Apache Spark

SAMEER MAHAJAN

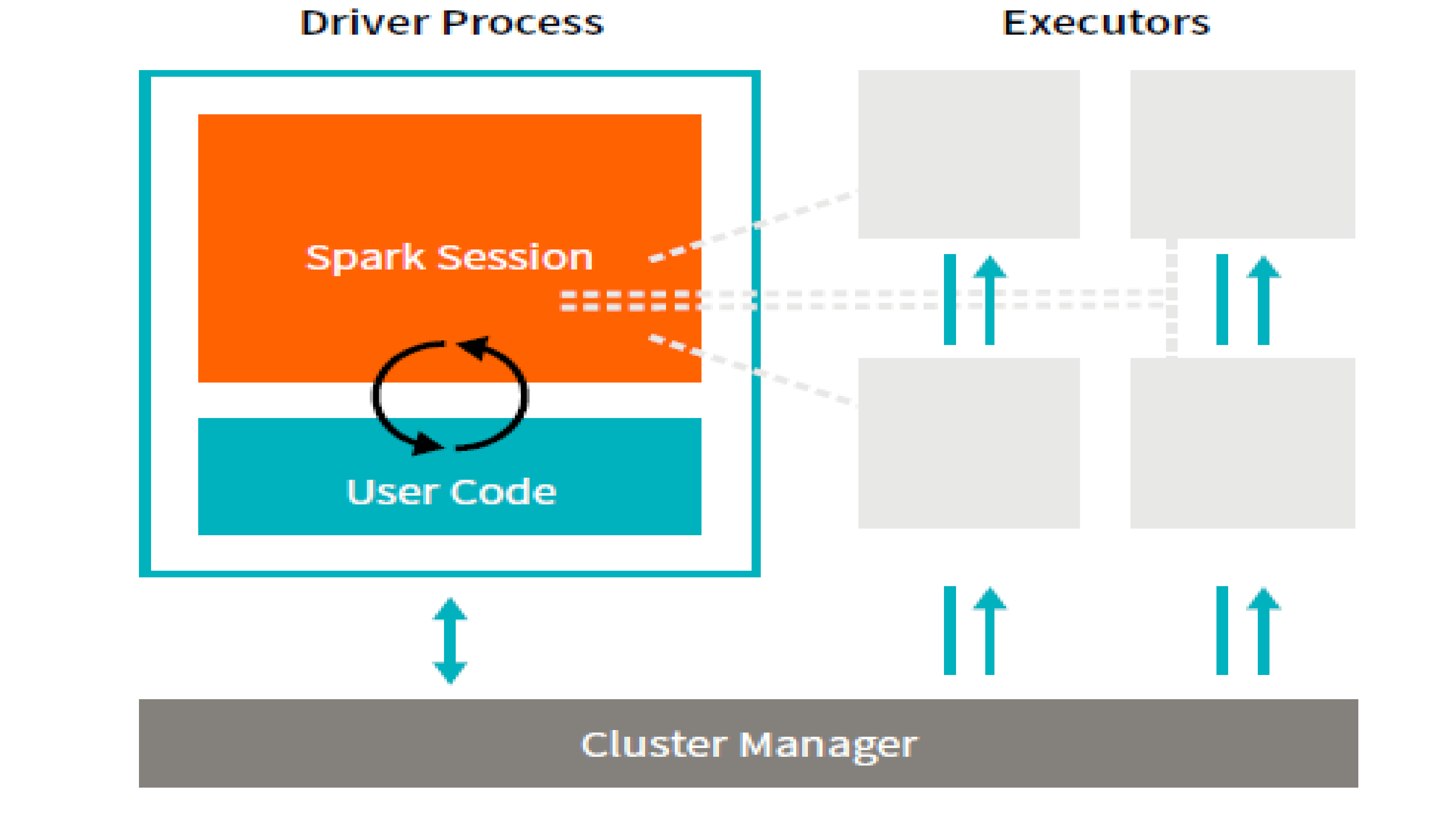
History

- 2009 research project in UC Berkeley
- Learnt from Hadoop
  - Cluster computing
  - Multi pass over data
- API based on functional programming that could succinctly express multi-step application
- in-memory data sharing across computation steps
- Version 1: Batch Processing only, Version 2: Interactive
- Shark: SQL queries, 2011
- AMPlab contributed Spark to Apache and started Databricks
- Apache Spark 1.0 in 2014 (Spark SQL API) and 2.0 in 2016
- Structured Streaming, 2017

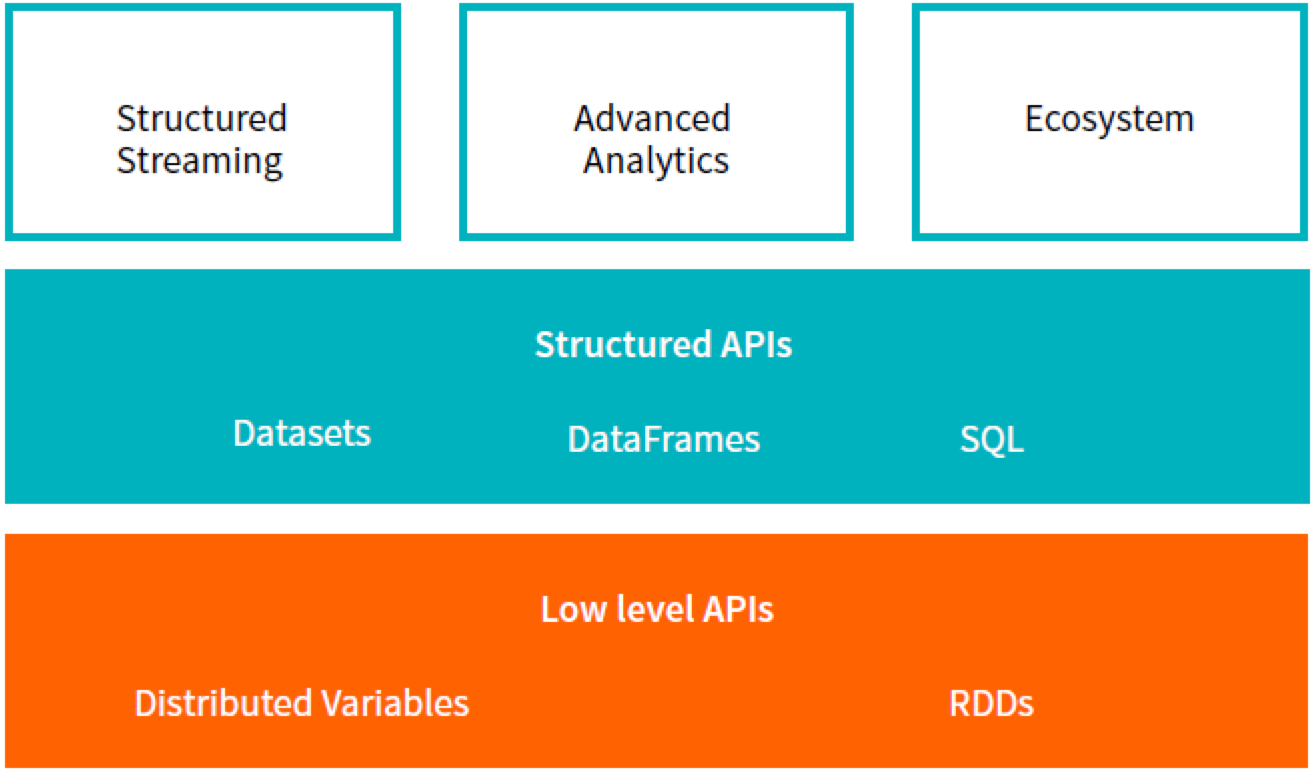
Key Benefits

- Parallelism on a cluster
- Most actively developed open source engine
- Wide support for programming languages (Python, Java, Scala, R)
- Unified analytics compute engine for loading, querying, streaming, ML
- Consistent, composable APIs
- Runs from a laptop to a cluster of thousands of servers
- Works with s3, Hadoop, Cassandra, Kafka etc.
- Large number of internal and external (http://spark-packages.org) libraries

High level architecture



High level view



Structured Streaming

Advanced Analytics

Ecosystem

Structured APIs

Datasets

DataFrames

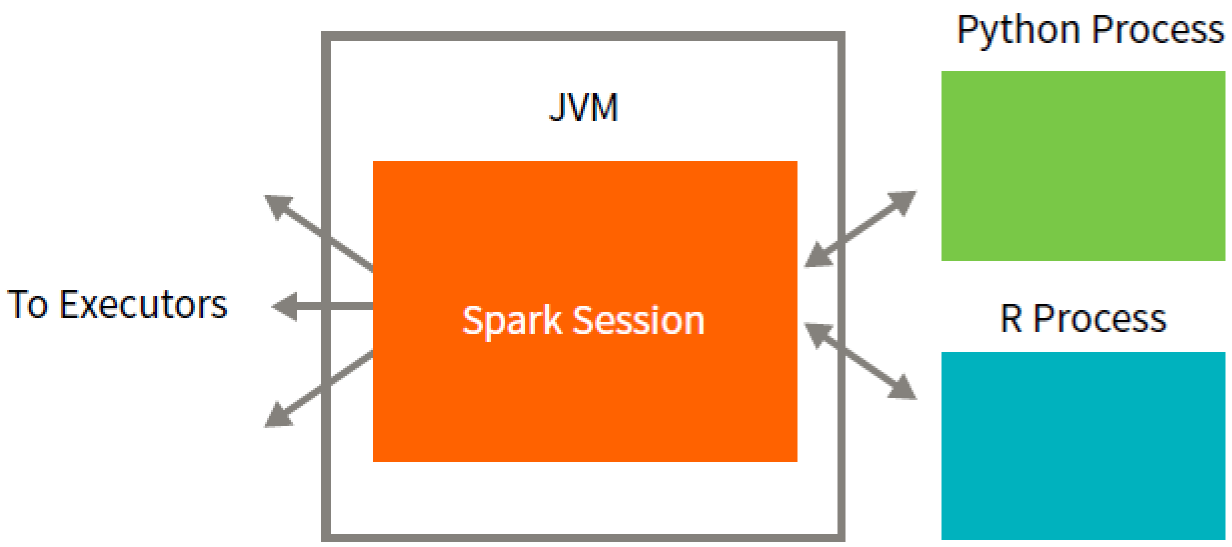
SQL

Low level APIs

Distributed Variables

RDDs

Language support



Data Structures and Some Basic Concepts

- Data Structures
  - Dataframe
  - Dataset
  - SQL Table
  - Resilient Distributed Dataset (RDD)
- Some Basic Concepts
  - Partitions
  - Transformations
  - Lazy evaluation
  - Action: triggers the computation

Getting started

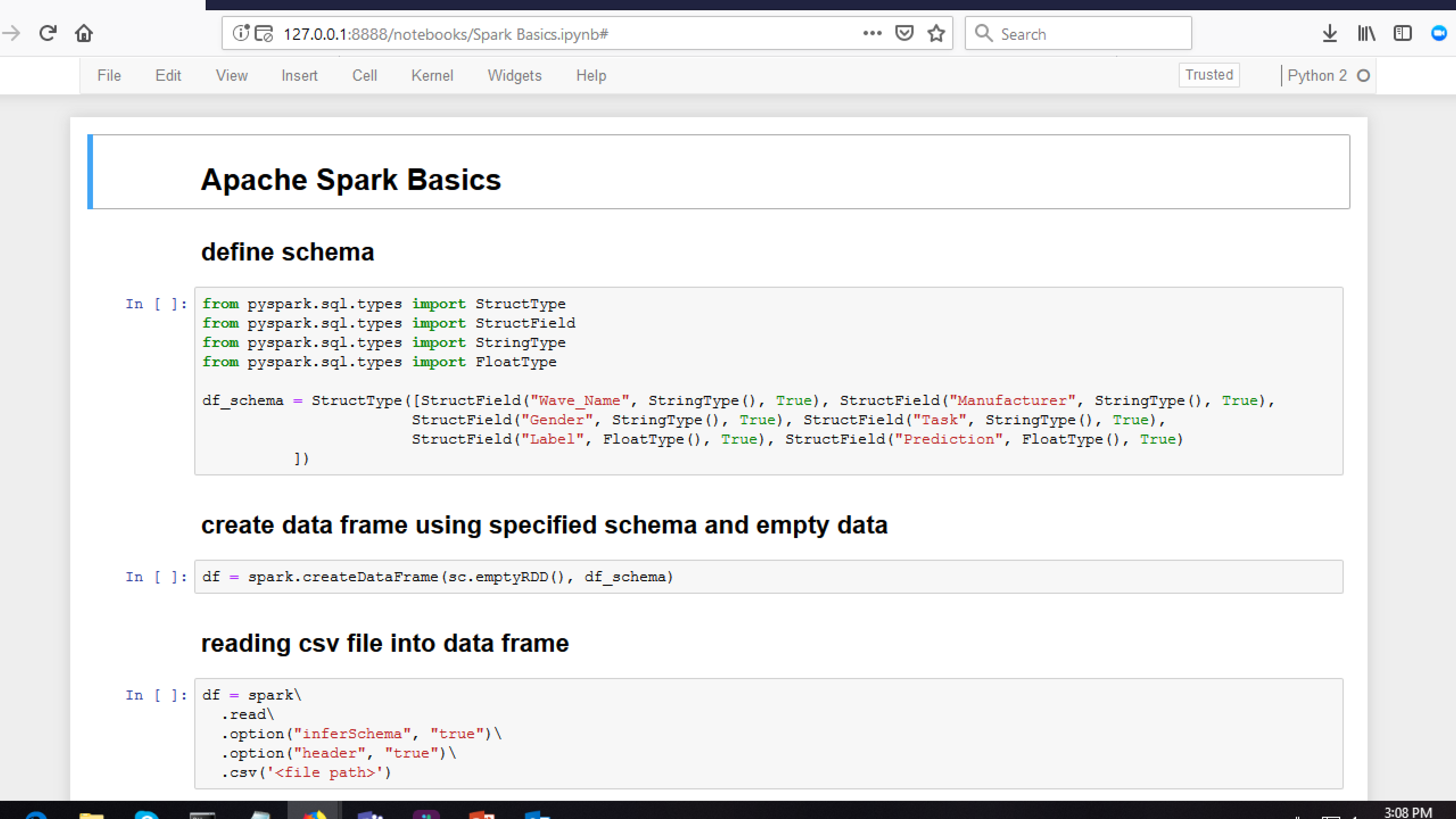
- Operate at spark dataframe level as much as possible
- Spark application is controlled through a driver process
- Driver process manifests as spark session
- One to one mapping between instance of spark session and spark application

Basics

- spark.read to read csv
- spark.createDataFrame
  - define schema, specify values
- .repartition(1).write.csv('location') : write spark DF to a single file

Some conversions

- .rdd to conver to rdd (.rdd.map (lambda row: row[0]) to convert a column to rdd)
- rdd.toDF() : to convert rdd to spark DF
- toPandas() : convert spark DF to pandas DF
- SQLContext(sc). createDataFrame(pandas_df) : convert pandas DF to spark DF
- df.createOrReplaceTempView(‘sql_table’) : convert spark DF into sql table



Apache Spark Basics

define schema

```
In [ ]: from pyspark.sql.types import StructType
        from pyspark.sql.types import StructField
        from pyspark.sql.types import StringType
        from pyspark.sql.types import FloatType

        df_schema = StructType([StructField("Wave_Name", StringType(), True), StructField("Manufacturer", StringType(), True),
                                StructField("Gender", StringType(), True), StructField("Task", StringType(), True),
                                StructField("Label", FloatType(), True), StructField("Prediction", FloatType(), True)
                  ])
```

create data frame using specified schema and empty data

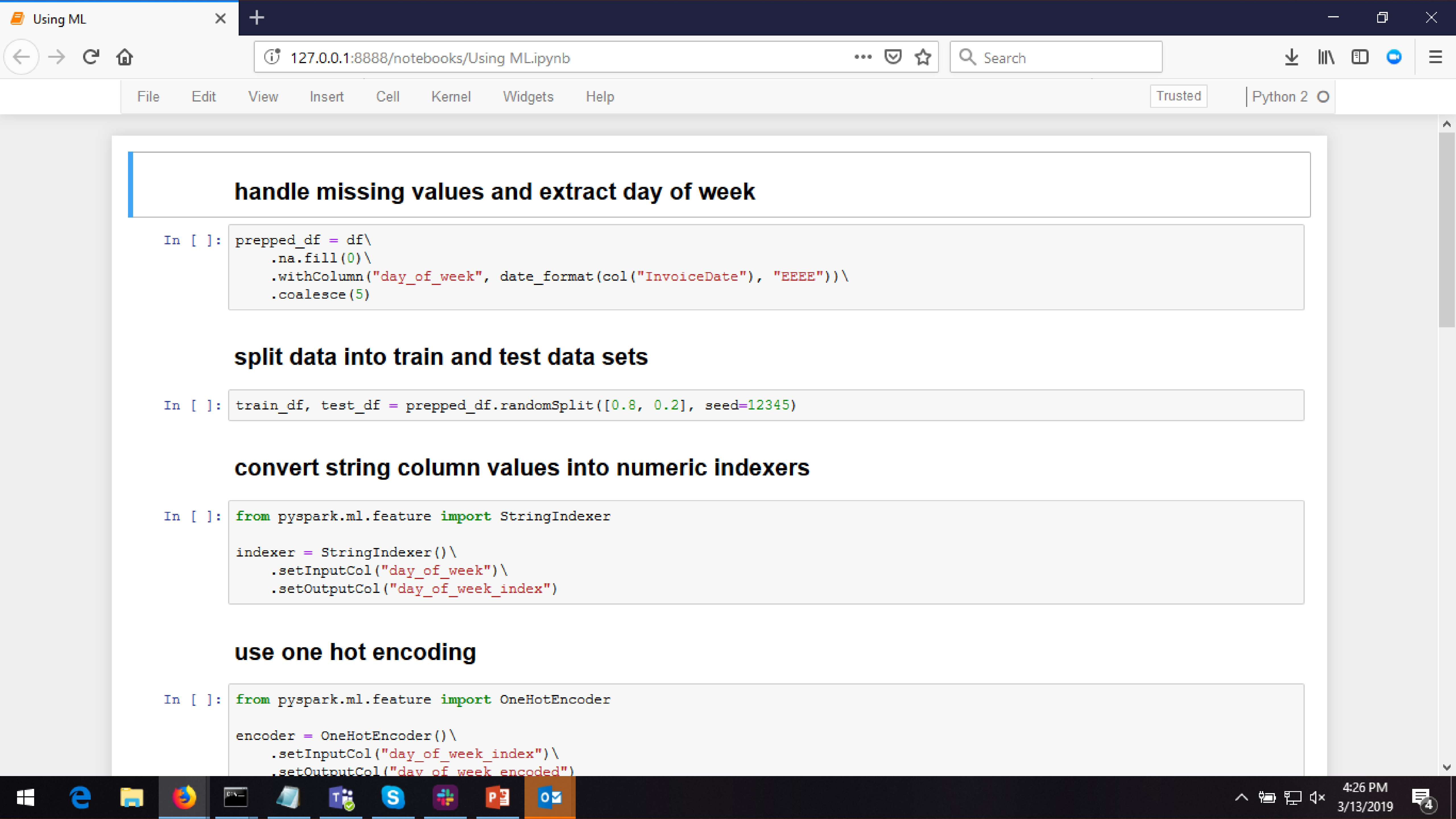
```
In [ ]: df = spark.createDataFrame(sc.emptyRDD(), df_schema)
```

reading csv file into data frame

```
In [ ]: df = spark\
          .read\
          .option("inferSchema", "true")\
          .option("header", "true")\
          .csv('<file path>')
```

Machine learning libraries

- ml
  - newer implementation
  - still under development with some experimental classes
  - works on spark Dataframe
  - watch out for ‘Experimental’
- mllib
  - older implementation
  - works on RDD



handle missing values and extract day of week

In [ ]:
```
prepped_df = df\
    .na.fill(0)\
    .withColumn("day_of_week", date_format(col("InvoiceDate"), "EEEE"))\
    .coalesce(5)
```

split data into train and test data sets

In [ ]:
```
train_df, test_df = prepped_df.randomSplit([0.8, 0.2], seed=12345)
```

convert string column values into numeric indexers

In [ ]:
```
from pyspark.ml.feature import StringIndexer

indexer = StringIndexer()\
    .setInputCol("day_of_week")\
    .setOutputCol("day_of_week_index")
```

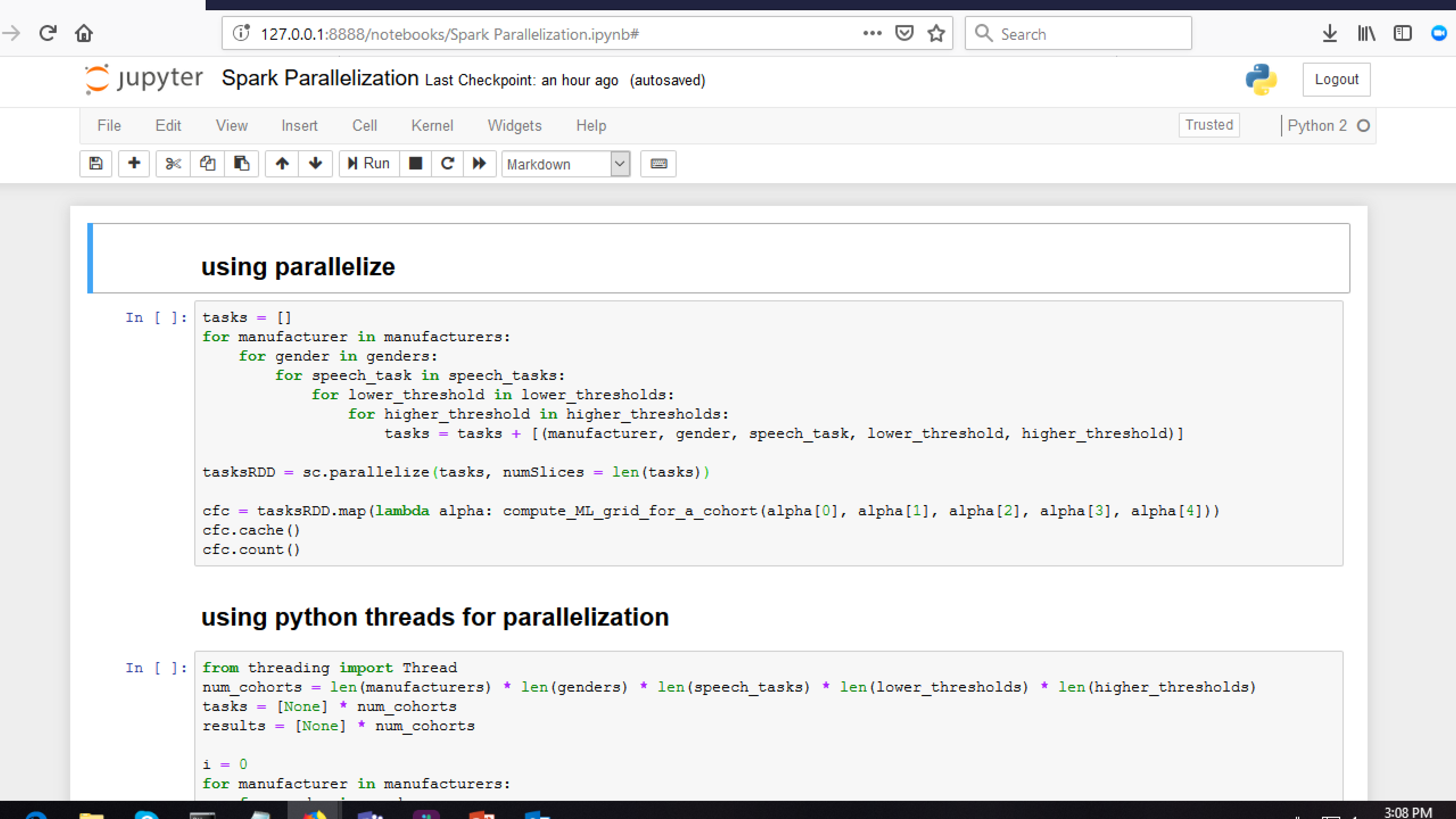
use one hot encoding

In [ ]:
```
from pyspark.ml.feature import OneHotEncoder

encoder = OneHotEncoder()\
    .setInputCol("day_of_week_index")\
    .setOutputCol("day_of_week_encoded")
```

Parallel execution

- spark manages splitting into and aggregating results from individual tasks distributed on workers
- sc.parallelize and map lambda
  - parallel task runs on a worker (not driver) node
  - limitations on what can be passed to and used inside the parallel task
- python threads


127.0.0.1:8888/notebooks/Spark Parallelization.ipynb#
Search
jupyter
Spark Parallelization Last Checkpoint: an hour ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 2
Run
Markdown
3:08 PM

using parallelize

In [ ]:

```
tasks = []
for manufacturer in manufacturers:
    for gender in genders:
        for speech_task in speech_tasks:
            for lower_threshold in lower_thresholds:
                for higher_threshold in higher_thresholds:
                    tasks = tasks + [(manufacturer, gender, speech_task, lower_threshold, higher_threshold)]

tasksRDD = sc.parallelize(tasks, numSlices = len(tasks))

cfc = tasksRDD.map(lambda alpha: compute_ML_grid_for_a_cohort(alpha[0], alpha[1], alpha[2], alpha[3], alpha[4]))
cfc.cache()
cfc.count()
```

using python threads for parallelization

In [ ]:

```
from threading import Thread
num_cohorts = len(manufacturers) * len(genders) * len(speech_tasks) * len(lower_thresholds) * len(higher_thresholds)
tasks = [None] * num_cohorts
results = [None] * num_cohorts

i = 0
for manufacturer in manufacturers:
```

References

- http://spark.apache.org/
- https://pages.databricks.com/gentle-intro-spark.html
- http://shop.oreilly.com/product/0636920034957.do
- Spark Python API Documentation: https://spark.apache.org/docs/latest/api/python/index.html