MLOps for Sales Forecasting in Databricks

Presented By : Khoo Zi Wei

Objective

1

To forecast the **average monthly sales** for the upcoming year using a **Random Forest** model

2

To implement **automated MLOps** processes utilizing the **Random Forest** model

3

To **evaluate** and **compare** the performance of the current model with an **alternative model or dataset**

Data Collection

Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name
42433	AG-2011-2040	1/1/2011	6/1/2011	Standard Class	TB-11280	Toby Braunhardt
22253	IN-2011-47883	1/1/2011	8/1/2011	Standard Class	JH-15985	Joseph Holt
48883	HU-2011-1220	1/1/2011	5/1/2011	Second Class	AT-735	Annie Thurman
11731	IT-2011-3647632	1/1/2011	5/1/2011	Second Class	EM-14140	Eugene Moren
22255	IN-2011-47883	1/1/2011	8/1/2011	Standard Class	JH-15985	Joseph Holt
22254	IN-2011-47883	1/1/2011	8/1/2011	Standard Class	JH-15985	Joseph Holt
21613	IN-2011-30733	1/2/2011	3/2/2011	Second Class	PO-18865	Patrick O'Donnell
34662	CA-2011-115161	1/2/2011	3/2/2011	First Class	LC-17050	Liz Carlisle
44508	AO-2011-1390	1/2/2011	4/2/2011	Second Class	DK-3150	David Kendrick
23688	ID-2011-56493	1/2/2011	3/2/2011	Second Class	SP-20650	Stephanie Phelps
25293	IN-2011-36074	1/2/2011	5/2/2011	Second Class	DK-13150	David Kendrick
8483	US-2011-118892	1/2/2011	6/2/2011	Standard Class	DH-13075	Dave Hallsten
41445	IR-2011-6550	1/2/2011	6/2/2011	Standard Class	PO-8850	Patrick O'Brill
16727	ES-2011-5268439	1/2/2011	3/2/2011	Second Class	GH-14485	Gene Hale

- **Retail dataset** from a global superstore spanning the years **2011 to 2014**
- Total rows: **51,290**
- Total columns: **24**

https://www.kaggle.com/datasets/jr2ngb/superstore-data

Data Preprocessing

Check for duplicates

```
salesDF:  pyspark.sql.dataframe.DataFrame = [Row ID: long, Order ID: string ... 22 more fields]

Number of duplicates: 0
```

Remove irrelevant column

Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name
42433	AG-2011-2040	1/1/2011	6/1/2011	Standard Class	TB-11280	Toby Braunhardt
22253	IN-2011-47883	1/1/2011	8/1/2011	Standard Class	JH-15985	Joseph Holt
48883	HU-2011-1220	1/1/2011	5/1/2011	Second Class	AT-735	Annie Thurman
11731	IT-2011-3647632	1/1/2011	5/1/2011	Second Class	EM-14140	Eugene Moren
22255	IN-2011-47883	1/1/2011	8/1/2011	Standard Class	JH-15985	Joseph Holt
22254	IN-2011-47883	1/1/2011	8/1/2011	Standard Class	JH-15985	Joseph Holt
21613	IN-2011-30733	1/2/2011	3/2/2011	Second Class	PO-18865	Patrick O'Donnell
34662	CA-2011-115161	1/2/2011	3/2/2011	First Class	LC-17050	Liz Carlisle
44508	AO-2011-1390	1/2/2011	4/2/2011	Second Class	DK-3150	David Kendrick
23688	ID-2011-56493	1/2/2011	3/2/2011	Second Class	SP-20650	Stephanie Phelps
25293	IN-2011-36074	1/2/2011	5/2/2011	Second Class	DK-13150	David Kendrick
8483	US-2011-118892	1/2/2011	6/2/2011	Standard Class	DH-13075	Dave Hallsten
41445	IR-2011-6550	1/2/2011	6/2/2011	Standard Class	PO-8850	Patrick O'Brill

24 columns

	Order Date	1.2 Sales
1	1/4/2011	496.584
2	1/4/2011	67.608
3	1/8/2011	16.116
4	1/9/2011	473.61
5	1/9/2011	44.28
6	2/8/2011	101.52
7	3/1/2011	159.444

2 columns

Check for null values

null_values: pyspark.sql.dataframe.DataFrame = [Order Date: long, Sales: long]

Table

	Order Date	Sales
1	0	0

Rename column

	Order Date	1.2 Sales
1	1/4/2011	496.584
2	1/4/2011	67.608
3	1/8/2011	16.116
4	1/9/2011	473.61
5	1/9/2011	44.28
6	2/8/2011	101.52
7	3/1/2011	159.444

"Order Date" column

	Order_Date	1.2 Sales
1	1/4/2011	496.584
2	1/4/2011	67.608
3	1/8/2011	16.116
4	1/9/2011	473.61
5	1/9/2011	44.28
6	2/8/2011	101.52
7	3/1/2011	159.444

"Order_Date" column

Transform data types

- Contain **mixed date formats**: '**dd/MM/yyyy**' and '**dd-MM-yyyy**'
- Convert to a **standardized date format:** '**yyyy-MM-dd**'

	Order_Date	1.2 Sales
1	2011-04-01	496.584
2	2011-04-01	67.608
3	2011-08-01	16.116
4	2011-09-01	473.61
5	2011-09-01	44.28
6	2011-08-02	101.52
7	2011-01-03	159.444
8	2011-08-03	41.85

Create new features for feature engineering

	1.2 Sales	123 Year	123 Month
1	496.584	2011	4
2	67.608	2011	4
3	16.116	2011	8
4	473.61	2011	9
5	44.28	2011	9
6	101.52	2011	8
7	159.444	2011	1
8	41.85	2011	8

Load Cleaned Data into Databricks Default Database

Split the dataset into **training** and **holdout** sets for later **batch inference**

```
# Split the data frame into training and holdout set, holdout set is used to test the best model after training
training, holdout = salesDF.randomSplit([0.95, 0.05], seed=12345)
```

```
training:  pyspark.sql.dataframe.DataFrame = [Sales: double, Year: integer ... 1 more field]
holdout:  pyspark.sql.dataframe.DataFrame = [Sales: double, Year: integer ... 1 more field]
```

```
# Count the number of rows in the training and holdout set
print(f"Training set count: {training.count()}")
print(f"Holdout set count: {holdout.count()}")
```

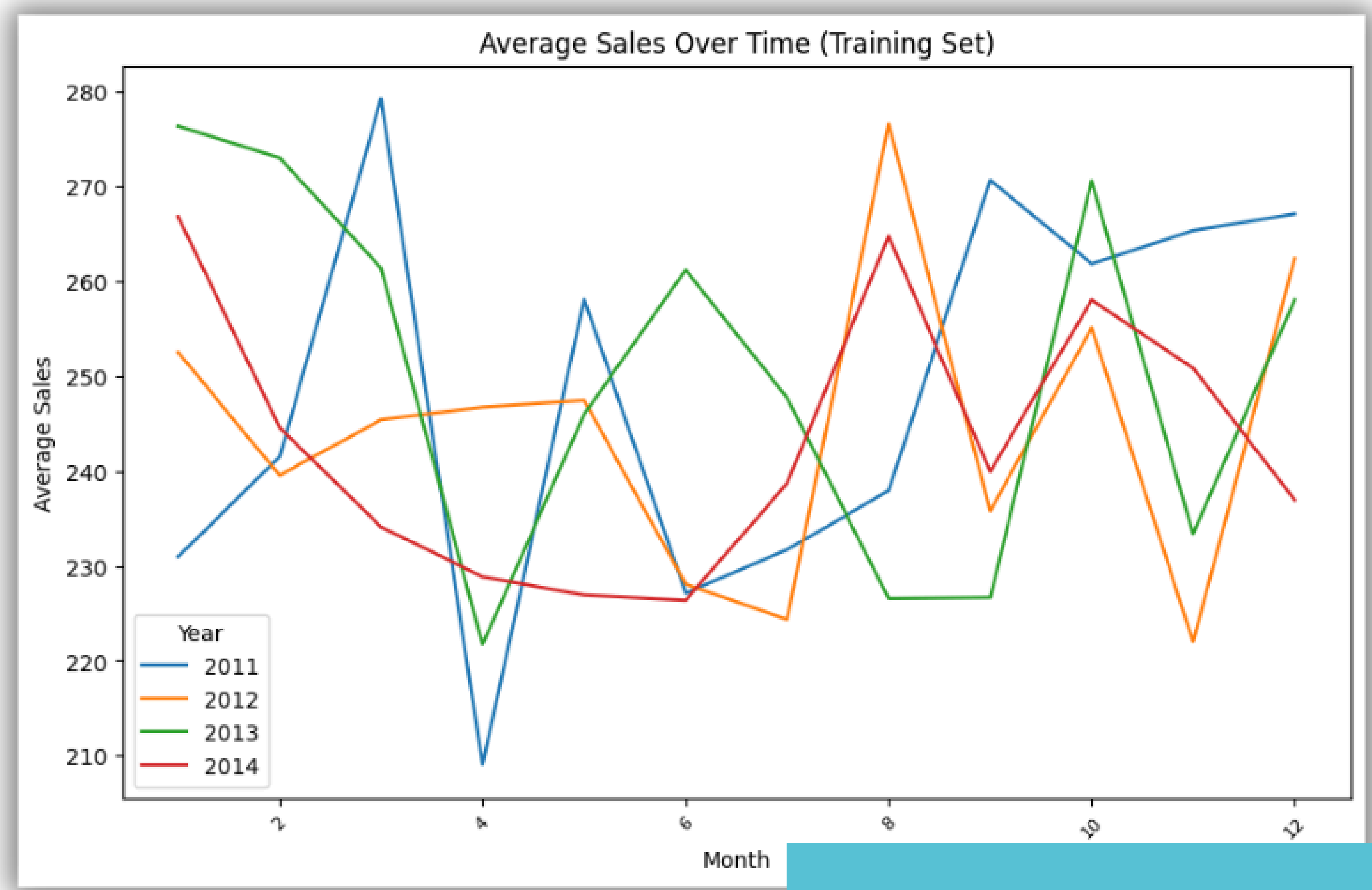
(6) Spark Jobs

```
Training set count: 48711
Holdout set count: 2579
```

```
# Write the training and holdout sets to the default database
salesDF.write.mode("overwrite").saveAsTable("default.salesDF_all")
training.write.mode("overwrite").saveAsTable("default.salesDF_training")
holdout.write.mode("overwrite").saveAsTable("default.salesDF_holdout")
```

EDA Visualization

Total monthly sales show a **consistent increase** from 2011 to 2014



Average monthly sales **remained stable** from 2011 to 2014, ranging between 210 and 280

Feature Engineering

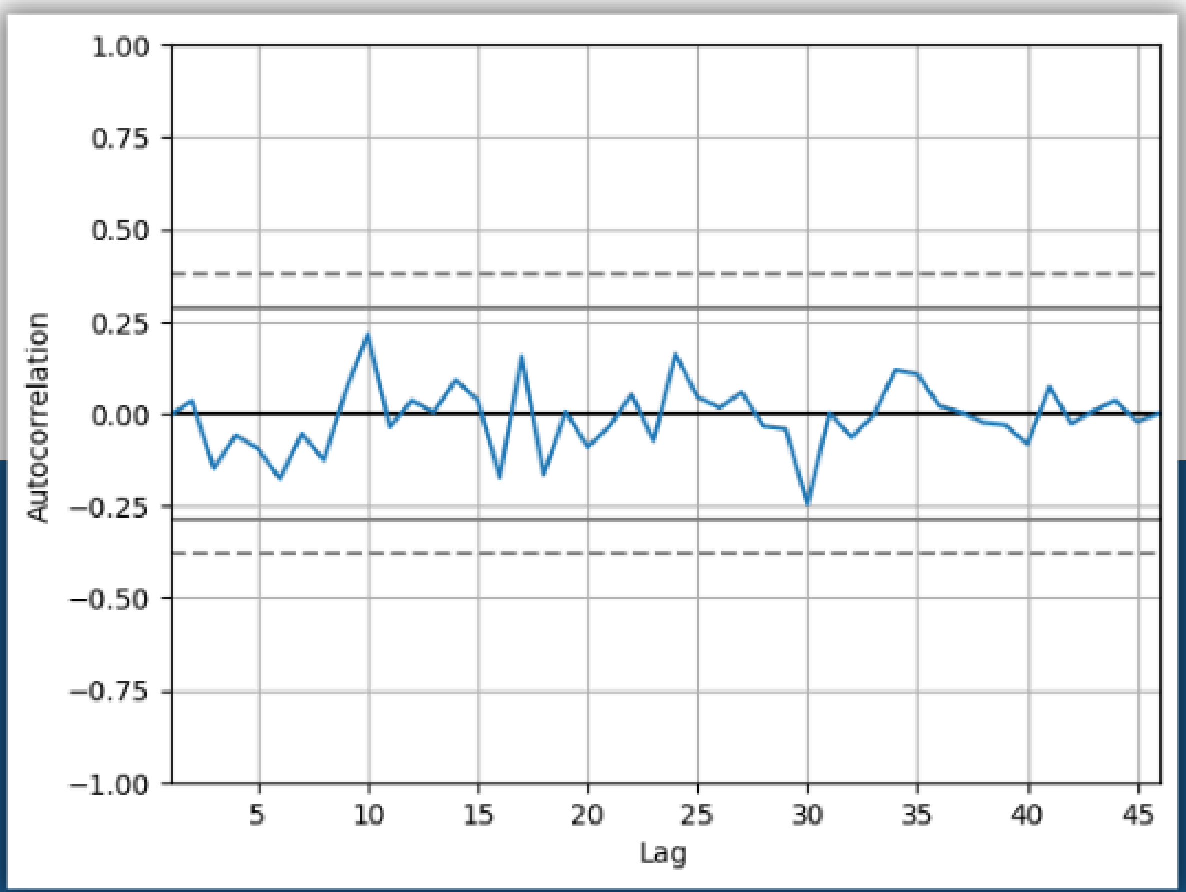
	Year	Month	Average_Sales
1	2011	1	230.97967605839415
2	2011	2	241.51787669467785
3	2011	3	279.257287251462
4	2011	4	209.10857233333334
5	2011	5	258.0917827037037
6	2011	6	227.14450491389206
7	2011	7	231.72045761194028
8	2011	8	237.96321896882492
9	2011	9	270.6569906361828
10	2011	10	261.87341229931974
11	2011	11	265.3569929879741
12	2011	12	267.10185366500826
13	2012	1	252.5330889453125
14	2012	2	239.55358009803916

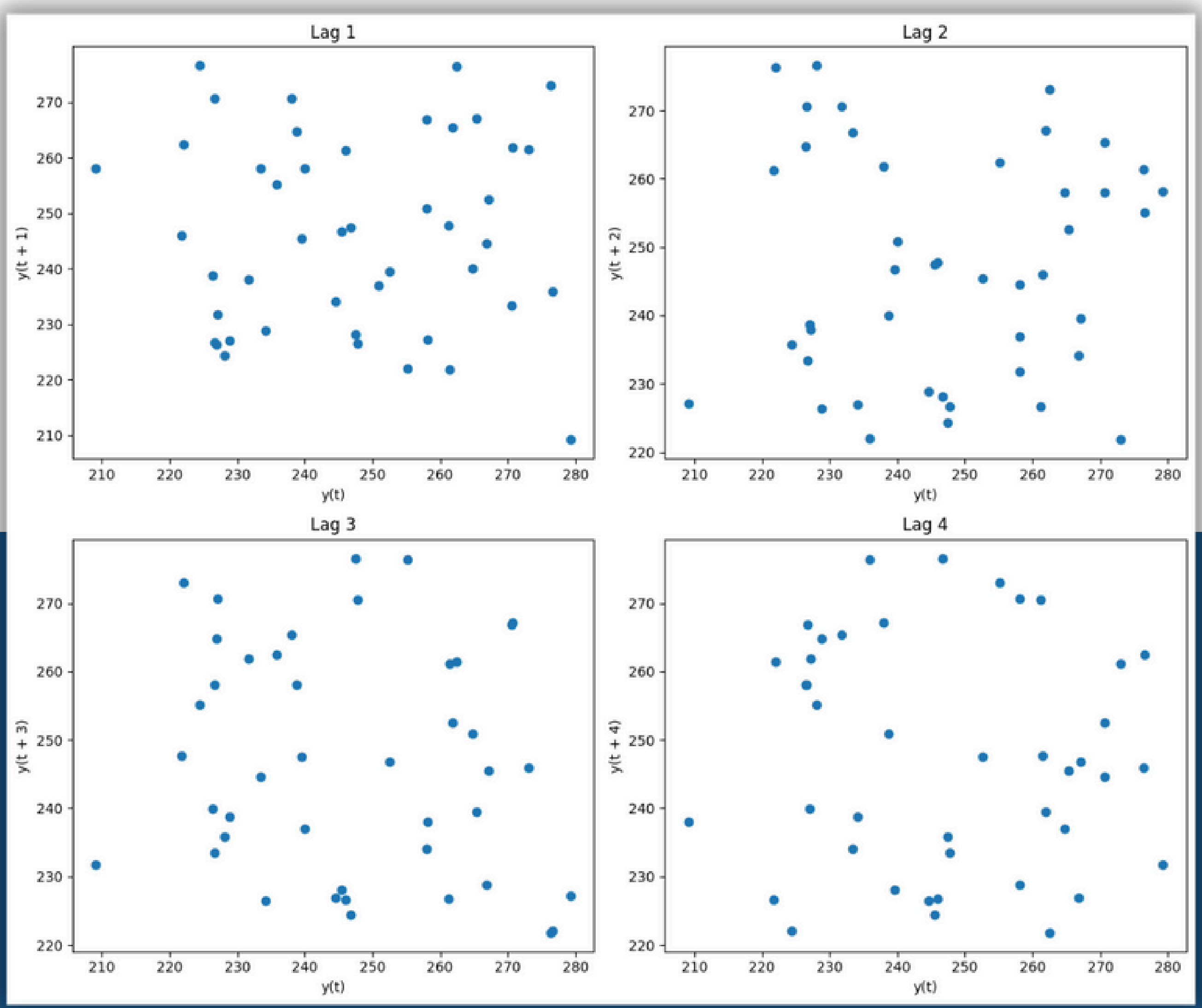
Aggregate Monthly Sales

Calculate the average monthly sales from 2011 to 2014

Presented By : Khoo Zi Wei

Feature Engineering





Autocorrelation Plot

No strong correlation** detected at any lag; autocorrelation values fall **within the confidence interval**, indicating **no statistical significance

Lag Plot

*Data appears **scattered** and **non-linear**, confirming the **absence of autocorrelation***

Feature Engineering

Create Lag Features

Generate 2 lag features from previous monthly sales to enhance time-series forecasting

	Year	Month	Average_Sales	Lag_1	Lag_2
1	2011	3	279.257287251462	241.517876694677...	230.979676058394...
2	2011	4	209.10857233333334	279.257287251462	241.517876694677...
3	2011	5	258.0917827037037	209.108572333333...	279.257287251462
4	2011	6	227.14450491389206	258.0917827037037	209.108572333333...
5	2011	7	231.72045761194028	227.144504913892...	258.0917827037037
6	2011	8	237.96321896882492	231.720457611940...	227.144504913892...
7	2011	9	270.6569906361828	237.963218968824...	231.720457611940...
8	2011	10	261.87341229931974	270.6569906361828	237.963218968824...
9	2011	11	265.3569929879741	261.873412299319...	270.6569906361828
10	2011	12	267.10185366500826	265.3569929879741	261.873412299319...
11	2012	1	252.5330889453125	267.101853665008...	265.3569929879741
12	2012	2	239.55358009803916	252.5330889453125	267.101853665008...
13	2012	3	245.42898313787634	239.553580098039...	252.5330889453125
14	2012	4	246.74043825525044	245.428983137876...	239.553580098039...

Model Development

Modelling - Forecasting

Model Applied

Random Forest

Feature & Target Variables

- Features (X): "Lag_1", "Lag_2"
- Target (y): "Average_Sales"

Data Splitting

- Training Set: 80%
- Testing Set: 20%
- Random Seed: 42

Hyperparameter Tuning

Grid Search

- Used to identify the **optimal parameter values** that yield the **highest score** for the Random Forest Model

Parameter Grid Range

- **n_estimators**: 100, 300, 500
- **max_depth**: 10, 20, 30, None
- **min_samples_split**: 2, 5, 10
- **min_samples_leaf**: 1, 2, 4
- **max_features**: 1.0, sqrt, log2

```
[CV] END max_depth=None, max_features=log2, min_samples_leaf=4, min_samples_split=2, n_estimators=500; total time=   0.6s
[CV] END max_depth=None, max_features=log2, min_samples_leaf=4, min_samples_split=5, n_estimators=100; total time=   0.1s
[CV] END max_depth=None, max_features=log2, min_samples_leaf=4, min_samples_split=5, n_estimators=100; total time=   0.1s
[CV] END max_depth=None, max_features=log2, min_samples_leaf=4, min_samples_split=5, n_estimators=300; total time=   0.4s
[CV] END max_depth=None, max_features=log2, min_samples_leaf=4, min_samples_split=5, n_estimators=500; total time=   0.6s
[CV] END max_depth=None, max_features=log2, min_samples_leaf=4, min_samples_split=5, n_estimators=500; total time=   0.6s
Best Parameters: {'max_depth': 10, 'max_features': 1.0, 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': 100}
Best Score: -320.7661111890394
```

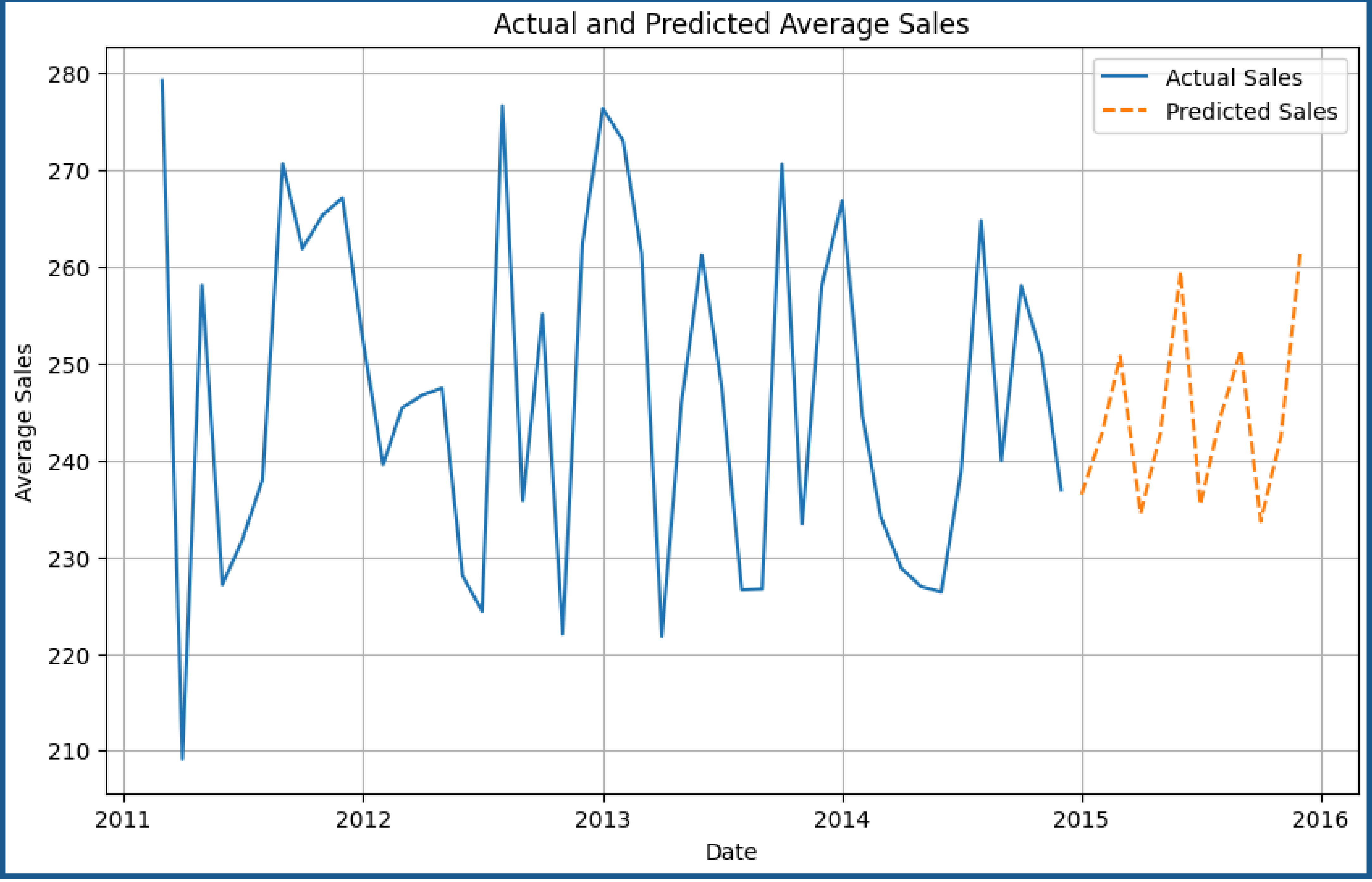
Model Evaluation

- **RMSE**: 18.406539339764727
- **Mean Absolute Error (MAE)**: 15.23723403125409
- **R-squared**: -1.146062560508613
- **Cross-validation MAE scores**: [158.70134892, 460.54660561, 384.15971386, 360.48321615, 500.90874148]
- **Mean cross-validation MAE**: 372.95992520480445
- **MAPE**: 0.06490659892435668

The **cross-validation MAE** is **significantly higher** than the **regular MAE**, which suggests that the model might be **overfitting** the training data and **may not perform well** on **new data**.

MAPE measures the **percentage difference** between **predicted and actual values**. Since the MAPE is **under 10%**, it indicates that the model is **accurate** and **performs well in forecasting**.

Forecasting Result



Model Deployment

Register ML model

- **Log** and **register** the Random Forest model in **MLflow** to **keep track** of its different versions and **manage** them

```
import mlflow
import mlflow.sklearn

# Log the model
mlflow.sklearn.log_model(model, "random-forest-model")

# Get the run ID of the current active run
run_id = mlflow.active_run().info.run_id

# Register the model
model_uri = f"runs:/{run_id}/random-forest-model"
mlflow.register_model(model_uri, "RandomForestModel")
```

```
from mlflow.tracking import MlflowClient

# Initialize the MLflow Client
client = MlflowClient()

# Get the latest version of the registered model
model_name = "RandomForestModel"
latest_version_info = client.get_latest_versions(model_name, stages=["None"])[0]

# Transition the registered model version to the "Staging" stage
client.transition_model_version_stage(
    name=model_name,  # The name of the registered model
    version=latest_version_info.version,  # The version number of the model
    stage="Staging"  # The stage to transition to ("Staging", "Production", or "Archived")
)

print(f"Model version {latest_version_info.version} set to 'Staging' stage.")
```

Set the model stage to Staging

- Set the model stage to "**Staging**" to show that the Random Forest model is currently **being tested or validated**

Experiment with new model

Model

XGBoost: A strong model for forecasting; use it to train a more accurate model

Hyperparameter Tuning

HyperOpt: Used to find the best parameter values

Parameter Search Range

- max_depth: Integer value between 4 and 100
- learning_rate: Log-uniform distribution between 10^−3 and 1
- reg_alpha: Log-uniform distribution between 10^-5 and 10^−1
- reg_lambda: Log-uniform distribution between 10^−6 and 10^−1
- min_child_weight: Log-uniform distribution between 10^−1 and 100

```
# Search for the best run based on the MAPE metric
best_run = mlflow.search_runs(order_by=['metrics.mape ASC']).iloc[0]

# Print the MAPE of the best run
print(f'MAPE of Best Run: {best_run["metrics.mape"]}')

MAPE of Best Run: 0.057944726094620355
```

MAPE value of XGBoost model is **less than 0.06**, **outperforms** the Random Forest model

Experiment with new model

```
# Archive the old model version
client.transition_model_version_stage(
    name=model_name,
    version=latest_version_info.version,
    stage="Archived"
)

# Promote the new model version to Production
client.transition_model_version_stage(
    name=model_name,
    version=new_model_version.version,
    stage="Production"
)
```

Update the Production Model:

Archive the Random Forest model and promote the XGBoost model to production.

Version	Registered at	Created by	Stage
Version 11	2024-08-26 15:59:47	zkhoo@slb.com	Production
Version 10	2024-08-26 15:53:30	zkhoo@slb.com	Archived

Batch Inference

Load Dataset

- Load the previously defined holdout set from the Databricks default database

```
# redefining key variables here because %pip and %conda restarts the Python interpreter
input_table_name = "default.salesDF_holdout" # Test the best Random Forest model with holdout set
output_table_path = "/FileStore/batch-inference/salesDF_output"
```

Run the Current Model

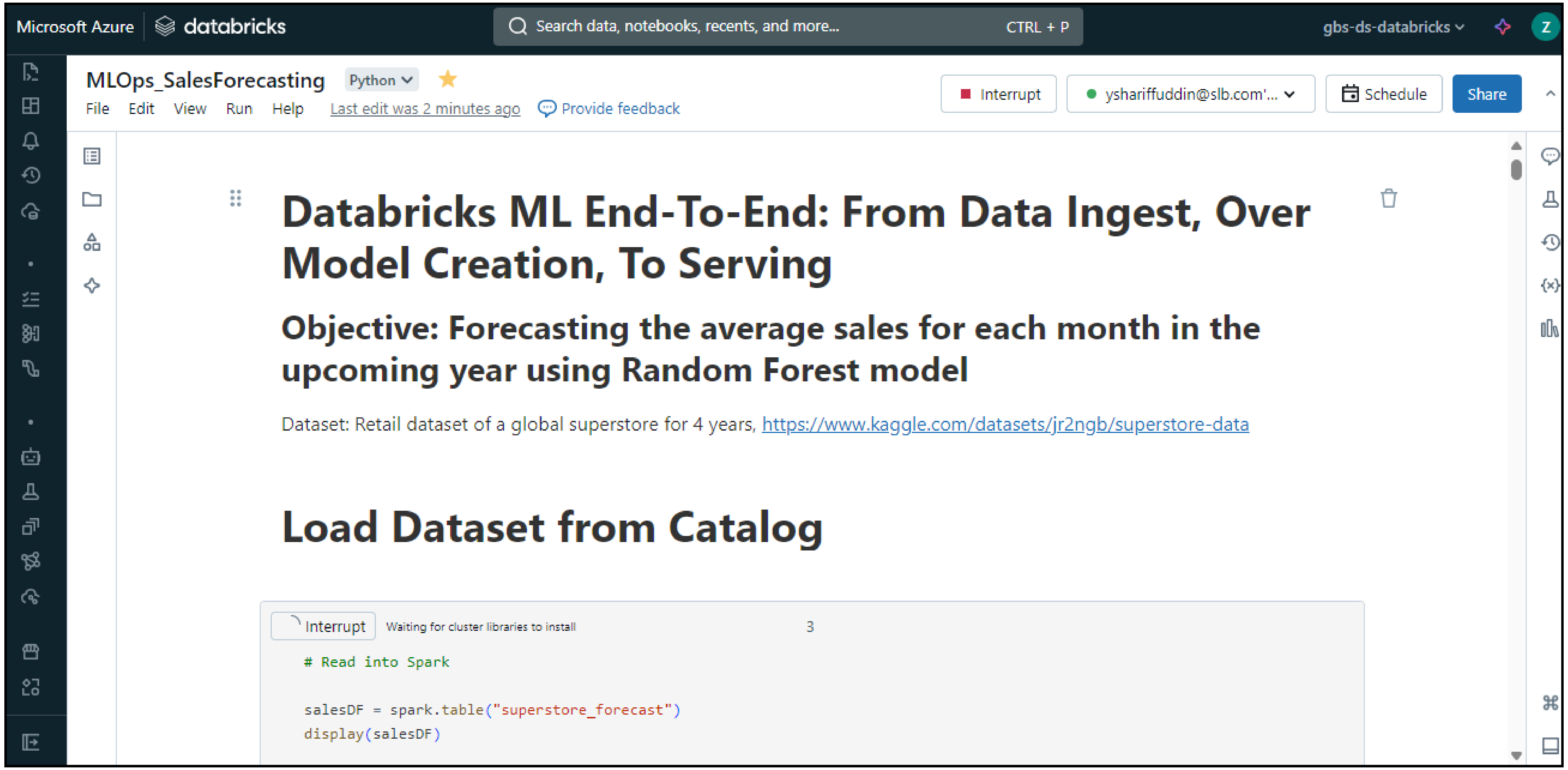
- Apply the same feature engineering steps to the data and run the current Random Forest model

Evaluate Model

- If the MAPE value is below 10%, register the model as "Production." If not, skip the registration

```
Model did not meet the success threshold. MAPE: 0.3000004982295548. Registration skipped.
```

Demonstration



Presented By : Khoo Zi Wei

MLOps_SalesForecasting - Databricks (azuredatabricks.net)

Thank You