SmarterAnalytics IBM

Machine Learning with Big R Tutorial

Learn how to use machine learning with IBM® InfoSphere® BigInsights™ Big R to perform statistical analysis and modeling on big data. You must download, license, and install the appropriate R software before using Big R.

About

In this scenario, you will perform statistical analysis and modeling on a sample *airline* data set to leverage the machine learning capabilities of Big R. Use Big R to predict the arrival delay for the flights by using other columns as predictors.

The *airline* data set contains a small sample of US flight information from 1987-2008 provided in the Big R package.

Procedure

Run commands from an R environment.

1. Access the airline dataset on HDFS.
2. Perform data transformations required for the machine learning algorithms.
3. (Optional) Calculate descriptive statistics.
4. Create training and testing sets to use for the following models:
   - Create a *linear regression* model for the arrival delay of flights, and use it to generate predictions.
   - Create a *support vector machine* classifier for the arrival delay of flights.

Accessing data on HDFS

```
# Connect to BigInsights.
> bigr.connect(host="myhost.ibm.com", port=7052, user="my_user",
               password="my_password")

# Access the airline dataset on HDFS. useMapReduce by default is TRUE.
# The sample data set is not large, so set the parameter to FALSE
# to run the data faster.
# To run the example on a large dataset, set the useMapReduce parameter
# to TRUE.
> airline <- bigr.frame(bigr.env$TEXT_FILE, "/user/airline_lab.csv", ",",
          coltypes=ifelse(1:29 %in% c(9,11,17,18,23), "character", "integer"),
          header=TRUE, na.string = "NA", useMapReduce=FALSE)

# Display the data set. The data set has 29 columns.
> str(airline)
'bigr.frame': 29 variables:
 $ Year               : int 2004 2004 2004 2004 2004 2004
 $ Month              : int 2 2 2 2 2 2
```

```
 $ DayofMonth        : int 12 16 18 19 21 24
 $ DayOfWeek         : int 4 1 3 4 6 2
 $ DepTime           : int 633 2115 700 1140 936 1117
 $ CRSDepTime        : int 635 2120 700 1145 935 1120
 $ ArrTime           : int 935 2340 817 1427 1036 1922
 $ CRSArrTime        : int 930 2350 820 1420 1035 1930
 $ UniqueCarrier     : chr "B6" "B6" "B6" "B6" "B6" "B6"
 $ FlightNum         : int 165 199 2 67 68 206
 $ TailNum           : chr "N553JB" "N570JB" "N544JB" "N570JB" "N544JB" "N548JB"
 $ ActualElapsedTime: int 182 325 77 167 60 305
 $ CRSElapsedTime    : int 175 330 80 155 60 310
 $ AirTime           : int 162 114 49 141 41 468
 $ ArrDelay          : int 5 -10 -3 7 1 -8
 $ DepDelay          : int -2 -5 0 -5 1 -3
 $ Origin            : chr "JFK" "JFK" "JFK" "RSW" "JFK" "LGB"
 $ Dest              : chr "TPA" "LAS" "BUF" "JFK" "SYR" "JFK"
 $ Distance          : int 1005 2248 301 1074 209 2465
 $ TaxiIn            : int 3 8 2 7 3 7
 $ TaxiOut           : int 17 23 26 19 16 10
 $ Cancelled         : int 0 0 0 0 0 0
 $ CancellationCode : chr "NA" "NA" "NA" "NA" "NA" "NA"
 $ Diverted          : int 0 0 0 0 0 0
 $ CarrierDelay      : int 0 0 0 0 0 0
 $ WeatherDelay      : int 0 0 0 0 0 0
 $ NASDelay          : int 0 0 0 0 0 0
 $ SecurityDelay     : int 0 0 0 0 0 0
 $ LateAircraftDelay: int 0 0 0 0 0 0
```

Perform data transformations

```
# Filter relevant columns for modeling and statistical analysis.
> airlineFiltered <- airline[, c("Month", "DayofMonth", "DayOfWeek", "CRSDepTime",
                              "Distance", "ArrDelay")]

# Discretize the ArrDelay column into three categories: Low, Medium, and High.
# The categories are used to make predictions.
> airlineFiltered$Delay <- ifelse(airlineFiltered$ArrDelay > 15, "High",
                           ifelse(airlineFiltered$ArrDelay < 5,  "Low",
                                                                 "Medium"))

# Machine learning algorithms use objects from class bigr.matrix as input.
# A bigr.matrix object are numeric data sets on HDFS. Use the bigr.transform
# function to recode non-numeric columns.
> airlineMatrix <- bigr.transform(airlineFiltered,
                        outData="/user/airlinef.sample.matrix",
                        transformPath="/user/airline.sample.transform")

# Display the recoded data. Notice that the “Delay” column was recoded into
# numeric values {1, 2, 3} corresponding to {“Low”, “Medium”, “High”}.
> str(airlineMatrix)
'bigr.matrix': 7 variables:
 $ Month      : scale 2 2 2 2 2 2
 $ DayofMonth: scale 12 16 18 19 21 24
 $ DayOfWeek : scale 4 1 3 4 6 2
 $ CRSDepTime: scale 635 2120 700 1145 935 1120
 $ Distance  : scale 1005 2248 301 1074 209 2465
```

```
 $ ArrDelay  : scale 5 -10 -3 7 1 -8
 $ Delay     : nominal 2 1 1 2 1 1
```

Calculate descriptive statistics

```
# Perform the following descriptive statistics with the data: boundaries, mean,
# variance, standard deviation, standard error in mean, coefficient of variation,
# skewness, kurtosis, standard error in skewness, standard error in kurtosis,
# median, interquartile mean, number of categories, and number of modes.
> bigr.univariateStats(airlineMatrix)
                Month   DayofMonth    DayOfWeek    CRSDepTime     Distance      ArrDelay Delay
Min.      1.000000000  1.000000000  1.000000000  0.000000e+00 0.000000e+00 -6.800000e+01    NA
Max.     12.000000000 31.000000000  7.000000000  2.359000e+03 4.983000e+03  1.016000e+03    NA
Range    11.000000000 30.000000000  6.000000000  2.359000e+03 4.983000e+03  1.084000e+03    NA
Mean      6.556821182 15.682723814  3.947690038  1.334173e+03 7.009630e+02  6.957357e+00    NA
Var      11.855746085 77.360205146  3.948726893  2.279813e+05 3.040063e+05  9.433493e+02    NA
SD        3.443217403  8.795465033  1.987140381  4.774738e+02 5.513676e+02  3.071399e+01    NA
SEM       0.009594523  0.024508557  0.005537165  1.330480e+00 1.536385e+00  8.558452e-02    NA
CoV       0.525135170  0.560837845  0.503367884  3.578800e-01 7.865859e-01  4.414606e+00    NA
Skewness -0.020943817  0.014407088  0.045535081 -3.318663e-02 1.651622e+00  5.628932e+00    NA
Kurtosis -1.204467468 -1.189566184 -1.221353204 -8.037885e-01 3.448230e+00  7.194442e+01    NA
SES       0.006825422  0.006825422  0.006825422  6.825422e-03 6.825422e-03  6.825422e-03    NA
SEK       0.013650738  0.013650738  0.013650738  1.365074e-02 1.365074e-02  1.365074e-02    NA
Median    7.000000000 16.000000000  4.000000000  1.326000e+03 5.450000e+02  0.000000e+00    NA
IQM       6.575758987 15.646758289  3.903672645  1.330305e+03 5.625964e+02  4.798043e-01    NA
# cat.             NA           NA           NA            NA           NA            NA     3
# modes            NA           NA           NA            NA           NA            NA     1

# Compute the Pearson's correlation between the predictors and the response
# variable. For example, ArrDelay:
> bigr.bivariateStats(airlineMatrix, cols1=c("Month", "DayofMonth", "DayOfWeek",
"CRSDepTime", "Distance"), cols2=c("ArrDelay"))
           X        Y          Cor
1      Month ArrDelay -0.008673369
2 DayofMonth ArrDelay  0.005967325
3  DayOfWeek ArrDelay  0.004634345
4 CRSDepTime ArrDelay  0.105184454
5   Distance ArrDelay  0.009198216
```

Create training and testing sets

```
# Split the data into 70% for training and 30% for testing.
> samples <- bigr.sample(airlineMatrix, perc=c(0.7, 0.3))
> train <- samples[[1]]
> test <- samples[[2]]

# Check that the training and testing sets are split correctly.
> nrow(train) / nrow(airlineMatrix)
[1] 0.6994487
> nrow(test) / nrow(airlineMatrix)
[1] 0.3005513
```

Create a *linear regression* model

```
# Build a linear regression model on the training set for the arrival delay using
# all other columns in the training set as predictors. The model will be stored
# on the specified HDFS location.
> lm <- bigr.lm(ArrDelay ~ ., data=train, directory="/user/lm.airline")
```

```
# Display the coefficients of the Linear Regression model for each predictor
# column.
> coef(lm)
 (Intercept)      Month DayofMonth DayOfWeek   CRSDepTime      Distance     Delay
          NA -0.895564 -0.2924151 -1.442185 -0.006449712 -0.004677833 23.05393

# Evaluate the model against the testing set and store the output on HDFS/GPFS.
> pred <- predict(lm, test, "/user/lm.airline.preds")

# Display the results of the evaluation including predictions and statistics that
# assess the quality of the model.
> pred
$preds
        preds
1   26.386420
2   -5.413295
3   49.194540
4   -1.665189
5   32.698154
6   26.066165
7   46.975960
8   -3.830847
9   -7.096804
10 -11.654695
...  showing first 10 rows only.

$statistics
                Name Y-column Scaled         Value
1          LOGLHOOD_Z       NA  FALSE           NaN
2     LOGLHOOD_Z_PVAL       NA  FALSE           NaN
3          PEARSON_X2       NA  FALSE  2.159397e+07
4    PEARSON_X2_BY_DF       NA  FALSE  5.581278e+02
5     PEARSON_X2_PVAL       NA  FALSE  0.000000e+00
6         DEVIANCE_G2       NA  FALSE  2.159397e+07
7   DEVIANCE_G2_BY_DF       NA  FALSE  5.581278e+02
8    DEVIANCE_G2_PVAL       NA  FALSE  0.000000e+00
9          LOGLHOOD_Z       NA   TRUE           NaN
10    LOGLHOOD_Z_PVAL       NA   TRUE           NaN
11         PEARSON_X2       NA   TRUE  2.159397e+07
12   PEARSON_X2_BY_DF       NA   TRUE  5.581278e+02
13    PEARSON_X2_PVAL       NA   TRUE  0.000000e+00
14        DEVIANCE_G2       NA   TRUE  2.159397e+07
15  DEVIANCE_G2_BY_DF       NA   TRUE  5.581278e+02
16   DEVIANCE_G2_PVAL       NA   TRUE  0.000000e+00
17          AVG_TOT_Y        1     NA  7.163350e+00
18        STDEV_TOT_Y        1     NA  3.088156e+01
19          AVG_RES_Y        1     NA -1.211733e+00
20        STDEV_RES_Y        1     NA  2.359393e+01
21     PRED_STDEV_RES        1   TRUE  1.000000e+00
22           PLAIN_R2        1     NA  4.148338e-01
23        ADJUSTED_R2        1     NA  4.147582e-01
24    PLAIN_R2_NOBIAS        1     NA  4.163735e-01
25 ADJUSTED_R2_NOBIAS        1     NA  4.162830e-01
```

Create a support vector machine classifier

```
# Build an SVM model.
> svmModel <- bigr.svm(formula=Delay ~ ., data=train,
directory="/user/svm.airline")

# Display the coefficients of the model.
> coef(svmModel)
                     Low        Medium         High
Month       9.545485e-04 -3.040180e-06 -0.0013980276
DayofMonth  1.722937e-03 -7.107980e-06 -0.0028906842
DayOfWeek   4.674919e-04 -1.848519e-06 -0.0007403553
CRSDepTime  2.010074e-04 -3.315430e-04 -0.0004699866
Distance    2.923583e-05 -1.521188e-04 -0.0001873042
ArrDelay   -3.635988e-02  9.555523e-07  0.0355064272

# Evaluate the model against the testing set and store the output on HDFS/GPFS.
> predSVM <- predict(svmModel, test, "/user/svm.preds.airline", returnScores=T)

# Display the results of the evaluation including overall accuracy,
# confusion matrix, and the scores for each example and class.
> predSVM
$accuracy
[1] 79.1826

$ctable
         Low Medium High
Low    23826   6615  558
Medium     0      0    0
High       4    881 6824

$scores
            Low     Medium       High
1  -0.007668927 -0.3597726 -0.3548827
2   0.622849678 -0.6164869 -1.1750341
3  -1.626988731 -0.5585051  0.9624977
4   1.398264835 -0.5139710 -1.7943678
5  -4.645066794 -0.9445551  3.3536020
6  -0.239887320 -0.3716973 -0.1592355
7  -3.436321972 -0.6938454  2.5247219
8   0.667641212 -0.4536836 -1.0925997
9   0.975276859 -0.9322711 -1.7988017
10  0.994386065 -0.6001759 -1.5456743
...  showing first 10 rows only.
```