ISA 414 – Managing Big Data

Lecture 24 – Introduction to Spark

(Part II)

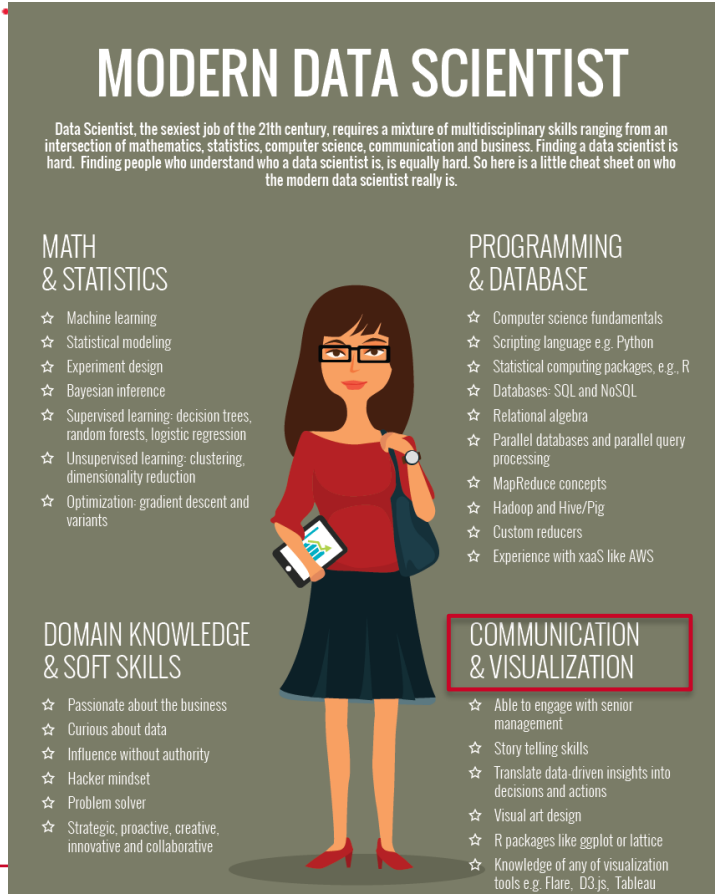
Dr. Arthur Carvalho

arthur.carvalho@miamioh.edu

MIAMI UNIVERSITY

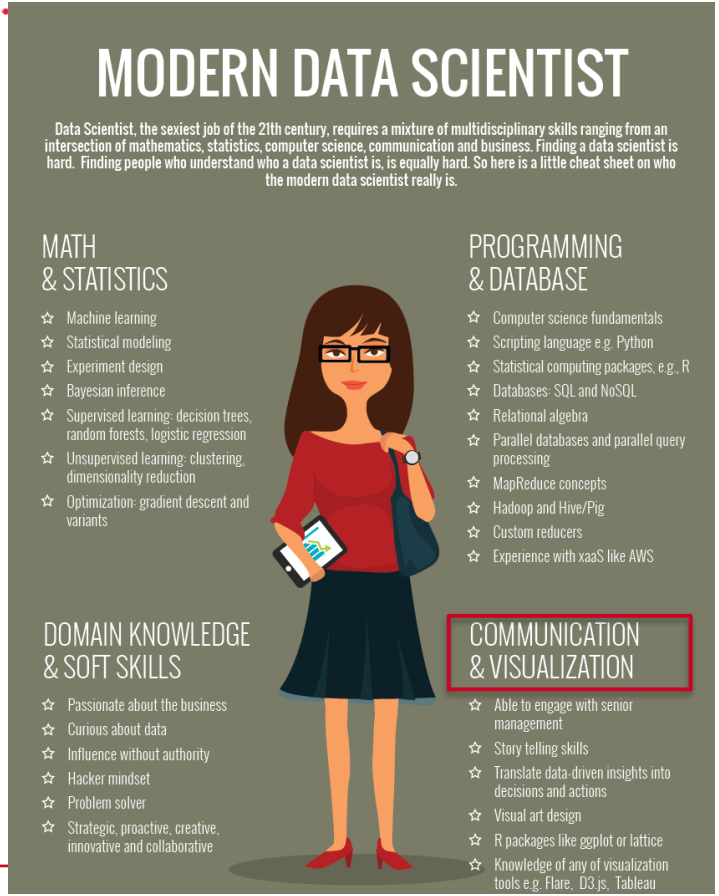
Agenda

- 11/16 and 11/18: presentation of (preliminary) project ideas
  - Goals:
    1. Strengthen your project ideas
    2. Soft skills: presentation
  - Duration
    - ISA 414: 15 minutes
    - ISA 514: 20 minutes
  - Presentation structure
    - Business background
    - Problem definition
    - Proposed solution (bird's-eye view)
    - Potential pros and cons



Agenda

- 11/16 and 11/18: presentation of (preliminary) project ideas
  - Your project grade will also reflect this presentation
    - Hint: take it seriously and come prepared
  - Non-presenters
    - Your behavior during the presentations will be reflected in your individual, final grade
  - The presentation slides must be sent to the instructor by email 24h before the day of the presentation



Agenda

- Presentation order (**Tuesday, 11/16**)
  - **ISA 514** (10:05 am – 11:25 am)
    - Group D: Alexander Amata, Jack Boyd, Will Dunn, Ben Keeley
    - Group A: Brendan Byrnes, Griffin Lester, Vish Nalagatla
  - **ISA 414** (1:15 pm – 2:35 pm)
    - Group D: Sierra Hessinger, Elianna Pecha, Rebekah Poth, Jacob Steele
    - Group A: John Doll, Abby Larson, Aubrey Liu, Spencer Townes
    - Group H: Hannah Le, Matt Madias, Carly Schechtman, Anthony Troiano
    - Group C: Charlie Fox, Ava Kunar, Tara Morrison, Nick Telerico

Agenda

- Presentation order (**Thursday, 11/18**)
  - **ISA 514** (10:05 am – 11:25 am)
    - Group B: Maddie Banyas, Amara Cummins, Ashley Lonsinger, Meghana Muvva
    - Group C: Arnav Damodhar, Abay Ismailov, Jason Lantz
  - **ISA 414** (1:15 pm – 2:35 pm)
    - Group F: Natalie Day, Sam Groth, Aidan McGaughy, Andre Su
    - Group G: Benjamin Boczulak, Ben Cawley, Caroline Davis, Jack Laux
    - Group B: Cecilia Dauer, Julia Edelman, Macayla Temple
    - Group I: Nick Cimarusti, Mitch Gray, Thomas Hemsworth, Brendan Keck
    - Group E: Nina Gollapudy, Nicholas Hesselgesser, Esha Kallam, Chau Vu

Agenda

- 11/23 and 11/30: in-class group work
- 12/02: review session

Lecture Objectives

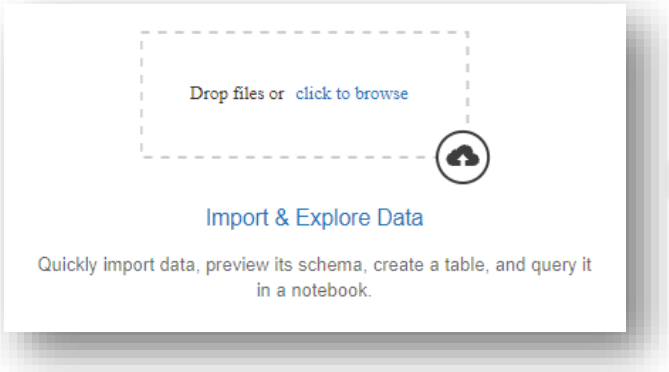
- Review Homework 10
- Discuss Assignment 4
- Continue to learn about Spark and its libraries
  - Learn how to use machine learning algorithms in Spark
    - ML Library
  - Learn how to use SQL to obtain data from RDDs in tabular format
    - SQL Library

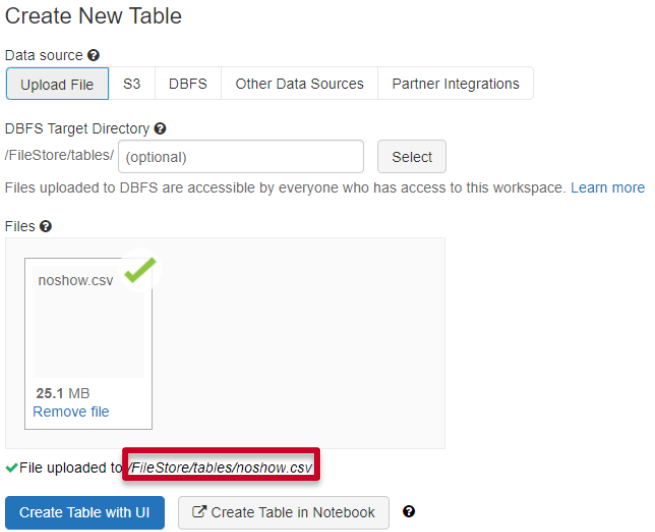
Lecture Instructions

- Download the file *Lecture 24.ipynb* from Canvas
- Download the data set *noshow.csv* available on Canvas

Lecture Instructions

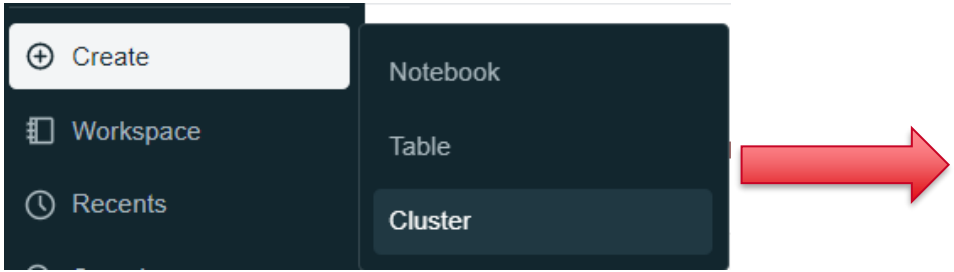
- Go to https://community.cloud.databricks.com/
  - Sign in
- Upload the file noshow.csv to Databricks
  - Copy the path to the file

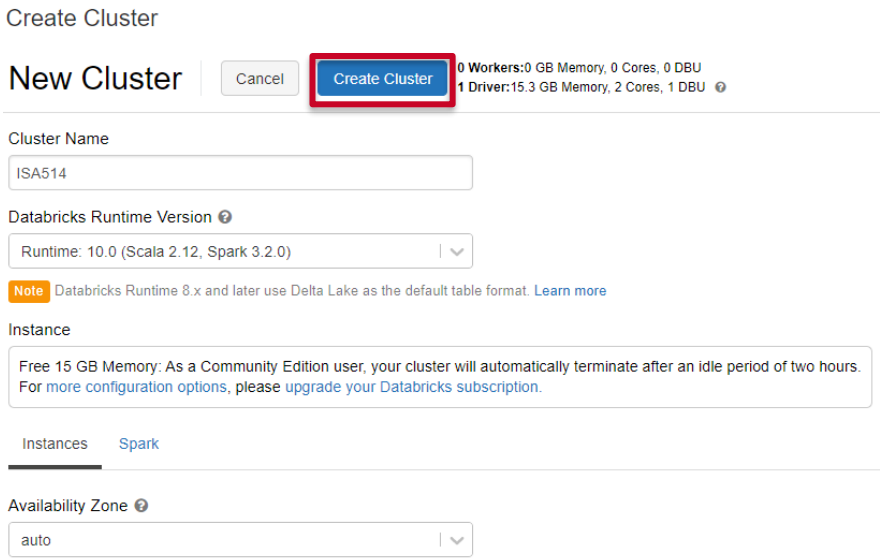




Lecture Instructions

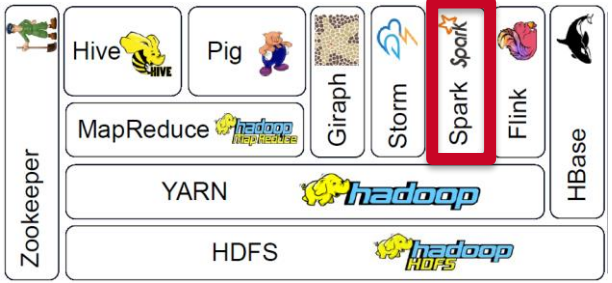
- Next, create a cluster
  - Go to the left panel, “Create” -> “Cluster”





MIAMI UNIVERSITY

The Hadoop Ecosystem

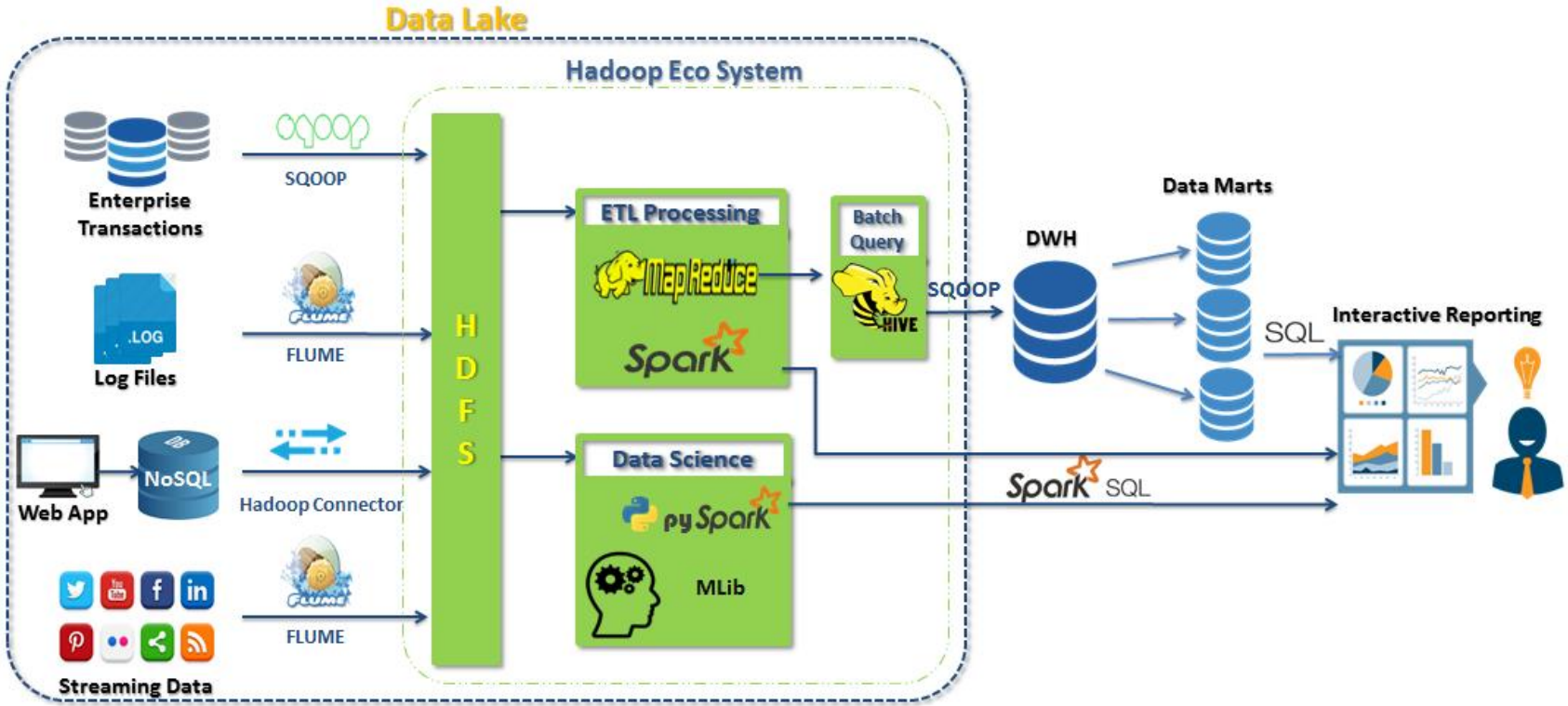


- Overview and agenda
  - HDFS
    - Hadoop Distributed File System
    - Scalable and reliable storage
  - Yarn
    - Schedule jobs/task over HDFS storage
  - MapReduce
    - Programming model that simplifies parallel/distributed computations
    - Two functions: Map (apply) and Reduce(summarize)
  - **Spark**
    - **Built for real-time, in memory processing of data**

Data Lake

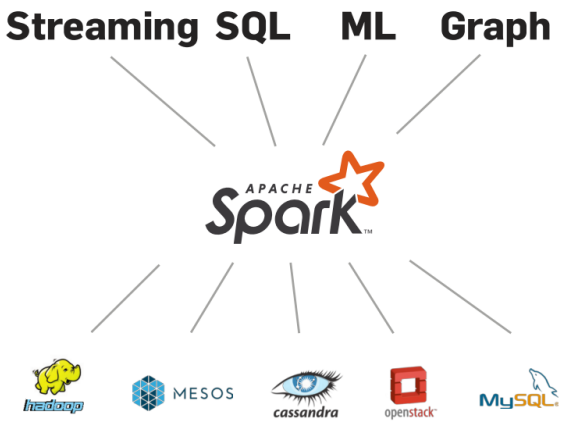
- How does data get into Hadoop?
  - Standard approach
    - Hadoop services automatically extract and send data to HDFS
    - Examples:
      - *Sqoop*: transfers bulk data from relational databases to HDFS
      - *Flume*: transfers log data from web servers to HDFS
      - …

Data Lake


Data Lake
Hadoop Eco System
Enterprise Transactions
SQOOP
Log Files
.LOG
FLUME
Web App
NoSQL
Hadoop Connector
Streaming Data
FLUME
H
D
F
S
ETL Processing
MapReduce
Spark
Batch Query
HIVE
SQOOP
DWH
Data Marts
SQL
Interactive Reporting
Data Science
pySpark
MLib
Spark SQL

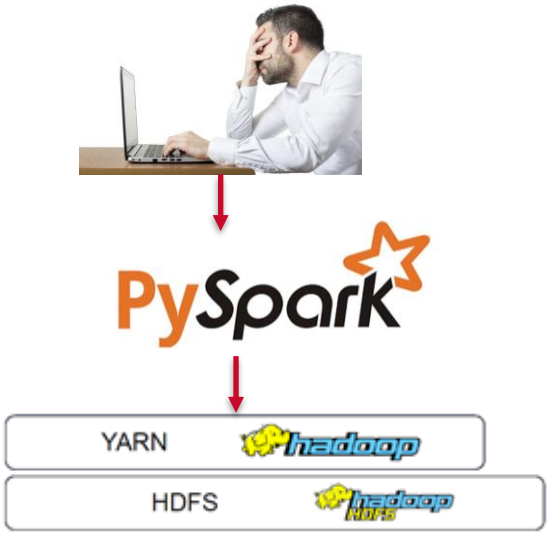
Spark

- Spark
  - Core definitions: RDD, transformations, and actions
  - Libraries: Streaming, SQL, ML, GraphX
    - Our focus today: ML and SQL



Spark

- Spark
  - Always keep the following perspective in mind



We can simply use **Spark libraries** to send commands to a spark cluster

PySpark

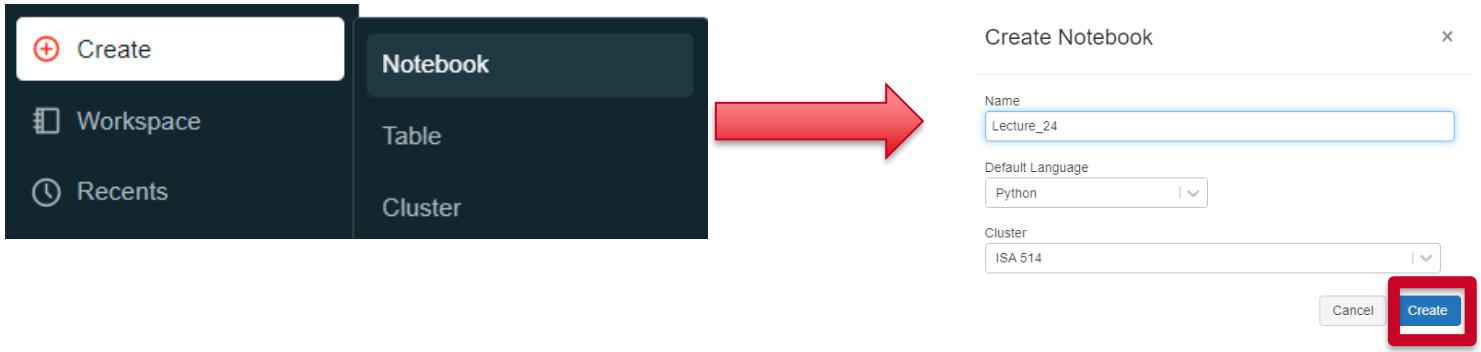
- Documentation
  - All what you need to know about PySpark:
    - https://spark.apache.org/docs/latest/api/python/reference/index.html
  - We shall see some of these functions in class today

Spark

- No-show case study using Spark
  - Question: will a certain person who booked a doctor's appointment show up?
  - Background: real-life data from Brazil
    - Government-sponsored healthcare: "*I will not pay anything if I book an appointment and don't show up*"
    - No shows take spots from people who need the service
  - Idea: if we can predict who will show up, we can estimate the number of no-shows
    - Classification problem: dependent variable *Status*
    - Overbooking: good policy
      - That is what some airlines do

Spark

- Let's connect to Spark
  - Create a notebook: Go to the left side panel, "*Create*" -> "*Notebook*"



  - As we progress, copy the code from "*Lecture 24.ipynb*" into Databricks

Spark

- Spark heavily relies on the concepts of ***pipes*** (or pipelines)
  - Many chained commands
    - The output of one command is the input to another one
  - Consequences
    - Shorter code
    - Fewer variables (lower memory consumption)

Spark

- Loading a CSV file from a cluster to the main memory

```
# File location and type
file_location = "/FileStore/tables/noshow.csv"

# inferSchema := detect data types
# header := whether first row contains column names
raw_data = spark.read.load(file_location, format="csv", inferSchema="true", header="true")
```

- Retrieving and showing the first 20 rows in our data frame

```
raw_data.show()
```

Spark

- Let’s do some data preprocessing now
  - Documentation: https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrame.html
  - Check variable types: df.dtypes
  - Is the data set unbalanced?

```
df.select("Status").groupBy("Status").count().show()
```

Spark

- Let's do some data preprocessing now
  - Let's reduce the number of values for the variable DayOfTheWeek
    - replace function: https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrame.replace.html#pyspark.sql.DataFrame.replace

```
df = df.replace(["Monday","Tuesday","Wednesday","Thursday","Friday"],
                "WeekDay", "DayOfTheWeek")
df = df.replace(["Saturday","Sunday"], "WeekEnd", "DayOfTheWeek")
```

Spark

- Let’s do some data preprocessing now
  - Basic statistics for the variable Age
    - summary() function: https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrame.summary.html#pyspark.sql.DataFrame.summary

      df.select("Age").summary().show()
  - Filtering negative Age
    - filter() function: https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrame.filter.html#pyspark.sql.DataFrame.filter

      df = df.filter(df["age"] >=0)

ML Library

- It is time to build a predictive model
  - Like with sklearn, we have to transform qualitative variables to numeric values
    - Qualitative values to indexes: StringIndexer function
    - Qualitative values to 3+ dummies: OneHotEnconder function

```
from pyspark.ml.feature import StringIndexer

stringToIndex = StringIndexer(inputCol = 'Gender', outputCol = 'GenderIndex')
df = stringToIndex.fit(df).transform(df)

stringToIndex = StringIndexer(inputCol = 'DayOfTheWeek', outputCol = 'DayOfTheWeekIndex')
df = stringToIndex.fit(df).transform(df)

stringToIndex = StringIndexer(inputCol = 'Status', outputCol = 'StatusIndex')
df = stringToIndex.fit(df).transform(df)
```

ML Library

- It is time to build a predictive model
  - PySpark ML Library requires that all the predictors must be combined into a single vector

```
from pyspark.ml.feature import VectorAssembler

predictors = ['Age', 'GenderIndex', 'DayOfTheWeekIndex', 'Diabetes',
              'Alcoolism', "HiperTension", "Handcap", "Smokes",
              "Scholarship", "Tuberculosis",  "Sms_Reminder",
              "AwaitingTime"]
assembler = VectorAssembler(inputCols=predictors, outputCol="predictors")
df = assembler.transform(df)
```

ML Library

- It is time to build a predictive model
  - Remember the discussion: training vs test sets

```
training_set, test_set = df.select(["StatusIndex","predictors"]).randomSplit([0.75,0.25])
```

  - Training a random forest with 100 trees

```
from pyspark.ml.classification import RandomForestClassifier

model = RandomForestClassifier(numTrees=100,
                               featuresCol= "predictors",
                               labelCol='StatusIndex')
model = model.fit(training_set)
```

ML Library

- Evaluation
  - Predictions on the test set

```
predictions = model.transform(test_set)
```

  - Model's overall accuracy

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

evaluator = MulticlassClassificationEvaluator(labelCol="StatusIndex",
                                              predictionCol="prediction",
                                              metricName = "accuracy")

accuracy = evaluator.evaluate(predictions)
print("Accuracy = %s" % (accuracy))
```

SQL Library

- Hadoop has services that allow one to use SQL to retrieve data from structured files inside HDFS, such as CSV files
  - Spark SQL
  - Hive (not covered in this course)
  - Impala (not covered in this course)

Spark

- Spark SQL
  - Once a data set is loaded into a Spark cluster, SQL commands can be used to retrieve data from the RDD

    raw_data.createOrReplaceTempView("TABLE")

  - SQL commands are translated into transformations and actions
    - Examples

```
spark.sql("SELECT AGE FROM TABLE").show()
spark.sql("SELECT Status, COUNT(Status) FROM TABLE GROUP BY Status").show()
spark.sql("SELECT * FROM TABLE WHERE AGE < 0").show()
```

Spark

- You might be asking yourself: "*is this whole thing worth it?*"
  - "*I could do the same thing with fewer lines of code and running quicker without Spark*"

- Always keep in mind that Spark is used when there are tons of data and many available nodes
  - Only then, you can see how Spark speeds up data processing
  - Think about a data set having millions or even billions of appointments

Summary

- We learned about Spark ML and SQL libraries
  - To use Spark in Python, one simply needs to learn about a bunch of predefined functions

- Next lecture: project (idea) presentations