BIG DATA ANALYSIS WITH IBM CLOUD DATABASES

PHASE 4 : DEVELOPMENT PART-2

PROBLEM STATEMENT:

Continue building the big data analysis solution by applying advanced analysis techniques and visualizing the results.
Apply more complex analysis techniques, such as machine learning algorithms, time series analysis, or sentiment analysis, depending on the dataset and objectives. Create visualizations to showcase the analysis results. Use tools like Matplotlib, Plotly, or IBM Watson Studio for creating graphs and charts.

SOLUTION:

Certainly, building a big data analysis solution that incorporates advanced techniques and visualizations is essential for deriving meaningful insights from your data. Let's continue with the process:

1. Select Appropriate Analysis Techniques :

Depending on the nature of your dataset and specific objectives, consider various

Advanced analysis techniques:

Machine Learning Algorithms:
Use supervised or unsupervised machine learning algorithms like decision trees, random forests, support vector machines, or clustering algorithms for predictive modeling or pattern recognition.
Time Series Analysis:
If your data involves time-based data points, use time series analysis techniques to identify trends, seasonality, and forecast future values.
Sentiment Analysis:
Apply natural language processing techniques to extract sentiment from text data, useful for social media or customer reviews analysis.

2. Data Preprocessing:

Ensure your data is prepared for analysis, which includes data cleaning, feature engineering, and data transformation. This step is crucial for the success of advanced analytics.

3. Machine Learning Model Development:

If you're using machine learning, split your dataset into training and testing sets, train various models, and evaluate their performance through metrics such as accuracy, precision, recall, and F1-score.

4. Time Series Analysis:

If you're working with time series data, perform decomposition to separate the time series into trend, seasonality, and residual components. Then, use forecasting techniques like ARIMA or LSTM neural networks to make predictions.

5. Sentiment Analysis:

For sentiment analysis, tokenize and preprocess text data, then employ NLP libraries like NLTK or spaCy for feature extraction and sentiment classification.

6. Create Visualizations:

Use visualization libraries like Matplotlib, Plotly, or IBM Watson Studio for showcasing the analysis results. Here are some examples of the types of visualizations you can create:

Bar Charts and Pie Charts: **For showing categorical data or proportions.**

Line Charts: **Ideal for visualizing time series data and trends over time.**

Scatter Plots: **To explore relationships between variables or clusters.**

Box Plots: **To visualize data distributions and identify outliers.**

7. Interactive Dashboards:

For a more dynamic and interactive representation of your analysis, consider building web-based dashboards using tools like Tableau, Power BI, or Dash by
Plotly. Dashboards allow stakeholders to explore the data and insights in real time.

8. Documentation and Communication:

Document your analysis process, including the steps taken, parameters used, and the rationale behind your decisions. Effective communication is crucial for
sharing insights and findings with your team or stakeholders.

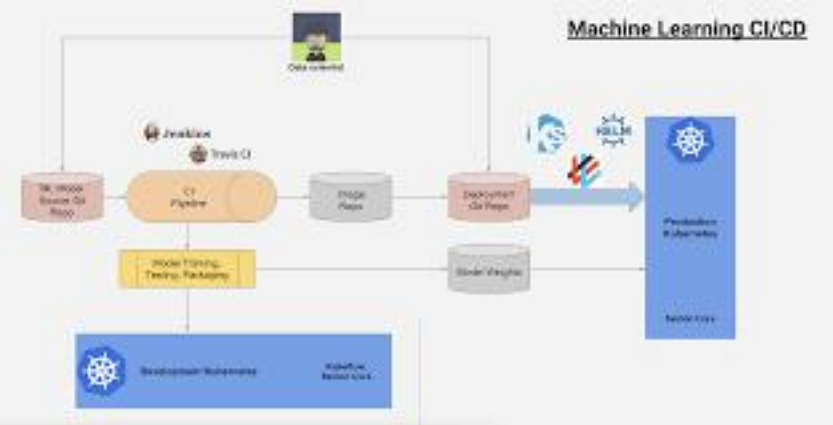
9. Iterate and Refine:

Big data analysis is often an iterative process. Analyze the results, gather feedback, and refine your analysis to gain deeper insights or improve model performance.

10. Automate and Operationalize:

If this analysis needs to be regularly updated or used in a production environment, consider automating the process and integrating it into your organization's systems

MACHINE LEARNING DEPLOYMENT:

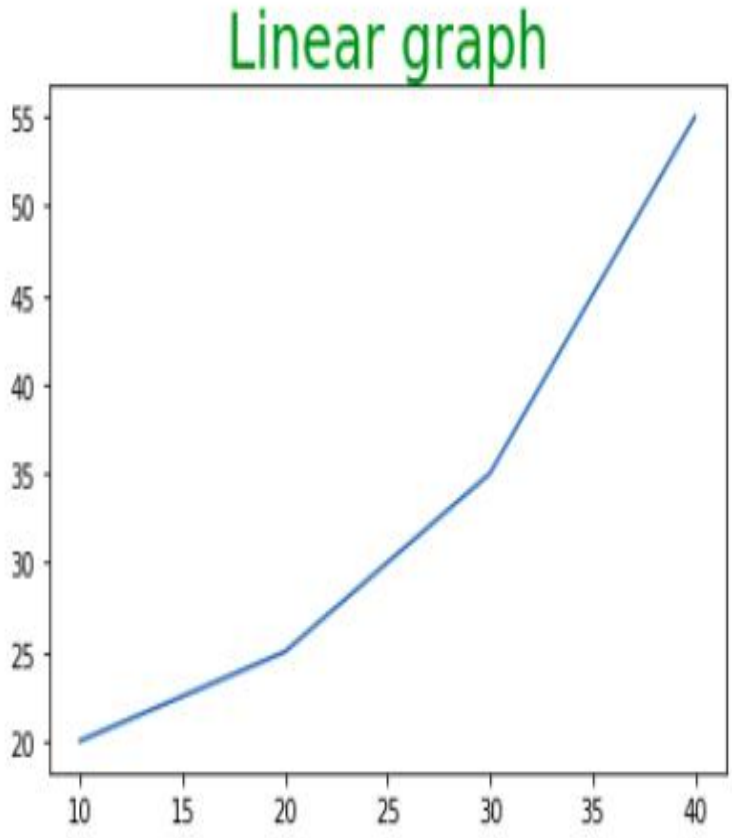


USING PYTHON:

```
import matplotlib.pyplot as plt
```

```
# initializing the data
x = [10, 20, 30, 40]
y = [20, 25, 35, 55]
# plotting the data
plt.plot(x, y)
# Adding title to the plot
plt.title("Linear graph", fontsize=25, color="green")
plt.show()
```

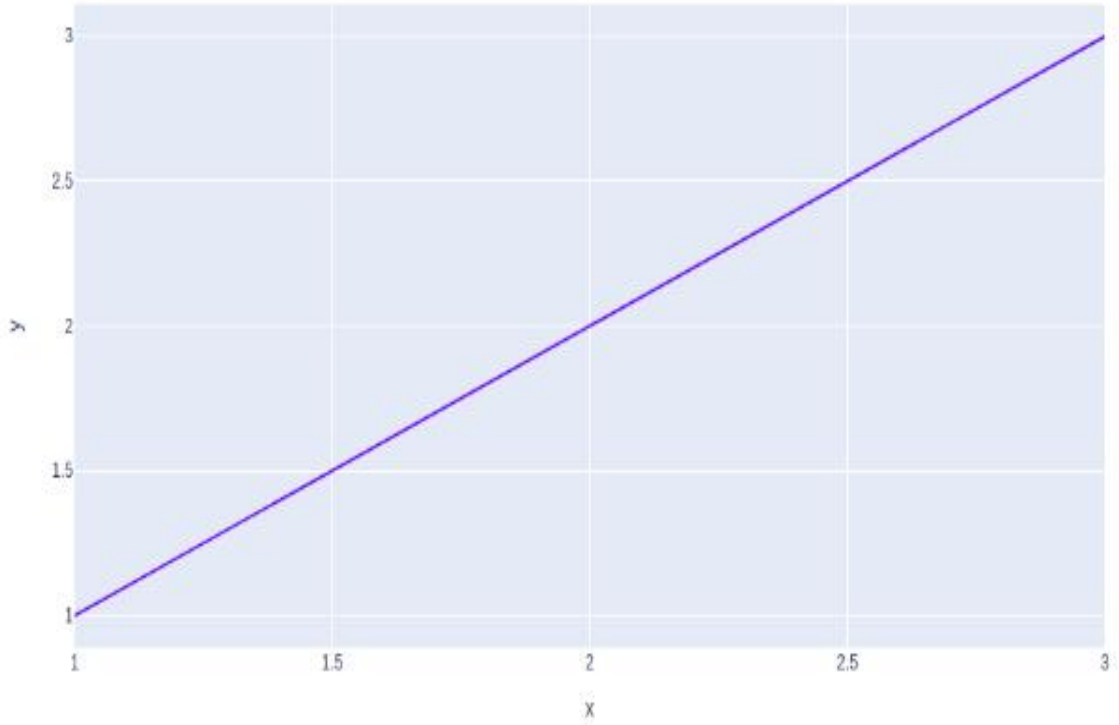
Output:



USING PLOTLY:

```
import plotly.express as px
# Creating the Figure instance
fig = px.line(x=[1,2, 3], y=[1, 2, 3])
# printing the figure instance
print(fig)
```

Output:



2. Column Chart :

A column chart is used to show a comparison among different attributes, or it can show a comparison of items over time.

```
# Dataframe of previous code is used here
```

```
# Plot the bar chart for numeric values
# a comparison will be shown between

# all 3 age, income, sales
df.plot.bar()

# plot between 2 attributes
plt.bar(df['Age'], df['Sales'])
plt.xlabel("Age")
plt.ylabel("Sales")
plt.show()
```

OUTPUT: