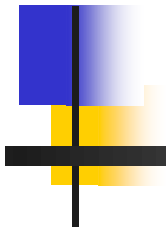
Exploratory Data Analysis

IOE 373 Lecture 20



Topics

- Exploratory Data Analysis
- Data Preprocessing
- Intro to Visualization

Exploratory Data Analysis (EDA)

What is EDA?

- Combination of visualization techniques and statistical methods
- Exploring and summarizing key information within the data

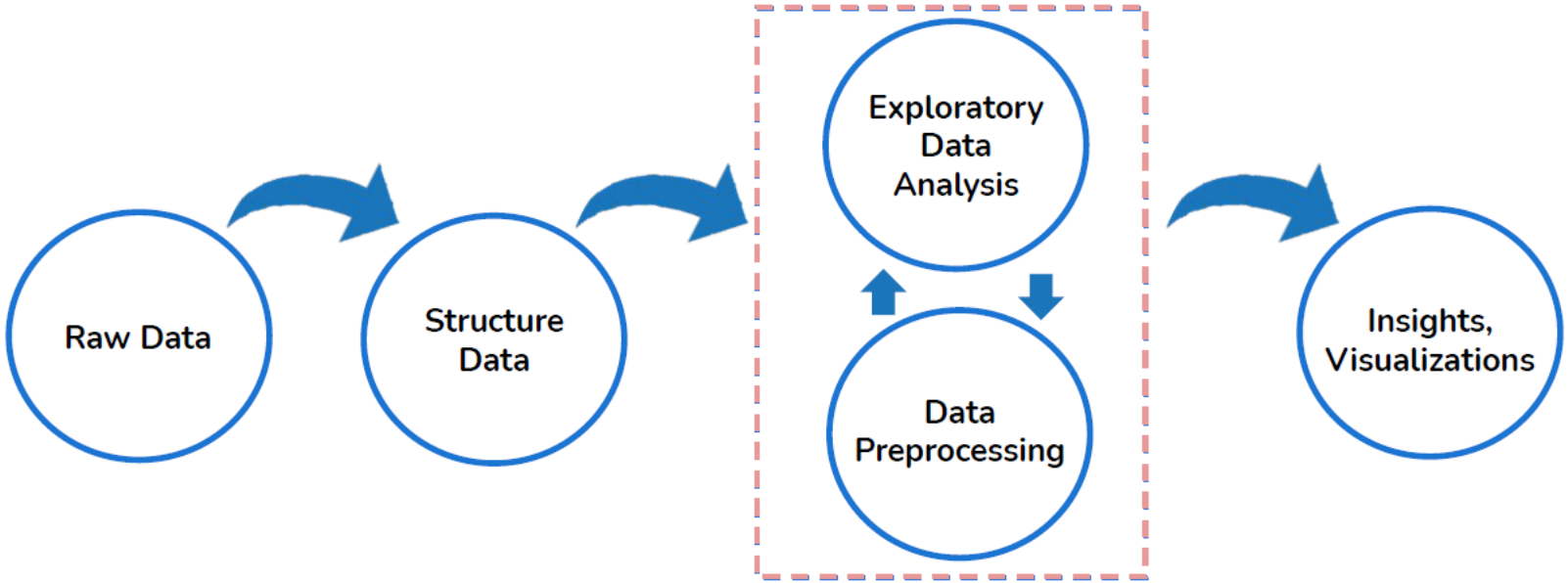
Why of EDA?

- First step in any analysis
- Gain good insights into the data
- Uncover underlying structure of the data
- Detect and figure out the best strategy to handle *unclean data* (missing values, outliers etc.)
- Identify initial set of observations and insights

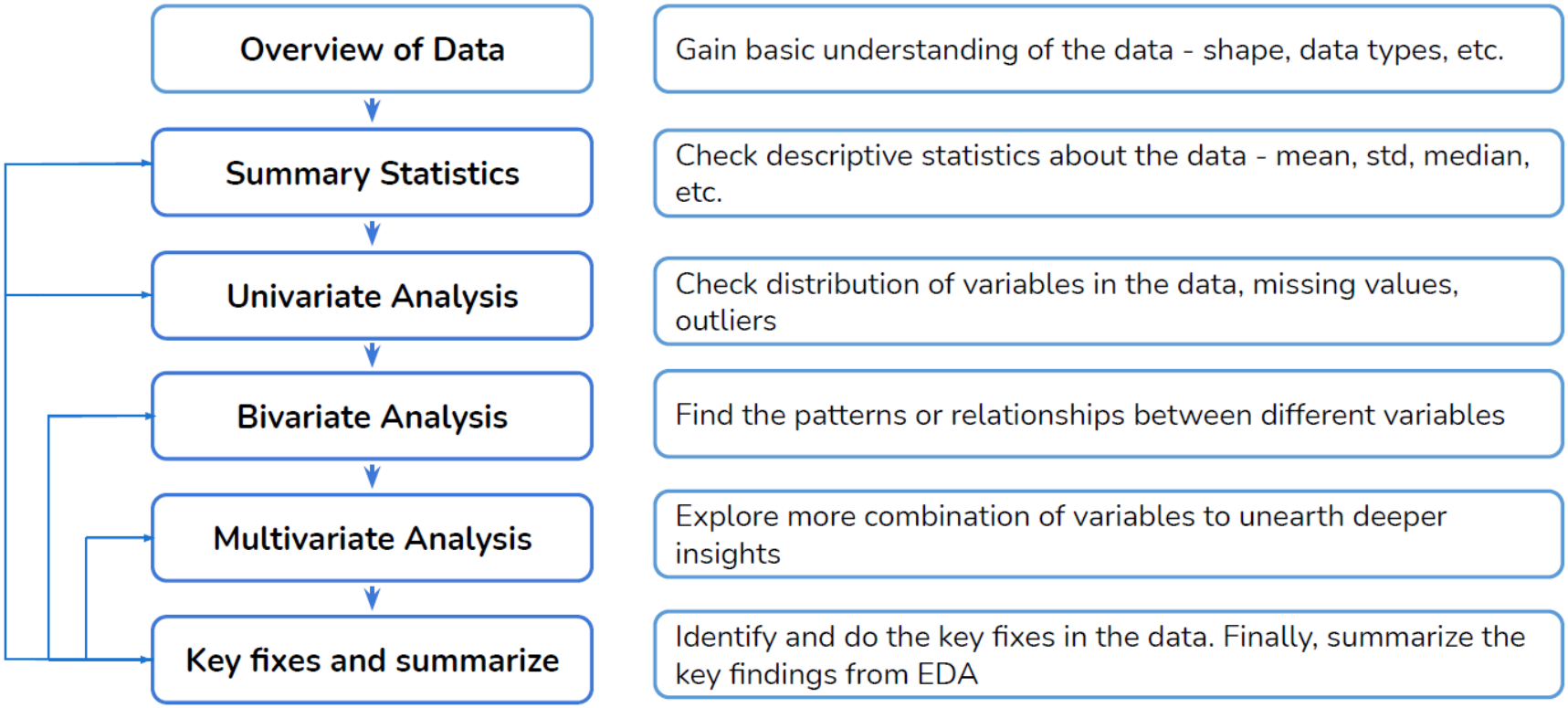
Data Preprocessing

- Data preprocessing is the process of preparing and organizing the raw data into a structured format before building a machine learning model
  - Raw data is often incomplete, inconsistent and may have other issues.
    - It might lead to wrong insights
    - Decisions taken from this data can be counter productive for the organization

EDA and Data Preprocessing



Key Steps in EDA



Missing Values

- What are missing values?
  - Missing values occur when no data value is stored for the variable in an observation
  - Missing values can have a significant effect on the inferences that are drawn from the data
- What are some different types of missing values?
  - Values which are not actually missing and represents some information about the data
    - Example - Number of hours an employee works everyday. Missing values can mean that employee was absent or on leave that particular day.
  - Values which are actually missing and provides no information about the data
    - Example - A weighing scale that is running out of batteries. Some data will simply be missing randomly.

How to deal with missing values?

- If values are not actually missing (e.g. the null has meaning) then we can replace the missing values with the value they actually represent in the data.
  - In the employee missing working hours, we can replace all the missing working hours of an employee with values of 0.

How to deal with missing values?

- If values are actually missing, then we must explore the importance and extent of missing values in the data
  - If the variable has large percentage of missing values (say more than 70%) and is not significant for our analysis, then we can drop that variable
  - If the percentage of missing values is small or the variable is significant for our analysis, then we can replace the missing values in that variable using:
    - Mean or median of that variable if the variable is continuous
    - Mode of that variable if the variable is categorical
    - Sometimes we use functions like min, max, etc. to replace the missing values depending on the dataset

https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html

Handling Methods

Argument	Description
`dropna`	Filter axis labels based on whether values for each label have missing data, with varying thresholds for how much missing data to tolerate.
`fillna`	Fill in missing data with some value or using an interpolation method such as `'ffill'` or `'bfill'`.
`isnull`	Return like-type object containing boolean values indicating which values are missing / NA.
`notnull`	Negation of `isnull`.

KNN Imputation

- Imputation for completing missing values using k-Nearest Neighbors.

- Each sample's missing values are imputed using the mean value from n_neighbors
  - Two samples are close if the features that neither is missing are close.

KNN Imputation - Example

```
import numpy as np
import pandas as pd
from sklearn.impute import KNNImputer
```

```
X = [[1, 2, np.nan], [3, 4, 3], [np.nan, 6, 5], [8, 8, 7]]
df=pd.DataFrame(X)
df
```

	0	1	2
0	1.0	2	NaN
1	3.0	4	3.0
2	NaN	6	5.0
3	8.0	8	7.0

Closest 2 neighbors to Missing one (rows 1 and 2)

	0	1	2
0	1.0	2	NaN
1	3.0	4	3.0
2	NaN	6	5.0
3	8.0	8	7.0

Closest 2 neighbors to Missing one (rows 1 and 3)

```
imputer = KNNImputer(n_neighbors=2)
imputer.fit_transform(df)
```

```
array([[1. , 2. , 4. ],
       [3. , 4. , 3. ],
       [5.5, 6. , 5. ],
       [8. , 8. , 7. ]])
```

Imputed value is the average based on the closest neighbors (3+5)/2=4

```
imputer = KNNImputer(n_neighbors=2)
imputer.fit_transform(df)
```

```
array([[1. , 2. , 4. ],
       [3. , 4. , 3. ],
       [5.5, 6. , 5. ],
       [8. , 8. , 7. ]])
```

Imputed value is the average based on the closest neighbors (3+8)/2=5.5

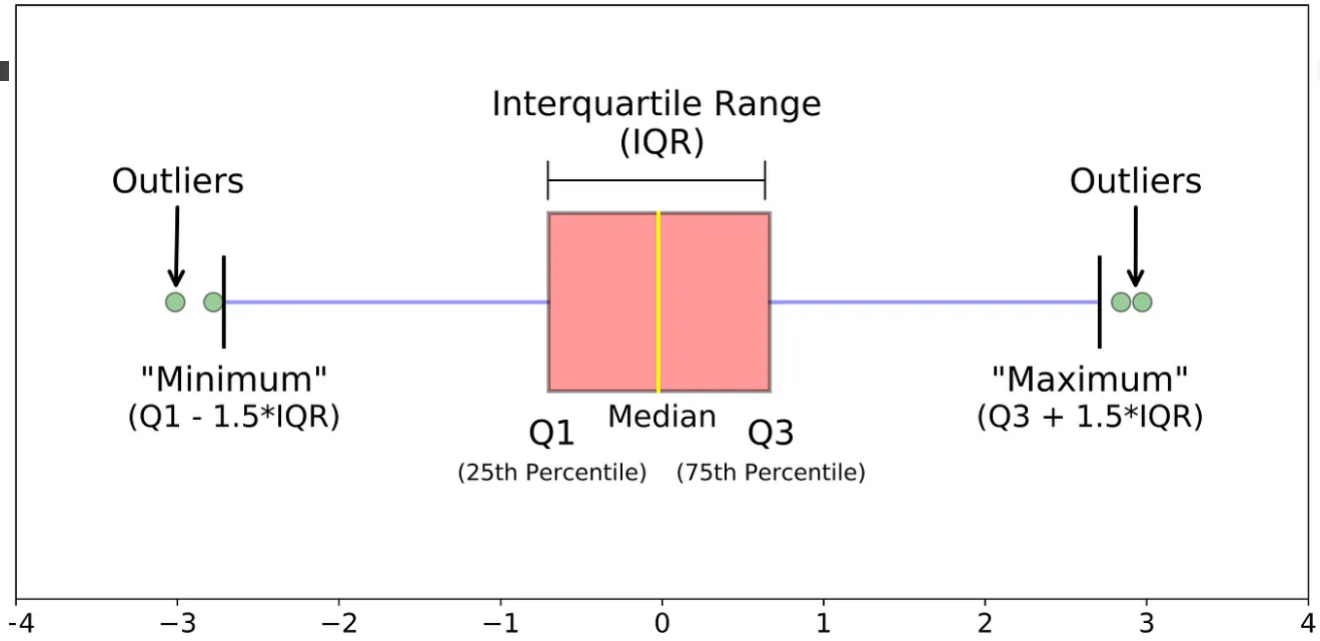
How to deal with outliers?

- Not a unique way of dealing with outliers. But some common practices are:
  - Analyze outliers before treating them.
  - If an outlier represents the general trend then there is no need to treat it
    - Example - Income is generally a skewed variable but all extreme points might not be outliers
  - If we decide to treat the outlier after analyzing it, then:
    - we can drop them but we would lose information in other columns of the data
    - we can cap outliers at certain values, say 5th percentile or 95th percentile
    - We can set a threshold using Interquartile Range (IQR) and remove the outliers greater than that threshold value

Outliers

- What are outliers?
  - Outliers are the observations which are very different/extreme with respect to other observations. In order to analyze the data, sometimes we have to work with outliers.
- How to detect outliers?
  - Box plot: We can visualize the outliers by using box plot (and boxplot outlier rules)
  - Scatter plot: We can check outliers using scatter plot by identifying the data points that are far from other observations.

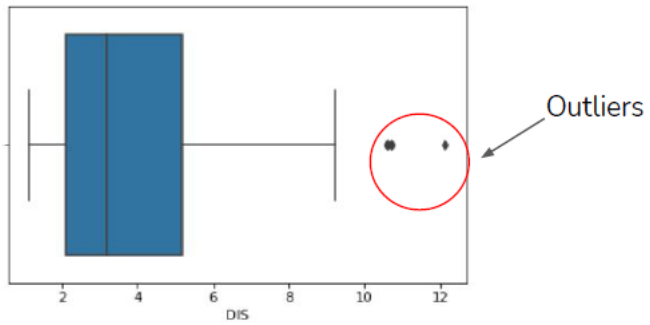
Box Plot Method



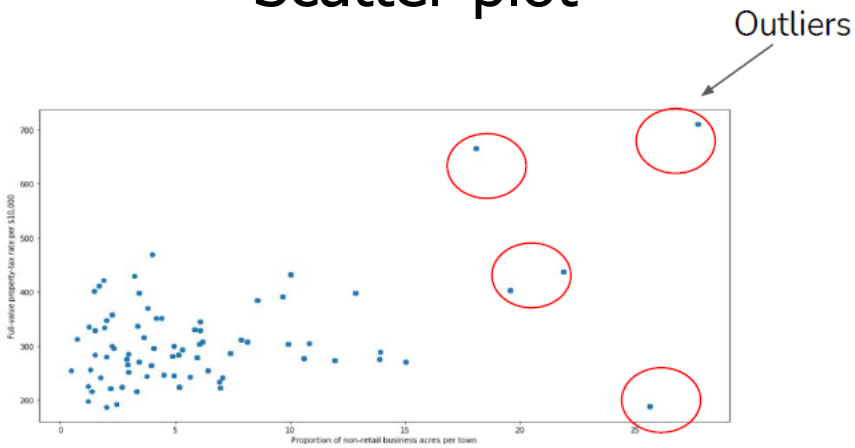

```python
def remove_outliers(df, out_cols, T=1.5):
    # Copy of df
    new_df = df.copy()
    init_shape = new_df.shape
    # For each column
    for c in out_cols:
        q1 = new_df[c].quantile(.25)
        q3 = new_df[c].quantile(.75)
        col_iqr = q3 - q1
        col_max = q3 + T * col_iqr
        col_min = q1 - T * col_iqr
        # Filter data without outliers and ignoring nan
        filtered_df = new_df[(new_df[c] <= col_max) & (new_df[c] >= col_min)]
    return new_df
```

Outliers - Visualization

Box plot



Scatter plot



Box plot shows:

- The minimum
- Q1 (the first quartile, or the 25th percentile)
- The median (the second quartile, or the 50th percentile)
- Q3 (the third quartile, or the 75th percentile)
- The maximum
- Outliers: larger than Q3 by at least 1.5 times the interquartile range (IQR), or
- smaller than Q1 by at least 1.5 times the IQR.
- IQR=Q3-Q1

How to deal with outliers?

- Not a unique way of dealing with outliers. But some common practices are:
  - Analyze outliers before treating them.
  - If an outlier represents the general trend then there is no need to treat it
    - Example - Income is generally a skewed variable but all extreme points might not be outliers
  - If we decide to treat the outlier after analyzing it, then:
    - we can drop them but we would lose information in other columns of the data
    - we can cap outliers at certain values, say 5th percentile or 95th percentile
    - We can set a threshold using Interquartile Range (IQR) and remove the outliers greater than that threshold value

Pandas Key Operations

Operation	Pandas Function
Load or import the data from different sources/formats	read_csv(), read_excel(), read_html(), read_json()
Information about the data - dimension, column dtypes, non-null values and memory usage	info()
View of basic statistical details of numeric data - quartiles, min, max, mean, std	describe()
Merge two data frames with different types of join - inner join, left join, right join, and full outer join	merge()
Explore data frames by different groups, and apply summary functions on each group	groupby()

Describe - Recap

```
In [133]: by_comp.describe()
Out[133]:
        Sales
        count   mean         std    min     25%    50%     75%    max
Company
FB        2.0  296.5   75.660426  243.0  269.75  296.5  323.25  350.0
GOOG      2.0  160.0   56.568542  120.0  140.00  160.0  180.00  200.0
MSFT      2.0  232.0  152.735065  124.0  178.00  232.0  286.00  340.0
```

```
In [134]: by_comp.describe().transpose()
Out[134]:
Company              FB        GOOG        MSFT
Sales count    2.000000    2.000000    2.000000
      mean   296.500000  160.000000  232.000000
      std     75.660426   56.568542  152.735065
      min    243.000000  120.000000  124.000000
      25%    269.750000  140.000000  178.000000
      50%    296.500000  160.000000  232.000000
      75%    323.250000  180.000000  286.000000
      max    350.000000  200.000000  340.000000
```

```
In [142]: by_comp.describe().transpose()['GOOG']
Out[142]:
Sales  count      2.000000
       mean     160.000000
       std       56.568542
       min      120.000000
       25%      140.000000
       50%      160.000000
       75%      180.000000
       max      200.000000
Name: GOOG, dtype: float64
```

Introduction to Visualization

- What is Visualization?
  - Visual Representation of data
  - Helps to observe & communicate patterns and trends with naked eye
- Why is data visualization important?
  - Data visualization helps to communicate information in a manner that is universal, fast, and effective
  - Communicating insights to non-technical decision makers is one of the most critical phases in an analytics project

Common Libraries for Visualization

Matplotlib

- Matplotlib is one of the most popular libraries for data visualizations.
- It provides high-quality graphics and a variety of plots such as histograms, bar charts, pie charts, etc.
- Some important functions - **plot(), hist(), bar(), pie(), scatter()**, text(), legend(), etc.

Seaborn

- Seaborn is complementary to Matplotlib and it specifically targets statistical data visualizations.
- A saying around matplotlib and seaborn is, “matplotlib tries to make easy things easy and hard things possible, seaborn tries to make a well-defined set of hard things easy too.”
- Some important functions - **displot(), boxplot(), stripplot(), pairplot()**

Visualization Guidelines

- Which visualization to use?
  - Choosing right visualization for right purpose is very important
  - Numerous types of plots available in Matplotlib and Seaborn

Visualization Guidelines

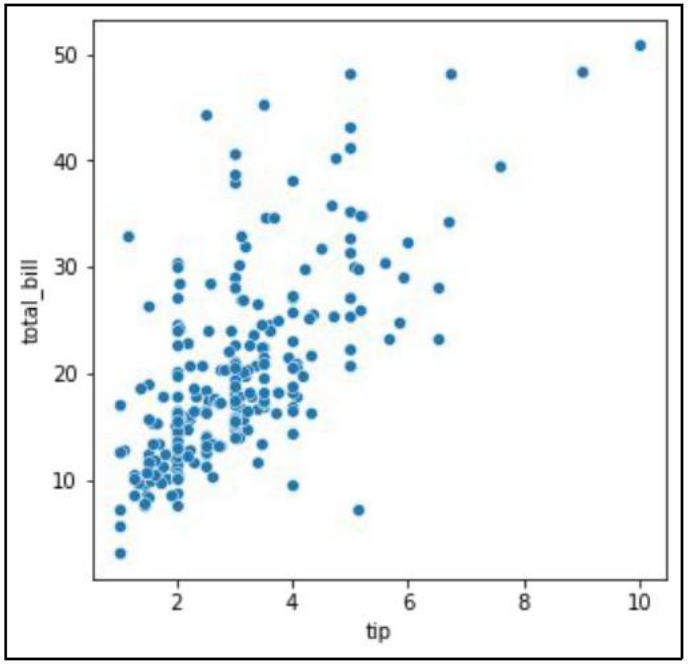
Type	X Variable	Y Variable	Purpose of analysis	Type of chart
Univariate	Continuous	-	How the values of the X variable are distributed?	Histogram, Distribution plot
Univariate	Categorical	-	What is the count of observations in each category of X variable?	Count Plot
Bivariate	Continuous	Continuous	How Y is correlated with X?	Scatter plot
Bivariate	Time Related (months, hours, etc.)	Continuous	How Y changes over time?	Line Plot
Bivariate	Continuous	Categorical	How range of X varies for various category levels?	Box plot, Swarm Plot
Bivariate	Categorical	Categorical	What is the number or % of records of X which falls under each category of Y?	Stacked Bar plot

Visualization Guidelines

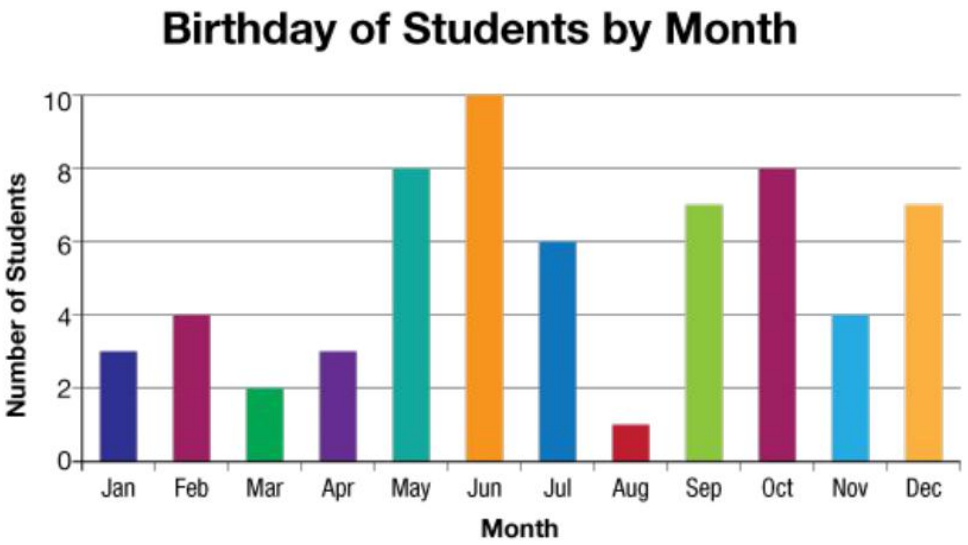
Type	Variables	Purpose of analysis	Type of chart
Multivariate	Continuous (more than two)	How to visualize relationship across multiple combination of variables?	Pair Plot
Multivariate	Continuous (more than two)	How to visualize the spread of values in the data with color-encoding?	Heatmap

Examples of Visualizations

Scatter Plot

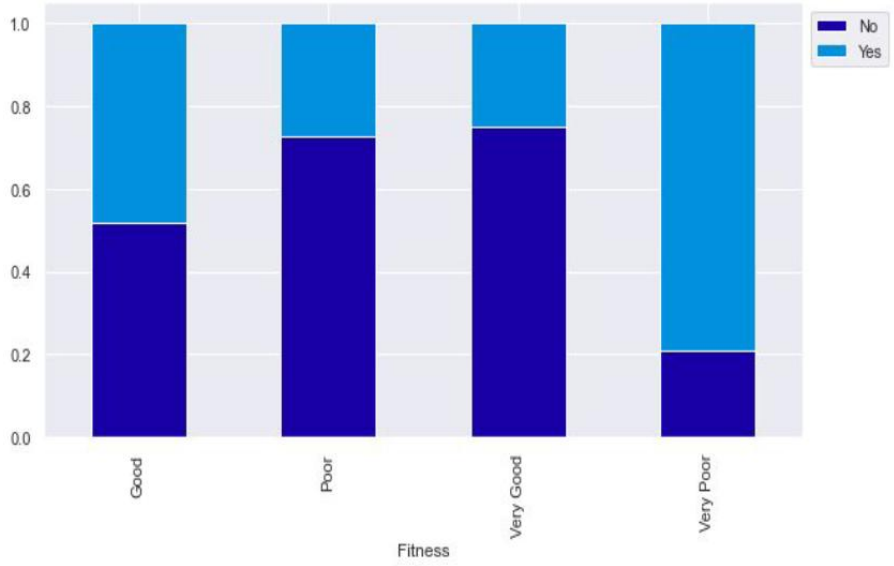


Bar Plot

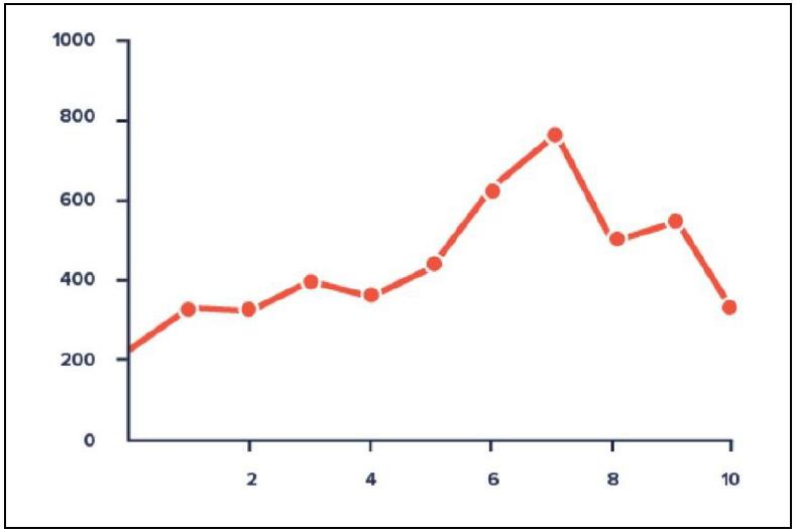


Examples of Visualizations

Stacked Bar Plot

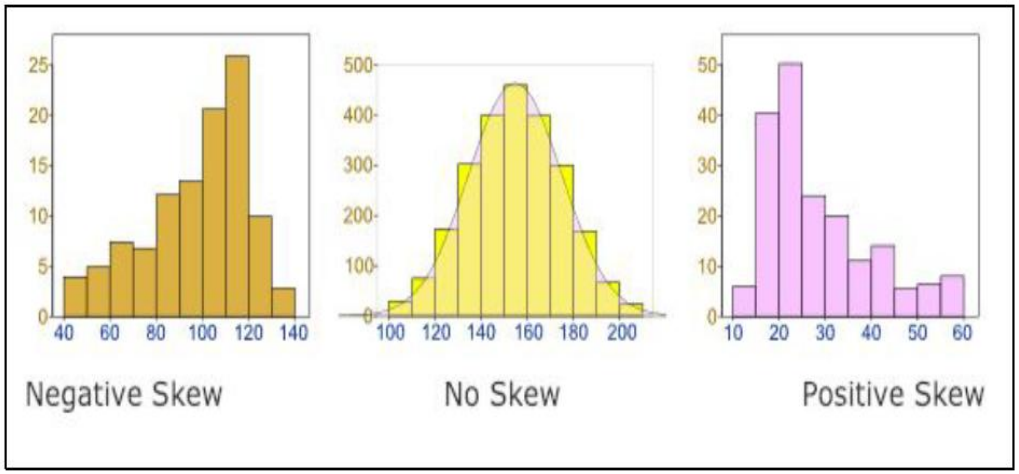


Line Plot

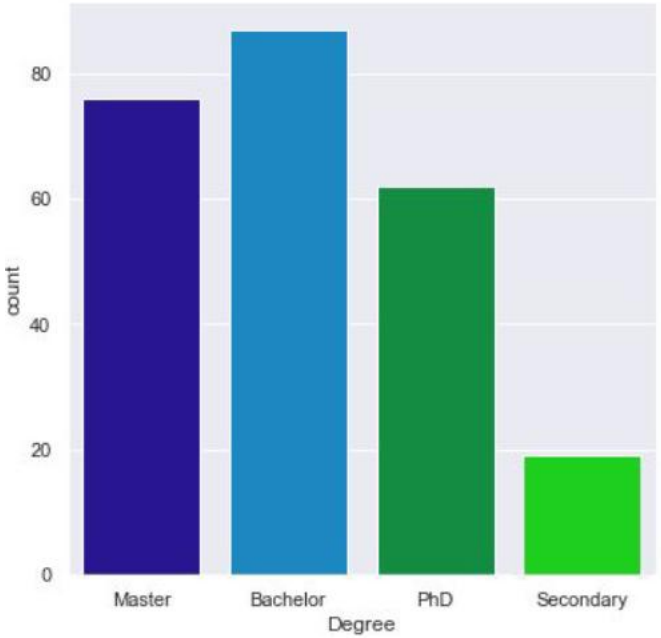


Examples of Visualizations

Histogram and Skewness

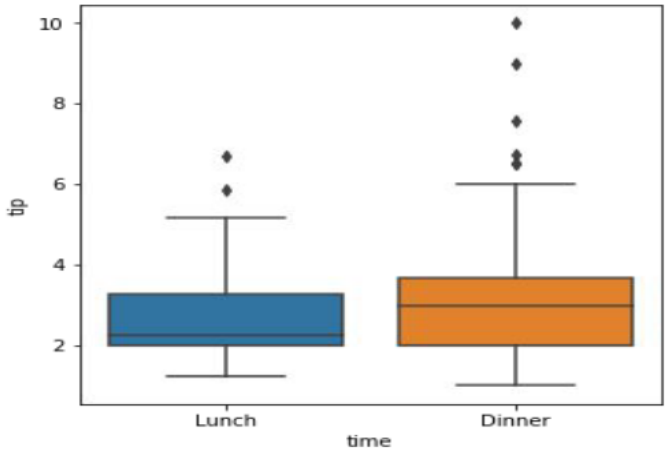


Count Plot

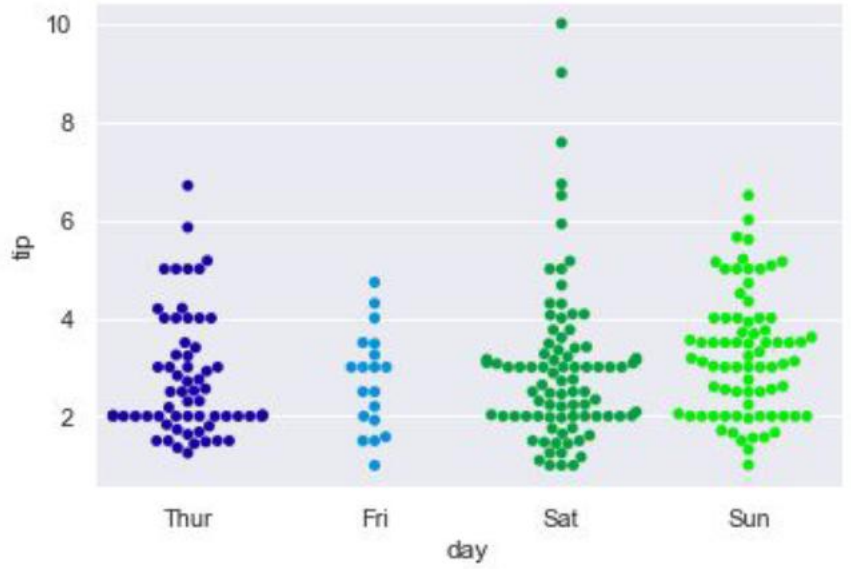


Examples of Visualizations

Box Plot

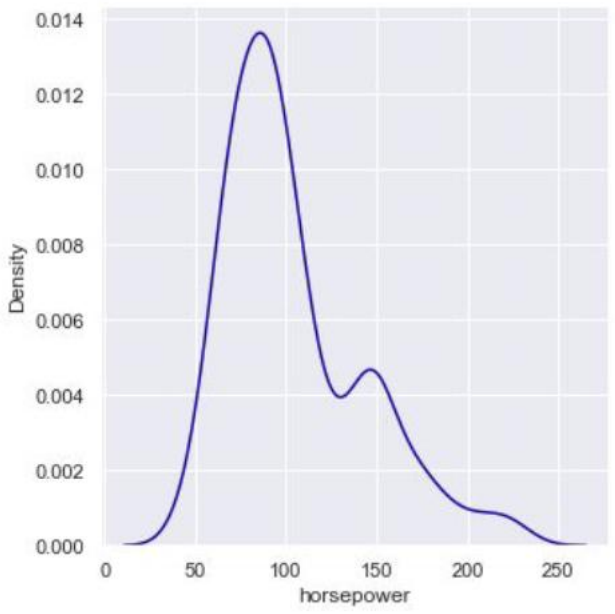


Swarm Plot

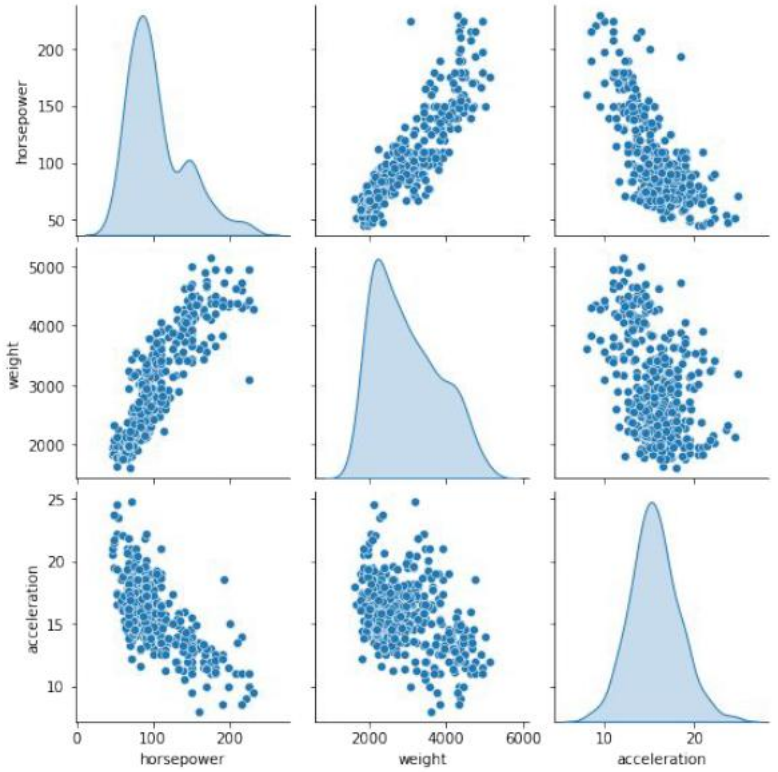


Examples of Visualizations

Distribution Plot

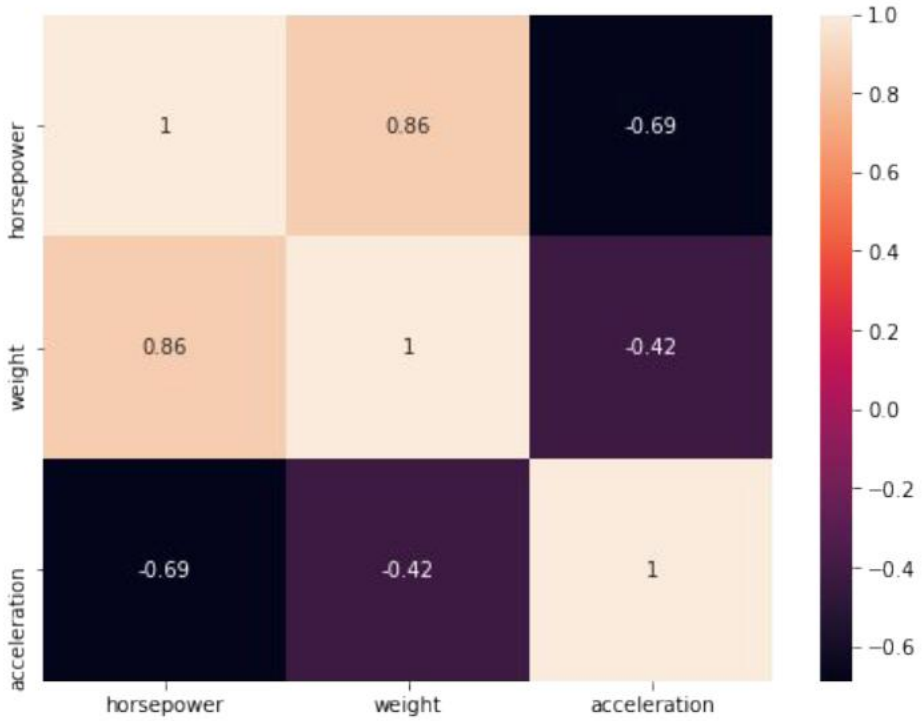


Pair Plot



Examples of Visualizations

Heat Map



This heatmap shows the correlation coefficient between three variables