



Microsoft Azure 自習書シリーズ

Microsoft Azure HDInsight トレーニング

Step-4: HDInsight を使用した機械学習アルゴリズムの活用

この自習書では、Microsoft が提供するパブリッククラウドサービスである Microsoft Azure を利用し、HDInsight を使用したデータ分析の一連の流れをハンズオン形式で学習体験します。

更新履歴

版数	発行日	更新履歴
第 1 版	2017 年 7 月 XX 日	初版発行
第 2 版	2018 年 1 月 24 日	第 2 版発行

目次

1. はじめに	4
2. Spark MLlib の概要	5
3. Spark MLlib を活用した予測データの生成	7

1. はじめに

本自習書をご利用いただきありがとうございます。この自習書では、Microsoft が提供するパブリッククラウドサービスである Microsoft Azure を利用し、HDInsight を使用したデータ分析の一連の流れをハンズオン形式で学習体験します。

2. Spark MLlib の概要

この章では、Spark MLlib の概要について紹介いたします。

トレーニングの冒頭で述べた通り、**Apache Spark** はいくつかのライブラリがパッケージングされたデータ分析スイートになっています。キーとなるコンポーネントは以下の通りです。

- **SparkSQL**

SparkSQL は様々なデータソースからのデータを読み込み、自由に加工することができるライブラリです。読み込めるデータは CSV,JSON といったローカルファイルから、様々な RDB, NoSQL にも対応します。さらに **AWS S3, Azure Storage Blob** といったクラウドストレージへの対応もビルトインのライブラリによって簡便に行うことができます。**Apache Spark2.0** 以降 **SparkSQL** は **SQL:2003** 規格に準拠した **SQL** 文を使用したデータ加工にも対応しています。

- **Spark Streaming**

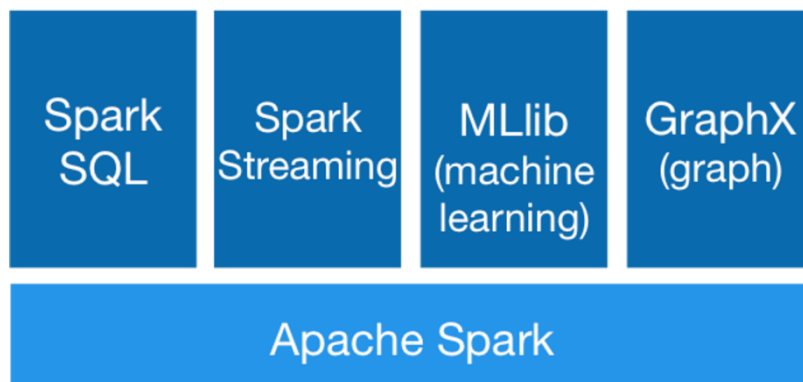
Spark Streaming は様々なストリームデータを取り込み、逐次処理を行うことができるライブラリです。IoT データ、SNS データのような不定期、非定型のデータにうまく対応し、リアルタイムのデータ加工、保存に対応します。

- **MLlib**

MLlib はまさに機械学習アルゴリズムのパッケージです。さまざまな回帰(**Regression**)、分類(**Classification**)、クラスタリングなどのアルゴリズムが搭載されています。使用するためには基本的な統計の知識が必要ですが、既存のデータから将来を予測したり、大量のデータ群の中から異常値(**Anomaly**)を発見したりすることができるようになります。よくみかけるリコメンデーションエンジンの機能も使用することが可能です。

- **GraphX**

GraphX はグラフデータを操作することができるライブラリです。日常的に使用することはあまりないかもしれませんが、解決したい課題によってはグラフデータを使用することで極めて効率的に、高速にデータを処理することができる場合があります。



Apache Spark の機械学習ライブラリは、歴史的な理由から 2 種類のライブラリ群が存在します。1 つは **Apache Spark 1.x** 系からある **MLlib**(以下 **spark.mllib**)と、**Apache Spark 1.5** から実装され始め、**2.0**

で正式リリースとなった、DataFrame ベースの MLlib(以下 spark.ml)があります。

2つのライブラリの最も大きな違いは対応するデータ形式です。spark.mllib は Spark に従来からある RDD(Resilient Distributed Dataset)を入出力データとして扱うのに対して、spark.ml はより簡便に扱うことができる DataFrame を入出力データとして扱うことができます。

HDInsight で使用できる MLlib は、どのバージョンの HDInsight をデプロイするかによって変わってきます。このトレーニングでは HDInsight version 3.5 を使用しており、使用できる Spark のバージョンは 2.0.2 となり、MLlib は DataFrame ベースの spark.ml となります。Spark.mllib も後方互換性のために残されていますが、Maintenance only となっており、新規機能が追加されることはありません。今後のプロジェクトで Apache Spark の MLlib を使用する場合には、どのバージョンの HDInsight を使用するかについても気を付けるようにしてください。

Apache Spark の機械学習ライブラリ MLlib は多種多様なライブラリが実装されており、ここでそのすべてを網羅することはできません。また機械学習ライブラリを使用する前提となる、基礎的な統計の知識についてもここでは触れません。ここではいくつかのトレーニングリソースを記述します。

➤ 機械学習について

✧ Microsoft Virtual Academy: Data Series: Analytics: Big Data: Spark on HDInsight

https://mva.microsoft.com/en-US/training-courses/data-series-analytics-big-data-spark-on-hdinsight-17760?!=wVyEnEn4D_2611787171

➤ Apache Spark MLlib

✧ Spark Summit

<https://spark-summit.org/>

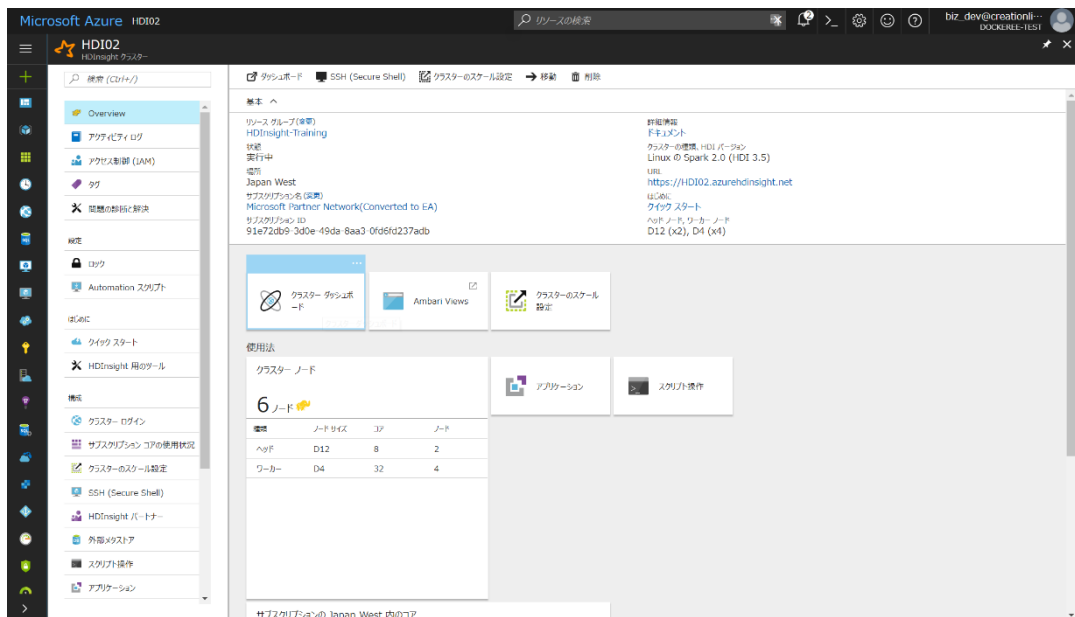
カンファレンス内で 1,2 日のハンズオントレーニングが併設されており、Apache Spark を活用した機械学習アプリケーションの構築を体験することができます。

3. Spark MLlib を活用した予測データの生成

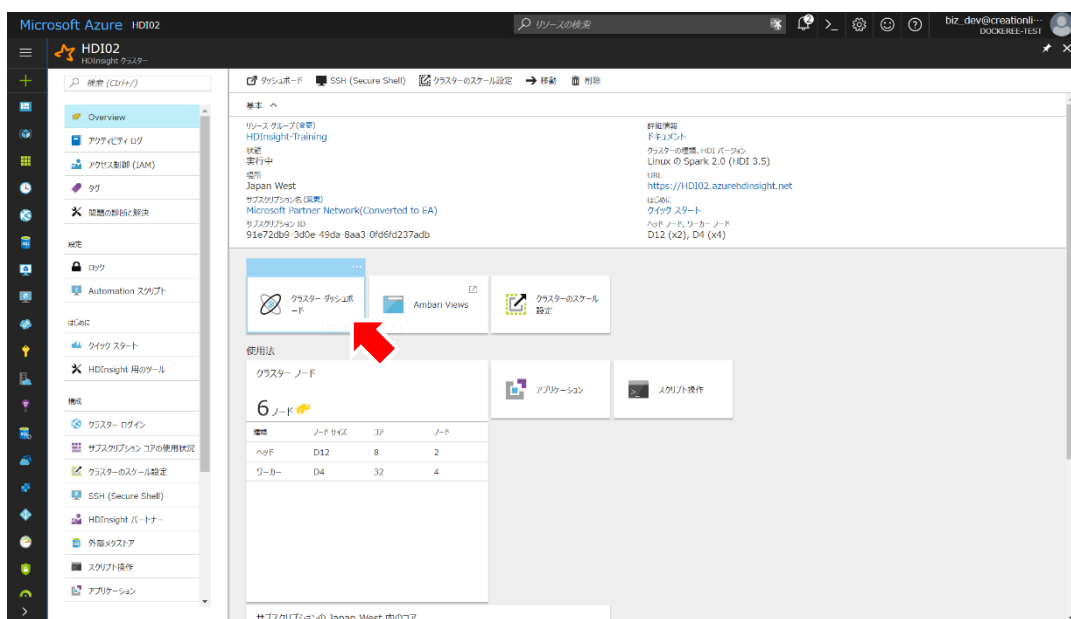
ここでは Spark MLlib を使用して、既にあるデータから予測データを生成してみます。ここでは最も簡便な線形回帰(Linear Regression)を使用します。

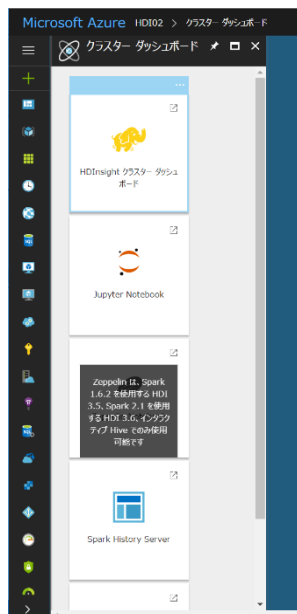
ここでは配布資料に同梱された「linearRegressionDemo.ipynb」を使用します。

1. 実習用 PC で Internet Explorer / Google Chrome / Firefox などの Web ブラウザを起動して Azure ポータルを開き、HDInsight のブレードを開きます。

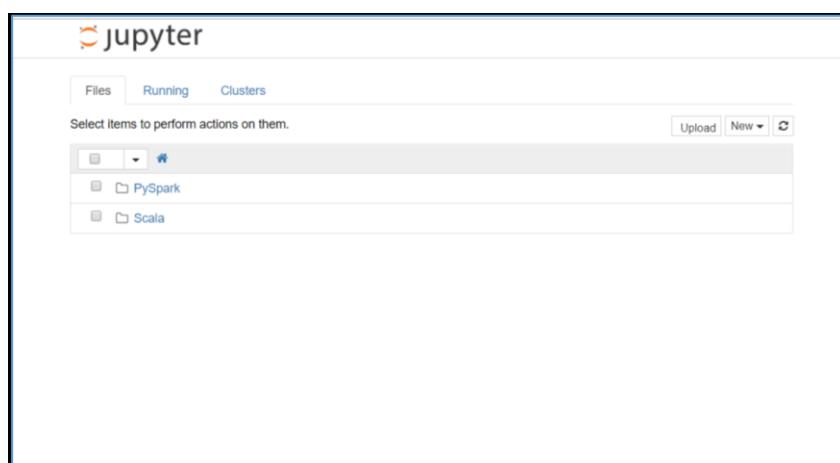
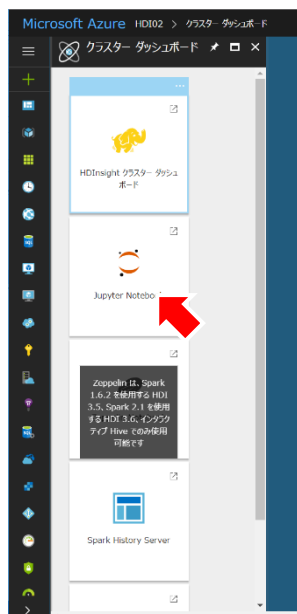


2. [クラスターダッシュボード] ボタンをクリックし、クラスターダッシュボードブレードを開きます

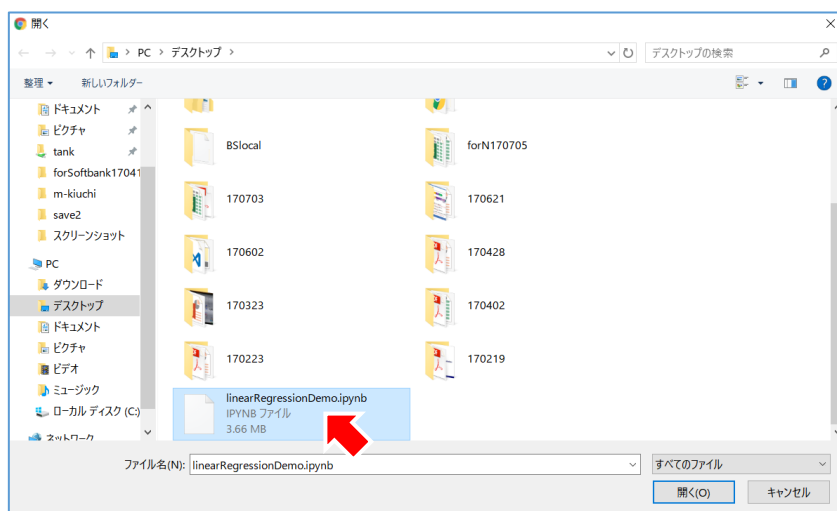
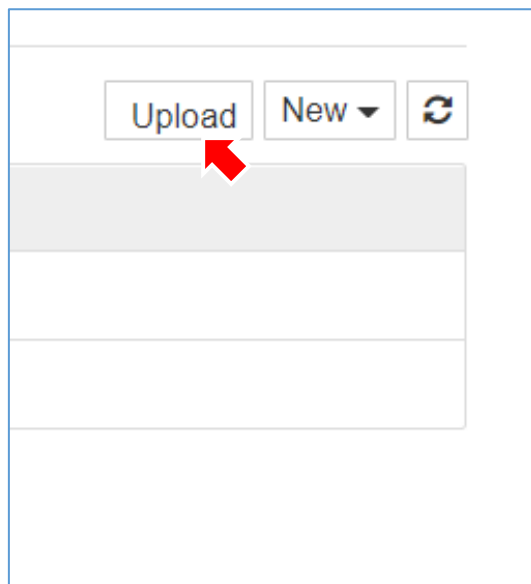




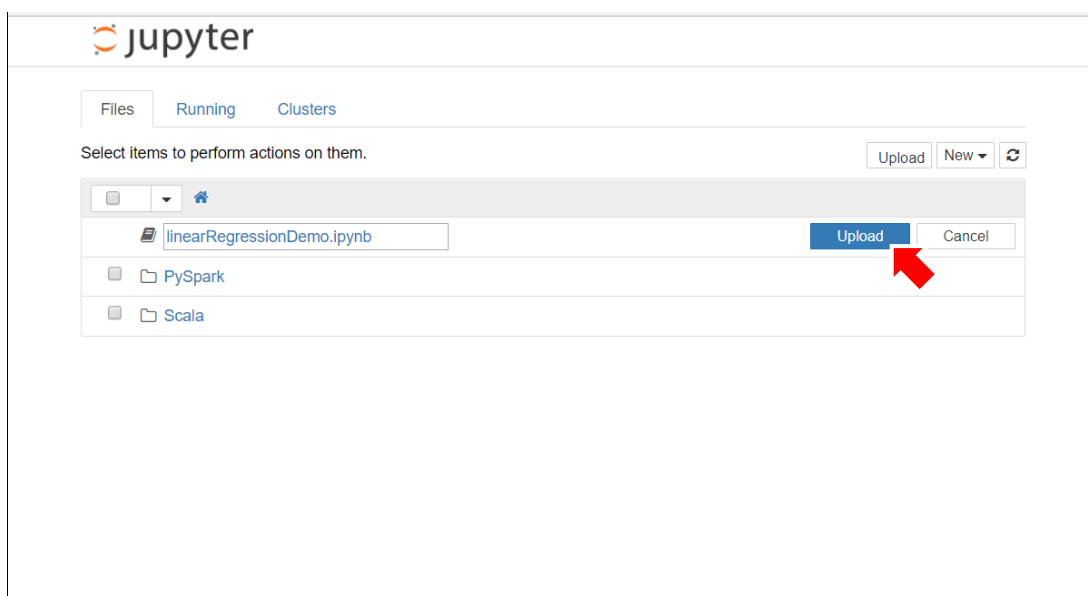
3. [Jupyter Notebook]パネルをクリックし、Jupyter Notebook を開きます。

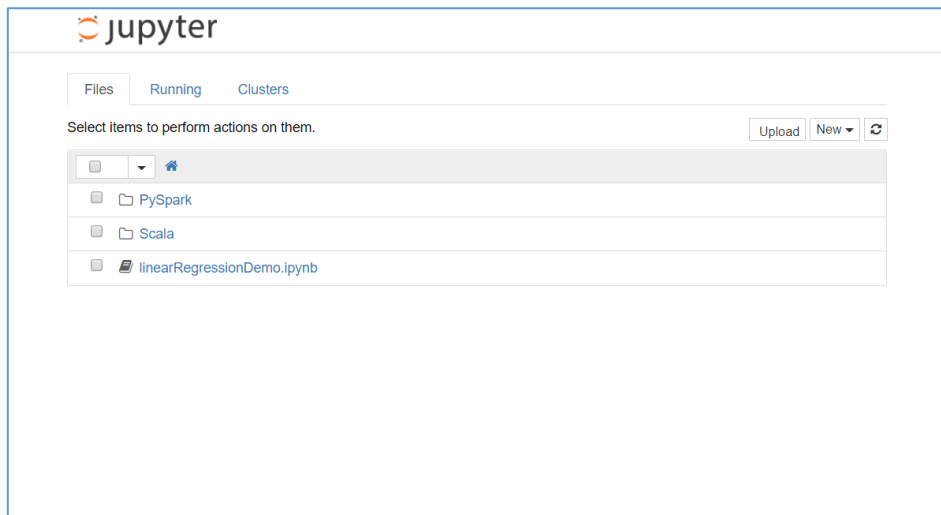


4. 画面右上の[Upload]ボタンをクリックし、配布資料中の “linearRegressionDemo.ipynb” を選択します。

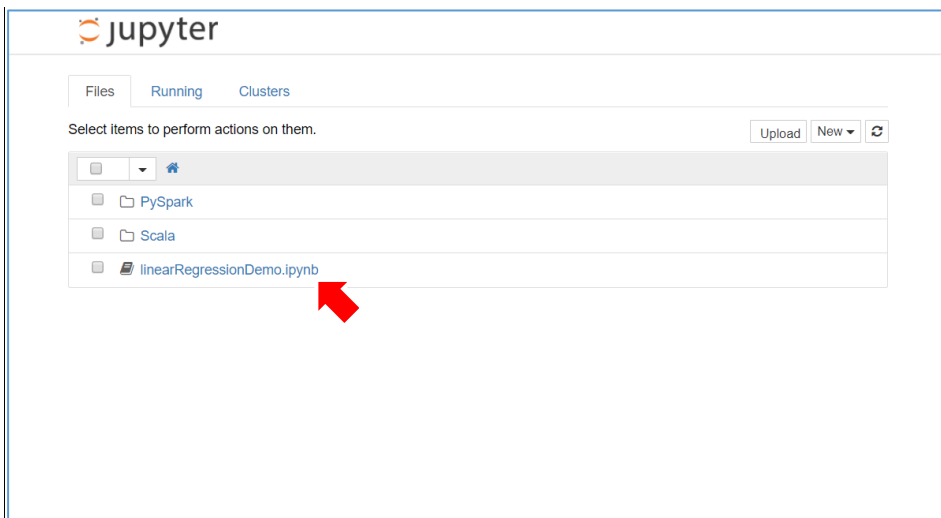


5. [Upload]ボタンをクリックすると、ノートブックが HDInsight にアップロードされます。

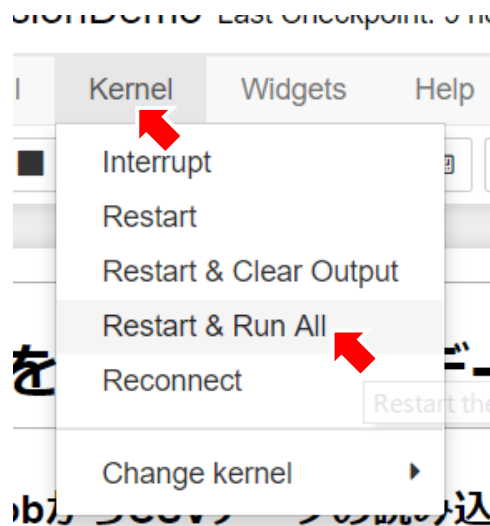




6. アップロードしたノートブックをクリックすると、内容の詳細画面が開きます。



7. ノートブックの内容を実行するには、上部のメニューから[Kernel] – [Restart & Run All]を選択します。実行環境にもよりますが、おおよそ 5~10 分で処理が完了します。



ここからは内容の説明に入ります。

1. まず、Azure Storage Blob にプリセットされているデモデータの CSV ファイルを読み出しています。読み出したデータは DataFrame 形式になり、変数 “csvFile” に格納されます。

```
csvFile =
spark.read.csv('wasb:///HdiSamples/HdiSamples/SensorSampleData/hvac/HVAC.csv',
header=True, inferSchema=True)
```

2. まずはデモデータの形式を確認します。後で使用するために、Hive テーブル(テーブル名: hvac)として保存します。

```
csvFile.createOrReplaceTempView("HVACTemp")
sqlContext.sql("drop table if exists hvac")
sqlContext.sql("create table hvac as select * from HVACTemp")
```

3. 保存した Hive テーブル“hvac”から読み出します。“%%”で始まる語は Jupyter Notebook 固有の“Magic” 機能と呼び出す識別子です。ここでは “%%local”でこのセルが分散処理の対照ではないこと、“%%sql -o hvacdf”で、Hive テーブルに SQL クエリを実行し、結果を Pandas の DataFrame として変数保持することを示しています。

```
%%local
%%sql -o hvacdf
SELECT * FROM hvac
```

【備考】 この部分を実行した際に以下のエラーが表示されることがあります。これは HDInsight が使用する Jupyter Notebook のバージョンが古いことが原因です。このエラーは本トレーニングの進行には影響を与えません。

エラー内容

```

AttributeErrorTraceback (most recent call last)
/usr/bin/anaconda/lib/python2.7/site-packages/IPython/core/formatters.pyc in __call__(self, obj)
    902         pass
    903     else:
--> 904         printer(obj)
    905         return True
    906         # Finally look for special method names

/usr/bin/anaconda/lib/python2.7/site-packages/autovizwidget/widget/utils.pyc in
display_dataframe(df)
    114
    115 def display_dataframe(df):
--> 116     selected_x = select_x(df)
    117     selected_y = select_y(df, selected_x)
    118     encoding = Encoding(chart_type=Encoding.chart_type_table, x=selected_x, y=selected_y,

/usr/bin/anaconda/lib/python2.7/site-packages/autovizwidget/widget/utils.pyc in select_x(data, order)
    70     _validate_custom_order(order)
    71
---> 72     d = _classify_data_by_type(data, order)
    73
    74     chosen_x = None

/usr/bin/anaconda/lib/python2.7/site-packages/autovizwidget/widget/utils.pyc in
_classify_data_by_type(data, order, skip)
    48     for column_name in data:
    49         if column_name not in skip:
---> 50             typ = infer_vegalite_type(data[column_name])
    51             d[typ].append(column_name)
    52

/usr/bin/anaconda/lib/python2.7/site-packages/autovizwidget/widget/utils.pyc in infer_vegalite_type(data)
    14     """
    15
---> 16     typ = pd.api.types.infer_dtype(data)
    17
    18     if typ in ['floating', 'mixed-integer-float', 'integer',
AttributeError: 'module' object has no attribute 'api'

```

4. 読み出した Pandas DataFrame の内容を、Matplotlib を使用して視覚化します。ここでは buildingid=1 の actualtemp を散布図としてプロットしています。

```

%%local

%matplotlib inline

import matplotlib

import matplotlib.pyplot as plt

import matplotlib.colors as colors

colorlist = colors.cnames

a = hvacdf[hvacdf["buildingid"] == 1].copy().sort_values("date").drop_duplicates(["date"])

hfmt = matplotlib.dates.DateFormatter('%Y/%m/%d %H:%M:%S')

fig = plt.figure(figsize=(20,10))

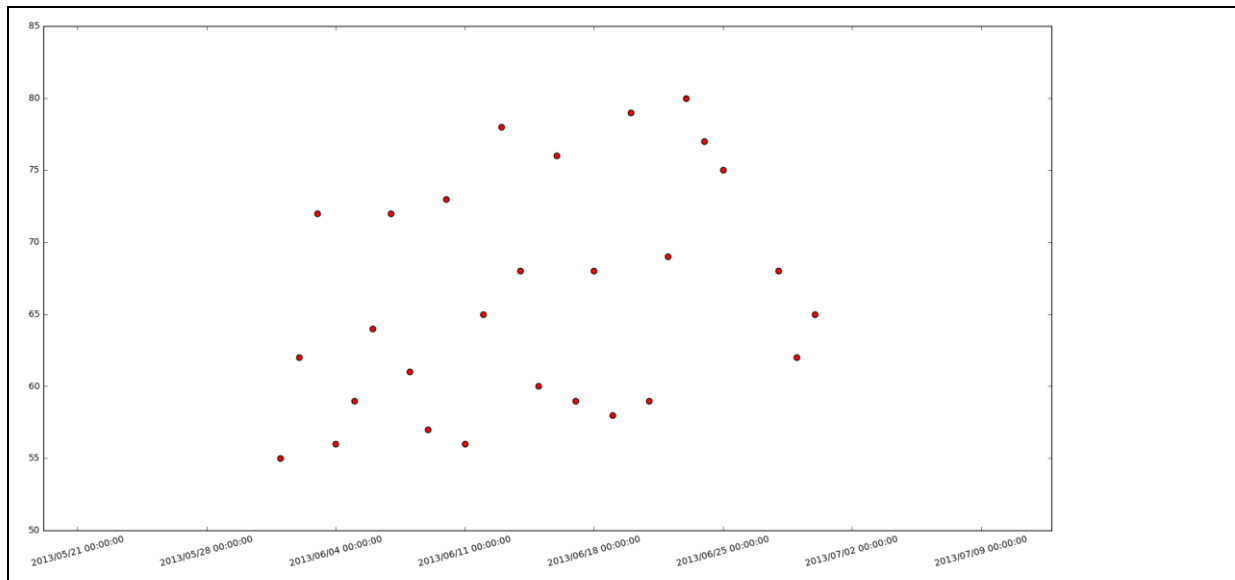
ax = fig.add_subplot(1,1,1)

ax.xaxis.set_major_formatter(hfmt)

plt.setp(ax.get_xticklabels(), rotation=15)

plt.scatter(a.date.tolist(), a.actualtemp.tolist(), s=40, c="#ff0000")

```



5. 同じ Hive テーブル"hvac"のデータを読み込み、Apache Spark の DataFrame にします。合わせて後の可視化のために別の Hive テーブル(テーブル名: training)として保存します。

元 DataFrame 中の列"date"を解釈し、epoch-second の列を追加したうえで、特徴変数としてベクトル化します。列"feature"が説明変数, 列"actualtemp"が目的変数となります。このデータが線形回帰モデルのトレーニングデータ(教師データ)となります。

```
from pyspark.sql.functions import udf
from pyspark.ml.feature import VectorAssembler
from pyspark.sql.types import *
import datetime as dt

def dateToSeconds(datestr):
    [month, day, year] = datestr.split('/')
    datevar = dt.datetime(int(year)+2000, int(month), int(day))
    elseconds = int((datevar - dt.datetime(1970,1,1)).total_seconds())
    return(elseconds)

udf_dateToSeconds = udf(dateToSeconds, IntegerType())

vectorAssembler = VectorAssembler(inputCols=["seconds"],outputCol="features")

df = sqlContext.sql("SELECT * FROM hvac").filter("buildingid = 1").dropDuplicates(["date"])
df2 = df.withColumn("seconds", udf_dateToSeconds(df["date"]))
df3 = vectorAssembler.transform(df2)

df3.registerTempTable("traindf")
sqlContext.sql("drop table if exists training")
sqlContext.sql("create table training as select date, actualtemp from traindf")
```

6. 次に予測データを格納するための空の **DataFrame** を作成します。この **DataFrame** には予測のための説明変数のみを格納します。

```
def futureDate(buildingid):
    startDate = 1
    ar = []
    for i in range(0,30):
        day = "7/" + str(startDate+i) + "/13"
        ar.append((day,None,None,0,None,None,buildingid))
    return(ar)

schema = StructType([
    StructField("date", StringType(), False),
    StructField("time", StringType(), True),
    StructField("targettemp", IntegerType(), True),
    StructField("actualtemp", IntegerType(), True),
    StructField("system", IntegerType(), True),
    StructField("systemage", IntegerType(), True),
    StructField("buildingid", IntegerType(), True)])

c = sqlContext.createDataFrame(futureDate(1), schema)
c2 = c.withColumn("seconds", udf_dateToSeconds(c["date"]))
c3 = vectorAssembler.transform(c2)
```

7. トレーニングデータを使用して線形回帰モデルをトレーニングします。トレーニングされたモデルは変数**model**に格納されます。変数 **model** のメソッド **translate** を使用して、予測データを生成します。生成した予測データは、別の **Hive** テーブル(テーブル名: **prediction**)として保存します。

```
from pyspark.ml.regression import LinearRegression
lr = LinearRegression(featuresCol="features", labelCol="actualtemp",
predictionCol="predicttemp", maxIter=10, regParam=0.3)
model = lr.fit(df3)

result = model.transform(c3)
result2 = result.drop("actualtemp").withColumnRenamed("predicttemp", "actualtemp")

result2.registerTempTable("preddf")
sqlContext.sql("drop table if exists prediction")
sqlContext.sql("create table prediction as select date, actualtemp from preddf")
```

8. 保存した Hive テーブル"training", "prediction"をそれぞれ呼び出し、Matplotlib を使用して可視化します。

```
%%local
%%sql -o traindf2
SELECT * FROM training
```

```
%%local
%%sql -o preddf2
SELECT * FROM prediction
```

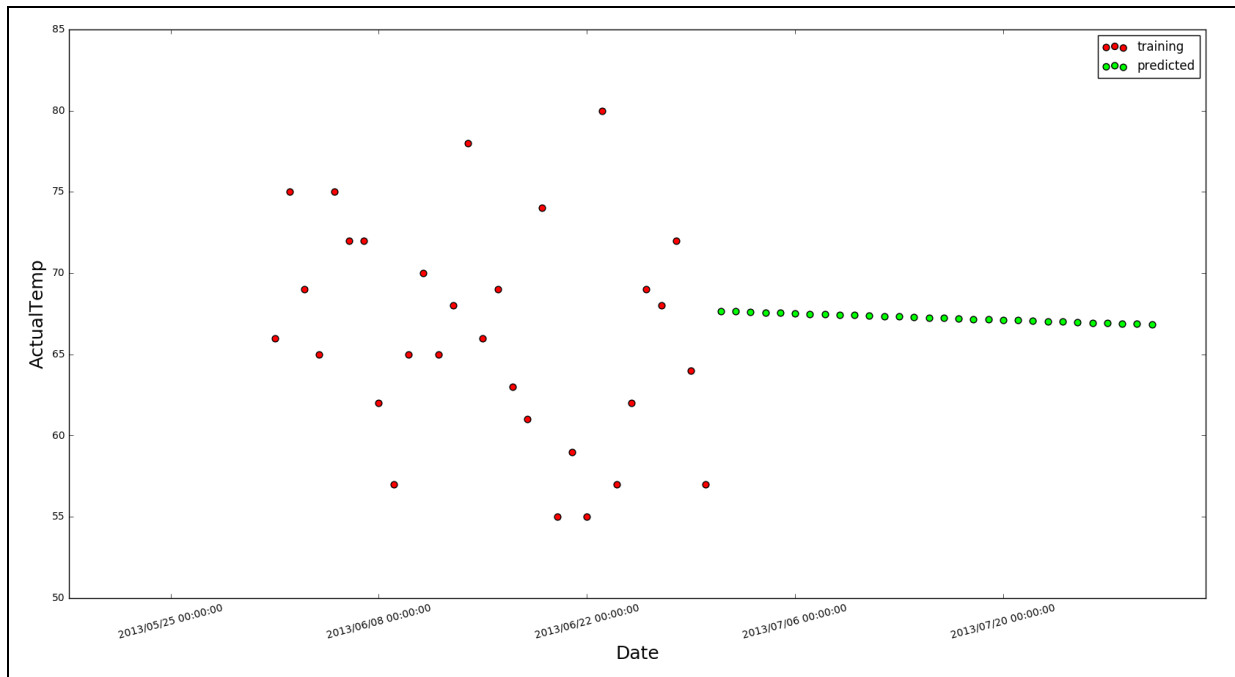
```
%%local
import matplotlib
import matplotlib.colors as colors
colorlist = colors.cnames

a = traindf2.copy().sort_values("date").drop_duplicates(["date"])
b = preddf2.copy().sort_values("date").drop_duplicates(["date"])

hfmt = matplotlib.dates.DateFormatter('%Y/%m/%d %H:%M:%S')
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(1,1,1)
ax.xaxis.set_major_formatter(hfmt)
plt.setp(ax.get_xticklabels(), rotation=15)

plt.scatter(a.date.tolist(), a.actualtemp.tolist(), s=40, c="#ff0000",
label="training")
plt.scatter(b.date.tolist(), b.actualtemp.tolist(), s=40, c="#00ff00",
label="predicted")

plt.xlabel(u'Date', fontsize=18)
plt.ylabel(u'ActualTemp', fontsize=18)
plt.legend()
```



【備考】この部分を実行した際に以下のエラーが表示されることがあります。これは HDInsight が使用する Jupyter Notebook のバージョンが古いことが原因です。このエラーは本トレーニングの進行には影響を与えません。

エラー内容

```
AttributeErrorTraceback (most recent call last)
/usr/bin/anaconda/lib/python2.7/site-packages/IPython/core/formatters.pyc in __call__(self, obj)
    902         pass
    903         else:
--> 904             printer(obj)
    905             return True
    906             # Finally look for special method names

/usr/bin/anaconda/lib/python2.7/site-packages/autovizwidget/widget/utils.pyc in
display_dataframe(df)
    114
    115 def display_dataframe(df):
--> 116     selected_x = select_x(df)
    117     selected_y = select_y(df, selected_x)
    118     encoding = Encoding(chart_type=Encoding.chart_type_table, x=selected_x, y=selected_y,

/usr/bin/anaconda/lib/python2.7/site-packages/autovizwidget/widget/utils.pyc in select_x(data, order)
    70     _validate_custom_order(order)
    71
--> 72     d = _classify_data_by_type(data, order)
    73
    74     chosen_x = None

/usr/bin/anaconda/lib/python2.7/site-packages/autovizwidget/widget/utils.pyc in
_classify_data_by_type(data, order, skip)
    48     for column_name in data:
    49         if column_name not in skip:
--> 50             typ = infer_vegalite_type(data[column_name])
    51             d[typ].append(column_name)
    52

/usr/bin/anaconda/lib/python2.7/site-packages/autovizwidget/widget/utils.pyc in infer_vegalite_type(data)
    14     """
    15
--> 16     typ = pd.api.types.infer_dtype(data)
    17
    18     if typ in ['floating', 'mixed-integer-float', 'integer',

AttributeError: 'module' object has no attribute 'api'
```

本章の内容は以上です。