



Project 4: West Nile Virus Prediction

Bryan, Maybelle, Kah Fei, Song Yi, Cheryl

Overview

- 1) Problem Statement and Background
- 2) Data Cleaning / Feature Engineering / EDA
- 3) Hyperparameter Tuning and Modelling
- 4) Results
- 5) Conclusion and Recommendations



Problem Statement and Background

Problem Statement

We are a group of consultants for the City of Chicago and Chicago Department of Public Health (CDPH). Our task is to:

- predict when and where different species of mosquitoes will test positive for **West Nile Virus**; and
- Investigate if the effect of spray could be optimized.

The data provided are: weather, traps information, and spraying data.

Background

- Mosquitoes obtain the virus from birds. The virus is transmitted to us when the mosquitoes bite us.
- Identification of the virus was first made in 1937 in Uganda. The virus reach the borders of USA in 1999.
- 20% who are infected with the virus develop symptoms ranging from a persistent fever, to serious neurological illnesses that can result in death.

Background

- First human case of West Nile Virus were reported in Chicago in 2002.
- Every week from late spring through the fall, mosquitos in traps across the city are tested for the virus.
- The results influence when and where the city will spray airborne pesticides to control adult mosquito populations.
- A more accurate method of predicting outbreaks of West Nile Virus in mosquitoes will aid the City of Chicago and CDPH more efficiently and effectively allocate resources towards the prevention of the virus.



Data Cleaning and EDA

EDA



Data Cleaning

All datasets

- `pd.read_csv(infer_datetime_format=True, parse_dates=[int])`
- Check for null and missing values
- Set dates as index, and sort by date

Train/Test.csv

- Countvectorized ``species`` feature – separate ``pipiens/restuans``, add ``unspecified`` for train.csv

Data Cleaning (cont'd)

Weather.csv

- Replace 1609 missing values in `CodeSum` feature with np.nan, then fillna('M') for count vectorization
- `CodeSum` are 2-letter codes for weather conditions, some values had code combinations. Used count vectorization to separate combinations, and account for individual codes within combinations
- Drop original `CodeSum` feature
- Dropped depart, heat, cool - likely collinearity with temperature features
- Dropped depth, water1, snowfall - station 2 missing these data

Data Cleaning (cont'd)

Weather.csv (cont'd)

- Duplicate sunrise, sunset values from station 1 to station 2

Merge preparation

- Split weather.csv into station 1 and station 2
- Split train and test.csv using latitudes and longitudes (arbitrary split based on proximity to each weather station)
- Merge train and weather / test and weather using train/test date index by stations since train/test date range is narrower
- Merge stations together

Data Cleaning (cont'd) + Feature engineering

Train/Test + weather dataset

- Dropped 44 datapoints from `WetBulb` and `StnPressure` features that share 'M' values
- `PrecipTotal` feature has 'T' and 'M' values. Replaced 'T' values with 0.00, and dropped 21 datapoints containing 'M' values
- Checked for non-numeric dtypes – convert to numeric
- `pd.get_dummies(columns=['Trap'])`
- test.csv has traps not found in train.csv. Added traps to train data, to match test data.

Data Cleaning (cont'd) + Feature engineering

Using Spray.csv as reference

- Dropped `Time` feature - not using to that granularity
- Only has 10 unique dates, however 2011-08-29 is irrelevant as all datapoints on that day are outliers (95)
- Create new feature `spray` using spray dates and locations. Took an arbitrary date range after each spray (date range must be within dataset), and also arbitrary radius around spray - to assign traps that were sprayed vs traps that were not
- For test set, `spray` feature was assigned zero values to signify no spray (no spray data for test set dates)

2011-08-29

2011-09-07

2013-07-17

2013-07-25

2013-08-08

2013-08-15

2013-08-16

2013-08-22

2013-08-29

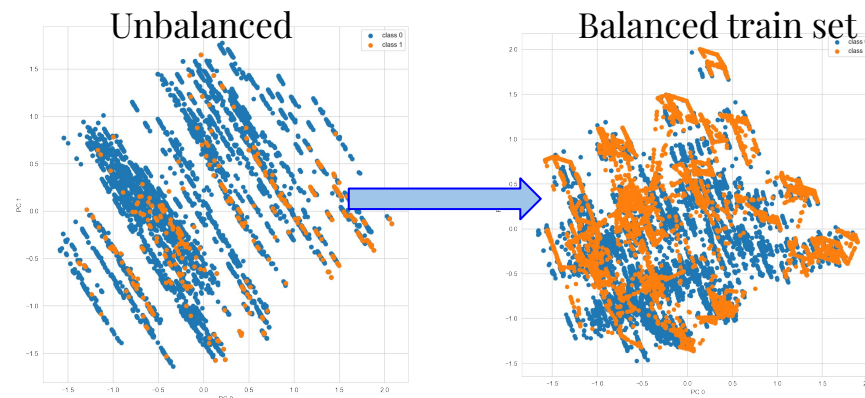
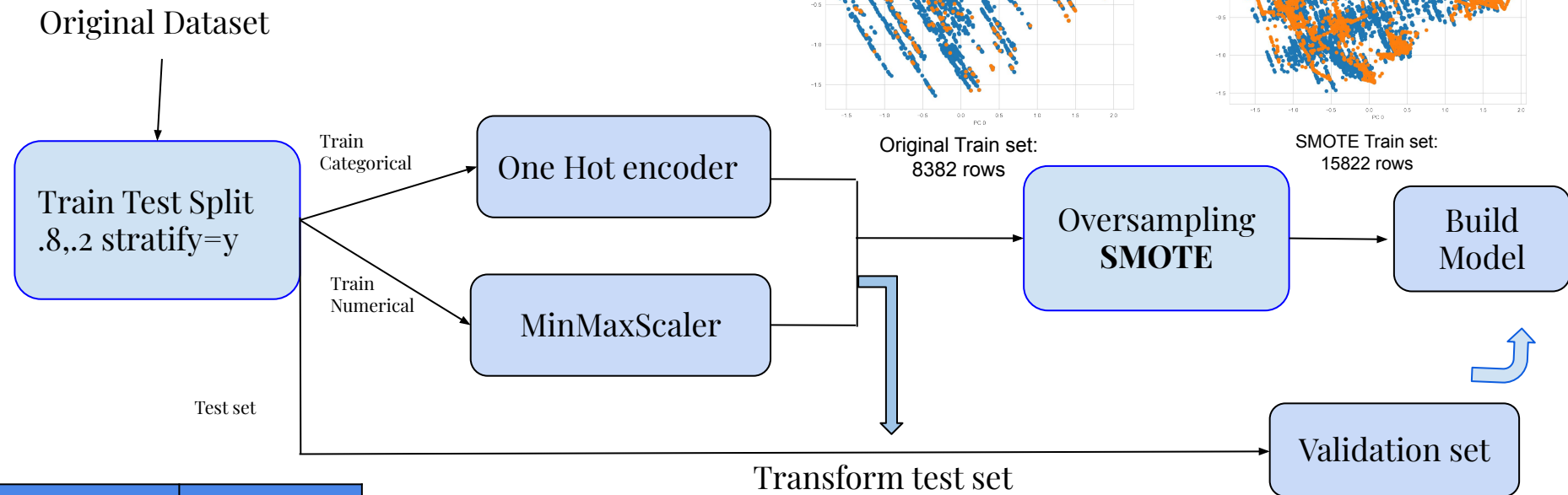
2013-09-05

Data Cleaning (cont'd)

Dates as features

- `df.reset_index()` since have been using dates as index
- Separate dates into new ``day``, ``month``, ``year`` features
- Dropped original ``Date`` feature
- `pd.get_dummies(columns=['Year','Month','Day'])`
- Add new dates features to both train and test sets to make sure features match (dropped ``NumMosquitos``)
- Export files for machine learning

Dealing with unbalanced data

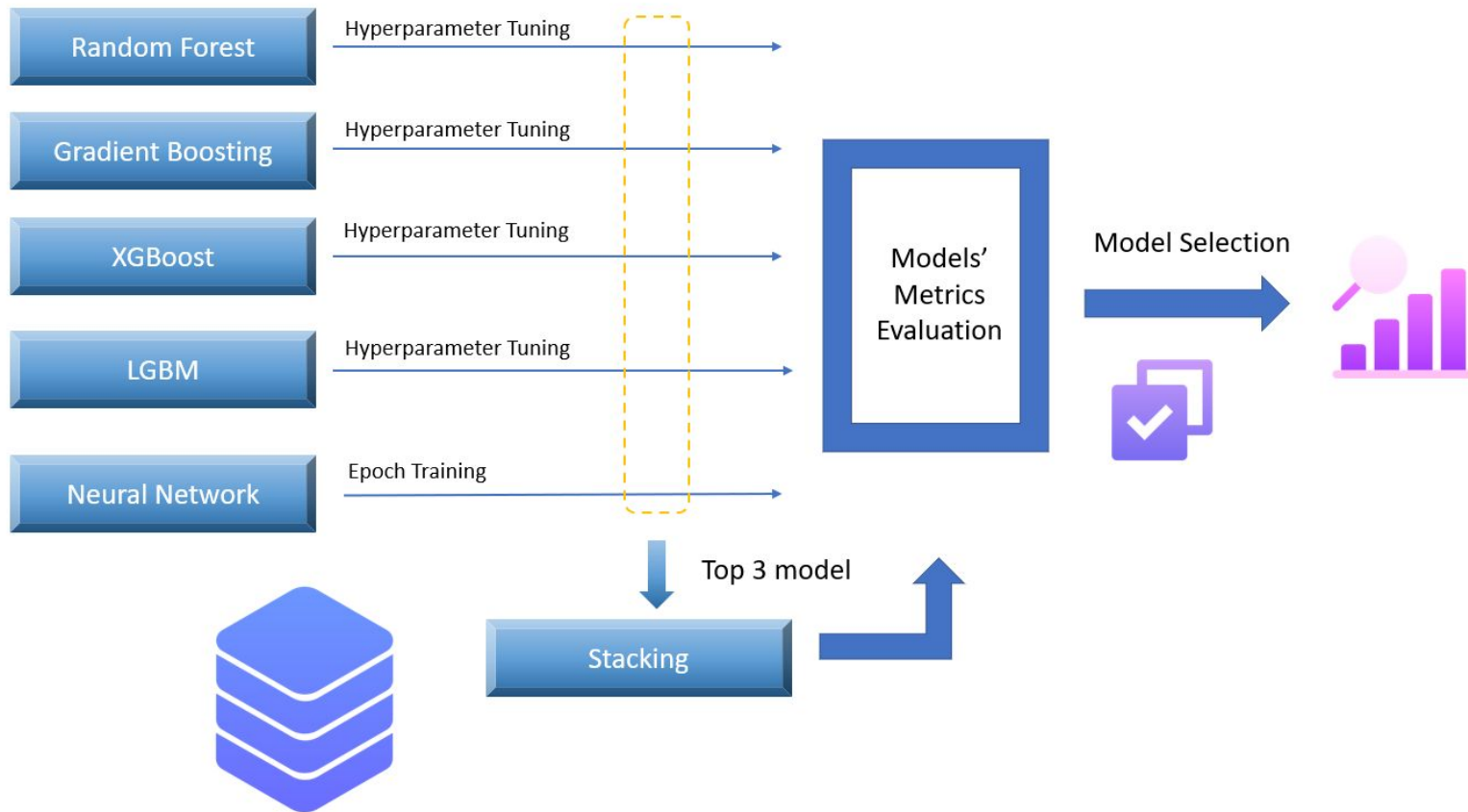


Class	Weight
Negative	94.72%
Positive	5.28%

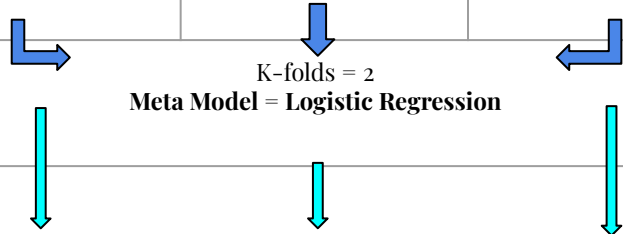


Hyperparameter Tuning and Modelling

Modelling

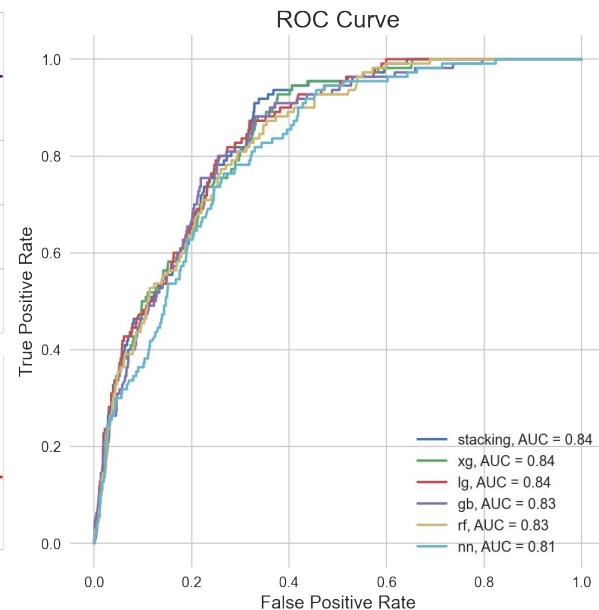
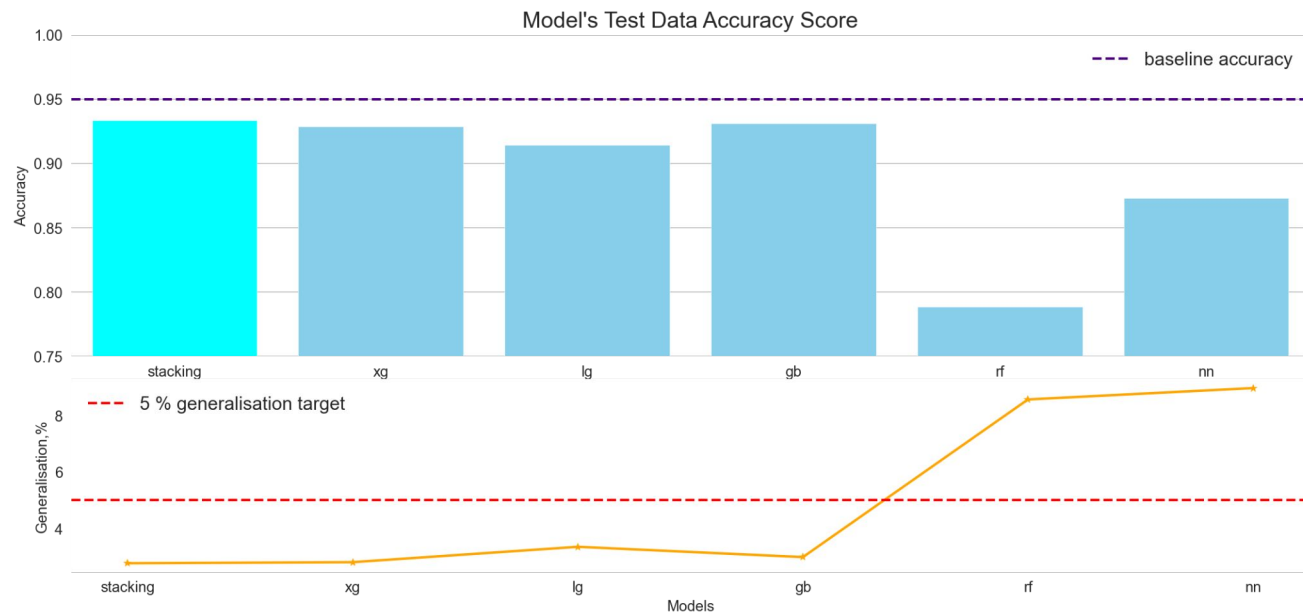


Gridsearch Result and Stacking

Model	XGboost	LGBM	Gradient Boost	Random Forest	Neural Network
Hyperparameter	Learning_rate = 0.2 Objective=binary:logistic Max_depth = 4 N_estimators = 300 tree_method = hist	objective=binary N_estimators=500 learning_rate=0.01 num_leaves=50 max_depth=35 reg_alpha=0.3	n_estimators=400 max_depth=10 min_samples_split=2 max_features=log2 learning_rate=0.15	n_estimators=1600 max_depth=10 min_samples_split=10 min_samples_leaf=5	Input layer = 100 neurons, activation 'relu' Hidden layer = 1 (100 neurons, activation 'relu') Output activation 'sigmoid' Optimizer : Adam Loss: binary_crossentropy
Test Accuracy	0.929	0.914	0.931	0.788	0.873
Stacking	 <p>K-folds = 2 Meta Model = Logistic Regression</p>				

Due to data imbalance in the test (5% positive class), evaluation metrics including :
Accuracy, ROC-AUC, Generalisation and F1 score

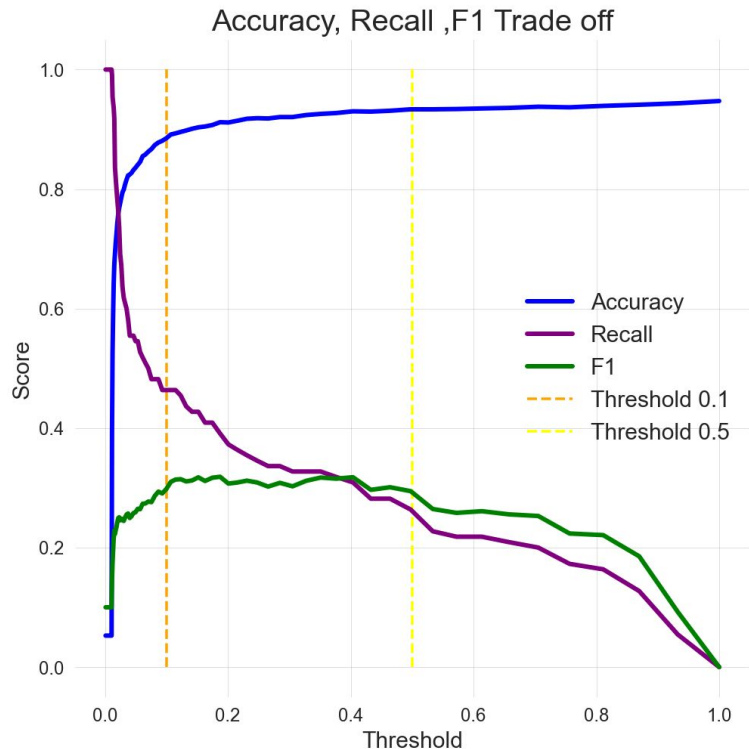
Evaluation and Model Selection - Stacking is the best performer



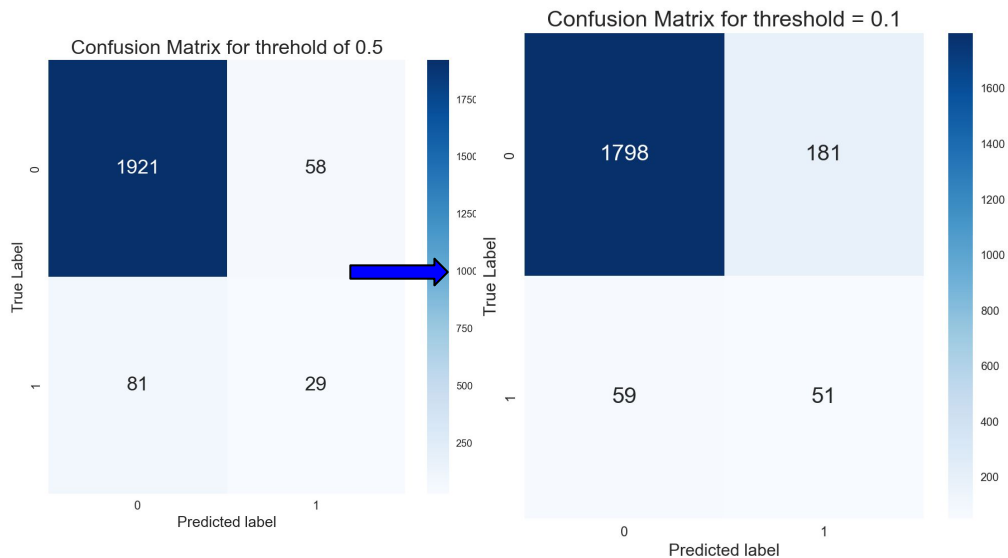
Top 2	Accuracy	Generalisation	ROC-AUC
Stacking	0.933	2.77	0.844
Gradient Boost	0.931	2.99	0.833

- Slightly lower than baseline accuracy (<2%)
- Stacking method has the overall best metrics
- Not worth the engineering resources to stack the models, could have just use Gradient Boosting
- Kaggle Score : 0.6995

Model Tuning - Sacrifice Accuracy for better recall & F1 score



Metrics	Threshold = 0.5	Threshold = 0.1	Changes
Accuracy	0.93	0.89	-4.3%
F1 score	0.28	0.3	7%
Recall	0.26	0.46	77%

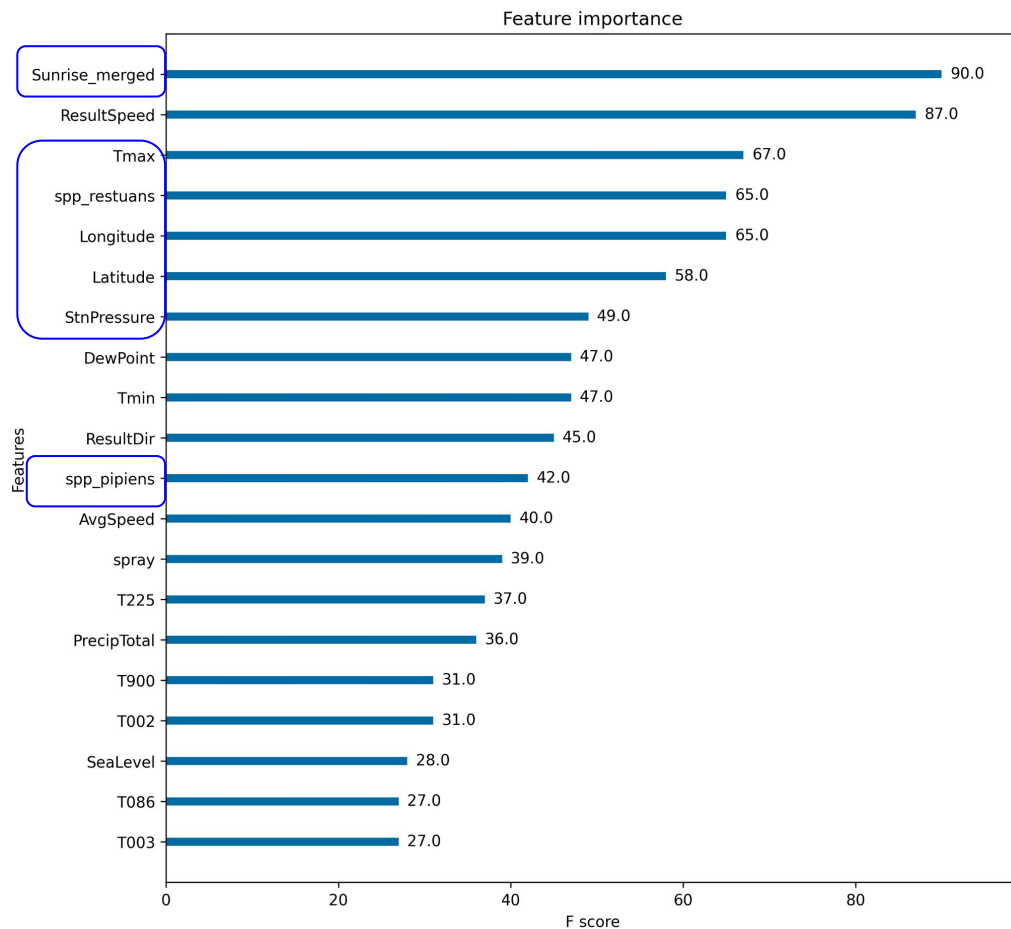


- In the study on the presence of WNV, the accuracy in predicting the “True Positive” is important
- Prefer high recall and high F1 score for positive class
- Minimize “False Negative” (miss out the positive virus)



Results

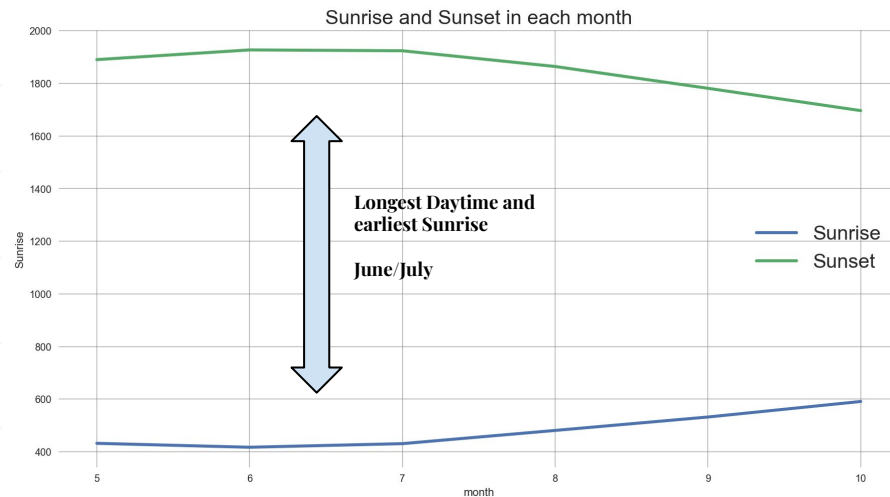
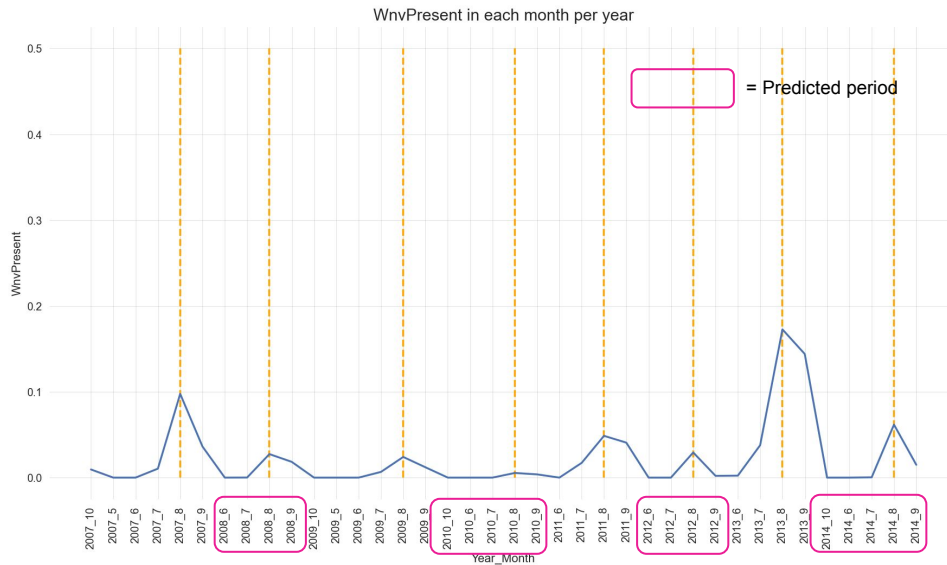
Result Analysis - Targeted features



3 Categories of factors:

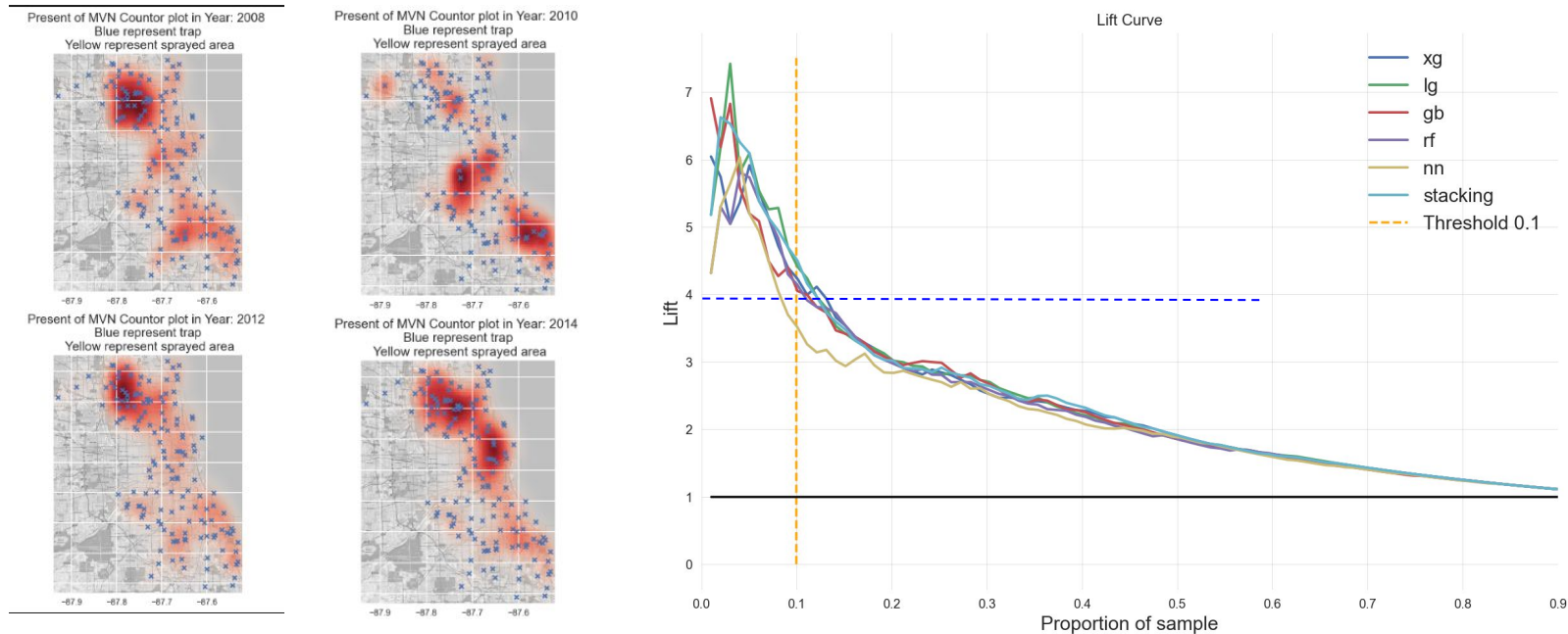
1. Time of the Year
 - Temperature
 - Sunrise /Sunset
 - Standard Pressure
2. Location
 - Latitude
 - Longitude
3. Species
 - Culex Restuans
 - Culex Pipiens

Result Analysis - Optimise spraying time



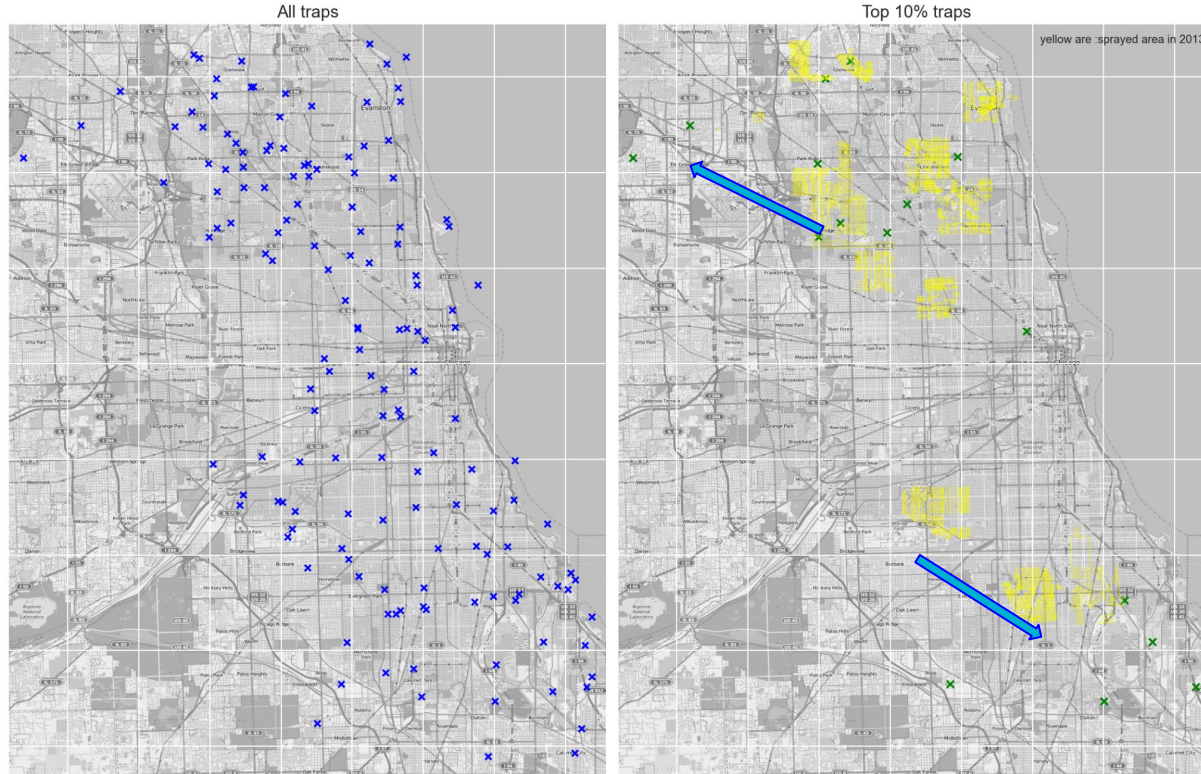
- The sprays were conducted in August 2013 (the peak month) however they were not effective.
- From the months of June to July, length of day increases, correspondingly average temperature spikes.
- The mosquitoes are active in early morning, which contrasts with human behaviour.
- Hence, mosquitoes tend to feed on birds instead, which is the virus main medium of transmission.
- Alter the **spraying period from August to June/July**

Result Analysis - Optimise spraying location



- Target locations present with high clusters of West Nile Virus
- The top 10% of the clustered locations has 4 times higher chances of positive detection than to select randomly over all the traps

Result Analysis - Optimise spraying location



The spray locations can be optimized to target to the top 10% of the location.

Chances of “hitting” the mosquitoes is 4 times higher.

Let $\$X$ be the original cost, we only need $\$ 0.25X$ to achieve the same result.

Cost savings: 75%
(assuming the spray is effective)



Conclusion and Recommendations

Conclusion and Recommendations

1. The effect of spraying in 2013 is inconclusive since there are no significant changes in the following weeks/months.
2. In future, we encourage the data collection team to collect more data so that we can generate more insights on the result of spraying. E.g.: at least 12-24 months of data after the spray.
3. The effect of spray could be optimized through the following ideas based on our team's analysis:
 - a. Change the spraying period from August to June or July.
 - b. With limited budget, to focus the spray on the top 10% of locations that having higher cluster of WNV carrier
 - c. Genetically engineer the insecticide to target the species: Culex Restuans/Pipiens.
4. The base boosting model (Gradient Boost/Adaboost) could be the best fit for this data science problem. Stacking of models is not worth the engineering resources due to time taken and the insignificant improvement.



Thank you. Questions?

Copyright © 2022

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law.