeezy_max_saleprice = round(yeezy['Sale Price'].max(), ndigits=2)
nike_max_saleprice = round(nike['Sale Price'].max(), ndigits=2)

#Range
yeezy_range_saleprice = yeezy_max_saleprice - yeezy_min_saleprice
nike_range_saleprice = nike_max_saleprice - nike_min_saleprice

#Variance
yeezy_var_saleprice = round(yeezy['Sale Price'].var(), ndigits=2)
nike_var_saleprice = round(nike['Sale Price'].var(), ndigits=2)

#Standard deviation
```

```
yeezy_std_saleprice = round(yeezy['Sale Price'].std(), ndigits=2)
nike_std_saleprice = round(nike['Sale Price'].std(), ndigits=2)


#representing them in a dataframe
print('Basic statistics on sale price data:\n')
describe_saleprice = pd.DataFrame([["Nike", nike_avg_saleprice,␣
 ↪nike_q1_saleprice, nike_med_saleprice, nike_q3_saleprice,␣
 ↪nike_iqr_saleprice, nike_min_saleprice, f'{nike_max_saleprice:,}',␣
 ↪f'{nike_range_saleprice:,}', f'{nike_var_saleprice:,}', nike_std_saleprice],
                                   ["Yeezy", yeezy_avg_saleprice,␣
 ↪yeezy_q1_saleprice, yeezy_med_saleprice, yeezy_q3_saleprice,␣
 ↪yeezy_iqr_saleprice, yeezy_min_saleprice, f'{yeezy_max_saleprice:,}',␣
 ↪f'{yeezy_range_saleprice:,}', f'{yeezy_var_saleprice:,}',␣
 ↪yeezy_std_saleprice]],
                                   columns=["Brand", "Mean", "Q1", "Median",␣
 ↪"Q3", "IQR", "Min", "Max", "Range", "Variance", "Standard Dev."])
describe_saleprice
```

```
Basic statistics on sale price data:
```

[41]:

	Brand	Mean	Q1	Median	Q3	IQR	Min	Max	Range	Variance	\
0	Nike	671.48	498.0	610.0	770.0	272.0	203	4,050	3,847	111,993.93	
1	Yeezy	360.03	268.0	316.0	399.0	131.0	186	2,300	2,114	20,658.93	

	Standard Dev.
0	334.65
1	143.73

[50]:

```
#SHOE SIZE

#mean
yeezy_avg_size = round(yeezy['Shoe Size'].mean(), ndigits=2)
nike_avg_size = round(nike['Shoe Size'].mean(), ndigits=2)

#q1
yeezy_q1_size = np.quantile(yeezy["Shoe Size"], 0.25)
nike_q1_size = np.quantile(nike["Shoe Size"], 0.25)


#median
yeezy_med_size = round(yeezy['Shoe Size'].median(), ndigits=2)
nike_med_size = round(nike['Shoe Size'].median(), ndigits=2)

#q3
yeezy_q3_size = np.quantile(yeezy["Shoe Size"], 0.75)
```

```
nike_q3_size = np.quantile(nike["Shoe Size"], 0.75)

#IQR
yeezy_iqr_size = yeezy_q3_size - yeezy_q1_size
nike_iqr_size = nike_q3_size - nike_q1_size

#Max and min
yeezy_min_size = round(yeezy['Shoe Size'].min(), ndigits=2)
nike_min_size = round(nike['Shoe Size'].min(), ndigits=2)

yeezy_max_size = round(yeezy['Shoe Size'].max(), ndigits=2)
nike_max_size = round(nike['Shoe Size'].max(), ndigits=2)

#Range
yeezy_range_size = yeezy_max_size - yeezy_min_size
nike_range_size = nike_max_size - nike_min_size

#Variance
yeezy_var_size = round(yeezy['Shoe Size'].var(), ndigits=2)
nike_var_size= round(nike['Shoe Size'].var(), ndigits=2)

#Standard deviation
yeezy_std_size = round(yeezy['Shoe Size'].std(), ndigits=2)
nike_std_size = round(nike['Shoe Size'].std(), ndigits=2)
print()
print('Basic statistics on shoe size distribution:\n')
describe_size = pd.DataFrame([["Nike", nike_q1_size, nike_med_size,␣
↪nike_q3_size, nike_min_size,nike_max_size],
                        ["Yeezy", yeezy_q1_size, yeezy_med_size, yeezy_q3_size,␣
↪yeezy_min_size,yeezy_max_size]],
                        columns=["Brand", "Q1", "Median", "Q3", "Min", "Max",])

describe_size
```

Basic statistics on shoe size distribution:

[50]:

	Brand	Q1	Median	Q3	Min	Max
0	Nike	8.5	10.0	11.0	3.5	17.0
1	Yeezy	8.0	9.5	11.0	3.5	17.0

[51]:

```
# BUYER REGION

yeezy_regions_count = yeezy['Buyer Region'].value_counts()
nike_regions_count = nike['Buyer Region'].value_counts()
total_regions_count = np.add(yeezy_regions_count, nike_regions_count)
```

```
regions = yeezy['Buyer Region'].unique()

buyer_regions = pd.DataFrame({'Yeezy' : yeezy_regions_count,'Nike x OW' :␣
↪nike_regions_count, 'Total orders' : total_regions_count},
                             index = regions)
print()
print('The following dataframe is showing the best 5 countries for number of␣
↪orders:\n')
buyer_regions.head(5)
```

```
The following dataframe is showing the best 5 countries for number of orders:
```

[51]:

	Yeezy	Nike x OW	Total orders
California	13113	6236	19349
Rhode Island	261	86	347
Michigan	2209	553	2762
New York	12103	4422	16525
Kansas	275	65	340

1.5 Step 3: Data Visualization

Data analysis and visualization are processes that allow to clean, transform and visualize large set of raw data,to better understand their meaning. Through these techniques it is possible to transform large and complex datasets in visive information, making them easy to understand even by who is not familiar with data analysis. Its main purose is to simplify the methods to identify tendencies, data analisys models and data anomalies when we are dealing with big data.

Using the data previusly cleaned and partially elaborated, we are going to process some visualization to better undestand for each brand the: - Best-selling models for each brand - Distrubution of sale price for each size of each Brand - Geographical distribution

The three following visualizations show which are the best models for each brand and what is the sale price distribution according to shoe size.

[44]:
```
sns.set_theme(context='notebook')
```

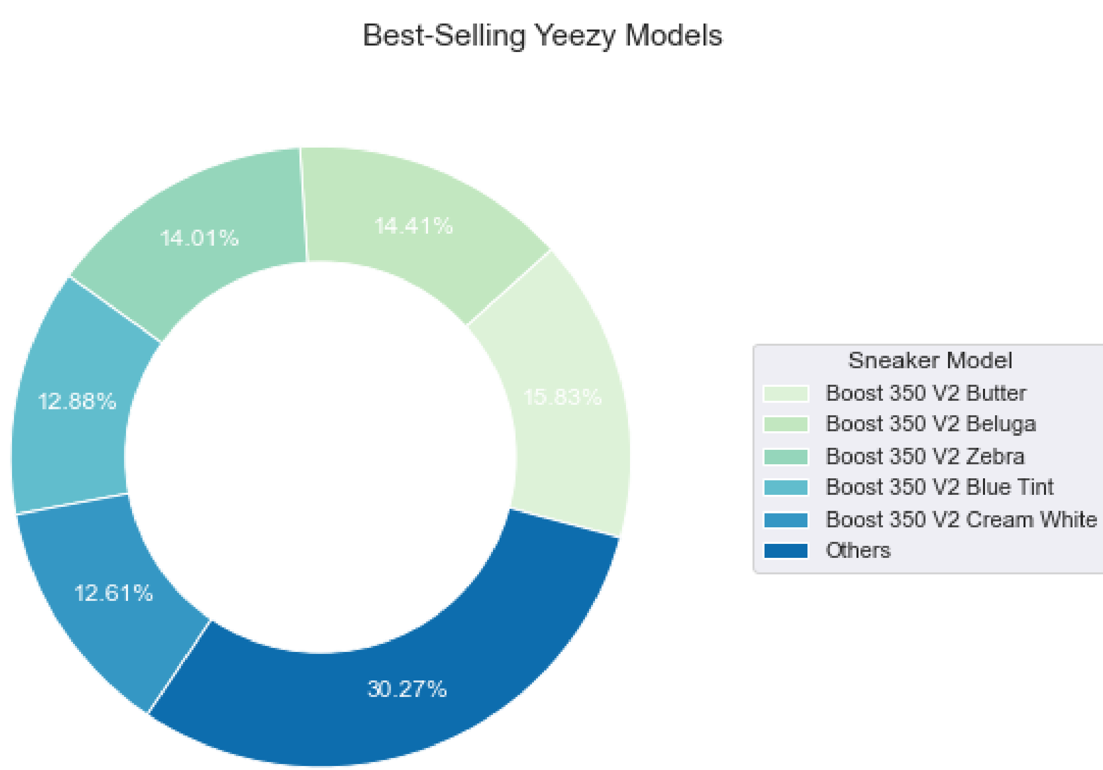
[45]:
```
# donut chart best-selling nike

#grouping top 5 Nike models and Nike others
nike_models_main = nike_models.loc[:4]
nike_models_main_count = list(nike_models_main['Count'].astype(int))
nike_models_others = nike_models.loc[5:]
nike_models_others_count = [nike_models_others['Count'].sum().astype(int)]
nike_models_pie = nike_models_main_count + nike_models_others_count

# chart
```

```
plt.pie(x=nike_models_pie,
        radius = 1.75,
        colors = ['#ddf2d8', '#c2e7c0', '#95d6bb', '#61bdcd', '#3597c4',␣
↪'#0d6dae'],
        autopct = '%0.2f%%',
        pctdistance=0.80,
        startangle = 345,
        shadow = False,
        textprops={'color':"#ffffff"}
        )

centre_circle = plt.Circle((0,0),1.1,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.axis='equal'
plt.title('Best-Selling Nike x OW Models', fontdict = {'fontsize': 15}, y=1.
↪4,x=1)
plt.legend(title = 'Sneaker Model',
          bbox_to_anchor=(1.45, 0,1, 1),
          loc = 'center left',
          labels = ['Jordan 1 Retro High UNC',
                    'Air Presto Black','Air Presto White','Vapormax',
                    'Blazer Mid Hallows Eve','Others']
           )
print()
plt.show()
print()
```

[46]:

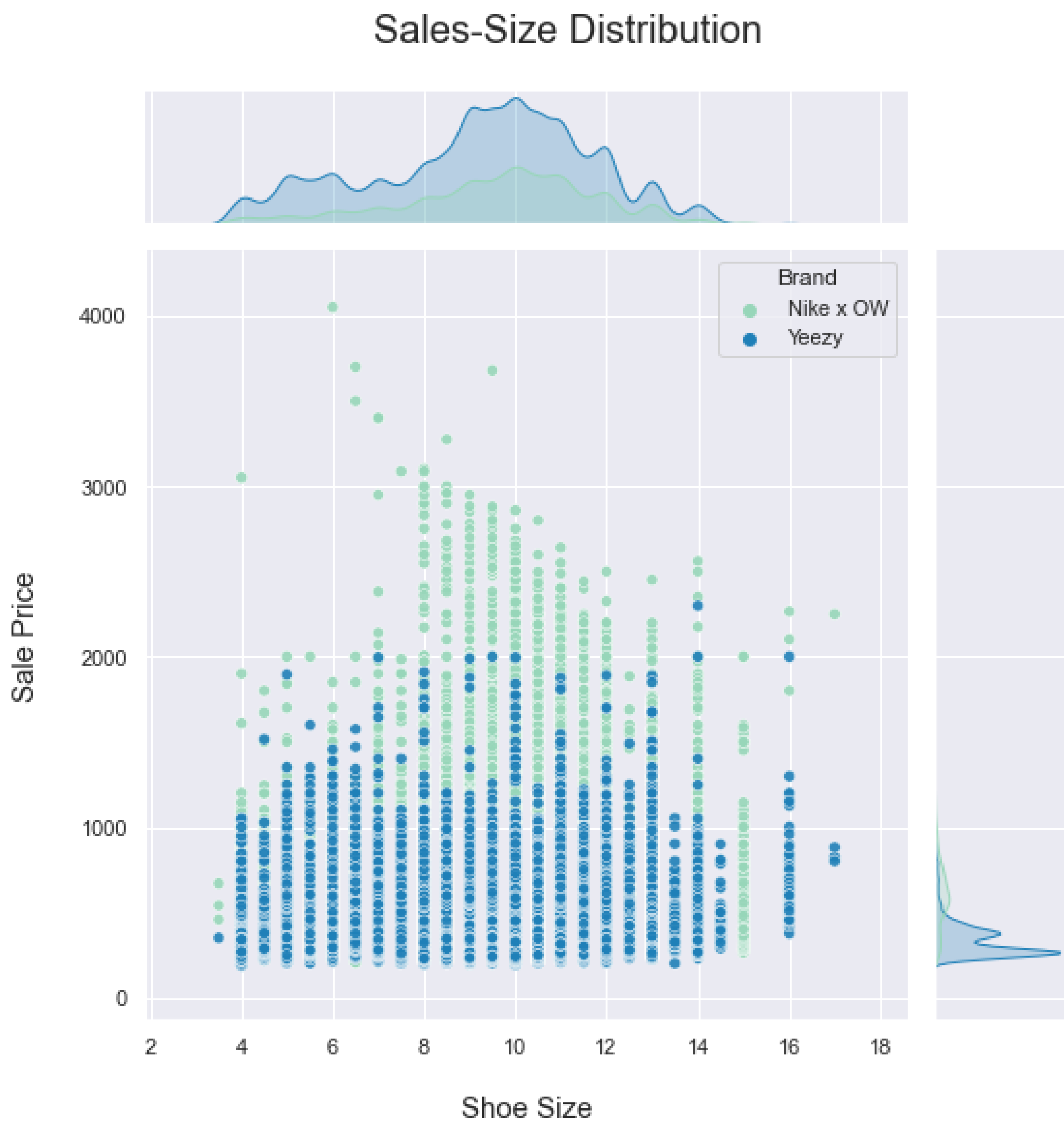
```
# donut chart best-selling yeezy

# grouping top 5 Yeezy models and Yeezy others
yeezy_models_main = yeezy_models.loc[:4]
yeezy_models_main_count = list(yeezy_models_main['Count'].astype(int))
yeezy_models_others = yeezy_models.loc[5:]
yeezy_models_others_count = [yeezy_models_others['Count'].sum().astype(int)]
yeezy_models_pie = yeezy_models_main_count + yeezy_models_others_count

# chart
plt.pie(x=yeezy_models_pie,
        radius = 1.75,
        colors = ['#ddf2d8', '#c2e7c0', '#95d6bb', '#61bdcd', '#3597c4',
 ↪'#0d6dae'],
        autopct = '%0.2f%%',
        pctdistance=0.80,
        startangle = 345,
        shadow = False,
        textprops={'color':"#ffffff"}
        )
```

```
centre_circle = plt.Circle((0,0),1.1,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.axis='equal'
plt.title('Best-Selling Yeezy Models', fontdict = {'fontsize': 15}, y=1.4,x=1)
plt.legend(title = 'Sneaker Model',
          bbox_to_anchor=(1.45, 0,1, 1),
          loc = 'center left',
          labels = ['Boost 350 V2 Butter','Boost 350 V2 Beluga',
                   'Boost 350 V2 Zebra','Boost 350 V2 Blue Tint',
                   'Boost 350 V2 Cream White','Others']
           )
print()
plt.show()
print()
```



[47]:

```python
# distribution of sales x size

joint = sns.jointplot(data=df1,
              y='Sale Price',
              x='Shoe Size',
              height=7.5,
              alpha=0.9,
              hue = 'Brand',
              palette = "YlGnBu"
              )

joint.set_axis_labels("Shoe Size", "Sale Price", fontsize=15, labelpad=20)
joint.fig.suptitle("Sales-Size Distribution", y=1.05, fontsize=20)



print()
plt.show(joint)
print()
```



The geographic map and heatmap models below show us that California, New York, Oregon and Florida are the states where most of the orders have been placed. We can also notice the enourmous gap between the numbers of those regions versus central regions like Montana, Wyoming, South/North Dakota.

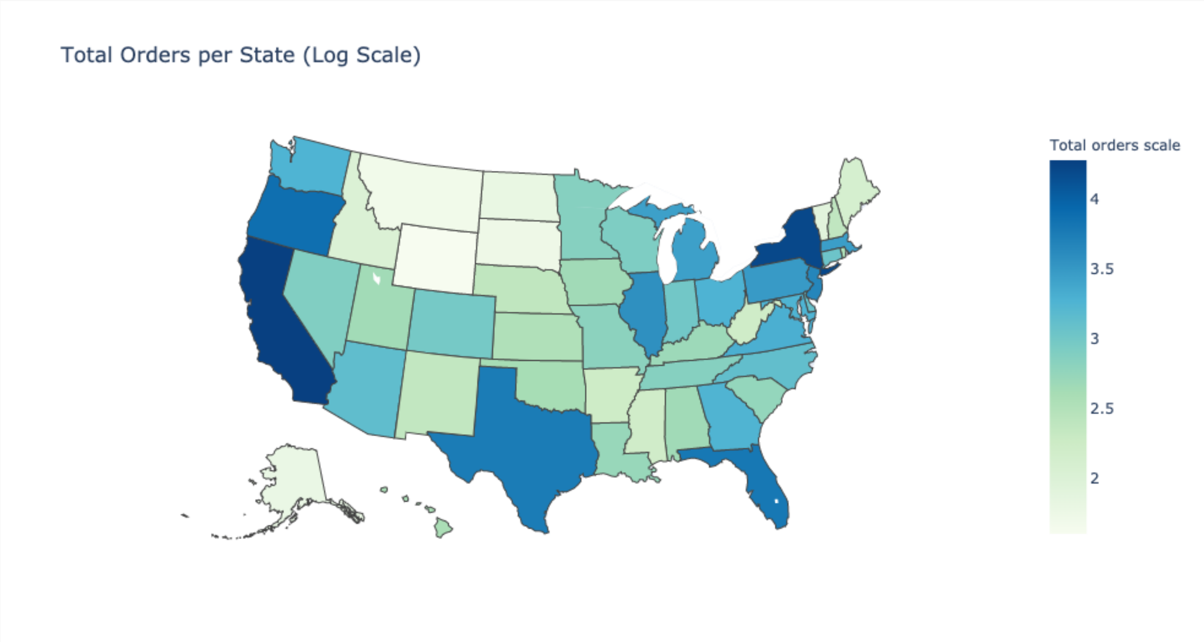
The reason could be related to multiple economical, social and demographic variables, for sure states like NY and CA are much more linked with the fashion industry and the effects can be seen even throught these maps

[52]:

```
#Map Total Sales

buyer_regions.reset_index(inplace=True)
buyer_regions.rename(columns={0 : 'State Name'})
buyer_regions.sort_values(by="Total orders")
buyer_regions

#importing json map template
states = json.load(urlopen('https://raw.githubusercontent.com/PublicaMundi/
↪MappingAPI/master/data/geojson/us-states.json'))

#giving id numb for each state
state_id_map = {}
for feature in states['features']:
  feature['id'] = feature['id']
  state_id_map[feature['properties']['name']] = feature['id']

buyer_regions['id'] = buyer_regions['index'].apply(lambda x: state_id_map[x])

#in order to get the right color scaling on the map
buyer_regions['Yeezy orders scale'] = np.log10(buyer_regions['Yeezy'])
buyer_regions['Total orders scale'] = np.log10(buyer_regions['Total orders'])

#setting the map
fig2 = px.choropleth(
      buyer_regions,
      locations = 'id',
      geojson=states,
      scope = 'usa',
      color='Total orders scale',
      hover_name = 'index',
      hover_data =['Yeezy','Nike x OW','Total orders'],
      color_continuous_scale = px.colors.sequential.GnBu,
      title = 'Total Orders per State (Log Scale)'
      )

#fig2.show()
## due to the pdf conversion we were not be able to make visible the map. We␣
↪attached the image of the output
#positioning the pointing over each state the map is going to show: Name of the␣
↪state, id numb, Yeezy Sales, Nike Sales, Total Sales, Total orders scale␣
↪number
```

[54]:

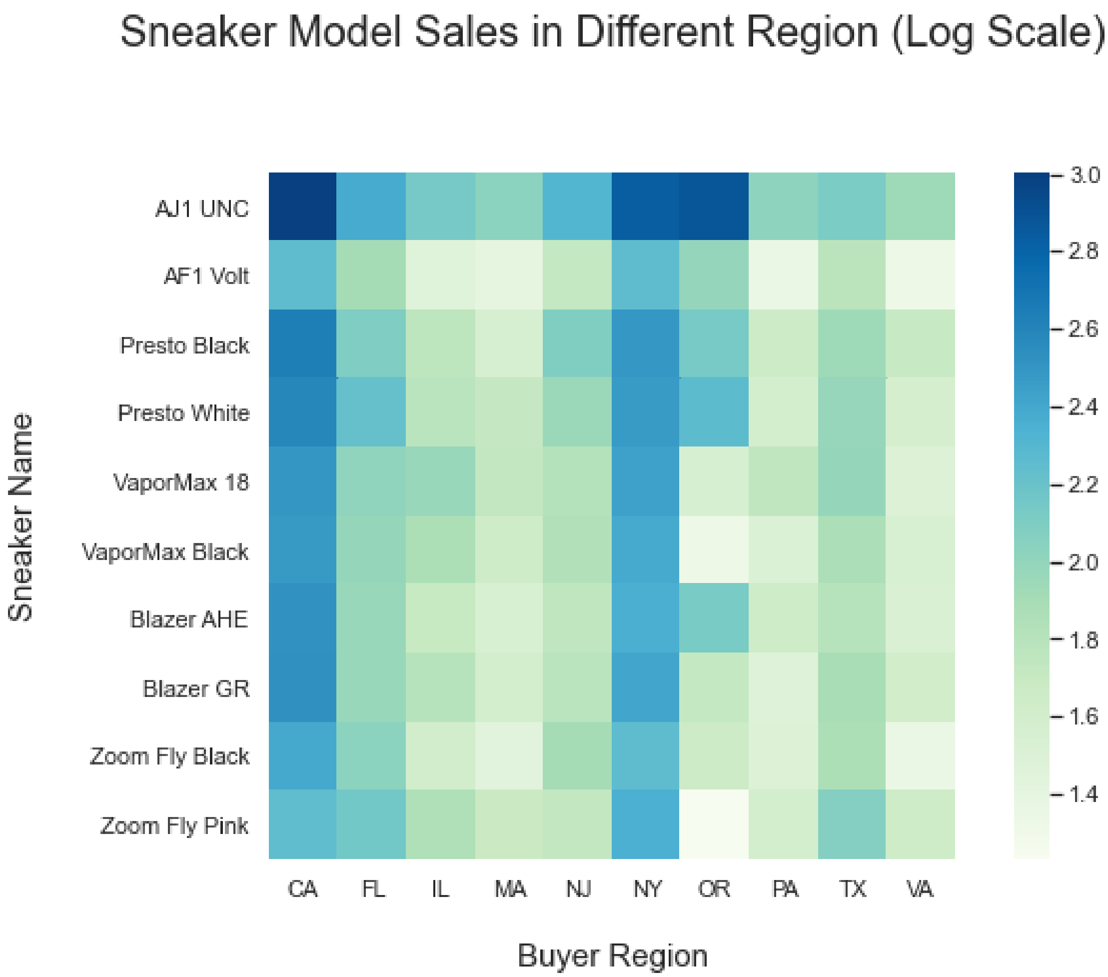
```
#
```



#

[49]:
```
regions = df1["Buyer Region"].unique()
regions.sort()

nike_top_10 = nike_models.loc[:9]

nike_top_10_regions = nike_regions_count.head(10)
nike_top_10_regions = pd.DataFrame({"Region": nike_top_10_regions.index,␣
↪"Count": nike_top_10_regions.values})
nike_top_10_regions.rename(columns={"Region": "Buyer Region"}, inplace=True)

temp = nike[(nike["Buyer Region"].isin(nike_top_10_regions["Buyer Region"]) ==␣
↪True) & (nike["Sneaker Name"].isin(nike_top_10["Sneaker Model"]) == True)]
temp = temp[["Sneaker Name", "Buyer Region"]]

temp = temp.groupby(by=["Sneaker Name", "Buyer Region"]).size().to_frame().
↪reset_index()
temp.rename(columns={0: "Count"}, inplace=True)

temp["Count_log"] = np.log10(temp["Count"])
```

```
pivot = pd.pivot_table(temp, index="Sneaker Name", columns="Buyer Region",␣
↪values="Count_log")

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

sns.heatmap(pivot,
                 cmap="GnBu",
                 square="equal",
                 xticklabels=["CA", "FL", "IL", "MA", "NJ", "NY", "OR", "PA",␣
↪"TX", "VA"],
                 yticklabels=["AJ1 UNC", "AF1 Volt", "Presto Black", "Presto␣
↪White", "VaporMax 18", "VaporMax Black", "Blazer AHE", "Blazer GR", "Zoom␣
↪Fly Black", "Zoom Fly Pink"],

                 )
plt.title("Sneaker Model Sales in Different Region (Log Scale)",␣
↪fontdict={"fontsize":20}, pad=60)
plt.xlabel("Buyer Region", fontdict={"fontsize":15}, labelpad=20)
plt.ylabel("Sneaker Name", fontdict={"fontsize":15}, labelpad=20)
print()
plt.show()
print()
```



Notes: some parts of step 2 and 3 have been reduced in order to meet the requirements. A complete work would require more intagrated visualization using more variables and models.