Capstone:

Classifying Mosquitos Carrying the West Nile Virus in Chicago

Christopher Broll

December 3, 2018

West Nile virus is most commonly spread to humans through infected mosquitos. Around 20% of people who become infected with the virus develop symptoms ranging from a persistent fever, to serious neurological illnesses that can result in death.

In 2002, the first human cases of West Nile virus were reported in Chicago. By 2004 the City of Chicago and the Chicago Department of Public Health (CDPH) had established a comprehensive surveillance and control program that is still in effect today.

Every week from late spring through the fall, mosquitos in traps across the city are tested for the virus. The results of these tests influence when and where the city will spray airborne pesticides to control adult mosquito populations.

Given weather, location, testing, and spraying data, this competition asks you to predict when and where different species of mosquitos will test positive for West Nile virus. A more accurate method of predicting outbreaks of West Nile virus in mosquitos will help the City of Chicago and CPHD more efficiently and effectively allocate resources towards preventing transmission of this potentially deadly virus.

We've jump-started your analysis with some visualizations and starter code in R and Python on Kaggle Scripts. No data download or local environment setup needed!

Acknowledgements

This competition is sponsored by the Robert Wood Johnson Foundation. Data is provided by the Chicago Department of Public Health.

Mosquitoes Trapped

135k

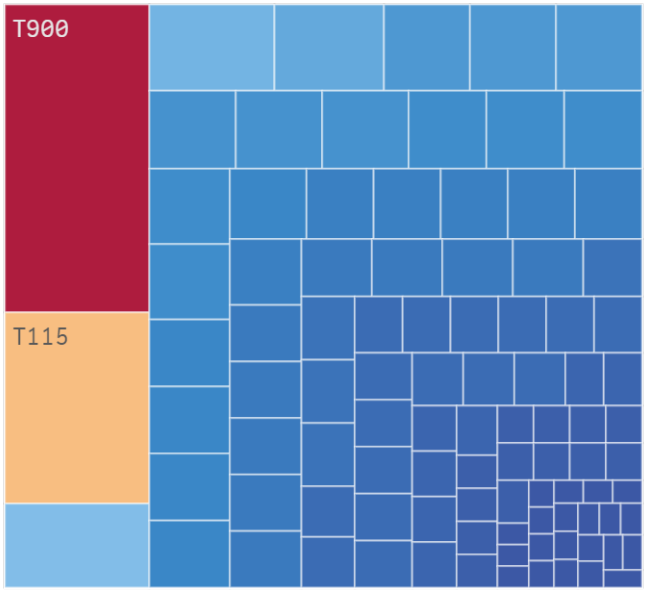
WNV Present

551

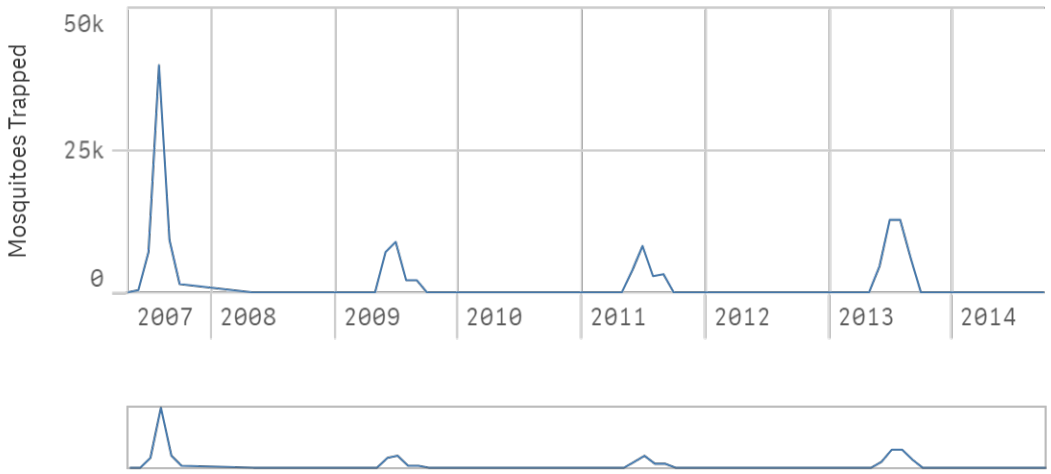
Mosquitoes Carrying

5.24%

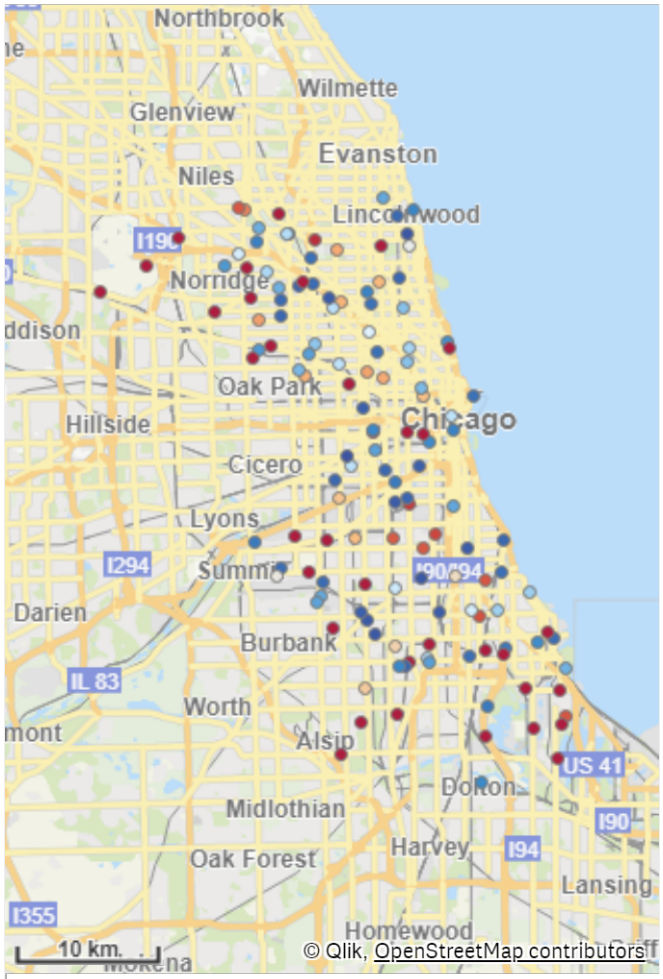
Traps *



** The data set contains negative or zero values that can...*



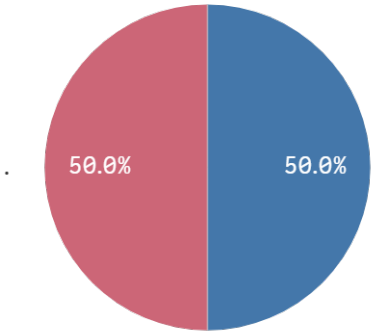
Location of Traps



train.Longitude_train.Latitude
Point layer

Num Mosquitos 0 1k

Station



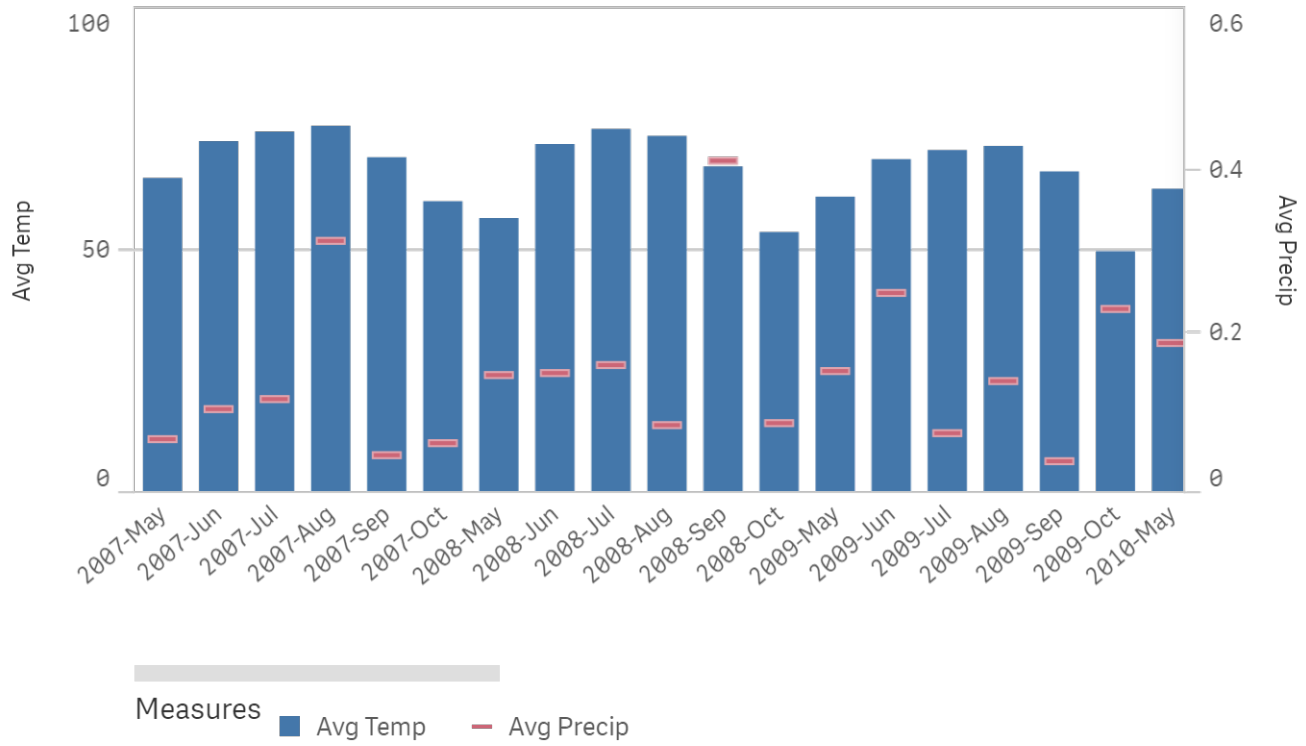
Tmax

76.17

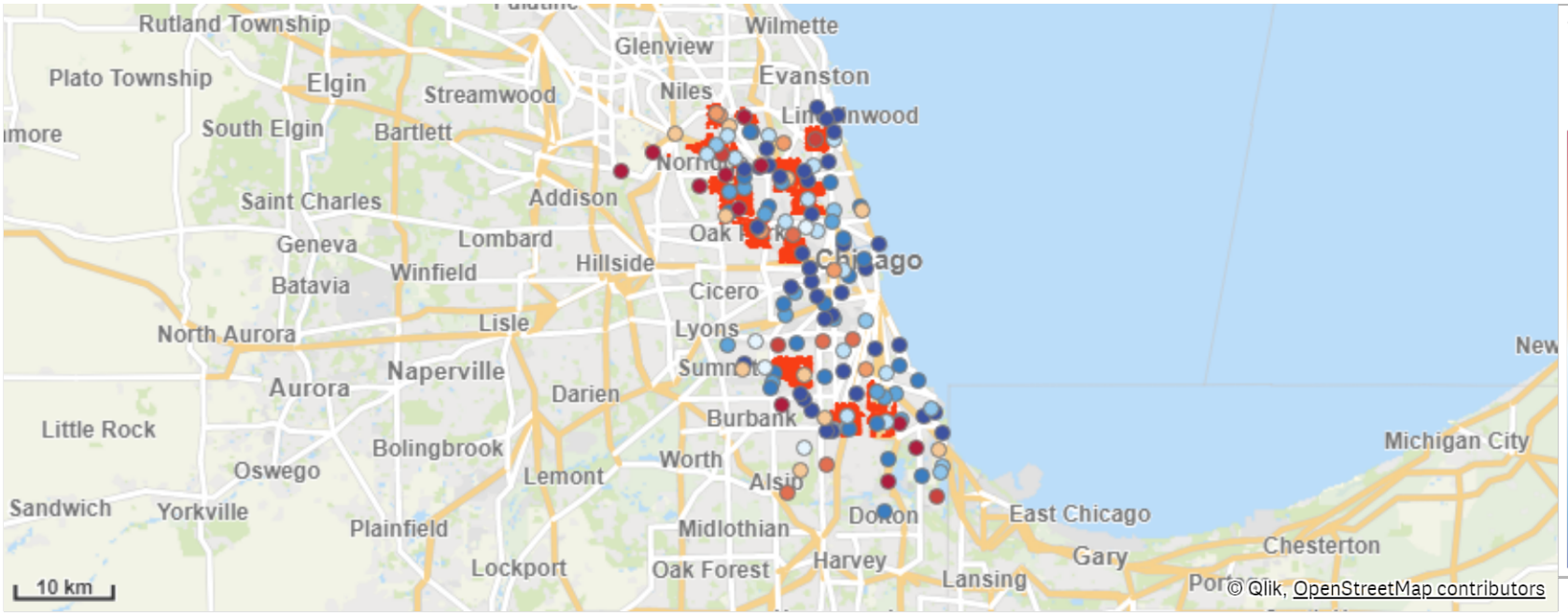
Tmin

57.81

Temperature vs Precipitation



Date	Station	Tavg	Tmax	Tmin	CodeSum	StnPr...	WetB...	AvgSp...
2007-05-01	1	67	83	50		29.10	56	9.2
2007-05-01	2	68	84	52		29.18	57	9.6
2007-05-02	1	51	59	42	BR	29.38	47	13.4
2007-05-02	2	52	60	43	BR HZ	29.44	47	13.4
2007-05-03	1	56	66	46		29.39	48	11.9



Trap
Point layer

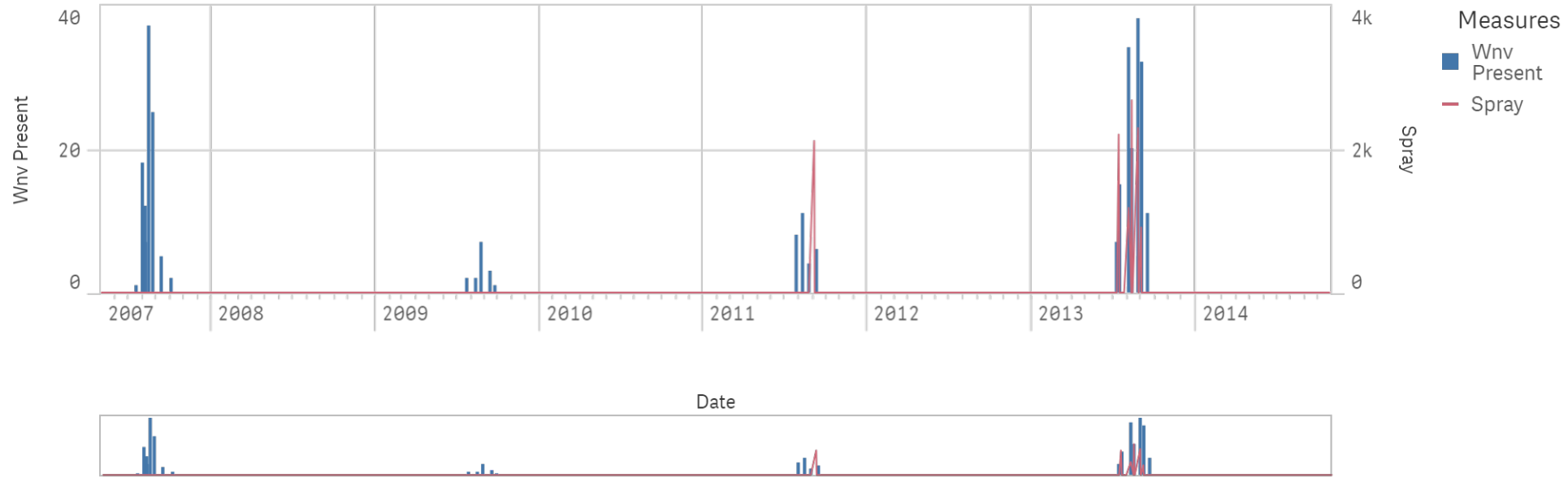
Count of Wnv

10

0

Spray
Point layer

** Currently showing a limited data set.*



Datasets: train.csv, weather.csv, spray.csv

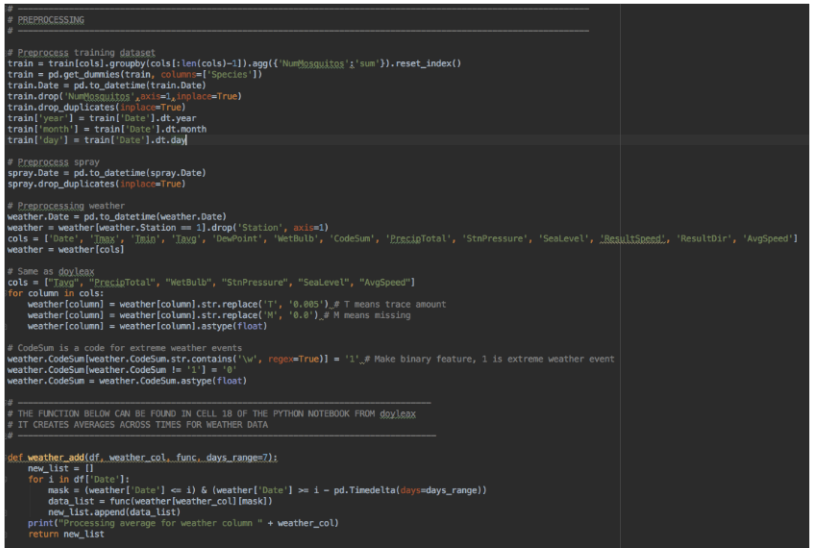
Impute values: T: 0.005, M: 0.0, '\w': 1

weather_add: averaging features in weather.csv across multiple days

Not Included: spray.csv 2011 and 2013

Random Forest: feature selection via feature importance

Standardization: StandardScaler() from scikit learn



```
# ------------------------------------------------------------------------------
# PREPROCESSING
# ------------------------------------------------------------------------------

# Preprocess training dataset
train = train[cols].groupby(cols[:len(cols)-1]).agg({'NumMosquitos':'sum'}).reset_index()
train = pd.get_dummies(train, columns=['Species'])
train.Date = pd.to_datetime(train.Date)
train.drop('NumMosquitos',axis=1,inplace=True)
train.drop_duplicates(inplace=True)
train['year'] = train['Date'].dt.year
train['month'] = train['Date'].dt.month
train['day'] = train['Date'].dt.day

# Preprocess spray
spray.Date = pd.to_datetime(spray.Date)
spray.drop_duplicates(inplace=True)

# Preprocessing weather
weather.Date = pd.to_datetime(weather.Date)
weather = weather[weather.Station == 1].drop('Station', axis=1)
cols = ['Date', 'Tmax', 'Tmin', 'Tavg', 'DewPoint', 'WetBulb', 'CodeSum', 'PrecipTotal', 'StnPressure', 'SeaLevel', 'ResultSpeed', 'ResultDir', 'AvgSpeed']
weather = weather[cols]

# Same as doyleax
cols = ["Tavg", "PrecipTotal", "WetBulb", "StnPressure", "SeaLevel", "AvgSpeed"]
for column in cols:
    weather[column] = weather[column].str.replace('T', '0.005')# T means trace amount
    weather[column] = weather[column].str.replace('M', '0.0')# M means missing
    weather[column] = weather[column].astype(float)

# CodeSum is a code for extreme weather events
weather.CodeSum[weather.CodeSum.str.contains('\w', regex=True)] = '1'# Make binary feature, 1 is extreme weather event
weather.CodeSum[weather.CodeSum != '1'] = '0'
weather.CodeSum = weather.CodeSum.astype(float)

# ------------------------------------------------------------------------------
# THE FUNCTION BELOW CAN BE FOUND IN CELL 18 OF THE PYTHON NOTEBOOK FROM doyleax
# IT CREATES AVERAGES ACROSS TIMES FOR WEATHER DATA
# ------------------------------------------------------------------------------

def weather_add(df, weather_col, func, days_range=7):
    new_list = []
    for i in df['Date']:
        mask = (weather['Date'] <= i) & (weather['Date'] >= i - pd.Timedelta(days=days_range))
        data_list = func(weather[weather_col][mask])
        new_list.append(data_list)
    print("Processing average for weather column " + weather_col)
    return new_list
```

Preprocessing code in PyCharm

Longitude	0.226376
Latitude	0.209275
Tmin_20	0.095229
month	0.071492
Tmax_20	0.063669
DewPoint	0.053605
Tmin_3	0.045496
Tmin	0.032415
Species_CULEX RESTUANS	0.032042
year	0.030785
Tmax_3	0.030325
day	0.028887
Species_CULEX PIPIENS/RESTUANS	0.022610
PrecipTotal	0.018866
Species_CULEX PIPIENS	0.018848
Tmax	0.014096
Species_CULEX TERRITANS	0.005139
Species_CULEX SALINARIUS	0.000720
Species_CULEX TARSALIS	0.000125
Species_CULEX ERRATICUS	0.000000

Feature importance from random forest model

Neural Network: Multilayer perceptron
No. Hidden Layers: 3
Input Size: 8 features
Transfer Functions: ReLU and Tanh

Performance Index: Cross Entropy
Optimizer: Adam
Class Weights: 1 and 9.5
Learning Rate: 1e-3
Num Epochs: 10,000

```
# ----------------------------------------------------------------------------------------------------
# COMPUTATIONAL GRAPH FOR MODEL
# ----------------------------------------------------------------------------------------------------
# MLP model
model = torch.nn.Sequential(
    torch.nn.Linear(8, 7),
    torch.nn.ReLU(),
    torch.nn.Linear(7, 5),
    torch.nn.ReLU(),
    torch.nn.Linear(5, 3),
    torch.nn.ReLU(),
    torch.nn.Linear(3, 2),
    torch.nn.Tanh(),
)

# ----------------------------------------------------------------------------------------------------
# HYPER PARAMETERS AND OPTIMIZATION
# ----------------------------------------------------------------------------------------------------

# Imbalanced dataset where 5% of target are 1's
class_weights = torch.FloatTensor([1.0, 9.5])

# Initialize performance index
performance_index = torch.nn.CrossEntropyLoss(weight=class_weights)

# Initialize learning rate
learning_rate = 1e-3

# Initialize optimizer
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
```

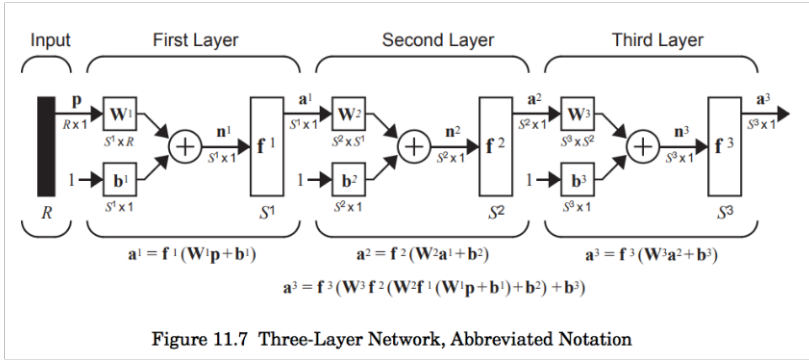
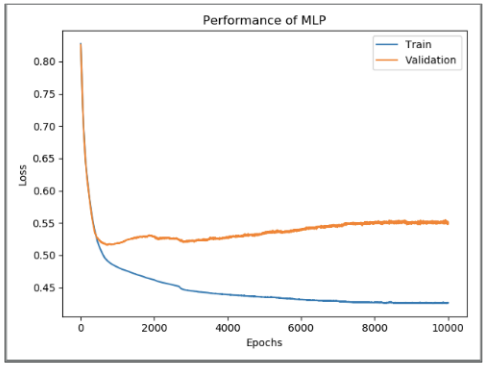
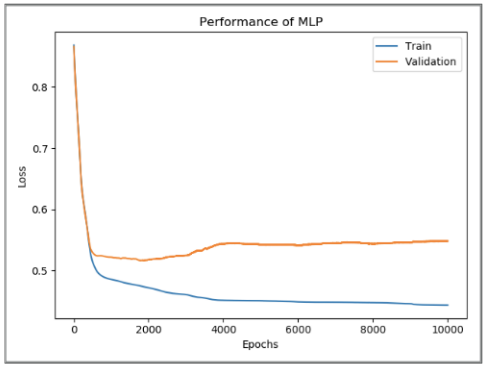


Figure 11.7 Three-Layer Network, Abbreviated Notation

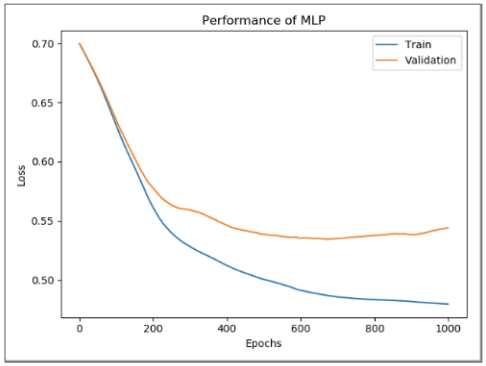
Validation and Overfitting:



No early stoppage or weight decay



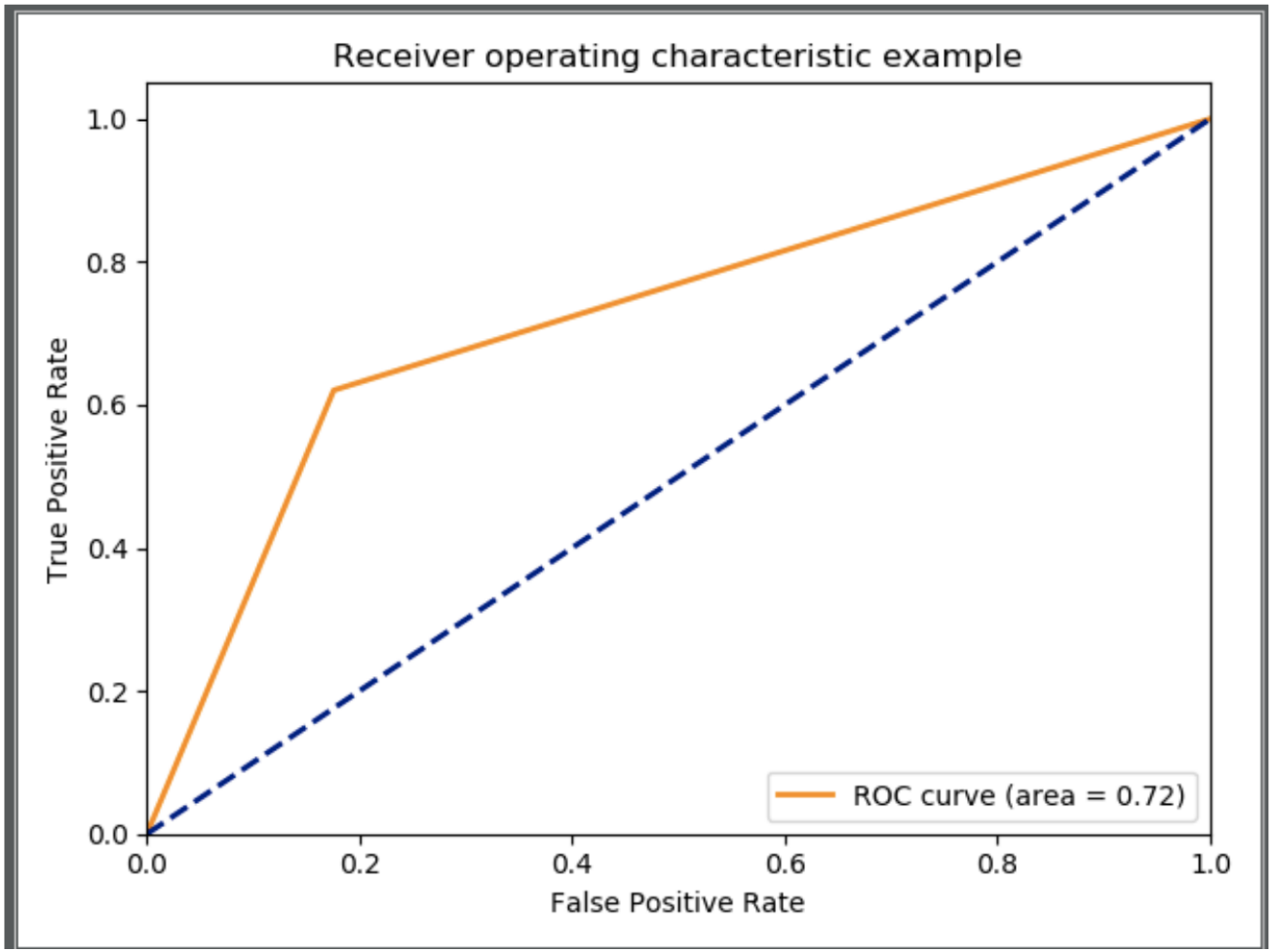
weight_decay = 0.005



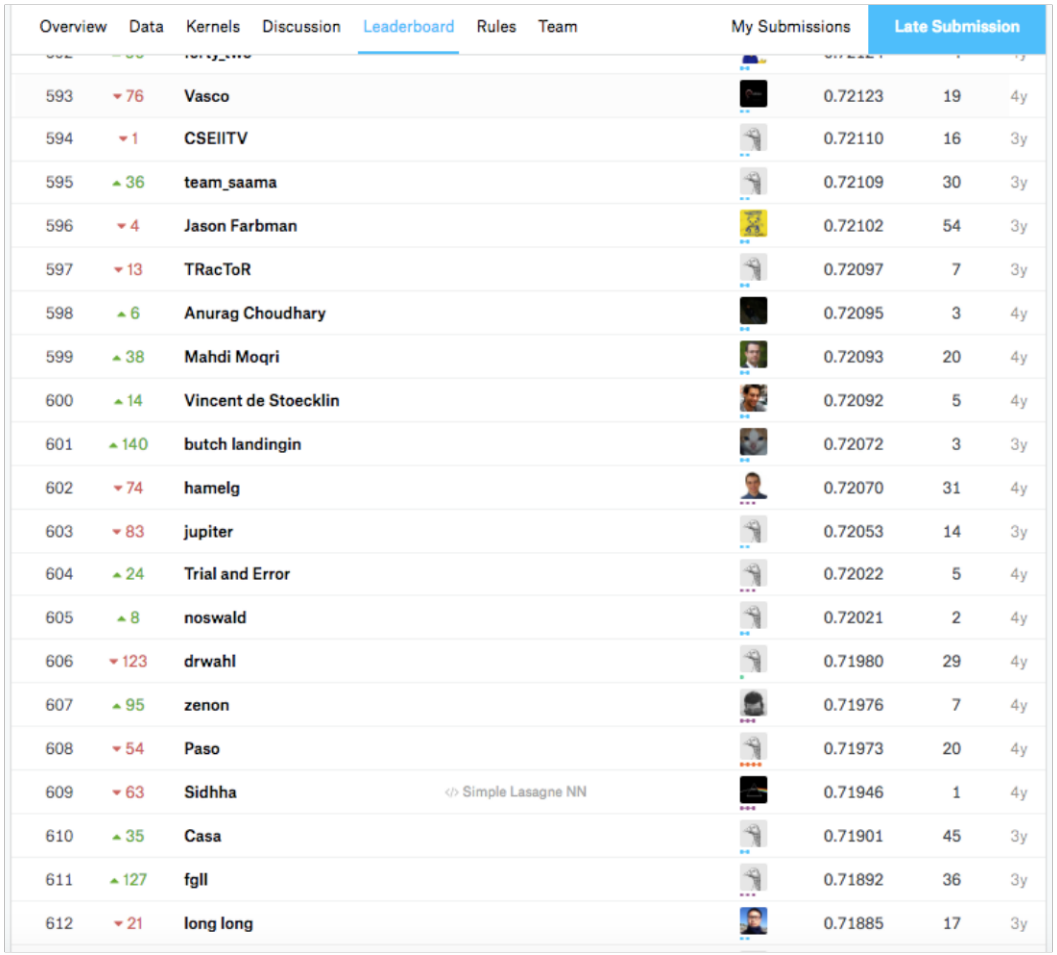
Early stoppage at Epochs = 1000

```
# Split into features and targets
X = df.iloc[:,:-1].values
y = df.iloc[:,-1:].values.ravel()

# Split train, validation and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, stratify=y, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.15, stratify=y_train, random_state=42)
```



Note: Evaluation of the model AUC (area under the ROC), which ranged from 0.68 to 0.72



Overview Data Kernels Discussion Leaderboard Rules Team My Submissions Late Submission

593	▼76	Vasco		0.72123	19	4y
594	▼1	CSEIITV		0.72110	16	3y
595	▲36	team_saama		0.72109	30	3y
596	▼4	Jason Farbman		0.72102	54	3y
597	▼13	TRacToR		0.72097	7	3y
598	▲6	Anurag Choudhary		0.72095	3	4y
599	▲38	Mahdi Moqri		0.72093	20	4y
600	▲14	Vincent de Stoecklin		0.72092	5	4y
601	▲140	butch landingin		0.72072	3	3y
602	▼74	hamelg		0.72070	31	4y
603	▼83	jupiter		0.72053	14	3y
604	▲24	Trial and Error		0.72022	5	4y
605	▲8	noswald		0.72021	2	4y
606	▼123	drwahl		0.71980	29	4y
607	▲95	zenon		0.71976	7	4y
608	▼54	Paso		0.71973	20	4y
609	▼63	Sidhha	</> Simple Lasagne NN	0.71946	1	4y
610	▲35	Casa		0.71901	45	3y
611	▲127	fgll		0.71892	36	3y
612	▼21	long long		0.71885	17	3y

Score: 0.72
Position: Between 605 and 606
Submissions: 1305
Percentile: ~ 50th

Lessons Learned:
- Need nonparametric statistics background, i.e., generalized additive models (GAM) top of leaderboard

- Feature engineering (data mining techniques)

- Domain expertise

- Optimal hyperparameters

- Optimal neurons and layers in a network

- Suitable transfer functions

- Avoid local mins (SGD momentum)