Modeling, Prediction, Recommendation from Large-Scale Fitness Data

Project 4

Exercise Freak Consulting, LLC

The Predictomondo Team

David Doerner

Chief Analytics Officer

Jason Gilberg

Chief Business Development Officer

Patrick Mulrooney

Chief Data Architect

Masashi Omori

Chief Financial Officer

Advisor

Prof. Julian McAuley

Recap of Progress To Date

Clean Data

- Correct timestamps
- Remove extreme outliers
- Visualize data for cleaning

Designed Machine Learning Pipeline

- Dimensionality Reduction
- Clustering
- Regression

Set Up Systems

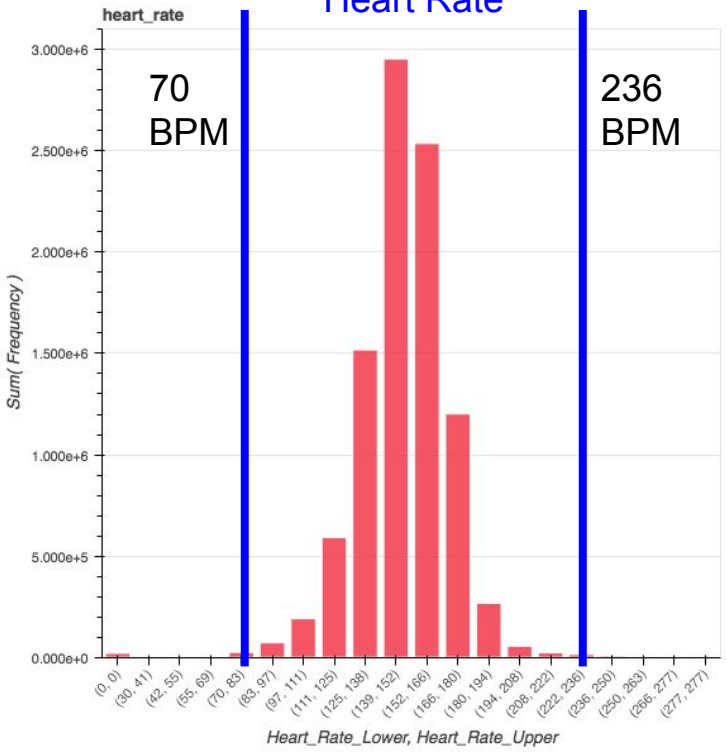
- KanbanFlow
- Postgres
- Spark

Initial Clustering

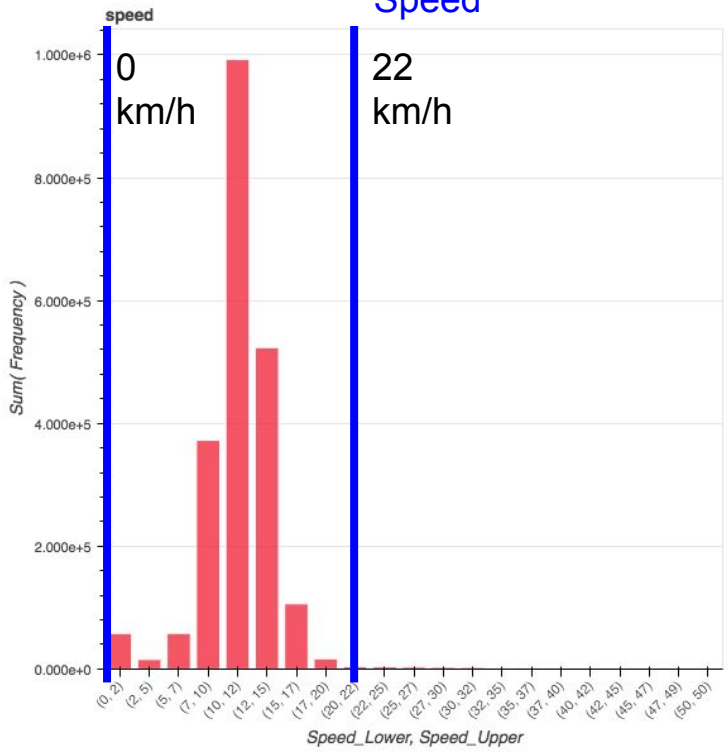
- Extract Summary Stats as Features
- Clustered on workout summaries

Visualizations -Summary of “Clean” Data

Heart Rate



Speed



Process

ABV (Always be visualizin')

1

2

Extract Summary Features

Workout (Time-Series) Dimensionality Reduction

Visualization

Cluster Route Info

Route Recommender

Cluster Performance

Predict Workouts

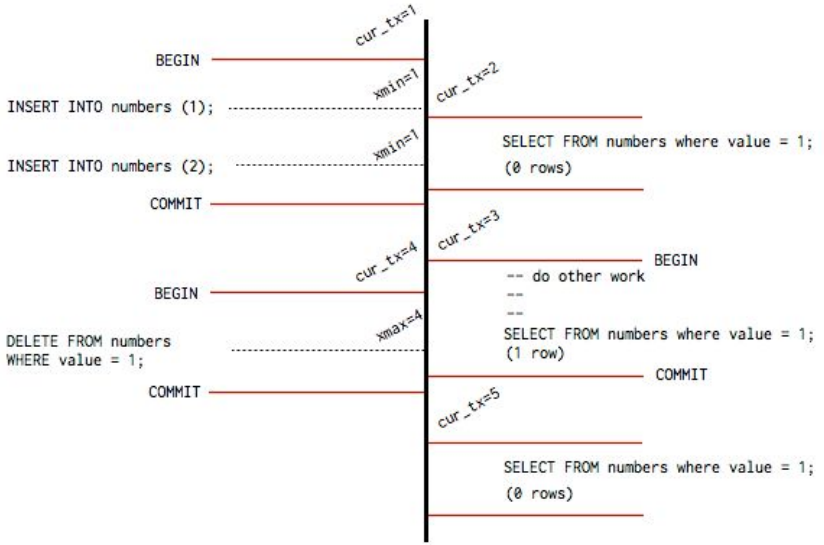
Iterate

Data Infrastructure - Postgres Pains

Multiversion Concurrency Control - Usually awesome, problematic when dealing with large data and limited disk space (and we don't need it)

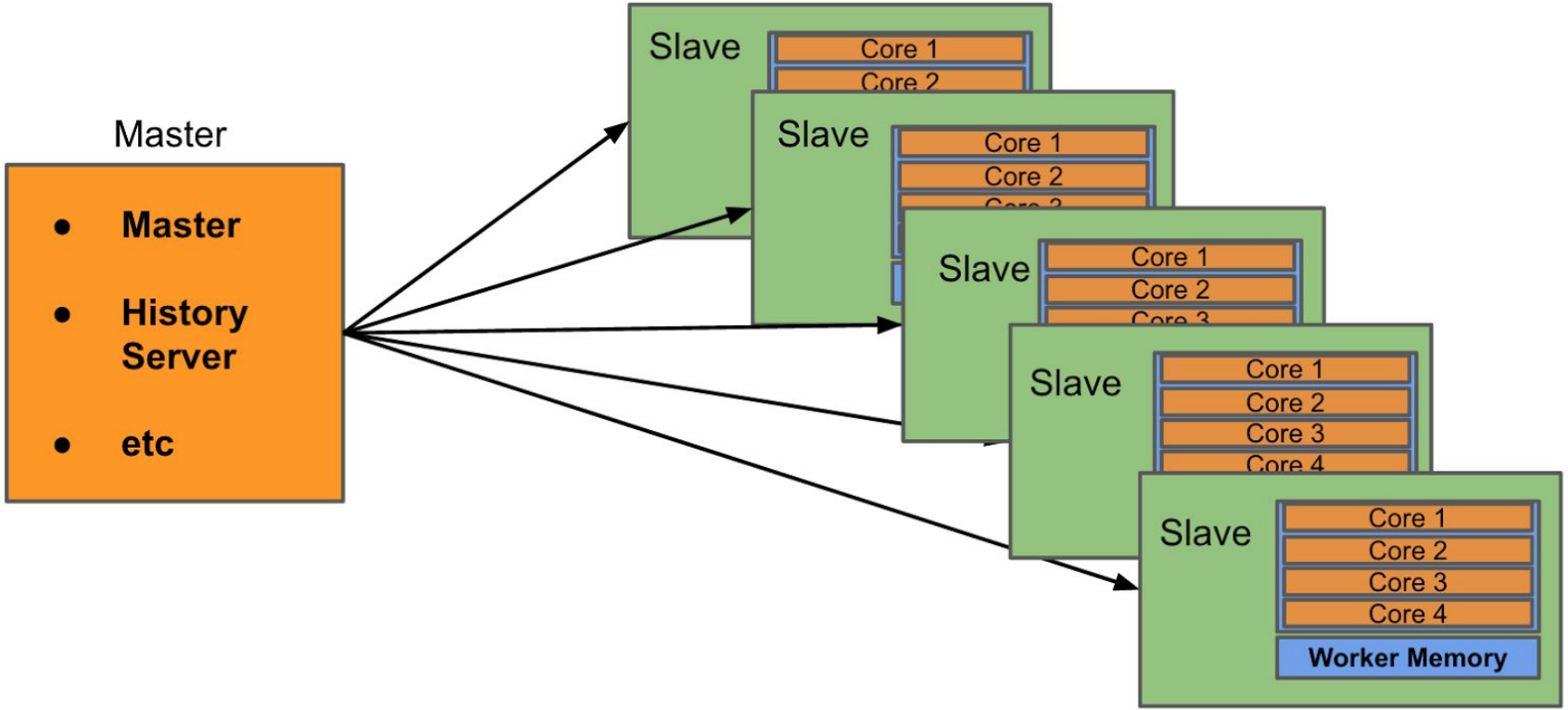
- Data not actually deleted
- Updating / adding columns causes size on disk to grow rapidly
- Difficult (impossible?) to disable
- Divide and conquer!
- ... and wait

- Use CLUSTER and Window functions (subqueries)



https://devcenter.heroku.com/articles/postgresql-concurrency

Spark Standalone Cluster - Architecture



http://2.bp.blogspot.com/-cmyATBs7QXw/VfCfBuVvYWI/AAAAAAAAMYs/Ylb3uSeo96c/s1600/3.png

Easy to setup your own

On all nodes:

- Ubuntu 16.04 install
- Install Java & Spark & a few other packages
- Add them to your environment

On the master:

- Open firewall to all slaves
- Start master process using provided script
- Start a pyspark notebook

On the slaves:

- Open firewall to all slaves
- Start slave process using provided script, point them at master.

**Detailed setup will be in our repo

Difficulties:

- Hive - permissions (even though we don't use it)
- One Spark context per cluster at a time

Spark Dashboard

Spark 2.1.0 — Jobs | Stages | Storage | Environment | Executors | SQL — **PySparkShell** application UI

Spark Jobs (?)

User: ubuntu
Total Uptime: 15.6 h
Scheduling Mode: FIFO
Completed Jobs: 217

▸ Event Timeline

Completed Jobs (217)

Page: 1 2 3 > 3 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
216	count at NativeMethodAccessorImpl.java:0	2017/05/10 22:56:23	0.3 s	2/2	9/9
215	collect at <ipython-input-33-919ee2ed89f0>:3	2017/05/10 22:56:22	0.4 s	2/2	208/208
214	count at NativeMethodAccessorImpl.java:0	2017/05/10 22:56:22	0.2 s	2/2	9/9
213	count at NativeMethodAccessorImpl.java:0	2017/05/10 22:56:05	0.3 s	2/2	9/9
212	collect at <ipython-input-32-6eaa215227f5>:3	2017/05/10 22:56:04	0.4 s	2/2	208/208
211	count at NativeMethodAccessorImpl.java:0	2017/05/10 22:56:04	0.2 s	2/2	9/9
210	count at NativeMethodAccessorImpl.java:0	2017/05/10 22:55:55	0.3 s	2/2	9/9
		22:55:54	0.5 s	2/2	208/208
		22:55:53	0.3 s	2/2	9/9
		22:55:37	0.3 s	2/2	9/9
		22:55:37	0.2 s	1/1	4/4
		22:55:37	93 ms	1/1	1/1
		22:55:37	0.2 s	2/2	9/9
		22:55:29	0.2 s	2/2	9/9
		22:55:29	0.2 s	2/2	9/9

Spark 2.1.0 — Jobs | Stages | Storage | Environment | Executors | SQL — **PySparkShell** application UI

Executors

Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write
Active(4)	0	0.0 B / 1.5 GB	0.0 B	24	0	0	2955	2955	4.0 min (11 s)	0.0 B	0.0 B	629.7 KB
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B
Total(4)	0	0.0 B / 1.5 GB	0.0 B	24	0	0	2955	2955	4.0 min (11 s)	0.0 B	0.0 B	629.7 KB

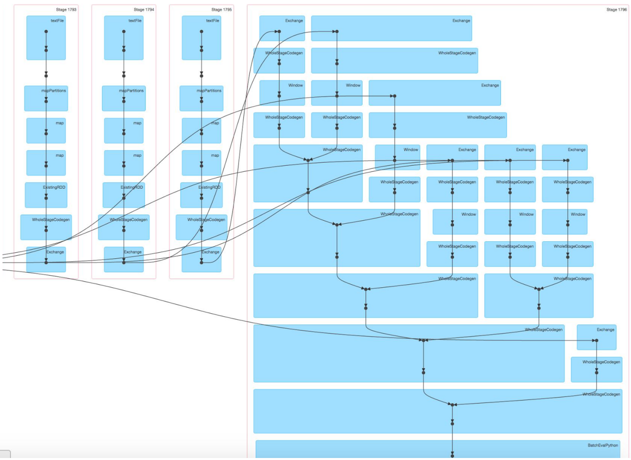
Executors

Show 20 entries Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Logs	Thread Dump
driver	172.31.29.1:40169	Active	0	0.0 B / 384.1 MB	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B		Thread Dump
0	172.31.42.143:34832	Active	0	0.0 B / 384.1 MB	0.0 B	8	0	0	2955	2955	4.0 min (11 s)	0.0 B	0.0 B	629.7 KB	stdout stderr	Thread Dump
1	172.31.33.137:46598	Active	0	0.0 B / 384.1 MB	0.0 B	8	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
2	172.31.40.182:37228	Active	0	0.0 B / 384.1 MB	0.0 B	8	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump

Showing 1 to 4 of 4 entries

Previous 1 Next

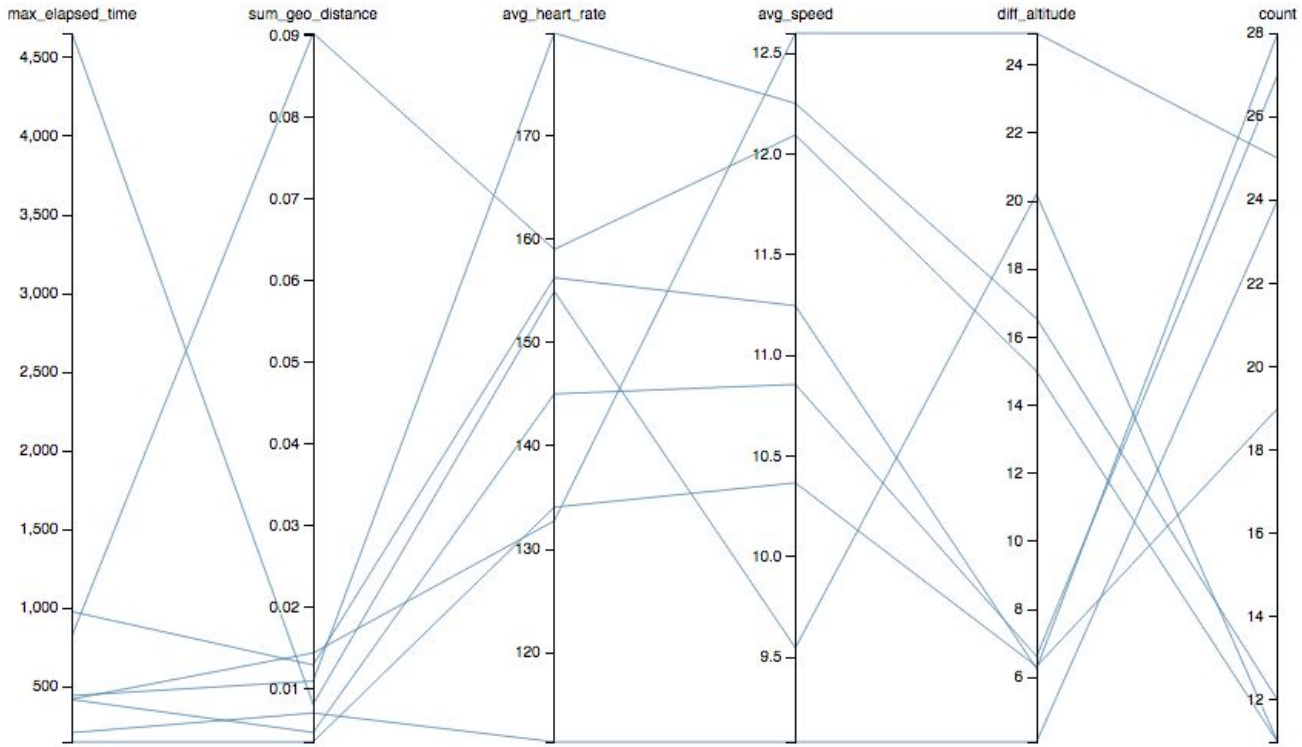


localhost:4040/stages/stage?id=1791&attempt=0

Initial Clustering

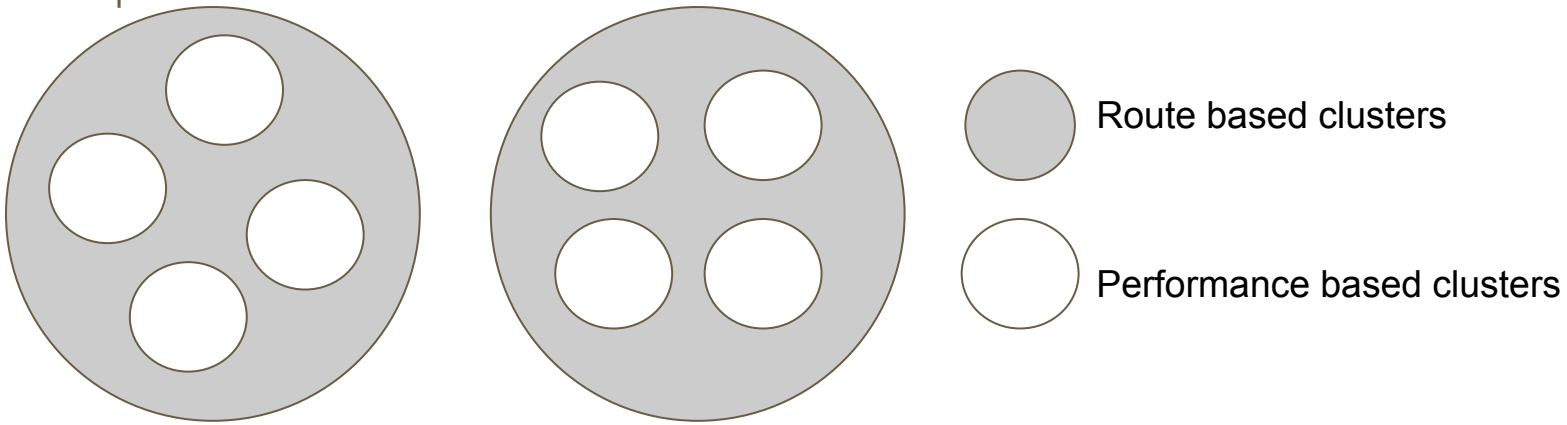
- Created summary statistics of workouts in the code
  - Altitude difference, distance, time, heart rate, speed
  - Required huge read of data from DB and calculation
  - Creating summaries in DB takes away ~ 50% of the process time of getting data to creating clusters
- Clustered based on all the above attributes
  - Elbow curve shows massive number of clusters to hit a good ‘elbow point’
  - Difficult to categorize too many clusters

Cluster - Parallel Coordinates Visualization



New approach - 2 Step Clustering

- Cluster based on route info first.
- Then in each of those clusters, cluster based on user performance info.
  - Given a user and a route, we can assign a smaller cluster to them, make better predictions!



Preliminary Regression Timing

(before adding slaves and 2 step clustering)

Number of Features = 4, Number of Workouts = 157, Number of Clusters = 7

Model(s)	CV Folds	Parameter 1 Map	Parameter 2 Map	Time
Linear Regression, Decision Tree Regressor, Gradient Boosted Trees, Random Forest Regressor	0	N/A (All Defaults)	N/A (All Defaults)	6 min 19 s
Linear Regression	2	MaxIter = [5, 10, 100]	RegParam = [0, 0.1, 0.01]	26 min 9 s
Decision Tree Regressor	2	MaxDepth = [3, 5]	MinInfoGain = [0, 0.1, 1]	15 min 32 s
Gradient Boosted Trees	2	MaxDepth = [3, 5]	MaxIter = [10, 20, 40]	33 min 59 s
Random Forest Regressor	2	MaxDepth = [3, 5]	NumIter = [10, 20, 40]	17 min 48 s

Next Steps

- Tune clusters based on visualizations
- Scale and time with more workouts and features
- Use regression coefficients to aid cluster tuning and feature engineering
  - Identify collinearity and interactions
- Create ensembles of regression models to reduce RMSE
- Create true test/train split to use with cross-validation