Assignment No. 3

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

Theory

Time series analysis is a method used to analyze time-ordered data to identify patterns, trends, and seasonal variations. It plays a crucial role in forecasting traffic volume for different modes of transportation, enabling better resource planning, congestion management, and scheduling.
Key concepts in time series analysis include:

- Trend: The overall movement in the data, indicating whether traffic volume is increasing or decreasing over time.
- Seasonality: Repeating patterns in traffic at specific intervals (e.g., daily rush hours, weekend variations).
- Cyclic Patterns: Irregular fluctuations that do not follow a fixed schedule but occur due to external factors like holidays or special events.
- Forecasting Models: Methods such as Linear Regression, ARIMA (AutoRegressive Integrated Moving Average), and LSTM (Long Short-Term Memory) are used for predicting future values based on past observations.

In this case, Linear Regression (LR) is used as a simple and interpretable forecasting model. It helps in modeling relationships between traffic volume and time-based features like hour, day, month, and year.

Observation

1. Traffic Volume Shows a Clear Trend: The dataset contains time-stamped traffic data, and analysis reveals whether traffic is increasing or decreasing over time.
2. Time-Based Features Impact Traffic: Certain hours, days, or months show higher traffic due to work schedules, holidays, and seasonal variations.
3. Linear Regression Provides a Basic Forecast: While it captures general trends, it may not handle complex patterns such as non-linearity or sudden traffic spikes due to unexpected events.
4. Model Accuracy Depends on Data Quality: Missing values, data inconsistencies, or external factors (e.g., weather conditions, roadblocks) can affect predictions.

Interpretation

- Traffic Trends: By analyzing historical data, we can determine peak hours, off-peak periods, and seasonal fluctuations.
- Prediction Accuracy: Linear Regression provides a simple yet effective way to predict future traffic volumes based on past data, but more advanced models may be required for better accuracy.

- Feature Importance: The extracted features (Hour, Day, Month, Year) contribute differently to traffic predictions, with time of day typically having the highest impact.
- Use Cases: The insights derived from this analysis can help in traffic management, public transport planning, infrastructure development, and congestion control.

Design Decisions

1. Using PySpark for Scalability
    - Since traffic data can be large and continuously generated, PySpark is chosen for distributed processing.
2. Feature Engineering for Better Predictions
    - Extracting hour, day, month, and year helps in identifying daily, weekly, and yearly trends.
3. Choosing Linear Regression
    - A simple model like Linear Regression is computationally efficient and interpretable.
    - More advanced models (like LSTM or ARIMA) can be used for better accuracy in future work.
4. Train-Test Split for Model Evaluation
    - The dataset is split into 80% training and 20% testing, ensuring the model learns from past data before making forecasts.
5. Visualization for Better Understanding
    - Matplotlib is used to plot traffic volume over time, helping in trend analysis and performance evaluation.

Conclusion

Time series analysis is essential for predicting traffic trends in transportation systems. By leveraging Linear Regression and time-based features, this analysis helps forecast traffic volume effectively. While this method provides a good starting point, more complex models like ARIMA or LSTM may be needed to capture non-linear trends and sudden fluctuations more accurately. Proper data preprocessing, feature selection, and model validation play a crucial role in improving prediction accuracy. The insights gained from this analysis can be applied to traffic optimization, urban planning, and public transportation scheduling, making transportation systems more efficient and responsive to demand.

Code:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, to_date, year, month, dayofmonth, count
from pyspark.ml.feature import VectorAssembler
```

```
from pyspark.ml.regression import LinearRegression
from pyspark.sql.functions import when


# Step 1: Initialize Spark Session
spark = SparkSession.builder.appName("NYC_Taxi_Distributed").getOrCreate()


# Step 2: Load the Parquet File (Update file path)
file_path = "/content/yellow_tripdata_2024-12.parquet"
df = spark.read.parquet(file_path)

# Step 3: Extract Date Features
df = df.withColumn("pickup_date", to_date(col("tpep_pickup_datetime"))) \
       .withColumn("year", year(col("pickup_date"))) \
       .withColumn("month", month(col("pickup_date"))) \
       .withColumn("day", dayofmonth(col("pickup_date")))

# Step 4: Aggregate Number of Trips Per Day
daily_trips = df.groupBy("year", "month", "day").agg(count("*").alias("trip_count"))

# Step 5: Use VectorAssembler to Create Feature Vector
feature_cols = ["year", "month", "day"]
vector_assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
assembled_data = vector_assembler.transform(daily_trips).select("features", "trip_count")

# Step 6: Split Data into Training & Testing Sets
train_data, test_data = assembled_data.randomSplit([0.8, 0.2], seed=42)

# Step 7: Train a Linear Regression Model for Forecasting
lr = LinearRegression(featuresCol="features", labelCol="trip_count")
model = lr.fit(train_data)

# Step 8: Predict Future Demand (Assuming Next 30 Days)
future_dates = spark.createDataFrame([
    (2024, 3, day) for day in range(1, 31)
], ["year", "month", "day"])

future_features = vector_assembler.transform(future_dates)
predictions = model.transform(future_features)

# Step 9: Apply ReLU
predictions = predictions.withColumn("prediction", when(col("prediction") < 0,
0).otherwise(col("prediction")))
```

#Step 10: Display the output
predictions.select("year", "month", "day", "prediction").show(17)

# Stop Spark session
spark.stop()

Output:

year	month	day	prediction
2024	3	1	35251.345864336006
2024	3	2	33163.27369796205
2024	3	3	31075.20153158903
2024	3	4	28987.129365215078
2024	3	5	26899.057198841125
2024	3	6	24810.98503246717
2024	3	7	22722.91286609415
2024	3	8	20634.840699720196
2024	3	9	18546.768533346243
2024	3	10	16458.69636697229
2024	3	11	14370.624200599268
2024	3	12	12282.552034225315
2024	3	13	10194.479867851362
2024	3	14	8106.407701477408
2024	3	15	6018.3355351043865
2024	3	16	3930.2633687304333
2024	3	17	1842.19120235648