

# Choosing the Algorithm

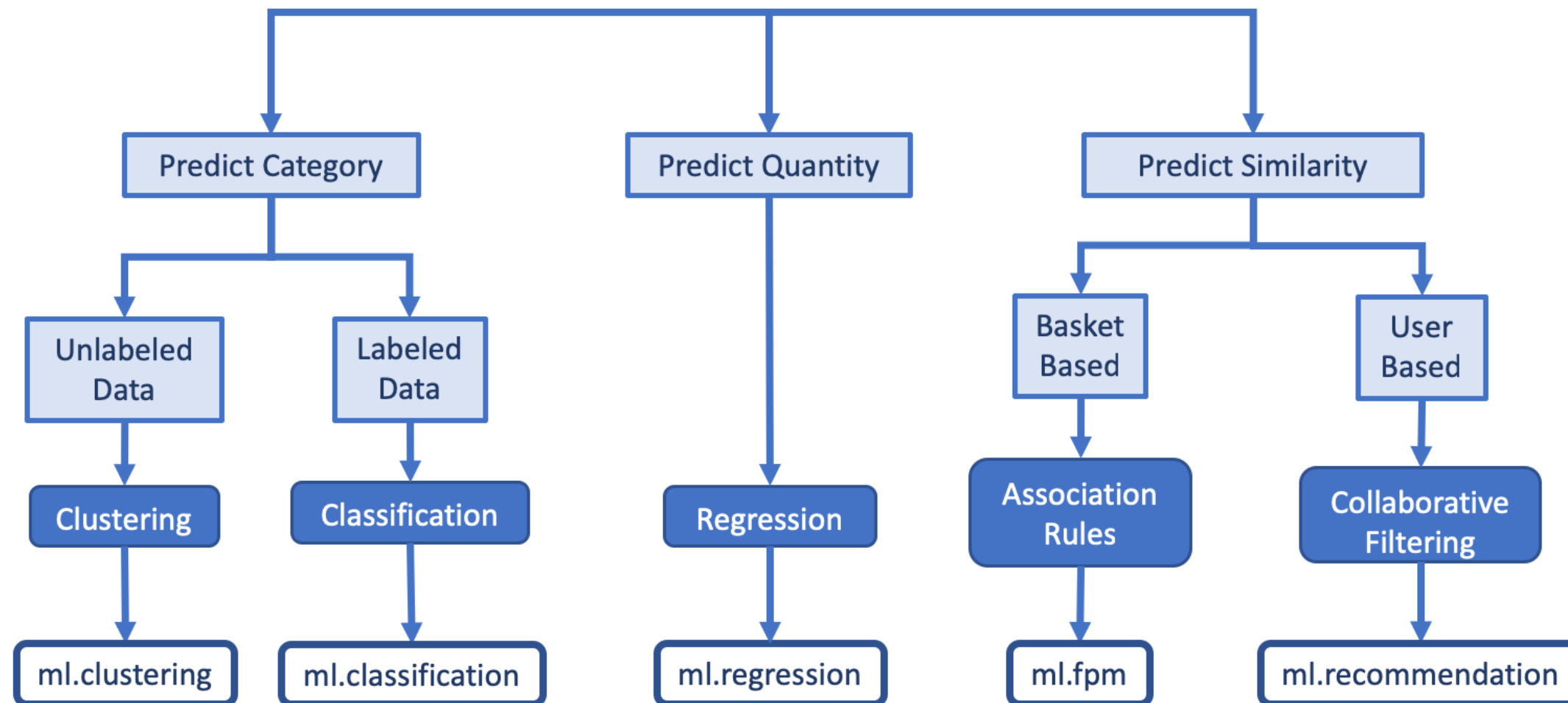
FEATURE ENGINEERING WITH PYSPARK



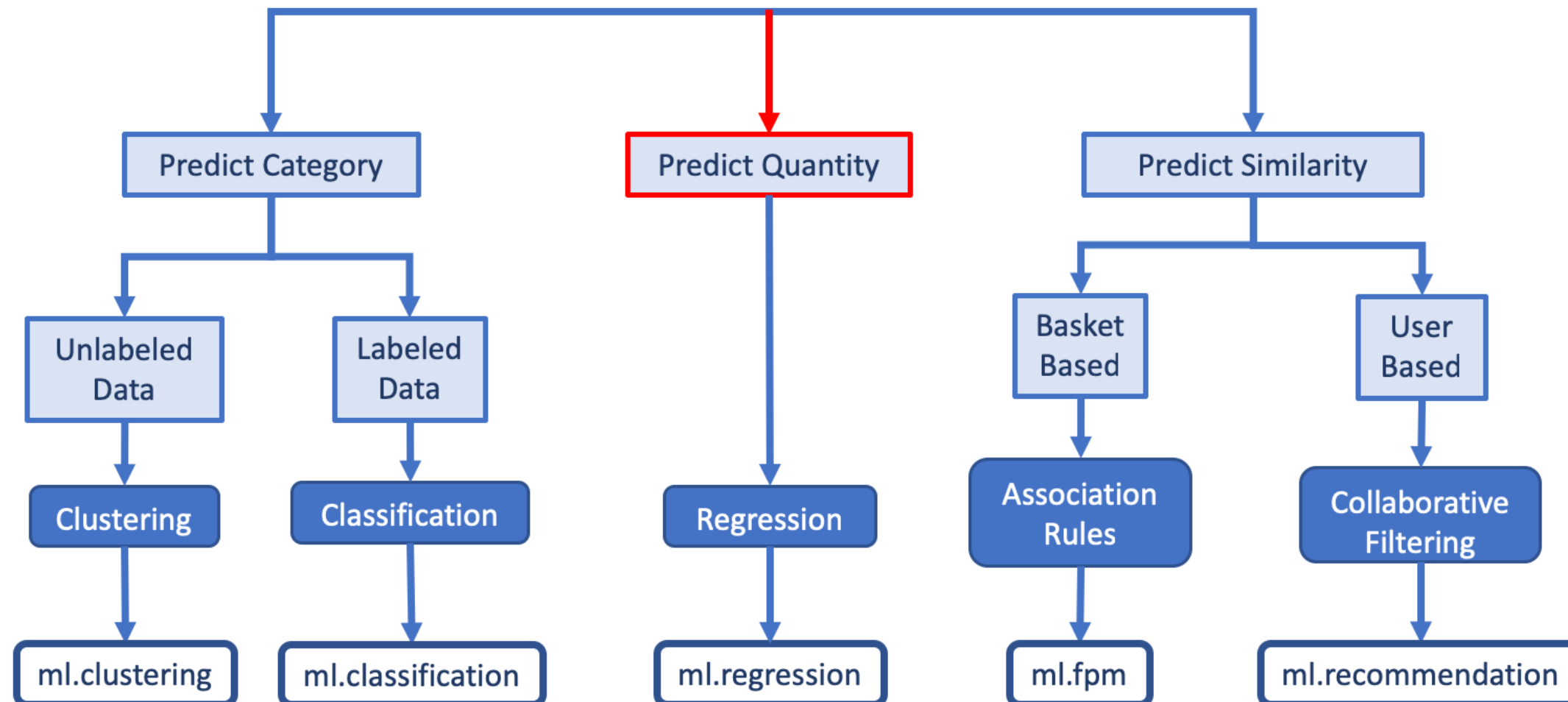
**John Hogue**

Lead Data Scientist, General Mills

# Spark ML Landscape



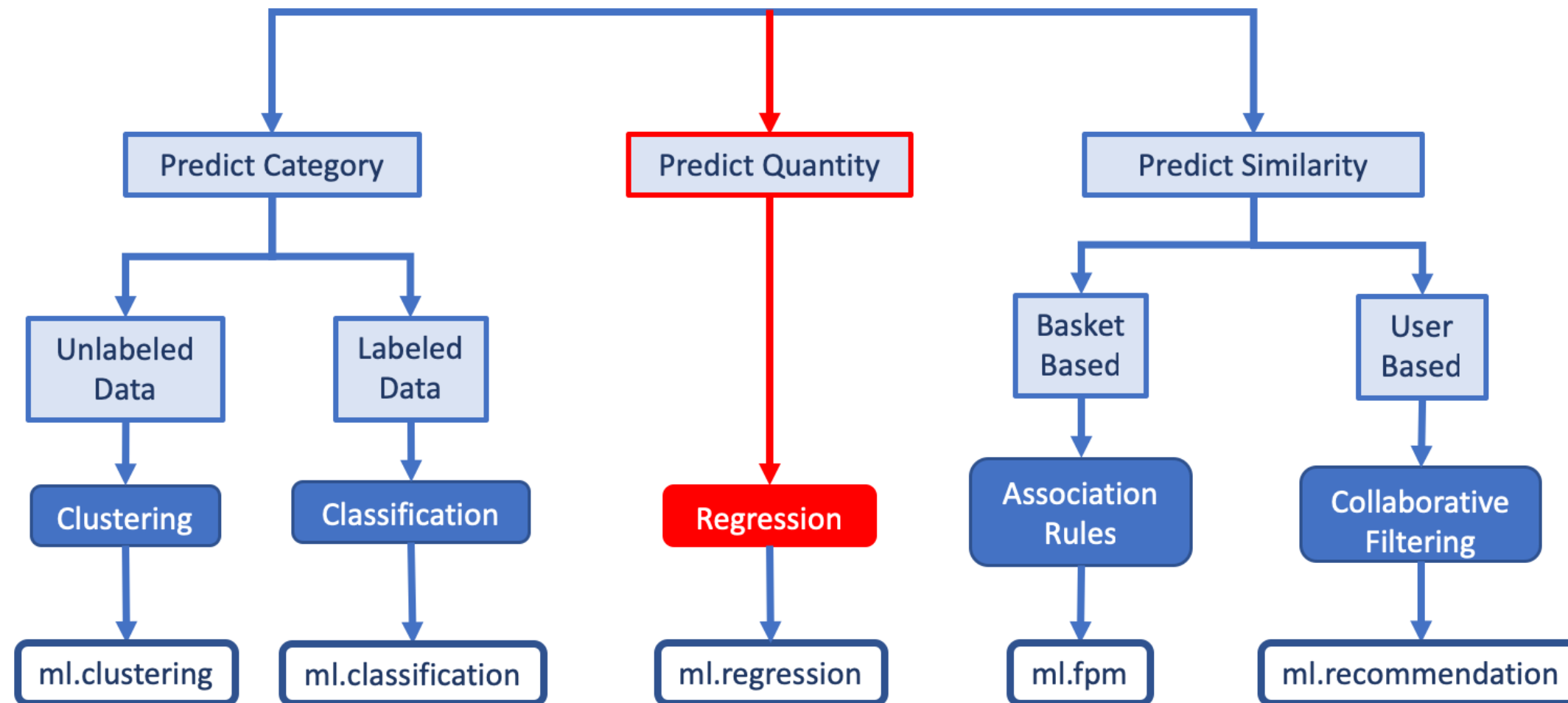
# Spark ML Landscape



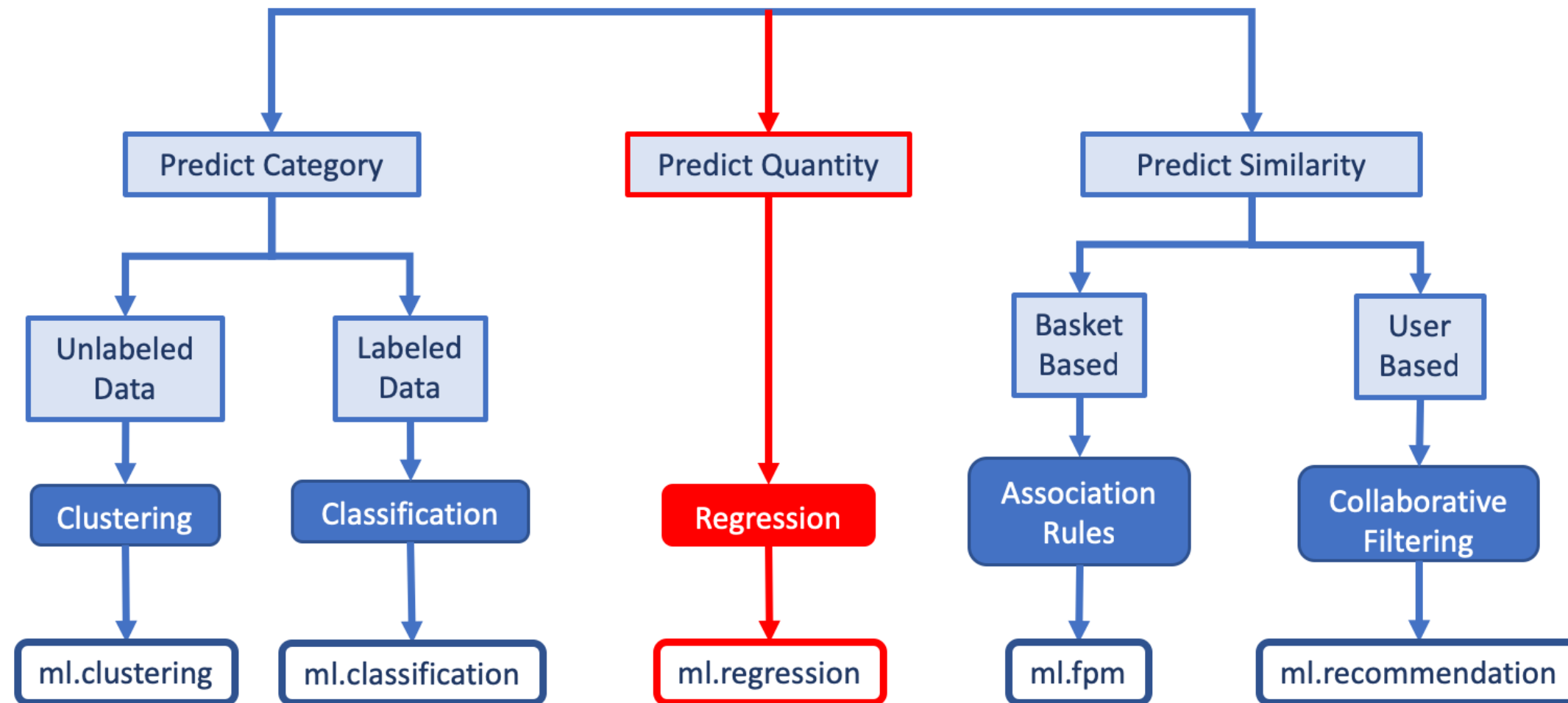
# Spark ML Landscape



©John Hogue



# Spark ML Landscape



# PySpark Regression Methods

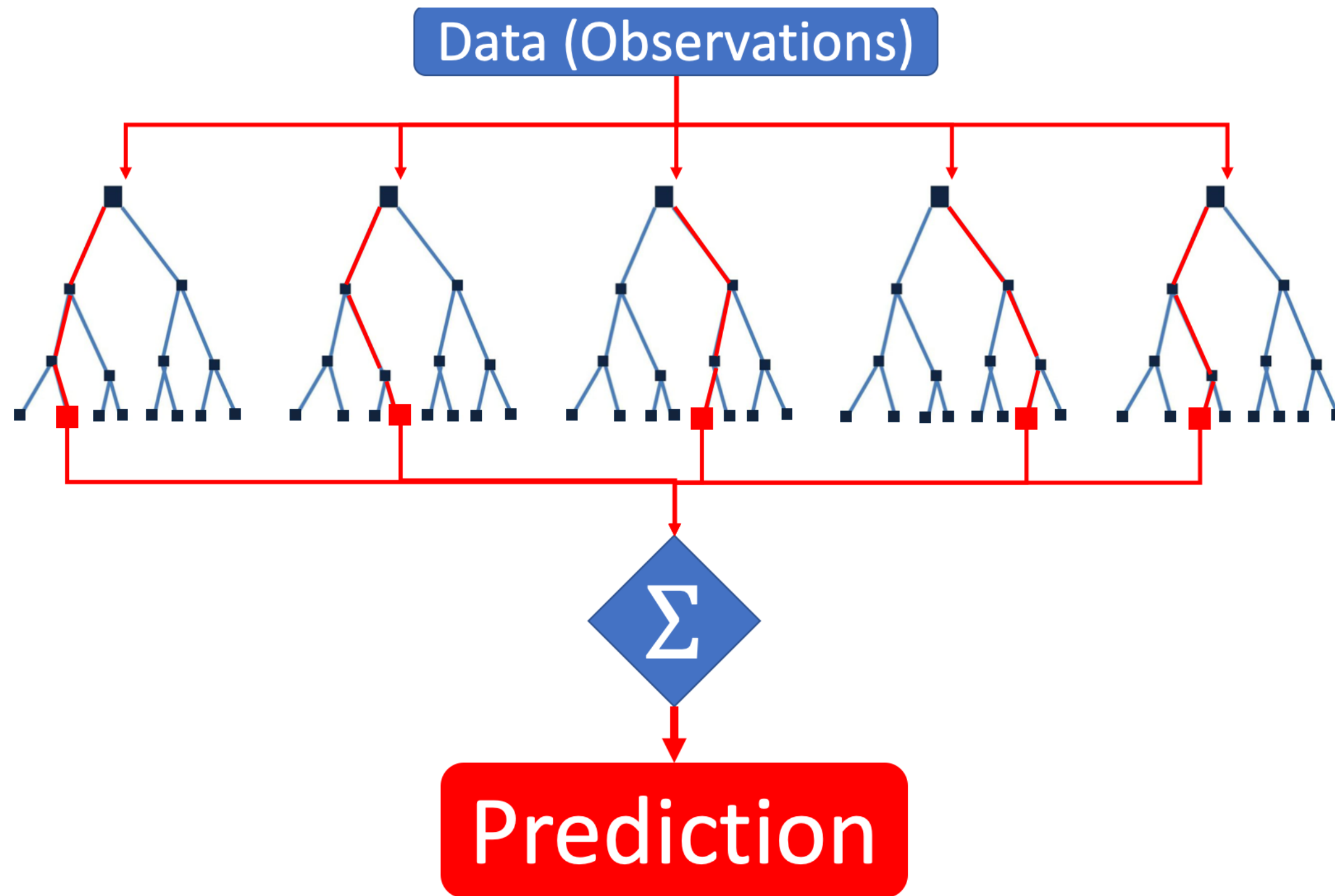
Methods in `ml.regression` :

- `GeneralizedLinearRegression`
- `IsotonicRegression`
- `LinearRegression`
- `DecisionTreeRegression`
- `GBTRegression`
- `RandomForestRegression`

# PySpark Regression Methods

Methods in `ml.regression` :

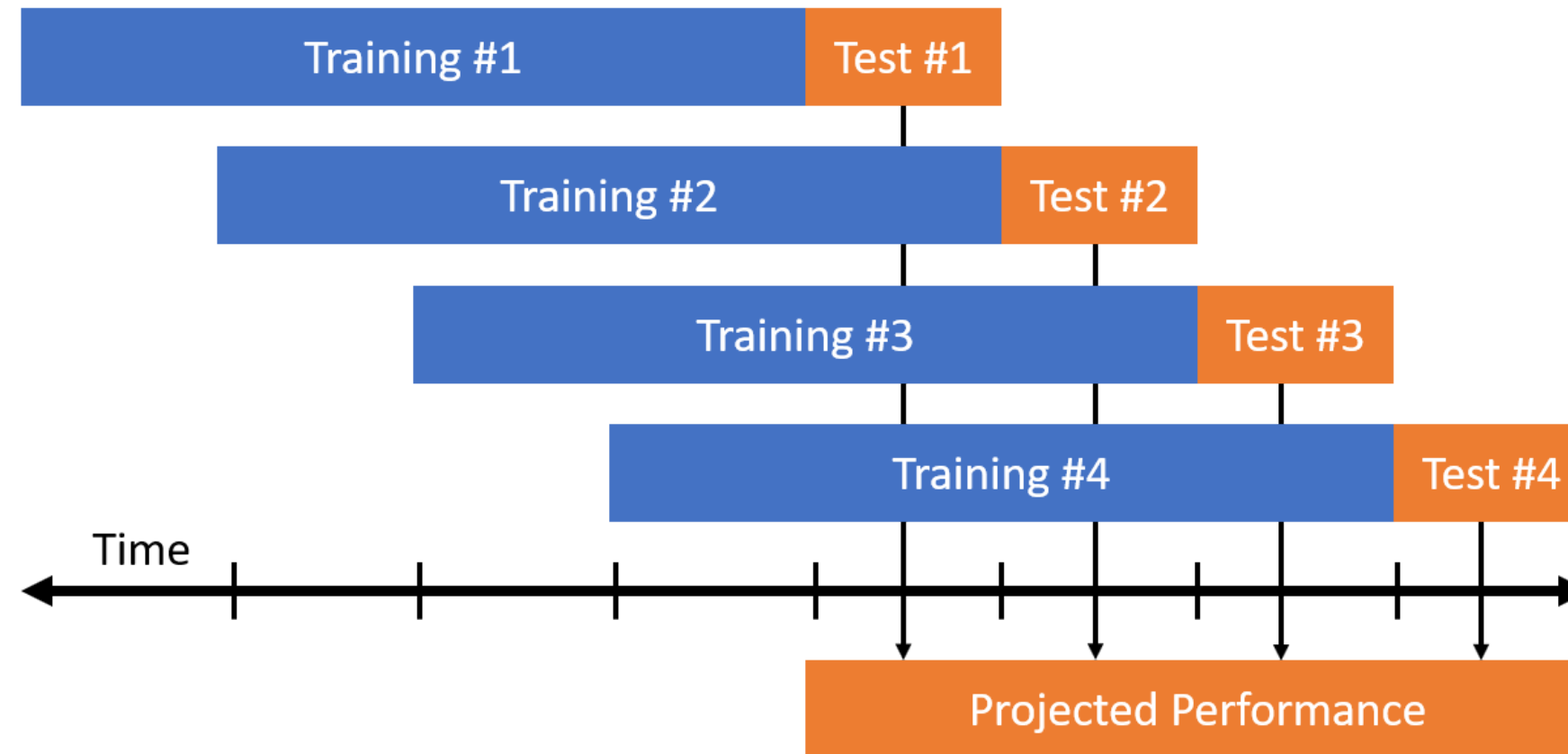
- `GeneralizedLinearRegression`
- `IsotonicRegression`
- `LinearRegression`
- `DecisionTreeRegression`
- `GBTRegression`
- `RandomForestRegression`





# Test and Train Splits for Time Series

## Walk-Forward Optimization for Time-Series



<https://www.kaggle.com/c/santander-value-prediction-challenge/discussion/61408>

# Test and Train Splits for Time Series

```
# Create variables for max and min dates in our dataset
max_date = df.agg({'OFFMKTDATE': 'max'}).collect()[0][0]
min_date = df.agg({'OFFMKTDATE': 'min'}).collect()[0][0]
```

```
# Find how many days our data spans
from pyspark.sql.functions import datediff
range_in_days = datediff(max_date, min_date)
```

```
# Find the date to split the dataset on
from pyspark.sql.functions import date_add
split_in_days = round(range_in_days * 0.8)
split_date = date_add(min_date, split_in_days)
```

```
# Split the data into 80% train, 20% test
train_df = df.where(df['OFFMKTDATE'] < split_date)
test_df = df.where(df['OFFMKTDATE'] >= split_date)\
    .where(df['LISTDATE'] >= split_date)
```

# Time to practice!

FEATURE ENGINEERING WITH PYSPARK

# Preparing for Random Forest Regression

FEATURE ENGINEERING WITH PYSPARK



**John Hogue**

Lead Data Scientist, General Mills

# Assumptions Needed for Features

## Random Forest Regression

- Skewed/Non Normal Data? OK
- Unscaled? OK
- Missing Data? OK
- Categorical Data? OK



# Appended Features

## Economic

- 30 Year Mortgage Rates

## Governmental

- Median Home Price for City
- Home Age Percentages for City
- Home Size Percentages for City

## Social

- Walk Score
- Bike Score

## Seasonal

- Bank Holidays

# Engineered Features

## Temporal Features

- Limited value with one year of data
- Holiday Weeks

## Rates, Ratios, Sums

- Business Context
- Personal Context

## Expanded Features

- Non-Free Form Text Columns
- Need to Remove Low Observations

```
# What is shape of our data?  
print((df.count(), len(df.columns)))
```

```
(5000, 126)
```

# Dataframe Columns to Feature Vectors

```
from pyspark.ml.feature import VectorAssembler
```

```
# Replace Missing values  
df = df.fillna(-1)
```

```
# Define the columns to be converted to vectors  
features_cols = list(df.columns)
```

```
# Remove the dependent variable from the list  
features_cols.remove('SALESCLOSEPRICE')
```



# Dataframe Columns to Feature Vectors

```
# Create the vector assembler transformer
vec = VectorAssembler(inputCols=features_cols, outputCol='features')

# Apply the vector transformer to data
df = vec.transform(df)

# Select only the feature vectors and the dependent variable
ml_ready_df = df.select(['SALESCLOSEPRICE', 'features'])

# Inspect Results
ml_ready_df.show(5)
```

```
+-----+-----+
| SALESCLOSEPRICE |          features|
+-----+-----+
| 143000           | (125, [0, 1, 2, 3, 5, 6...|
| 190000           | (125, [0, 1, 2, 3, 5, 6...|
| 225000           | (125, [0, 1, 2, 3, 5, 6...|
| 265000           | (125, [0, 1, 2, 3, 4, 5...|
| 249900           | (125, [0, 1, 2, 3, 4, 5...|
+-----+-----+
only showing top 5 rows
```

**We are now ready  
for machine  
learning!**

**FEATURE ENGINEERING WITH PYSPARK**

# Building a Model

FEATURE ENGINEERING WITH PYSPARK



**John Hogue**

Lead Data Scientist, General Mills

# RandomForestRegressor

## Basic Model Parameters

- `featuresCol="features"`
- `labelCol="label"`
- `predictionCol="prediction"`
- `seed=None`

## Our Model Parameter values

- `featuresCol="features"`
- `labelCol="SALESCLOSEPRICE"`
- `predictionCol="Prediction_Price"`
- `seed=42`

# Training a Random Forest

```
from pyspark.ml.regression import RandomForestRegressor
```

```
# Initialize model with columns to utilize
rf = RandomForestRegressor(featuresCol="features",
                           labelCol="SALESCLOSEPRICE",
                           predictionCol="Prediction_Price",
                           seed=42
                           )
```

```
# Train model
model = rf.fit(train_df)
```

# Predicting with a Model

```
# Make predictions
predictions = model.transform(test_df)
```

```
# Inspect results
predictions.select("Prediction_Price", "SALESCLOSEPRICE").show(5)
```

```
+-----+-----+
| Prediction_Price|SALESCLOSEPRICE|
+-----+-----+
|426029.55463222397|      415000|
| 708510.8806005502|      842500|
| 164275.7116183204|      161000|
| 208943.4143642175|     200000|
|217152.43272221283|     205000|
+-----+-----+
only showing top 5 rows
```

# Evaluating a Model

```
from pyspark.ml.evaluation import RegressionEvaluator
```

```
# Select columns to compute test error
evaluator = RegressionEvaluator(labelCol="SALESCLOSEPRICE",
                                predictionCol="Prediction_Price")
```

```
# Create evaluation metrics
rmse = evaluator.evaluate(predictions, {evaluator.metricName: "rmse"})
r2 = evaluator.evaluate(predictions, {evaluator.metricName: "r2"})
```

```
# Print Model Metrics
print('RMSE: ' + str(rmse))
print('R^2: ' + str(r2))
```

```
RMSE: 22898.84041072095
R^2: 0.9666594402208077
```

# Let's model some data!

FEATURE ENGINEERING WITH PYSPARK



# Interpreting, Saving & Loading Models

FEATURE ENGINEERING WITH PYSPARK



**John Hogue**

Lead Data Scientist, General Mills

# Interpreting a Model

```
import pandas as pd
```

```
# Convert feature importances to a pandas column  
fi_df = pd.DataFrame(model.featureImportances.toArray(),  
                      columns=[ 'importance' ])
```

```
# Convert list of feature names to pandas column  
fi_df[ 'feature' ] = pd.Series(feature_cols)
```

```
# Sort the data based on feature importance  
fi_df.sort_values(by=[ 'importance' ], ascending=False, inplace=True)
```

# Interpreting a Model

```
# Interpret results  
model_df.head(9)
```

| feature                 | importance |
|-------------------------|------------|
| LISTPRICE               | 0.312101   |
| ORIGINALLISTPRICE       | 0.202142   |
| LIVINGAREA              | 0.124239   |
| SQFT_TOTAL              | 0.081260   |
| LISTING_TO_MEDIAN_RATIO | 0.075086   |
| TAXES                   | 0.048452   |
| SQFTABOVEGROUND         | 0.045859   |
| BATHSTOTAL              | 0.034397   |
| LISTING_PRICE_PER_SQFT  | 0.018253   |

# Saving & Loading Models

```
# Save model
model.save('rfr_real_estate_model')
```

```
from pyspark.ml.regression import RandomForestRegressionModel

# Load model from
model2 = RandomForestRegressionModel.load('rfr_real_estate_model')
```

# On to your last set of exercises!

FEATURE ENGINEERING WITH PYSPARK

# Final Thoughts

FEATURE ENGINEERING WITH PYSPARK



**John Hogue**  
Lead Data Scientist

# What you learned!

- Inspecting visually & statistically
- Dropping rows and columns
- Scaling and adjusting data
- Handling missing values
- Joining external datasets
- Generating features
- Extracting variables from messy fields
- Binning, bucketing and encoding
- Training and evaluating a model
- Interpreting model results

# Time to learn something new!

FEATURE ENGINEERING WITH PYSPARK