# DATA ANYLISIS

## 1 EXPLORITORY ANALYSIS

MO Create a histogram of all columns :

Data = pd.column.hist(bins = 50)

## CLEANING DATA

Rows should contain individual observations (for the y=axis)

1. melting, we turned columns into rows
2. Pivoting: turn unique values into separate columns

Add missing values by

1. omitting the data row
2. fit the mean value of the column set

# FB PROPHET

## Changepoints

1. Default value is 0.05. Increase to 0.15
2. Controls sensitivity of projects.
   1. Increased value, increases sensitivity. Decreased flexibility. May cause over-fitting or over-generisation.
   2. Decrease in value, loer vari reduces sensitivity, more flexibility, more variance. Under-fitting to much bias.

# VECTORS

1. Scala value – describes magnitude
2. Vector – describes magnitude and direction
   1. Distance vs. displacement
   2. Speed vs. velocity
3. Vectors can be polar or cetaisiean

# DATA FILE SIZES

|  |  |  |
| --- | --- | --- |
|  | Data Size (Min) | Data Size (Min) Max |
| pandas | 0 | 5 GB |
| pandas(chunksize) | 5GB | 30 GB |
| dask | 30GB | 200 GB |
| pyspark | 200GB, 1 TB | Unlimited PB |

1 Petabyte = 1000 Terabytes = 1Million = GBs

## PANDAS(Chunksize)

import pandas as pd

a = pd.read\_csv(‘xxx.csv’, chunksize =1000000)

print(a.head())

Chuncksize = number of rows

Disadvantage: no error handling functions!

## DASK

*Dask* scales *numpy*, *pandas* and *scikit-learn*.

Generate 100million random numbers between 10 and 100 and store them in variable a

import pandas as pd  
import dask.array as da  
a = da.random.randint(10,100,100000000)

print(a.compute())

a = da.random.randint(10,100,100000000).reshape(1000000, 100)

For *dataframes*:

import pandas as pd  
import dask.dataframe as dd

a = dd.read\_csv(‘xxx.csv’)

print(dd.compute())

## PySPARK

import pyspark as spark  
from pyspark import SparkContext, SQLContext

ac = SparkContext()  
s = SQLContext(ac)

a = s.read.options(header=True).read\_csv(‘xxx.csv’)

a.data.show()

# TENSORFLOW

1. Node = maths operation
2. Edge (Tensor) = multidimensional array
3. Edges & Nodes are Python Objects

Competition:

1. PyTorch
2. CNTK (Microsoft)
3. NEXT (Apache)

Advantages: Open Source, Extensive Library & Support

Pipeline, Network:

1. Choose
2. Compile
3. Fit
4. Evaluate
5. Make Predictions

# SCIKIT-LEARN

# SPARK

## Lazy Evaluation

Strategy that holds evaluation of an expression until its value is needed i.e. when an action is triggered.

## Classes

1. pyspark.sql.SQLContext
2. pyspark.sql.Dataframe
3. pyspark.sql.Column
4. pyspark.sql.Row
5. pyspark.sql.GroupedData
6. pyspark.sql.DataFrameNafunctions
7. pyspark.sql.StatFunctions
8. pyspark.sql.functions
9. pyspark.sql.window

## Resident Distributed Datasets (Rdd)

Support:

1. Transformations
2. Actions (computations to output a result)

## Spark Ecosystem

|  |  |  |  |
| --- | --- | --- | --- |
| SparkSQL | Spark Streaming | MLlib  (Machine Lerning) | Graph X  (Graph computation) |
| Spark Core API  R Python(pyspark) Java Scala | | | |