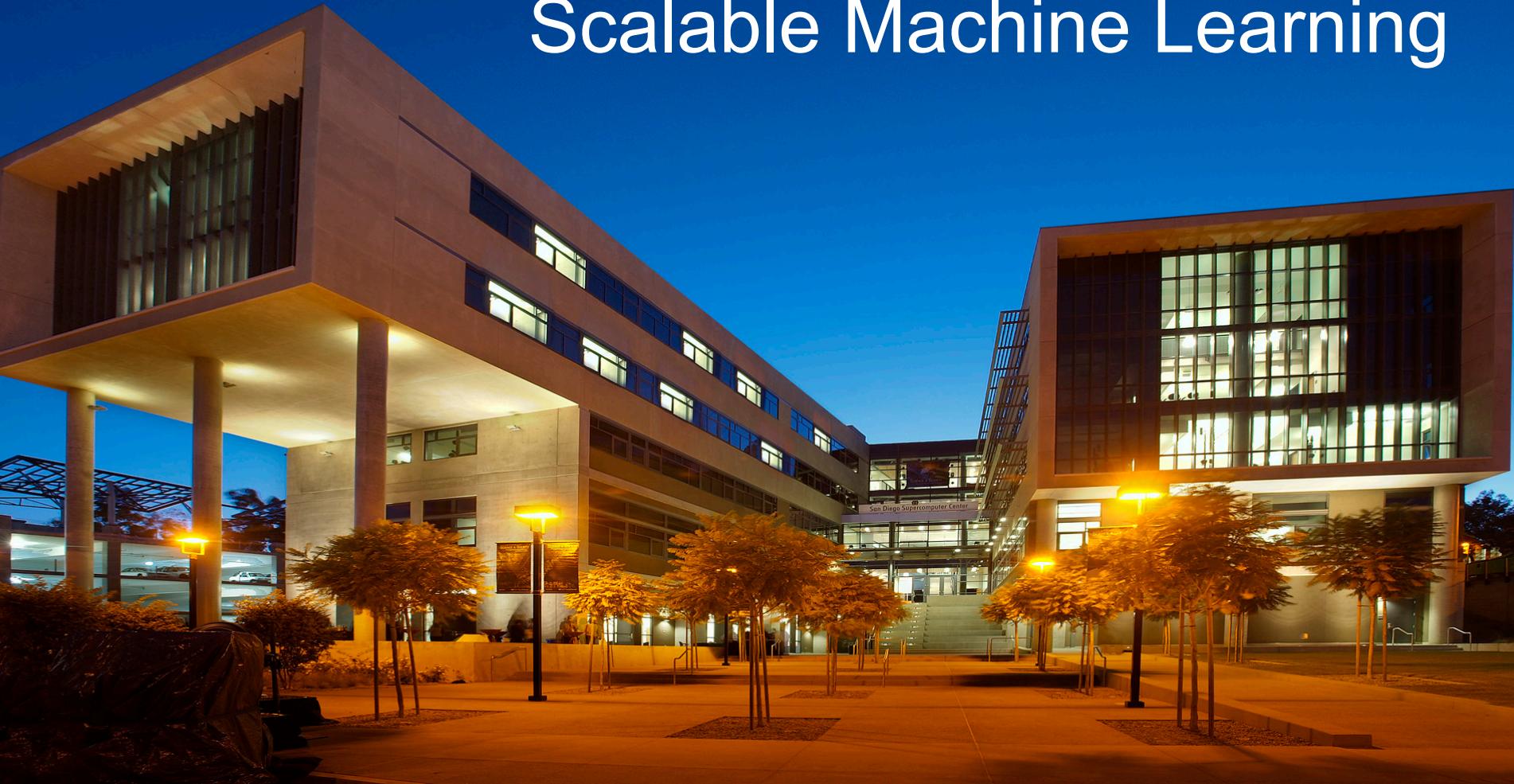


# 2018 SDSC Summer Institute Scalable Machine Learning



# 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")`

# Server Setup

- **Go to sparkR directory**

*cd <SI2018\_dir>/datasci1\_scalable\_machine\_learning/spark/sparkR*

- **Request compute node and start jupyter**
  - *sbatch --res=SI2018DAY2 sparkR.slurm*

# 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

# Server Setup (cont.)

- **Check queue**
  - `squeue -u $USER`
  - Note compute node name (e.g., comet-14-43)
    - 18223060 compute bash <user> R 1:19 1 comet-14-43
- **Check that Jupyter Notebook has started**
  - `ls -l sparkR*.out`  
*-rw-r--r-- 1 <user> sds148 840 Aug 4 09:32 sparkR.18317995.comet-14-43.out*  
• `tail sparkR*.out`  
• Copy token
    - *Copy/paste this URL into your browser when you connect for the first time, to login with a token:*  
`http://localhost:8888/?token=395fe3d456cb951e34ef125adf27e6a62`

# Browser Setup

- In browser, type:
  - comet-xx-xx.sdsc.edu:8888
  - Paste token

comet-14-43.sdsc.edu:8888/login?next=%2Ftree%3F



Password or token:

## Token authentication is enabled

If no password has been configured, you need to open the notebook server with its login token in the URL, or paste it above. This requirement will be lifted if you [enable a password](#).

# Open SparkR Data Exploration Notebook



Files

Running

Clusters

Select items to perform actions on them.

0 /

- [completed-notebooks](#)
- [data-exploration-R.ipynb](#)
- [daily\\_weather.csv](#)

# 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"))
```

# 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	11.175514003175877
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

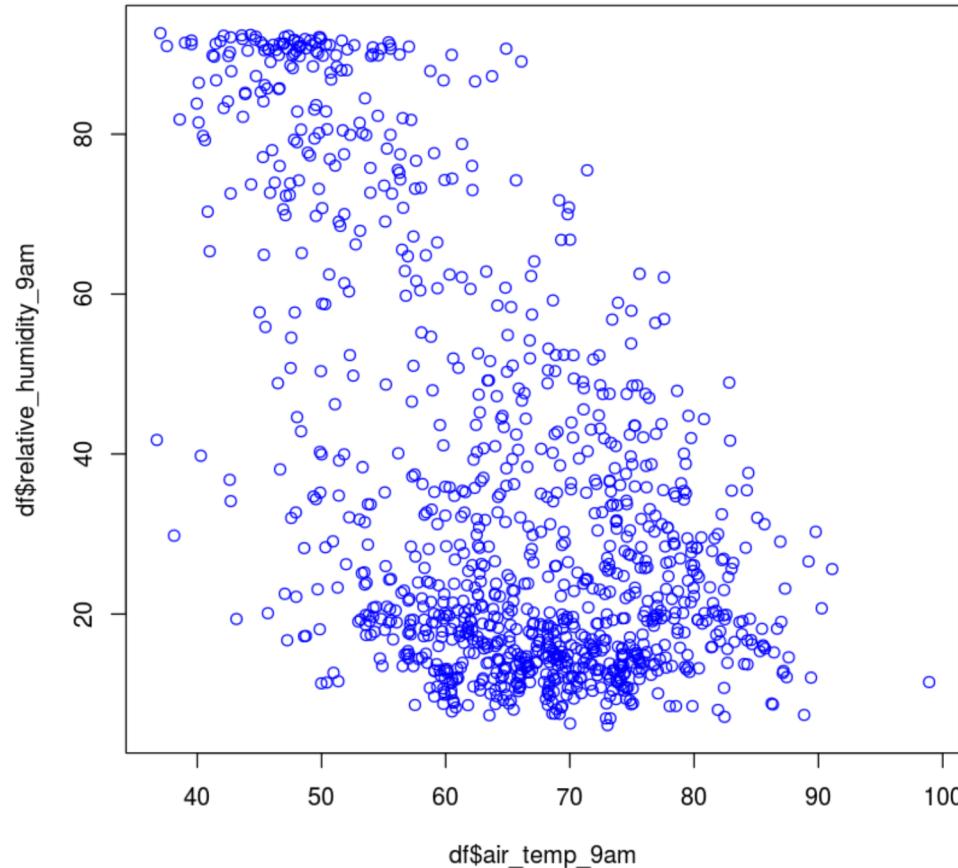
# Correlation Between Air Temperature and Relative Humidity

```
corr(sdf, "air_temp_9am", "relative_humidity_9am",  
method="pearson")
```

**-0.536670...**

# Scatter Plot of Air Temperature vs. Humidity (\*)

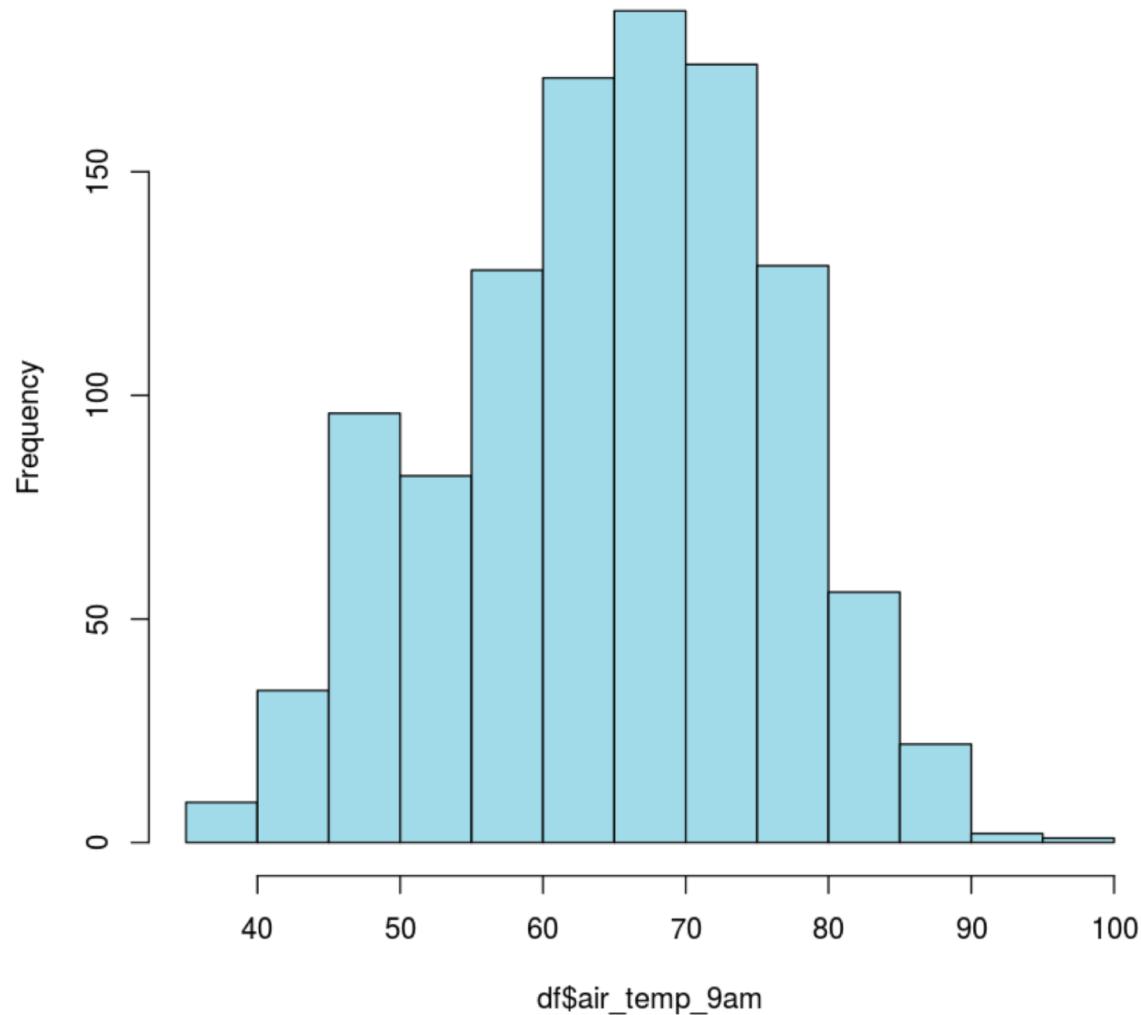
```
plot(df$air_temp_9am, df$relative_humidity_9am,  
     col="blue")
```



*hist(df\$air\_temp\_9am, col="lightblue")*

Histogram of df\$air\_temp\_9am

# Histogram of Air Temperature



# Exit Notebook

jupyter data-exploration-R Last Checkpoint: a

The screenshot shows a Jupyter Notebook interface. At the top, there's a toolbar with icons for file operations like 'New Notebook', 'Open...', 'Run', 'Cell', 'Kernel', and 'Widget'. Below the toolbar is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Widget'. A dropdown menu is open under the 'File' tab, listing options: 'New Notebook', 'Open...', 'Make a Copy...', 'Save as...', 'Rename...', 'Save and Checkpoint', 'Revert to Checkpoint', 'Print Preview', 'Download as', 'Trusted Notebook', and 'Close and Halt'. The 'Close and Halt' option is highlighted with an orange rectangle. To the right of the menu, the main notebook area displays a code cell starting with 'SparkR library' and some R code. The code cell has a pink background.

```
SparkR library
(SparkR, lib.loc = c(file.path("lib", "sparkr")))

# Load message:
# age 'SparkR' was built under
# hing package: 'SparkR'

# The following objects are masked
# from 'SparkR':
#   filter, lag, na.omit, pr
# The following objects are masked
# from 'base':
#   as.data.frame, colnames, col
```

# 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



Files

Running

Clusters

Select items to perform actions on them.

A screenshot of a Jupyter file browser interface. At the top, there is a header with three tabs: 'Files' (selected), 'Running', and 'Clusters'. Below the header, a message says 'Select items to perform actions on them.' A toolbar contains a checkbox, the number '0', a dropdown arrow, a folder icon, and a separator slash. The main area shows a list of items:

- [completed-notebooks](#)
- [data-exploration-R.ipynb](#)
- [wine-R.ipynb](#) (This item is highlighted with a red rectangular border.)
- [daily\\_weather.csv](#)

# Wine Classification Results

```
mean(actual_vs_predicted$actual == actual_vs_predicted$predicted)  
table(actual_vs_predicted$actual, actual_vs_predicted$predicted)
```

---

0.616621983914209

	average	bad	good
average	443	165	73
bad	163	342	7
good	153	11	135

# Clean Up

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

# 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

# SparkR 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>
- **Spark R API**
  - <https://spark.apache.org/docs/latest/api/R/>

# Questions?



# Scalable Machine Learning Topics

- **R in HPC**
  - Scaling R, running R on HPC
  - Scaling R linear models
- **Machine Learning with Spark**
  - Spark stack, RDDs
  - MLlib
- **PySpark Hands-On**
  - Data exploration
  - Clustering
- **SparkR Hands-On**
  - Data exploration
  - Random forest for classification