Solutions R Replica

Samuele Zolarsi

Data Import and Setup

```
y_test = read.csv("C:/Users/Samuele/Downloads/sample_submission.csv")
y_test = y_test$emission
test = read.csv("C:/Users/Samuele/Downloads/test.csv/test.csv")
x_test = test[,6:75]

train = read.csv("C:/Users/Samuele/Downloads/train.csv/train.csv")
x_train = train[,6:75]
y_train = train[,76]
```

Standardization

We can use “cor = TRUE” as a parameter in princomp() to avoid scaling here

```
#x_train = scale(x_train)
#x_test = scale(x_test)
```

Data Imputation

We can refer to this library for an easy median imputation of missing values.

```
library(missMethods)
x_train = impute_median(x_train)
x_test = impute_median(x_test)
```

Principal Component Analysis

Let us check whether R’s PCA is comparable to the one obtained using scikit-learn. We can notice that the result is analogous to Python with Scikit-Learn by in terms of explaination up to the 12th component.

Notice also cor = True as parameter in princomp

```
x_train_pca = princomp(x_train, cor = TRUE)
print("Cumulative explained variance")
```

```
## [1] "Cumulative explained variance"
```

```
cumsum(x_train_pca$sdev^2 / sum(x_train_pca$sdev^2))[1:12]
```

```
##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6    Comp.7    Comp.8
## 0.1406255 0.2421315 0.3212564 0.3777114 0.4306439 0.4771368 0.5182511 0.5591322
##    Comp.9   Comp.10   Comp.11   Comp.12
## 0.5980960 0.6306413 0.6598324 0.6863280
```

Let us overwrite x_train_pca in order to retain only the first 12 components.

```
x_train_pca = x_train_pca$scores[,1:12]
```

Repeat the same here for test set.

```
x_test_pca = princomp(x_test, cor = TRUE)
x_test_pca = x_test_pca$scores[,1:12]
```

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(22)
rf_model = randomForest(y_train~., data = x_train_pca, ntree = 100,maxnodes = 5)
y_pred = predict(rf_model, newdata = x_test_pca)
```

From the following results we can notice that by using the same naive Python's tuning and provided the difference in the default parameters and algorithm construction, R seems better in this case.

```
library("MLmetrics")
```

```
##
## Attaching package: 'MLmetrics'
```

```
## The following object is masked from 'package:base':
##
##     Recall
```

```
cat("MSE: ", mean((y_test - y_pred)^2));
```

```
## MSE:  12.04837
```

```
cat("MAPE:", round(MAPE(y_pred, y_test)*100,2), "%")
```

```
## MAPE: 3.4 %
```