@learneverythingai

DATA ANALYSIS USING PYTHON

SHIVAM MODI
@learneverythingai

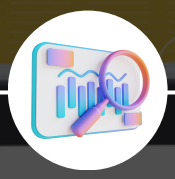
IMPORTING DATA

Before we start analyzing data, we need to import it into our Python environment. One of the most popular libraries for handling data is Pandas. With just a few lines of code, we can read data from various sources, such as CSV files, Excel files, and databases. Let's see how it's done!

```python
import pandas as pd

# Read data from a CSV file
data = pd.read_csv('data.csv')
```

SHIVAM MODI
@learneverythingai



EXPLORING DATA

Before we dive deep into analysis, it's essential to get a sense of the data's structure and distribution. Pandas allows us to inspect the first few rows and generate summary statistics quickly. Understanding the data's basic characteristics will help us make better decisions during analysis.

```
# Display first few rows
print(data.head())

# Summary statistics
print(data.describe())
```

SHIVAM MODI
@learneverythingai

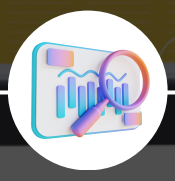
DATA CLEANING

Messy data can be a headache for analysts. But worry not, we've got you covered! Data cleaning is a crucial step to ensure the accuracy of our insights. We'll check for missing values and remove or fill them appropriately. By the end of this step, our data will be pristine and ready for analysis.

```
# Check for missing values
print(data.isnull().sum())

# Drop rows with missing values
data_cleaned = data.dropna()
```

SHIVAM MODI
@learneverythingai



DATA VISUALIZATION

They say a picture is worth a thousand words, and that holds true for data analysis. Visualizations can uncover patterns and trends that might be hidden in raw numbers. We'll use Matplotlib and Seaborn libraries to create various plots, such as scatter plots, bar plots, and heatmaps. Let's make our data come to life!

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

# Create a scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x='age', y='income', data=data_cleaned)
plt.title('Age vs. Income')
plt.xlabel('Age')
plt.ylabel('Income')
plt.show()
```

SHIVAM MODI
@learneverythingai

DATA ANALYSIS - CORRELATION

Now that we have our data cleaned and visualized, it's time to dive deeper into the relationships between variables. Correlation analysis helps us identify which features are positively or negatively related. We'll visualize the correlation matrix as a heatmap to gain better insights into these connections.

```
# Calculate correlation matrix
correlation_matrix = data_cleaned.corr()

# Plot a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

SHIVAM MODI
@learneverythingai

DATA ANALYSIS - GROUPING AND AGGREGATION

Grouping data allows us to explore patterns and trends across different categories. We'll group our data based on specific features, such as gender, and then perform aggregate functions like calculating the mean, median, or count. This way, we can uncover valuable insights based on different segments of the data.

```
# Group by gender and calculate mean income
grouped_data = data_cleaned.groupby('gender')['income'].mean()

# Create a bar plot
plt.figure(figsize=(6, 4))
sns.barplot(x=grouped_data.index, y=grouped_data.values)
plt.title('Average Income by Gender')
plt.xlabel('Gender')
plt.ylabel('Average Income')
plt.show()
```

SHIVAM MODI
@learneverythingai

MACHINE LEARNING - LINEAR REGRESSION

Machine learning takes data analysis to the next level by enabling us to make predictions and create models. We'll start with a simple linear regression model, which can predict a continuous numerical value based on other features. Evaluating the model's performance with metrics like mean squared error helps us assess its accuracy.

```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Prepare data
X = data_cleaned[['age']]
y = data_cleaned['income']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate mean squared error
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

SHIVAM MODI
@learneverythingai

SHIVAM MODI
@learneverythingai

LIKE THIS POST?

- *Follow Me*
- *Share with your friends*
- *Check out my previous posts*

www.learneverythingai.com

Follow

SHARE

SHIVAM MODI
@learneverythingai