# Fighting the West Nile Virus

Cameron Cruz, Aayush Agrawal
Stanford University
{camcruz, aayush2k} @stanford.edu

## Abstract

*Mosquito carriers of the West Nile Virus have spread the disease throughout Chicago for the last 15 years. Although efforts are in place to perform laboratory tests on mosquito traps, the turnaround time is far too long. Thus, we propose making same-day predictions using a neural network - a model chosen after experimenting with logistic regression, decision trees, random forests, and support vector machines. In this paper, we discuss the process we used to build all of our models, and assess their accuracy. Finally, we derive insights from our best model (neural network) and analyse its results.*

## 1. Introduction

Every year, around a million people die from mosquito-borne diseases including malaria and West Nile Virus (WNV). As a result, city health institutions must be able to detect the presence and movement of mosquitoes with the ability to carry these deadly diseases. Currently, cities like Chicago use mosquito traps and manual laboratory testing to detect WNV carriers. However, this system demands significant manual effort and has a turnaround time of about a week, costing the city resources and lead time that could be used to address mosquitoes carrying WNV.

### 1.1. Task Definition

Thus, our goal was to reduce this lead time - allowing Chicago to place preventative measures quicker. We concluded that the ability to make same-day predictions on where mosquitoes carrying WNV will appear did just that. As a result, we had the task of modelling the presence of WNV in Chicago over time. For the sake of thoroughness, we tried many models including logistic regression (with L1 Lasso Penalty), decision trees, random forests, support vector machines, and neural networks. From the fits - especially the most accurate ones, we gathered insights on WNV as well.

## 2. Literature Review

Most of the literature related to mosquito-borne diseases is understandably more centered on the identification of mosquitoes. This is a valiant goal, as there are certain species that are genetically predisposed to having certain diseases. Most projects with the same used convolution neural networks to identify mosquitoes known for carrying malaria, Zika, and dengue [7]. However, such projects are outside of the scope of our investigation; if anything, identifying mosquitoes quickly would serve as an interesting follow-up.

More recently, computer scientists have been using AI to track disease spread in humans. While this is not synonymous with our project - as we were more interested with tracking the disease before it infects humans, the type of data being analyzed is very similar. The paper that caught our eye was 2015 study in which researchers were able to implement a dynamic neural network for predicting the risk of Zika in real time [1]. This is because of the parallels to our WNV investigation in Chicago, as the researchers wanted to use temporal and geo-spatial data about Zika cases for the purpose of allocating preventative surveillance resources. A Nonlinear AutoRegressive neural network with eXogenous inputs (NARX) was used, and proved to be an accurate predictor with 85% test accuracy. However, we were not inspired to use the same model as the researchers had access to much more varied data (including socioeconomic and airline travel data) that would necessitate a much more flexible model than ours.

We were also able to find a study that extremely similar to ours. Researchers from the University of North Dakota were also motivated to tackle WNV, and thus decided to model the occurrences of mosquitoes which are predisposed to WNV in North Dakota over a 10 year period [3]. Notice that while the project is similar to ours, it deals with a much more relaxed problem - as it is only tracking the number of mosquitoes, not the actual positive readings of WNV. The researchers used Partial Least Squares Regression (PLSR) on mosquito capture data as well as meteorological data such as rainfall, temperature, precipitation, and relative humidity. We were inspired by the sheer amount of work also done during the pre-processing stage, including normalizing and organizing the different data sources. Additionally, we were still interested in using more complex models than PSLR because wanted to model a more complex relationship - one that also dealt with the actual appearance of WNV as well.

## 3. Approach

### 3.1. Data

We use West Nile Virus Prediction data set curated by Kaggle for our experiments. This data set consists of data from the Chicago Department of Public Health (CHPH) [9]. Every week, the CDPH tests various mosquito traps located around the city for the presence of West Nile Virus (WNV). Each observation in the trap testing data set contains the location of the mosquito trap, the trap ID, the mosquito species captured, the number of mosquitoes, the date/time the trap was tested, and a ground truth label indicating whether a mosquito in the trap tested positive for WNV.

Instead of only using this data set, we also decided to use weather data provided by the National Oceanic and Atmospheric Administration (NOAA) [4]. We were inspired by the literature review, and specifically the North Dakota study about WNV. The weather data includes readings from two weather stations at the Chicago O'Hare International Airport and Chicago Midway International Airport. This weather data includes information on surrounding temperature, precipitation, dew point, pressure, etc.

As a result, we had data from around 100 traps throughout Chicago from 2007 to 2014, including weather data for those days. There were about 10,000 total observations to train and test on; however, our data was unbalanced. 95% of the data points show no presence of WNV, meaning that only 5% of data points are positive for WNV.

#### 3.1.1 Pre-processing

We processed the data in many ways in order to give our model the best chance of success. As we had weather data from two stations, we ensured that we were using data from the closest weather station for each trap. Additionally, we normalized all of the variables by subtracting by mean and dividing by standard deviation. This step was particularly important for the latitude and longitude as although the differences in the data can be extremely small, they actually could translate to differences in city blocks. After that, we also separated the species column in the mosquito trap data into separate Boolean features in order to better see relationships between specific species and other predictors.

#### 3.1.2 Correlation Analysis

After the pre-processing, we wanted to create a correlation matrix for our predictors in order to gain an understanding of their pairwise correlation.

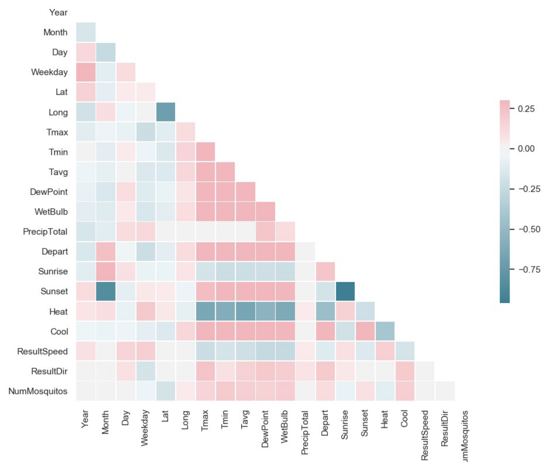


Figure 1: Correlation Map

Most of our predictors were uncorrelated; however, there were a few pairwise relationships that indicated quite high levels of correlation. At first, it seemed surprising that latitude and longitude were correlated. However, we soon realized that this was because the latitudes and longitudes were fixed on the mosquito traps. Thus, the locations were not random, and it would have been possible to predict the latitude from the longitude - or vice versa. Others included the correlation between sunset and sunrise (which is fairly intuitive), and the correlation between heat and the various representations of temperature (Dew Point Temperature, Wet-Bulb Temperature, etc.).

After removing correlated variables like the various temperature readings, we were done processing the data.

### 3.2. Models

After analyzing and processing our data, we fit various models to the mosquito and weather data.

#### 3.2.1 Baseline: Logistic Regression

For our baseline, we decided to perform simple logistic regression using the GPS coordinates of the mosquito traps, month and day, and multiple weather statistics as features [11]. We can say that for a trap at a particular location, the linear classifier assesses if the season/time of the year and weather conditions are conducive for mosquitoes carrying WNV.

Commonly used to model the probability of a binary output, LR was used in our case to predict whether or not WNV would be present. Specifically, the logistic function is used to model the probability - shown below with arbitrary X inputs and beta coefficients:

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

Using thresholding, we can designate a minimum probability needed to predict that Y=1. In order to make these predictions with the beta variables, LR learns the beta coefficients from the training set.

#### 3.2.2 Decision Tree

Approaching the problem by prioritizing interpretability and insight-derivation, we also decided to use decision trees [6]. They model sequential, hierarchical decisions by splitting the predictor space into distinct and non-overlapping regions and using the most commonly occurring class as predictions. The quality of split is evaluated by classification error rate, which is a function of the number of times a prediction is made wrong.

#### 3.2.3 Random Forest

We also used Random Forest (RF) [2]. RF is an example of an ensemble learning method, which means that it improves its performance by using many learning algorithms. The algorithm works by first creating many decision trees during training time. After determining the mode of the classes of each trees , it outputs the class (WNV present or not in our case). RF is known for fixing a common issue of decision trees - namely, overfitting to the training set.

#### 3.2.4 Support Vector Machines

Additionally, we also experimented with support vector machines [10]. Formally, they construct one or more hyperplanes (flat affine subspace of dimension p-1) which can be used for prediction. The general separation to aim for is a hyperplane that is able to separate the training points from each other by classification; however, the SVM allows non-linear class boundaries by using kernels which are linear, polynomial, and radial.

#### 3.2.5 Neural Network

Finally, we tried neural networks [5]. They are defined as a system of interconnected, layered elements (nodes) known as neurons that process information using state responses to inputs. Values introduced in the input layer are distributed through to the hidden layers present in the network, influenced by a set of weights, biases, and activation function (sigmoid, ReLu, etc.). Eventually, after the complete 'forward pass', the final hidden layer is connected to the output layer which has a single neuron in our case (WNV present or not).

### 3.3. Evaluation Metrics

For each of the models, we record AUC, precision, recall, and F1-score [8]. AUC is the area under the ROC curve, which is a line showing the performance of a classification model. Precision is defined as $\frac{truepositives}{truepositives+falsepositives}$ where true positives are accurate positive predictions and false positives are actual negative results that are predicted as positive; it measures how often a positive prediction is actually positive. Recall is defined as $\frac{truepositives}{truepositives+falsenegatives}$ where false negatives are actual positive results that are predicted as negative; it measures how often positive results are predicted correctly as positive. Finally, the F-1 score is the harmonic mean of the precision and recall, defined as: $\frac{2*precision*recall}{precision+recall}$.

As the stakes are so high in our problem - we don't want to miss a true positive, we decided to favor recall to precision when evaluating the models.

## 4. Results

### 4.1. Experimental Setup

After manual hyperparameter tuning, we found best results with 3-layer Fully Connected neural network with dropout in all layers. The neural network was implemented using Keras. We used scikit-learn for Logistic Regression, Decision Tree, Random Forest, and SVM models. Additional engineering was for data parsing, preprocessing, and augmentation. We provide a link to our CodaLab worksheet for reproducibility:

https://worksheets.codalab.org/worksheets/0x65b605b88c0d471eaec8355d0134617b

### 4.2. Experiments Addressing Class Imbalance

As mentioned in Section 3.1, our dataset has extreme class imbalance. 95% of observations belong to the class $WnvPresent = 0$, and there are only 5% of observations where $WnvPresent = 1$. To address this, we experimented with data augmentation and weighted loss functions discussed in section 3.3.

It is common practice when using images as input data to apply various augmentations such as random cropping, flipping, adding noise, etc. These are successful since oftentimes the original image is still easily perceptible despite the additional transforms. However, in our case we have much smaller feature vectors with both continuous and categorical variables. When exploring literature on data augmentation in this space, we were unable to find a clear, reliable strategy for augmenting mixed continuous and categorical feature vectors without running the risk of changing the appropriate class assignment.

The method we implemented takes advantage of the $NumMosquitos$ feature. This feature indicates the number of mosquitoes that were found in the trap when it was checked. This feature appeared to be uncorrelated with any other features. However, we believe that if a trap tests positive for WNV, it is likely many other mosquitos in the same trap have WNV as well. We decided to augment the data by assuming every other mosquito in the trap also is carrying WNV or is not carrying WNV. This assumption is reflected in our augmentation: we duplicate an example such that the number of copies in the training set is equal to $NumMosquitos$.

After performing this augmentation, the classes were 90% $WnvPresent = 0$ and 10% $WnvPresent = 1$.

Another strategy we found was to upweight the loss term computed for the positive class in our loss functions for various models. The weights we used were computed from the inverse class frequencies according to the training set. For example, weighted binary cross-entropy (BCE) is computed as follows:

$$a[labels * -log(\hat{p}] + \beta[(1 - labels) * -log(1 - \hat{p}]$$

This weighted loss function strategy was found to be a very effective approach to addressing our class imbalance issue.

For the sake of simplicity we only present results comparing these techniques when applied to the neural network, and found performance differences were consistent with other models as well.

| | AUC | Precision | Recall | F1 |
|---|---|---|---|---|
| Plain NN | 0.854 | 0.30 | 0.05 | 0.09 |
| Data Augmentation | 0.693 | 0.20 | 0.16 | 0.17 |
| Weighted BCE | **0.851** | 0.12 | **0.90** | 0.21 |
| Data Aug + Weighted BCE | 0.700 | 0.17 | 0.28 | 0.21 |

Table 1: Performance of different solutions to large class imbalance. Metrics reported on the validation set for the WnvPresent=1 class only.

According to 1, the best strategy for addressing the class imbalance problem was using the weighted loss function, achieving the best AUC and recall scores. We follow this approach in our experiments below.

We also tried using a smaller subset of $WnvPresent = 0$ examples, but performance was either the same or worse than applying our data augmentation method.

### 4.3. Experiments with Different Models

The Neural Network achieved the highest AUC and Recall compared to the three other methods. We attribute this to the complexity of the neural network compared to the other methods we implemented.

| | AUC | Precision | Recall | F1 |
|---|---|---|---|---|
| LR (Baseline) | 0.809 | 0.13 | 0.78 | 0.23 |
| RF | 0.809 | 0.33 | 0.33 | 0.33 |
| SVM | 0.835 | 0.14 | 0.78 | 0.24 |
| NN | **0.851** | 0.12 | **0.90** | 0.21 |

Table 2: Performance of different machine learning models. Metrics reported on the validation set for the WnvPresent=1 class only.

## 5. Analysis

### 5.1. Interpretation of Results

By using the respective weighted loss function for each model in our experiments, we managed to achieve decent AUC and Recall. However, we see that precision is relatively low across all our methods. Such results indicate that using the weighted loss functions causes the models start to over-predict the positive class. This speaks to the classical precision/recall tradeoff experienced in machine learning. As discussed, we favor recall over precision due to the relative infrequency of WNV occurrences. However, we believe that additional data could close this precision-recall gap. Unfortunately, when looking to access CDPH data for more recent years outside this Kaggle dataset, the location data of traps is obfuscated and requires additional permissions for use from the City of Chicago.

### 5.2. Feature Importance with L1 Penalty

To understand which features our baseline model found to be strong indicators of WNV, we looked at the feature weights from the logistic regression model found when using the L1 penalty. When using L1, the weights for a particular feature can become 0, indicating 0 importance. We see that WetBulb, Month, and DewPoint seem to be the top 3 important features. WetBulb is a measure of temperature measured using a thermometer covered in damp cloth, so this indicates the combination of temperature and humidity is an important predictor for WNV. Dew point is the temperature at which air must be cooled to be saturated with water vapor, again another measure of an interation between humidity and temperature. Subsequently, Month indicates that seasonality is important to the model – summer months are likely to have more WNV occurrences than winter months.

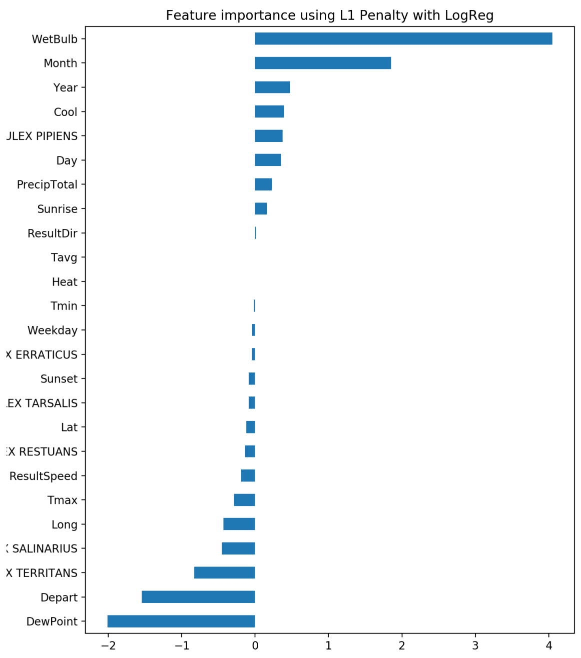


Figure 2: Histogram comparing the importance of different features to the Logistic Regression model found when using the L1 Regularization penalty.

## 5.3. Ablation Study with NN

In the ablation study, we tried removing features such as latitude and longitude, species, weather, and date. We found that all the variables were extremely important neural network because performance drops all around for each of the experiments. As a result, we did not remove any features from the neural network.

## 6. Conclusion

According to prior results and analysis, the neural network proved to be the best predictor for the WNV in Chicago. Although its AUC score is not great, we valued recall over any other evaluation metric as we wanted to make sure that positive cases were found.

We began with the problem of having to make same-day predictions for the presence of WNV in Chicago, presenting with mosquito trap and weather data. After that, we decided to pre-process the data in order to take take of issues regarding normalization, Boolean features, and correlation between variables. Subsequently, we began the process of fitting a number of models, including logistic regression (our baseline), decision trees, random forests, support vector machines, and neural networks. We made variance design decisions for each model, ranging from using the L1 Lasso penalty for logistic regression to using an unbalanced loss function from the neural net that weighted positive results heavier. Our results showed that the neural network had the best AUC and recall, which was enough to make it our best model for the problem. Within the neural network, we also carried out an ablation study that helped to improve our model even further.

| | AUC | Precision | Recall | F1 |
|---|---|---|---|---|
| All Features | 0.85 | 0.14 | 0.84 | 0.24 |
| Without Lat/Long | 0.84 | 0.10 | 0.93 | 0.17 |
| Without Species | 0.84 | 0.11 | 0.95 | 0.19 |
| Without Weather | 0.84 | 0.12 | 0.90 | 0.21 |
| Without Date | 0.85 | 0.12 | 0.91 | 0.21 |

Table 3: Ablation study focusing on removing location, species, weather, and date related features. This was done using the NN with weighted BCE.

In terms of future work, we believe that further improvements can be made on our model with the introduction of new data involving transportation, how often and what locations the city was spraying for mosquitoes, and fine-grained weather that is able to separate data points better. In terms of expanding the scope of the project, we also would consider implementing a convolution neural network with the task of identifying mosquitoes that are predisposed to have WNV (as previously mentioned). It is the next logical step in fighting WNV, and thus would be an important consideration as future work.

## 6.1. Acknowledgements

We would like to acknowledge Susanna Maria Baby, who has provided helpful guidance throughout this process. Finally, we also thank Percy Liang and Dorsa Sadigh and the rest of the teaching staff.

## References

[1] M. Akhtar, M. U. Kraemer, and L. M. Gardner. A dynamic neural network model for predicting risk of zika in real-time. *bioRxiv*, page 466581, 2019. 1

[2] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 3

[3] M. Campion, C. Bina, M. Pozniak, T. Hanson, J. Vaughan, J. Mehus, S. Hanson, L. Cronquist, M. Feist, P. Ranganathan, et al. Predicting west nile virus (wnv) occurrences in north dakota using data mining techniques. In *2016 Future Technologies Conference (FTC)*, pages 310–317. IEEE, 2016. 1

[4] Z. Guo, N. H. Wilson, and A. Rahbee. Impact of weather on transit ridership in chicago, illinois. *Transportation Research Record*, 2034(1):3–10, 2007. 2

[5] S. Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994. 3

[6] D. M. Magerman. Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 276–283. Association for Computational Linguistics, 1995. 3

[7] D. Motta, A. Á. B. Santos, I. Winkler, B. A. S. Machado, D. A. D. I. Pereira, A. M. Cavalcanti, E. O. L. Fonseca, F. Kirchner, and R. Badaró. Application of convolutional neural networks for classification of adult mosquitoes in the field. *PloS one*, 14(1):e0210829, 2019. 1

[8] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011. 3

[9] M. O. Ruiz, E. D. Walker, E. S. Foster, L. D. Haramis, and U. D. Kitron. Association of west nile virus illness and urban landscapes in chicago and detroit. *International Journal of Health Geographics*, 6(1):10, 2007. 2

[10] J. A. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999. 3

[11] R. E. Wright. Logistic regression. 1995. 2

## 7. Appendix

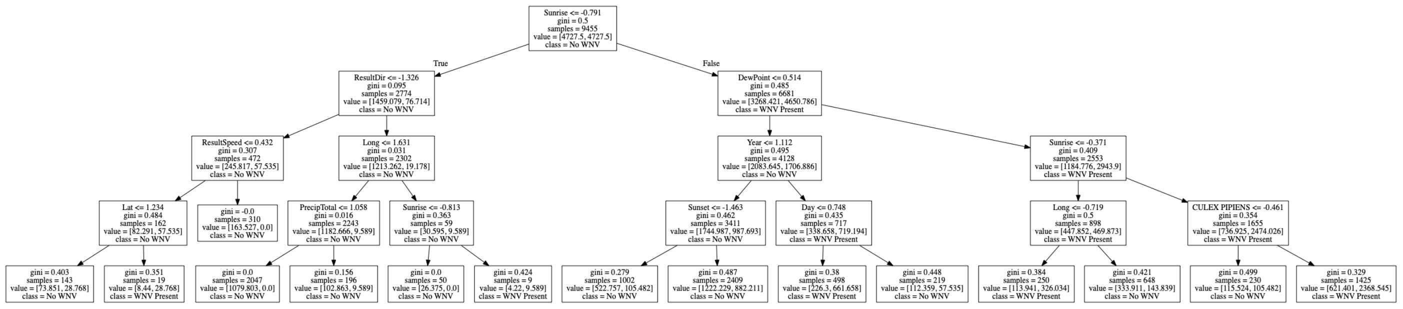
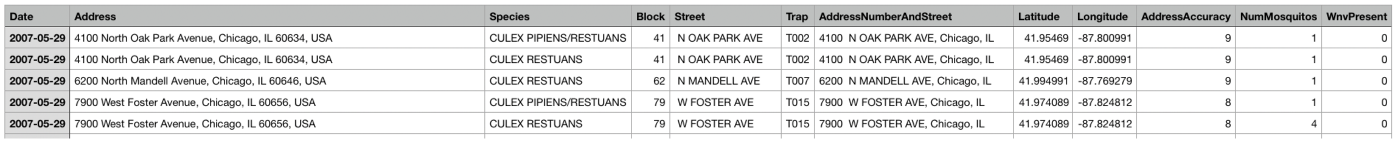
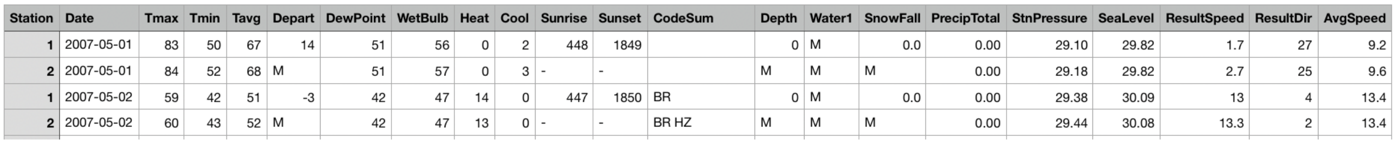


Figure 3: Simple Decision Tree with max depth = 5.



| Date | Address | Species | Block | Street | Trap | AddressNumberAndStreet | Latitude | Longitude | AddressAccuracy | NumMosquitos | WnvPresent |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2007-05-29 | 4100 North Oak Park Avenue, Chicago, IL 60634, USA | CULEX PIPIENS/RESTUANS | 41 | N OAK PARK AVE | T002 | 4100 N OAK PARK AVE, Chicago, IL | 41.95469 | -87.800991 | 9 | 1 | 0 |
| 2007-05-29 | 4100 North Oak Park Avenue, Chicago, IL 60634, USA | CULEX RESTUANS | 41 | N OAK PARK AVE | T002 | 4100 N OAK PARK AVE, Chicago, IL | 41.95469 | -87.800991 | 9 | 1 | 0 |
| 2007-05-29 | 6200 North Mandell Avenue, Chicago, IL 60646, USA | CULEX RESTUANS | 62 | N MANDELL AVE | T007 | 6200 N MANDELL AVE, Chicago, IL | 41.994991 | -87.769279 | 9 | 1 | 0 |
| 2007-05-29 | 7900 West Foster Avenue, Chicago, IL 60656, USA | CULEX PIPIENS/RESTUANS | 79 | W FOSTER AVE | T015 | 7900 W FOSTER AVE, Chicago, IL | 41.974089 | -87.824812 | 8 | 1 | 0 |
| 2007-05-29 | 7900 West Foster Avenue, Chicago, IL 60656, USA | CULEX RESTUANS | 79 | W FOSTER AVE | T015 | 7900 W FOSTER AVE, Chicago, IL | 41.974089 | -87.824812 | 8 | 4 | 0 |

Figure 4: Example of processed mosquito trap data



| Station | Date | Tmax | Tmin | Tavg | Depart | DewPoint | WetBulb | Heat | Cool | Sunrise | Sunset | CodeSum | Depth | Water1 | SnowFall | PrecipTotal | StnPressure | SeaLevel | ResultSpeed | ResultDir | AvgSpeed |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2007-05-01 | 83 | 50 | 67 | 14 | 51 | 56 | 0 | 2 | 448 | 1849 | | 0 | M | 0.0 | 0.00 | 29.10 | 29.82 | 1.7 | 27 | 9.2 |
| 2 | 2007-05-01 | 84 | 52 | 68 | M | 51 | 57 | 0 | 3 | - | - | | M | M | M | 0.00 | 29.18 | 29.82 | 2.7 | 25 | 9.6 |
| 1 | 2007-05-02 | 59 | 42 | 51 | -3 | 42 | 47 | 14 | 0 | 447 | 1850 | BR | 0 | M | 0.0 | 0.00 | 29.38 | 30.09 | 13 | 4 | 13.4 |
| 2 | 2007-05-02 | 60 | 43 | 52 | M | 42 | 47 | 13 | 0 | - | - | BR HZ | M | M | M | 0.00 | 29.44 | 30.08 | 13.3 | 2 | 13.4 |

Figure 5: Example of processed Chicago weather data

### 7.1. Links to Code and Codalab

https://github.com/cameroncruz/cs221-virus-prediction