Air Quality Forecasting Documentation

1. Introduction

Air quality forecasting is essential for predicting pollution levels and taking preventive actions. This document outlines the steps to forecast air quality using PySpark's MLlib, ARIMA, and SARIMA models.

2. Data Preprocessing

Before applying any forecasting models, ensure that your data is preprocessed:

- **Merge Data:** Combine city-level data into state-level data.
- **Handle Missing Values:** Impute or remove missing data.
- **Feature Selection:** Select relevant features such as PM2.5, PM10, NO2, CO, etc.
- **Normalization/Standardization:** Normalize or standardize features for better model performance.

Note: The source of the processed data (whether from Hive or MongoDB) is not specified at this stage. This could be added to the data flow structure based on the final data pipeline setup.

3. PySpark Setup

3.1 Install and Import Libraries

python
Copy code

```
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import RandomForestRegressor,
GBTRegressor, LinearRegression
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
from statsmodels.tsa.arima.model import ARIMA
from pmdarima import auto_arima
```

3.2 Initialize Spark Session

python
Copy code

```
spark =
SparkSession.builder.appName("AirQualityForecast").getOrCreate()
```

4. Feature Engineering with PySpark

4.1 Vector Assembler

Use `VectorAssembler` to transform your features into a single vector:

python
Copy code

```
assembler = VectorAssembler(
    inputCols=["PM10", "PM2.5", "NO2", "CO", "SO2", "Temp"],
    outputCol="features"
)
final_data = assembler.transform(data)
```

Note: Feature reduction can be applied based on the structure of the incoming data. Common techniques include merging columns that show high correlation or checking for linear relationships. Features in the dataset could also be grouped as pollutants (e.g., PM10, NO2) and other metrics (e.g., solar radiation, temperature, relative humidity, rainfall).

5. Model Training using PySpark MLlib

5.1 Split Data

python
Copy code

```
train_data, test_data = final_data.randomSplit([0.8, 0.2])
```

5.2 Example: Linear Regression Model

While this example uses Linear Regression, other models should also be tested for performance comparison:

python
Copy code

```
lr = LinearRegression(labelCol="PM2.5", featuresCol="features")
lr_model = lr.fit(train_data)
```

5.3 Model Evaluation

Evaluate the model using metrics like RMSE (Root Mean Squared Error):

python
Copy code
```
evaluator = RegressionEvaluator(predictionCol="prediction",
labelCol="PM2.5", metricName="rmse")
rmse = evaluator.evaluate(lr_model.transform(test_data))
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

6. Time Series Forecasting using ARIMA/SARIMA

6.1 ARIMA Model

ARIMA is a classic time series forecasting method that includes components for autoregression, differencing, and moving average:

python
Copy code
```
from statsmodels.tsa.arima.model import ARIMA

# Example: Forecasting PM2.5 levels
model = ARIMA(train_data['PM2.5'], order=(1, 1, 1))
model_fit = model.fit()
forecast = model_fit.forecast(steps=10)
```

6.2 SARIMA Model

SARIMA extends ARIMA by adding seasonal components:

python
Copy code
```
from pmdarima import auto_arima

# Example: Finding the best SARIMA model
sarima_model = auto_arima(train_data['PM2.5'], seasonal=True, m=12)
sarima_model.fit(train_data['PM2.5'])
forecast = sarima_model.predict(n_periods=10)
```

7. Model Tuning and Validation

- Use cross-validation and grid search for hyperparameter tuning.

- Evaluate models on validation data and select the best performing one.

8. Deployment

Once the model is validated, deploy it for real-time predictions or integrate it into a dashboard using Tableau or other visualization tools.

9. Conclusion

This document provided a step-by-step guide to preprocess data, engineer features, train models using PySpark MLlib, and forecast using ARIMA/SARIMA. By following these steps, you can effectively predict air quality levels and take preventive actions as needed.