Assignment - 3

Title: Write a program for Time Series Analysis - Use time series and forecast traffic on a mode of transportation.

Theory

1. Introduction

A **time series** is a sequence of data points collected or recorded at successive time intervals, often with consistent spacing between them (e.g., hourly, daily, monthly). Time series data is typically used to understand patterns, trends, and relationships that evolve over time.

The key components of time series data include:

1. **Trend**: The long-term movement or direction in the data.
2. **Seasonality**: Regular patterns or cycles that occur at consistent intervals.
3. **Noise**: Random variations or errors that do not follow a predictable pattern.

Applications of Time Series:

Time series analysis is used across various fields for different purposes:

1. **Forecasting**: Predicting future values based on historical data (e.g., sales forecasting, weather predictions).
2. **Stock Market Analysis**: Analyzing and predicting stock prices, trends, and market behavior.
3. **Demand and Supply Management**: Optimizing inventory and resource allocation based on demand forecasts.
4. **Healthcare**: Monitoring patient data over time for early disease detection or treatment planning.

Time Series for Traffic Data:

In the context of **traffic volume prediction** , time series analysis plays a vital role in forecasting traffic patterns based on historical data. The key goals for traffic data time series analysis include:

1. **Predicting Traffic Volume**: By analyzing past traffic volume trends (e.g., daily or hourly traffic), we can forecast future traffic volumes. This helps in managing traffic flow, optimizing routes, and improving transportation planning.

2. **Identifying Patterns and Trends**: By extracting features such as the **hour of the day**, **day of the week**, or **season**, we can identify peak traffic periods, congestion patterns, and seasonal trends (e.g., higher traffic during holidays or rush hours).

3. **Improving Traffic Management**: By understanding traffic trends, cities can better allocate resources, optimize traffic signals, and design better road infrastructure.

2. Input and Output:

Input

The input to the model is a **CSV file** containing traffic data. Specifically, the dataset contains main columns like:

1. **Timestamp**: The timestamp of the traffic data entry (in string format, which is converted to a `Timestamp`).
2. **Traffic Volume**: The numeric value indicating the traffic volume (which is cast to a `DoubleType`).

Output

The output consists of **predictions** of traffic volume for the test data, generated by the trained **Linear Regression model**. Specifically:

1. **Predicted Traffic Volume**: The model predicts the traffic volume based on the extracted time features.
2. **Timestamp**: The original timestamp from the test dataset.
3. **Actual Traffic Volume**: The actual traffic volume value from the test dataset.

3. Designed Decisions:

1. Data Loading and Preprocessing

- **Decision:** The dataset is loaded using `spark.read.csv()` with `header=True` and `inferSchema=True` to automatically infer the data types of the columns.
- **Reason:** This ensures that the data is loaded correctly with proper data types and allows Spark to automatically handle data types like `Timestamp`, `Boolean`, `Double`, etc.

2. Timestamp Conversion

- **Decision:** The `Timestamp` column is converted to the datetime format using `to_timestamp()`.
- **Reason:** Converting the timestamp into a datetime object ensures that we can work with time-based features effectively. It allows for further time-based extraction (like Hour, Day, Month, Year) in the analysis.

3. Type Casting for 'Events' and 'Traffic Volume'

- **Decision:**
  - The `Events` column is cast to a `BooleanType()`.
  - The `Traffic Volume` is cast to a `DoubleType()`.
- **Reason:** Casting the `Events` column to a boolean helps in future analysis, particularly if this column were to be used in model features later. `Traffic Volume` needs to be a numeric type (`DoubleType`) for regression analysis and model training.

4. Feature Engineering

- **Decision:**
  - Extract features like Hour, Day, Month, and Year from the `Timestamp` column.
- **Reason:** These time-based features are crucial for understanding trends and patterns in traffic volume over time. These features will allow the model to learn how the traffic volume varies during different hours, days, months, or years.

5. Vector Assembly

- **Decision:** The features `Hour`, `Day`, `Month`, and `Year` are assembled into a single feature vector using `VectorAssembler`**.**
- **Reason:** Spark MLlib requires input features to be in a vector format for machine learning tasks. This transformation makes it suitable for regression models by combining individual time-based features into a single vector.

6. Model Training (Linear Regression)

- **Decision:** A Linear Regression model is used to predict `Traffic Volume` based on the features (`Hour`, `Day`, `Month`, `Year`).
- **Reason:** Linear regression is chosen because it is a straightforward method for predicting continuous numerical values. The features extracted from the timestamp are expected to have linear relationships with the traffic volume.

7. Model Evaluation

- **Decision:** The model's predictions are shown by selecting `Timestamp`, `Traffic Volume`, and `prediction`.
- **Reason:** Displaying the actual traffic volume alongside the predicted values helps assess the model's performance visually, but further evaluation metrics (such as RMSE, R-squared) can be considered for a more formal performance evaluation.

8. Data Conversion and Plotting

- **Decision:** The predictions are converted into a Pandas DataFrame for plotting.
- **Reason:** Plotting is done using matplotlib, which is easier with Pandas DataFrames. This allows for a clear visualization of traffic volume over time, making it possible to analyze trends and patterns.

Why PySpark is used for this problem statement ?

Scalability:

- PySpark can handle large traffic datasets efficiently by distributing the data processing across a cluster, ensuring quick computations even with large amounts of data.

Distributed Computing:

- The program processes traffic data in parallel, speeding up data transformations and model training, which would be slow on a single machine for large datasets.

Efficient Data Preprocessing:

- PySpark provides optimized functions (like `withColumn`, `to_timestamp`, `VectorAssembler`) to preprocess and transform data in a distributed manner.

Feature Engineering:

- PySpark extracts time-based features (`Hour`, `Day`, `Month`, `Year`) efficiently across large datasets, necessary for analyzing time series patterns.

Machine Learning with MLlib:

- The **Linear Regression** model from PySpark's **MLlib** scales well, allowing training on large datasets without performance issues.

Big Data Integration:

- PySpark integrates with big data tools (e.g., **Hadoop**, **Hive**), enabling easy loading and saving of traffic data stored in distributed file systems.

Flexibility with Data Sources:

- It can read from various data sources (e.g., **CSV**, **Parquet**, **HDFS**), making it adaptable to diverse data storage environments.

4. Code Walkthrough

Initialize Spark Session:

- `spark=SparkSession.builder.appName("TimeSeriesAnalysis").getOrCreate()`
- Sets up the Spark environment.

Load Dataset:

- `df=spark.read.csv("/content/traffic_dataset_with_trend.csv", header=True, inferSchema=True)`
- Loads the traffic data into a DataFrame.

Convert Data Types:

- Converts `Timestamp` to a proper timestamp, `Events` to Boolean, and `Traffic Volume` to double.

Extract Time Features:

- Extracts Hour, Day, Month, and Year from the `Timestamp` for feature engineering.

Assemble Features:

- `VectorAssembler` combines time features into a single feature vector.

Train-Test Split:

- Splits the data into 80% training and 20% testing.

Train Model:

- Uses Linear Regression to train a model predicting traffic volume based on time features.

Make Predictions:

- Applies the model to the test set and shows predictions alongside actual values.

Plot Traffic Volume:

- Converts the predictions to Pandas, and plots Traffic Volume over time.

5. Conclusion:

This program demonstrates the use of PySpark for time series analysis to predict traffic volume. By leveraging PySpark's distributed computing capabilities, the code efficiently processes large traffic datasets and extracts relevant time-based features such as hour, day, month, and year from the Timestamp column. These features are then used to train a Linear Regression model to predict traffic volume.

This approach can be expanded by integrating more complex models or additional features, such as weather or events, to improve prediction accuracy. Overall, the program provides a solid foundation for traffic volume forecasting and can be adapted for real-time applications and further optimization.

Assignment – 3

Title - Write a program for Time Series Analysis: Use time series and forecast traffic on a mode of transportation

Code –

```
from pyspark.sql import SparkSession

from pyspark.sql.functions import col, to_timestamp

from pyspark.sql.types import BooleanType, DoubleType

from pyspark.ml.feature import VectorAssembler

from pyspark.ml.regression import LinearRegression

import matplotlib.pyplot as plt


# Initialize Spark session

spark = SparkSession.builder.appName("TimeSeriesAnalysis").getOrCreate()


# Load dataset

df = spark.read.csv("/content/traffic_dataset_with_trend.csv", header=True,
inferSchema=True)


# Convert Timestamp to datetime format

df = df.withColumn("Timestamp", to_timestamp(col("Timestamp")))


# Convert Events to boolean

df = df.withColumn("Events", col("Events").cast(BooleanType()))
```

```
# Convert Traffic Volume to double
df = df.withColumn("Traffic Volume", col("Traffic Volume").cast(DoubleType()))

# Select relevant columns
df = df.select("Timestamp", "Traffic Volume")

# Feature Engineering: Extract time-based features
df = df.withColumn("Hour", col("Timestamp").substr(12, 2).cast("int"))
df = df.withColumn("Day", col("Timestamp").substr(9, 2).cast("int"))
df = df.withColumn("Month", col("Timestamp").substr(6, 2).cast("int"))
df = df.withColumn("Year", col("Timestamp").substr(1, 4).cast("int"))

# Assemble features
vector_assembler = VectorAssembler(inputCols=["Hour", "Day", "Month", "Year"],
outputCol="features")
df = vector_assembler.transform(df)

# Train-test split
train_df, test_df = df.randomSplit([0.8, 0.2], seed=42)

# Train Linear Regression model
lr = LinearRegression(featuresCol="features", labelCol="Traffic Volume")
model = lr.fit(train_df)
```

```
# Make predictions

predictions = model.transform(test_df)


# Show predictions

predictions.select("Timestamp", "Traffic Volume", "prediction").show(10)


# Convert to Pandas for plotting

pdf = predictions.select("Timestamp", "Traffic Volume").toPandas()


# Plot Traffic Volume over Time

plt.figure(figsize=(12, 6))

plt.plot(pdf["Timestamp"], pdf["Traffic Volume"], label="Traffic Volume", color="blue")

plt.xlabel("Timestamp")

plt.ylabel("Traffic Volume")

plt.title("Traffic Volume Over Time")

plt.legend()

plt.grid()

plt.show()
```

Output –

```
+-------------------+--------------+------------------+
|          Timestamp|Traffic Volume|        prediction|
+-------------------+--------------+------------------+
|2023-01-01 02:00:00|         582.0|1234.6460427523778|
|2023-01-01 06:00:00|        3713.0|1241.3545233690759|
|2023-01-01 08:00:00|        4234.0| 1244.708763677425|
|2023-01-01 13:00:00|        2953.0|1253.0943644482977|
|2023-01-01 19:00:00|        1293.0|1263.1570853733447|
|2023-01-01 23:00:00|         900.0|1269.8655659900428|
|2023-01-02 05:00:00|         952.0|1239.6641798977962|
|2023-01-02 11:00:00|        1309.0|1249.7269008228434|
```

```
|2023-01-02 21:00:00|          748.0|1266.4981023645887|
|2023-01-02 22:00:00|          773.0|1268.1752225187633|
+-------------------+---------------+------------------+
only showing top 10 rows
```