Efficiently Building Machine Learning Models for Predictive Maintenance in Oil & Gas Industry with Databricks

Daili Zhang, Varun Tyagi
Data Scientists
Halliburton
Introduction

- **Halliburton Digital Solution (HDS)**
  - Support and Consolidate Digital Transformation across all PSLs
    - Provide common platforms/architectures from data warehouse/data governance, analytics development, to BI reporting
    - Streamline and consolidate various digital processes across all PSLs
    - Provide and build a strong talent pipeline for software/digital development

- **Data Science Team**
  - Develop analytics/ML models to
    - Improve operational efficiency
    - Increase productive uptime
    - Reduce operational cost
    - Provide insights at the right time to the right people to help make business-level decisions
Analytics Life Cycle in Halliburton

ML model training & testing takes less than 5% time
What Data Do We Have?

- **Operational Data**
  - Historical data (ADI (proprietary format), Parquet, csv)
    - One Product Service Line (PSL): 500,000+ ADI files (3GB per file in Parquet format) for fracturing jobs collected over 12+ years (1,500TB+ data)
    - Real-time data (edge device, growing significantly over time)

- **Hardware Configuration/Maintenance/Event Data**
  - SAP (for example, 5M maintenance orders in less than 2.5 years)
  - SQL database
  - Files

- **External Data**
  - Weather data
  - Geological & geophysical data

---

No lack of data, but lack of data with QUALITY
Predictive Maintenance Project (example)

- Objective
  - Reduce annual maintenance cost by 10% through field operation optimization based on avoiding failure modes identified by big data analytics for transmissions
Data Cleaning & Aggregation

- Marry the operational data and the configuration/maintenance data in a consistent way
  - Different sample frequency (from 1hz to 1000Hz)
  - Free text input in maintenance records
  - Mixed equipment identification
  - Data discontinuity
  - Missing data
  - Wrong data

- Use Databricks cluster and run time
- Leverage Delta Lake
- Use pandas_udf functions to gain 10-100x speed
Feature Engineering (example)

Select High Load Windows with Continuous Data
- Load Threshold
- Window Size
- # of Windows

Welch Fourier Transform for Each Window
- # of points for each section

Create Features from Windowed Data
- Lag Window Size for Correlation
- # of peaks to select in frequency domain
- Etc.

Combine Features and Cleaning
- Classification/Regression
- Time window to prediction failure
- Balance Data or Not
Model Training/Testing/Selection (example)

- Explored various methods
  - Spark ML
  - Deep Learning
  - Azure AutoML
  - XGBoost
  - Sklearn

- Evaluated the models with various metrics
  - Recall rate
  - F1 score
  - Accuracy
  - Business impact with dollar value

<table>
<thead>
<tr>
<th>Method</th>
<th>Data Preprocess Time</th>
<th>Model Training Time</th>
<th>1 week window</th>
<th>3 weeks window</th>
<th>6 weeks window</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spark ML</td>
<td>Days</td>
<td>Days</td>
<td>0.34</td>
<td>0.68</td>
<td>0.81</td>
</tr>
<tr>
<td>Deep Learning</td>
<td>Days</td>
<td>Weeks</td>
<td>n/a</td>
<td>0.63</td>
<td>n/a</td>
</tr>
<tr>
<td>AutoML</td>
<td>&lt; 1hour</td>
<td>&lt; 1hour</td>
<td>0.44</td>
<td>0.68</td>
<td>0.82</td>
</tr>
</tbody>
</table>
Model Deployment and Visualization

- Modularize the whole process from extracting data to model prediction to results visualization into different notebooks
- Use the Notebooks workflows to synchronize different notebooks runs
- Use user-set widget parameters to pass the parameters used in different notebooks
- Schedule the job to run on daily basis through the notebook UI
- The results are visualized in PowerBI for end users
Model Performance Monitoring

- Store the prediction into blob storage continuously
- Store the actual results into blob storage continuously
- View the discrepancy along the time in PowerBI
- Alerts are set to send emails to users based on specified thresholds
- Investigate the model drifting and re-train the models
Model Management

- Prior using MLflow, manually wrote the model specific information into a .csv file and stored the models into a blob storage with certain name conventions.
- MLflow greatly simplifies the process with consistency and quality.

<table>
<thead>
<tr>
<th>Run Name</th>
<th>Source</th>
<th>failure_win</th>
<th>win_time</th>
<th>test_f1</th>
<th>test_precision</th>
<th>test_recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>run_012</td>
<td>05_automate_all_mlflow</td>
<td>14.0</td>
<td>30.0</td>
<td>0.542</td>
<td>0.52</td>
<td>0.565</td>
</tr>
<tr>
<td>run_011</td>
<td>05_automate_all_mlflow</td>
<td>7.0</td>
<td>30.0</td>
<td>0.316</td>
<td>0.545</td>
<td>0.222</td>
</tr>
<tr>
<td>run_010</td>
<td>05_automate_all_mlflow</td>
<td>1.0</td>
<td>30.0</td>
<td>0.154</td>
<td>0.5</td>
<td>0.091</td>
</tr>
<tr>
<td>run_009</td>
<td>05_automate_all_mlflow</td>
<td>14.0</td>
<td>30.0</td>
<td>0.5</td>
<td>0.667</td>
<td>0.4</td>
</tr>
<tr>
<td>run_008</td>
<td>05_automate_all_mlflow</td>
<td>7.0</td>
<td>30.0</td>
<td>0.424</td>
<td>0.7</td>
<td>0.304</td>
</tr>
<tr>
<td>run_006</td>
<td>05_automate_all_mlflow</td>
<td>14.0</td>
<td>5.0</td>
<td>0.492</td>
<td>0.571</td>
<td>0.432</td>
</tr>
<tr>
<td>run_005</td>
<td>05_automate_all_mlflow</td>
<td>7.0</td>
<td>5.0</td>
<td>0.316</td>
<td>1</td>
<td>0.188</td>
</tr>
<tr>
<td>run_004</td>
<td>05_automate_all_mlflow</td>
<td>1.0</td>
<td>5.0</td>
<td>0.381</td>
<td>0.667</td>
<td>0.267</td>
</tr>
<tr>
<td>run_003</td>
<td>05_automate_all_mlflow</td>
<td>14.0</td>
<td>5.0</td>
<td>0.609</td>
<td>0.875</td>
<td>0.467</td>
</tr>
<tr>
<td>run_002</td>
<td>05_automate_all_mlflow</td>
<td>7.0</td>
<td>5.0</td>
<td>0.276</td>
<td>0.444</td>
<td>0.2</td>
</tr>
<tr>
<td>run_001</td>
<td>05_automate_all_mlflow</td>
<td>1.0</td>
<td>5.0</td>
<td>0.286</td>
<td>1</td>
<td>0.187</td>
</tr>
</tbody>
</table>