Statistical Analysis and Visualization Python Project

By: Samriddhi Matharu

Link to my full project code:
https://github.com/samriddhi-m1227/Statistical-Analysis-and-Visualization-Python/blob/main/StatsAnalysis_Visualization_PythonProject.ipynb

Table of contents

01 Generating the Dataset

	Age	Height	Weight	Income	Gender
0	32	171.876445	70.121012	43431.051334	Male
1	27	136.206968	68.500429	70294.139595	Male
2	34	161.049736	73.275950	41282.356885	Male
3	42	172.526055	72.747140	51454.489301	Male
4	33	164.300733	80.754174	55269.000561	Male
..	...	...	...	...	...
995	42	176.340219	52.594902	58370.222614	Male
996	21	153.190109	78.118743	48008.687082	Male
997	27	176.487122	81.766551	81376.252914	Male
998	25	183.983726	79.667585	42375.692310	Female
999	40	173.482688	74.407724	37815.671741	Male

[1000 rows x 5 columns]

How did I generate the synthetic Dataset ?

```
#import all libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

<u>Utilized NumPy to create the random Data.</u>

1) I first imported all the required libraries for this project:
2) Made note of the conditions
   - Sample size of 1000. **ex) sample=1000**
   - The dataset should include the following columns: Age, Height, Weight, Income, and Gender
   - Specific mean and standard deviations were given for each variable
3) For the numerical values, I used **np.random.normal=(mean, std, sample)**
4) For Gender, I used **np.random.choice([’Male’, ’Female’], sample)**

Data Generation

```
#Generate a synthetic Dataset:

#sample size
sample=1000

#Use NumPy to create columns present in the dataset that are normally distributed
age= np.random.normal(35, 10, sample)
height= np.random.normal(170, 15, sample)   #in cm
weight= np.random.normal(70, 10, sample)    #in kg
income= np.random.normal(50000, 15000, sample)
gender= np.random.choice(['Male', 'Female'], sample)

#Round of the Age values and convert them to integers
age=np.round(age).astype(int)
```

<u>Utilized Pandas to make the DataFrame</u>

1) Using **data= pd.DataFrame(..)**

```
#Make the DataFrame using pandas
data= pd.DataFrame({'Age':age, 'Height':height, 'Weight': weight,'Income': income,
'Gender':gender,})

print(data)
```

I used a normal distribution because many real-world categories such as age, height, weight, and income, often follow a normal distribution due to the Central Limit Theorem. This distribution allows us to generate synthetic data that mimics natural variability observed in real populations, making the dataset more realistic for statistical analysis.

02

Descriptive Statistics

	Age	Height	Weight	Income	Gender
0	32	171.876445	70.121012	43431.051334	Male
1	27	136.206968	68.500429	70294.139595	Male
2	34	161.049736	73.275950	41282.356885	Male
3	42	172.526055	72.747140	51454.489301	Male
4	33	164.300733	80.754174	55269.000561	Male
..	...	...	...	...	...
995	42	176.340219	52.594902	58370.222614	Male
996	21	153.190109	78.118743	48008.687082	Male
997	27	176.487122	81.766551	81376.252914	Male
998	25	183.983726	79.667585	42375.692310	Female
999	40	173.482688	74.407724	37815.671741	Male

[1000 rows x 5 columns]

Descriptive Statistics Result

```
#Measure of Central Tendency

#Mean Values
mean_values=data[['Age', 'Height', 'Weight', 'Income']].mean()
print("\nMean: ")
print(mean_values)


#Median values
median_values=data[['Age', 'Height', 'Weight', 'Income']].median()
print("\nMedian: ")
print(median_values)
```

```
Mean:
Age          35.603000
Height      170.420044
Weight       69.824209
Income    50166.844531
dtype: float64

Median:
Age          36.000000
Height      171.615813
Weight       70.088983
Income    50444.651066
dtype: float64
```

```
#Measure of dispersion

#Standard Deviation values
std_values=data[['Age', 'Height', 'Weight', 'Income']].std()
print("\nStandard Deviation: ")
print(std_values)

#Variance values
var_values=data[['Age', 'Height', 'Weight', 'Income']].var()
print("\nVariance: ")
print(var_values)
```

```
Standard Deviation:
Age           9.996365
Height       14.176739
Weight        9.701166
Income    14694.192726
dtype: float64

Variance:
Age       9.992732e+01
Height    2.009799e+02
Weight    9.411263e+01
Income    2.159193e+08
dtype: float64
```

Here you can see that I used various functions on 'data' such as .mean() .median() std() and .var() only on the numerical columns, then I used print() to display the results

Age Analysis

Descriptive Statistics

- Mean Age: 35.60 years
- Median Age: 36 years
- Standard Deviation: 9.996 years
- Variance: 99.93 years^2

Interpretation

The age distribution in this dataset is centered around 35.60 years--so most people in this dataset are around this age. With most individuals falling within a 10-year range of this mean (between approximately 25.60 and 45.60 years). The close alignment of the mean and median indicates a symmetric distribution, while the standard deviation and variance values suggest a moderate level of variability. Overall, the age data appears to be normally distributed.

Height Analysis

Descriptive Statistics

- Mean Height: 170.42 cm
- Median Height: 171.62 cm
- Standard Deviation: 14.18 cm
- Variance: 200.98 cm^2

Interpretation

The height distribution in this dataset is centered around 170.42 cm, with most individuals falling within a 14 cm range of this mean (between approximately 156.24 and 184.60 cm). Since the median is slightly larger than the mean, this implies that there are a few shorter values that pulls the average down a bit causing a slight right skew. The standard deviation and variance values suggest a moderate level of variability. Overall, the height data appears to be normally distributed with minimal skewness.

Weight Analysis

Descriptive Statistics

- Mean Weight: 69.82 kg
- Median Weight: 70.09 kg
- Standard Deviation: 9.70 kg
- Variance: 94.11 kg^2

Interpretation

The weight distribution in this dataset is centered around 69.82 kg–so most people in the dataset are around this weight, with most individuals falling within a 9.70 kg range of this mean (between approximately 60.12 and 79.52 kg). The close alignment of the mean and median indicates a symmetric distribution. The standard deviation and variance values suggest a moderate level of variability. Overall, the weight data appears to be normally distributed.

Income Analysis

Descriptive Statistics

- Mean Income: $50,166.84
- Median Income: $50,444.65
- Standard Deviation: $14,694.19
- Variance: $215,919,300.00

Interpretation

The income distribution in this dataset is centered around $50,166.84, with most individuals falling within a $14,694.19 range of this mean (between approximately $35,472.65 and $64,861.03). The close alignment of the mean and median indicates a symmetric distribution, while the standard deviation is moderate, the variance is a very large number which means there is high variability in the data. Overall, the income data appears to be normally distributed with considerable variability..

Mode for Gender

```
#Calculate the Mode for Gender
#Mode: Most frequently occurred value in a dataset

mode_value=data[['Gender']].mode()
print("\nMode: ")
print(mode_value)
```

```
Mode:
  Gender
0   Male
```

The Mode calculates the most frequently occurred value in a dataset. In this case for gender, the most frequently occurred gender in the dataset were Males.

03

Histogram and KDE plots

```
     Age      Height     Weight        Income  Gender
0     32  171.876445  70.121012  43431.051334    Male
1     27  136.206968  68.500429  70294.139595    Male
2     34  161.049736  73.275950  41282.356885    Male
3     42  172.526055  72.747140  51454.489301    Male
4     33  164.300733  80.754174  55269.000561    Male
..   ...         ...        ...           ...     ...
995   42  176.340219  52.594902  58370.222614    Male
996   21  153.190109  78.118743  48008.687082    Male
997   27  176.487122  81.766551  81376.252914    Male
998   25  183.983726  79.667585  42375.692310  Female
999   40  173.482688  74.407724  37815.671741    Male

[1000 rows x 5 columns]
```

Histograms

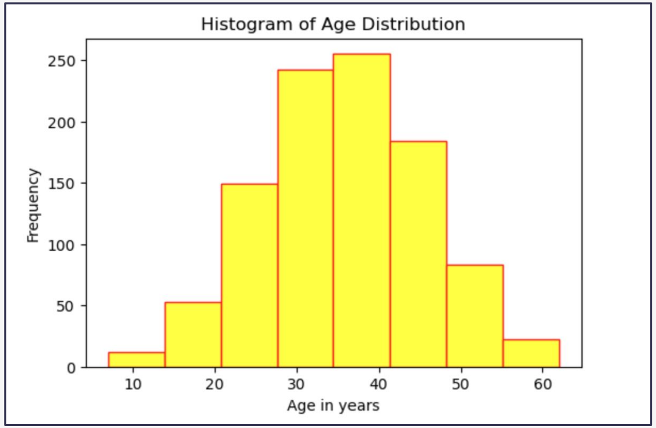
How do you visualize data using a Histogram?

Code example on Histogram for Age :

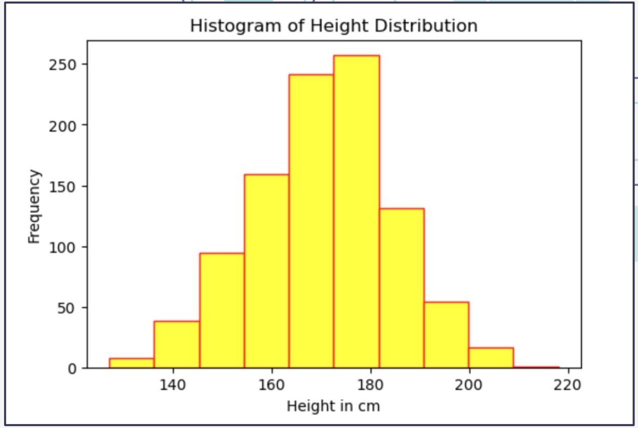
```
# Graph Analysis: Histogram

#Histogram for Age:
plt.figure(figsize=(6,4))
plt.hist(data['Age'], bins=8, color="yellow", edgecolor="red")
plt.xlabel("Age in years")
plt.ylabel("Frequency")
plt.title("Histogram of Age Distribution")
plt.show()
```

- **plt.hist()** takes in parameters such as the Data Column, number of bins you want, color of bins, etc..
- Then name your x-axis, y-axis, and title, and use **plt.show()** to display your Histogram!

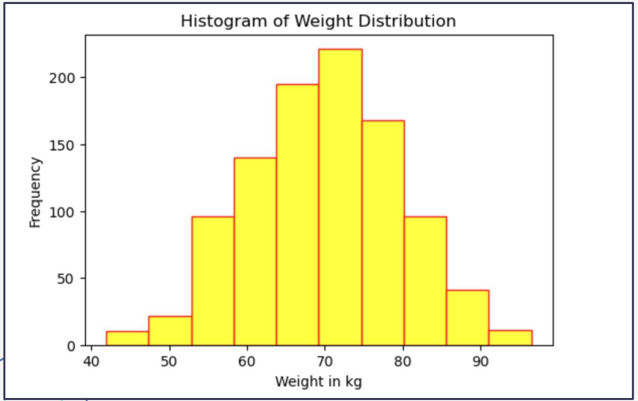
Histograms



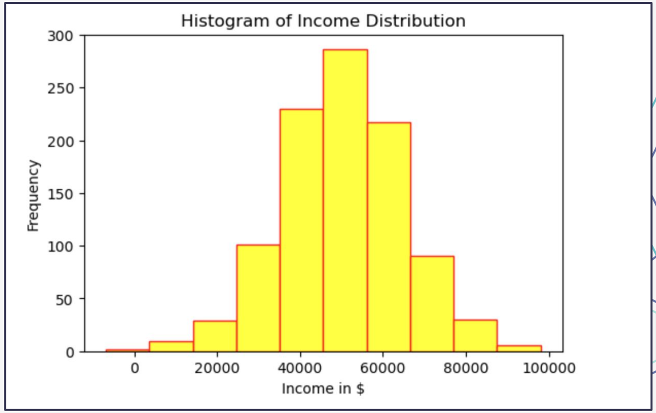
Normally distributed



Normal distribution with slight right skew



Normally distributed



Normally distributed, some variability

KDE Plots

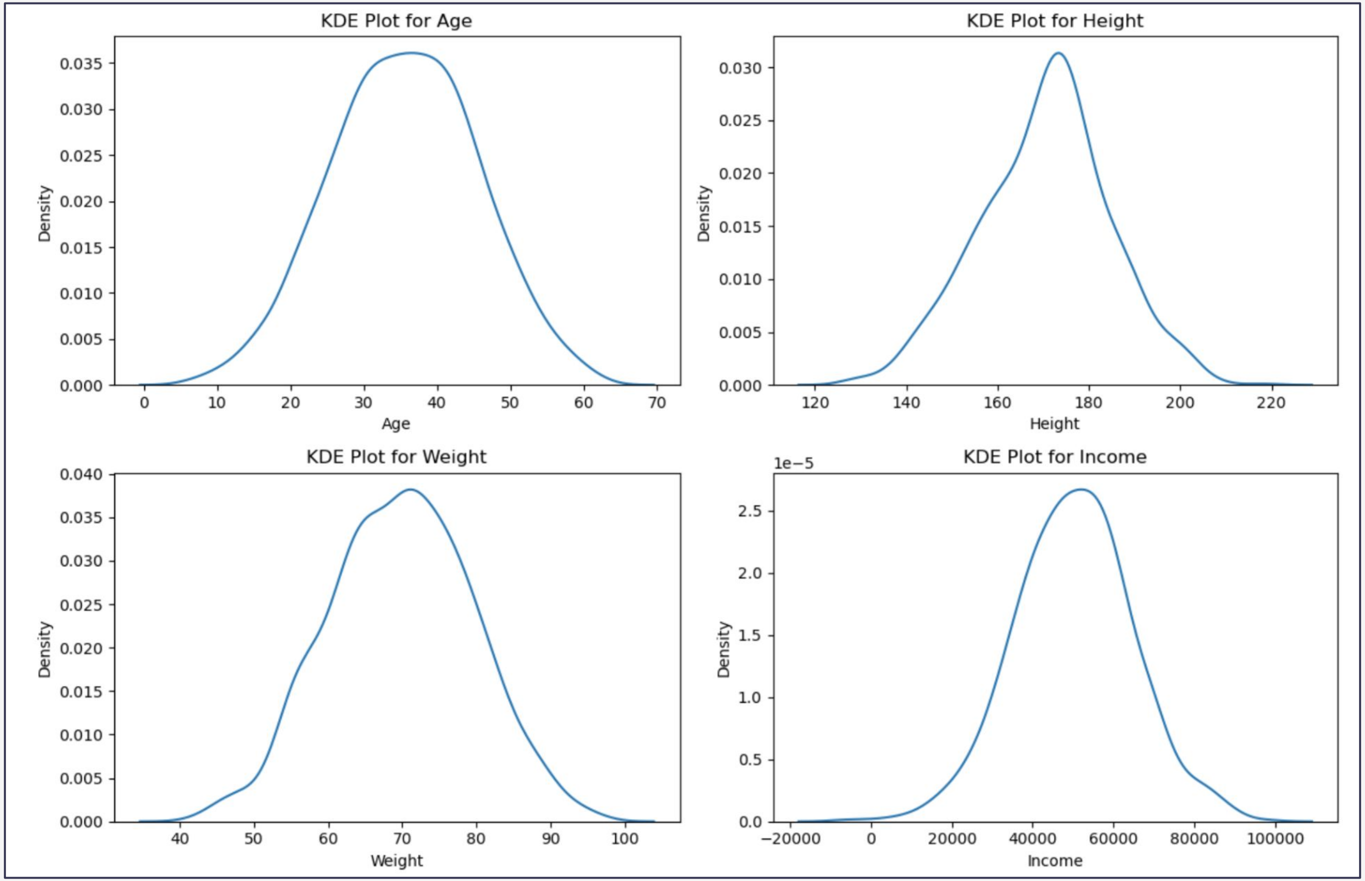
How do you visualize data using KDE Plots?

Code example on KDE plots

```
#KDE plots
plt.figure(figsize=(12,8))

#KDE plot for Age:
plt.subplot(2, 2, 1)
sns.kdeplot(data['Age'])
plt.title('KDE Plot for Age')


# KDE plot for Height
plt.subplot(2, 2, 2)
sns.kdeplot(data['Height'])
plt.title('KDE Plot for Height')


# KDE plot for Weight
plt.subplot(2, 2, 3)
sns.kdeplot(data['Weight'])
plt.title('KDE Plot for Weight')


# KDE plot for Income
plt.subplot(2, 2, 4)
sns.kdeplot(data['Income'])
plt.title('KDE Plot for Income')


# Adjust layout
plt.tight_layout()
plt.show()
```

- We use seaborn as sns for KDE plots
- I used one big figure size to display all my KDE plots using **.subplot()** with specific sizes.
- **sns.kdeplot(data[" "])** specifies which Column to visualize
- plt.title() is used to name all the subplots
- **plt.tight_layout()** is used to make sure all subplots are In dimension of each other and properly formatted
- **plt.show()** to display the plots!

KDE Plots


KDE Plot for Age
Density
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
10
20
30
40
50
60
70
Age
KDE Plot for Height
Density
0.030
0.025
0.020
0.015
0.010
0.005
0.000
120
140
160
180
200
220
Height
KDE Plot for Weight
Density
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
40
50
60
70
80
90
100
Weight
KDE Plot for Income
1e−5
Density
2.5
2.0
1.5
1.0
0.5
0.0
−20000
0
20000
40000
60000
80000
100000
Income

KDE Plots

- **Age**
  Most symmetric of all, and bell-shaped. The distribution closely resembles a normal distribution.
- **Height**
  More skinny and pointed tip with a narrower distribution of height values. The peak of the distribution is sharper, suggesting a specific mode for heights.
- **Weight**
  More rigid left side and smooth bell on the right makes an asymmetric distribution but is still bell shaped. This causes for some fluctuation and maybe fewer data points on the left side but is still mostly normally distributed.
- **Income**
  Slightly more narrower bell shape on the left side and a wider one on the right side makes an asymmetric distribution. This is because of the variability in the data; Overall, mostly normally distributed

04

Box Plots

```
     Age      Height     Weight        Income  Gender
0     32  171.876445  70.121012  43431.051334    Male
1     27  136.206968  68.500429  70294.139595    Male
2     34  161.049736  73.275950  41282.356885    Male
3     42  172.526055  72.747140  51454.489301    Male
4     33  164.300733  80.754174  55269.000561    Male
..   ...         ...        ...           ...     ...
995   42  176.340219  52.594902  58370.222614    Male
996   21  153.190109  78.118743  48008.687082    Male
997   27  176.487122  81.766551  81376.252914    Male
998   25  183.983726  79.667585  42375.692310  Female
999   40  173.482688  74.407724  37815.671741    Male

[1000 rows x 5 columns]
```

Box Plots

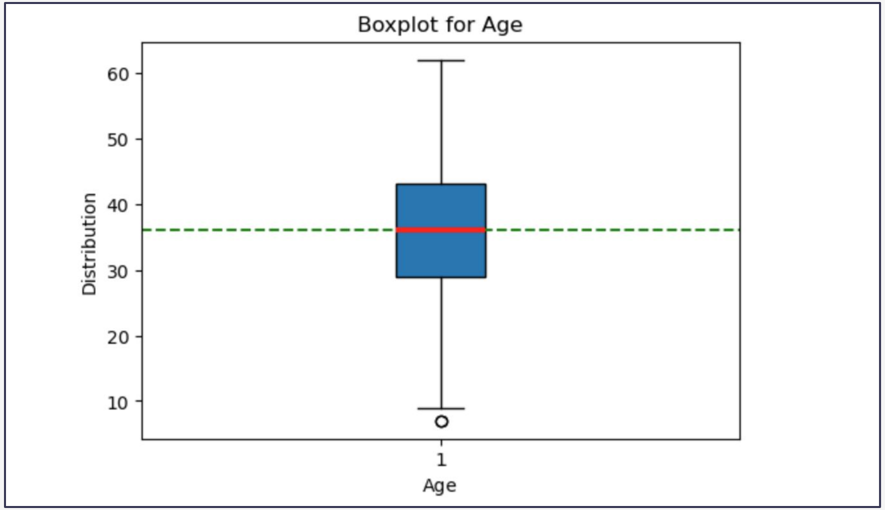
How do you visualize data using Box Plots

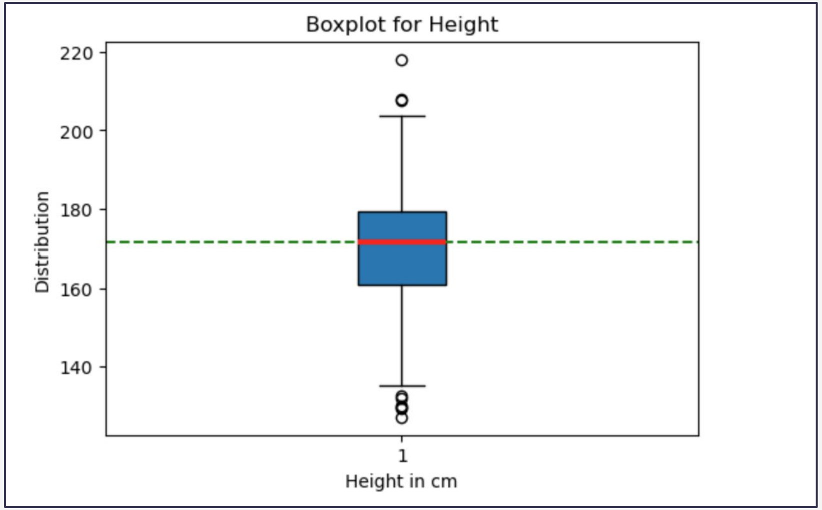
Code example on Box Plots:

```
#Boxplots to identify outliers

#Boxplot for Age:
plt.figure(figsize=(6,4))
plt.boxplot(data['Age'],patch_artist=True,  # Fill the boxplot with color
            medianprops=dict(color='red', linewidth=3))  # Customize median line

# Add a horizontal line to indicate the median value
median_value = data['Age'].median()
plt.axhline(y=median_value, color='green', linestyle='--', label=f'Median: {median_value}')


plt.xlabel('Age')
plt.ylabel('Distribution')
plt.title('Boxplot for Age')

plt.show()
```

- **plt.boxplot()** takes in parameters such as the data column, a color to fill in and more..
- I added more things such as a line at the median for a more visually appealing box plot

Box Plot Analysis

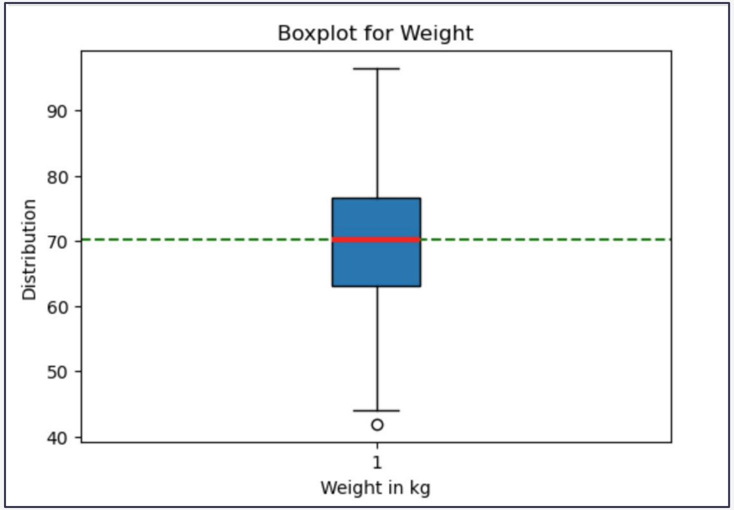


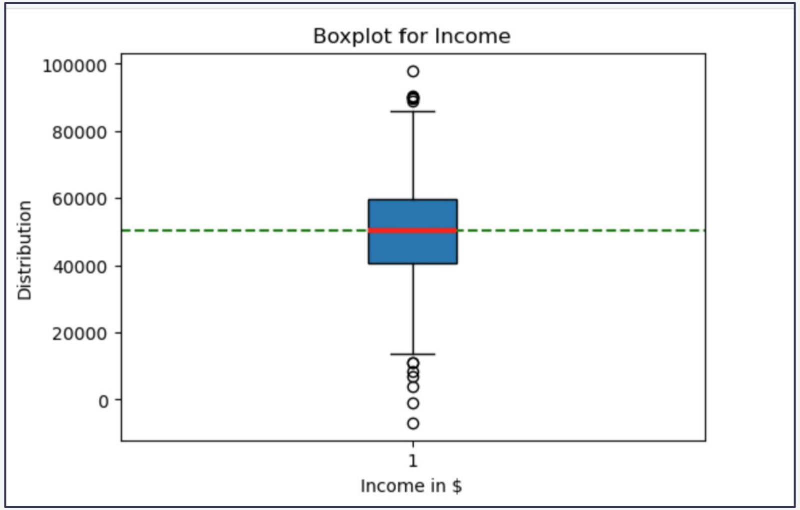


- I can interpret this Box Plot by noticing there is an even distribution on both sides of median. The median for Age is around 36 yrs old and that there is one outlier, who is a person that is younger than 10 yrs old.

I can interpret this Box Plot by noticing that there are more data points below the median which means more people have a height under the median. The median for Height is a little over 170 cm and there are multiple high and low outliers of people who are over approx 200 cm and under approx 130 cm

Box Plots Analysis





- I can interpret this Box Plot by noticing that median is about 70 kg and the data is evenly distributed on either side of the median. There seems to be low outlier in the dataset, which is someone that weighs a little under 45 kg

I can interpret this Box Plot by noticing that the median is about $50,000 in income for people in the dataset. There are a lot of outliers which is why this data is so variant. There are a couple high outliers of people who make over $80,000 and low outliers of people who make under approx $10,000

05

Correlation Matrix

```
      Age      Height     Weight        Income  Gender
0      32  171.876445  70.121012  43431.051334    Male
1      27  136.206968  68.500429  70294.139595    Male
2      34  161.049736  73.275950  41282.356885    Male
3      42  172.526055  72.747140  51454.489301    Male
4      33  164.300733  80.754174  55269.000561    Male
..    ...         ...        ...           ...     ...
995    42  176.340219  52.594902  58370.222614    Male
996    21  153.190109  78.118743  48008.687082    Male
997    27  176.487122  81.766551  81376.252914    Male
998    25  183.983726  79.667585  42375.692310  Female
999    40  173.482688  74.407724  37815.671741    Male

[1000 rows x 5 columns]
```

Correlation Matrix Analysis

How do you generate a Correlation Matrix ?

- We can declare a variable ex) correlation= data[['Age', 'Height', 'Weight', 'Income']].**corr()** to generate a correlation Matrix between these columns
- Then display it using **print(correlation)**

```
#Calculate the Pearson correlation coefficient

#r = 0: There is no correlation
#r = 1: There is a perfect positive correlation
#r = -1: There is a perfect negative correlation

correlation= data[['Age', 'Height', 'Weight', 'Income']].corr()
print("Correlation between Columns in DataFrame: ")
print(correlation)
```

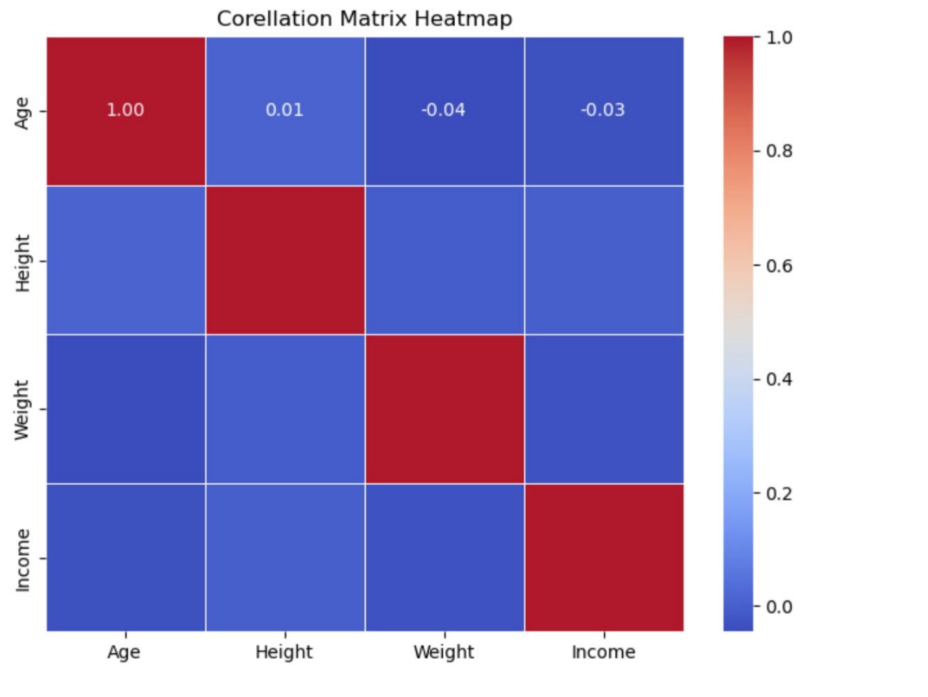
```
Correlation between Columns in DataFrame:
             Age    Height    Weight    Income
Age     1.000000  0.008627 -0.044783 -0.033634
Height  0.008627  1.000000 -0.005863 -0.003785
Weight -0.044783 -0.005863  1.000000 -0.029931
Income -0.033634 -0.003785 -0.029931  1.000000
```

Analysis:

The correlation matrix reveals weak or positive linear relationships between age, height, weight, and income in the dataset. Age shows minimal correlations with other variables, this indicates that changes in age does not significantly predict changes in height, weight, or income. Similarly, height exhibits weak correlations with the other variables, suggesting little to no linear relationship with age, weight, or income. Weight displays slight inverse correlations with age and income, while income shows very weak correlations with age, weight, and height. Overall, the correlation matrix implies that changes in one variable is not strongly predictive of changes in another variable; this suggests the independence of age, height, weight, and income within the dataset.

Visualize the Correlation Matrix

```python
#Visualize the correlation matrix, using a heatmap

plt.figure(figsize=(8,6))
sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)

plt.title("Corellation Matrix Heatmap")
plt.show()
```



06 Inferential Statistics

	Age	Height	Weight	Income	Gender
0	32	171.876445	70.121012	43431.051334	Male
1	27	136.206968	68.500429	70294.139595	Male
2	34	161.049736	73.275950	41282.356885	Male
3	42	172.526055	72.747140	51454.489301	Male
4	33	164.300733	80.754174	55269.000561	Male
..	...	...	...	...	...
995	42	176.340219	52.594902	58370.222614	Male
996	21	153.190109	78.118743	48008.687082	Male
997	27	176.487122	81.766551	81376.252914	Male
998	25	183.983726	79.667585	42375.692310	Female
999	40	173.482688	74.407724	37815.671741	Male

[1000 rows x 5 columns]

Inferential Statistics

- To conduct the Inferential Statistic, we need to first install **scipy** and import the following things:

```
!pip install scipy

from scipy.stats import ttest_ind, chi2_contingency, f_oneway
```

- Now for our T-Test we need to establish our Null and Alternative Hypothesis.

T-Tests

- Ho: There is no significant difference in Income between Male and Female
- H1: There is a significant difference between the Income for Male and Female

- To do this, we create variables for Male income and Female income, since we are comparing those two:

```
#Income for Male and Female
income_male=data[data['Gender']=='Male']['Income']
income_female=data[data['Gender']=='Female']['Income']

#Perform Independent Sample T-Test
t_stats =ttest_ind(income_male, income_female)
print(t_stats)
```

```
TtestResult(statistic=0.8823076299404343, pvalue=0.377822930683613, df=998.0)
```

T-Test Analysis

```
#Income for Male and Female
income_male=data[data['Gender']=='Male']['Income']
income_female=data[data['Gender']=='Female']['Income']

#Perform Independent Sample T-Test
t_stats =ttest_ind(income_male, income_female)
print(t_stats)
```

```
TtestResult(statistic=0.8823076299404343, pvalue=0.3778229300683613, df=998.0)
```

- The T-test shows that the <u>p-value is greater than 0.05</u> which suggests that our findings are not statistically significant and we do not have much reason to reject the null--therefore we can accept the the null hypothesis. Concluding that there is really no significant difference in Income between Male and Female
- The <u>statistic value is a positive number;</u> this suggests that on average the Male income exceeds the Female income. However, this difference is not statistically significant, as indicated by the p-value. Therefore we don't have substantial data to draw a meaningful conclusions.

Thank you

This concludes the Statistical Analysis and Visualization Python Project

Link to my full project code:
https://github.com/samriddhi-m1227/Statistical-Analysis-and-Visualization-Python/blob/main/StatsAnalysis_Visualization_PythonProject.ipynb