R4ML: Data Preparation and Dimensionality reduction

(Part 2)

[Alok Singh](#)

Contents

1. Introduction

In our first notebook we received an introduction to R4ML and conducted some exploratory data analysis with it. In this section, we will go over the typical operations a data scientist performs while cleaning and pre-processing the input data. Now, we will look into dimensionality reduction, first starting out with the boiler plate code from Part I for the purposes of loading etc. The following code is copy pasted from part I:

In [1]:

```
library(R4ML)
library(SparkR)

r4ml.session()
# read the airline dataset
airt <- airline
# testing, we just use the small dataset
airt <- airt[airt$Year >= "2007",]
airt <- airline
# testing, we just use the small dataset
airt <- airt[airt$Year >= "2007",]
air_hf <- as.r4ml.frame(airt)

airs <- r4ml.sample(air_hf, 0.1)[[1]]
```
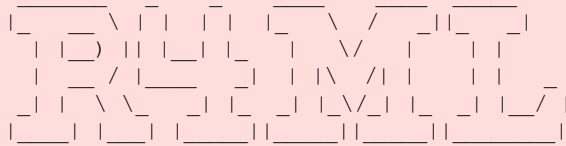
```
Warning message:
"WARN[R4ML]: Reloading SparkR"Loading required namespace: SparkR
```

2. Data Preparation and Pre-processing?

2.1. Why data preparation and pre-processing?

- Real life data has many missing values.
- Outliers are present, which will affect the model predictive powers.
- Data may be corrupted.
- Certain features (columns) of data are expressed as strings but most of the ML algorithms, works in the double and integers.
- Without pre-processing, the analytics power of the models is reduced significantly.

2.2. Typical time spent by a Data Scientist.

Here is the detail. And as we can see that there is the need for unified API for doing the data preparation to simply the journey of data scientist.
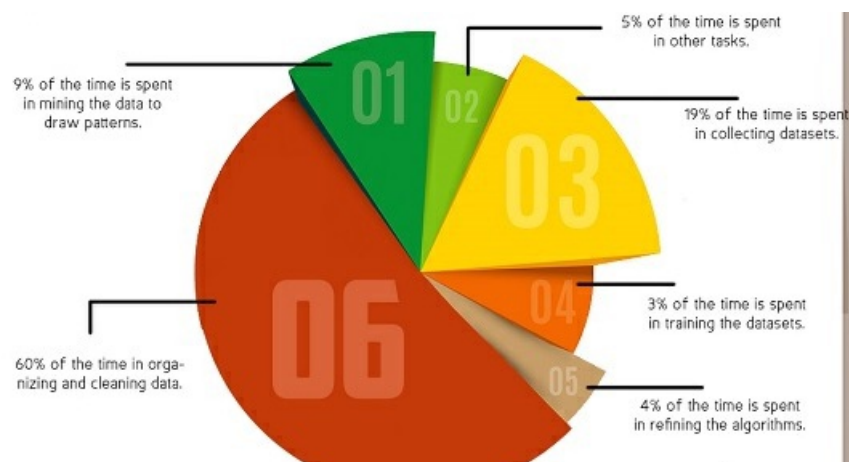
2.3. How does R4ML Address Data Preparation and Pre-Processing?

- R4ML provides a unified API to run the various data preprecessing tasks from one entry point.
- The user can also run the individual data preprocessing via the r4ml.frame and r4ml.matrix api.
- Its API is more closer to the model of [R's caret](#) package.

  The function is called **r4ml.ml.preprocess** and it supports all major pre-processing tasks, feature transformation routines e.g R4ML supports all the major pre-processing task

Method	R4ML options	Description
NA Removal	imputationMethod, imputationValues, omit.na	this options let one remove the missing data or user can substitute with the constant or with the mean (if it is numeric)
Binning	binningAttrs, numBins	One of the typical use case of binning is say we have the height of the people in feet and inches and if we only care about three top level category i.e short , medium and tall, user can use this option
Scaling and Centering	scalingAttrs	Most of the algo's prediction power or the result become better if it is normalized i.e substract the mean and then divide it by stddev.
Encoding (Recode)	recodeAttrs	Since most of the machine learning algorithm is the matrix based linear algebra. Many times the categorical columns have inherit order in it like height or shirt_size, it options provide user the ability to encode those string columns into the ordinal categorical numeric value
Encoding (OneHot or DummyCoding)	dummyCodeAttrs	when categorical columns have no inherit orders like people's race or states they lives in, in that case, we would like to do one hot encoding for those.

2.4. An Example of Data Processing.

We already have the airs r4ml.frame and we will first cache it (to improve the performance we need to cache before heavy lifting)

In the next cell, we regroup the various columns into the above methods and in the cell after, that we run the r4ml.ml.preprocess.

In [3]:

```
airs_t_res <- r4ml.ml.preprocess(
    airs,
    dummycodeAttrs = c(),
    binningAttrs = c(), numBins=4,
    missingAttrs = airs_ia,
    imputationMethod = airs_im,
    imputationValues = airs_iv,
    omit.na=na.cols, # we remove all the na, this is just the dummy
    recodeAttrs=cat_feat # recode all the categorical features
 )


 # the transformed data frame
 airs_t <- airs_t_res$data
 # the relevant metadata list
 airs_t_mdb <- airs_t_res$metadata

 # cache the transformed data
 dummy <- cache(airs_t)
 showDF(airs_t, n = 2)
```

```
INFO[r4ml.ml.preprocess]: running proxy.omit.na
INFO[r4ml.ml.preprocess]: running proxy.impute
INFO[r4ml.ml.preprocess]: running proxy.normalize
INFO[r4ml.ml.preprocess]: running proxy.binning
INFO[r4ml.ml.preprocess]: running proxy.recode
INFO[r4ml.ml.preprocess]: running proxy.onehot
```

```
+----+-----+----------+---------+-------+----------+-------+---------+-------------+---------+-------+
----------------+-------------+-------+-------+--------+------+----+-------+------+-------+--------
-+--------------+-------+----------+-----------+--------+------------+----------------+
|Year|Month|DayofMonth|DayOfWeek|DepTime|CRSDepTime|ArrTime|CRSArrTime|UniqueCarrier|FlightNum|TailNum|
```

```
ActualElapsedTime|CRSElapsedTime|AirTime|ArrDelay|DepDelay|Origin|Dest|Distance|TaxiIn|TaxiOut|Cancelle
d|CancellationCode|Diverted|CarrierDelay|WeatherDelay|NASDelay|SecurityDelay|LateAircraftDelay|
+----+-----+----------+--------+-------+----------+-------+----------+------------+--------+-------+
----------------+--------------+-------+--------+--------+------+----+--------+------+-------+--------
-+--------------+--------+------------+-----------+--------+------------+----------------+
|2007|    9|        24|       1|  1257|      1300|  1319|      1330|          12|     683|    748|
82|           90|     67|     -11|     -3|    51| 150|      435|      1|     14|      0|
 4|           1|        0|           0|          0|       0|
|2007|    9|        10|       1|    32|        40|   542|       607|          13|     672|    600|
190|          207|    171|     -25|     -8|   157| 121|     1589|      7|     12|      0|
 4|           1|        0|           0|          0|       0|
+----+-----+----------+--------+-------+----------+-------+----------+------------+--------+-------+
----------------+--------------+-------+--------+--------+------+----+--------+------+-------+--------
-+--------------+--------+------------+-----------+--------+------------+----------------+
only showing top 2 rows
```

2.5. Next steps for Advanced Users.

All the values are numeric now and we are ready for the next steps of the pipeline. However, we want to end this topic with a note to the user that in using the custom DML (explained later), the advanced user can write the advanced data pre-processing step independently.

Other data pre-processing steps, that are typically done:

Input data transformation (Box Cox) We saw in the previous section that the input data when transformed by log gave us a better feature. That was the special case of Box-Cox transformation. Alternatively, statistical methods can be used to empirically identify an appropriate transformation. Box and Cox proposes a family of transformations that are indexed by a parameter lambda, (this feature will be available in future.) Practically, one can calculate the kurtosis and skewness for various lambda or one can do the significance testing to check which lambda gives the better result.

Outlier Detection

Method	Description
Box Plot (univariate Stats)	This way one can visually detect outliers for each of dimensions. But again it is not the best as it is not explaning all the details.
mahalanobis distance (for multivariate stats)	This is like calculating the number of units of variance in each dimensions after rotating the axis in the direction of maximum variance.
use classifier like logistic regression	use the mahalanobis distance, to calculate the threshold i.e 3 sigma distance and then anything more than level it as Y = 1 and then run logistic regression to find out what other variables are outliers.
Spacial sign transformation	If a model is considered to be sensitive to outliers, one data transformation that can minimize the problem is the spatial sign. This procedure projects the predictor values onto a multidimensional sphere. This has the effect of making all the samples the same distance from the center of the sphere. Mathematically, each sample is divided by its squared norm.

2.6. Summary of Data Preparation

In this subsection, we saw how R4ML, helps simplify the 60% of the time spent by data scientists on data preparation by providing a unified and expandable API for pre-processing. We also saw a small example of R4ML in action for this and we gave a few advanced technique for power users.

3. Dimensionality Reduction

3.1. What is Dimensionality Reduction

Dimensionality reduction is choosing a basis or mathematical representation within which you can describe most but not all of the variance within your data, thereby retaining the relevant information, while reducing the amount of information necessary to represent it. There are a variety of techniques for doing this including but not limited to PCA, ICA, and Matrix Feature Factorization. These will take existing data and reduce it to the most discriminative components. All of these allow you to represent most of the information in your dataset with fewer, more discriminative features.

## 3.2. Why is Dimensionality Reduction Useful?

- In terms of performance, having data of high dimensionality is problematic because:
    - It can mean high computational cost to perform learning and inference and
    - It often leads to [overfitting](#) when learning a model, which means that the model will perform well on the training data but poorly on test data.

  Dimensionality reduction addresses both of these problems, while (hopefully) preserving most of the relevant information in the data needed to learn accurate, predictive models.

  Also note that, in general visualization of lower dimension data and it's interpretation are more straightforward and it coule be used for geting insights into the data.

3.3. R4ML for Dimensionality Reduction of Big data

R4ML provides the inbuilt r4ml.pca for dimensionality reduction for big dataset. This can allow one to convert the higher dimensional data into the lower dimensions . For example see the following illustration:
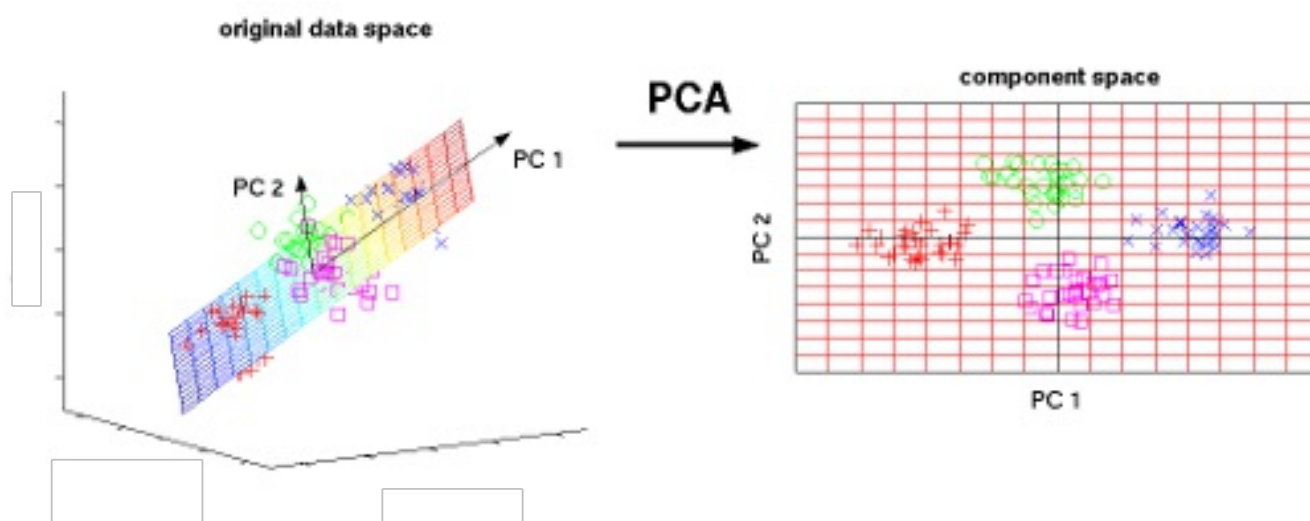
PCA transformation

Principal component analysis (PCA) rotates the original data space such that the axes of the new coordinate system point into the directions of highest variance of the data. The axes or new variables are termed principal components (PCs) and are ordered by variance: the first component, PC 1, represents the direction of the highest variance of the data. The direction of the second component, PC 2, represents the highest of the remaining variance orthogonal to the first component. This can be naturally extended to obtain the required number of components which together span a component space covering the desired amount of variance. Since components describe specific directions in the data space, each component depends on certain fraction of each of the original variables: each component is a linear combination of all original variables.

Dimensionality reduction

Low variance can often be assumed to represent undesired background noise. The dimensionality of the data can therefore be reduced, without loss of relevant information, by extracting a lower dimensional component space covering the highest variance. Using a lower number of principal components instead of the high-dimensional original data is a common pre-processing step that often improves results of subsequent analyses such as classification. For visualization, the first and second component can be plotted against each other to obtain a two-dimensional representation of the data that captures most of the variance (assumed to be most of the relevant information), useful to analyze and interpret the structure of a data set.

An Example:

We will use r4ml.pca and will experiment to see if there are any dependencies or co-releation between the scale (regression or continous) variables and reduce dimensions.

In [5]:

```
# since from the exploratory data analysis, we knew that certain
#  variables can be remove for the analysis, we will perform the pca
#  on the smaller set of features
airs_t3_tmp <- select(airs_t, names(rairs2)) # recall rairs2 from before
airs t3 <- as.r4ml.matrix(airs t3 tmp)
```

```
airs_t3 <- as.r4ml.matrix(airs_t3_tmp)
# do the PCA analysis with 12 components
airs_t3.pca <- r4ml.pca(airs_t3, center=T, scale=T, projData=F, k=12)
  # the eigen values for each of the components which is the square of
# stddev and is equivalent to the variance
airs_t3.pca@eigen.values
```

EigenValues1
4.05090927
3.49707946
2.16091551
1.08756389
0.97688625
0.94271383
0.64066263
0.39230312
0.13086497
0.05333747
0.03242792
0.02919244

In [6]:

```
# the corresponding eigenvectors are
airs_t3.pca@eigen.vectors
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
DepTime	-0.16159451	-0.46622520	0.11153934	-0.03769436	0.003752706	0.05391904	-0.086367203	-0.44510939
CRSDepTime	-0.15308935	-0.45172887	0.18167525	-0.03887363	-0.007003243	0.03623870	-0.063891340	-0.50254863
ArrTime	-0.17118516	-0.41602780	0.16609767	0.06190138	0.031943447	-0.03655623	0.196375995	0.59176698
CRSArrTime	-0.17449044	-0.43803111	0.18054977	0.01872730	0.028542667	-0.01978737	0.048320360	0.37242353
ActualElapsedTime	-0.46453724	0.17845112	0.01245669	0.01948689	-0.034406018	-0.06792019	0.037098041	-0.02203233
CRSElapsedTime	-0.45870036	0.18821635	0.05095632	-0.04714691	0.023687408	0.05345970	-0.006762444	-0.00933559
AirTime	-0.45725221	0.19046576	0.04727776	-0.09107592	0.041914821	0.03826202	-0.012018426	0.00312365
ArrDelay	-0.12883119	-0.17592635	-0.58567253	0.04449364	-0.046989575	-0.10024814	-0.276360732	0.08431413
DepDelay	-0.11410843	-0.18272551	-0.56558220	-0.12856096	0.061105838	0.06545151	-0.382759739	0.10075848
Distance	-0.45238927	0.18948966	0.05617556	-0.06905888	0.035905296	0.08897432	-0.029165559	-0.00475125
TaxiIn	-0.06039276	0.01896610	-0.04289010	0.56459120	0.645353960	-0.48399789	0.016617567	-0.11295393
TaxiOut				0.43583696			0.274036153	

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
	0.12520709	0.03299464	0.17457240		0.716535418	0.33319509		0.10046842
Cancelled	0.01637714	-0.01104306	0.01977276	-0.64879625	-0.006423997	-0.75357093	0.080558661	-0.02019410
CarrierDelay	-0.03940274	-0.09089537	-0.43660306	-0.17046321	0.239630850	0.22298977	0.800250723	-0.13895572

How many eigenvectors do we need?

Now the question is how many eigenvalues (and corresponding eigenvectors) we need, so that we can explain most of the variance (say 90%) of the input data. We will try to find it out in the following code:

In [7]:

```
sorted_evals <- sort(airs_t3.pca@eigen.values$EigenValues1, decreasing = T)
csum_sorted_evals <- cumsum(sorted_evals)
# find the cut-off i.e the number of principal component which capture
# 90% of variance
evals_ratio <- csum_sorted_evals/max(csum_sorted_evals)
evals_ratio
```

> 0.289457001252769 0.539340191714469 0.693748025120218 0.771459709447447 0.841262942723305
> 0.908624391906449 0.954402826487755 0.982434776123405 0.991785709019704 0.995596928175847
> 0.997914059217378 1

In [8]:

```
pca.pc.count <- which(evals_ratio > 0.9)[1]
pca.pc.count
```

> 6

Analytically we can see that we need the first six principal components. Let's also verify it intuitively or graphically. Here we will plot the PCA variance and see the overall area spanned.

In [14]:

```
library(ggplot2)

pca.plot.df <- data.frame(
    index=1:length(sorted_evals),
    PrincipalComponent=sprintf("PC-%02d", 1:length(sorted_evals)),
    Variances = sorted_evals)

  pca.g1 <- ggplot(data=pca.plot.df,
        aes(x=PrincipalComponent, y=Variances, group=1, colour=Variances))
  pca.g1 <- pca.g1 + geom_point() + geom_line()
  # highlight the area containing the 90% variances

  #subset region and plot
  pca.g_data <- ggplot_build(pca.g1)$data[[1]]
  #plot the next shaded graph
  pca.g2 <- pca.g1 + geom_area(data=subset(pca.g_data, x<=pca.pc.count),
  aes(x=x, y=y), fill="red4", inherit.aes=F)
  pca.g2 <- pca.g2 + geom_point() + geom_line()
  pca.g2
```
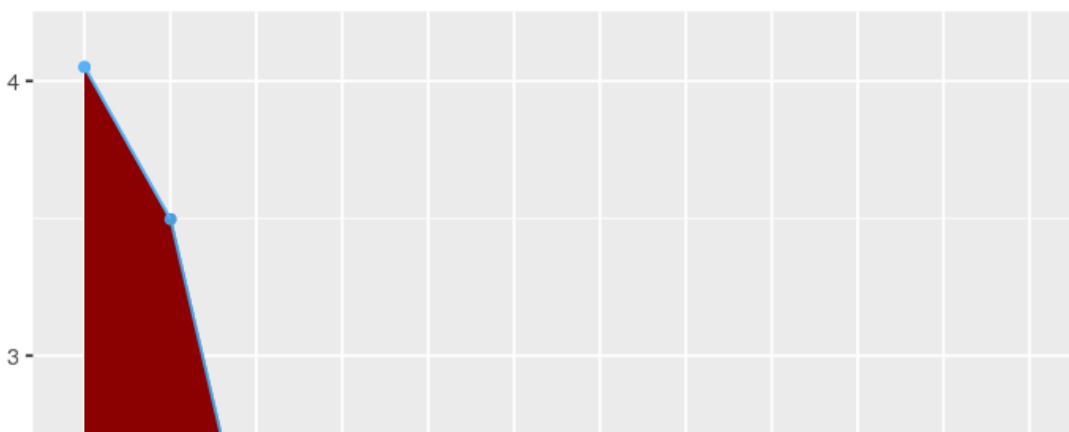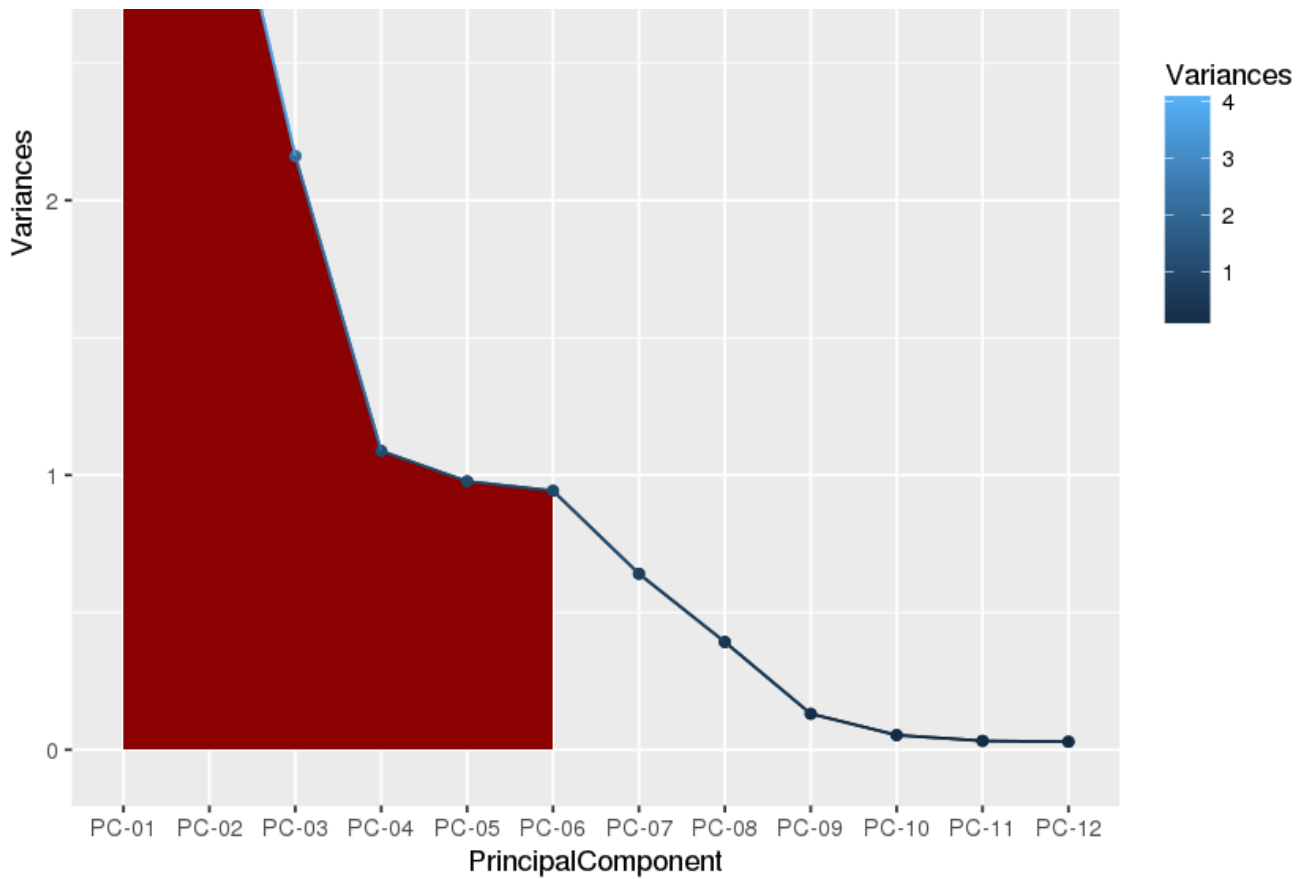
3.4. Summary of Dimensionality Reduction.

In this subsection, we went over why dimensionality reduction is important and how R4ML helps in this task by providing a scalable PCA implementation. We also went through an example to understand it more.

In general, using the custom algorithms, one can implement other dimensional reduction techniques like t-sne and kernel pca also.

4. Summary and next steps ...

This notebook is mostly focus on the data preparation and also on the data reduction using the PCA. In future notebooks, we will look into the actual modelling using the canned algorithm provided by R4ML and and also creating the custom ML models.