














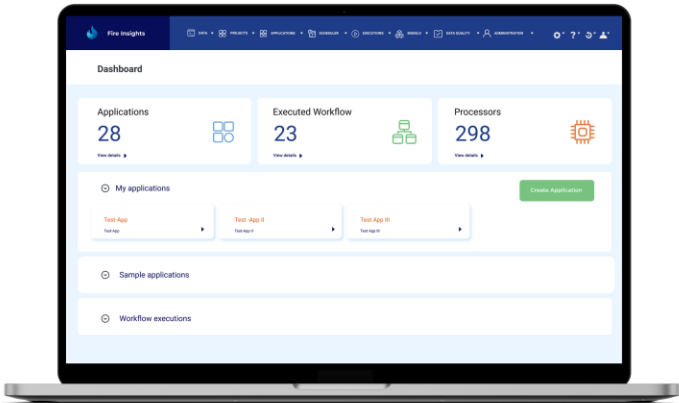
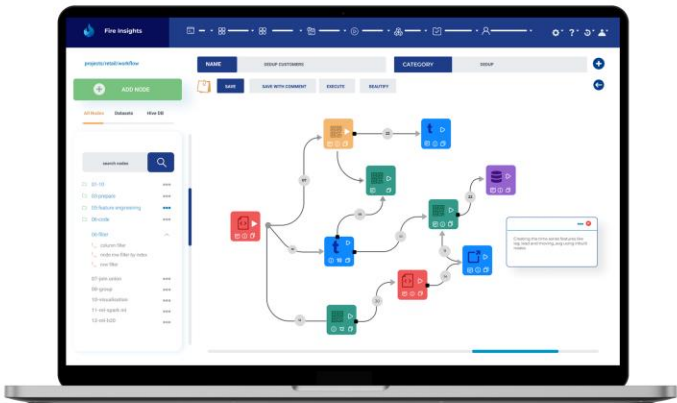




Self-Serve Advanced Analytics with  
Sparkflows on Databricks

# Sparkflows enables the following Self-Serve Capabilities on Databricks

Connectors	ETL / Data Preparation	Data Exploration / Feature Engineering	ML Modeling	Analytical Apps & Reports
 Connectors	 Joins / Unions	 Exploratory Data Analytics	 Modeling / Predictions <i>(Apache Spark ML, H2O, Scikit-learn, Prophet, ARIMA, XGBoost)</i>	 Analytical Apps
 Various File Formats	 Data Cleaning	 Feature Engineering	 Model Comparison / Export	 Reports & Dashboards
	 Data Cleaning	 Feature Scaling		
	 Filtering	 Feature Selection	 NLP	



Self-Serve Advanced Analytics  
with powerful workflows

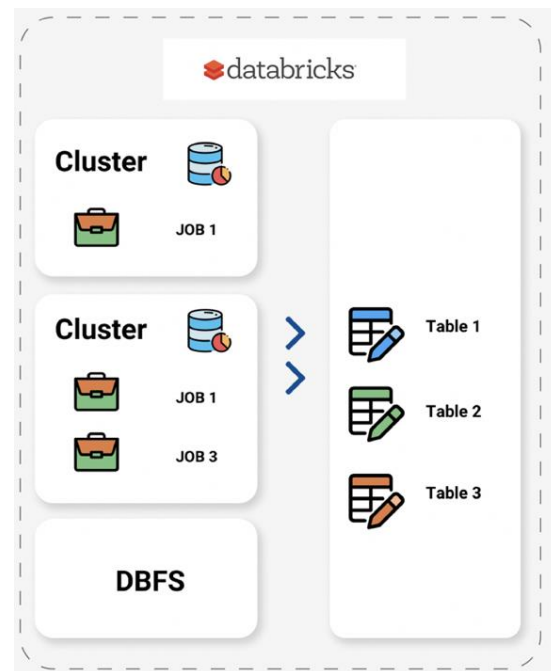
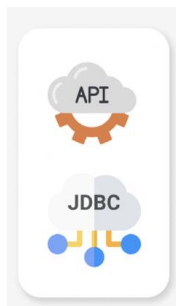
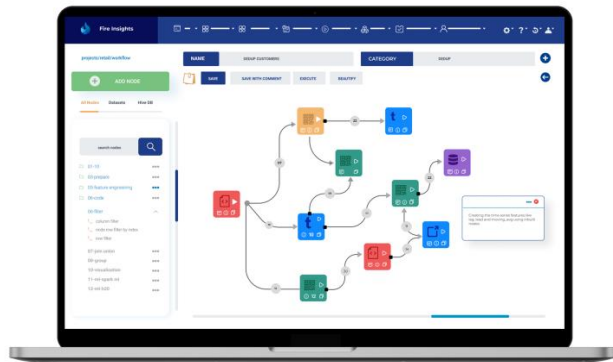
350+ Processors for connecting,  
transforming, exploring and  
building AI models

A multi-user collaborative platform  
which scales to Petabytes of data



# Sparkflows is deeply integrated with Databricks

## Sparkflows



- Sparkflows interacts with Databricks using REST API's and JDBC
- JDBC is used for getting the Databricks DB schema and fetching few records for interactive execution.
- REST API is used for submitting the job to the databricks cluster.

<https://docs.sparkflows.io/en/latest/databricks/index.html>

<https://docs.sparkflows.io/en/latest/databricks-user-guide/index.html>

# Sparkflows provides the following on Databricks



## Interact with Databricks Clusters

View, start, stop the Databricks Clusters.



## Create & Manage Databricks Connections

Connections to Databricks can be created at Global, Project and Group level.



## Integration with MLflow

Create experiments in MLflow  
Log metrics and artifacts to MLflow  
Save ML Models to MLflow



## Delta Lake

Easily Read, write and merge data into Delta Lake.



## Databricks DB

Seamlessly interact with the Databricks DB.  
Browse the Databases and Tables, run DDL and Query the tables.



## DBFS

Seamlessly interact with DBFS. Browse, Upload, Delete files and folders.



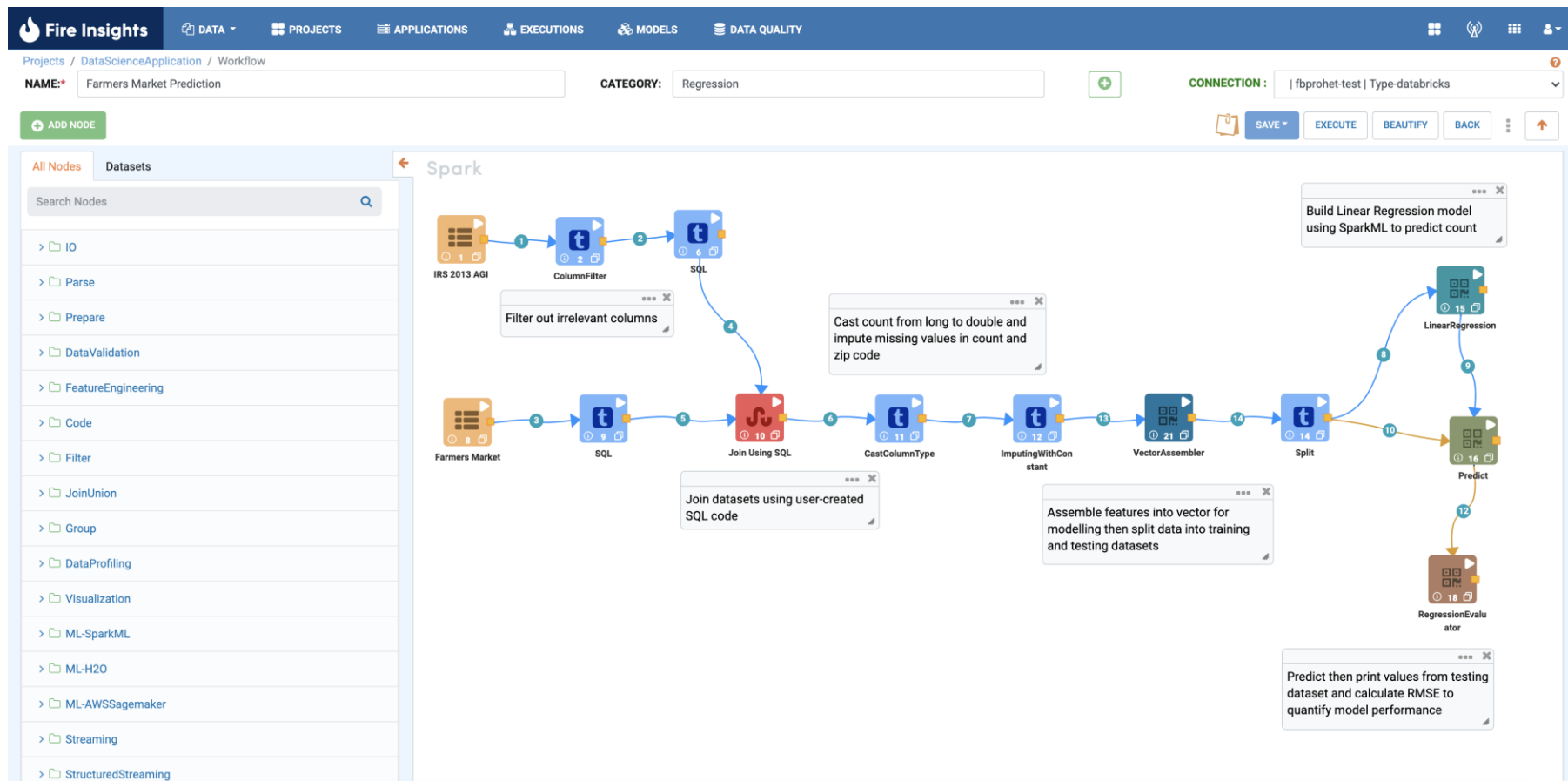
## Submit Jobs to Databricks & View Results

Submit workflows and jobs to be executed on Databricks. View the status of the jobs, results of the workflows executions in Sparkflows.

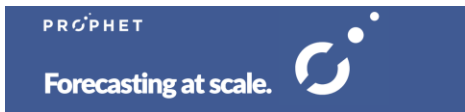
# Advantages of Sparkflows on Databricks



# Prepare, Profile, Explore & Build ML Models at Scale



# Execute various ML Algorithms on a variety of ML Engines






ARIMA





# Number of ML Algorithms supported out of the box

<div><div>Scikit Learn</div><div><b>Classification</b><ul style="list-style-type: none"><li>• Gradient Boosting Classifier</li><li>• Logistic Regression</li><li>• Random Forest Classifier</li></ul></div><div><b>Regression</b><ul style="list-style-type: none"><li>• Bayesian Ridge Regression</li><li>• Gradient Boosting Regression</li><li>• Lasso Regression</li><li>• Random Forest Regression</li><li>• Ridge Regression</li></ul></div><div><b>Evaluator</b><ul style="list-style-type: none"><li>• Regression Evaluator</li><li>• Classification Evaluator</li><li>• Custom Metrics</li></ul></div><div><b>Modeling</b><ul style="list-style-type: none"><li>• Model Predict</li><li>• Model Save</li><li>• Model Load</li></ul></div></div>	<div><div>H2O</div><div><ul style="list-style-type: none"><li>• Gradient Boosting Machine</li><li>• Generalized Linear Models</li><li>• Generalized Low Rank Models</li><li>• Distributed Random Forest</li><li>• Isolation Forest</li><li>• K-Means</li><li>• Naive Bayes</li><li>• Neural Network</li><li>• PCA</li><li>• Word to Vec</li><li>• XGBoost</li></ul></div></div>	<div><div>Spark ML</div><div><b>Clustering :</b><ul style="list-style-type: none"><li>• K-Means Clustering</li><li>• LDA</li><li>• Gaussian Mixture</li></ul></div><div><b>Regression</b><ul style="list-style-type: none"><li>• AFT Survival Regression</li><li>• Decision Tree Regression</li><li>• GBT Regression</li><li>• Linear Regression</li><li>• Random Forest Regression</li><li>• XGBoost Regression</li></ul></div><div><b>Classification</b><ul style="list-style-type: none"><li>• Decision Tree Classifier</li><li>• GBT Classifier</li><li>• Logistic Regression</li><li>• MultiLayer Perceptron</li><li>• Naive Bayes</li><li>• Random Forest Classifier</li><li>• XGBoost Classifier</li></ul></div><div><b>Collaborative Filtering:</b></div></div>
---	--	--

# Connect to various Data Sources

## Connectors

ORACLE

Oracle

SAP HANA

HANA



Databricks



Redshift



Snowflake



MySQL



IBM Netezza



Google Big Query



HP Vertica



IBM DB2



Azure HDInsights



HIVE



Elastic Search



Apache HBASE



Cassandra



Apache Kafka

# Read various File Formats

## File Formats



CSV



JSON



XML



Apache Parquet



Excel



Avro



Delta

LibSVM

LIVSBM



Shape File



Binary



Text File



Whole Text Field

# Powerful Data Preparation



Join



Union



Filter



Date Time



Parse



Data Cleaning



Math



String



Split



Code



Condition



Group

## Parse



- Apache Log
- Field Splitter
- Fixed Length Fields
- Multi Regex Extractor
- Parse JSON Col
- Regex Tokenizer
- OCR

## Join/Union



- Geo Join
- Join on Columns
- Join using SQL
- Union All
- Union Strict
- Join on Common Column
- Join on Common Columns
- Union Distinct

## Group



- Cube
- Group By
- Pivot By
- Roll up

## Date- Time



- Date Difference
- Date Time Field Extract
- Date to String
- String to Unix time
- Time Functions
- Unix time to string
- String to Date

## Data Cleaning



- Data Wrangling
- Dedup
- Drop Duplicate Rows
- Drop Rows with Null
- Find and replace Using Regex Multiple
- Imputing with constant
- Imputing with a mean value
- Imputing with mode value
- Remove Duplicate rows
- Remove unwanted characters
- Imputing with Median
- Find and Replace Using Regex
- Remove Unwanted characters Multiple

## Code



- SQL
- SQL Executor
- Python
- Pipe Python
- Pipe Python 2
- Scala
- Scala VDF
- Jython
- Pyspark
- MultiInput Pyspark
- MultiInput To MultiOutput Pyspark
- Run Hive QL
- Unix Shell Commands

## Filter



- Drop Columns
- Select Columns
- Filter by Date Range
- Row Filter
- Filter By String Length
- Filter By Number Range

## Math



- Math Expression
- Math Functions Multiple

## String



- String Functions
- String Functions Multiple
- Text Case Transformer

## Split



- Compare All Columns
- Compare All Columns Single Output
- Compare Specific Columns
- Split By Expression
- Split by Multiple Expressions

## Condition



- Assert
- Decision

## Cast Data Type



- CastColumnType
- CastMultipleColumnType

## Add Column



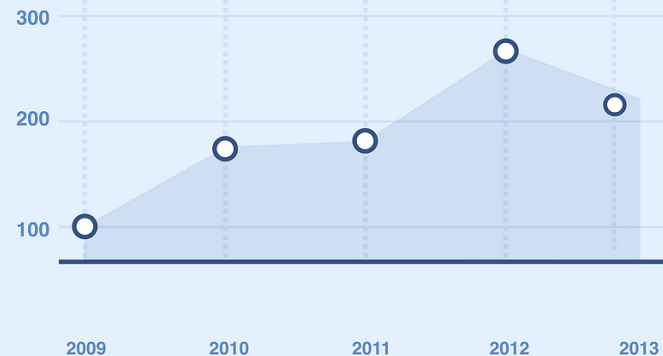
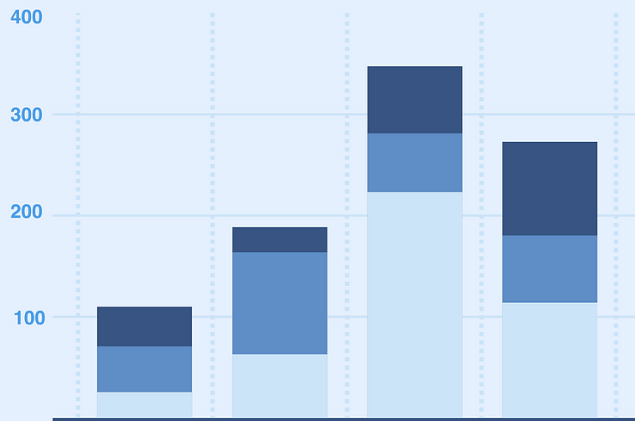
- Add columns
- Case When
- Concat Columns
- Expressions
- Generate UUID
- Generate UID
- Hash
- Zip with Index

## Others



- CDC Using Full Table Merge
- Columns Rename
- Count
- Geo IP
- Geo Point
- Multi Window Analytics
- Multi Window Ranking
- Recover Hive Partitions
- Register Temp Table
- Round Value
- Sample
- Sort By
- Sort Columns
- Transpose
- Window Analytics
- Window Ranking

# Explore & Visualize data at Scale



Histogram



Correlation matrix



Statistics



Charts



Box Plot



Line Chart



Bubble Chart



Gauge



Graph Group by Column



Graph Region Geo



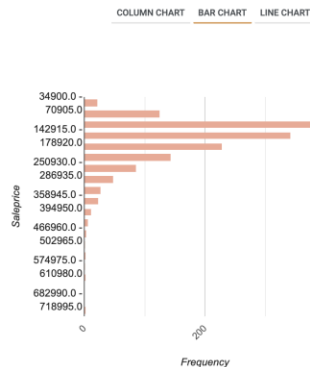
Graph Subplots



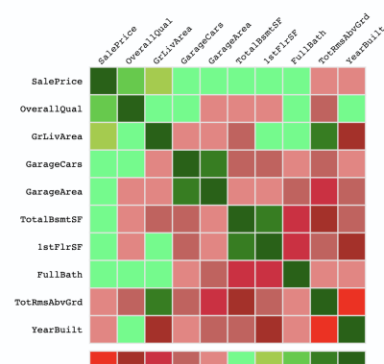
Graph Values

# Reports, Charts & Dashboards

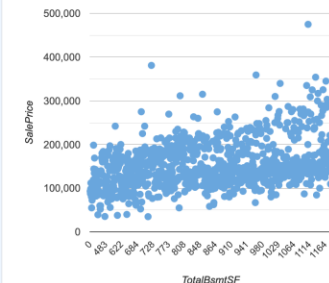
Histogram



Correlation Matrix

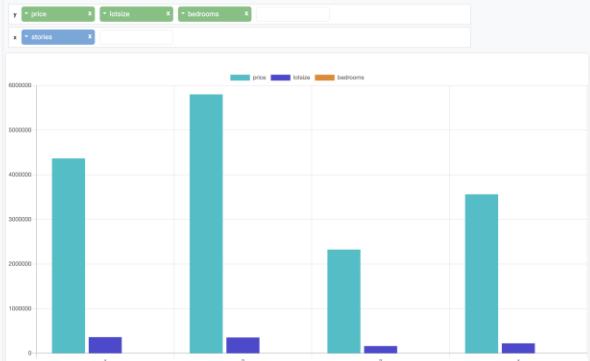


TotalBsmntSF vs SalePrice

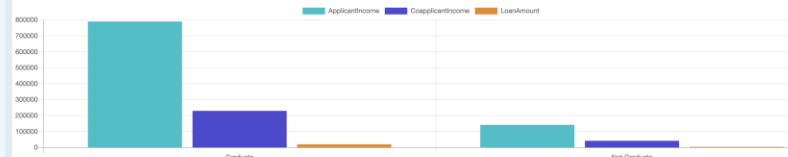


- Dataset
- COUNT
- id
- price
- lotsize
- bedrooms
- bathrooms
- stories
- driveway
- hvacroom
- fullbath
- garage
- area
- geogrid
- prefarea

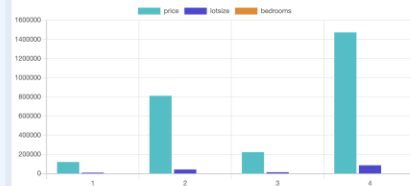
- Filters
- price
- lotsize
- bedrooms
- stories
- driveway
- hvacroom
- fullbath
- garage
- area
- geogrid
- prefarea



Loan chart Analysis



Housing Price vs Lotsize



Housing chart



- Filters
- Married
- Search...
- NO (213)
- YES (196)
- 0
- APPLY GLOBAL
- Dependents
- Search...
- 0 (344)
- 1 (102)
- 2 (100)
- 3+ (5)
- 0
- APPLY GLOBAL
- price
- 106132
- 121424
- APPLY GLOBAL

# Data Profiling



Columns  
Cardinality



Correlation



Cross tab



Distinct values  
in Column



Flag Outlier



Graph-Month  
Distribution



Graph-Year  
Distribution



Graph-Weekday  
Distribution



Graph-Year  
Distribution



Histogram



Null Values  
in Column



Summary  
Statistics

## Schedule Jobs or Trigger them



By Time



By Event

## Security



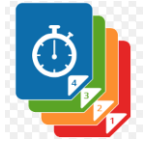
Kerberos

**Apache Ranger**

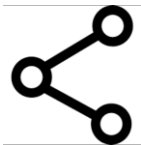


User Impersonation

## Versioning & Sharing of Workflows



Versioning



Sharing  
Application  
with Groups



Lock  
Workflows

## Git Integration



## Engines

Streaming

Batch

# Code in Workflows

SQL

SQL

python

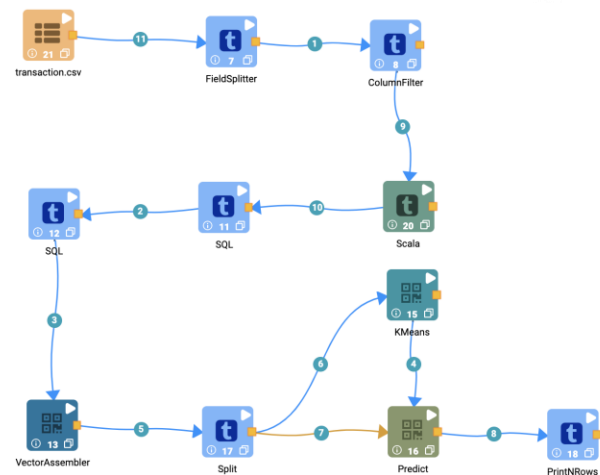
python

Scala

Scala

Jython

Jython



TEMP TABLE: ?

SQL \*: ?

```
1 select customer_id, rValue, fValue, mValue,
2 case when rValue <= 35 then 3.0
3     when rValue > 35 and rValue <= 112 then 2.0
4     else 1.0
5     end as rScore,
6 case when fValue >= 15 then 3.0
7     when fValue >= 4 and fValue < 15 then 2.0
8     else 1.0
9     end as fScore,
10 case when mValue >= 50000.0 then 3.0
11     when mValue >= 25000.0 and mValue < 50000.0 then 2.0
12     else 1.0
13     end as mScore,
14 mValue / fValue as avg_amount from fire_temp_table
```

SCALA: ?

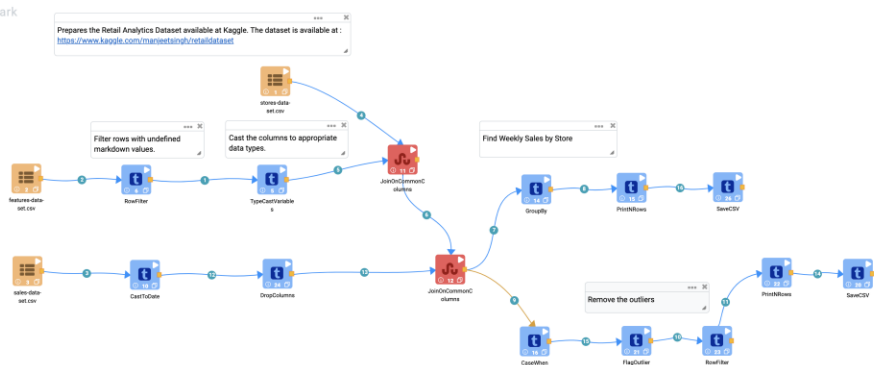
```
1 import org.apache.spark.sql.functions.{col,
2
3 val preprocess = inDF.select(col("customer_id"), col("amount"), to_date(col("dt")).as("t_date"))
4     .withColumn("current_date", current_date)
5     .withColumn("trans", lit(1))
6     .withColumn("diff", datediff(col("current_date"), col("t_date")))
7
8 val outDF = preprocess.groupBy(col("customer_id")).agg(Map("amount" -> "sum", "trans" -> "sum", "diff" -> "min"))
9     .select(col("customer_id"), col("sum(amount)").as("mValue"), col("sum(trans)").as("fValue"), col("min(diff)").as("rValue"))
10
11 outDF.registerTempTable("outDF")
```

PYSPARK: ?

```
16
17 def myfn(spark: SparkSession, workflowContext: WorkflowContext, id: int, inDF: DataFrame, cust_dict):
18     # Convert the Spark DataFrame to a Pandas DataFrame using Arrow
19     dataset = inDF.select("*").toPandas()
20
21     print(dataset.head())
22     print(dataset.shape)
23     print(dataset.describe())
24     dataset = dataset.fillna(method='ffill')
25
26     X = dataset[
27         ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide',
28          'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol']].values
29     print(X)
30
31     y = dataset['quality'].values
32     print(y)
33
34     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```



# Generate Pyspark code from Workflows



## Pyspark Code

```
12 #ReadCSV
13
14 df_2_ReadCSV= spark.read.format("csv").option("header", "true").option("sep", ",").load("s3a:/
15
16 #RowFilter
17
18 df_6_RowFilter= df_2_ReadCSV.filter("MarkDown1 != 'NA' OR MarkDown2 != 'NA' OR MarkDown3 != 'NA'
19
20 #TypeCastVariables
21
22 from pyspark.sql.functions import *
23 df_cast = df_6_RowFilter.withColumn("Date-new", to_date(from_unixtime(unix_timestamp(col("Date
24 df_5_TypeCastVariables=df_cast
25
26 from pyspark.sql.functions import *
27 df_cast = df_6_RowFilter.withColumn("Date-new", to_date(from_unixtime(unix_timestamp(col("Date
28 df_5_TypeCastVariables=df_cast
29 .withColumn("MarkDown1-new", df_6_RowFilter["MarkDown1"].cast(DoubleType)).drop("MarkDown1").w
30 df_5_TypeCastVariables=df_cast
31
32 from pyspark.sql.functions import *
33 df_cast = df_6_RowFilter.withColumn("Date-new", to_date(from_unixtime(unix_timestamp(col("Date
34 df_5_TypeCastVariables=df_cast
35 .withColumn("MarkDown1-new", df_6_RowFilter["MarkDown1"].cast(DoubleType)).drop("MarkDown1").w
36 df_5_TypeCastVariables=df_cast
37 .withColumn("MarkDown2-new", df_6_RowFilter["MarkDown2"].cast(DoubleType)).drop("MarkDown2").w
```

COPY TO CLIPBOARD

DOWNLOAD

CLOSE

# Model Browser

MODELS

All

Regression

Classification
















Clustering

H2O

Spark ML

Created

Production

ID	USER NAME	PROJECT ID	WORKFLOW NAME	ACTION	WORKFLOW EXECUTION ID	MODEL UUID
36	sparkflows	50	Sklearn Model Save ...	  	149	008d82da-7ac5-1
35	sparkflows	50	Sklearn Model Save ...	  	148	83862670-7ac4-1
34	sparkflows	50	Sklearn Model Save ...	  	147	7ab919cc-7ac3-1
33	sparkflows	50	house price - ridge r...	  	100	416d053e-78f2-1
2	sparkflows	49	Churn Prediction - R...	  	40	83c3758d-be31-4

### Test Metrics

Accuracy	0.932527693856999
Area Under ROC	0.9096353750199899
Area Under PR	0.822052227636293

### Confusion Matrix

TARGET_LABEL	PREDICTED_LABEL	COUNT
1.0	1.0	87
0.0	1.0	6
1.0	0.0	61
0.0	0.0	839

Fire Insights

DATA

PROJECTS

APPLICATIONS

EXECUTIONS

MODELS

DATA QUALITY

Models / Model Details

MODEL INFORMATION

Model Summary

Features

MODEL INTERPRETATION

Train Metrics

Features Importance

MODEL EVALUATION

Test Metrics

MODEL INFO

ID	NAME	ALGORITHM
2	Churn Prediction - RFC	Spark RandomForestClassifier

Model Path

Model has not been persisted


Model Summary

FeatureSubsetStrategy	auto
Impurity	gini
CacheNodeIds	false
MaxBins	32
MaxDepth	5
MaxMemoryInMB	256
MinInfoGain	0.0
MinInstancesPerNode	1
NumTrees	20
Seed	207336481
SubsamplingRate	1.0
LabelCol	label


# Accelerate model building with AutoML




Select Dataset



Auto profile



Auto model creation



Leaderboard & Deploy

## CONFIGURE AUTOML

SAVE

START

BACK

\* NAME:

Diabetes Patient Experiment

\* ML PROBLEM TYPE:

Classification

\* DATASET:

BROWSE

Patient\_diabetes

\* TARGET COLUMN:

Outcome

\* EVALUATION METRICS:

accuracy

STOPPING CONDITION:

\* TIMEOUT(MINUTES):

10

\* NUMBER OF TRIAL RUNS:

2

SCHEMA:

	NAME	DATA TYPE	FORMAT
1	Pregnancies	INTEGER	
2	Glucose	INTEGER	
3	BloodPressure	INTEGER	
4	SkinThickness	INTEGER	
5	Insulin	INTEGER	
6	BMI	DOUBLE	
7	DiabetesPedigreeFunction	DOUBLE	
8	Age	INTEGER	
9	Outcome	INTEGER	

# Build Analytical Apps dynamically



Build Workflow or Notebook



Build Pages dynamically



Execute with different selections



View Result

Leadership Details

Company Details

More Company Details

Run

LEADERSHIP DETAILS

EDUCATION LEVEL

☐ PHD
 ☐ MASTERS
 ☐ BACHELORS
 ☐ HIGH SCHOOL

GENDER

☐ MALE
 ☐ FEMALE

SELECT AGE (YEARS)

☐ 25 - 30
 ☐ 30 - 35
 ☐ 35 - 40
 ☐ 40 - 50
 ☐ 50 +

NEXT



Thank You!