INFO 7390 Syllabus Week 1 (Topic: Basic Data Statistics and Exploratory Data Analysis)

Kshitij Prabhu

NEU ID: 002769231

Email: prabhu.ks@northeastern.edu

Basic Data Statistics

Introduction

Basic data statistics involve analyzing and summarizing data to gain insights into its characteristics. Python offers several libraries, such as NumPy and pandas, that provide tools for performing statistical analysis on datasets.

```
import numpy as np
import pandas as pd
#importing all the required libraries
```

```
data = np.array([2, 2, 4, 6, 8, 10]) #creating a numpy array using np.array()
data_dict = {'values': [2, 4, 6, 8, 10]}
df = pd.DataFrame(data_dict) #creating a dataframe
```

Mean, Median, and Mode

When analyzing data, it is important to understand the central tendency of the data. The three measures of central tendency are the mean, median, and mode.

The mean is the average of all the values in the data set. To calculate the mean in Python, we can use the `mean()` function from the NumPy library.

```
#data = np.array([2, 2, 4, 6, 8, 10])
mean = np.mean(data)#using the mean() function on data array
# the output we get shoulf be 5.33
```

The median is the middle value in the data set when the values are arranged in order. To calculate the median in Python, we can use the `median()` function from the NumPy library.

```
#data = np.array([2, 2, 4, 6, 8, 10])
median = np.median(data)#using the median() function on data array
# the output we get should be 5.0
```

The **mode** is the value that occurs most frequently in the data set. To calculate the mode in Python, we can use the `bincount()` function and `argmax()` from the NumPy library.

```
#data = np.array([2, 2, 4, 6, 8, 10])
mode = np.bincount(data).argmax()
# the output we get should be 2.0
```

Standard Deviation and Variance

The standard deviation and variance are measures of the spread of the data. The standard deviation is the square root of the variance.

The variance is the average of the squared differences from the mean. To calculate the variance in Python, we can use the `var()` function from the NumPy library.

```
#data = np.array([2, 2, 4, 6, 8, 10])
variance = np.var(data)
# the output we get should be 8.88
```

The **standard deviation** is the square root of the variance. To calculate the standard deviation in Python, we can use the `std()` function from the NumPy library.

```
#data = np.array([2, 2, 4, 6, 8, 10])
std_dev = np.std(data)
# the output we get should be 2.98
```

Correlation and Covariance

Correlation is a measure of the relationship between two variables. A correlation of 1 indicates a perfect positive relationship, a correlation of -1 indicates a perfect negative relationship, and a correlation of 0 indicates no relationship.

To calculate the correlation in Python, we can use the `corr()` function from the Pandas library.

```
#using the dataframe that was previously created
correlation_matrix = df.corr()
```

Covariance is a statistical measure that quantifies the degree to which two random variables change together. In other words, it indicates whether the values of two variables tend to increase or decrease together or move in opposite directions. Covariance helps us understand the relationship and pattern of variability between two variables.

To calculate the correlation in Python, we can use the `cov()` function from the Pandas library.

```
#using the dataframe that was previously created
covariance_matrix = df.cov()
```

Exploratory Data Analysis(EDA)

Introduction

Exploratory Data Analysis (EDA) is a crucial step in the data analysis process that involves visually and quantitatively summarizing, understanding, and interpreting the main features of a dataset.

EDA helps data analysts and scientists to uncover patterns, relationships, anomalies, and trends within the data. Python, with its rich ecosystem of libraries, is a popular choice for performing EDA effectively.

Click on this to download the dataset : Dataset

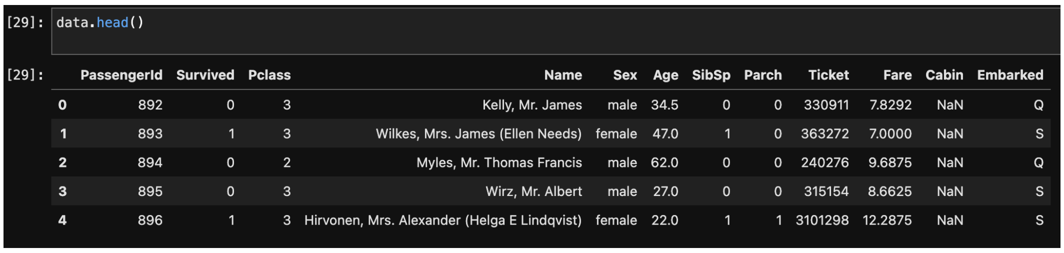
Importing Data

The first step in EDA is loading your dataset. Popular libraries for data manipulation include pandas, NumPy, and CSV.

```
import pandas as pd

data = pd.read_csv('tested.csv')
```

Data Summary

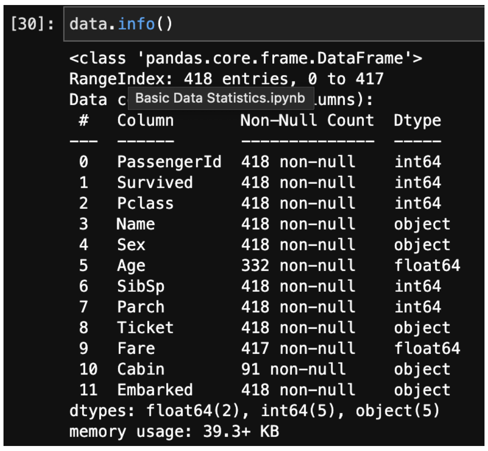
Understanding the basic characteristics of your data is essential. Use functions like `.head()`, `.info()`, `.describe()` to get a sense of the data's structure, types, and summary statistics.



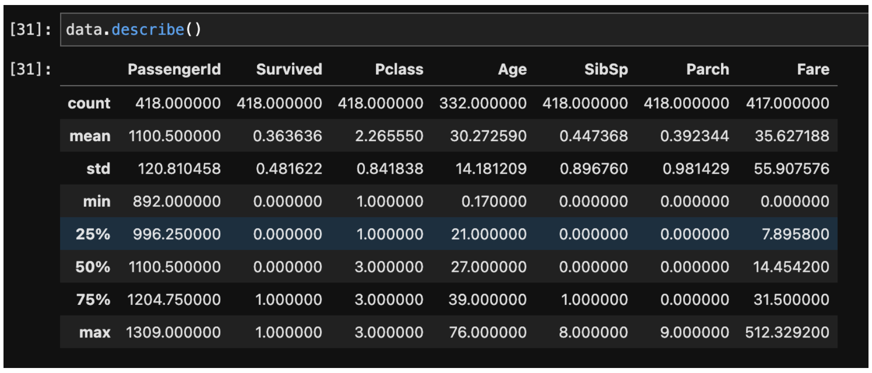
```
[29]: data.head()
```

[29]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S



```
[30]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data c Basic Data Statistics.ipynb umns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Survived     418 non-null    int64
 2   Pclass       418 non-null    int64
 3   Name         418 non-null    object
 4   Sex          418 non-null    object
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64
 7   Parch        418 non-null    int64
 8   Ticket       418 non-null    object
 9   Fare         417 non-null    float64
 10  Cabin        91 non-null     object
 11  Embarked     418 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
```



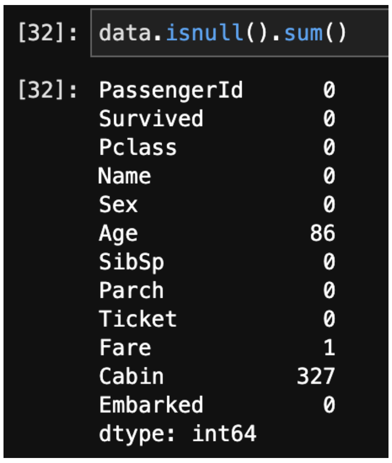
```
[31]: data.describe()
```

[31]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	0.363636	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.481622	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	0.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	0.000000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	0.000000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	1.000000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	1.000000	3.000000	76.000000	8.000000	9.000000	512.329200

Handling Missing Data

Missing data can skew your analysis. Identify missing values and decide on appropriate strategies for imputation or removal.

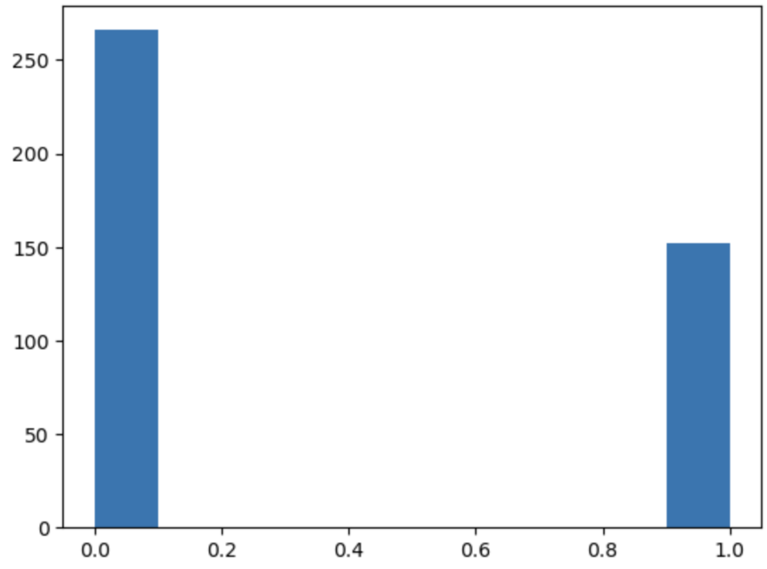


```
[32]: data.isnull().sum()

[32]: PassengerId      0
      Survived         0
      Pclass           0
      Name             0
      Sex              0
      Age             86
      SibSp            0
      Parch            0
      Ticket           0
      Fare             1
      Cabin          327
      Embarked         0
      dtype: int64
```

Data Visualization

Visualization is a powerful EDA tool. Libraries like Matplotlib, Seaborn, and Plotly offer various plotting options.

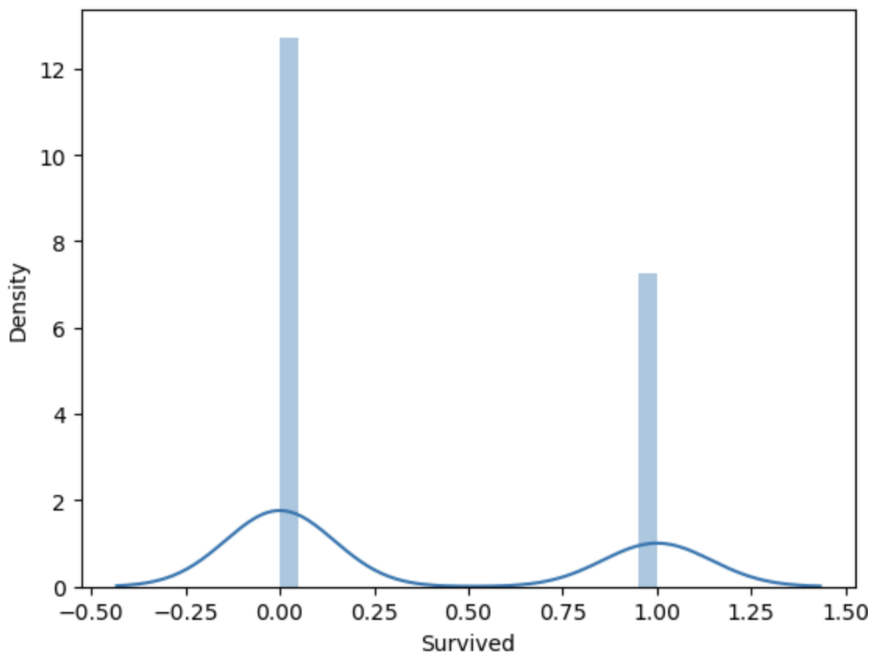
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.hist(data['Survived'])
```



Data Distribution

Understanding the distribution of your data helps in selecting appropriate statistical methods.

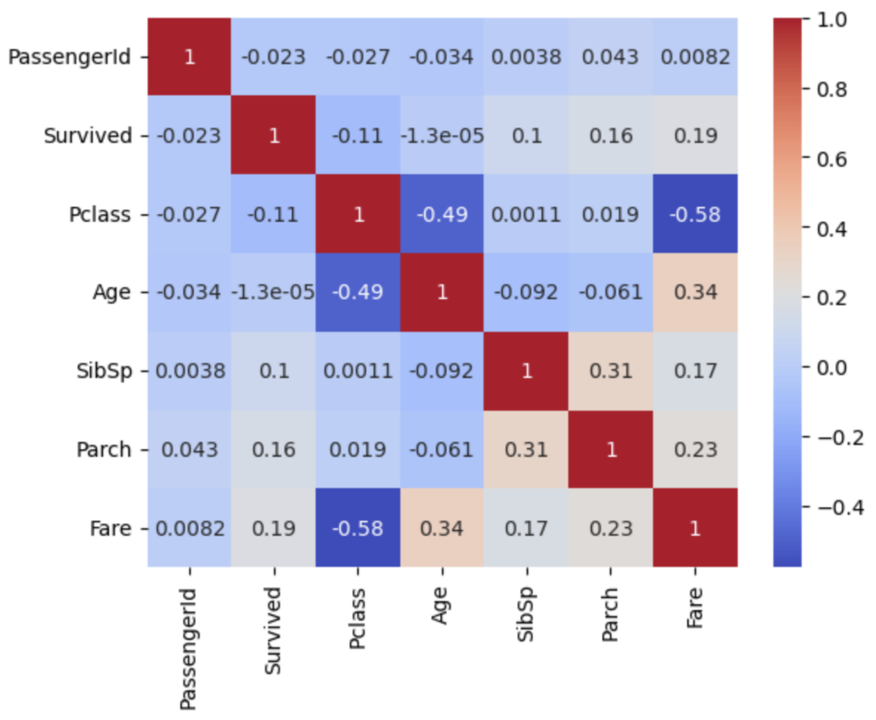
```
sns.distplot(data['Survived'], bins=20)
```



Correlation Analysis

Understanding relationships between variables is crucial. Calculate correlation coefficients and visualize using heatmaps. We will use the `.corr()` function we previously used.

```
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
```



Conclusion

Exploratory Data Analysis is a flexible process, where the techniques applied depend on the dataset and the insights you're seeking. Python's versatility and powerful libraries make it an excellent choice for effectively performing EDA and gaining a deep understanding of your data.