UWPCE DataScience3 KaggleWNVP Project Report

Kaggle Team 3

2 June 2015

Background

WNVP: "West Nile Virus Prediction"

URL of Kaggle Competition.

Goal of Competition: "Predict West Nile virus in mosquitos across the city of Chicago"

Team Goals:

1. Experiment with modelling alternative on a real-world dataset.
2. Learn how to participate in a Kaggle competition.
3. Non-Goal: win the competition, or even score highly.

Team Members

- Bethene Britt
- Andrew Ewing
- Gregory Hogue
- Patrick Leahy
- Michal Monselise
- Linghua Qiu
- Chris Ross
- Robert Russell
- Jim Stearns

Data Preparation

Obtain Original Datasets from Kaggle Website

Download the training, test, spray, and weather data from the Kaggle web site page for West Nile Virus Prediction.

Please see Appendix for R (and Python) code that:

- General Setup: Clears environment, sets working directory, loads libraries
- Downloads the data from the Kaggle site and unzips them.

```r
stopifnot(allKaggleFilesArePresent())
print("All unzipped Kaggle datasets found in PWD. Proceeding.")
```

```
## [1] "All unzipped Kaggle datasets found in PWD. Proceeding."
```

Create train and test datasets compatible with Weka (ARFF) and each other (same attributes)

NOT YET: Distance from weather stations. NOT YET: Weather data joined.

- Read in test and train csv-format files into data frames.
- Make the train and test datasets have the same attributes:
  - Train: Convert the WnvPresent column from numeric to factor with levels "Yes" and "No".
  - Test: Remove Id attribute.
  - Test: Add a NumMosquitos integer column, all 0.
  - Test: Add a WnvPresent factor column, all with "No" level.
- Combine train and test into a single data frame and transform all as follows:
  - Remove the block attributes of little use:
    - Address, Block, Street, AddressNumberAndStreet, AddressAccuracy.
  - Convert the Species factor into bit vectors (a new column for each of the factor levels, with a zero or 1 value)
  - Convert date into date format, add "Year" and "Month" attributes.
- Write the combined file out in two subsets (train and test), with both files in ARFF format with same ARFF header.

```r
train_df <- read.csv(paste0(dataSubDir, "/", wnvpTrainFilename))
train_df$WnvPresent <- as.factor(train_df$WnvPresent)
levels(train_df$WnvPresent) <- c("No", "Yes")

test_df <- read.csv(paste0(dataSubDir, "/", wnvpTestFilename))
test_df$Id <- NULL
test_df <- cbind(NumMosquitos=0, test_df)
WnvPresent <- factor("No", levels=c("No","Yes"))
test_df <- cbind(test_df, WnvPresent)

combined <- rbind(train_df, test_df)

attrsToRemove <- c("Address", "Block", "Street", "AddressNumberAndStreet", "AddressAccuracy")
combined <- combined[,!names(combined) %in% attrsToRemove]

# Convert Species into a bit vector, one column for each factor level
combined <- with(combined, cbind(model.matrix( ~ 0 + Species, combined), combined))

# Tries %Y-%m-%d by default, but what the heck, explicitly state the format.
combined$Date <- as.Date(combined$Date, format="%Y-%m-%d")
Year <- format(combined$Date, "%Y")
Month <- format(combined$Date, "%m")
# Prepend year and month, leaving classification attribute in last column
combined <- cbind(Year, Month, combined)
# Move Date to first column, near Year and Month.
combined <- combined[,c(11,1,2,3,4,5,6,7,8,9,10,12,13,14,15,16,17)]
# Remove the Species attribute - replaced by bit vectors.
combined$Species <- NULL
```

```r
write.arff(combined[1:nrow(train_df),],
           paste0(workingSubDir, "/", "train01", ".arff"),
           eol = '\n', relation="WNVPTrainDataset")
str(combined[1:nrow(train_df),])
```

```
## 'data.frame':    10506 obs. of  16 variables:
##  $ Date                      : Date, format: "2007-05-29" "2007-05-29" ...
##  $ Year                      : Factor w/ 8 levels "2007","2008",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Month                     : Factor w/ 6 levels "05","06","07",..: 1 1 1 1 1 1 1 1 1 1 ..
##  $ SpeciesCULEX ERRATICUS    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ SpeciesCULEX PIPIENS      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ SpeciesCULEX PIPIENS/RESTUANS: num  1 0 0 1 0 0 0 1 0 0 ...
##  $ SpeciesCULEX RESTUANS     : num  0 1 1 0 1 1 1 0 1 1 ...
##  $ SpeciesCULEX SALINARIUS   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ SpeciesCULEX TARSALIS     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ SpeciesCULEX TERRITANS    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ SpeciesUNSPECIFIED CULEX  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Trap                      : Factor w/ 149 levels "T001","T002",..: 2 2 7 14 14 33 34 36
##  $ Latitude                  : num  42 42 42 42 42 ...
##  $ Longitude                 : num  -87.8 -87.8 -87.8 -87.8 -87.8 ...
##  $ NumMosquitos              : num  1 1 1 1 4 2 1 1 2 1 ...
##  $ WnvPresent                : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
```

```r
write.arff(combined[nrow(train_df)+1:nrow(combined),],
           paste0(workingSubDir, "/", "test01", ".arff"),
           eol = '\n', relation="WNVPTestDataset")
str(combined[nrow(train_df)+1:nrow(combined),])
```

```
## 'data.frame':    126799 obs. of  16 variables:
##  $ Date                      : Date, format: "2008-06-11" "2008-06-11" ...
##  $ Year                      : Factor w/ 8 levels "2007","2008",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Month                     : Factor w/ 6 levels "05","06","07",..: 2 2 2 2 2 2 2 2 2 2 ..
##  $ SpeciesCULEX ERRATICUS    : num  0 0 0 0 0 0 0 1 0 0 ...
##  $ SpeciesCULEX PIPIENS      : num  0 0 1 0 0 0 0 0 0 0 ...
##  $ SpeciesCULEX PIPIENS/RESTUANS: num  1 0 0 0 0 0 0 0 1 0 ...
##  $ SpeciesCULEX RESTUANS     : num  0 1 0 0 0 0 0 0 0 1 ...
##  $ SpeciesCULEX SALINARIUS   : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ SpeciesCULEX TARSALIS     : num  0 0 0 0 0 1 0 0 0 0 ...
##  $ SpeciesCULEX TERRITANS    : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ SpeciesUNSPECIFIED CULEX  : num  0 0 0 0 0 0 1 0 0 0 ...
##  $ Trap                      : Factor w/ 149 levels "T001","T002",..: 2 2 2 2 2 2 2 2 7 7 .
##  $ Latitude                  : num  42 42 42 42 42 ...
##  $ Longitude                 : num  -87.8 -87.8 -87.8 -87.8 -87.8 ...
##  $ NumMosquitos              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ WnvPresent                : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
```

Distance of Trap from the Two Weather Stations

NOT YET USED

Distance function by "Curlew"

Alternative: R Geosphere Package

```r
# From https://conservationecology.wordpress.com/2013/06/30/distance-between-two-points-in-r/
# Calculate distance in kilometers between two points
earth.dist <- function (long1, lat1, long2, lat2)
{
    rad <- pi/180
    a1 <- lat1 * rad
    a2 <- long1 * rad
    b1 <- lat2 * rad
    b2 <- long2 * rad
    dlon <- b2 - a2
    dlat <- b1 - a1
    a <- (sin(dlat/2))^2 + cos(a1) * cos(b1) * (sin(dlon/2))^2
    c <- 2 * atan2(sqrt(a), sqrt(1 - a))
    R <- 6378.145
    d <- R * c
    return(d)
}
```

Appendix

General Setup: Clear environment, set working directory, load libraries

```r
# Clear the working environment of variables, data, functions
rm(list=ls())

# Set working directory for this Kaggle project. Default: pwd.
#kaggleProjHomeDir <- "."
kaggleProjHomeDir <- "/Users/jimstearns/GoogleDrive/Learning/Courses/UWPCE-DataScience/Course3_Da
setwd(kaggleProjHomeDir)
getwd()

#install.packages("rPython")  # For download from web site with login/pwd.
library(rPython) # For calling python function to download file w/login+pwd
# Package for writing Weka ARFF file format
stopifnot(require("foreign"))
library("foreign")
```

Dataset download and unpacking

This R and Python code downloads the WNVP datasets from Kaggle. Some setup is required:

- One's Kaggle username and password must be defined as environment variables where R is running.
- Easiest way to set environment variable for R: Create (add to) ~/.Renviron file (kaggleUsername="XXXX" and kagglePassword="YYYY").

Alternatively, files can be downloaded manually.

```r
wnvpTrainFilename <- "train.csv"
wnvpTestFilename <- "test.csv"
wnvpWeatherFilename <- "weather.csv"
wnvpSprayFilename <- "spray.csv"
kaggleDatasets = c(
    wnvpTrainFilename,
    wnvpTestFilename,
    wnvpWeatherFilename,
    wnvpSprayFilename)
dataSubDir <- "input"  # Kaggle convention
workingSubDir <- "working" # Kaggle convention: massaged datasets - and output - go here.

# If download from Kaggle required, and user and pwd are empty (default),
# then user will be prompted for these two values.
kaggleUsername <- ""
kagglePassword <- ""

allKaggleFilesArePresent <- function() {
    filesAllFound <- TRUE
    for (file in kaggleDatasets) {
        if (!file.exists(paste0(dataSubDir, "/", file))) {
            print(paste("Error: could not find unzipped Kaggle file in PWD:", file))
            filesAllFound <- FALSE
        }
    }
    return(filesAllFound)
}

downloadMissingKaggleFiles <- function() {
    python.load("src/UrlFileDownloaderWithLogin.py")

    kaggleUsername = Sys.getenv("kaggleUsername")
    kagglePassword = Sys.getenv("kagglePassword")
    if (kaggleUsername == "" || kagglePassword == "") {
        print("Please assign kaggleUsername and kagglePassword environment variables.")
        print("Place in ~/.Renviron entries such as kaggleUsername='YourName'.")
    }
    stopifnot(!(kaggleUsername == ""))
    stopifnot(!(kagglePassword == ""))

    wnvpKaggleDataUrl <-
        "https://www.kaggle.com/c/predict-west-nile-virus/download/"

    for (file in kaggleDatasets) {
        if (file.exists(file))
            next

        urlOfZip <- paste0(wnvpKaggleDataUrl, file, ".zip")
        print(urlOfZip)
        # Use a python method to download from URL with login and password.
        # Download to subdirectory "input" and filename w/o the .zip suffix.
        python.call("Download", urlOfZip,
                    kaggleUsername, kagglePassword ,
                    paste0(dataSubDir, "/", file, ".zip"))
    }
}

unzipDownloadedFiles <- function() {
    for (file in kaggleDatasets) {
        zippedFile <- paste0(dataSubDir, "/", file, ".zip")
        print(paste0("Unzip: ", zippedFile))
        if (file.exists(zippedFile)) {
            if (file.exists(file)) {
                print(sprintf("Warning: removing existing file %s\n", file))
                file.remove(file)
            }
            unzip(zippedFile, exdir=dataSubDir)
            print(sprintf("Unzipped: %s\n", zippedFile))
        }
    }
}

if (!allKaggleFilesArePresent()) {
    print(paste("Not all needed Kaggle datasets are present in PWD;",
                "attempting to download from Kaggle web site."))
    downloadMissingKaggleFiles()
    unzipDownloadedFiles()
}
```

File UrlFileDownloaderWithLogin.py:

```
__author__ = 'jimstearns'
""" Download a file at a URL at a web site that requires a user name and password.
"""

import logging
import os       # File utilities

# Python package "requests": "Python HTTP for Humans" by Kenneth Reitz. Current version: 2.7.0.
# Documented at http://docs.python-requests.org/en/latest/
# To install from the command line: "pip install requests"
# (On Mac, sudo may be required. Also pip2.7 instead of pip, depending on default Python version)

import requests # Http GET, POST

def Download(url, username, password, local_filename):
    # Login to web site such as Kaggle and retrieve the data. Use POST rather than GET as as to
    # send login info in body of HTTP request rather than in query string portion of URL.

    # Limitation: when used by Python version < 2.7.9, an "InsecureRequestWarning" is generated.
    # TODO: Fix. Details: https://urllib3.readthedocs.org/en/latest/security.html#insecureplatfor
    # Workaround: log warnings to file, not stdout.
    logging.captureWarnings(True)

    if (os.path.exists(local_filename)):
        os.remove(local_filename)

    # This won't get the file, but use the return value URL in a follow-on POST:
    r = requests.get(url)

    login_info = {'UserName': '{0}'.format(username), 'Password': '{0}'.format(password) }
    print(login_info)
    r = requests.post(r.url, data = login_info)
    print("POST (w/login info): {0}\n".format(r.status_code))

    # Write the data to a local file one chunk at a time.
    chunk_size = 512 * 1024 # Reads 512KB at a time into memory
    with open(local_filename, 'wb') as fd:
        for chunk in r.iter_content(chunk_size): # Reads 512KB at a time into memory
            if chunk: # filter out keep-alive new chunks
                fd.write(chunk)

    if (os.path.exists(local_filename)):
        return(True)
    else:
        return(False)
```