Solution for West Nile Virus Prediction
https://www.kaggle.com/c/predict-west-nile-virus
2nd place, Lucas Silva and Dmitry Efimov

Contents

Section 1. Approach summary

The purpose of this challenge was to predict when and where different species of mosquitos will test positive for West Nile Virus. Our solution contains the following steps.

Step 1. Building datasets.
Step 2. Models.
Step 3. Leaderboard feedback and multipliers.
Step 4. Ensembling.

Section 2. Building datasets

We have constructed two different datasets with different sets of features. Groups of instances with similar Trap, Date and Species have been combined in one instance with target WnvPresent calculated as max(WnvPresent) by all targets inside the group. Both datasets have been filtered by CULEX PIPIENS/RESTUANS, CULEX PIPIENS and CULEX RESTUANS species only (for the rest of species we have assigned zero probabilities for the final predictions). After that the original train set has been augmented by adding all non-existed combinations of Trap, Species and Date for each year. Those added instances were not used for training but to calculate structural features and perform time and location smoothing.

The dataset for the Gradient Boosting Classifier contains the following features:

1) **Weather features:** TmaxSt1Smth02, TminSt1Smth02, TavgSt1Smth02, DepartSt1Smth02, DewPointSt1Smth02, HeatSt1Smth02, CoolSt1Smth02, PrecipTotalSt1Smth02, SnowFallSt1Smth02, ResultDirSt1Smth02, TmaxSt2Smth02, TminSt2Smth02, TavgSt2Smth02, DewPointSt2Smth02, PrecipTotalSt2Smth02, ResultDirSt2Smth02, SunriseSt1, SunsetSt1.

The suffices St1, St2 in the feature names mean the weather information from the stations 1 and 2 correspondingly. The suffix Smth02 means that the feature has been smoothed inside the year with local polynomial regression fitting (function loess in R, [1]) with span = 0.4.

2) **Basic features:** SpeciesTree (species coded as ordinal sorted by average WnvPresent inside each Species), Block, Trap (traps coded as ordinal sorted by average WnvPresent inside each Trap), Latitude, Longitude, AddressAccuracy, Month, Year.

3) **Structural features:** TrapCount (number of mosquitos batches for fixed Trap, Date, Species), TrapCountPrevAge (the age in days that TrapCount was previously not 0, capped at 90 days), TrapCountPrev (the value of TrapCount at TrapCountPrevAge).

The dataset for Regularized Greedy Forest contains the following features:

1) **Weather features:** TS (binary feature, 1 if TS is present, 0, otherwise), FG(binary feature, 1 if FG is present, 0 otherwise), DewPoint, Tmin.

The weather information from the closest station has been collected for this dataset.

2) **Basic features:** Longitude, Latititude, DayYear, Species (set of binary features, corresponding to Species).

3) **Structural features:** NumBatches (the same as TrapCount in the previous dataset), NumBatchesPrevious30Sum (sum of NumBatches by Trap and Species for the previous 30 days), NumBatchesPrevious30Max (max of NumBatches by Trap and Species for the previous 30 days).

Section 3. Models

Our prediction was obtained with two decision tree based algorithms: Gradient Boosting Classification Decision trees [2] (package *scikit-learn* in *python*) and Regularized Greedy Forest [3].

1) The list of parameters for Gradient Boosting Classifier:

Parameter	**Value**
n_estimators	1000
learning_rate	0.0035
loss	deviance
max_features	8
max_depth	7
subsample	1

2) The list of parameters for Regularized Greedy Forest:

Parameter	Value
reg L2	0.2
reg sL2	0.07
algorithm	RGF
loss	Log
num iteration opt	7
num tree search	1
min pop	8
opt stepsize	0.7
max leaf forest	1400

It is worth to notice that we tried a lot of other methods (such as Random Forest Classifier [4], generalized linear methods [5], ExtraTrees [6]). Some of them showed good results, but, unfortunately, did not improve the final ensembling. Also the single Regularized Greedy Forest model gives very good accuracy which is enough to save the second position on the leaderboard.

We have applied the smoothing for the predictions based on the closest traps and dates. The Haversine distance [7] has been calculated to find the closest traps.

For the Gradient Boosting Classifier model predictions from the 4 closest traps, 3 previous days and 3 future days have been added to the weighted linear combination (the closer trap (date), the higher weight of the prediction). For the Regularized Greedy Forest model predictions for the 6 closest traps, 11 previous and 7 future days have been added to the simple average, the original prediction for each Trap, Species, Date has been removed from this average (in RGF model only).

Section 4. Leaderboard feedback, multipliers, ensembling

Two types of multipliers have been applied. AUC scores for each year have been obtained from the leaderboard. Using these AUC scores we have constructed the linear regression model to predict relative average PA_{year} of WnvPresent for each year, the same way we have obtained the relative average PA_{month} of WnvPresent by months (see the tables).

Year	PA_{year}
2008	0.21311596
2010	0.19412383
2012	1.00000000
2014	0.39061037

Year	Month	PA_{month}
2008	6	0.44132550

2008	7	0.00000000
2008	8	0.34004021
2008	9	1.00000000
2010	6	0.60622771
2010	7	0.00000000
2010	8	0.89552084
2010	9	0.83362353
2010	10	1.00000000
2012	6	0.06476890
2012	7	1.00000000
2012	8	0.89533165
2012	9	0.08189057
2014	6	0.04161481
2014	7	0.00000000
2014	8	1.00000000
2014	9	0.57483363
2014	10	0.35872873

For each model we have applied the multipliers calculated as PA_{year}/AP_{year}, where AP_{year} is an average of predictions by years, after we have applied multipliers calculated as PA_{month}/AP_{month}, where AP_{month} is an average of predictions by months. We have constructed the 1st level ensembling as a linear combination

$$P1 = (P * (PA_{year}/AP_{year}) + 6 * P * (PA_{year}/AP_{year}) * (PA_{month}/AP_{month}))/7,$$

where P is an original predictions from each algorithm.

We have applied additional multipliers to P1 based on the following table (multipliers have been obtained based on the leaderboard feedback).

Filter	**Multiplier**
Year = 2012 and Month = 7	1.6
Year = 2012 and Month = 9	0.3
Year = 2012 and Month = 8	0.8
Year = 2008 and Month = 9	1.2
Year = 2008 and Month = 7	0.85
Year = 2012 and Month = 8 and WeekYear = 35	0.3
Year = 2012 and Month = 8 and WeekYear = 34	0.3
Year = 2012 and WeekYear = 32	0.7
Year = 2012 and WeekYear = 31	1.6

Year = 2012 and WeekYear = 31	1.6
Year = 2012 and WeekYear = 30	1.6
Year = 2012 and WeekYear = 29	1.1
Year = 2012 and Month = 7 and WeekYear = 28	0.9
Species = “CULEX PIPIENS”	0.5
Species = “CULEX RESTUANS”	0.9

The final ensembling was the linear combination of prediction ranks from two models, normalized to probabilities: 3*rank(RGF) + 1*rank(GBC).

Section 5. File list and training

Task	File name	Description
Datasets building	data.build.gbc.R data.build.rgf.R	**Output:** data tables in the folder data/output-r
Models training	01-train.gbc.01.R 02-train.rgf.02.R	**Output:** data tables with predictions for the test set (saved in the folder data/output-r)
Ensembling	03-train.ens.R	**Output:** ensembled prediction for the test set (saved to the file data.ens.all.pred.sub.csv.7z in the folder data/submission)
Helper function	fn.base.R	The set of helper functions

To calculate the final ensembling:

1. Put all data files (extracted) into the "data/input" folder.
2. Open R session with folder "west-niles-virus-r" set as working dir.
3. Run the R script files in the folder "west-niles-virus-r" in the following order:
   3.1. *01-train.gbc.01.R*
   3.2. *02-train.rgf.01.R*
   3.3. *03-train.ens.R*

3. The predictions will be saved in the data/submission/data.ens.all.pred.sub.csv.7z after running 03_train.ens.R

Pre-requisites:

All files consider the folder "west-niles-virus-r" as the working dir. Using RStudio and loading the project inside will do it.

The model can be run on Linux Ubuntu 12.04 or Mac OS.

Dependencies:

Python: pandas 0.12.0, numpy 1.8.0, scikit-learn 0.16.1.

R: SOAR, doSNOW, foreach, cvTools, data.table, parallel, rlecuyer, SparseM, Matrix, *R:* SOAR, doSNOW, foreach, cvTools, data.table, parallel, rlecuyer, SparseM, Matrix, geosphere.

All the listed versions are the used ones. It will probably work with newer versions, but it was not tested. The R version used was 3.1.0, Python version used 2.7.3.

Section 6. References

[1] W. S. Cleveland, E. Grosse and W. M. Shyu (1992). *Local regression models. Chapter 8 of Statistical Models in S* eds J.M. Chambers and T.J. Hastie, Wadsworth & Brooks/Cole.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay (2011). *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, 12, pp. 2825-2830.

[3] R. Johnson and T. Zhang (2011). *Learning nonlinear functions using regularized greedy forest.* Technical report, Tech Report: arXiv:1109.0887.

[4] L. Breiman (2001). *Random Forests.* Machine Learning, pp.5-32.

[5] J. Friedman, T. Hastie and R. Tibshirani (2008). *Regularization paths for generalized linear models via coordinate descent.* Journal of Statistical Software, Vol. 33(1), 1-22.

[6] J. Simm and I. Magrans de Abril (2013). *Package for ExtraTrees method for classification and regression.*

[7] R.W. Sinnott (1984). *Virtues of the Haversine.* Sky and Telescope, 68 (2), 159.