Data Visualization Basics

Visualization - Great visualization gives great understanding of data and then leads to better analysis

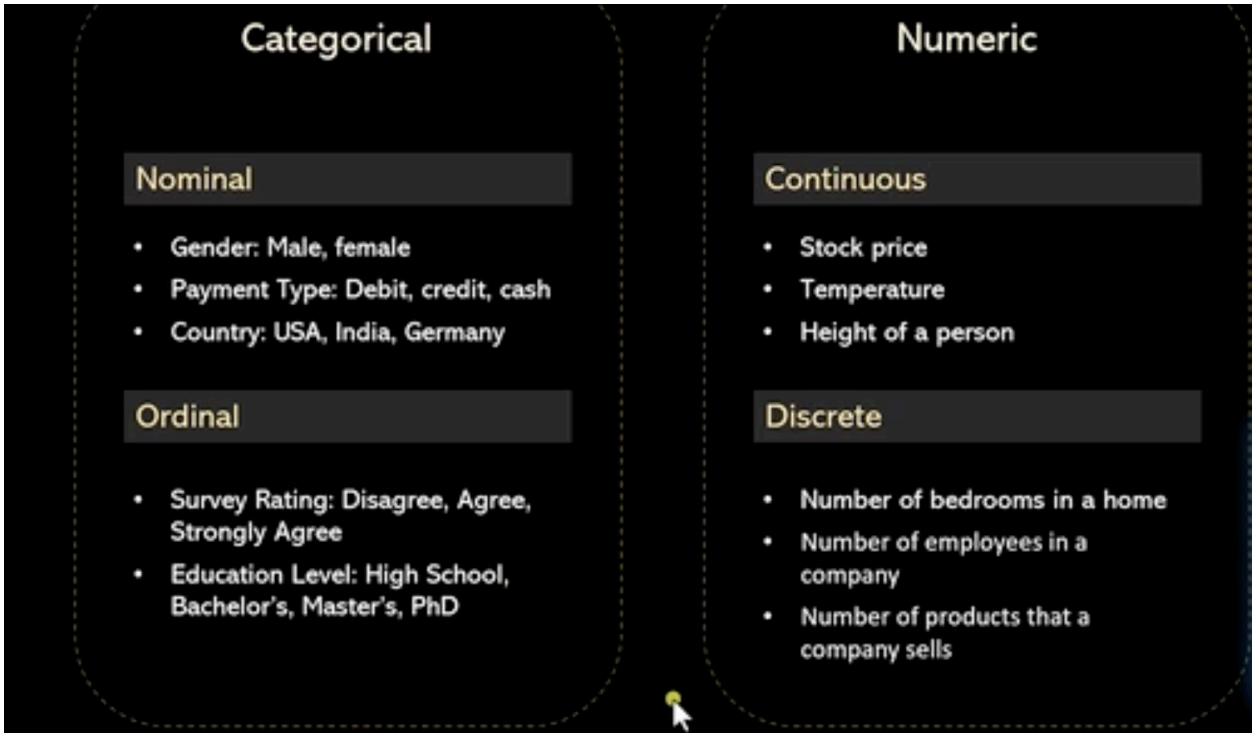
In the context of machine learning, data visualization plays a key role because we can analyze the data more effectively from a visual representation than from plain data.

Great! Here are some common data visualization techniques, especially useful in the context of machine learning and data analysis:

Lets get some understanding on type of data so we can further discuss how to visualize using plots like line char, bar, pie, histogram and so on

Data:

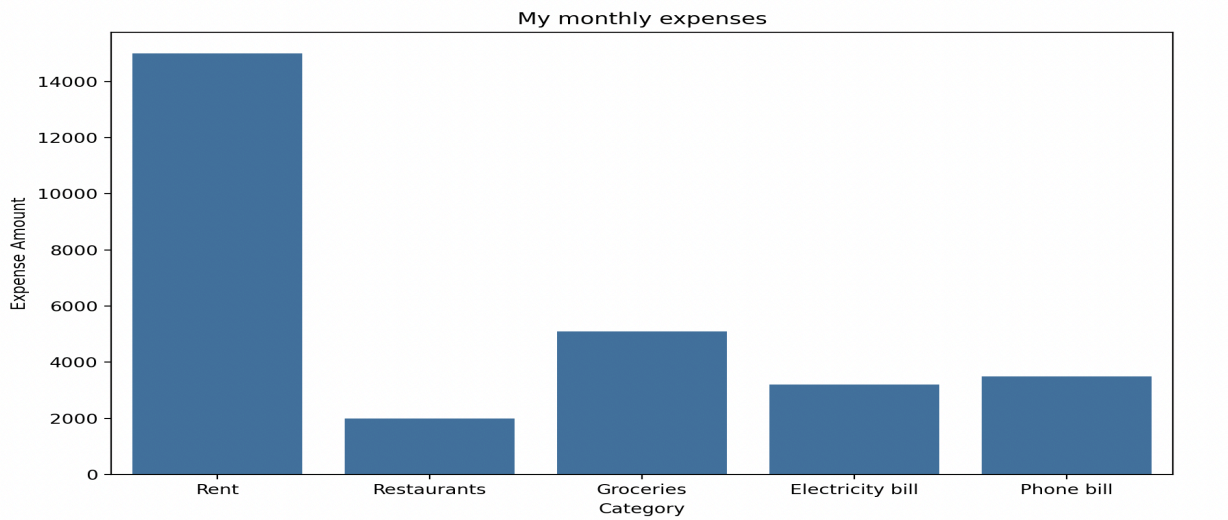
- Numerical - This can be continuous or Discrete
- Categorical - This can be Nominal or Ordinal



Bar Chat: Highlight distributions and outliers in your data.

Think about your daily expenses like Rent, Restaurants, Groceries, Electricity bill, Phone bill

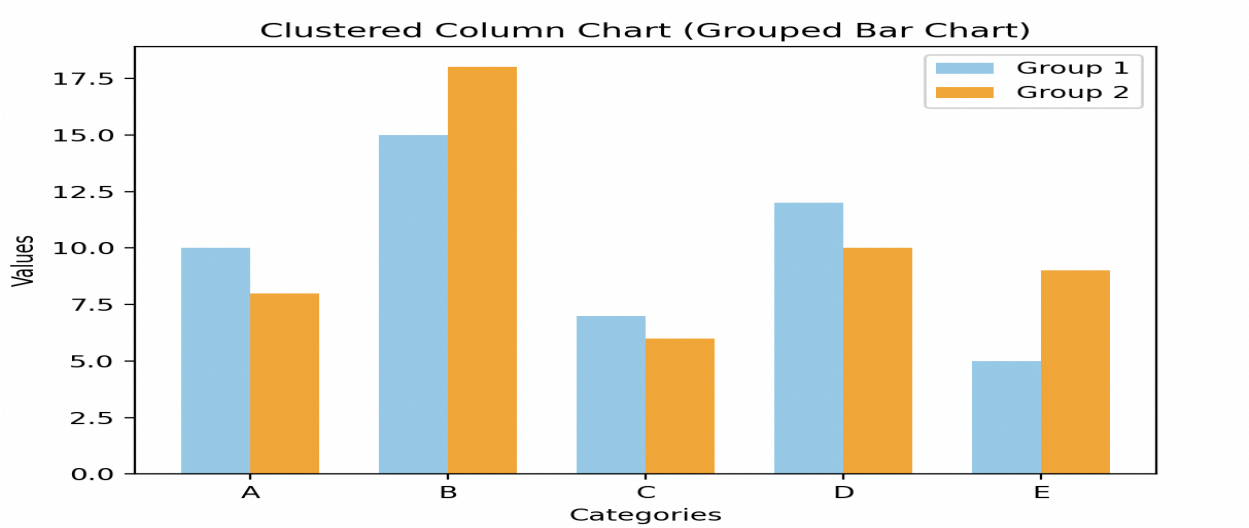
Purpose: Compare categories or discrete values.
Use Case: Showing the frequency of categories or comparing multiple groups side-by-side.




```python
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns

df = pd.read_excel('expenses_and_apple_revenue.xlsx')

print(df.columns)


# Set the width and height of the figure
plt.figure(figsize=(10,6))

#Add title
plt.title("My monthly expenses")
plt.xlabel('Category')
plt.ylabel('Expense Amount')

# Bar chart showing monthly expenses
sns.barplot(x=df['Category'], y=df['Expense Amount'])
#plt.bar(df['Category'], df['Expense Amount'])

plt.show()
```

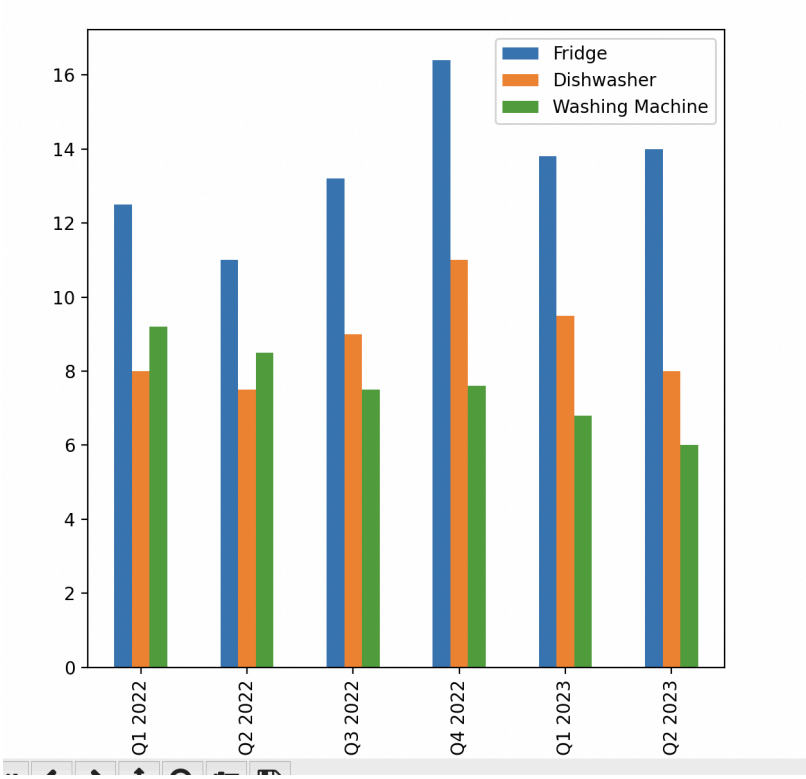
Tips - Use Bar chart when you have benchmark value to compare to
- Use a horizontal bar chart when category labels are long
- Use a vertical bar chart for time series data

If we have more columns - go for clustered column chart




```python
# Data for the grouped bar chart
categories = ['A', 'B', 'C', 'D', 'E']
values1 = [10, 15, 7, 12, 5]  # First group of values
values2 = [8, 18, 6, 10, 9]   # Second group of values

# The number of categories
n = len(categories)

# Bar width (the width of each bar)
bar_width = 0.35

# Positions of the bars on the x-axis for each group
index = np.arange(n)

# Create the figure and axis
fig, ax = plt.subplots()

# Create bars for both sets of data
ax.bar(index, values1, bar_width, label='Group 1', color='skyblue')
ax.bar(index + bar_width, values2, bar_width, label='Group 2', color='orange')

# Add labels and title
ax.set_xlabel('Categories')
ax.set_ylabel('Values')
ax.set_title('Clustered Column Chart (Grouped Bar Chart)')
ax.set_xticks(index + bar_width / 2)  # To center the ticks between the bars
ax.set_xticklabels(categories)

# Add a legend
ax.legend()
```

Another simplest way to plot the Bar chart is below:)

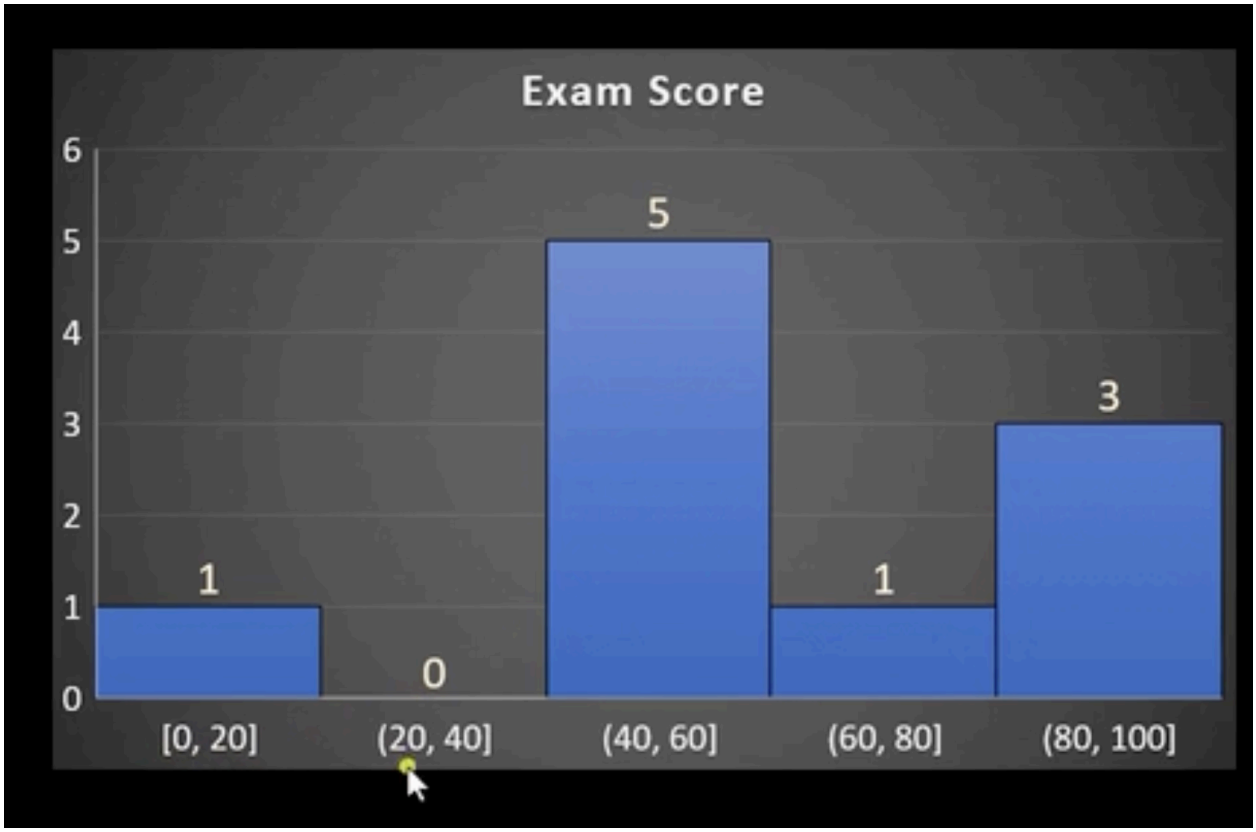



```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

df = pd.read_excel('linechart.xlsx')

print(df.columns)
df.plot(kind='bar', x='Quarter')

plt.show()
```

Histogram: Show the distribution of data (e.g., income ranges, test scores).

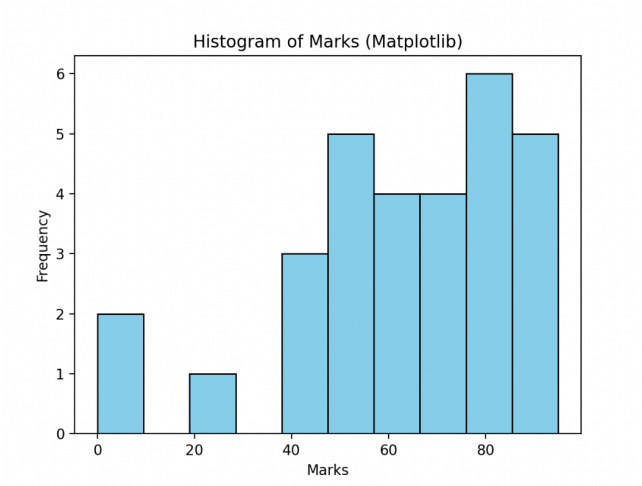
—----------

Histogram is similar to Bar chart, but in histogram will have bins/buckets.
When we want to plot range then we go for histogram
Each bin/category will have equal length



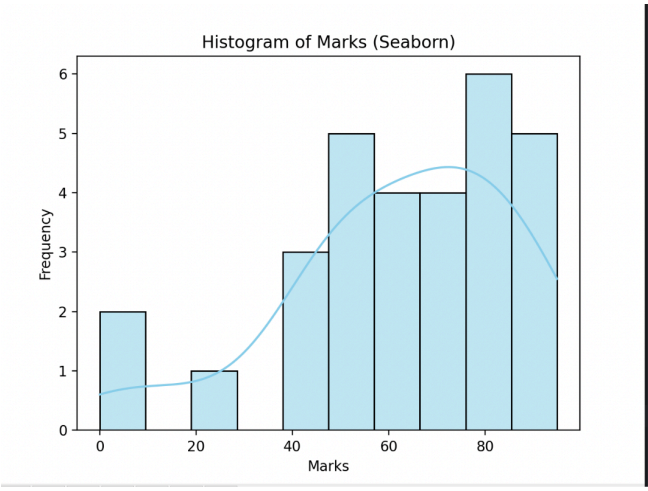
The brackets notation meaning:
(/) —> does not include the start/end value specified in the range
[/]-> includes the start/end value specified in the range

Example if we want to plot exam scores in the range and number of people in that range then we go for this.




```python
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd

df = pd.read_excel('histograms.xlsx')


plt.hist(df['Exam_Score'], bins=10, color='skyblue', edgecolor='black')
plt.xlabel('Marks')
plt.ylabel('Frequency')
plt.title('Histogram of Marks (Matplotlib)')
plt.show()
```

If we use seaborn we can specify **Kernel Density Estimate (KDE)** curve with kde=True

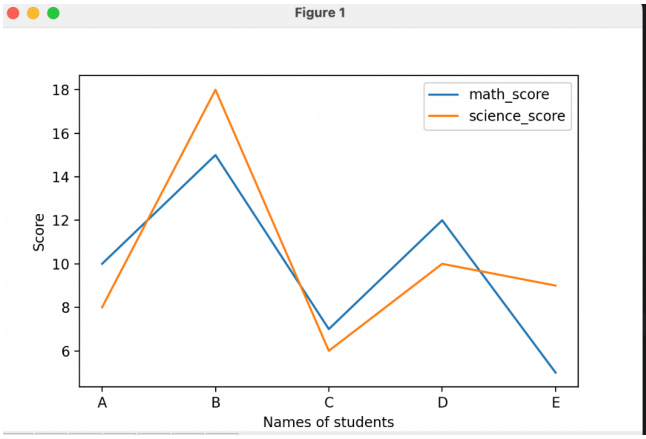



```python
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd

df = pd.read_excel('histograms.xlsx')


# plt.hist(df['Exam_Score'], bins=10, color='skyblue', edgecolor='black')
# plt.xlabel('Marks')
# plt.ylabel('Frequency')
# plt.title('Histogram of Marks (Matplotlib)')
# plt.show()


sns.histplot(df['Exam_Score'], bins=10, kde=True, color='skyblue')
plt.xlabel('Marks')
plt.ylabel('Frequency')
plt.title('Histogram of Marks (Seaborn)')
plt.show()
```

Line Chart:

——————————

Visualize trends over time, like stock market performance.
The syntax is similar to Bar chart we provide x and y values(columns)




```python
import matplotlib.pyplot as plt
import numpy as np

# Data for the grouped bar chart
names = ['A', 'B', 'C', 'D', 'E']
math_score = [10, 15, 7, 12, 5]  # First group of values
science_score = [8, 18, 6, 10, 9]   # Second group of values


plt.plot( *args: names, math_score, label='math_score')
plt.plot( *args: names, science_score, label='science_score')
plt.xlabel('Names of students')
plt.ylabel('Score')
plt.legend()
# Show the chart
plt.show()
```

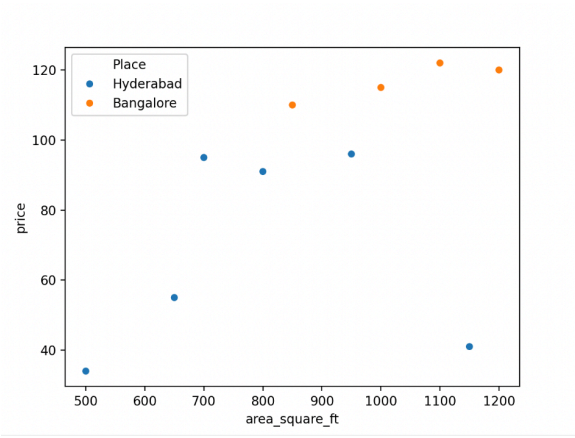
You can either use matplot or sns, both produce similar visualization.

Scatter Plot:

——————————

Visualize relationships between two variables, like hours studied vs. exam scores.
Below we used an example of area_square_ft and price.

We added a new column to the df called Place if the price is > 100 Banglore otherwise Hyderabad.
We can use this column in hue to differentiate the place with color coding.

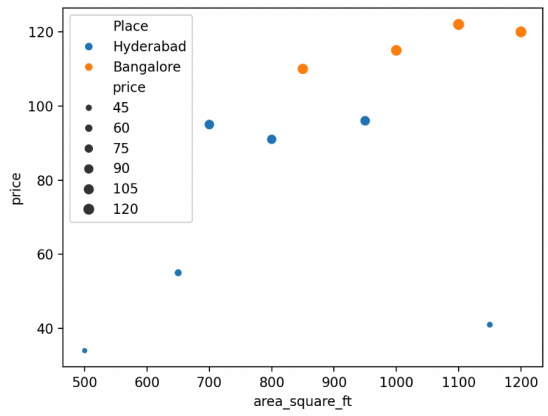



```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

df = pd.read_excel('scatter_plot.xlsx')

df["Place"] = df['price'].apply(lambda x: 'Bangalore' if x > 100 else 'Hyderabad')

print(df.columns)
sns.scatterplot(df, x='area_square_ft', y='price', hue='Place')

plt.show()
```

Scatter plot is widely used to understand the relationship between 2 datasets or to find outliers in a dataset.

We can go for bubble charts as well, there is slight variation between scatter plot and bubble plot.
Here we will also show the size of the bubble.

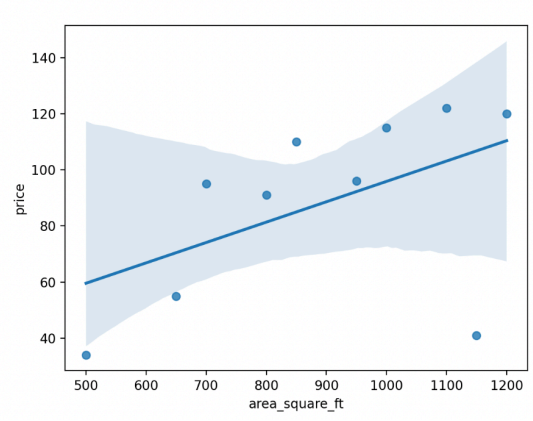
A **bubble chart** is a variation of a scatter plot where each point is represented by a bubble, and the size of the bubble can represent an additional dimension of data.




```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

df = pd.read_excel('scatter_plot.xlsx')

df["Place"] = df['price'].apply(lambda x: 'Bangalore' if x > 100 else 'Hyderabad')

print(df.columns)
sns.scatterplot(df, x='area_square_ft', y='price',size ='price',  hue='Place')

plt.show()
```

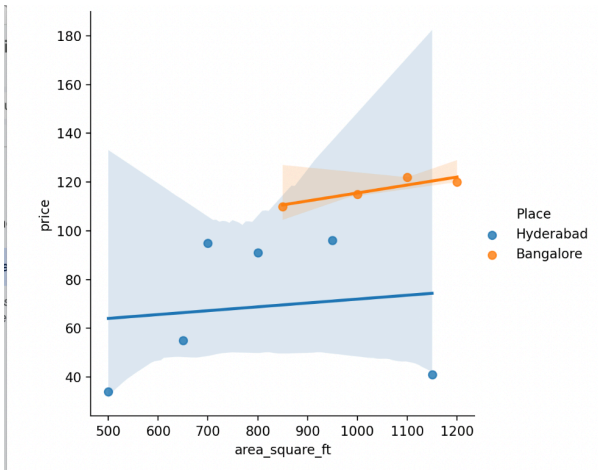
Scatter plot suggests that area_square_ft and price are highly correlated. where houses with higher area_square_ft typically also tend to have more in price.

We can also draw a regression line that best fits the data.



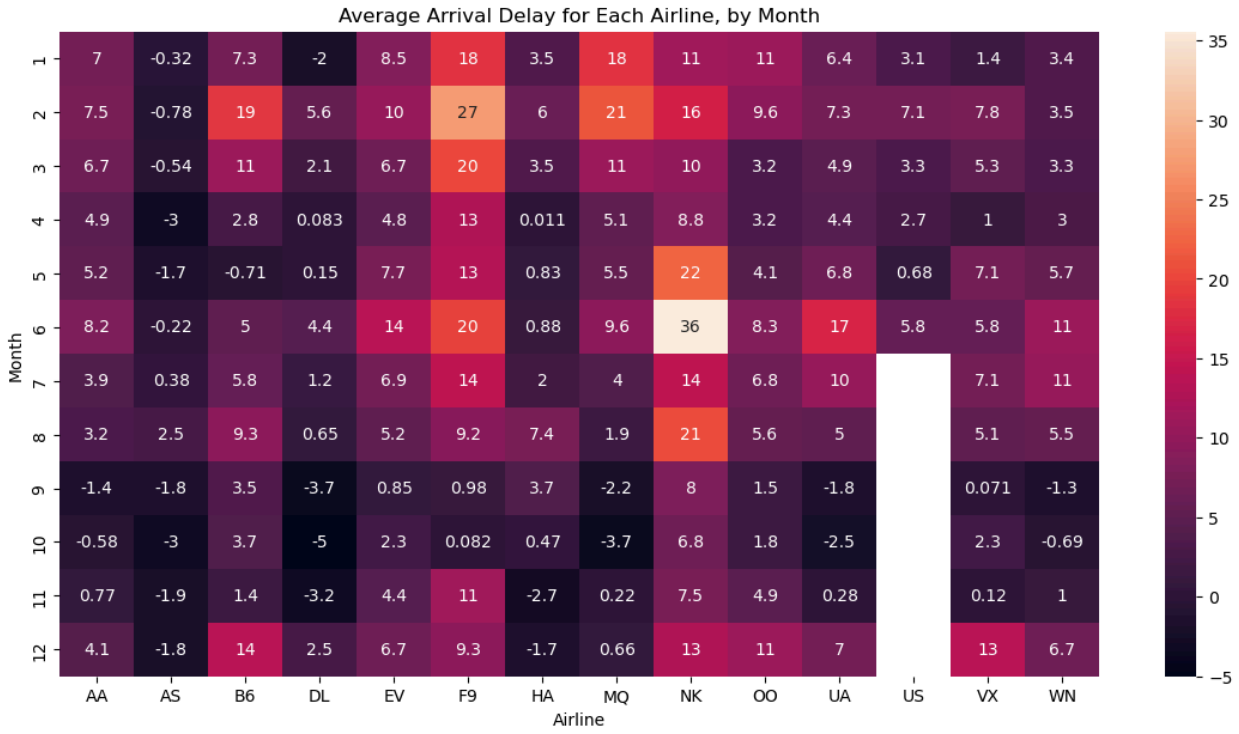

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

df = pd.read_excel('scatter_plot.xlsx')

df["Place"] = df['price'].apply(lambda x: 'Bangalore' if x > 100 else 'Hyderabad')

print(df.columns)
#sns.scatterplot(df, x='area_square_ft', y='price',size ='price',  hue='Place')
sns.regplot(x=df['area_square_ft'], y=df['price'])
plt.show()
```

We can use the sns.lmplot command to add two regression lines, corresponding to Hyderabad and Bangalore.

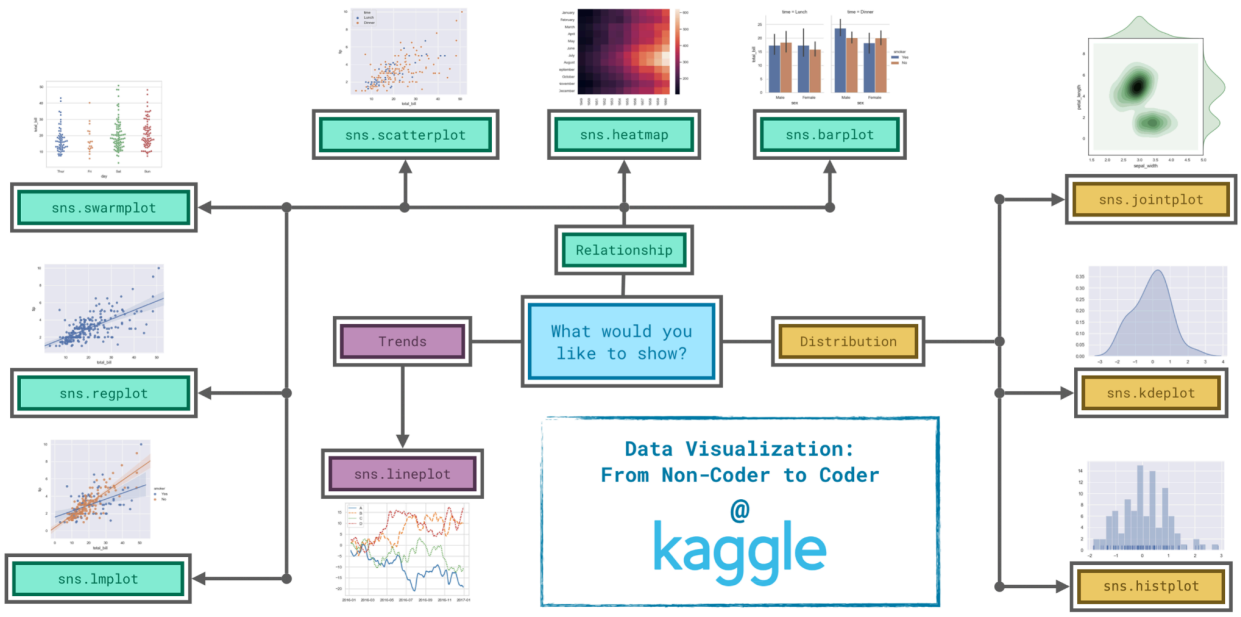



```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

df = pd.read_excel('scatter_plot.xlsx')

df["Place"] = df['price'].apply(lambda x: 'Bangalore' if x > 100 else 'Hyderabad')

print(df.columns)
#sns.scatterplot(df, x='area_square_ft', y='price',size ='price',  hue='Place')
#sns.regplot(x=df['area_square_ft'], y=df['price'])

sns.lmplot(x='area_square_ft', y= 'price', hue="Place", data=df)

plt.show()
```

Heatmaps: Show the correlation between different features.
It is very much useful when we have large features and find the correlation for data analysis and cleaning.



I got an image from Kaggle when to use which plot



Since it's not always easy to decide how to best tell the story behind your data, we've broken the chart types into three broad categories to help with this.

- **Trends** - A trend is defined as a pattern of change.
  - sns.line plot - **Line charts** are best to show trends over a period of time, and multiple lines can be used to show trends in more than one group.
- **Relationship** - There are many different chart types that you can use to understand relationships between variables in your data.
  - sns.barplot - **Bar charts** are useful for comparing quantities corresponding to different groups.
  - sns.heatmap - **Heatmaps** can be used to find color-coded patterns in tables of numbers.
  - sns.scatterplot - **Scatter plots** show the relationship between two continuous variables; if color-coded, we can also show the relationship with a third categorical variable.
  - sns.regplot - Including a **regression line** in the scatter plot makes it easier to see any linear relationship between two variables.
  - sns.lmplot - This command is useful for drawing multiple regression lines, if the scatter plot contains multiple, color-coded groups.
  - sns.swarmplot - **Categorical scatter plots** show the relationship between a continuous variable and a categorical variable.
- **Distribution** - We visualize distributions to show the possible values that we can expect to see in a variable, along with how likely they are.
  - sns.histplot - **Histograms** show the distribution of a single numerical variable.
  - sns.kdeplot - **KDE plots** (or **2D KDE plots**) show an estimated, smooth distribution of a single numerical variable (or two numerical variables).
  - sns.jointplot - This command is useful for simultaneously displaying a 2D KDE plot with the corresponding KDE plots for each individual variable.

Changing styles with seaborn

All of the commands have provided a nice, default style to each of the plots. However, you may find it useful to customize how your plots look, and thankfully, this can be accomplished by just adding one more line of code!

As always, we need to begin by setting up the coding environment. (*This code is hidden, but you can un-hide it by clicking on the "Code" button immediately below this text, on the right.*)