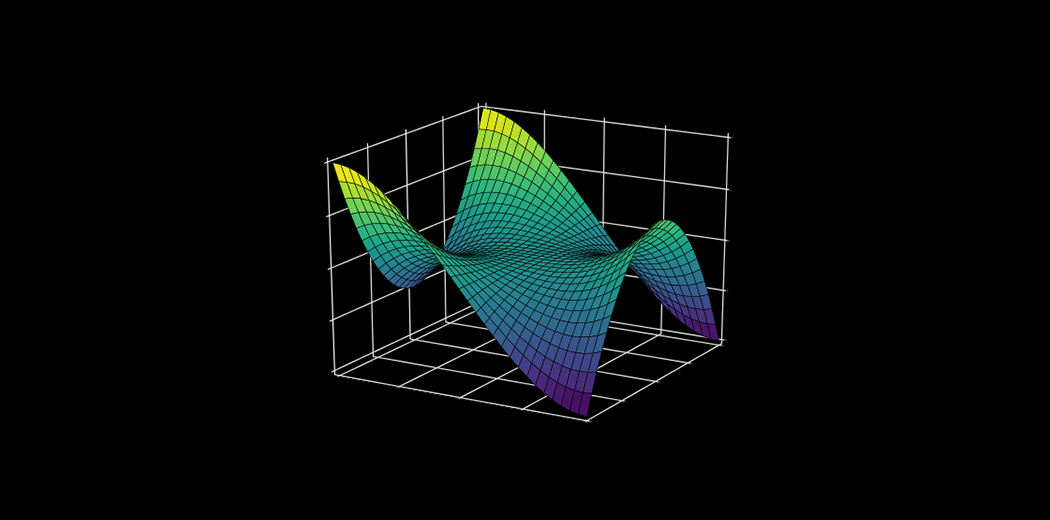
## matplotlib – The Most Popular Python Library for Data Visualization and Exploration

I love working with matplotlib in Python. It was the first [visualization](https://courses.analyticsvidhya.com/courses/tableau-2-0) library I learned to master and it has stayed with me ever since. There is a reason why matplotlib is the most popular Python library for data visualization and exploration – the flexibility and agility it offers is unparalleled!

Matplotlib provides an easy but comprehensive visual approach to present our findings. There are a number of visualizations we can choose from to present our results, as we’ll soon see in this tutorial.

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/matplot.png)

From histograms to scatterplots, matplotlib lays down an array of colors, themes, palettes, and other options to customize and personalize our plots. matplotlib is useful whether you’re performing data exploration for a machine learning project or simply want to create dazzling and eye-catching charts.

Note: If you’re new to the world of Python, we highly recommend taking the below popular free courses:

* [*Python for Data Science*](https://courses.analyticsvidhya.com/courses/introduction-to-data-science?utm_source=blog&utm_medium=beginner-guide-matplotlib-data-visualization-exploration-python)
* [*Pandas for Data Analysis in Python*](https://courses.analyticsvidhya.com/courses/pandas-for-data-analysis-in-python?utm_source=blog&utm_medium=beginner-guide-matplotlib-data-visualization-exploration-python)

## What is matplotlib?

Let’s put a formal definition to matplotlib before we dive into the crux of the article. If this is the first time you’ve heard of matplotlib, here’s the official description:

“Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.”

You can draw up all sorts of charts and visualization using matplotlib. I will be exploring the most common plots in the matplotlib Python library in this tutorial. We will first understand the dataset at hand and then start building different plots using matplotlib, including scatterplots and line charts!

Note: If you’re looking for a matplotlib alternative or want to explore other Python visualization libraries, check out the below tutorial on Seaborn:

* [*Become a Data Visualization Whiz with this Comprehensive Guide to Seaborn in Python*](https://www.analyticsvidhya.com/blog/2019/09/comprehensive-data-visualization-guide-seaborn-python/?utm_source=blog&utm_medium=beginner-guide-matplotlib-data-visualization-exploration-python)

## Here are the Visualization We’ll Design using matplotlib

* Bar Graph
* Pie Chart
* Box Plot
* Histogram
* Line Chart and Subplots
* Scatter Plot

## Understanding the Dataset and the Problem Statement

Before we get into the different visualizations and chart types, I want to spend a few minutes understanding the data. This is a critical part of the machine learning pipeline and we should pay full attention to it.

We will be analyzing the [Food Demand Forecasting](https://datahack.analyticsvidhya.com/contest/genpact-machine-learning-hackathon-1/?utm_source=blog&utm_medium=beginner-guide-matplotlib-data-visualization-exploration-python) project in this matplotlib tutorial. The aim of this project is to predict the number of food orders that customers will place in the upcoming weeks with the company. We will, of course, only spend time on the exploration stage of the project.

Let us first import the relevant libraries:

|  |  |
| --- | --- |
|  | import pandas as pd |
|  | import numpy as np |
|  | import matplotlib.pyplot as plt |
|  | plt.style.use('seaborn') |

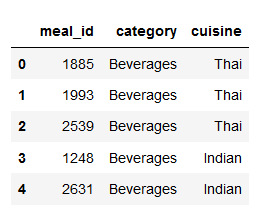
[**view raw**](https://gist.github.com/aniruddha27/f8952ac5a7a645cf2783ef1a7174a37a/raw/75a000988cc552fc6f4042bafa7f577d0c11b24a/Matplotlib_1.py)[**Matplotlib\_1.py**](https://gist.github.com/aniruddha27/f8952ac5a7a645cf2783ef1a7174a37a#file-matplotlib_1-py)hosted with  by [**GitHub**](https://github.com/)

I have used a matplotlib stylesheet to make our plots look neat and pretty. Here, I have used the ***‘seaborn’***stylesheet. However, there are plenty of other stylesheets in Matplotlib which you can use to best suit your presentation style.

Our dataset has three dataframes: ***df\_meal***describing the meals, ***df\_center*** describing the food centers, and ***df\_food***describing the overall food order. Have a look at them below:

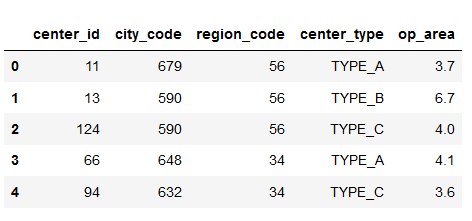
|  |  |
| --- | --- |
|  |  |
|  |  |

[**view raw**](https://gist.github.com/aniruddha27/4f7e36493a65d137f5e51324bbb2865b/raw/6950b756514285322f7581ce4a40516d9bbb718d/Matplotlib_2.py)[**Matplotlib\_2.py**](https://gist.github.com/aniruddha27/4f7e36493a65d137f5e51324bbb2865b#file-matplotlib_2-py)hosted with  by [**GitHub**](https://github.com/)



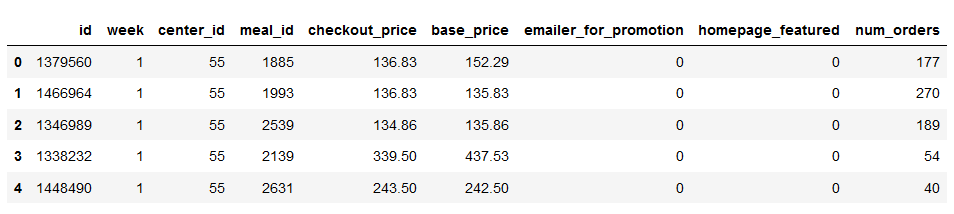
|  |  |
| --- | --- |
|  | df\_center = pd.read\_csv('C:\\Users\Dell\\Desktop\\train\_food\\fulfilment\_center\_info.csv') |
|  | df\_center.head() |

[**view raw**](https://gist.github.com/aniruddha27/810b6b9fd5a86bad3f1395023ba6df08/raw/9a27d58a3474303a2369622803ab1b98f8ef6436/Matplotlib_3.py)[**Matplotlib\_3.py**](https://gist.github.com/aniruddha27/810b6b9fd5a86bad3f1395023ba6df08#file-matplotlib_3-py)hosted with  by [**GitHub**](https://github.com/)

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/matplotlib_plotting_2.png)

|  |  |
| --- | --- |
|  | df\_food = pd.read\_csv('C:\\Users\Dell\\Desktop\\train\_food\\train\_food.csv') |
|  | df\_food.head() |

[**view raw**](https://gist.github.com/aniruddha27/cfe4a8a1b4ef3ba3814ec62e83910308/raw/2b9d8877030c8e0844450651044b698f46f0c68b/Matplotlib_4.py)[**Matplotlib\_4.py**](https://gist.github.com/aniruddha27/cfe4a8a1b4ef3ba3814ec62e83910308#file-matplotlib_4-py)hosted with  by [**GitHub**](https://github.com/)

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/matplotlib_plotting_3.png)

I will first merge all the three dataframes into a single dataframe. This will make it easier to manipulate the data while plotting it:

|  |  |
| --- | --- |
|  | df = pd.merge(df\_food,df\_center,on='center\_id') |
|  | df = pd.merge(df,df\_meal,on='meal\_id') |

[**view raw**](https://gist.github.com/aniruddha27/13b8e067b792d1d37dc12526ee3f6233/raw/bd20699dcccacbcfbb3e426292971575125c62e6/Matplotlib_5.py)[**Matplotlib\_5.py**](https://gist.github.com/aniruddha27/13b8e067b792d1d37dc12526ee3f6233#file-matplotlib_5-py)hosted with  by [**GitHub**](https://github.com/)

Right – now let’s jump into the different chart types we can create using matplotlib in Python!

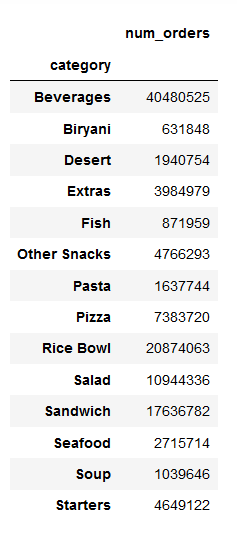
## 1. Bar Graph using matplotlib

First, we want to find the most popular food item that customers have bought from the company.

I will be using the Pandas ***pivot\_table***function to find the total number of orders for each category of the food item:

|  |  |
| --- | --- |
|  | table = pd.pivot\_table(data=df,index='category',values='num\_orders',aggfunc=np.sum) |
|  | table |

[**view raw**](https://gist.github.com/aniruddha27/f2a008305e5ba2df362bf1b4ba03ee9b/raw/e32451d0a6ffd400a551f1afa760c0f021f70e0a/Matplotlib_6.py)[**Matplotlib\_6.py**](https://gist.github.com/aniruddha27/f2a008305e5ba2df362bf1b4ba03ee9b#file-matplotlib_6-py)hosted with  by [**GitHub**](https://github.com/)

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/matplotlib_plotting_5.png)

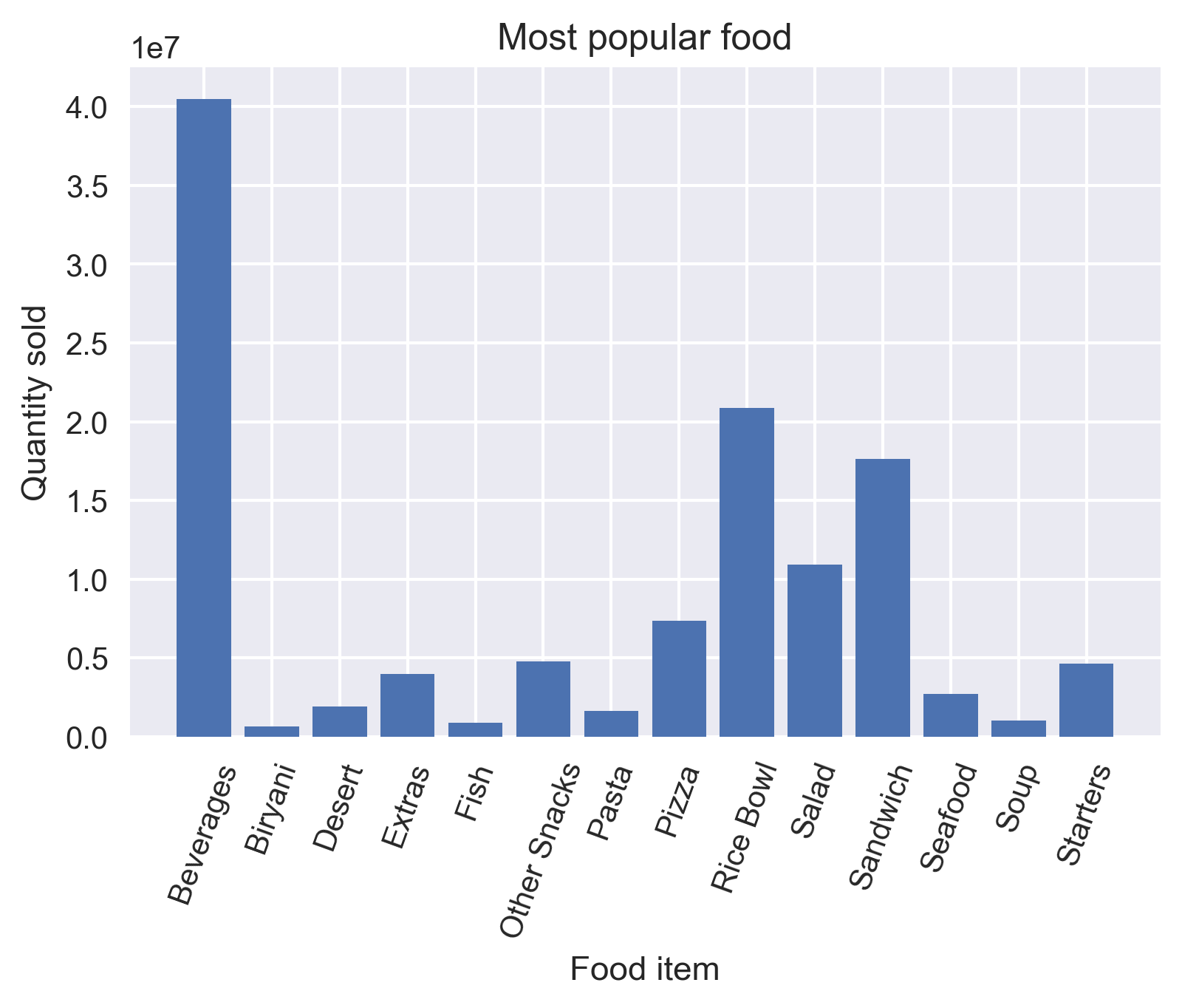
Next, I will try to visualize this using a bar graph.

Bar graphs are best used when we need to compare the quantity of categorical values within the same category.

***Bar graph*** is generated using ***plt.bar()*** in matplotlib:

|  |  |
| --- | --- |
|  | #bar graph |
|  | plt.bar(table.index,table['num\_orders']) |
|  |  |
|  | #xticks |
|  | plt.xticks(rotation=70) |
|  |  |
|  | #x-axis labels |
|  | plt.xlabel('Food item') |
|  |  |
|  | #y-axis labels |
|  | plt.ylabel('Quantity sold') |
|  |  |
|  | #plot title |
|  | plt.title('Most popular food') |
|  |  |
|  | #save plot |
|  | plt.savefig('C:\\Users\\Dell\\Desktop\\AV Plotting images\\matplotlib\_plotting\_6.png',dpi=300,bbox\_inches='tight') |
|  |  |
|  | #display |
|  | plot plt.show(); |

[**view raw**](https://gist.github.com/aniruddha27/a9a1f1ba492c9e77ee0f0fb8d646ad08/raw/e9fea65c753149eaebfb5428ce7b93666a1f6c50/Matplotlib_7.py)[**Matplotlib\_7.py**](https://gist.github.com/aniruddha27/a9a1f1ba492c9e77ee0f0fb8d646ad08#file-matplotlib_7-py)hosted with  by [**GitHub**](https://github.com/)

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/matplotlib_plotting_6.png)

It is always important to label your axis. You can do this by employing the ***plt.xlabel()*** and ***plt.ylabel()*** functions. You can use ***plt.title()*** for naming the title of the plot. If your xticks are overlapping, rotate them using the rotate parameter in ***plt.xticks()***so that they are easy to view for the audience.

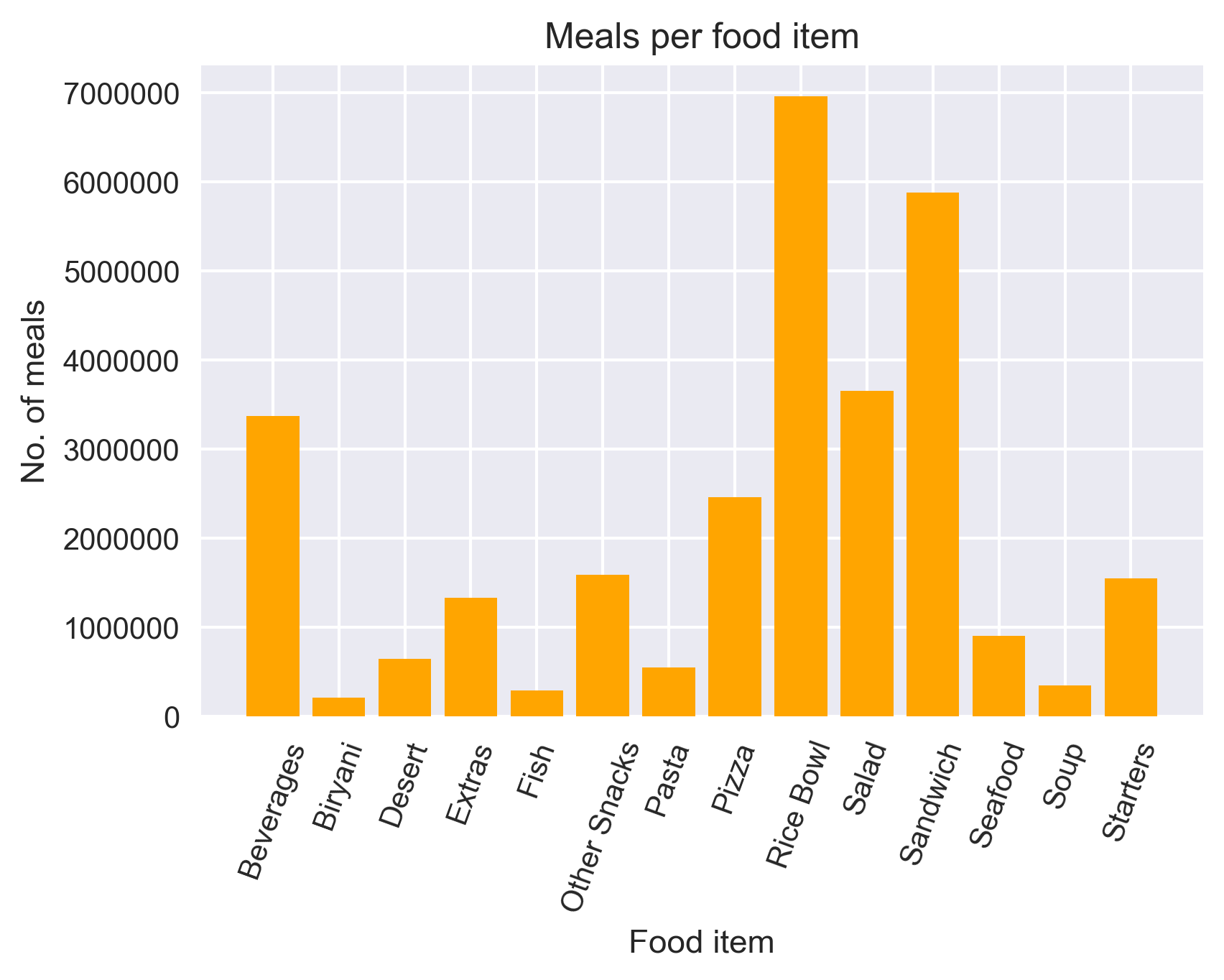
You can save your plot using the ***plt.savefig()***function by providing the file path as a parameter. Finally, always display your plot using ***plt.show()***.

While analyzing the plot, we can see that ***Beverages***were the most popular food item sold by the company. Wait, was it because they were sold with almost all the meals? Was ***Rice Bowl*** the most popular food item?

Let’s divide the total food item order by the number of unique meals it is present in.

|  |  |
| --- | --- |
|  | #dictionary for meals per food item |
|  | item\_count = {} |
|  |  |
|  | for i in range(table.index.nunique()): |
|  | item\_count[table.index[i]] = table.num\_orders[i]/df\_meal[df\_meal['category']==table.index[i]].shape[0] |
|  |  |
|  | #bar plot |
|  | plt.bar([x for x in item\_count.keys()],[x for x in item\_count.values()],color='orange') |
|  |  |
|  | #adjust xticks |
|  | plt.xticks(rotation=70) |
|  |  |
|  | #label x-axis |
|  | plt.xlabel('Food item') |
|  |  |
|  | #label y-axis |
|  | plt.ylabel('No. of meals') |
|  |  |
|  | #label the plot |
|  | plt.title('Meals per food item') |
|  |  |
|  | #save plot |
|  | plt.savefig('C:\\Users\\Dell\\Desktop\\AV Plotting images\\matplotlib\_plotting\_7.png',dpi=300,bbox\_inches='tight') |
|  |  |
|  | #display plot |
|  | plt.show(); |

[**view raw**](https://gist.github.com/aniruddha27/6c18b332a0873579340bc65f2002e200/raw/6f42f4852481418da0e6f6b62568cf576e446bec/Matplotlib_8.py)[**Matplotlib\_8.py**](https://gist.github.com/aniruddha27/6c18b332a0873579340bc65f2002e200#file-matplotlib_8-py)hosted with  by [**GitHub**](https://github.com/)

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/matplotlib_plotting_7.png)

Yes, our hypothesis was correct! ***Rice Bowl*** was indeed the most popular food item sold by the company.

Bar graphs should not be used for continuous values.

## 2. Pie Chart using matplotlib

Let us now see the ratio of orders from each cuisine.

A pie chart is suitable to show the proportional distribution of items within the same category.

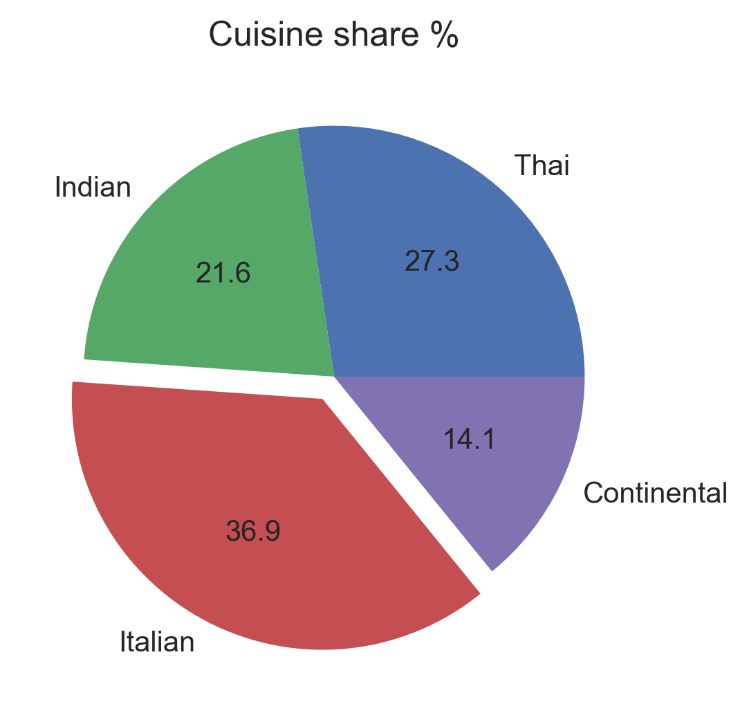
|  |  |
| --- | --- |
|  | #dictionary for cuisine and its total orders |
|  | d\_cuisine = {} |
|  |  |
|  | #total number of order |
|  | total = df['num\_orders'].sum() |
|  |  |
|  | #find ratio of orders per cuisine |
|  | for i in range(df['cuisine'].nunique()): |
|  |  |
|  | #cuisine |
|  | c = df['cuisine'].unique()[i] |
|  |  |
|  | #num of orders for the cuisine |
|  | c\_order = df[df['cuisine']==c]['num\_orders'].sum() |
|  | d\_cuisine[c] = c\_order/total |

[**view raw**](https://gist.github.com/aniruddha27/1d976786d57e9357214730bb1167975f/raw/5ed57dd2baafb1ffdd6a833daddf1d86a77f0c87/Matplotlib_8.py)[**Matplotlib\_8.py**](https://gist.github.com/aniruddha27/1d976786d57e9357214730bb1167975f#file-matplotlib_8-py)hosted with  by [**GitHub**](https://github.com/)

Let’s plot the pie chart:

|  |  |
| --- | --- |
|  | #pie plot |
|  | plt.pie([x\*100 for x in d\_cuisine.values()],labels=[x for x in d\_cuisine.keys()],autopct='%0.1f',explode=[0,0,0.1,0]) |
|  |  |
|  | #label the plot |
|  | plt.title('Cuisine share %') |
|  | plt.savefig('C:\\Users\\Dell\\Desktop\\AV Plotting images\\matplotlib\_plotting\_8.png',dpi=300,bbox\_inches='tight') |
|  | plt.show(); |

[**view raw**](https://gist.github.com/aniruddha27/d847f8aebc8d00eb5cedbb16fa7491e2/raw/f5aaa53e31860d1c6ae01a6715637437510e62da/Matplotlib_9.py)[**Matplotlib\_9.py**](https://gist.github.com/aniruddha27/d847f8aebc8d00eb5cedbb16fa7491e2#file-matplotlib_9-py)hosted with  by [**GitHub**](https://github.com/)

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/matplotlib_plotting_8.png)

* I used ***plt.pie()*** to draw the pie chart and adjust its parameters to make it more appealing
* The **autopct**parameter was used to print the values within the pie chart up to 1 decimal place
* The **explode**parameter was used to offset the Italian wedge to make it stand out from the rest. This makes it instantly clear to the viewer that people love Italian food!

A pie chart is rendered useless when there are a lot of items within a category. This will decrease the size of each slice and there will be no distinction between the items.

## 3. Box Plot using matplotlib

Since we are discussing cuisine, let’s check out which one is the most expensive cuisine! For this, I will be using a ***Box Plot***.

Box plot gives statistical information about the distribution of numeric data divided into different groups. It is useful for detecting outliers within each group.

* The lower, middle and upper part of the box represents the **25th, 50th,**and **75th** **percentile** values respectively
* The top whisker represents **Q3+1.5\*IQR**
* The bottom whisker represents **Q1-1.5\*IQR**
* Outliers are shown as scatter points
* Shows skewness in the data

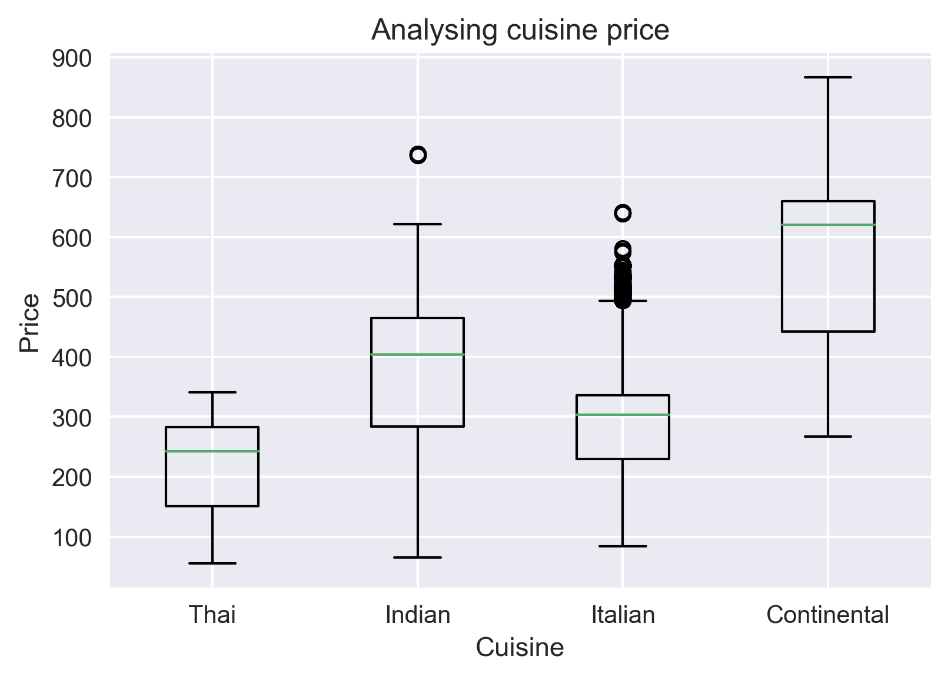
|  |  |
| --- | --- |
|  | #dictionary for base price per cuisine |
|  | c\_price = {} |
|  | for i in df['cuisine'].unique(): |
|  | c\_price[i] = df[df['cuisine']==i].base\_price |

[**view raw**](https://gist.github.com/aniruddha27/63806400d0b26492ee5cf3cdfcc770b8/raw/cf4b4a2d2d43f9bb0967c75dd632e3ce3f1c2641/Matplotlib_10.py)[**Matplotlib\_10.py**](https://gist.github.com/aniruddha27/63806400d0b26492ee5cf3cdfcc770b8#file-matplotlib_10-py)hosted with  by [**GitHub**](https://github.com/)

Plotting the boxplot below:

|  |  |
| --- | --- |
|  | #plotting boxplot |
|  | plt.boxplot([x for x in c\_price.values()],labels=[x for x in c\_price.keys()]) |
|  |  |
|  | #x and y-axis labels |
|  | plt.xlabel('Cuisine') |
|  | plt.ylabel('Price') |
|  |  |
|  | #plot title |
|  | plt.title('Analysing cuisine price') |
|  |  |
|  | #save and display |
|  | plt.savefig('C:\\Users\\Dell\\Desktop\\AV Plotting images\\matplotlib\_plotting\_9.png',dpi=300,bbox\_inches='tight') |
|  | plt.show(); |

[**view raw**](https://gist.github.com/aniruddha27/80df820a42d82be81401e161d2f9ea2b/raw/e0586e62dc44c774fae13a386eb7d8c9cebabe7f/Matplotlib_11.py)[**Matplotlib\_11.py**](https://gist.github.com/aniruddha27/80df820a42d82be81401e161d2f9ea2b#file-matplotlib_11-py)hosted with  by [**GitHub**](https://github.com/)

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/matplotlib_plotting_9.png)

***Continental***cuisine was the most expensive cuisine served by the company! Even its median price is higher than the maximum price of all the cuisines.

Box plot does not show the distribution of data points within each group.

## 4. Histogram using matplotlib

On the topic of prices, did we forget to inspect the base price and checkout price? Don’t worry, we will do that using a histogram.

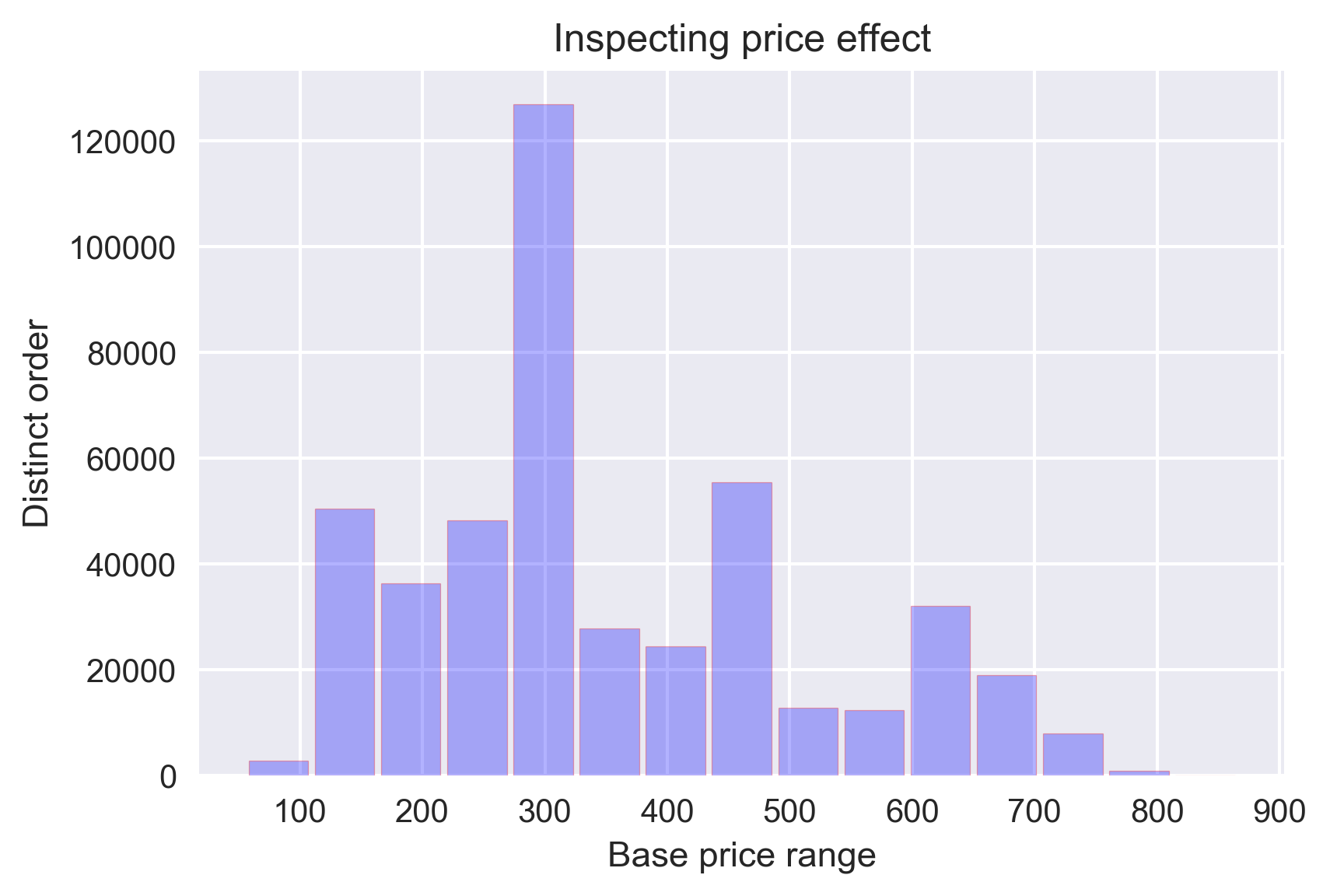
A histogram shows the distribution of numeric data through a continuous interval by segmenting data into different bins. Useful for inspecting skewness in the data.

Since **base\_price**is a continuous variable, we will inspect its range in different distinct orders using a histogram. We can do this using ***plt.hist()***.

But the confusing part is what should be the number of bins? By default, it is 10. However, there is no correct answer and you can vary it according to your dataset to best visualize it.

|  |  |
| --- | --- |
|  | #plotting histogram |
|  | plt.hist(df['base\_price'],rwidth=0.9,alpha=0.3,color='blue',bins=15,edgecolor='red') |
|  |  |
|  | #x and y-axis labels |
|  | plt.xlabel('Base price range') |
|  | plt.ylabel('Distinct order') |
|  |  |
|  | #plot title |
|  | plt.title('Inspecting price effect') |
|  |  |
|  | #save and display the plot |
|  | plt.savefig('C:\\Users\\Dell\\Desktop\\AV Plotting images\\matplotlib\_plotting\_10.png',dpi=300,bbox\_inches='tight') |
|  | plt.show(); |

[**view raw**](https://gist.github.com/aniruddha27/e00afa2019e63be0b520cfb1ca813d1f/raw/3cb9604552e7baa735715da5112c2b4d7d7d44d8/Matplotlib_12.py)[**Matplotlib\_12.py**](https://gist.github.com/aniruddha27/e00afa2019e63be0b520cfb1ca813d1f#file-matplotlib_12-py)hosted with  by [**GitHub**](https://github.com/)

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/matplotlib_plotting_10.png)

I have chosen the number of bins as 15 and it is evident that most of the orders had a base price of ~300.

It is easy to confuse histograms with bar plots. But remember, histograms are used with continuous data whereas bar plots are used with categorical data.

## 5. Line Plot and Subplots using matplotlib

A line plot is useful for visualizing the trend in a numerical value over a continuous time interval.

How are the weekly and monthly sales of the company varying? This is a critical business question that makes or breaks the marketing strategy.

Before exploring that, I will create two lists for storing the week-wise and month-wise revenue of the company:

|  |  |
| --- | --- |
|  | #new revenue column |
|  | df['revenue'] = df.apply(lambda x: x.checkout\_price\*x.num\_orders,axis=1) |
|  |  |
|  | #new month column |
|  | df['month'] = df['week'].apply(lambda x: x//4) |
|  |  |
|  | #list to store month-wise revenue |
|  | month=[] |
|  | month\_order=[] |
|  |  |
|  | for i in range(max(df['month'])): |
|  | month.append(i) |
|  | month\_order.append(df[df['month']==i].revenue.sum()) |
|  |  |
|  | #list to store week-wise revenue |
|  | week=[] |
|  | week\_order=[] |
|  |  |
|  | for i in range(max(df['week'])): |
|  | week.append(i) |
|  | week\_order.append(df[df['week']==i].revenue.sum()) |

[**view raw**](https://gist.github.com/aniruddha27/81d5357f0e98162b1ad1007ec1e136fa/raw/202f6868df4f565d47cb93e9d7cc65377fef2411/Matplotlib_13.py)[**Matplotlib\_13.py**](https://gist.github.com/aniruddha27/81d5357f0e98162b1ad1007ec1e136fa#file-matplotlib_13-py)hosted with  by [**GitHub**](https://github.com/)

I will compare the revenue of the company in every week as well as in every month using two line-plots drawn side by side. For this, I will be using the ***plt.subplots()*** function.

Matplotlib subplots makes it easy to view and compare different plots in the same figure.

To understand how this function works, you need to know what ***Figure***, ***Axes***, and ***Axis***are in a matplotlib plot.

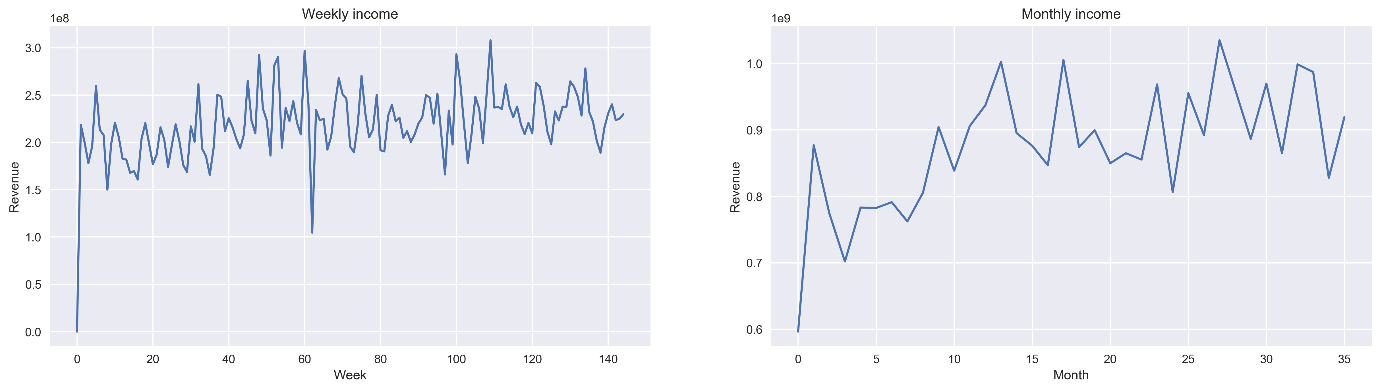
**Figure**is the outermost container for the Matplotlib plot(s). There can a single or multiple plots, called **Axes,**within a ***Figure***. Each of these Axes contains the x and y-axis known as the **Axis**.

The ***plt.subplots()*** figure returns the figure and axes. You can provide as an input to the function how you want to display the axes within the figure. These will be adjusted using the **nrows**and **ncols**parameters. You can even adjust the size of the figure using the **figsize**parameter.

Axes are returned as a list. To plot for specific axes, you can access them as a list object. The rest of the plotting is done the same way as simple plots:

|  |  |
| --- | --- |
|  | #subplots returns a Figure and an Axes object |
|  | fig,ax=plt.subplots(nrows=1,ncols=2,figsize=(20,5)) |
|  |  |
|  | #manipulating the first Axes |
|  | ax[0].plot(week,week\_order) |
|  | ax[0].set\_xlabel('Week') |
|  | ax[0].set\_ylabel('Revenue') |
|  | ax[0].set\_title('Weekly income') |
|  |  |
|  | #manipulating the second Axes |
|  | ax[1].plot(month,month\_order) |
|  | ax[1].set\_xlabel('Month') |
|  | ax[1].set\_ylabel('Revenue') |
|  | ax[1].set\_title('Monthly income') |
|  |  |
|  | #save and display the plot |
|  | plt.savefig('C:\\Users\\Dell\\Desktop\\AV Plotting images\\matplotlib\_plotting\_11.png',dpi=300,bbox\_inches='tight') |
|  | plt.show(); |

[**view raw**](https://gist.github.com/aniruddha27/7a7703e556191b560303e163cc0353a9/raw/2aa6992cabf0d8d79d63e6b19f9cb8e28d7153d4/Matplotlib_14.py)[**Matplotlib\_14.py**](https://gist.github.com/aniruddha27/7a7703e556191b560303e163cc0353a9#file-matplotlib_14-py)hosted with  by [**GitHub**](https://github.com/)

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/matplotlib_plotting_11.png)

We can see an increasing trend in the number of food orders with the number of weeks and months, though the trend is not very strong.

## 6. Scatter Plot using matplotlib

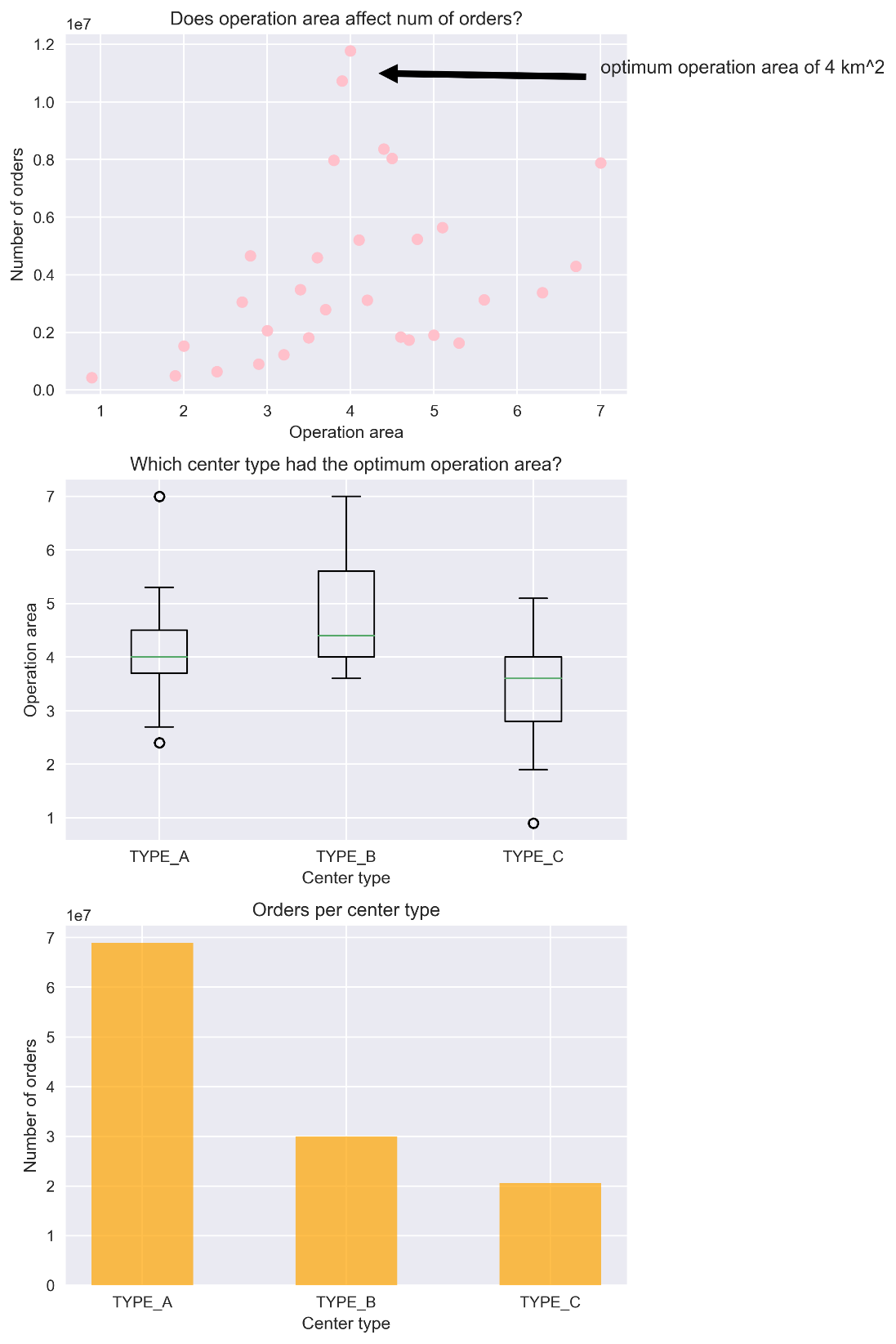
Finally, I will try to analyze whether the center type had any effect on the number of orders from different center types. I will do this by comparing a scatter plot, a boxplot and a bar graph in the same figure.

We have already seen the use of boxplots and bar graphs, but scatter plots have their own advantages.

Scatter plots are useful for showing the relationship between two variables. Any correlation between variables or outliers in the data can be easily spotted using scatter plots.

|  |  |
| --- | --- |
|  | center\_type\_name = ['TYPE\_A','TYPE\_B','TYPE\_C'] |
|  |  |
|  | #relation between op area and number of orders |
|  | op\_table=pd.pivot\_table(df,index='op\_area',values='num\_orders',aggfunc=np.sum) |
|  |  |
|  | #relation between center type and op area |
|  | c\_type = {} |
|  | for i in center\_type\_name: |
|  | c\_type[i] = df[df['center\_type']==i].op\_area |
|  |  |
|  | #relation between center type and num of orders |
|  | center\_table=pd.pivot\_table(df,index='center\_type',values='num\_orders',aggfunc=np.sum) |
|  |  |
|  | #subplots |
|  | fig,ax = plt.subplots(nrows=3,ncols=1,figsize=(8,12)) |
|  |  |
|  | #scatter plots |
|  | ax[0].scatter(op\_table.index,op\_table['num\_orders'],color='pink') |
|  | ax[0].set\_xlabel('Operation area') |
|  | ax[0].set\_ylabel('Number of orders') |
|  | ax[0].set\_title('Does operation area affect num of orders?') |
|  | ax[0].annotate('optimum operation area of 4 km^2',xy=(4.2,1.1\*10\*\*7),xytext=(7,1.1\*10\*\*7),arrowprops=dict(facecolor='black', shrink=0.05),fontsize=12) |
|  |  |
|  | #boxplot |
|  | ax[1].boxplot([x for x in c\_type.values()], labels=[x for x in c\_type.keys()]) |
|  | ax[1].set\_xlabel('Center type') |
|  | ax[1].set\_ylabel('Operation area') |
|  | ax[1].set\_title('Which center type had the optimum operation area?') |
|  |  |
|  | #bar graph |
|  | ax[2].bar(center\_table.index,center\_table['num\_orders'],alpha=0.7,color='orange',width=0.5) |
|  | ax[2].set\_xlabel('Center type') |
|  | ax[2].set\_ylabel('Number of orders') |
|  | ax[2].set\_title('Orders per center type') |
|  |  |
|  | #show figure |
|  | plt.tight\_layout() |
|  | plt.savefig('C:\\Users\\Dell\\Desktop\\AV Plotting images\\matplotlib\_plotting\_12.png',dpi=300,bbox\_inches='tight') |
|  | plt.show(); |

[**view raw**](https://gist.github.com/aniruddha27/bf7cadb77ee0eedc529621b44270ff1a/raw/44eb572d4340b8321a0a1604261c46295681ecee/Matplotlib_15.py)[**Matplotlib\_15.py**](https://gist.github.com/aniruddha27/bf7cadb77ee0eedc529621b44270ff1a#file-matplotlib_15-py)hosted with  by [**GitHub**](https://github.com/)

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/matplotlib_plotting_12.png)

The **scatter plot** makes it instantly visible that the optimum operation area of a center is 4 km sq. The boxplot shows that the TYPE\_A center type had the most number of optimum size centers because of a compact box with a median around 4 km sq. Because of this, they had more orders placed by customers than any other center type.