CS-23334 FUNDAMENTALS OF DATA SCIENCE

Abenanthan P 240701005

Experiment No: 1 **Date: 24.07.2025**

1.A Analyze the trend of data science job postings over the last decade

AIM:

To analyze and visualize the distribution of various data science roles using a bar chart

ALGORITHM:

1. **Collect the data:** Gather yearly counts of data science job postings (e.g., from a CSV or API).
2. **Import the data with Pandas:** Load the data into a Pandas DataFrame for easy manipulation.
3. **Preprocess the data:** Clean and organize the data, ensuring year and number of postings are correctly formatted.
4. **Visualize the trend:** Use Matplotlib or Seaborn to create a line plot showing job posting counts versus years.
5. **Interpret the trend:** Annotate or summarize key observations (e.g., peaks, growth periods).

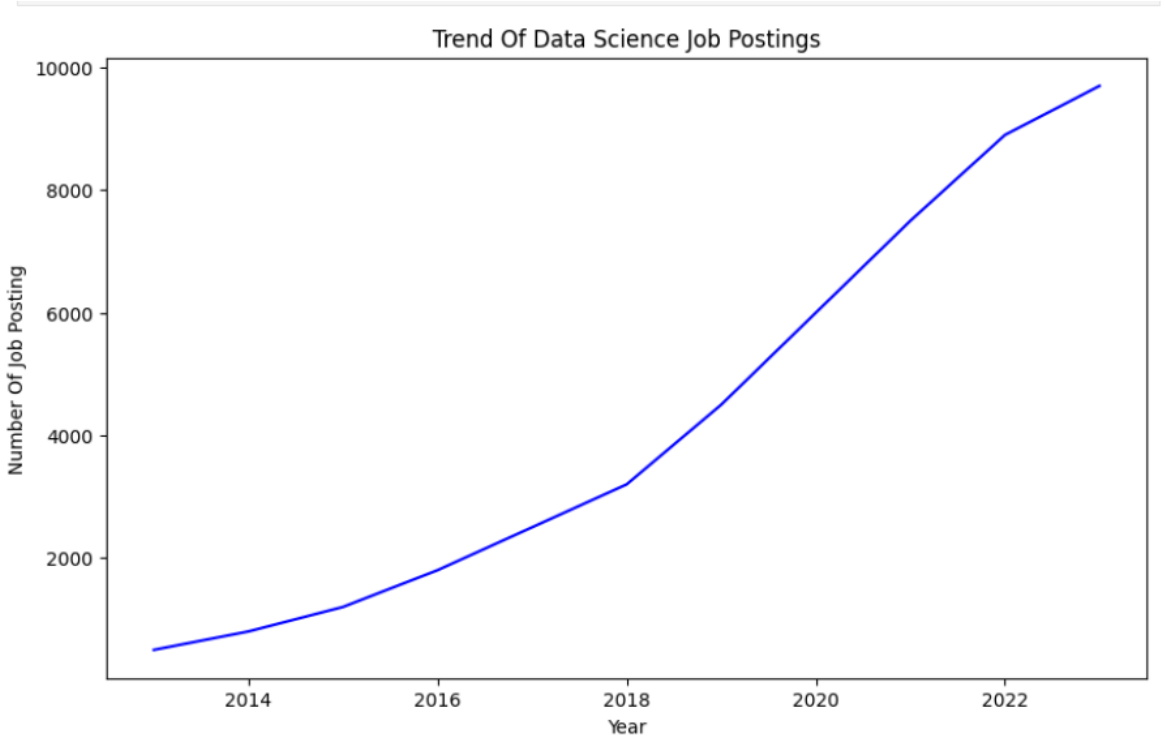
Code with Output:

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt

data={'Year':list(range(2013,2024)),
      'Job_Postings':[500,800,1200,1800,2500,3200,4500,6000,7500,8900,9700]
     }

df=pd.DataFrame(data)
plt.figure(figsize=(10,6))
plt.plot(df['Year'],df['Job_Postings'],color='blue')
plt.title('Trend Of Data Science Job Postings')
plt.xlabel('Year')
plt.ylabel('Number Of Job Posting')
plt.show()
```

Output:



RESULT:

The line graph shows a consistent and significant increase in data science job postings from 2013 to 2023, indicating growing demand in the field

1.B. Analyze and visualize the distribution of various data science roles (Data Analyst, Data Engineer, Data Scientist, etc.) from a dataset

AIM:

To analyze and visualize the distribution of various data science roles using a bar chart.

Algorithm:

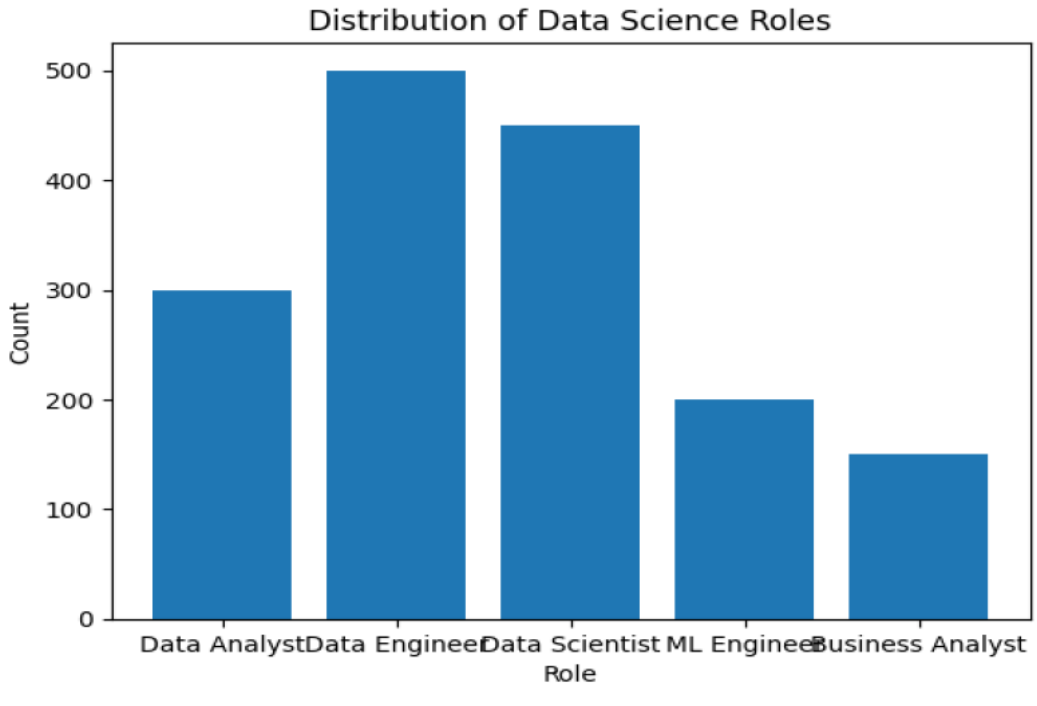
1. **Load job postings dataset:** Import the dataset into a Pandas Dataframe
2. **Identify and categorize roles:** Classify postings into categories (Data Analyst, Data Engineer, etc.).
3. **Count occurrences:** Tally the number of postings for each role.
4. **Visualize the distribution:** Use a pie chart or bar plot (Matplotlib/Seaborn) to display role counts.
5. **Summarize results:** Comment on which roles are most/least common.

Code With Output:

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

roles = ['Data Analyst', 'Data Engineer', 'Data Scientist', 'ML
Engineer', 'Business Analyst']
counts = [300, 500, 450, 200, 150]

plt.bar(roles, counts)
plt.title('Distribution of Data Science Roles')
plt.xlabel('Role')
plt.ylabel('Count')
plt.show()
```



RESULT:

The bar chart reveals that Data Engineer and Data Scientist roles are the most prevalent, followed by Data Analyst, ML Engineer, and Business Analyst.

1.C. Conduct an experiment to differentiate Structured , Un-structured and Semi structured data

Aim:

To differentiate data into structured , unstructured or semi-structured data

Algorithm:

1. **Create small example datasets:** Prepare a sample dataset for each type (structured, unstructured, semi-structured).
2. **Display the datasets:** Present the data in its raw form.
3. **Explain characteristics:** Describe features (e.g., schema, text/spatial data, tags).
4. **Compare and contrast:** Highlight differences between each type.
5. **Summarize findings:** Conclude on the use case and importance of each data type.

Code With Output:

```
# Structured data example

structured_data = pd.DataFrame({
'ID': [1, 2, 3],
'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [25, 30, 35]
})
print("Structured Data:\n", structured_data)
```

```
# Unstructured data example

unstructured_data = "This is an example of unstructured data. It can
be a piece of text, an image, or a video file."
print("\nUnstructured Data:\n", unstructured_data)
```

```
# Semi-structured data example (JSON)
semi_structured_data = {'ID': 1, 'Name': 'Alice', 'Attributes':
{'Height': 165, 'Weight': 68}}
print("\nSemi-structured Data:\n", semi_structured_data)
```

Output:

```
Structured Data:
    ID     Name  Age
0   1    Alice   25
1   2      Bob   30
2   3  Charlie   35

Unstructured Data:
 This is an example of unstructured data. It can be a piece of text,
an image, or a video file.

Semi-structured Data:
 {'ID': 1, 'Name': 'Alice', 'Attributes': {'Height': 165, 'Weight':
68}}
```

RESULT:

- Structured Data: Tabular format with defined schema (e.g., DataFrame with ID, Name, Age).
- Unstructured Data: Free-form text without predefined structure.
- Semi-structured Data: JSON-like format with nested attributes.

1.D. Conduct an experiment to encrypt and decrypt given sensitive data.

Aim:

To encrypt and decrypt sensitive data using the Fernet symmetric encryption method from the cryptography library.

Algorithm:

1. **Prepare sensitive data:** Define the data you want to encrypt (e.g., a string)
2. **Set up the cryptography environment:** Import the required library and generate encryption keys.
3. **Encrypt the data**: Use the library's function to encrypt your data.
4. **Decrypt the data**: Use the corresponding function to retrieve the original data

Code With Output:

```
from cryptography.fernet import Fernet
key = Fernet.generate_key()
f = Fernet(key)
token = f.encrypt(b"Abenanthan 240701005")
token
b'...'
f.decrypt(token)
b'Abenanthan 240701005'
key = Fernet.generate_key()
cipher_suite = Fernet(key)
plain_text = b"Abenanthan 240701005"
cipher_text = cipher_suite.encrypt(plain_text)
decrypted_text = cipher_suite.decrypt(cipher_text)
print("Original Data:", plain_text)
print("Encrypted Data:", cipher_text)
print("Decrypted Data:", decrypted_text)
```

Output:

```
Original Data: b'Abenanthan 240701005'
Encrypted Data:
b'gAAAAABo63XjX4by2WLWfqIDOt_JABlo6QlRY7UFPlF7imBNNTjF6vJNQhST0w0hzNjW
4_dSL-BvwiD6Jipje3GY8Ni3gpgwDn1xyqusL1Jb4YXVEN-Nao4='
Decrypted Data: b'Abenanthan 240701005'
```

RESULT:

The original data ("Abenanthan 240701005") was successfully encrypted into a secure token and decrypted back to its original form, demonstrating effective data protection.