SparkR Hands-On

Mai H. Nguyen, Ph.D.

SparkR

- R package that provides frontend to use Spark from R
- Supports distributed machine learning using R API
- Allows R script to connect to Spark cluster
- Can use with R shell or RStudio or other R IDEs.

SparkR

- **Uses MLlib for machine learning functionality**
- **Familiar R syntax:**
  - Read contents of file into a Spark dataframe
    - newdata <- read.df (“data.txt”, source=“csv”)
  - R formula operators for model fitting:
    - model <- spark.randomForest(training, label ~ features, "classification", numTrees = 10)
  - Get summary of fitted model
    - summary(model)
  - Apply model to make predictions
    - predictions <- predict(model, testDF)
  - Save model
    - write.ml (model, “mymodel”)

SparkR Hands-On

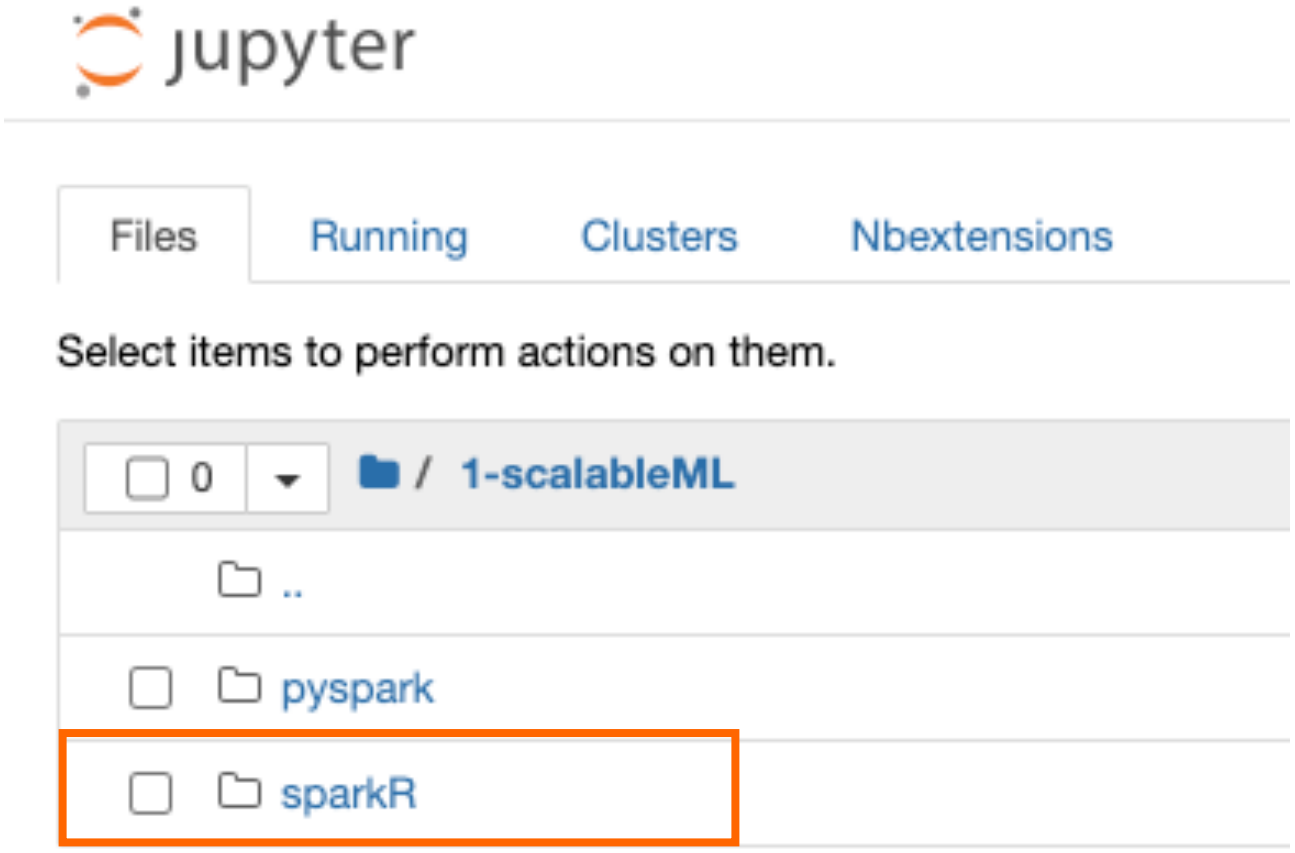
- **Data Exploration**
  - Using weather data
  - Similar to PySpark hands-on, except using SparkR
    - Load into Spark DataFrame
    - Describe schema
    - Show summary statistics
    - Calculate correlation between features
- **Classification**
  - Predict wine quality
  - Model: random forest

Dataset for Data Exploration

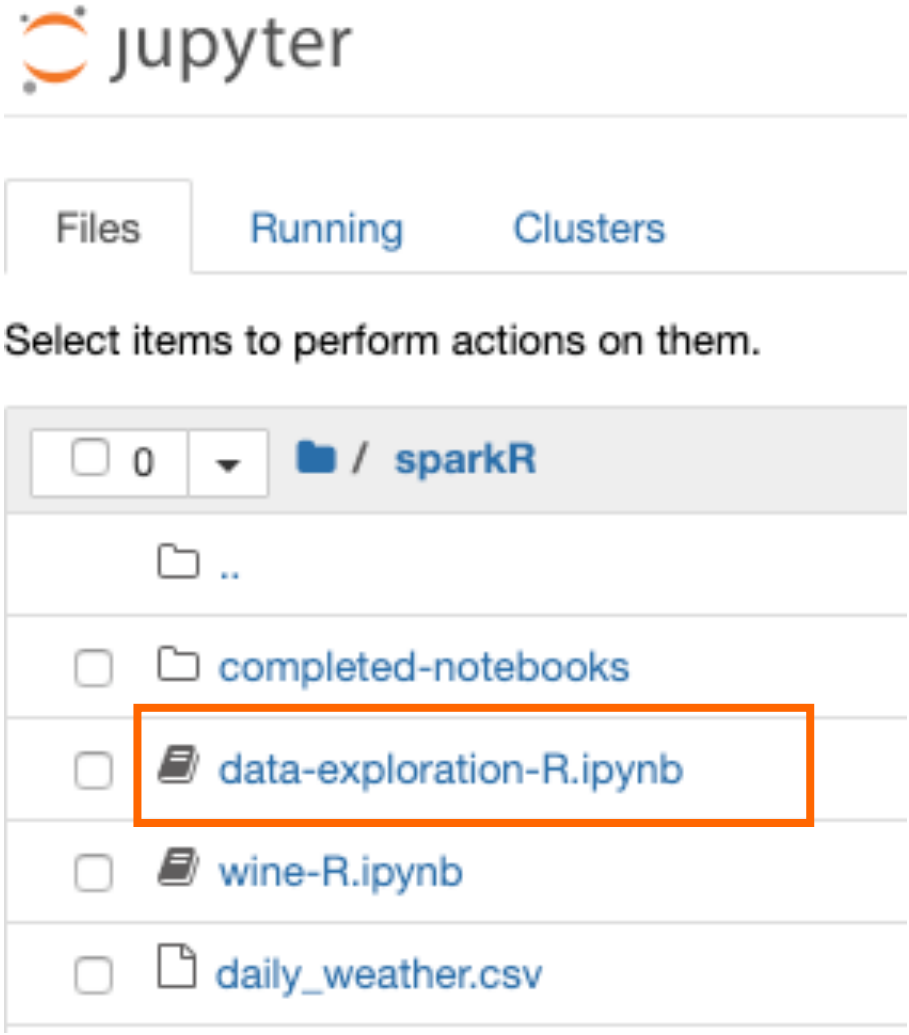
- **Measurements from weather station on Mt. Woodson, San Diego**
- **Air temperature, humidity, wind speed, wind direction, etc.**
- **Three years of data: Sep. 2011 - Sep. 2014**
- ***minute_weather.csv***
  - measurement every minute
- ***daily_weather.csv***
  - aggregated measurements

Data Exploration

Go to sparkR folder



Open SparkR Data Exploration Notebook



SparkR Setup

- Load Spark R library

```
library(SparkR, lib.loc = c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib")))
```

- Start up Spark session

```
sparkR.session(master="local[*]",
               sparkConfig=list(spark.driver.memory="2g"),
                                (spark.app.name="SparkR EDA"))
```

Read in Data

- Read data into a Spark DataFrame
- Cache Spark DF

```
sdf <- read.df("daily_weather.csv", "csv", header="true", inferSchema="true")
```

```
cache(sdf)
```

Fill in file name

Examine Schema

schema(sdf)

```
StructType
|-name = "number", type = "IntegerType", nullable = TRUE
|-name = "air_pressure_9am", type = "DoubleType", nullable = TRUE
|-name = "air_temp_9am", type = "DoubleType", nullable = TRUE
|-name = "avg_wind_direction_9am", type = "DoubleType", nullable = TRUE
|-name = "avg_wind_speed_9am", type = "DoubleType", nullable = TRUE
|-name = "max_wind_direction_9am", type = "DoubleType", nullable = TRUE
|-name = "max_wind_speed_9am", type = "DoubleType", nullable = TRUE
|-name = "rain_accumulation_9am", type = "DoubleType", nullable = TRUE
|-name = "rain_duration_9am", type = "DoubleType", nullable = TRUE
|-name = "relative_humidity_9am", type = "DoubleType", nullable = TRUE
|-name = "relative_humidity_3pm", type = "DoubleType", nullable = TRUE
```

Show Summary Statistics

head(summary(sdf))

summary	number	air_pressure_9am	air_temp_9am
count	1095	1092	1090
mean	547.0	918.8825513138094	64.93300141287072
stddev	316.24357700987383	3.184161180386833	11.17551400317587 7
min	0	907.9900000000024	36.752000000000685
25%	273	916.550000000009	57.272000000000354
50%	547	918.9020905167166	65.69599999999856

Get Number of Rows

nrow(sdf)

1095

Show First Few Rows

head(sdf)

number	air_pressure_9am	air_temp_9am	avg_wind_direction_9am	avg_wind_speed_9am
0	918.0600	74.82200	271.1000	2.080354
1	917.3477	71.40384	101.9352	2.443009
2	923.0400	60.63800	51.0000	17.067852
3	920.5028	70.13889	198.8321	4.337363
4	921.1600	44.29400	277.8000	1.856660
5	915.3000	78.40400	182.8000	9.932014

Names and Number of Columns

names(sdf)

'number' 'air_pressure_9am' 'air_temp_9am' 'avg_wind_direction_9am' 'avg_wind_speed_9am' 'max_wind_direction_9am' 'max_wind_speed_9am' 'rain_accumulation_9am' 'rain_duration_9am' 'relative_humidity_9am' 'relative_humidity_3pm'

---

ncol(sdf)

11

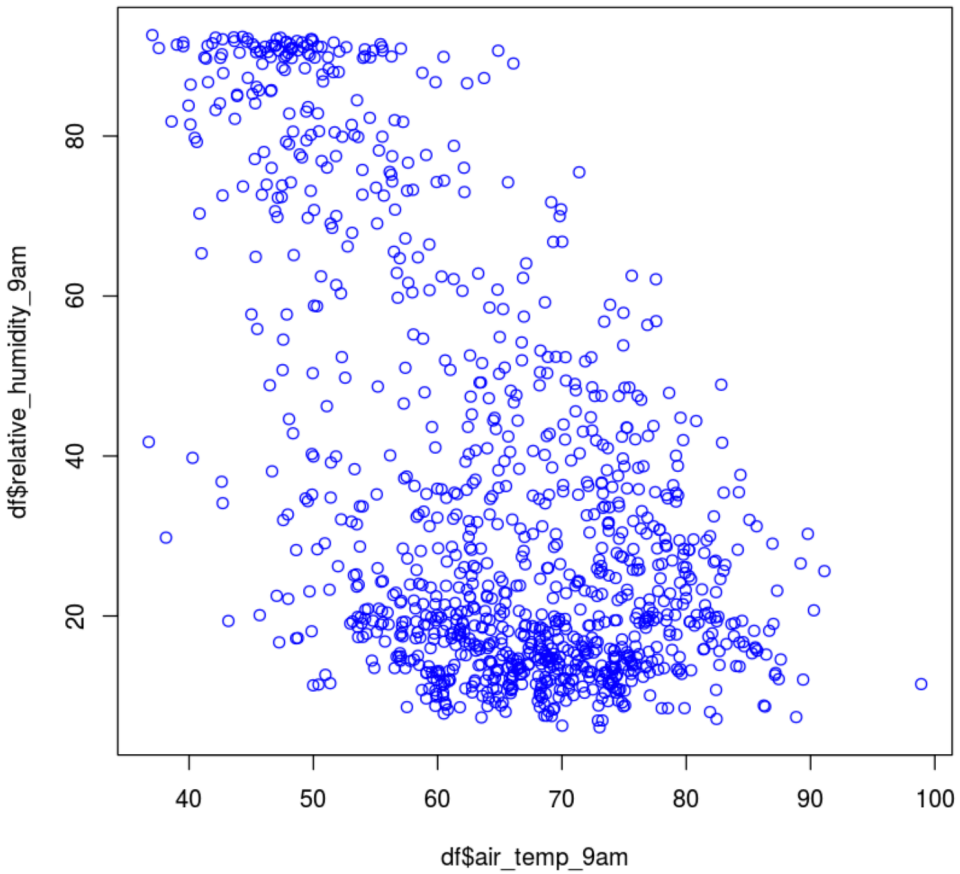
Correlation Between Air Temperature and Relative Humidity

corr(sdf, “air_temp_9am”, “relative_humidity_9am”, method=“pearson”)

-0.536670…

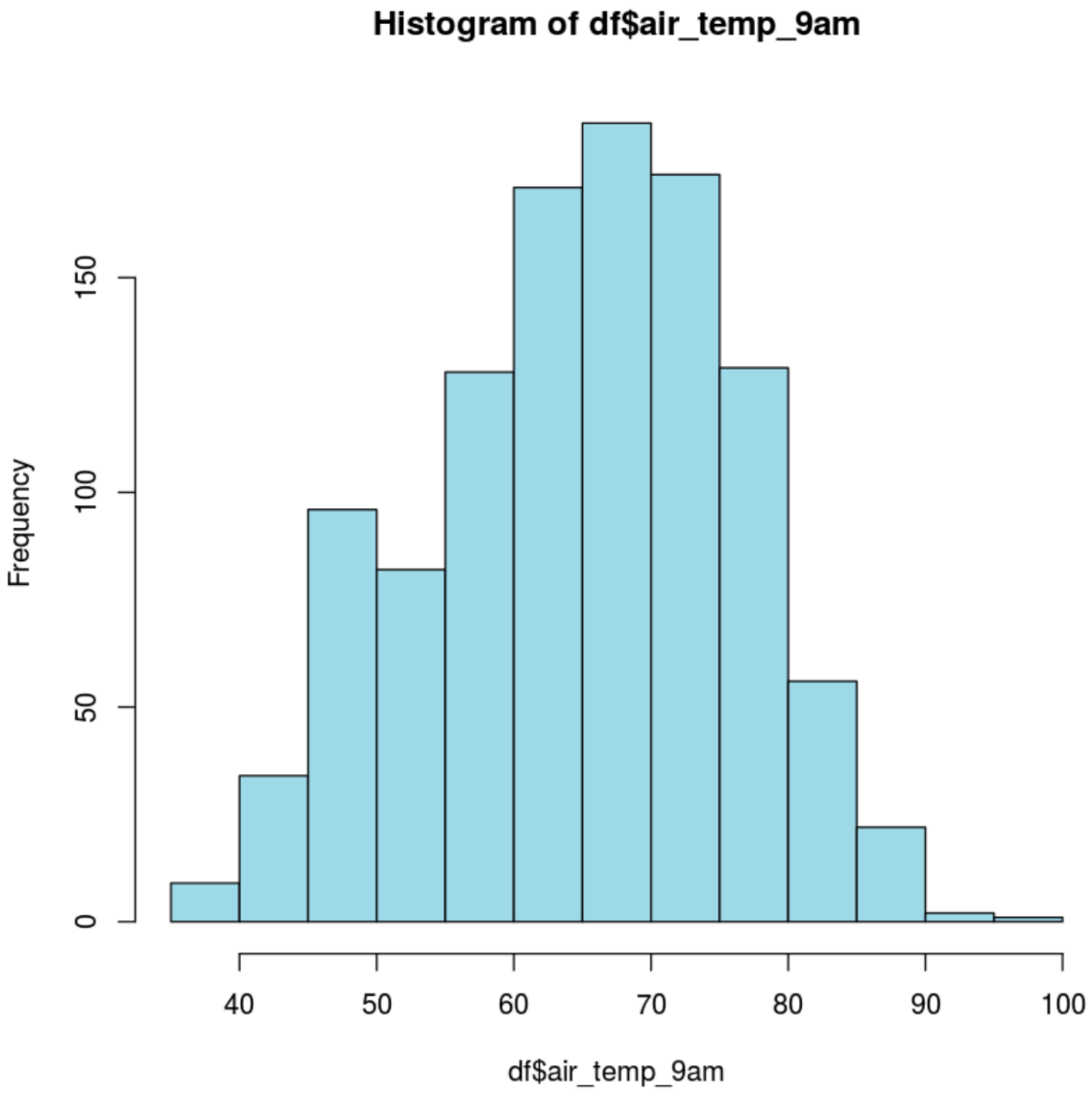
Scatter Plot of Air Temperature vs. Humidity (*)

plot(dfair_temp_9am, dfrelative_humidity_9am, col="blue")



hist(df$air_temp_9am, col="lightblue")

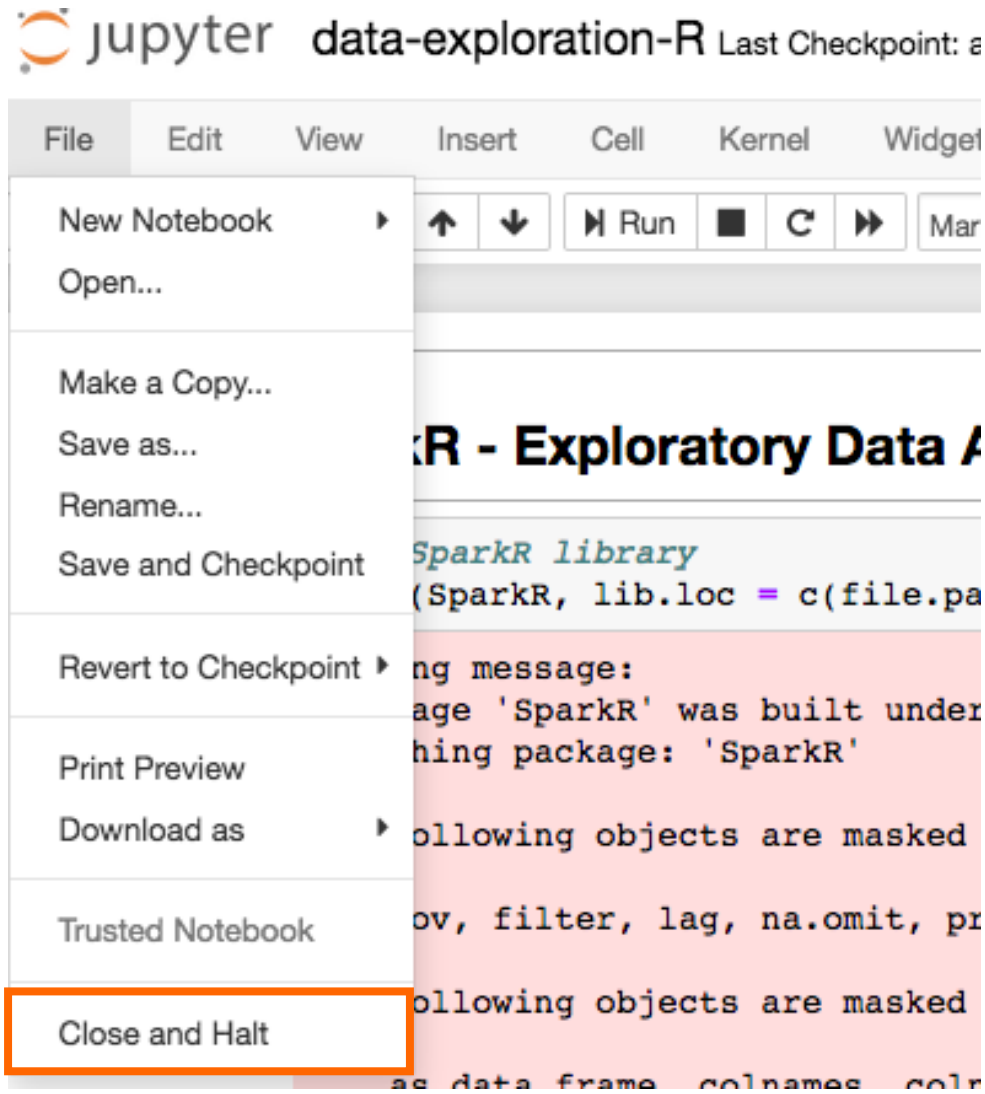
Histogram of Air Temperature



Stop Spark Session

sparkR.stop()

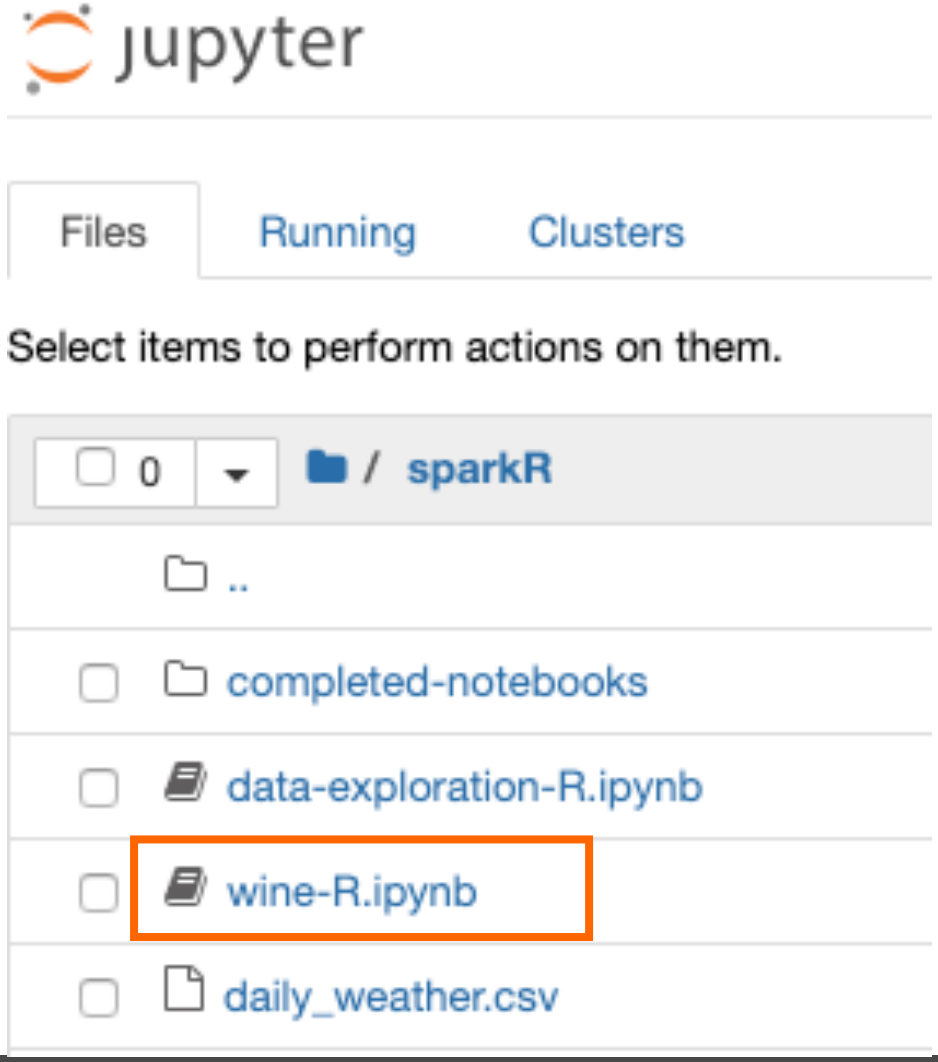
Exit Notebook



SparkR Classification Example

- **Task:**
  - Predict wine quality
- **Data:**
  - Wine dataset from UCI Machine Learning Repository
  - Features: fixed acidity, citric acid, total sulfur dioxide, etc.
  - Target:
    - Good (quality is 7-10)
    - Average (quality is 6)
    - Bad (quality is 0-5)
- **Approach:**
  - Random forest

Open wine-R Notebook



Read in Data

- Read data into a Spark DataFrame

```
sdf <- read.df("winequality-white.csv", "csv", header="true", inferSchema="true")
```

Fill in file name

- Cache Spark DF

```
cache(sdf)
```

Examine Schema

schema(sdf)

```
StructType
 |-name = "id", type = "IntegerType", nullable = TRUE
 |-name = "fixed_acidity", type = "DoubleType", nullable = TRUE
 |-name = "volatile_acidity", type = "DoubleType", nullable = TRUE
 |-name = "citric_acid", type = "DoubleType", nullable = TRUE
 |-name = "residual_sugar", type = "DoubleType", nullable = TRUE
 |-name = "chlorides", type = "DoubleType", nullable = TRUE
 |-name = "free_sulfur_dioxide", type = "DoubleType", nullable = TRUE
 |-name = "total_sulfur_dioxide", type = "DoubleType", nullable = TRUE
 |-name = "density", type = "DoubleType", nullable = TRUE
 |-name = "pH", type = "DoubleType", nullable = TRUE
 |-name = "sulphates", type = "DoubleType", nullable = TRUE
 |-name = "alcohol", type = "DoubleType", nullable = TRUE
 |-name = "quality", type = "StringType", nullable = TRUE
```

Prepare Data

```
seed <- 12345
train_sdf <- sample(sdf,
                    withReplacement=FALSE,
                    fraction=0.7, seed=seed)
test_sdf <- except (sdf, train_df)
dim(train_sdf)
dim(test_sdf)
```

⇒3406 13

⇒1492 13

Train Random Forest Model

```
model <- spark.randomForest(train_sdf, quality ~ .,
          type="classification", numTrees=30, seed=seed)
```

$formula

'quality ~ .'

$numFeatures

12

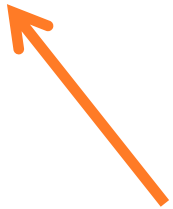
$features

1. 'id'
2. 'fixed_acidity'
3. 'volatile_acidity'
4. 'citric_acid'
5. 'residual_sugar'

Apply Model to Test Data

predictions_sdf <- predict(model, test_df)

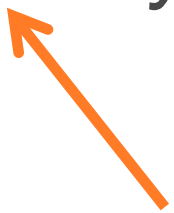
class(predictions_sdf)



=> 'SparkDataFrame'

Calculate Accuracy

```
accuracy <- mean(predictions_df$quality ==
                 predictions_df$prediction)
sprintf ("Accuracy on Test Data:  %f", accuracy)
```



=> 'Accuracy on Test Data: 0.605898'

Wine Classification Results

table(predictions_df$quality, predictions_df$prediction)

	average	bad	good
average	444	164	73
bad	167	337	8
good	165	11	123

Accuracy on Test Data: 0.605898

Save Model

write.ml (model,

“wine-model-sparkR”, 

overwrite=TRUE)

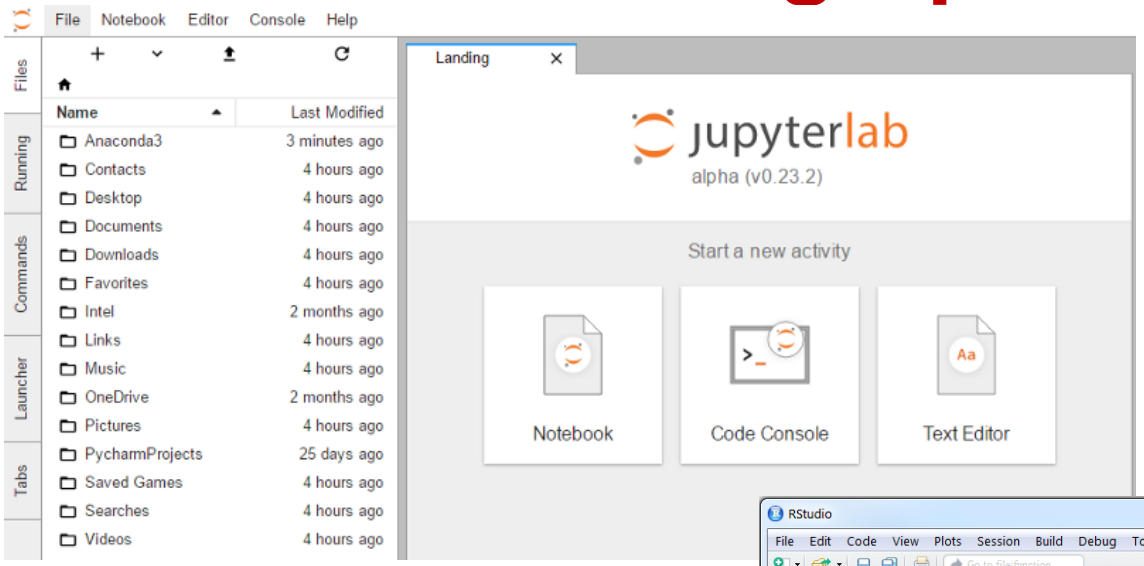
Stop Spark Session

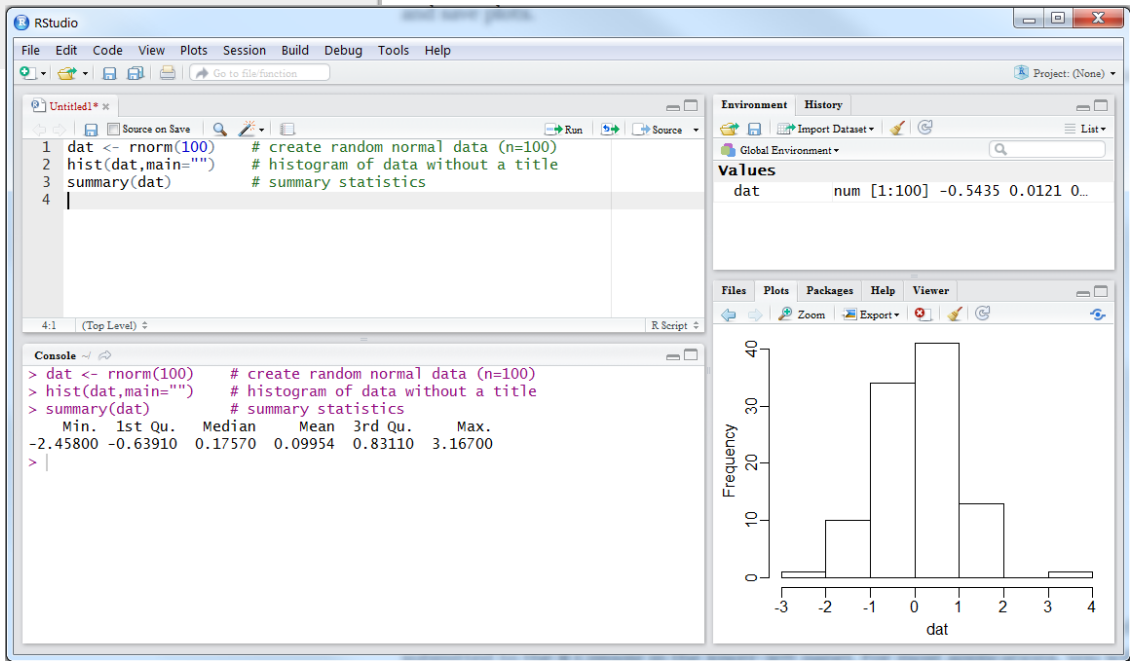
sparkR.stop()

Clean Up

- **Exit notebook**
  - File -> Close and Halt
- **Exit Jupyter Notebook**
  - Click on ‘Logout’

Running SparkR Code





RStudio®

sparklyr

- **R interface for Spark**
- **R Legacy**
  - From Rstudio
  - Available on CRAN
- **Spark backend to dplyr and SQL**
  - Interactively manipulate Spark data using dplyr and SQL
- **Access to Spark functionality**
  - Interface to Spark MLlib algorithms
- **Connect to Spark clusters**

sparklyr

- **Setup**
  - install.packages("sparklyr")
  - library(sparklyr)
  - spark_install ()
- **Connect to Spark**
  - sc <- spark_connect (master="local")
- **Using dplyr**
  - flights_sdf %>% group_by(tailnum)
    %>% filter(count > 20)
- **Machine Learning**
  - model <- ml_random_forest (
    train_sdf, quality ~ ., type="classification")

sparklyr Functions

- **Spark Operations**
  - Manage Spark connections (e.g., spark_config())
- **Spark Data Manipulation**
  - Read data in Spark DataFrame and perform operations (e.g., spark_read_csv(), tbl_cache())
- **Feature Transformers**
  - Transform data (e.g., ml_pca(), ft_bucketizer())
- **Distributed Machine Learning**
  - Access Spark Mllib algorithms (e.g., ml_kmeans())
- **Streaming**
  - Support streaming data operations (e.g., stream_read_json())
- **Extensions**
  - Interface to platforms for big data analysis, graph analytics, production

Machine Learning Algorithms in SparkR

Machine Learning

Algorithms

SparkR supports the following machine learning algorithms currently:

Classification

- spark.logit: Logistic Regression
- spark.mlp: Multilayer Perceptron (MLP)
- spark.naiveBayes: Naive Bayes
- spark.svmLinear: Linear Support Vector Machine

Regression

- spark.survreg: Accelerated Failure Time (AFT) Survival Model
- spark.glm or glm: Generalized Linear Model (GLM)
- spark.isoreg: Isotonic Regression

Tree

- spark.gbt: Gradient Boosted Trees for Regression and Classification
- spark.randomForest: Random Forest for Regression and Classification

Clustering

- spark.bisectingKmeans: Bisecting k-means
- spark.gaussianMixture: Gaussian Mixture Model (GMM)
- spark.kmeans: K-Means
- spark.lda: Latent Dirichlet Allocation (LDA)

Collaborative Filtering

- spark.als: Alternating Least Squares (ALS)

Frequent Pattern Mining

- spark.fpGrowth : FP-growth

Statistics

- spark.kstest: Kolmogorov-Smirnov Test

Spark in R Resources

- SparkR
  - https://spark.apache.org/docs/latest/sparkr.html
- SparkR Tutorial at useR 2016
  - https://databricks.com/blog/2016/07/07/sparkr-tutorial-at-user-2016.html
- SparkR API
  - https://spark.apache.org/docs/latest/api/R/
- sparklyr
  - https://spark.rstudio.com/

Questions?