ML Data Pipeline: Predicting Airline Flight Delays

1 Data Cleaning and Preprocessing

- Outlier Removal and Missing Value Handling: Identify and remove outliers in delay-related fields, such as `DEP_DELAY`, `DEP_DEL15`, `CARRIER_DELAY`, `WEATHER_DELAY`, and `NAS_DELAY`, which could skew the regression model. Handle missing values in both flight and weather data by either filling with median values (for continuous variables) or the most frequent category (for categorical variables). Ensure that important variables, such as `DEP_DELAY` and weather metrics, have minimal missing data.

- Data Integration and Alignment: join `Airline Data`, `Weather Data`, `Stations Data`, and `OTPW Data` using timestamps (matching `DATE` in weather data to `FL_DATE` in flight data) and geographic information from `Stations Data`. Ensure that weather information is matched to flights based on the closest station to the departure airport. Use `STATION` in weather data and proximity data from `Stations Data` to refine matches where possible.

Encoding Categorical Features: Apply one-hot encoding to categorical features, such as `DEP_TIME_BLK` (departure time block), `ORIGIN`, `DEST`, and any other relevant categorical fields to capture time-of-day and airport-specific effects.

Feature Scaling: Scale all numerical features (especially those with varying units (e.g., weather metrics like `HourlyWindSpeed` and `DailyMaximumDryBulbTemperature`), using standard scaling or normalization techniques to ensure uniform feature contribution to the regression model.

2 Feature Selection

- Identifying Relevant Features: Start with domain knowledge to prioritize features likely to impact delays, such as CRS_DEP_TIME, DEP_TIME, weather metrics (e.g., HourlyVisibility, DailyPrecipitation), and specific delay types (e.g., CARRIER_DELAY, WEATHER_DELAY).

- Feature Selection Techniques: Univariate Feature Selection: Perform statistical tests to identify the individual relevance of each feature for predicting delay times, focusing on top-performing features.

- Recursive Feature Elimination (RFE): Use RFE with a linear regression model to iteratively remove less informative features, selecting the subset that best predicts delay time.

- Random Forest Feature Importance: Train a random forest regressor to estimate feature importance, which can help prioritize features for the linear regression model.

- Correlation Analysis: Analyze feature correlation to remove redundant features that don't add new information.

3 Model Training

- Training Setup: Split the data into training and validation sets, ensuring stratification by key variables, such as the proportion of delayed versus non-delayed flights, to maintain representativeness. Balance data if necessary to prevent bias toward non-delayed predictions.

- Linear Regression Model: Train a linear regression model on the selected features to predict the total delay time (in minutes). Apply hyperparameter tuning (e.g., regularization with Ridge or Lasso if feasible within Spark's linear regression implementation) to improve model generalization.

4 Model Evaluation

- Holdout Dataset Evaluation: Evaluate the model's performance on a holdout test dataset that was not seen during training.

- Regression Metrics:

- Mean Absolute Error (MAE): Measures average prediction error in minutes for easy interpretation.

- Root Mean Squared Error (RMSE): Penalizes larger errors and provides an understanding of the model's typical prediction error.

- Precision: Measures the proportion of predicted delays (15+ minutes) that are true delays, helping reduce false alarms that might frustrate consumers.

- Recall (Sensitivity): Measures the model's ability to correctly identify all actual delays of 15 minutes or more, minimizing missed delay alerts.

- F1 Score: Provides a balanced metric that combines precision and recall, offering a single measure to evaluate the accuracy of detecting delays above the threshold.

5 Tools

Spark's MLlib and SparkML Libraries:

- Data Cleaning and Preprocessing: Utilize Spark's parallelized methods for data integration, outlier handling, encoding, and scaling.

- Feature Selection: Leverage Spark's implementation of feature selection techniques such as RFE or random forest feature importance to streamline the feature selection process in distributed environments.

- Model Training: Use Spark's linear regression module for training, with options for parameter tuning for optimal model fit.

- Model Evaluation: Spark's evaluation methods will help calculate MAE, RMSE, and R^2 on large datasets efficiently, making it feasible to assess model performance on distributed data.