Prediction as a service with Ensemble Model trained in SparkML and Python ScikitLearn on 1Bn observed flight prices daily

Josef Habdank

Lead Data Scientist & Data Platform Architect at INFARE

jha@infare.com www.infare.com

www.linkedin.com/in/jahabdank

@jahabdank

SPARK SUMMIT EUROPE 2016

Leading provider of
Airfare Intelligence Solutions
to the Aviation Industry

INFARE

150 Airlines & Airports

Collect and processes
1.2 billion distinct airfares daily

7 offices worldwide

https://www.youtube.com/watch?v=h9cQTooY92E

ROYAL BRUNEI AIRLINES SAS WESTJET

What is this talk about?

- Ensemble approach for large scale DataScience
  - Online learning for huge datasets
  - thousands simple models are better than one very complex
- N-billion rows/day machine learning system architecture
  - Implementation of parallel online training of tens of thousands of models in Spark Streaming and Python ScikitLearn

Ensemble approach on billions of rows

Batch vs Online model training

Batch

$$\text{Bn}\left[\begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}\right.$$

- Large variety of options available (historical reasons)
- Often more accurate
- Does not scale well
- **Model might be missing critical latest information**

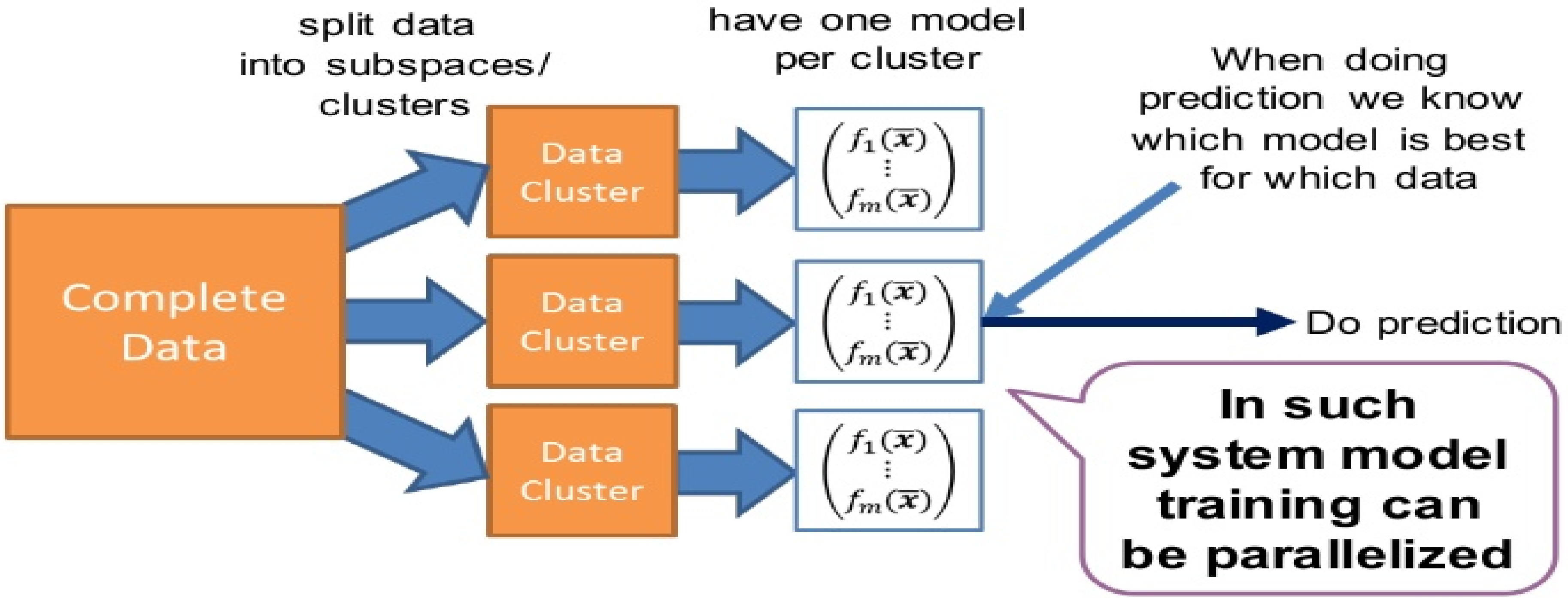
Online
can stream

$$\text{Bn}\left[\begin{array}{ll} \text{Mn} & \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix} \\ & \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix} \\ & \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix} \end{array}\right.$$

- Train on microbatches or individual observations
- Relies on Learning Rate and Sample Weighing
- Can be used in horizontally scalable environments
- **Model can be as up to date as possible**

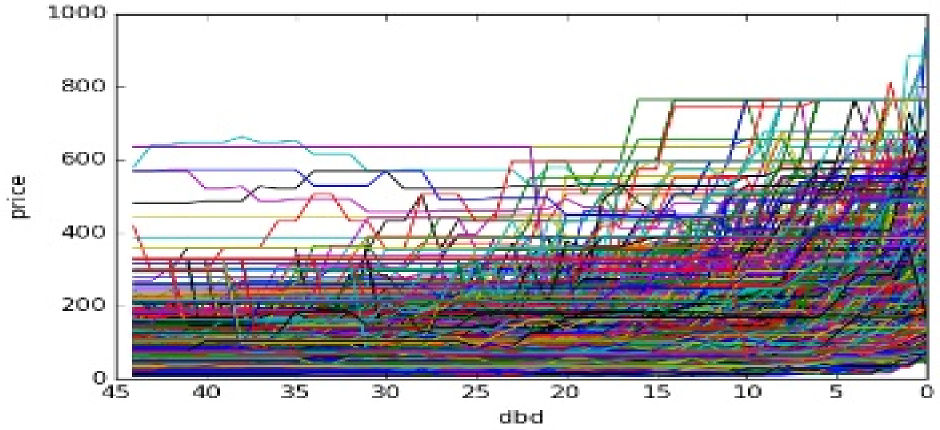
Especially critical in prediction of volatile signals

Ensemble approach to prediction in BigData



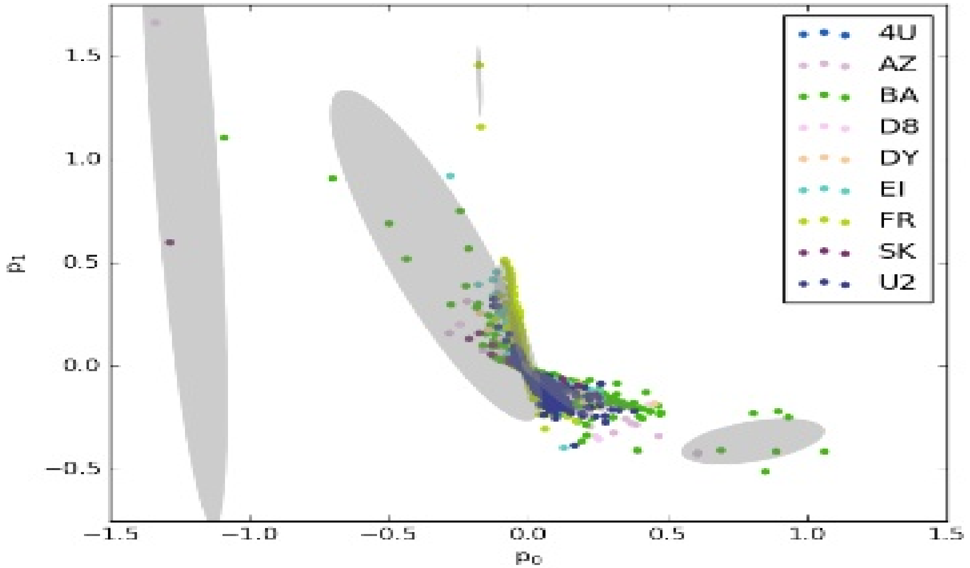
traditional ensemble mixes multiple models for one prediction,
here we simply select one best for the data segment

Segmenting the space



- Entire space consists of 1.2Bn time series
- Best results obtained when division is done using combination of knowledge (manual division) and clustering methods
- Optimal number of slices/clusters is between few thousand to hundreds of thousands
- Clustering methods need dimensionality reduction if subspace has still too many dimensions

Gaussian Mixture Model

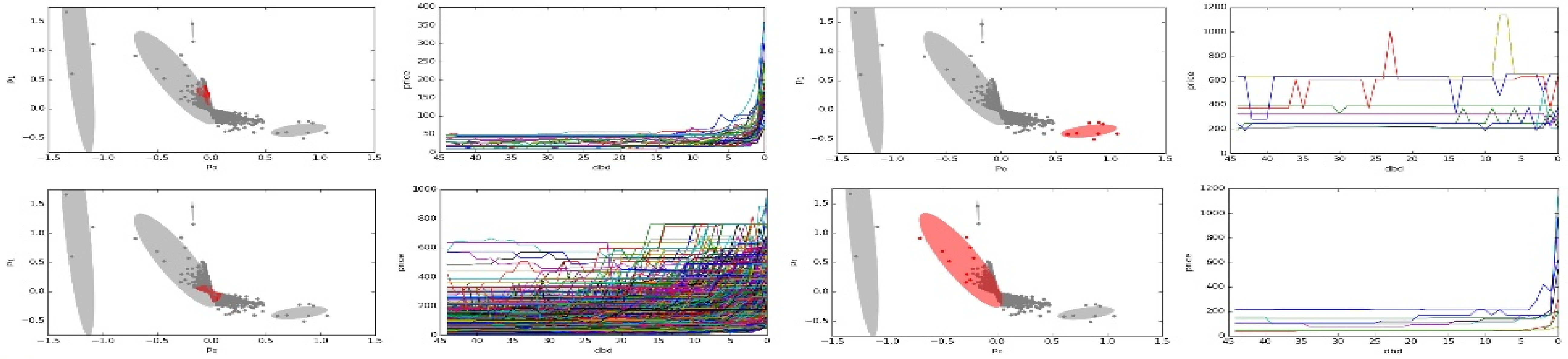


(showing only 2 first parameters)

- Fuzzy clustering method which gives probability of a point being in the cluster
- The probability could be used as a model weight, in case of model mixing

SparkML: org.apache.spark.ml.clustering.GaussianMixture
http://spark.apache.org/docs/latest/ml-clustering.html#gaussian-mixture-model-gmm

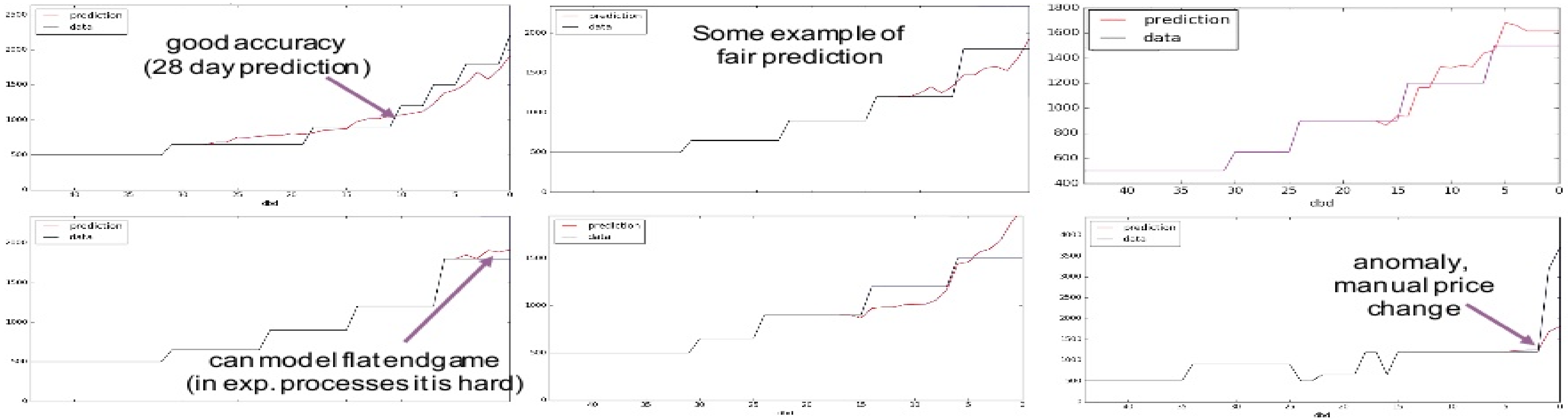
Gaussian Mixture Model Results



Feature and model selection

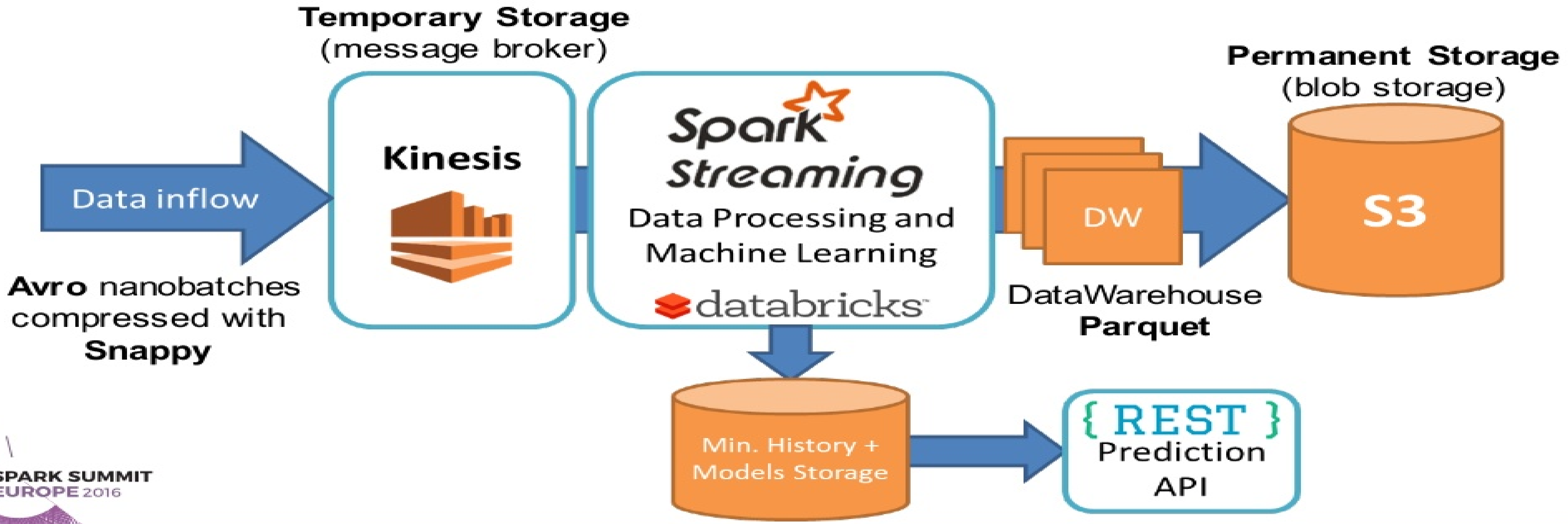
- OneHotEncoder:
  - can capture any nonlinear behavior
  - explodes exponentially dims
  - can reduce your problem to a hypercube
- Try assigning values to labels which carry information
  - $hour \in [0, 1, \ldots, 23] \rightarrow hour' \in [0.22, 0.45, \ldots, 0.03]$
- Try to capture nonlinear behavior using linear model, by adding meta-features
- Classification vs regression
  - if your problem can be converted into classification, try this as a first attempt
- Linear online models in Python:
  - sklearn.linear_model.SGDClassifier
    http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
  - sklearn.linear_model.SGDRegressor
    http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html
- Other interesting models:
  - whole sklearn.svm package
  - Kalman and ARIMA models
  - Particle Predictor (wrote own library)

Prediction results



N-billion rows/day machine learning architecture using DataBricks

N-billion rows/day machine learning architecture using DataBricks



SPARK SUMMIT EUROPE 2016

Training models in parallel in Spark Streaming

* we already know the clusters

Grouping in Spark DataFrames with collect_list()

```
> # FLight TimeSeries DataFrame
  fltsdf = dspmin \
    .groupBy("cluster_id") \
    .agg(
         expr("collect_list(days_before_departure) as
  days_before_departure_list"),
         expr("collect_list(price_inc) as price_inc_list"),
         expr("count(price_inc) as price_count"),
         expr("min(days_before_departure) as days_before_departure_min"),
         expr("max(days_before_departure) as days_before_departure_max")
        )
```

cluster_id	days_before_departure	price_inc	price_exc
345130379	18	404	380
345130379	24	60	34
345130379	26	128	102
345130379	29	240	214
345130379	40	352	326
345130379	42	124	100
345130379	86	270	244
345130379	103	242	216
345130379	104	218	194

cluster_id	days_before_departure_list	price_inc_list	price_exc_list
1780623700	[0,1,2,3,4,5,6,7,8,9,10,11,12,1	[158,134,112,112,92,62,52,	[158,134,112,112,92,62,52
268037612	[0,1,2,5,6,7,8,9,12,13,14,15,1	[66,46,46,46,18,18,18,18,1	[66,46,46,46,18,18,18,18,
-2009081663	[0,1,4,5,6,7,8,11,12,13,14,15,	[240,134,92,66,66,66,46,32	[240,134,92,66,66,66,46,3
-634639562	[0,1,2,3,4,5,6,7,8,9,10,11,12,1	[66,34,28,28,32,18,16,14,1	[66,34,28,28,32,18,16,14,
1411515385	[0,3,4,5,6,7,10,11,12,13,14,15	[190,112,112,92,92,92,78,6	[190,112,112,92,92,92,78,
-286460299	[0,1,2,3,6,7,8,9,10,13,14,15,1	[240,240,240,158,92,92,78,	[240,240,240,158,92,92,7
-610713405	[0,3,4,5,6,7,10,11,12,13,14,17	[134,66,66,60,34,34,32,28,	[134,66,66,60,34,34,32,26
1190364950	[0,3,4,5,6,7,10,11,12,13,14,17	[158,158,134,112,78,92,66,	[158,158,134,112,78,92,6

Wrapping model training in UDF

```
sgdlinreg_models = flt \
  .withColumn("sgdlinreg", sgdlinreg_udf(
      flt.cluster_id,
      flt.price_inc_list_zoh,
      flt.price_inc_list_lag1,
      flt.price_inc_list_lag2,
      flt.price_inc_list_lag3,
      flt.price_inc_list_lag4,
      flt.price_inc_list_lag5,
      flt.price_inc_list_lag6,
      flt.price_inc_list_lag7
    ))

display(sgdlinreg_models.select("sgdlinreg"))
```

cluster_id	days_before_departure_list	price_inc_list	price_exc_list
1780623700	[0,1,2,3,4,5,6,7,8,9,10,11,12,1	[158,134,112,112,92,82,52,	[158,134,112,1
268037612	[0,1,2,5,6,7,8,9,12,13,14,15,1	[66,46,46,46,18,18,18,18,1	[66,46,46,46,18
-2009081663	[0,1,4,5,6,7,8,11,12,13,14,15,	[240,134,92,66,66,66,46,32	[240,134,92,66,
-834839582	[0,1,2,3,4,5,6,7,8,9,10,11,12,1	[66,34,28,28,32,18,18,14,1	[66,34,28,28,32
1411515385	[0,3,4,5,6,7,10,11,12,13,14,15	[190,112,112,92,92,92,78,6	[190,112,112,92
-286460299	[0,1,2,3,6,7,8,9,10,13,14,15,1	[240,240,240,158,92,92,78,	[240,240,240,1
-610713405	[0,3,4,5,6,7,10,11,12,13,14,17	[134,66,66,60,34,34,32,26,	[134,66,66,60,3

sgdlinreg
[0.312164452051516,0.2500195199688649,0.1469679429207753,0.073977448
[0.1333874452618317,0.1211296843112874,0.1113091276669698,0.10185600
[0.2355943347534875,0.1960476424388375,0.1619729900436198,0.12321143
[-0.1001631560308695,-0.1645861349644310,-0.2781127171197163,-0.33133
[0.1466883871278040,0.1312359972135596,0.1201749975059930,0.0993076
[1.168599501383796,0.4523697946893281,0.5174869129611258,0.49467024834
[0.1568171724141056,0.1389715287935264,0.1163411458288500,0.10410808

Wrapping model training in UDF

```
def sgdlinreg(cluster_id, x0, x1, x2, x3, x4, x5, x6, x7):

    # create a data matrix from columns
    X = np.transpose(np.matrix([x1, x2, x3, x4, x5, x6, x7]))

    # normalize the data
    X = normalize_using_db_norm(cluster_id, X)
    Y = np.reshape(normalize_using_db_norm(cluster_id, x0)[0],\
                   (1, len(x0)))[0]

    # generate sample weights to adjust the learning rate
    sample_weights = generate_sample_weights(X)

    # get the model:
    sgd_model = get_sgdregressor_from_db(cluster_id)

    # model train
    sgd_model.partial_fit(X, Y, sample_weights)

    # copy the coeffs (list of numpy floats) into native list
    # of python doubles (for Spark type compatibility)
    retval = []
    [retval.append(p.item()) for p in sgd_model.coef_]

    return(retval)

sgdlinreg_udf = udf(sgdlinreg, ArrayType(DoubleType()))
```

Prepare the Matrix with inputs for model training

Normalize the data using normalization defined for this particular cluster

Generate sample weights which enable controlling the learning rate

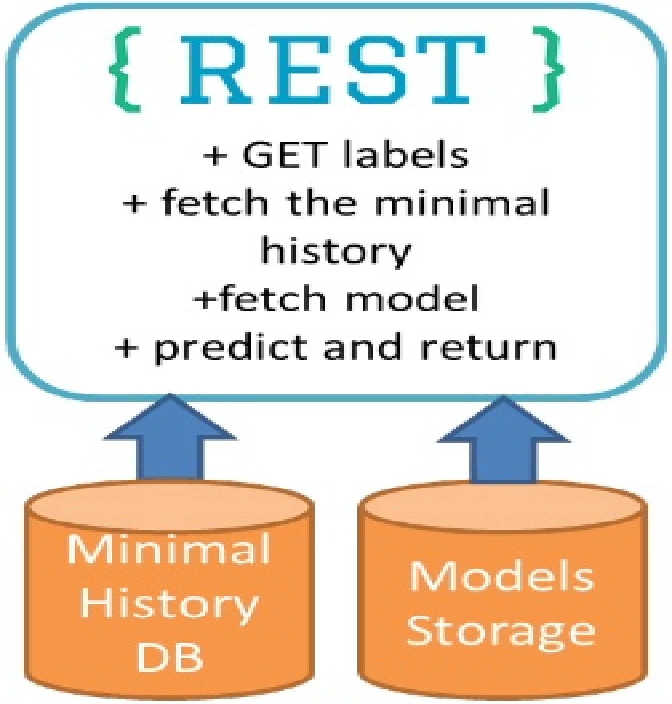
Get the current model from DB (use in memory DB for fast response)

Preform partial fit using the sample weights

Important trick:
Converts numpy.ndarray[numpy.float64] into native python list[float] which then can be autoconverted to Spark List[Double]

Register UDF which returns Spark List[Double]

Time Series Prediction as a Service



- Provided the labels identify time series and lookup the model
- Get historical data (performance is the key)
- Recursively predict next price, shifting the window for the desired length

$$f\begin{pmatrix} x_{n-l} \\ \vdots \\ x_n \end{pmatrix} = x_{n+1} \Rightarrow f\begin{pmatrix} x_{n-l+1} \\ \vdots \\ x_{n+1} \end{pmatrix} = x_{n+2} \Rightarrow \ldots$$

- The same workflow for any model: SGDClassifier, SGDRegressor, ARIMA, Kalman, Particle Predictor

Summary

- **Spark + Python is AWESOME for DataScience ☺**
- **Large scale DataScience needs correct infrastructure** (Kafka-Kinesis, Spark Streaming, in memory DB, Notebooks)
- It is much easier to work with large volumes of models, then very few ones
- Gaussian Mixture is great for fuzzy clustering, has very mature and fast implementations
- Spark DataFrames with UDF can be used to efficiently palletize the model training in tens and even hundreds of thousands models

Want to work with cutting edge
100% Apache Spark + Python projects? We are hiring!!!

Senior Data Scientist, working with
Apache Spark + Python doing Airfare/price forecasting

Senior Big Data Engineer/Senior Backend Developer,
working with Apache Spark/S3/MemSql/Scala + MicroAPIs

job@infare.com
http://www.infare.com/jobs/

THANK YOU!

Q/A

Remember, we are hiring!

Josef Habdank
Lead Data Scientist & Data Platform Architect at INFARE
jha@infare.com www.infare.com
www.linkedin.com/in/jahabdank
@jahabdank

SPARK SUMMIT EUROPE 2016