**BIG DATA ANALYSIS WITH IBM CLOUD DATABASES**

**PHASE 4 :Development Part 2**

1. Machine Learning Algorithms:

a. Classification with Scikit-Learn:

Implementation:

* Choose appropriate algorithms such as Random Forest, Support Vector Machines, or Neural Networks.
* Perform hyperparameter tuning to optimize model performance.

Visualization:

* Use Matplotlib or Plotly to visualize feature importance.
* Create precision-recall curves, ROC curves, and confusion matrices for model evaluation.

b. Clustering with K-Means:

Implementation:

* Apply K-Means or hierarchical clustering to uncover hidden patterns in the data.
* Explore silhouette scores to determine the optimal number of clusters.

Visualization:

* Plot clusters in 2D or 3D using Matplotlib or Plotly.
* Use interactive plots to explore data points within each cluster.

PROGRAM :

# Import necessary libraries

import pandas as pd

import matplotlib.pyplot as plt

import plotly.express as px

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix

from statsmodels.tsa.seasonal import seasonal\_decompose

from textblob import TextBlob # Assuming you have TextBlob installed

# Load your comprehensive dataset (replace 'your\_comprehensive\_dataset.csv' with your actual dataset)

df\_comprehensive = pd.read\_csv('your\_comprehensive\_dataset.csv')

df\_comprehensive['Timestamp'] = pd.to\_datetime(df\_comprehensive['Timestamp'])

df\_comprehensive.set\_index('Timestamp', inplace=True)

# Machine Learning (Random Forest Classifier for sentiment analysis)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df\_comprehensive['Text'], df\_comprehensive['Sentiment'], test\_size=0.2, random\_state=42)

# Assume you have a function to preprocess text data, e.g., preprocess\_text()

X\_train\_processed = X\_train.apply(preprocess\_text)

X\_test\_processed = X\_test.apply(preprocess\_text)

# Use TextBlob for sentiment analysis

def get\_sentiment(text):

analysis = TextBlob(text)

return 'positive' if analysis.sentiment.polarity > 0 else 'negative' if analysis.sentiment.polarity < 0 else 'neutral'

y\_pred\_sentiment = X\_test\_processed.apply(get\_sentiment)

# Evaluate sentiment analysis

accuracy\_sentiment = accuracy\_score(y\_test, y\_pred\_sentiment)

conf\_matrix\_sentiment = confusion\_matrix(y\_test, y\_pred\_sentiment)

print(f"Sentiment Analysis Accuracy: {accuracy\_sentiment}")

print(f"Confusion Matrix for Sentiment Analysis:\n{conf\_matrix\_sentiment}")

# Time Series Analysis (assuming 'Value' is the time series feature)

result = seasonal\_decompose(df\_comprehensive['Value'], model='additive', period=12)

# Visualize time series components

fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(10, 8), sharex=True)

ax1.plot(df\_comprehensive['Value'], label='Original')

ax1.legend(loc='upper left')

ax1.set\_title('Original Time Series')

ax2.plot(result.trend, label='Trend')

ax2.legend(loc='upper left')

ax2.set\_title('Trend Component')

ax3.plot(result.seasonal, label='Seasonal')

ax3.legend(loc='upper left')

ax3.set\_title('Seasonal Component')

ax4.plot(result.resid, label='Residual')

ax4.legend(loc='upper left')

ax4.set\_title('Residual Component')

plt.tight\_layout()

plt.show()

# Visualize sentiment analysis results using Plotly

fig\_sentiment = px.bar(x=['Positive', 'Negative', 'Neutral'], y=conf\_matrix\_sentiment.flatten(), labels={'y': 'Count', 'x': 'Sentiment'}, title='Sentiment Analysis Results')

fig\_sentiment.show()

Make sure you have the required libraries installed by running:

SYNTAX: pip install textblob

3. Sentiment Analysis:

a. Advanced Sentiment Analysis Techniques:

Implementation:

* Explore deep learning models for sentiment analysis (e.g., using pre-trained models like BERT).
* Consider aspect-based sentiment analysis for more nuanced insights.

Visualization:

* Create sentiment heatmaps or histograms to display sentiment distribution.
* Use word embeddings to visualize word relationships in positive and negative sentiments.

4. Visualizations:

a. Matplotlib, Plotly, or IBM Watson Studio:

Implementation:

* Utilize Matplotlib or Plotly for creating detailed visualizations.
* Leverage IBM Watson Studio for collaborative analysis and reporting.

Examples:

* Create interactive dashboards showcasing key metrics using Plotly.
* Design visually appealing charts for presentations or reports using Matplotlib.

Collaborate on Jupyter Notebooks within IBM Watson Studio.