PhishTrap

Optimal Hyper Parameters for the Machine Learning Algorithms

kNN

We used a pipeline of a Variance Threshold Selector, PCA and the kNN classifier.

Variance Threshold Selector

threshold: 0

PCA

pca__n_components: 57
pca__whiten: True,
pca__svd_solver: randomized
pca__tol: 1e-07

kNN

knn__n_neighbors: 5
knn__weights: distance
knn__metric: Manhattan
knn__p: 6
knn__algorithm: kd_tree
knn__leaf_size: 30

Metrics Results:

Accuracy	90.81%
Precision	70.24%
Recall	93.77%
FNR	6.23%

Where $FNR = 1 - Recall$

MLP

We used a pipeline of a Variance Threshold Selector and the MLP classifier

Variance Threshold Selector

threshold: 0

MLP

mlp__hidden_layer_sizes: 2500,
mlp__activation: relu,
mlp__alpha: 0.01,
mlp__learning_rate: adaptive,

mlp__learning_rate_init: 0.1,
mlp__momentum: 0.8,
mlp__solver: sgd

Metrics Results:

Accuracy	93.79%
Precision	78.32%
FNR	4.67%

Random Forest

n_estimators: 450,
criterion: gini,
max_depth: 300,
max_features: 0.15,
min_samples_leaf: 1,
min_samples_split: 2,
oob_score: False,
bootstrap: False,
warm_start: False

Metrics Results:

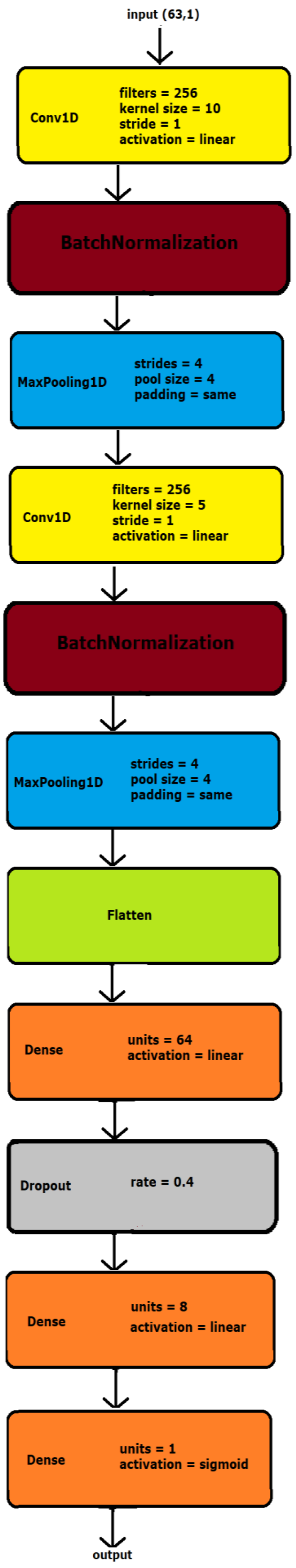
Accuracy	94.90%
Precision	82.05%
FNR	4.67%

Gradient Boosting

n_estimators: 15000,
criterion: mse,
learning_rate: 0.005,
loss: deviance,
max_depth: 60,
max_features: 16,
min_samples_leaf: 3,
min_samples_split:250
subsample: 0.8,
warm_start: True

Metrics Results:

Accuracy	95.77%
Precision	84.67%
FNR	3.73%



CNN

For the CNN we tried using 2 Convolutional layers followed by a MaxPooling layer. We added a BatchNormalization layer in between because it assisted with the internal covariate shift problem.

On the Fully Connected layers we added a Dropout layer to reduce the overfitting.

The image on the left shows the architecture of the CNN and the hyperparameters of each layer.

Metrics Results:

Accuracy	88.40%
Precision	63.75%
FNR	2.60%

Voting Scheme

As described in the paper Kuncheva, L.I., Rodríguez, J.J. “A weighted voting framework for classifiers ensembles”. *Knowl Inf Syst* **38,** 259–275 (2014), for the weighted majority combiner. This combiner is proven to be optimal on imbalanced datasets, after experimentation.

Metrics Results:

Accuracy	94.60%
Precision	80.15%
FNR	2.97%