Weather Forecast with Linear Models

Mark Sidorovskiy

18 December 2016

Abstract

In this project I apply machine learning technics to the weather forecast. The idea is to predict the temperature using the conditions in previous days and climate average. As a benchmark I parsed the archive of forecasts based on meteorological model. Simple linear model is worse than the official predictions. However, the model which uses both the previous records and official forecast was closer to the truth, than only official predictions.

1. Introduction

In 2015 Yandex, one of the largest European internet companies and the leading search provider in Russia, has launched a service offering hyperlocal weather information based on its proprietary weather forecasting technology, Meteum. Powered by machine learning, it gives accurate forecasts for areas as local as specific parts of a city or even individual buildings.

To calculate the weather forecast, Yandex's new technology uses data from meteorological stations, as well as from other sources indirectly indicating the situation – about 9 terabytes of information every day. Traditional meteorology models are used to process the initial data, and then the intermediate results are processed using Yandex's machine learning technology MatrixNet.

In this project I am about to reproduce some of their results, using the data from one meteorology station in Moscow (VDNH) from 03.09.2008 to 17.12.2016 and the simplest machine learning method, namely linear regression.

The rest of the paper is organized as follows. Section 2 describes my data sources and the way in which I construct the data sets for training. In Section 3, I train my models. In Section 4 I examines the results. Finally, Section 5 concludes.

2. Data and Sample Construction

Request some packages

```
library(rvest)
library(stringr)
library(ggplot2)
library(dplyr)
library(zoo)
library(data.table)
```

The weather records in the last 5 years can be found in csv format at http://rp5.ru/archive.php?wmo_id=27612&lang=ru

```
real <- read.csv("moscow.csv", header = T, sep = ";", comment.char = '#', row.names = NULL)

#names were read a litle bit wrong
names(real) <- names(real[-1])

head(real[,1:6])
```

```
##          .      T    Po     P   Pa  U
## 1 17.12.2016 21:00 -3.8 752.2 767.4 -0.9 95
## 2 17.12.2016 18:00 -4.6 753.1 768.3 -0.1 92
## 3 17.12.2016 15:00 -5.6 753.2 768.5 -0.8 89
## 4 17.12.2016 12:00 -6.7 754.0 769.4 -0.6 88
## 5 17.12.2016 09:00 -8.0 754.6 770.1  0.3 89
## 6 17.12.2016 06:00 -8.6 754.3 769.8 -0.7 90
```

The archive of weather forecasts can not be found that easy. I had to parse it from http://meteoinfo.ru/archive-forecast/russia/moscow, the code below runs for about 1.5 hours.

```
# Store web url
start <- as.Date("2008-09-04")
finish <- as.Date("2016-12-17")
day <- seq(from = start, to = finish, by = "day")
day <- gsub("-", "/", as.character(day))
dat <- sub("^", "http://meteoinfo.ru/archive-forecast/russia/moscow-area/moscow/", day)

forecast <- sapply(dat, function(x) {
                            weather <- html(x)
                            rating <- weather %>%
                              html_nodes("tr:nth-child(4) .pogodacell") %>%
                              html_text()
                          })
```

```
head(forecast, 3)
```

```
## $`http://meteoinfo.ru/archive-forecast/russia/moscow-area/moscow/2008/09/04`
## [1] "15  /  26" "15  /  24" "15  /  24" "17  /  25" "15  /  23" "15  /  19"
## [7] "11  /  19"
##
## $`http://meteoinfo.ru/archive-forecast/russia/moscow-area/moscow/2008/09/05`
## [1] "16  /  25" "16  /  25" "16  /  25" "14  /  23" "12  /  16" "10  /  18"
## [7] "8  /  20"
##
## $`http://meteoinfo.ru/archive-forecast/russia/moscow-area/moscow/2008/09/06`
## [1] "16  /  27" "16  /  26" "16  /  22" "13  /  18" "8  /  14"  "5  /  12"
## [7] "5  /  9"
```

The data are located in lists, convert in into the data frame.

```
m <- lapply(forecast, function(x) strsplit(x, "\\s\\s.\\s\\s"))
n <- lapply(m, unlist)
#remove empty lists
n <- n[!sapply(n, is.null)]

official_forecast <- t(as.data.frame(n))
row.names(official_forecast) <- gsub("http...meteoinfo.ru.archive.forecast.russia.moscow.area.moscow.",
                                     "",row.names(official_forecast))
row.names(official_forecast) <- gsub("[[:punct:]]", "-", row.names(official_forecast))
head(official_forecast)
```

```
##            [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
## 2008-09-04 "15" "26" "15" "24" "15" "24" "17" "25" "15" "23"  "15"  "19"
## 2008-09-05 "16" "25" "16" "25" "16" "25" "14" "23" "12" "16"  "10"  "18"
## 2008-09-06 "16" "27" "16" "26" "16" "22" "13" "18" "8"  "14"  "5"   "12"
## 2008-09-07 "18" "25" "16" "24" "12" "15" "6"  "9"  "7"  "10"  "6"   "9"
## 2008-09-08 "16" "23" "14" "14" "8"  "10" "5"  "8"  "5"  "8"   "4"   "10"
## 2008-09-09 "14" "14" "8"  "10" "4"  "7"  "5"  "8"  "5"  "8"   "6"   "8"
##            [,13] [,14]
## 2008-09-04 "11"  "19"
## 2008-09-05 "8"   "20"
## 2008-09-06 "5"   "9"
## 2008-09-07 "4"   "12"
## 2008-09-08 "4"   "10"
## 2008-09-09 "5"   "12"
```

Get the climate average from http://meteoinfo.ru/clim-moscow-daily.

```
climate_average <- read.csv("climate_average.csv", header = F, sep = ";", row.names = NULL)
head(climate_average)
```

```
##     V1   V2   V3  V4   V5   V6   V7   V8   V9 V10 V11  V12
## 1 -8,2  -10   -5 1,1 10,4 15,6 17,3 18,8 13,5 8,1 1,6 -3,9
## 2 -8,2 -9,8 -4,8 1,4 10,6 15,7 17,3 18,6 13,3 7,9 1,4   -4
## 3 -8,3 -9,6 -4,6 1,7 10,8 15,7 17,4 18,5 13,1 7,7 1,2 -4,2
## 4 -8,4 -9,5 -4,4   2 10,9 15,8 17,4 18,3 12,9 7,5   1 -4,3
## 5 -8,5 -9,3 -4,2 2,3 11,1 15,9 17,5 18,1 12,7 7,5 0,8 -4,4
## 6 -8,6 -9,1   -4 2,7 11,3 15,9 17,5   18 12,6 7,1 0,6 -4,6
```

Now I construct the data set, which I will use for the training.

First, convert climate average to vector:

```
d <- as.vector(as.matrix(climate_average))
d <- d[!is.na(d)]
avg <- as.numeric(sub(",", ".", d, fixed = TRUE))
head(avg, 10)
```

```
##  [1] -8.2 -8.2 -8.3 -8.4 -8.5 -8.6 -8.6 -8.6 -8.7 -8.8
```

Then, I split the official forecast on day and night parts. For the further research I will use only the day part.

```
off_night <- official_forecast[,c(1,3,5,7,9,11,13)]
off_day <- official_forecast[,c(2,4,6,8,10,12,14)]
```

From the weather records I take only temperature, pressure, humidity and wind speed. The observations are given for every 4 hours. I match the observations at 15.00 and at 03.00 with day and night official forecasts respectively.

```
train <- real[,c(1,2,3,6,8)]
time_train <- data.frame(matrix(unlist(strsplit(train$    .    , " ")), ncol=2, byrow=T))
train <- cbind(train, time_train)
train <- train[,-1]
day_train <- filter(train, X2 == "15:00")
night_train <- filter(train, X2 == "03:00")
```

This period consists of 3027 observations and in every group there are some missing rows. Fill the gaps with NA.

```
#off_day
day <- as.Date(day, format = "%Y/%m/%d")

dates_missing_off_day <- day[!(day %in% off_day$date)]
missing_data_off_day <- data.frame(date = dates_missing_off_day, V1 = NA,
                                   V2 = NA, V3 = NA, V4 = NA, V5 = NA, V6 = NA, V7 = NA)
off_day <- rbind(missing_data_off_day, off_day)
off_day <- off_day[order(off_day$date),]

#day_train

dates_missing_day_train <- day[!(day %in% day_train$date)]
missing_data_day_train <- data.frame(date = dates_missing_day_train, `T` = NA,
                                     Po = NA, U = NA, Ff = NA, X1 = NA, X2 = NA)
day_train <- rbind(missing_data_day_train, day_train)
day_train <- day_train[order(day_train$date),]
```

Finally, I construct the dataset from 7 previous temperature observations, 7 pressure observations, 3 humidity observations and 3 wind speed observations.

```
day_train_2 <- as.data.table(day_train[,c(1:5)])

for (i in 1:7) {
  day_train_2[, paste('T_day', i, sep = '_') := shift(day_train_2$`T`, i)]
}

for (i in 1:7) {
  day_train_2[, paste('Po_day', i, sep = '_') := shift(day_train_2$Po, i)]
}

for (i in 1:3) {
  day_train_2[, paste('U_day', i, sep = '_') := shift(day_train_2$U, i)]
}

for (i in 1:3) {
  day_train_2[, paste('Ff_day', i, sep = '_') := shift(day_train_2$Ff, i)]
}

day_train_2 <- day_train_2[-c(1:10),]
day_train_2 <- day_train_2[,-c(3:5)]
```

Then, add the climate average

```
avg_2 <- c(avg[257:366], avg[-60], avg[-60], avg[-60], avg, avg[-60], avg[-60], avg[-60], avg[1:351])
day_train_2 <- cbind(day_train_2, avg_2)
head(day_train_2)
```

```
##          date    T T_day_1 T_day_2 T_day_3 T_day_4 T_day_5 T_day_6 T_day_7
## 1: 2008-09-14  9.4    13.1     7.7     6.9     9.1    15.1    27.2    27.9
## 2: 2008-09-15  7.9     9.4    13.1     7.7     6.9     9.1    15.1    27.2
## 3: 2008-09-16  6.6     7.9     9.4    13.1     7.7     6.9     9.1    15.1
## 4: 2008-09-17  8.2     6.6     7.9     9.4    13.1     7.7     6.9     9.1
## 5: 2008-09-18  8.9     8.2     6.6     7.9     9.4    13.1     7.7     6.9
## 6: 2008-09-19 11.7     8.9     8.2     6.6     7.9     9.4    13.1     7.7
##    Po_day_1 Po_day_2 Po_day_3 Po_day_4 Po_day_5 Po_day_6 Po_day_7 U_day_1
## 1:    746.6    746.5    747.8    748.0    746.3    744.8    748.1      69
## 2:    749.0    746.6    746.5    747.8    748.0    746.3    744.8      64
## 3:    751.6    749.0    746.6    746.5    747.8    748.0    746.3      85
## 4:    753.9    751.6    749.0    746.6    746.5    747.8    748.0      68
## 5:    756.3    753.9    751.6    749.0    746.6    746.5    747.8      68
## 6:    757.4    756.3    753.9    751.6    749.0    746.6    746.5      70
##    U_day_2 U_day_3 Ff_day_1 Ff_day_2 Ff_day_3 avg_2
## 1:      94      90        1        2        2  11.2
## 2:      69      94        2        1        2  11.1
## 3:      64      69        3        2        1  10.9
## 4:      85      64        1        3        2  10.8
## 5:      68      85        1        1        3  10.6
## 6:      68      68        0        1        1  10.4
```

Finally, I fill NA in official forecast with the forecast in previous available day.

```
off_day <- as.data.frame(apply(off_day, 2, na.locf))
off_day[,-1] <- apply(off_day[,-1], 2, as.numeric)
head(off_day)
```

```
##                  date V1 V2 V3 V4 V5 V6 V7
## 2008-09-04 2008-09-04 26 24 24 25 23 19 19
## 2008-09-05 2008-09-05 25 25 25 23 16 18 20
## 2008-09-06 2008-09-06 27 26 22 18 14 12  9
## 2008-09-07 2008-09-07 25 24 15  9 10  9 12
## 2008-09-08 2008-09-08 23 14 10  8  8 10 10
## 2008-09-09 2008-09-09 14 10  7  8  8  8 12
```

3. Train models

In sum, I trained 6 different models: 2 types of linear model for 1, 2 and 3 days forecasts. I split data on train and test and first make predictions for the next day.

```
train_day <- day_train_2[1:2665,]
test_day <- day_train_2[-c(1:2665),]
test_ans_day <- test_day[,1:2]

train_day <- train_day[,-1]
test_day <- test_day[,-c(1:2)]
```

```r
linear_model <- lm(data = train_day, `T`~.)
summary(linear_model)
```

```
##
## Call:
## lm(formula = T ~ ., data = train_day)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.1868  -1.8335   0.1803   2.0919  13.6452
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -41.698433   9.858676  -4.230 2.42e-05 ***
## T_day_1       0.877616   0.024578  35.708  < 2e-16 ***
## T_day_2      -0.065990   0.032750  -2.015 0.044014 *
## T_day_3      -0.038179   0.030841  -1.238 0.215851
## T_day_4       0.020700   0.025934   0.798 0.424838
## T_day_5       0.019279   0.025607   0.753 0.451594
## T_day_6      -0.016127   0.025411  -0.635 0.525720
## T_day_7       0.001514   0.019619   0.077 0.938499
## Po_day_1      0.162122   0.016418   9.875  < 2e-16 ***
## Po_day_2     -0.085860   0.023636  -3.633 0.000286 ***
## Po_day_3     -0.005565   0.023366  -0.238 0.811780
## Po_day_4     -0.006644   0.020614  -0.322 0.747257
## Po_day_5      0.018859   0.020150   0.936 0.349420
## Po_day_6     -0.008030   0.019716  -0.407 0.683846
## Po_day_7     -0.020542   0.014270  -1.440 0.150117
## U_day_1       0.041320   0.005610   7.366 2.37e-13 ***
## U_day_2      -0.008621   0.006504  -1.325 0.185155
## U_day_3      -0.014168   0.005665  -2.501 0.012447 *
## Ff_day_1     -0.076640   0.075885  -1.010 0.312617
## Ff_day_2      0.153598   0.077987   1.970 0.049001 *
## Ff_day_3      0.250307   0.076434   3.275 0.001072 **
## avg_2         0.246029   0.020520  11.990  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.16 on 2519 degrees of freedom
##   (124 observations deleted due to missingness)
## Multiple R-squared:  0.9281, Adjusted R-squared:  0.9275
## F-statistic:  1548 on 21 and 2519 DF,  p-value: < 2.2e-16
```

```r
predictions_lm <- predict(linear_model, test_day)
```

Obviously, the 1-2 day lag features are statistically significant. The same for the magnitude of the coffiecients in every cathegory - for 1 day lag temperature it's 0.87 and for 2 day lag temperature it's only -0.06. The regression captures the most part of the variance in the dependent variable (R^2 is 0.9281).

For the next model I mix previous dataset with the official predictions to improve them.

```r
day_train_3 <- cbind(day_train_2, off_day[-c(1:9, nrow(off_day)),])
day_train_3 <- day_train_3[,-24]

train_day_expand <- day_train_3[1:2665,]
test_day_expand <- day_train_3[-c(1:2665),]

train_day_expand <- train_day_expand[,-1]
test_day_expand <- test_day_expand[,-c(1:2)]

linear_model_2 <- lm(data = train_day_expand, `T`~.)
summary(linear_model_2)
```

```
##
## Call:
## lm(formula = T ~ ., data = train_day_expand)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5291  -1.0621   0.1964   1.2340  13.7256
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.788e+01  6.049e+00  -4.609 4.25e-06 ***
## T_day_1      1.688e-01  1.857e-02   9.088  < 2e-16 ***
## T_day_2     -7.340e-02  2.002e-02  -3.666 0.000251 ***
## T_day_3     -2.100e-02  1.884e-02  -1.115 0.265061
## T_day_4     -1.310e-04  1.584e-02  -0.008 0.993404
## T_day_5      8.365e-03  1.563e-02   0.535 0.592552
## T_day_6     -6.721e-03  1.553e-02  -0.433 0.665135
## T_day_7     -1.027e-02  1.200e-02  -0.856 0.392144
## Po_day_1     6.849e-02  1.019e-02   6.722 2.22e-11 ***
## Po_day_2    -4.941e-02  1.447e-02  -3.414 0.000651 ***
## Po_day_3     2.681e-02  1.428e-02   1.877 0.060567 .
## Po_day_4    -6.732e-03  1.259e-02  -0.535 0.592866
## Po_day_5     3.016e-03  1.231e-02   0.245 0.806392
## Po_day_6    -2.625e-03  1.206e-02  -0.218 0.827694
## Po_day_7    -3.510e-03  8.729e-03  -0.402 0.687644
## U_day_1     -6.522e-04  3.491e-03  -0.187 0.851842
## U_day_2      2.296e-04  3.977e-03   0.058 0.953962
## U_day_3      4.513e-03  3.491e-03   1.293 0.196215
## Ff_day_1    -3.554e-02  4.661e-02  -0.763 0.445829
## Ff_day_2     1.449e-02  4.775e-02   0.304 0.761486
## Ff_day_3     8.158e-02  4.675e-02   1.745 0.081076 .
## avg_2       -3.055e-02  1.490e-02  -2.050 0.040442 *
## V1           8.971e-01  2.102e-02  42.671  < 2e-16 ***
## V2           5.537e-02  2.437e-02   2.271 0.023201 *
## V3           1.180e-02  2.678e-02   0.441 0.659488
## V4          -2.870e-03  2.795e-02  -0.103 0.918211
## V5          -6.496e-02  2.928e-02  -2.218 0.026631 *
## V6           5.800e-02  3.136e-02   1.850 0.064487 .
## V7          -2.207e-02  2.219e-02  -0.994 0.320112
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.928 on 2512 degrees of freedom
##   (124 observations deleted due to missingness)
## Multiple R-squared:  0.9733, Adjusted R-squared:  0.973
## F-statistic:  3271 on 28 and 2512 DF,  p-value: < 2.2e-16
```

```
predictions_lm_expand <- predict(linear_model_2, test_day_expand)
```

To estimate the efficiency of predictions, I use the mean absolute error (MAE).

```
mae <- function(actual, predicted) {
  error <- actual - predicted
  mean(abs(error), na.rm = T)
}

mae_11 <- mae(actual = test_ans_day$T, predicted = predictions_lm)
mae_21 <- mae(actual = test_ans_day$T, predicted = predictions_lm_expand)

off_test <- off_day[-c(1:9, nrow(off_day)), 2]
mae_31 <- mae(actual = test_ans_day$T, predicted = off_test[-c(1:2665)])

results <- cbind(test_ans_day, predictions_lm, predictions_lm_expand, off_test[-c(1:2665)])
```

I perform the same models for the predictions in 2 and 3 days forward.

```
day_train_4 <- day_train_2
day_train_4[,2] <- c(day_train_2$T[-1], 0)

train_day_2nd_day <- day_train_4[1:2665,]
test_day_2nd_day <- day_train_4[-c(1:2665),]
test_ans_day_2nd_day <- test_day_2nd_day[,1:2]

train_day_2nd_day <- train_day_2nd_day[,-1]
test_day_2nd_day <- test_day_2nd_day[,-c(1:2)]

###

linear_model_2nd_day <- lm(data = train_day_2nd_day, `T`~.)
predictions_lm_2nd_day <- predict(linear_model_2nd_day, test_day_2nd_day)

####

day_train_5 <- cbind(day_train_2, off_day[-c(1:9, nrow(off_day)),])
day_train_5[,2] <- c(day_train_2$T[-1], 0)
day_train_5 <- day_train_5[,-24]

train_day_expand_2nd_day <- day_train_5[1:2665,]
test_day_expand_2nd_day <- day_train_5[-c(1:2665),]

train_day_expand_2nd_day <- train_day_expand_2nd_day[,-1]
test_day_expand_2nd_day <- test_day_expand_2nd_day[,-c(1:2)]

linear_model_2_2nd_day <- lm(data = train_day_expand_2nd_day, `T`~.)
predictions_lm_expand_2nd_day <- predict(linear_model_2_2nd_day, test_day_expand_2nd_day)
```

Then, estimate MAE.

```
mae_12 <- mae(actual = test_ans_day_2nd_day[-352,]$T, predicted = predictions_lm_2nd_day[-352])
mae_22 <- mae(actual = test_ans_day_2nd_day[-352,]$T, predicted = predictions_lm_expand_2nd_day[-352])

off_test_2nd_day <- off_day[-c(1:8, nrow(off_day),  nrow(off_day) - 1), 3]
mae_32 <- mae(actual = test_ans_day_2nd_day[-352,]$T, predicted = off_test_2nd_day[-c(1:2665, 3017)])

results_2nd_day <- cbind(test_ans_day_2nd_day, predictions_lm_2nd_day, predictions_lm_expand_2nd_day, o
```

```
##############
### 3d day forecast

day_train_6 <- day_train_2
day_train_6[,2] <- c(day_train_2$T[-c(1:2)], 0, 0)

train_day_3d_day <- day_train_6[1:2665,]
test_day_3d_day <- day_train_6[-c(1:2665),]
test_ans_day_3d_day <- test_day_3d_day[,1:2]

train_day_3d_day <- train_day_3d_day[,-1]
test_day_3d_day <- test_day_3d_day[,-c(1:2)]

###

linear_model_3d_day <- lm(data = train_day_3d_day, `T`~.)
predictions_lm_3d_day <- predict(linear_model_3d_day, test_day_3d_day)

####

day_train_7 <- cbind(day_train_2, off_day[-c(1:9, nrow(off_day)),])
day_train_7[,2] <- c(day_train_2$T[-c(1:2)],0, 0)
day_train_7 <- day_train_7[,-24]

train_day_expand_3d_day <- day_train_7[1:2665,]
test_day_expand_3d_day <- day_train_7[-c(1:2665),]

train_day_expand_3d_day <- train_day_expand_3d_day[,-1]
test_day_expand_3d_day <- test_day_expand_3d_day[,-c(1:2)]

linear_model_2_3d_day <- lm(data = train_day_expand_3d_day, `T`~.)
predictions_lm_expand_3d_day <- predict(linear_model_2_3d_day, test_day_expand_3d_day)

###

mae_13 <- mae(actual = test_ans_day_3d_day[-c(351,352),]$T, predicted = predictions_lm_3d_day[-c(351,35
mae_23 <- mae(actual = test_ans_day_3d_day[-c(351,352),]$T, predicted = predictions_lm_expand_3d_day[-c

off_test_3d_day <- off_day[-c(1:7, nrow(off_day),  nrow(off_day) - 1, nrow(off_day) - 2), 4]
mae_33 <- mae(actual = test_ans_day_3d_day[-c(351,352),]$T, predicted = off_test_3d_day[-c(1:2665, 3017

results_3d_day <- cbind(test_ans_day_3d_day, predictions_lm_3d_day, predictions_lm_expand_3d_day, off_t
```

4. Results

Put all MAE coefficients in one table

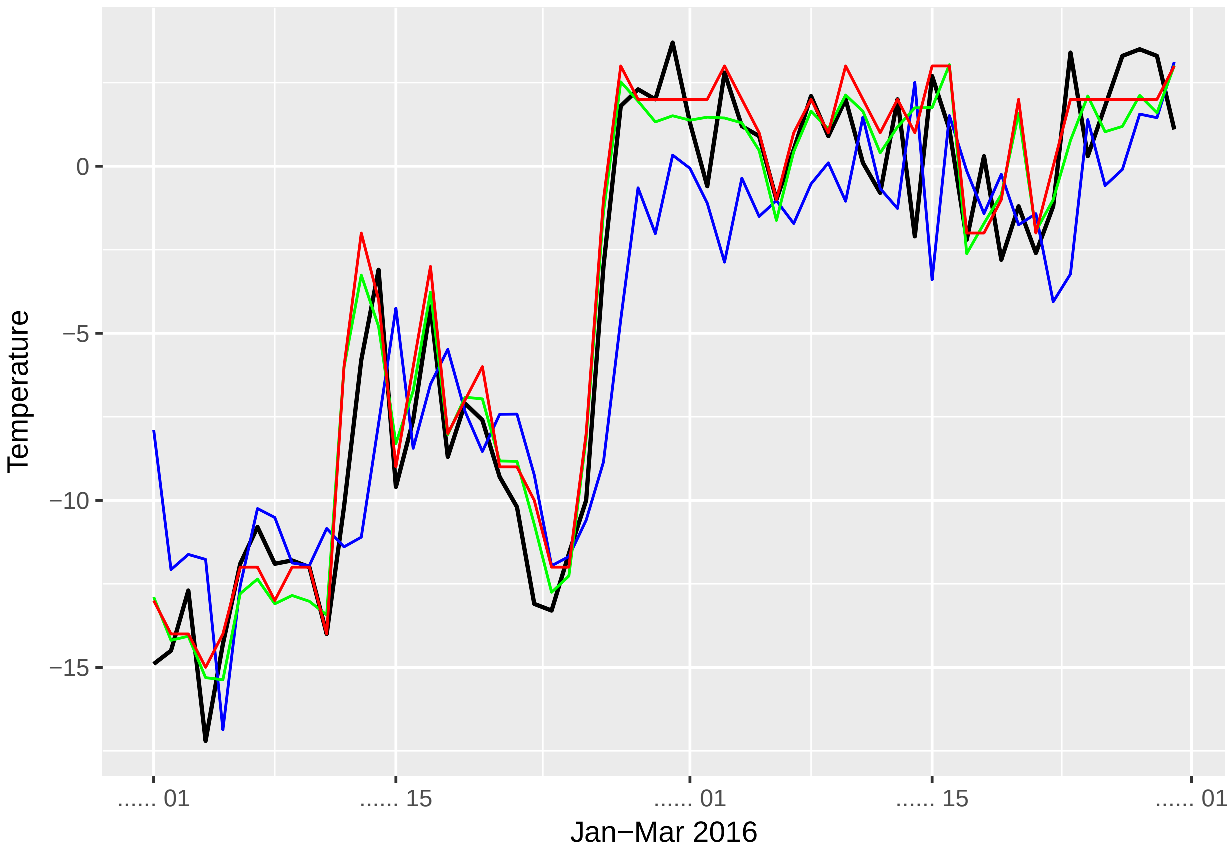
```
mae_total <- round(data.frame(c(mae_11, mae_31, mae_21), c(mae_12, mae_32, mae_22), c(mae_13, mae_33, ma
names(mae_total) <- c("1st day", "2nd day", "3d day")

mae_total
```

```
##                    1st day 2nd day 3d day
## linear model          2.57    3.22   3.51
## offical forecast      1.49    2.44   3.12
## lm + official         1.37    1.50   1.77
```

The predictions based only on the previous observations can give moderate results in weather forecasting. Traditional predictions give a good estimation of real weather conditions, but results decline with the increase of horizon. The mixture of these two approaches shows the best performances on this lenght.

The difference in predictions is represented on the graph below. They all have the same trend.

```
ggplot(results[1:60,], aes(x = date)) +
  geom_line(aes(y = `T`), size = 0.75) +
  geom_line(aes(y = predictions_lm), colour = "blue") +
  geom_line(aes(y = predictions_lm_expand), colour = "green") +
  geom_line(aes(y = V4), colour = "red") +
  labs(x="Jan-Mar 2016",y="Temperature")
```



This model is not a final decision and other methods (say, xgboost) can likely reach the better score.