Predicting the Direction of Stock Market Price Using Tree-Based Classifiers: A Machine Learning Approach: *Supplementary File for On-line Publication Only*

1. Key Definitions

In this appendix, we introduce some key terms which have been used frequently through the paper. A basic understanding of the key concepts will bolster the reader's understanding of the methods used for predictive analysis.

1. **Data Structure**: In computer science, a *data structure* refers to the means of organizing and storing data for analysis and manipulation (Horowitz, Sahni & Anderson-Freed, 1992). Knowledge of data structures is fundamental for a computer scientist to judge how data should be stored for a particular application or task. Some popular data structures include stacks, queues, linked-lists, and trees.

2. **Node**: A node in a data structure is essentially an instance of that structure that contains data (Horowitz, Sahni & Anderson-Freed, 1992). A data structure is made of multiple nodes. It is the way in which nodes are connected in a data structure that defines it's characteristics.

3. **Tree Data Structure**: In a tree data structure, every node is the child of a single node (with the exception of the root node) and can have one or more children nodes (Horowitz, Sahni & Anderson-Freed, 1992). The name is inspired from how this data structure looks as subsequent levels of nodes *branch* or *spread out* like in a tree (Figure 1; notice how the subsequent levels of nodes branch out).

4. **Root Node**: In a tree data structure (henceforth referred to as just *tree*), the first node from which other nodes branch out is called the *root node* (Horowitz, Sahni & Anderson-Freed, 1992).

5. **Child Node**: A node in a tree can have one or more subordinate nodes or hierarchically lower nodes. These are called as child nodes of the respective node. In any tree, only the root node is not a child node.

6. **Level**: The level of a set of nodes in a tree is an index of how far away a node is from the root node. The level of the root node is 0. The level of the immediate children of the root node is 1. The level of the children nodes of the nodes at level 1 is 2, and so on.

7. **Parent Node**: If a node exists in a tree, it must be the subordinate of another node (with the exception of the root node) called it's parent node. A parent node of a child is the node that is immediately higher in hierarchy to the respective child node. The level of a node's parent is always one less than the level of the respective child.

8. **Leaf Node**: A leaf node is a node that does not have any children. All finite trees end with leaf nodes. In a classification tree, all leaf nodes need to be *pure*, i.e., they should have entities belonging to any one class only (exceptions to this may be incorporated by *pruning* a tree to prevent overfitting; pruning, in a general sense, is the removal of some leaf nodes with very few entities in them, before or after the tree has been built, so as to prevent overfitting) (Horowitz, Sahni & Anderson-Freed, 1992; Breiman, 2001).

9. **Probability Distribution**: $X = (X_1, ..., X_d)$ is an array of random variables defined on a probability space called as random vectors. The joint distribution of $X_1, ..., X_d$ is a measure on μ on R^d, $\mu(A) = P(X \in A)$, $A \in R^d$ where $d = 1, ..., m$. For example, Let $x = (x_i, ..., x_d)$ be an array of data points. Each feature x_i is defined as a random variable with some distribution. Then the random vector X has joint distribution identical to the data points, x.

10. **Classification Tree**: We define a classification tree (Quinlan, 1992) where each node is endowed with a binary decision: *whether* $x_i \leq k$ *or not*; where x_i is a feature in the data set, and k is some threshold. The topmost node in the classification tree contains all the data points and the set of data is subdivided among the children of each node as defined by the classification. The process of subdivision continues until pure leaf nodes are achieved. Each node is characterized by the feature x_i and threshold k chosen in such a way that minimizes diversity among the children nodes. This is often referred to as Gini impurity.

11. **Random Forest**: Let us represent $h_k(x) = h(x|\theta_k)$ implying decision tree k leading to a classifier $h_k(x)$. Thus, a random forest is a classifier based on a family of classifiers $h(x|\theta_1), ..., h(x|\theta_k)$, which is an ensemble of classification trees with model parameters θ_k randomly chosen from model random vector θ. Each classifier, $h_k(x) = h(x|\theta_k)$ is a predictor of the number of training samples. $y = \pm 1$ is the outcome associated with input data, x for the final classification function, $f(x)$, which can result by a majority voting mechanism of the individual classifiers (Breiman, 2001).

12. **Linear Separability**: Before feeding the training data to the random forest classifier, the two classes of data are tested for linear separability by finding their convex hulls. Linear Separability is a property of two sets of data points where the two sets are said to be linearly separable if there exists a hyperplane such that all the points in one set lies on one side of the hyperplane and all the points in other set lies on the other side of the hyperplane.

    Mathematically, two sets of points X_0 and X_1 in n dimensional Euclidean space are said to be linearly separable if there exists an n dimensional normal vector W of a hyperplane and a scalar k, such that every point $x \in X_0$ gives $W^T x > k$ and every point $x \in X_1$ gives $W^T x < k$. Two sets can be checked for linearly separability by constructing their convex hulls.

13. **ROC**: In statistics, a *receiver operating characteristic* (ROC) curve is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, recall or probability of detection[1] in machine learning. The false-positive rate is also known as the fall-out or probability of false alarm[1] and can be calculated as (1− specificity). The ROC curve is thus the sensitivity as a function of fall-out. In general, if the probability distributions for both detection and false alarm are known, the ROC curve can be generated by plotting the cumulative distribution function (area under the probability distribution from -∞ to the discrimination threshold) of the detection probability in the y-axis versus the cumulative distribution function of the false-alarm probability in x-axis.

    There are four possible outcomes from a binary classifier. If the outcome from a prediction is positive (p) and the actual value is also p, then it is called a *true positive* (TP); however if the actual value is negative (n) then it is said to be a false positive (FP). Conversely, a true negative (TN) has occurred when both the prediction outcome and the actual value are n, and false negative (FN) is when the prediction outcome is n while the actual value is p. To draw an ROC curve, only the true positive rate (TPR) and false positive rate (FPR) are needed (as functions of some classifier parameter). The TPR defines how many correct positive results occur among all positive samples available during the test. FPR, on the other hand, defines how many incorrect positive results occur among all negative samples available during the test.

    An ROC space is defined by FPR and TPR as x and y axes respectively, which depicts relative trade-offs

between true positive (benefits) and false positive (costs). Since TPR is equivalent to sensitivity and FPR is equal to 1− specificity, the ROC graph is sometimes called the sensitivity vs (1− specificity) plot. Each prediction result or instance of a confusion matrix represents one point in the ROC space. The diagonal divides the ROC space. Points above the diagonal represent good classification results (better than random), points below the line represent poor results (worse than random).

14. **AUC**: The percentage area under the curve (AUC) value is the area covered by the ROC curve in the ROC space. It gives us a quantitative measure of the performance of a classifier, and a quantitative measure of the ROC at a glance.

15. **Brier Score**: The Brier score is the square of the difference between the predicted value or label of an input object and the real value or label. It is similar to the mean squared error and quantifies how aberrant the predictions of the classifier or regression function is.. For our work on classification, the Brier score is given as:

$$BS = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{1}$$

where y_i is the true class label of the i^{th} sample and $\hat{y}_i$ is the predicted label of the same sample, and N is the number of test samples.

2. Random Forest

2.1. The Shannon Entropy

Another function in addition to the Gini impurity which can be used to judge the quality of a split is the Shannon Entropy. It measures the disorder in the information content (in this context, it measures how mixed the population in a node is). The entropy in a node N can be calculated as follows:

$$S(N) = -P_1 log(P_1) - P_{-1} log(P_{-1}) \tag{2}$$

where d is number of classes considered and $P(\omega_i)$ is the proportion of the population labeled as i. Entropy is the highest when all the classes are contained in equal proportion in the node. It is the lowest when there is only one class present in a node (when the node is pure).

In most cases, the Gini impurity and the Shannon entropy can be used interchangeably.

2.2. Advantages of CART

Some of the main advantages of using random forests of CART are:

1. CART-based random forests are *non-metric*. This means that there are no inherent assumptions of distributions in data, and neither are there any parameters which need to be tweaked in order to obtain optimal performance.
2. Random Forests are recommended to be used with *bagging*, i.e., test data is sampled with replacement. Bagging ensures that with the increase in the number of tree estimators in a random forest, the chance of error decreases. A simple assertion of this is made (using the binomial distribution stuff) and is proved using Chebyshev's Inequality in (enter section number).
3. CART can handle categorical and continuous values naturally.
4. It is easy to understand and quick to fit, even if the data is cluttered and/or if the problem is inherently large.
5. The accuracy of random forests compares well with traditional methods in ML, such as Naïve Bayes', etc.
6. Random forests are *stable*. A slight change in the input data may affect individual trees, but the characteristics of the forest remains largely unchanged.

2.3. OOB error and Convergence of the Random Forest

Given an ensemble of decision trees $h_1(X), h_2(x), h_3(x), ..., h_k(x)$ as in (Breiman, 2001), we define margin function as:

$$mg(X,Y) = av_k I(h_k(X) = Y) - max_{j \neq Y} av_k I(h_k(X) = j) \tag{3}$$

where X, Y are randomly distributed vectors from which the training set is drawn. Here, $I(\cdot)$ is the indicator function. The generalization error is given by:

$$PE^* = P_{X,Y}(mg(X,Y) < 0) \tag{4}$$

The X and Y subscripts indicate that probability is calculated over X, Y space. In random forests, the k^{th} decision tree $h_k(x)$ can be represented as $h(x, \theta_k)$ where x is the input vector and θ_k is the bootstrapped dataset which is used to train the k^{th} tree. For a sequence of bootstrapped sample sets $\theta_1, \theta_2, ..., \theta_k$ which are generated from the original dataset θ, it is found that PE^* converges to:

$$P_{X,Y}(P_\theta(h(X,\theta) = Y) - max_{j \neq Y} P_\theta(h(X,) = j) < 0) \tag{5}$$

The proof can be found in Appendix I in (Breiman, 2001). To practically prove this theorem with respect to our dataset, the generalization error is estimated using out of bags estimates (Bylander & Hanzlik, 1999). The out of Bag (OOB) error measures the prediction error of Random forests algorithm and other machine learning algorithms which are based on Bootstrap aggregation.

Note: The average margin of the ensemble of classifiers is the extent to which the average vote count for the correct class flag exceeds the count for the next best class flag.

2.4. Random Forest as Ensembles: An Analytical Exploration

As defined earlier, a Random Forest model specifies θ as classification tree marker for $h(X|\theta)$ and a fixed probability distribution for θ for diversity determination in trees is known. The margin function of an RF is:

$$margin_{RF}(x,y) = P_\theta(h(x|\theta) = y) - max_{j \neq y} P_\theta(h(x|\theta) = j) \tag{6}$$

The strength of the forest is defined as the expected value of the margin:

$$s = E_{x,y}(margin_{RF}(x,y)) \tag{7}$$

The generalization error is bounded above by Chebyshev's inequality and is given as:

$$Error = P_{x,y}(margin_{RF}(x,y) < 0) \leq P_{x,y}(|margin_{RF}(x,y) - s| \geq s) \leq \frac{var(margin_{RF}(x,y))}{s^2} \tag{8}$$

Remark: We know that the average margin of the ensemble of classifiers is the extent to which the average vote count for the correct class flag exceeds the count for the next best class flag. The strength of the forest is the expected value of this margin. When the margin function gives a negative value, it means that an error has been made in classification. The generalization error is the probability that the margin is a negative value. Since margin itself is a random variable, Equation 8 shows that it is bounded above by its variance divided by the square of the threshold. As the strength of the forest grows, error in classification decreases.

We present below the Chebyshev's inequality as the inspiration for the error bound.

2.5. Chebyshev's Inequality

Let X be any random variable (not necessarily non-negative) and C > 0. Then,

$$P(|X - E(X)| \geq c) \leq \frac{var(x)}{c^2} \tag{9}$$

It is easy to relate the inequality to the error bound of the Random Forest learner.

2.6. Proof of Chebyshev's Inequality:

We require a few definitions before the formal proof.

A) Indicator Random Variable

$$I(X \geq c) = \begin{cases} 1 & \text{if } X \geq c \\ 0 & \text{Otherwise} \end{cases} \tag{10}$$

B) Measurable space

$$A = \{x \in \Omega | X(x) \geq c\} \tag{11}$$

$$E(X) = \sum_{x \in A} P(x)X(x) = \mu \tag{12}$$

Proof:

Define $A = \{x \in \Omega | X(x) - E(x) \geq c\}$

Thus,

$$\begin{aligned} var(X) &= \sum_{x \in \Omega} P(X = x)(X(x) - E(x))^2 \\ &= \sum_{x \in A} P(X = x)(X(x) - E(x))^2 + \sum_{x \notin A} P(X = x)(X(x) - E(x))^2 \geq 0 \\ &\geq \sum_{x \in A} P(x = x)(X(x) - E(x))^2 \\ &\geq P(X = x)c^2 \quad \text{since,} X(x) - E(x) \geq C; \forall x \in A \\ &= c^2 P(A) = c^2 P(|X - E(X)| \geq c) \\ \Rightarrow \frac{var(X)}{c^2} &\geq P(|X - E(X)| \geq c) \end{aligned} \tag{13}$$

Remark: This means that the probability of the deviation of a data point from its expected value being greater than c, a threshold, is bounded above by the variance of the data points divided by the square of the threshold, c. As c increases, the upper bound decreases which implies the probability of a large deviation of a data point from its expected value is less likely.

2.7. Algorithms

Algorithm 1: GiniGain

input : $(x_i, y_i)_1^n$ is the labeled training data
c_L is the left child node
c_R is the right child node

output: The Gini gain of the current split

$G_N \leftarrow$ Gini impurity of root node;
$G_L \leftarrow$ Gini impurity of left child;
$G_R \leftarrow$ Gini impurity of right child;
$L_n \leftarrow$ number of samples in c_L;
$R_n \leftarrow$ number of samples in c_R;
$P_L \leftarrow L_n/n$;
$P_R \leftarrow R_n/n$;
return $G_N - (P_L \times G_L) - (P_R \times G_R)$;

Algorithm 2: DecisionTree

input : $X = (x_i, y_i)_1^n$ is the labeled training data
l is the current level of the tree
M is the set of features used to grow a tree

output: A tree which is configured to predict the class label of a test sample

---

$l \leftarrow l + 1$;
$C_L \leftarrow$ null;
$C_R \leftarrow$ null;
; `/* ` C_L, C_R `are the left and right children of this node respectively */`
$MaxGain \leftarrow 0$;
; `/* ` $MaxGain$ `stores the value of the maximum possible gain that can be achieved from splitting a node - based on this, the final split is determined */`
for j *in* M **do**
  Sort $(x_i, y_i)_1^n$ in the increasing order of the j^{th} feature
  for $i := 1$ *to* n **do**
    $c_L \leftarrow X[0 : i]$;
    $c_R \leftarrow X[i + 1 : n]$;
    $Gain \leftarrow$ **GiniGain**$((x_i, y_i)_1^n[j], c_L, c_R)$;
    if $Gain > MaxGain$ **then**
      $MaxGain \leftarrow Gain$;
      $C_L, C_R \leftarrow c_L, c_R$;
  end
end
if C_L ***does not satisfy the desired level of purity*** **then**
  DecisionTree(C_L, l, M);
if C_R ***does not satisfy the desired level of purity*** **then**
  DecisionTree(C_R, l, M);
; `/* A node is not pure when it has samples belonging to more than one class. In such a case, it needs to be split in turn. In this way, through successive function calls, the feature space is recursively partitioned. */`

3. eXtreme Gradient Boosting (XGBoost)

3.1. Gradient Boosted Decision Trees: An Analytic Exploration

The tree approximation by aggregating many functions is done by *additive learning*. Each node is hence built sequentially, with each successive approximated function trying to better classify the residuals of the

previous learner. XGBoost employs an additive strategy wherein every subsequent approximated function optimizes an objective function. Hence, a series of functions are built such that the node is ultimately approximated using an aggregate of all the functions and is gradually optimized. The objective function may be represented as:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^t) + \sum_{k=1}^{t} \Omega(f_k) \tag{14}$$

where l is a loss function and Ω is the regularization term, which is used to measure the complexity of the model. Ω is of the form: $\Omega(f) = \gamma T + \lambda \|w\|^2$, thus making use of an L2 regularization.

Additive learning over t iterations for the i^{th} tree happens as follows:

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
\hat{y}_i^{(3)} &= f_1(x_i) + f_2(x_i) + f_3(x_i) = \hat{y}_i^{(2)} + f_3(x_i) \\
&\dots \\
\hat{y}_i^{(t)} &= f_1(x_i) + ... + f_n(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned} \tag{15}$$

And hence, the expression:

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{16}$$

Hence, the objective function may be expanded as:

$$\begin{aligned}
\mathcal{L}^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k) \\
&= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constants
\end{aligned} \tag{17}$$

The constants arise because the regularization till step $t-1$ is considered to be a constant at step t. Hence, Equation 17 represents the loss function in it's general form. This loss function can be approximated using the Taylor Series approximation till the n^{th} order term. The approximation till the 2^{nd} order is considered. The Taylor Series expansion is done as follows.

Let us consider a function $F(u) = l(X(u), Y(u)) = l(X + u\Delta X, Y + u\Delta Y)$. At $u = 1$, the function becomes: $F(1) = l(X + \Delta X, Y + \Delta Y)$. Hence, in the current context, at $u = 1$, the following considerations can be made: $X \equiv y_i$, $\Delta X = 0$, $Y \equiv \hat{y}_i^{(t-1)}$, $\Delta Y \equiv f_t(x_i)$. Applying the chain rule, we can find the n^{th} order derivatives of $F(u)$. The expression for the first order derivative is derived in Equation 18.

$$\begin{aligned}
&\frac{d}{du}\big[F(u)\big] = \frac{\partial}{\partial X}\big[l(X(u), Y(u))\big]\frac{d}{du}(X(u)) + \frac{\partial}{\partial Y}\big[l(X(u), Y(u))\big]\frac{d}{du}(Y(u)) \\
\Rightarrow &\frac{d}{du}\big[F(u)\big] = \frac{\partial}{\partial X}\big[l(X(u), Y(u))\big]\Delta X + \frac{\partial}{\partial Y}\big[l(X(u), Y(u))\big]\Delta Y \\
\Rightarrow &\frac{d}{du}\big[F(u)\big] = \frac{\partial}{\partial X}\big[l(X(u), Y(u))\big] \cdot 0 + \frac{\partial}{\partial Y}\big[l(X(u), Y(u))\big] \cdot f_t(x_i) \\
\Rightarrow &\frac{d}{du}\big[F(u)\big] = \frac{\partial}{\partial \hat{y}_i^{(t-1)}} l(\hat{y}_i, \hat{y}_i^{(t-1)}) \cdot f_t(x_i) \\
&\Rightarrow F'(u) = g_i f_t(x_i)
\end{aligned} \tag{18}$$

where

$$g_i = \frac{\partial}{\partial \hat{y}_i^{(t-1)}} l(\hat{y}_i, \hat{y}_i^{(t-1)})$$

The expression for the second order derivative is derived in Equation 19:

$$\begin{aligned}
&\frac{d^2}{du^2}\big[F(u)\big] = \left\{ \frac{\partial}{\partial X}\Big[\frac{\partial}{\partial Y}\big[l(X(u),Y(u))\big]\Big] + \frac{\partial}{\partial Y}\Big[\frac{\partial}{\partial Y}\big[l(X(u),Y(u))\big]\Big] \right\} \cdot \frac{d}{du}(Y(u)) \\
\Rightarrow &\frac{d^2}{du^2}\big[F(u)\big] = \left\{ \frac{\partial^2}{\partial X \partial Y}\big[l(X(u),Y(u))\big] \cdot \frac{d}{du}(X(u)) + \frac{\partial^2}{\partial Y^2}\big[l(X(u),Y(u))\big] \cdot \frac{d}{du}(Y(u)) \right\} \cdot \frac{d}{du}(Y(u)) \\
\Rightarrow &\frac{d^2}{du^2}\big[F(u)\big] = \left\{ \frac{\partial^2}{\partial X \partial Y}\big[l(X(u),Y(u))\big] \cdot \Delta X + \frac{\partial^2}{\partial Y^2}\big[l(X(u),Y(u))\big] \cdot \Delta Y \right\} \cdot \frac{d}{du}(Y(u)) \\
\Rightarrow &\frac{d^2}{du^2}\big[F(u)\big] = \left\{ \frac{\partial^2}{\partial X \partial Y}\big[l(X(u),Y(u))\big] \cdot 0 + \frac{\partial^2}{\partial Y^2}\big[l(X(u),Y(u))\big] \cdot \Delta Y \right\} \cdot \Delta Y \\
\Rightarrow &\frac{d^2}{du^2}\big[F(u)\big] = \left\{ \frac{\partial^2}{\partial Y^2}\big[l(X(u),Y(u))\big] \cdot \Delta Y \right\} \cdot \Delta Y \\
\Rightarrow &\frac{d^2}{du^2}\big[F(u)\big] = \frac{\partial^2}{\partial \big(\hat{y}_i^{(t-1)}\big)^2} l(y_i, \hat{y}_i^{(t-1)}) \cdot \Delta Y \cdot \Delta Y \\
\Rightarrow &\frac{d^2}{du^2}\big[F(u)\big] = \frac{\partial^2}{\partial \big(\hat{y}_i^{(t-1)}\big)^2} l(y_i, \hat{y}_i^{(t-1)}) \cdot f_t^2(x_i) \\
&\Rightarrow F''(u) = h_i f_t^2(x_i)
\end{aligned} \tag{19}$$

where

$$h_i = \frac{\partial^2}{\partial \big(\hat{y}_i^{(t-1)}\big)^2} l(y_i, \hat{y}_i^{(t-1)})$$

The Taylor series approximation till the second order is then given as:

$$\begin{aligned}
l\big[y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\big] &= F(0) + F'(0) + \frac{1}{2}F''(0) \\
&= l(\hat{y}_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)
\end{aligned} \tag{20}$$

Thus the objective function from Equation 14 can be approximated as:

$$\begin{aligned}
\mathcal{L}^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k) \\
&= \sum_{i=1}^{n} \big[l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i))\big] + \Omega(f_t) + constants \\
&= \sum_{i=1}^{n} \big[l(\hat{y}_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\big] + \Omega(f_t) + constants
\end{aligned} \tag{21}$$

As constants are not required in optimization, all the constant terms (note that the term $l(\hat{y}_i, \hat{y}_i^{(t-1)})$ can be considered to be constant in the t^{th} iteration) may be removed, thus consolidating the objective function further to:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{22}$$

The optimization, hence, only depends on g_i and h_i. XGBoost can support any loss function by taking g_i and h_i as input. These are known as the *gradient statistics* of the structure.

Each tree $f(x)$ is defined as:

$$f_t(x) = w_{q(x)}, w \in R^T, q : R^d \to 1, 2, ..., T \tag{23}$$

Here, w is the vector of scores on leaves q is a function assigning each data point to the corresponding leaf T is the number of leaves.

The regularization function Ω is defined as:

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \tag{24}$$

Here, γ is the minimum gain a branch needs to contribute to the entire learner in order to be added to the overall structure. The idea of γ becomes clearer later in the derivation. There are multiple ways of defining the regularization function, but the one used here is an L2 regularization.

From equations (previous two), the objective value with the t^{th} iteration is as follows:

$$\begin{aligned} Obj^{(t)} &\approx \sum_{i=1}^{n} \left[g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2 \right] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} \left[\Big(\sum_{i \in I_j} g_i \Big) w_j + \frac{1}{2} \Big(\sum_{i \in I_j} h_i + \lambda \Big) w_j^2 \right] + \gamma T \end{aligned} \tag{25}$$

where $I_j = i | q(x_i) = j$ is the set of indices of data points assigned to the j^{th} leaf, and λ is the *learning rate*. If $G_j = \sum\limits_{i \in I_j} g_i$ and $H_j = \sum\limits_{i \in I_j} h_i$, then Equation 25 becomes:

$$Obj^{(t)} = \sum_{j=1}^{T} \left[G_j w_j + \frac{1}{2} \big(H_j + \lambda \big) w_j^2 \right] + \gamma T \tag{26}$$

The part of Equation 39 within the summation is in a quadratic form. The optimal condition of a second degree objective function is at the point where the first derivative is zero. Hence, substituting the value of w_j at the point where the slope of the objective function is zero gives the optimal value of the objective function.

$$\begin{aligned} &\frac{d}{dw_j} \left[G_j w_j^* + \frac{1}{2} \big(H_j + \lambda \big) (w_j^2)^* \right] = 0 \\ &\Rightarrow G_j + 2 \cdot \frac{1}{2} \big(H_j + \lambda \big) w_j^* = 0 \\ &\Rightarrow \big(H_j + \lambda \big) w_j^* = -G_j \\ &\Rightarrow w_j^* = \frac{-G_j}{\big(H_j + \lambda \big)} \end{aligned} \tag{27}$$

Thus, substituting w_j^* in the objective function, we get the optimal value of the objective (from Equation 39):

$$
\begin{aligned}
Obj^* &= \sum_{j=1}^{T} \left[G_j \cdot \left(\frac{-G_j}{H_j + \lambda} \right) + \frac{1}{2}(H_j + \lambda) \left(\frac{-G_j}{H_j + \lambda} \right)^2 \right] + \gamma T \\
&= \sum_{j=1}^{T} \left[\frac{-G_j^2}{H_j + \lambda} + \frac{1}{2} \cdot \frac{G_j^2}{H_j + \lambda} \right] + \gamma T \\
&= -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T
\end{aligned} \tag{28}
$$

Now we have an objective function which can tell us about the goodness of a tree. But how do we decide what is the best split? Analogous to the GiniGain measure in the case of random forests, the Gain in XGBoost is used to split a node. The form of the gain function is similar to GiniGain; the best split at a node results in the best Gain.

$$
Gain = -\frac{1}{2} \cdot [Gain_L + Gain_R + Gain_{Root}] - \gamma \tag{29}
$$

The loss function commonly used in XGBoost is the squared loss function, which is given by:

$$
\mathcal{L} = \left[y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)) \right]^2 \tag{30}
$$

For which the gradient scores are calculated as:

$$
\begin{aligned}
g_i &= \frac{\partial}{\partial \hat{y}_i^{(t-1)}} \left[y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)) \right]^2 \\
&= 2 \cdot \left[y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)) \right] \cdot \left[0 - (1 + 0) \right] \\
&= -2 \left[y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)) \right] \\
\Rightarrow h_i &= \frac{\partial}{\partial \hat{y}_i^{(t-1)}} \Big\{ -2 \left[y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)) \right] \Big\} \\
&= -2 \left[0 - (1 + 0) \right] \\
&= 2
\end{aligned} \tag{31}
$$

The algorithm of GBDTs are explained in Algorithms 3 and 4.

3.2. Algorithms

Algorithm 3: GradientBoostedTree

input : $(x_i, y_i)_1^n$ is the labeled training data
γ is the minimum required structure score
L is the maximum number of levels in the tree
l is the current level of the tree
λ is the learning rate
N is the number of training steps

output: A tree which is configured to predict the class label of a test sample

---

$l \leftarrow l + 1$;
if $l \leq L$ **then**
  $t \leftarrow 0$;
  $f \leftarrow 0$;
  while $t < N$ **do**
    estimate f_t as a regressor function, density, or distribution;
    $f \leftarrow f + f_i$;
  end
  initialize array of scores $S[:]$;
  for $j := 1$ *to* n **do**
    $G_j \leftarrow 0$;
    $H_j \leftarrow 0$;
    for $i := 1$ *to* N **do**
      $G_j \leftarrow G_j + g_{ji}$;
      $H_j \leftarrow H_j + h_{ji}$;
      /* g_{ji} and h_{ji} are the gradient statistics of the j^{th} sample for the i^{th} function */
    end
    $S[j] \leftarrow -0.5 \cdot G_j^2 \div (H_j + \lambda)$;
  end
  $C_L, C_R \leftarrow$ **MaxGain**$((x_i, y_i)_1^n, S)$; // C_L, C_R are the left and right children of this node respectively
  if C_L ***does not satisfy the desired level of purity*** **then**
    GradientBoostedTree$(C_L, \gamma, L, l, \lambda, N)$;
  if C_L ***does not satisfy the desired level of purity*** **then**
    GradientBoostedTree$(C_R, \gamma, L, l, \lambda, N)$;

Algorithm 4: MaxGain

input : $(x_i, y_i)_1^m$ is the labeled training data
$S[:]$ is the list of structure scores

output: Left and right child nodes, ensuring maximum purity after split

---

Sort $(x_i, y_i)_1^m$ and $S[:]$ in the increasing order of $S[:]$;
$Gain_{split} \leftarrow -\infty$ /* $Gain_{split}$ stores the value of the gain after each split */
$Gain_N \leftarrow \sum S[:]$ /* the overall score of the parent node */
$SplitPoint \leftarrow -1$ /* stores the best split point */
$f \leftarrow 0$;
for $j := 1$ *to* $m - 1$ **do**
  $N_L \leftarrow S[0 : j]$ /* list of scores of left child */
  $N_R \leftarrow S[j + 1 : m]$ /* list of scores of right child */
  $G_L \leftarrow \sum N_L$ /* overall gain of left child */
  $G_R \leftarrow \sum N_R$ /* overall gain of right child */
  $Gain \leftarrow G_L + G_R - G_N$ /* gain calculated in the j^{th} iteration */
  /* the best gain is selected iteratively */
  if $Gain > Gain_{split}$ ***and*** $Gain > \gamma$ **then**
    $Gain_{split} \leftarrow Gain$;
    $SplitPoint \leftarrow j$;
end
return $(x_i, y_i)_1^j, (x_i, y_i)_{j+1}^m$;

4. Exposition of Random Forests and XGBoost

4.1. Random Forest

4.1.1. Decision Trees: Background

Decision trees (Geurts & Louppe, 2014; Breiman, 2001) can be used for various machine learning applications. But trees that are grown really deep to learn highly irregular patterns tend to over-fit the training sets. Noise in the data may cause the tree to grow in a completely unexpected manner. Random Forests overcome this problem by training multiple decision trees on different subspaces of the feature space at the cost of slightly increased bias. This means that none of the trees in the forest sees the entire training data. The data is recursively split into partitions. At a particular node, the split is done by asking a question on an attribute. The choice for the splitting criterion is based on some impurity measures such as Gini impurity or Shannon Entropy.[1]

Gini impurity is used as the function to measure the quality of split in each node. Gini impurity at node N is given by:

$$G(N) = 1 - (P_1)^2 - (P_{-1})^2 \tag{32}$$

where P_i is the proportion of the population with class label i. The obvious heuristic approach to choose the best splitting decision at a node is the one that reduces the impurity as much as possible. In order words, the best split is characterized by the highest gain in information or the highest reduction in impurity. The information gain due to a split can be calculated as follows:

$$Gain = I(N) - p_L I(N_L) - p_R I(N_R) \tag{33}$$

where $I(N)$ is the impurity measure (Gini, or any other) of node N, $I(N_L)$ is the impurity in the left child of node N after the split and similarly, $I(N_R)$ is is the impurity in the right child of node N after the split; N_L and N_R are the left and right children of N respectively' p_L and p_R are the proportions of the samples in the left and right children nodes with respect to the parent node.

Note that Equations 2, 32, and 33 can be used when there are only two splits at each node. However, depending on the algorithm used, there may be more than two splits and hence these formulae may be generalized as:

$$\begin{aligned} G(N) &= 1 - \sum_{j=1}^{n} P_j \\ Gain &= I(N) - \sum_{j=1}^{n} p_j I(N_j) \end{aligned} \tag{34}$$

In our work, we have used the CART (Classification and Regression Trees) algorithm, which splits every node into only two children nodes.

Bootstrap aggregating, also known as *bagging*, is an ensemble learning method that improves the stability and accuracy of learning algorithms while reducing variance and over-fitting, which are common problems while constructing decision trees. Given a sample dataset D of shape $n \times m$, bagging generates B new sets of shape $n' \times m'$ by sampling uniformly from D with replacement, where $n < n'$ and $m < m'$.

4.1.2. A Relevant Example Based on the Data

In random forests, decision tree learners are constructed by randomly selecting m out of M features and n out of N samples. Here, we illustrate the working of random forests by randomly considering 20 samples from the data set as the training set and 5 samples as the test set; the training and test sets are mutually exclusive. They are shown in Tables 1 and 2 respectively. Let us take $m = \sqrt{M}$, so that in each tree constructed, $\sqrt{M}$ features are randomly considered. In the data set, since there are 6 features, $\sqrt{M} = \sqrt{6} \approx 2.45$, hence, in each tree, 2 features will be considered. We shall take the subsample size to be equal to the sample size, i.e., 20 in each tree constructed here.

[1]Preprint is available in Arxiv and ResearhGate. Plagiarism similarity score may be misleading.

S.Id	Class Label	RSI	SCH	W%R	MACD	PROC60	OBV
1	-1	91.02	74.0	-54.07	1.12	0.03	259336600.0
2	1	11.09	17.12	-35.42	-0.8	-0.05	23149700.0
3	1	75.95	77.76	-54.6	0.64	-0.02	263929400.0
4	-1	88.25	85.8	-50.08	1.02	0.01	369776000.0
5	-1	16.71	14.19	-35.08	-0.67	0.04	1434446700.0
6	1	9.55	11.48	-35.28	-1.5	-0.02	1264042100.0
7	1	66.21	77.38	-48.78	-0.23	-0.02	198021100.0
8	-1	67.82	77.98	-50.25	0.36	0.02	276163600.0
9	-1	71.88	49.71	-50.82	0.58	0.08	560679400.0
10	1	4.66	11.6	-44.17	-1.56	-0.06	135643900.0
11	1	1.52	16.01	45.27	-0.33	-0.13	-2263609600.0
12	-1	17.41	24.08	-44.97	-0.12	0.02	-3417917300.0
13	1	22.27	31.57	-40.62	-0.22	-0.03	-3265559400.0
14	-1	70.79	77.25	-45.93	0.09	0.02	-2846701500.0
15	1	77.74	84.71	-72.5	0.36	-0.08	-3516514200.0
16	-1	57.5	73.5	-44.46	-0.12	0.09	-2926120100.0
17	1	42.81	54.22	-52.63	-0.26	-0.02	-3213555400.0
18	1	61.83	72.97	-66.66	-0.09	-0.1	-3904360800.0
19	-1	18.86	16.0	-36.27	-0.13	0.03	-3645587700.0
20	-1	63.08	52.02	-50.11	-0.03	0.02	-2182828500.0

Table 1: Sample Training Set

S.Id	Class Label	RSI	SCH	W%R	MACD	PROC60	OBV
1	-1	24.97	15.54	-43.74	-0.57	0.15	-1373395100.0
2	-1	85.06	75.77	-59.75	0.37	0.08	-1711049700.0
3	-1	86.88	70.44	-66.76	0.78	0.03	-1910292300.0
4	1	0.69	13.22	-36.48	-1.35	-0.19	-2573216900.0
5	1	78.9	80.95	-47.66	1.71	-0.01	-2371874000.0

Table 2: Sample Test Set

S.Id	Class Label	RSI	SCH
11.0	1	1.52	16.01
10.0	1	4.66	11.6
6.0	1	9.55	11.48
2.0	1	11.09	17.12
5.0	-1	16.71	14.19
12.0	-1	17.41	24.08
19.0	-1	18.86	16.0
13.0	1	22.27	31.57
17.0	1	42.81	54.22
16.0	-1	57.5	73.5
18.0	1	61.83	72.97
20.0	-1	63.08	52.02
7.0	1	66.21	77.38
8.0	-1	67.82	77.98
14.0	-1	70.79	77.25
9.0	-1	71.88	49.71
3.0	1	75.95	77.76
15.0	1	77.74	84.71
4.0	-1	88.25	85.8
1.0	-1	91.02	74.0

Table 3: Training set sorted in increasing order of RSI.

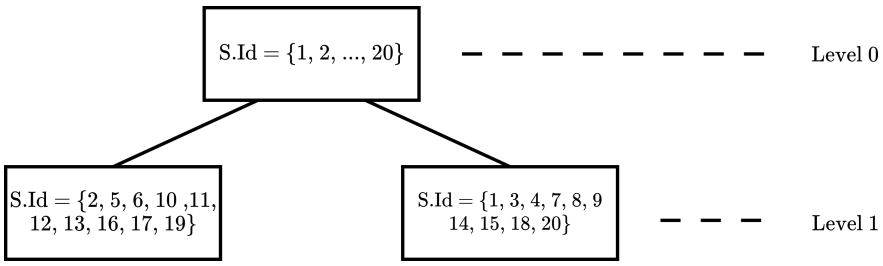


Figure 1: Tree creation by arbitrarily using the first 10 samples from Table 3 as the left child and the last 10 samples as the right child

The best *split* for a node in a decision tree is determined by sorting the samples in a node with respect to a feature (considered by the respective tree) and then partitioning it into two parts: the left and right children of that node. The partitioning is done by selecting the best threshold (hence, the sorting) for the corresponding feature and calculating the Gini gain (insert equation number). For example, let us consider a tree constructed by using the first two features in the training set, RSI and Stochastic Oscillator (SCH), the training set as seen by the decision tree, after being sorted based on RSI, appears as shown in Table 3.

The root node of the decision tree will perform the first partition of the data in Table 3. Hence, the constituents of the root node of the tree (or any tree) will have all the samples in the training set with 2 features selected at random. Let us consider arbitrarily the first 10 samples in Table 3 to constitute the *left child* and the last 10 samples to constitute the *right child* of the root node. The tree, after just the first split, will appear as shown in Figure 1.

Here, the impurity of the parent node (in this case, at level 0; the root node) is 0.5. This is easily understandable as there are 10 samples belonging to Class 1 and 10 samples belonging to Class -1. So the Gini Impurity is calculated as shown in Equation 35.

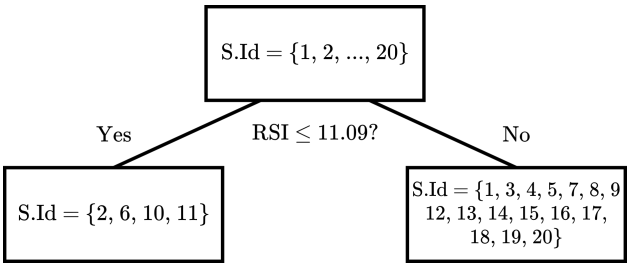


Figure 2: The best split for the root node when considering RSI and SCH.

$$\begin{aligned} G(N) &= 1 - (P_1)^2 - (P_{-1})^2 \\ &= 1 - \left(\frac{10}{20}\right)^2 - \left(\frac{10}{20}\right)^2 \\ &= 1 - 0.25 - 0.25 \\ &= 0.5 \end{aligned} \tag{35}$$

However, as the quality of the split depends on the *GiniGain*, the Gini impurities of the left and right children also need to be calculated. They are calculated in a manner similar to Equation 35.

$$\begin{aligned} G(N_L) &= 0.48 \\ G(N_R) &= 0.48 \end{aligned} \tag{36}$$

Note that the left child has 6 samples belonging to Class 1 and 4 samples belonging to Class -1, and the right child has 6 samples belonging to Class -1 and 4 samples belonging to Class 1. Hence, in this case, $G(N_L)$ and $G(N_R)$ are equal. It is not always necessary for them to be equal. The *Gain*, is calculated as in Equation 37.

$$Gain = 0.5 - \frac{10}{20} \times 0.48 - \frac{10}{20} \times 0.48 = 0.02 \tag{37}$$

This is how a node in a tree is split. At the next level of the tree, in each child node, the best split is determined again in a similar way. But now the partitions of the training sample space belonging to the children nodes are considered for partitioning each respective child node. This kind of partitioning is done recursively till *pure* nodes are reached. In decision trees, all leaf nodes need to be pure nodes (a leaf node is a node that is not split further; see 1 for more definitions and details). A node is said to be *pure* if all the samples in that node belong to the same class such that it does not require any further splitting. When an element needs to be classified, through a series of partitioning criteria, it reaches a leaf node and is assigned the class label of all the elements in the respective leaf node.

Returning to the illustration, it should be noted that a 50:50 split on the root node, with the first 10 samples from Table 3 constituting the left child node and the last 10 samples constituting the right child node is not the best split for that node. In fact, the best split is when the left child node has the first 4 samples and the right child node has the last 16 samples, such that the split as shown in Figure 2. For this split, the *GiniGain* is 0.125, which is greater than that for the split shown in Figure 1.

Now, how is this number 11.09, as the splitting threshold, determined? Looking at Table 3, we see that it is the value of the RSI attribute of the fourth sample. Reiterating, the samples in this table are sorted in the increasing order of RSI. As the partition till the 4^{th} sample for the left child node and the last 16 samples for the right child node results in the best GiniGain, the RSI value of the last sample comprising the left partition is used as a boundary value, or the split value (notice how the first four samples all belong to Class 1: this should help develop the readers' intuition on the working of *GiniGain* for determining the best split).

S.Id	RSI	SCH	Class Label	Predicted Label
1	24.97	15.54	-1	1
2	85.06	75.77	-1	-1
3	86.88	70.44	-1	-1
4	0.69	13.22	1	1
5	78.9	80.95	1	-1

Table 4: Results of classification the sample test set using **Tree 1**

Thus, the split point for the root node becomes $(RSI, 11.09)$. When an element is being classified using this tree, it essentially needs to reach a leaf node by going through a series of splits. Hence, the first split will be done using $(RSI, 11.09)$: if the value of RSI is ≤ 11.09) for an element to be classified, it will traverse the left branch of the root node (which, in this case, is a pure node), and if the value of RSI > 11.09, it will traverse the right branch, and consequently, a series of more splits, till it reaches a leaf node. The complete tree, built recursively such that all leaf nodes are pure nodes, is shown in Figure 3.

The geometric interpretation of a decision tree is that of partitions in the feature space corresponding to each class. Using the thresholds determined using the *Gain* for each split, the feature space can be considered to be divided into partitions or *pockets*, with each partition corresponding to one class. The partitions of the feature space for Tree 1 are shown in Figure 4. In this graph, all the partitions in blue color correspond to Class 1, and all the partitions in red color correspond to Class 2. On this graph, if we were to plot each sample in the test set (Table 2) by considering only (RSI, SCH), then the *color* of the partition a sample belonged to would indicate which class the respective sample is a part of (as the color of the partition represents the Class label)! Thus, random forests can be used to effectively handle non-linear trends in data or separability of data.

Now that a tree is constructed using the training set, the obvious task at hand is to classify the samples in the test set using this tree. Reiterating, in order to classify a test sample, we need to trace a test sample down the tree until a leaf node is reached. As leaf nodes are pure, the class of the leaf node will be assigned to the test sample being classified.

From Table 4, we can see that **Tree T1** misclassifies 2 out of 5 test samples, samples 1 and 5. Hence, it can be said that **T1** has a classification accuracy of 60%. Generally, a 60% accuracy for a classifier is not considered to be good enough. So in order to have a complete understanding of how a random forest might work, we shall construct two more trees and consider the majority vote for each test sample.

Let us construct the next tree in the forest by considering the features W%R and SCH. Notice that we are re-sampling the values of SCH. This is the principle of bootstrap aggregation, more commonly known as *bagging*: to sample data with replacement. Bagging has certain advantages. While growing a tree, a feature may or may not be chosen to be a feature using which the tree grows. This is especially relevant for high-dimensional datasets. Since each feature has a certain probability of success or failure of being chosen when a tree is being grown, it can be said that the probability of a feature being used in a random forest depends on the number of trees being constructed, with the probability being *binomially distributed*.

In the dataset used for the current work, as we have already mentioned, there are 6 features and for each tree being constructed, 2 features are considered. This implies that the probability of *success* (success being the condition that a feature is chosen to grow a decision tree) is $P(success) = 2/6 = 1/3 = 0.33$. The probability of *failure* (failure being the condition that a feature is not chosen to grow a decision tree) is $P(failure) = 1 - P(success) = 0.67$. Let us represent the probability of success as p and the probability of failure as q. Then the probability that a feature x_a is used in growing k out of n trees is given by Equation 38.

$$P(k \ successes) = \binom{n}{k} p^k q^{n-k} \tag{38}$$

It is easy to see that the LHS of Equation 38 increases as the value of n increases. This can be used to understand the idea of *stability* of a random forest: that upon increasing the number of trees in a random

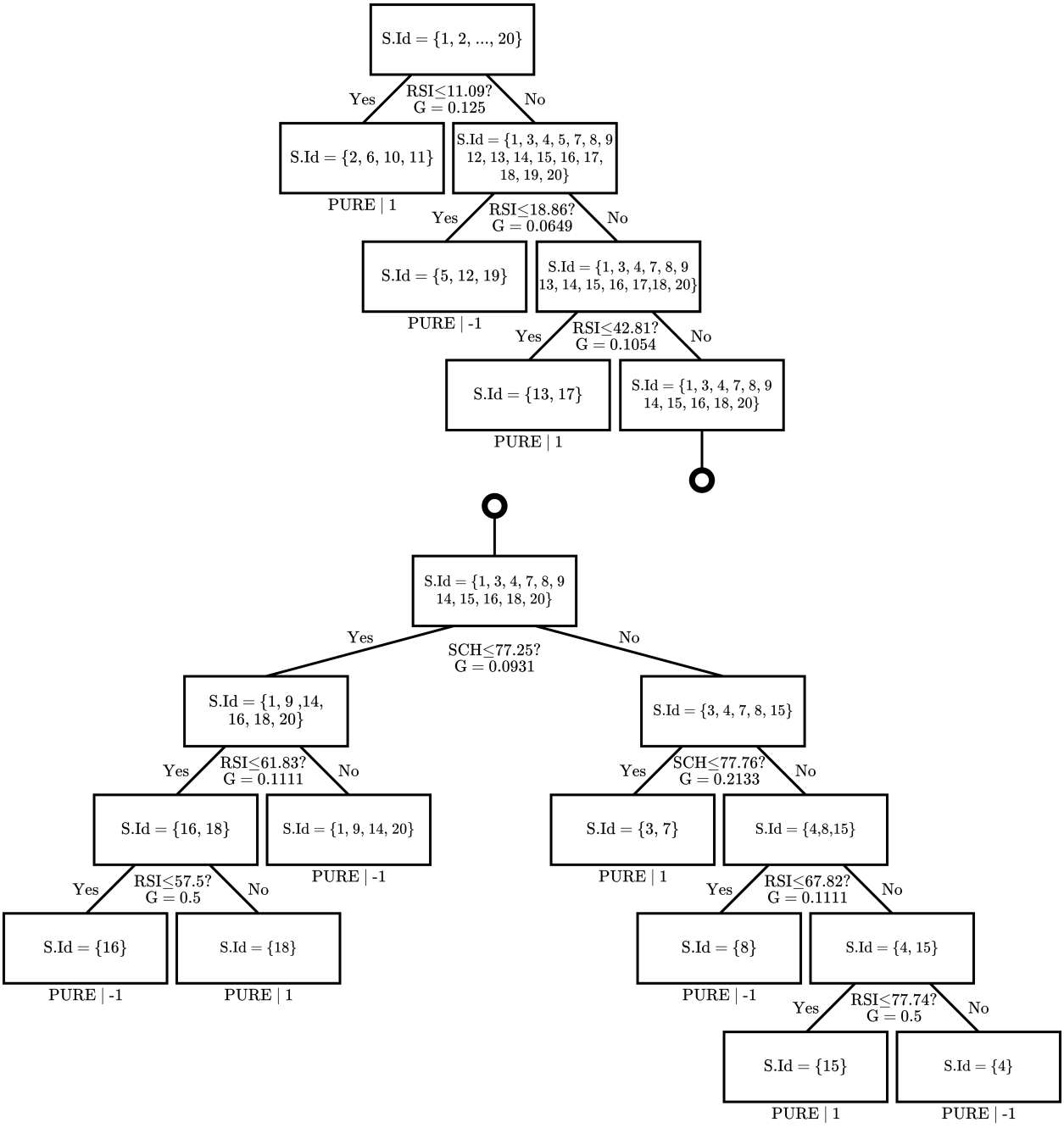


Figure 3: **Tree 1**: Random tree constructed considering RSI and SCH

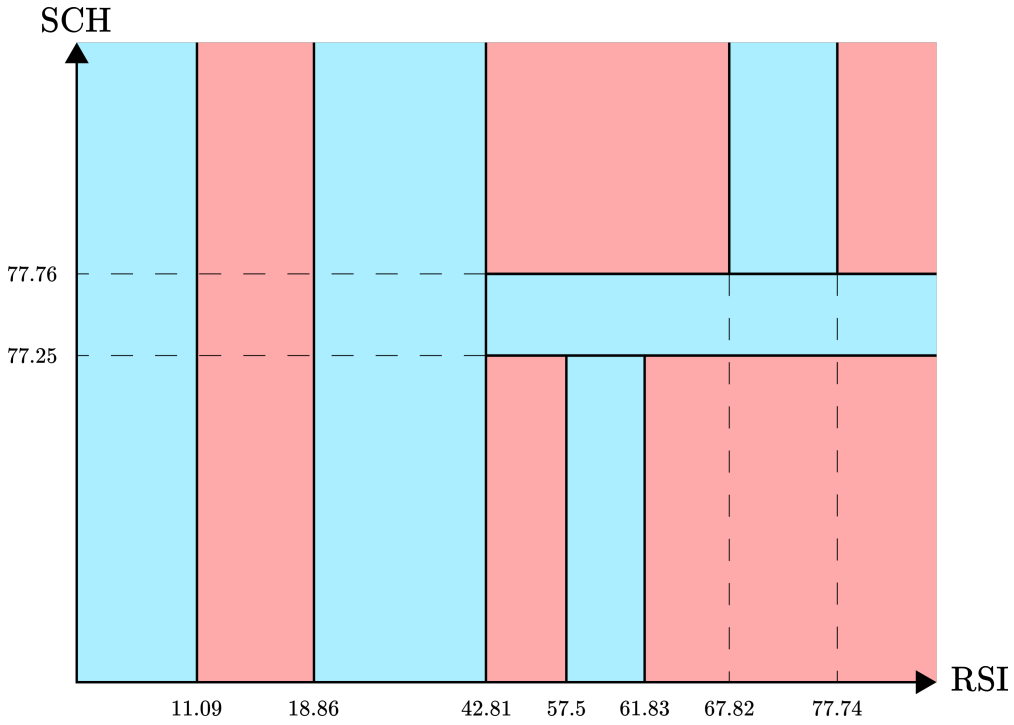


Figure 4: Partitions of sample space of Tree 1 (graph not drawn to scale)

S.Id	W%R	SCH	Class Label	Predicted Label
1	-43.74	15.54	-1	-1
2	-59.75	75.77	-1	1
3	-66.76	70.44	-1	1
4	-36.48	13.22	1	-1
5	-47.66	80.95	1	1

Table 5: Results of classification the sample test set using **Tree 2**

forest, the expected results are stabilized by continuous re-sampling of data so that the forest as a whole is representative of the class to which a sample to be classified belongs.

From Table 5, we can see that 2 out of 5 samples are classified correctly. This performance, as an individual tree, is not good. Upon comparing the predicted results from **Tree 1** and **Tree 2**, we see that the results from both trees combined are accurate only 50% of the time. The system as a whole is not accurate enough to predict the class of a previously unobserved sample accurately. To move a step closer towards an accurate system, let us construct another tree. We shall construct **Tree 3** using SCH and PROC60.

The decision tree constructed using SCH and PROC60 is a rather interesting one: there's only one split in the tree and the children of the root nodes are pure! The results of classification using just this tree is given in Table 6. This single tree gives us a 100% classification accuracy! The question arises: *why not use just this tree to classify the sample?*. The reasons go back to the notions of stability: that if there's some change in the data, some of the trees might not perform well, but the results of the forest should largely remain unchanged. In general, a forest classifier can be used to reduce the effect of muddy data. As this is an illustrative example, we are dealing with only 20 samples. However, the trend in millions of samples may not be as easy to deal with a single threshold. Besides, more recent innovations in tree based classifiers (such as AdaBoost, XGBoost) can discern or decide how to grow trees based on the performance of previously grown trees in the forest.

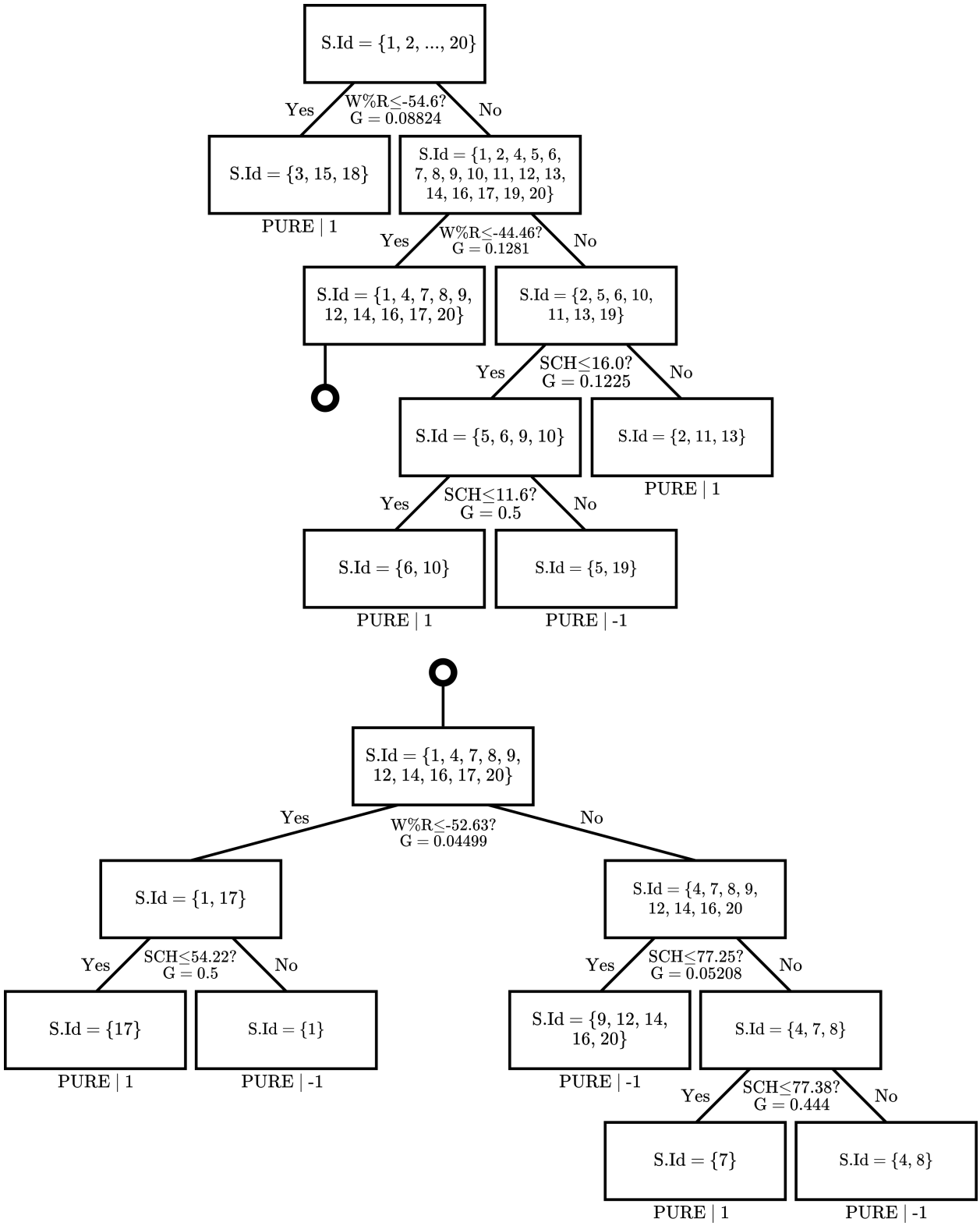


Figure 5: **Tree 2**: Random tree constructed considering W%R and SCH

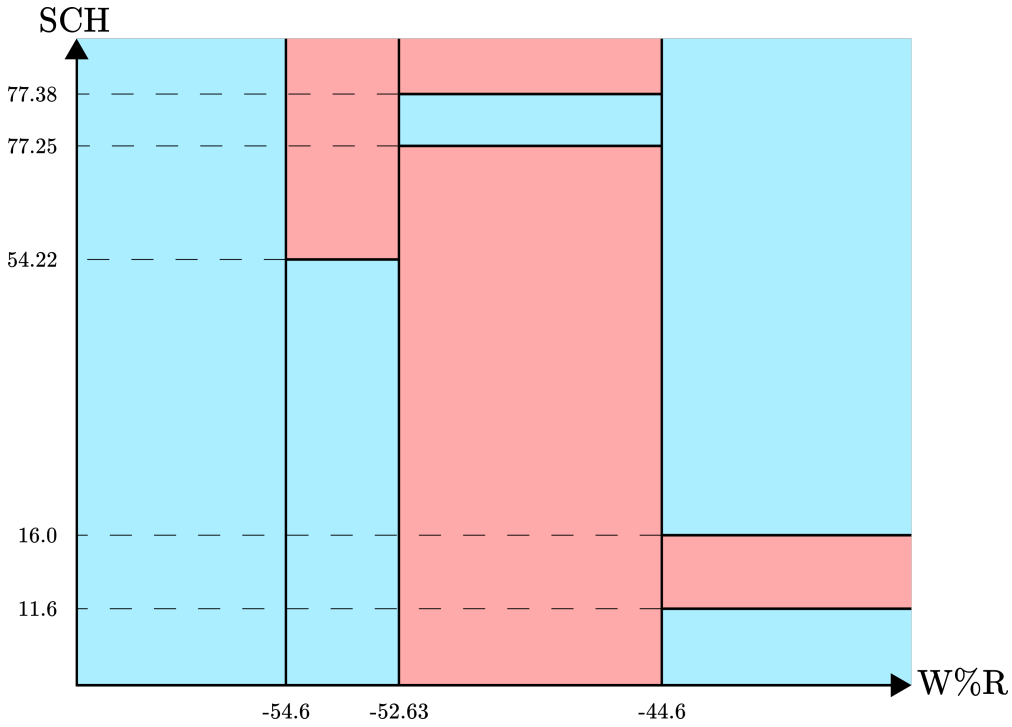


Figure 6: Partitions of sample space of Tree 2 (graph not drawn to scale)

Now that there are three trees in our forest, we can perform a majority voting. The result of the entire tree using **Tree 1**, **Tree 2** and **Tree 3** is given in Table 7. By combining the performance of all the trees in the forest, the results are perfect, and also stable.

The algorithm used to build decision trees is described in Algorithms 2 and 1.

4.1.3. Out-of-Bag (OOB) Error Visualization

After creating all the decision trees in the forest, for each training sample $Z_i = (X_i, Y_i)$ in the original training set T, we select all bagged sets T_k which does not contain Z_i. This set contains bootstrap datasets which do not contain a particular training sample from the original training dataset. These sets are called out of bags examples. There are n such sets for each n data samples in the original training dataset. OOB error is the average error for each Z_i calculated using predictions from the trees that do not contain z_i in their respective bootstrap sample. OOB error is an estimate of generalization error which measures how accurately the random forest predicts previously unseen data. We plotted the OOB error rate for our random forest classifier using the AAPL dataset.

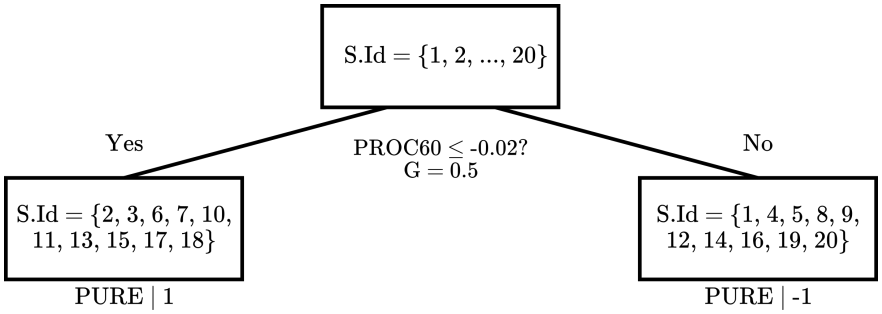


Figure 7: **Tree 3**: Random tree constructed considering SCH and PROC60

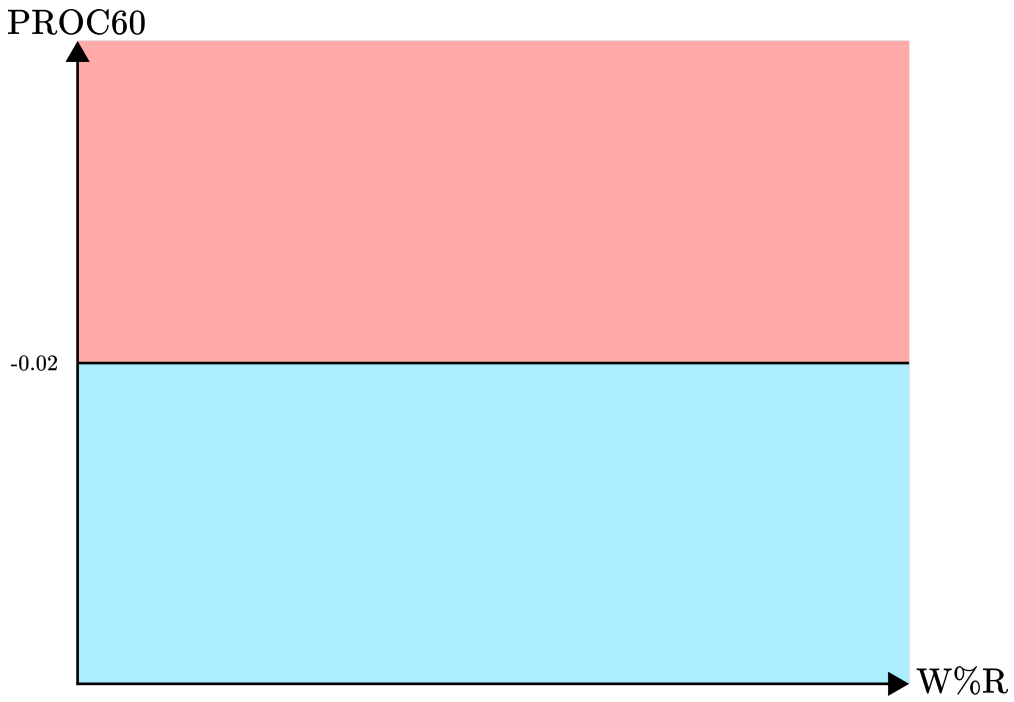


Figure 8: Partitions of sample space of Tree 3 (graph not drawn to scale)

S.Id	SCH	PROC60	Class Label	Predicted Label
1	15.54	0.15	-1	-1
2	75.77	0.18	-1	-1
3	70.44	0.03	-1	-1
4	13.22	-0.19	1	1
5	80.95	-0.10	1	1

Table 6: Results of classification the sample test set using **Tree 3**

S.Id	Class Label	Tree 1	Tree 2	Tree 3	Predicted Label
1	-1	1	-1	-1	-1
2	-1	-1	1	-1	-1
3	-1	-1	1	-1	-1
4	1	1	-1	1	1
5	1	-1	1	1	1

Table 7: Results of classification of the entire forest using majority voting

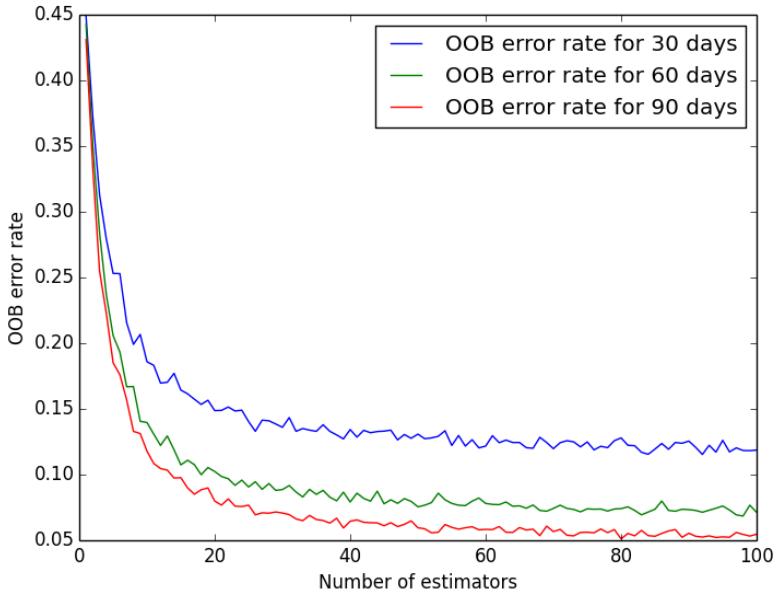


Figure 9: OOB error rate vs Number of estimators

From Figure 9 OOB plot, we can see that the OOB error rate decreases dramatically as more number of trees are added in the forest. However, a limiting value of the OOB error rate is reached eventually. The plot shows that the Random Forest converges as more number of trees are added in the forest. This result also explains why random forests do not over-fit as more number of trees are added into the ensemble.

$L(y, \gamma)$ is the differentiable loss function and $h_m(x)$ is the base learner, connected by the following relation: $F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$.

4.2. eXtreme Gradient Boosting (XGBoost)

4.2.1. Gradient Boosted Decision Trees: A Convex Optimization Approach to Tree Classifiers

The tree approximation in gradient boosted trees is done by aggregating many functions by *additive learning*. Each node is hence built sequentially, with each successive approximated function trying to better classify the residuals of the previous learner. XGBoost employs an additive strategy wherein every subsequent approximated function optimizes an objective function. Hence, a series of functions are built such that the node is ultimately approximated using an aggregate of all the functions and is gradually optimized. The objective function may be represented as:

$$Obj^{(t)} = \sum_{j=1}^{T} \left[G_j w_j + \frac{1}{2}\big(H_j + \lambda\big) w_j^2 \right] + \gamma T \tag{39}$$

where $I_j = i|q(x_i) = j$ is the set of indices of data points assigned to the j^{th} leaf, λ is the *learning rate*, $G_j = \sum_{i \in I_j} g_i$ and $H_j = \sum_{i \in I_j} h_i$, and g_i and h_i are the first and second partial derivatives of the loss function respectively.

The part of Equation 39 within the summation is in a quadratic form. The optimal condition of a second degree objective function is at the point where the first derivative is zero. Hence, substituting the value of w_j at the point where the slope of the objective function is zero gives the optimal value of the objective function.

$$w_j^* = \frac{-G_j}{\big(H_j + \lambda\big)} \tag{40}$$

S. Id	g_i	h_i	$\hat{y}_i^{(t-1)}$	y_i
2	0	2	0	1
6	0	2	0	1
10	0	2	0	1
11	0	2	0	1

Table 8: Gradient statistics for left child node of root node of Tree 1.

S. Id	g_i	h_i	$\hat{y}_i^{(t-1)}$	y_i
1	0	2	0	-1
3	-2	2	0	-1
4	0	2	0	-1
5	0	2	0	-1
7	-2	2	0	-1
8	0	2	0	-1
9	0	2	0	-1
12	0	2	0	-1
13	-2	2	0	-1
14	0	2	0	-1
15	-2	2	0	-1
16	0	2	0	-1
17	-2	2	0	-1
18	-2	2	0	-1
19	0	2	0	-1
20	0	2	0	-1

Table 9: Gradient statistics for right child node of root node of Tree 1.

Thus, substituting w_j^* in the objective function, we get the optimal value of the objective (from Equation 39):

4.2.2. A Relevant Example for XGBoost

Consider the split made in the root node in Tree 1 (Figure 4). Utilizing the definitions laid down of g_i, h_i, Gain, λ, and γ, it can be considered from the perspective of gradient boosted trees that the left child node of the root node estimates $\hat{y}_i$ as Class 1 and the right node estimates it as -1, as the majority of the samples used for building the tree belong to classes $+1$ and -1 in the left and right children nodes, respectively. Thus, the structure score of the left branch may be calculated as (from Table 8):

From Table 9, G_j and H_j may be calculated as: $G_j = \sum g_i = 0$ and $H_j = \sum h_i = 8$. All the entities in this node truly belong to Class 1, and by the definition of g_i, the value of G_i sums up to 0. As the root node is the first in the series, it does not classify any of the entities and hence $\hat{y}_i^{(t-1)}$ for this split is 0 (note that the class labels are $+1$ and -1, signifying an increase and decrease in prices respectively). Consider the value of learning rate, λ, to be 1, arbitrarily. The structure score can thence be calculated as:

$$obj^* = -\frac{1}{2} \cdot \frac{G_L^2}{H_L + \lambda} = -\frac{1}{2} \cdot \frac{0}{8+1} = 0 \tag{41}$$

Similarly, for the right child node, the gradient statistics may be calculated using Table 9, in the following manner:

Here, $G_j = -12$ and $H_j = 32$. The structure score then becomes:

$$obj^* = -\frac{1}{2} \cdot \frac{G_R^2}{H_R + \lambda} = -\frac{1}{2} \cdot \frac{-12^2}{32+1} = -4.36 \tag{42}$$

Next, we shall see how the Gain of the split is calculated. For this, let us consider the value of γ to be 0.3, arbitrarily:

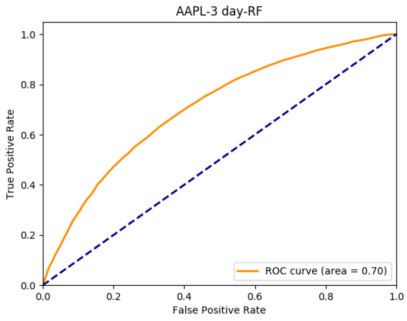
$$\begin{aligned} Gain &= -\frac{1}{2} \cdot \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G_{Root}^2}{H_{Root} + \lambda} \right] - \gamma \\ &= -\frac{1}{2} \cdot \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma \\ &= -\frac{1}{2} \cdot \left[0 + \frac{144}{32 + 1} - \frac{(0 - 12)^2}{8 + 32 + 1} \right] - 0.3 \\ &= 0.8514 - 0.3 \\ &= 0.5514 \end{aligned} \tag{43}$$

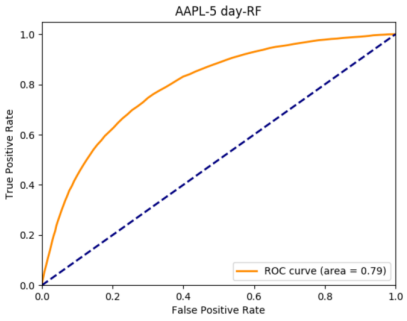
This is how the gain is calculated. In a similar fashion, the remaining levels of this tree are constructed. The parameters λ and γ can be tuned for optimal performance. Once this tree is complete, the next tree is built by bagging. Thus, with every level and every tree in sequence, the forest of trees is able to reduce the error in classification.

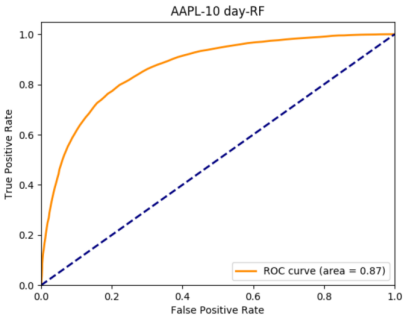
5. Receiver Operating Characteristic Plots

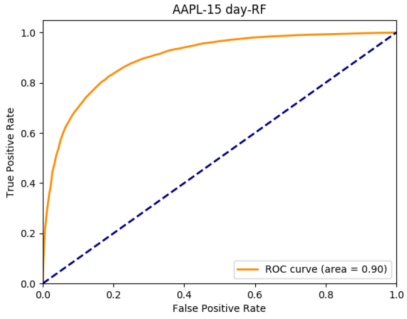
In this section, we present the ROC curves of the classifiers' outputs for the results presented and discussed in the main paper in Section 4.

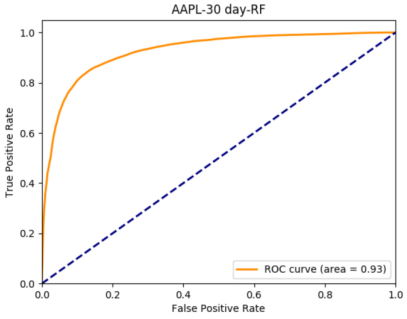
AAPL

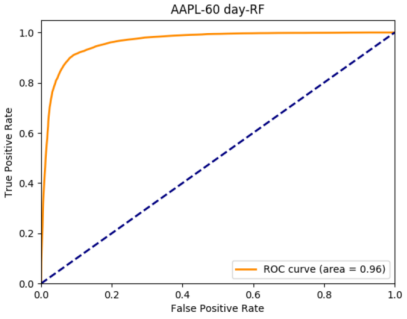

AAPL-3 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.70)

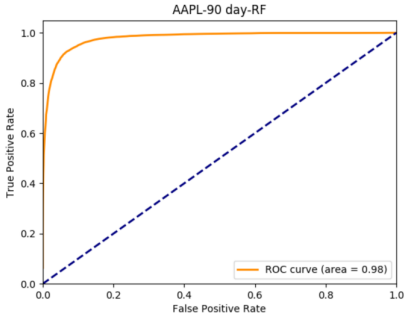

AAPL-5 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.79)

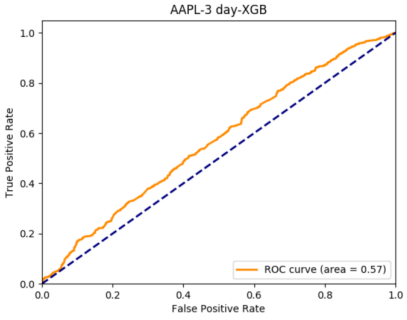

AAPL-10 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.87)

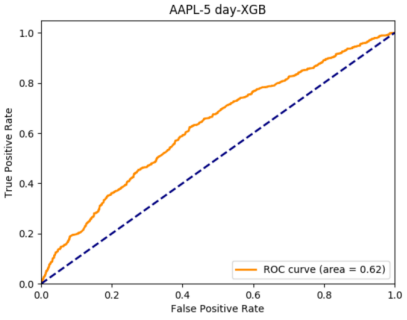

AAPL-15 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.90)

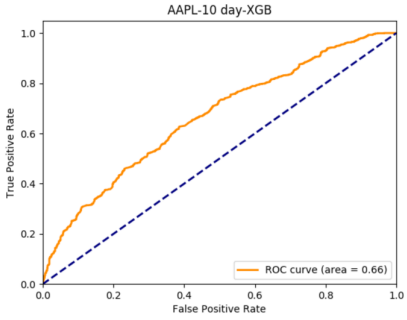

AAPL-30 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.93)

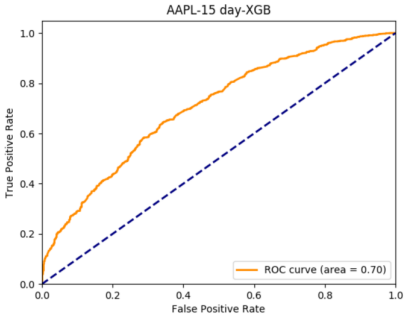

AAPL-60 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)

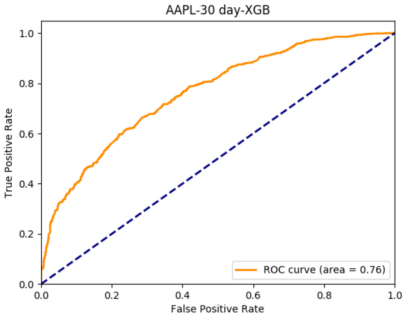

AAPL-90 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.98)

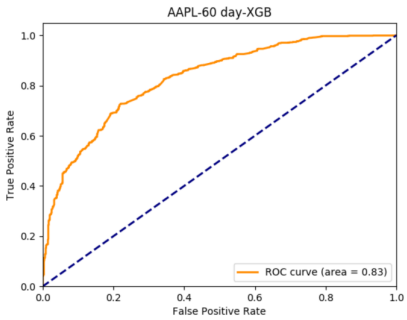

AAPL-3 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.57)

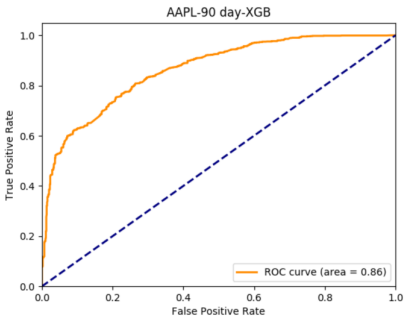

AAPL-5 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.62)


AAPL-10 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.66)

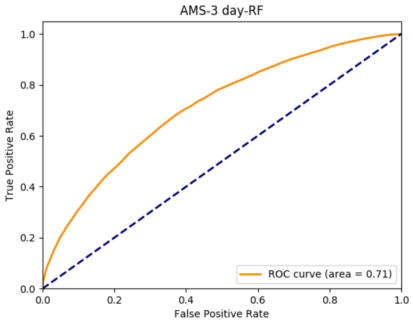

AAPL-15 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.70)

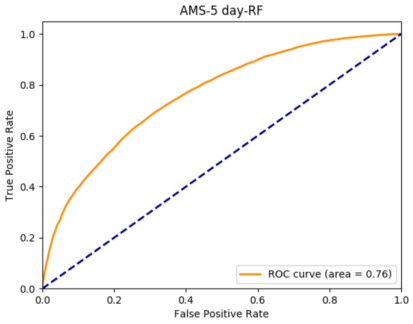

AAPL-30 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.76)

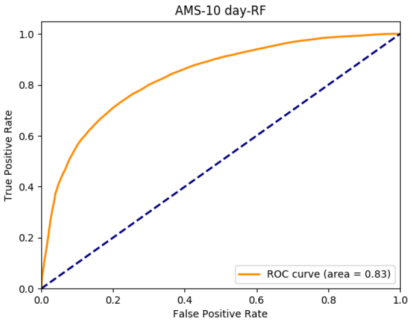

AAPL-60 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.83)

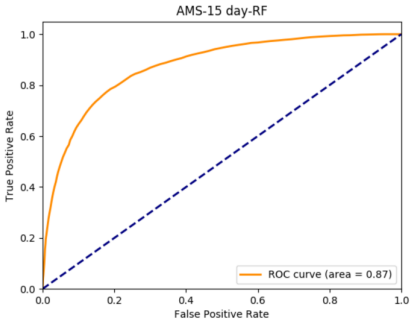

AAPL-90 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.86)

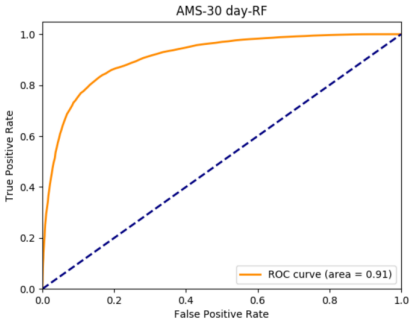
AMS

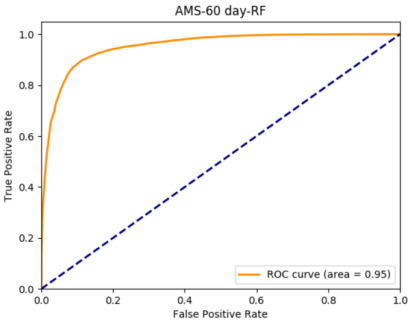

AMS-3 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.71)

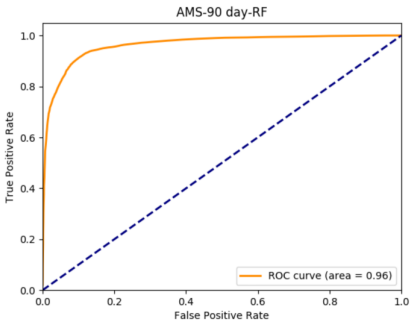

AMS-5 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.76)

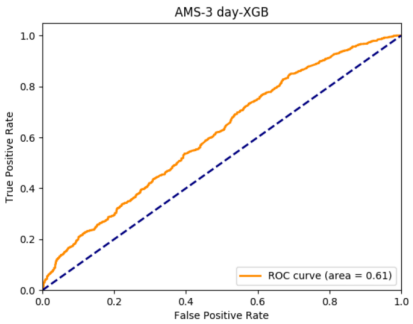

AMS-10 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.83)

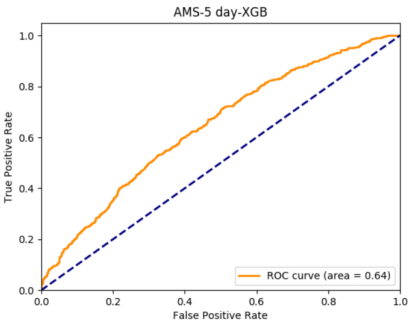

AMS-15 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.87)

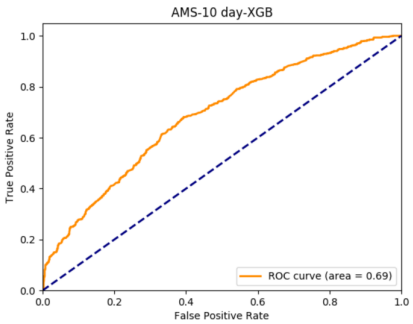

AMS-30 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.91)

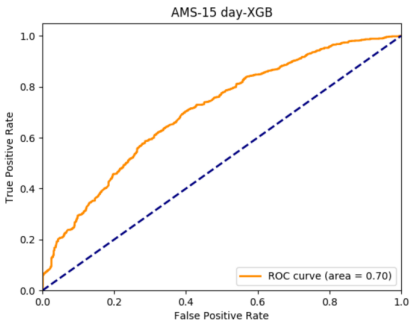

AMS-60 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.95)

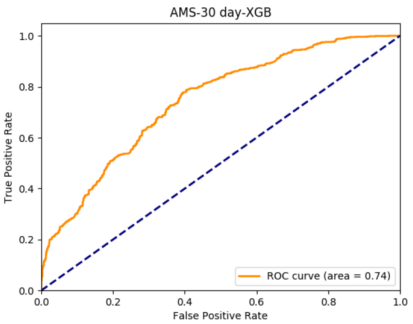

AMS-90 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)

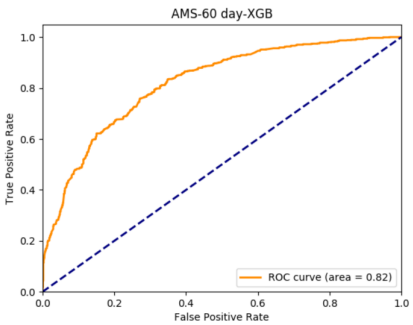

AMS-3 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.61)

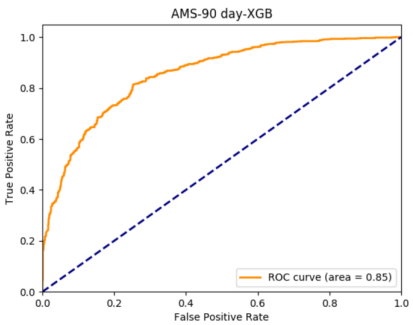

AMS-5 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.64)


AMS-10 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.69)

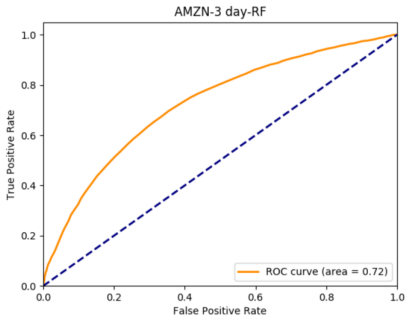

AMS-15 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.70)

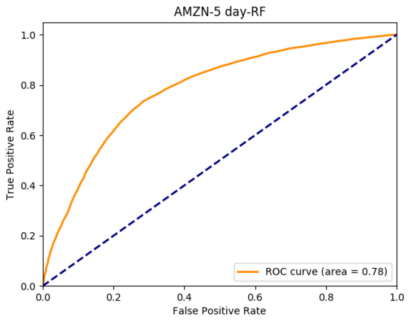

AMS-30 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.74)

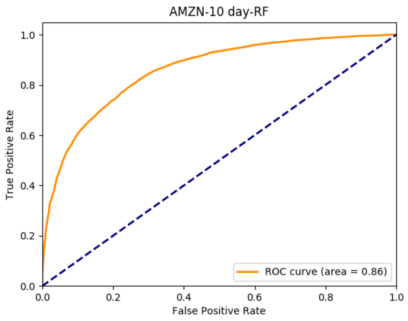

AMS-60 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.82)

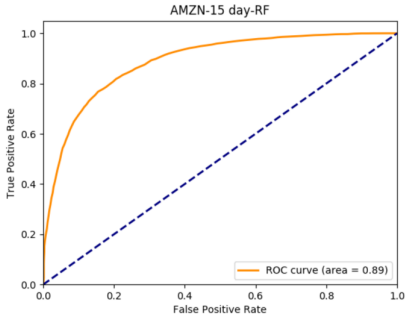

AMS-90 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.85)

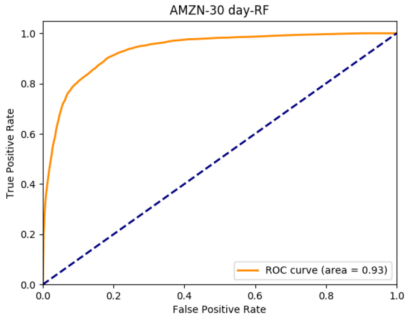
AMZN

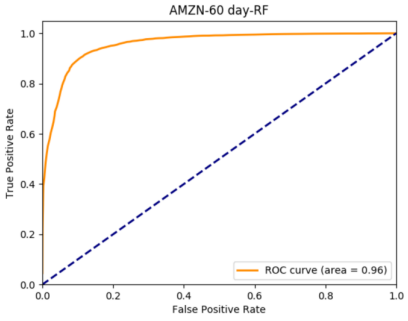

AMZN-3 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.72)

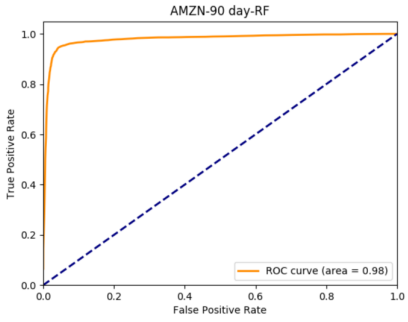

AMZN-5 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.78)


AMZN-10 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.86)


AMZN-15 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.89)

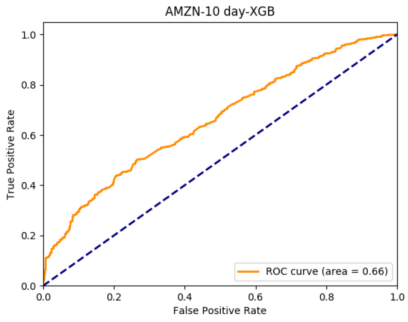

AMZN-30 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.93)

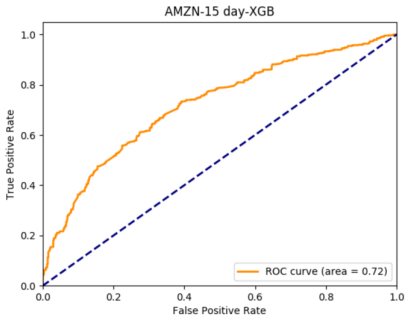

AMZN-60 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)

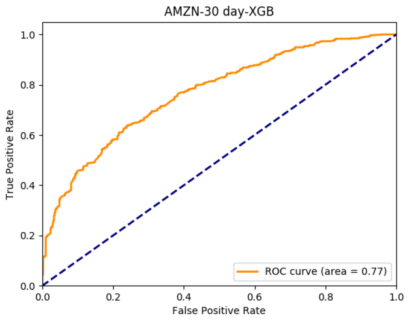

AMZN-90 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.98)

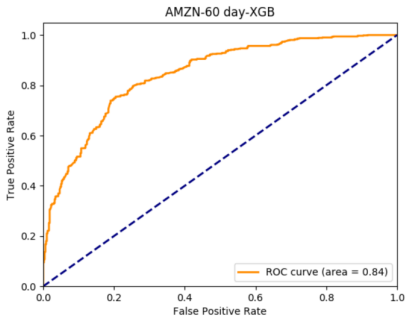

AMZN-3 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.59)

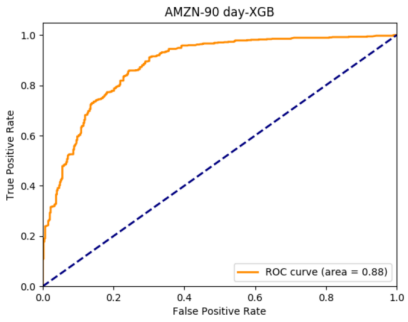

AMZN-5 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.63)


AMZN-10 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.66)

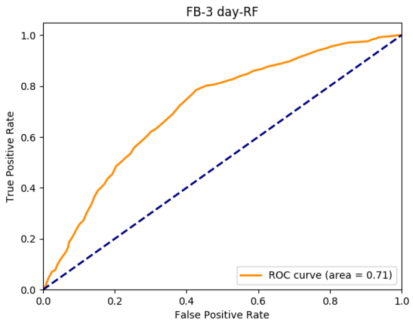

AMZN-15 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.72)

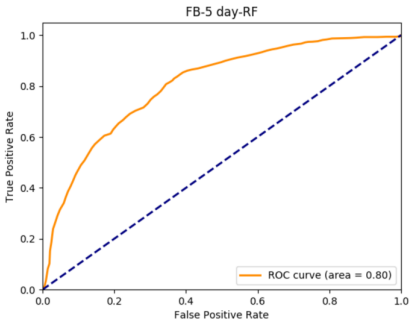

AMZN-30 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.77)

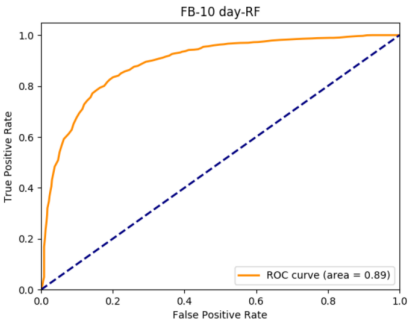

AMZN-60 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.84)

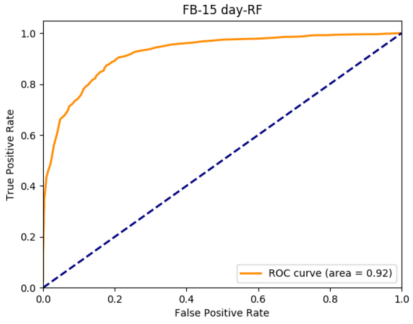

AMZN-90 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.88)

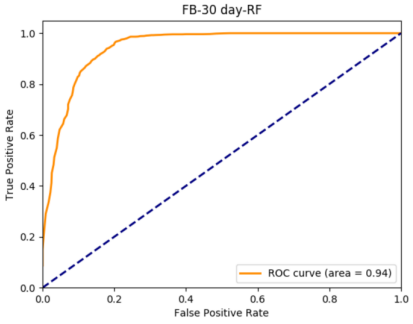
FB

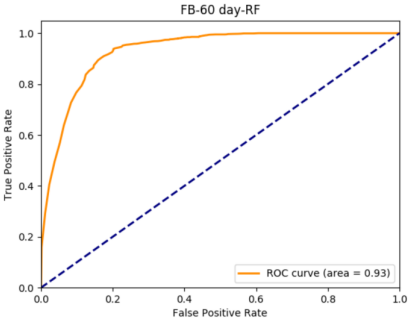

FB-3 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.71)

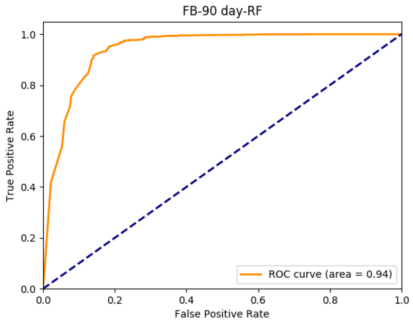

FB-5 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.80)

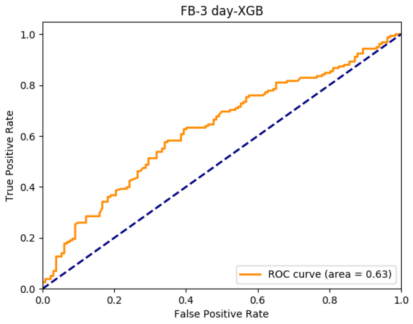

FB-10 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.89)

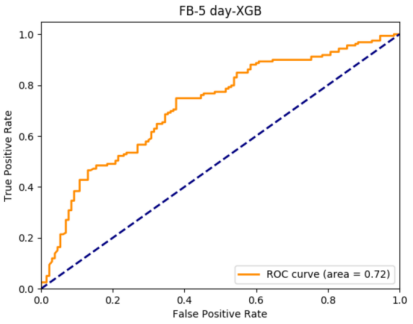

FB-15 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.92)

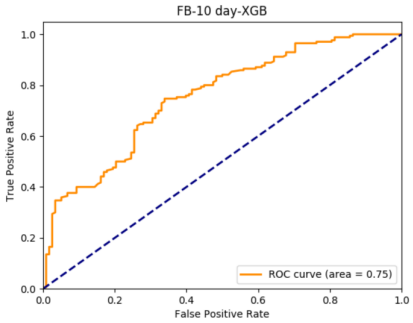

FB-30 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.94)

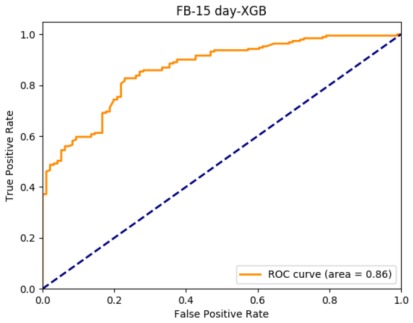

FB-60 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.93)

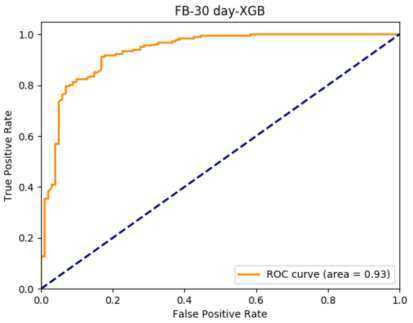

FB-90 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.94)

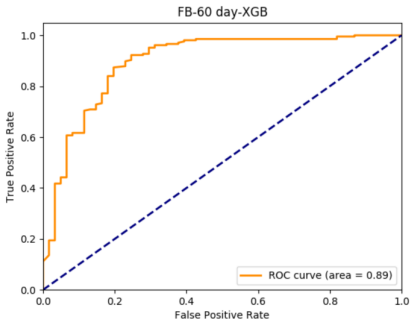

FB-3 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.63)

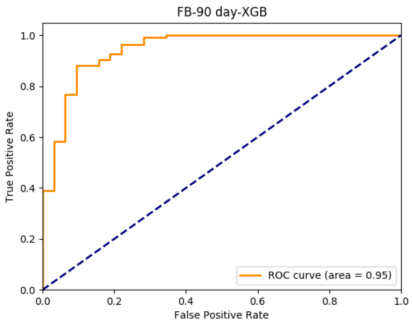

FB-5 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.72)


FB-10 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.75)

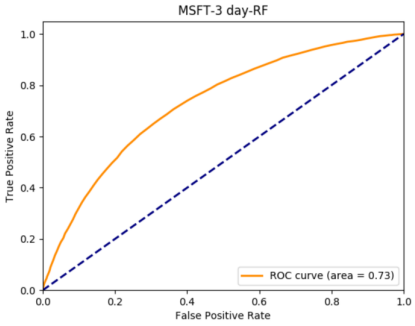

FB-15 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.86)

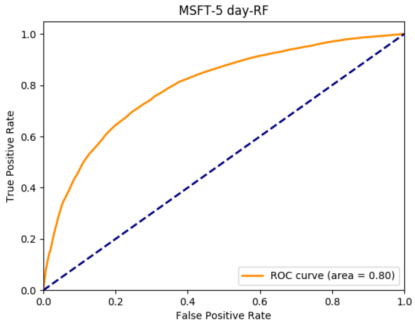

FB-30 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.93)

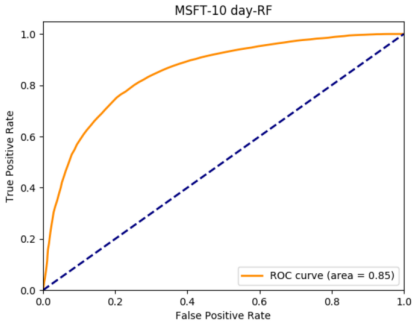

FB-60 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.89)

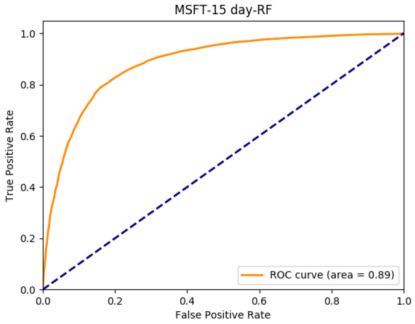

FB-90 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.95)

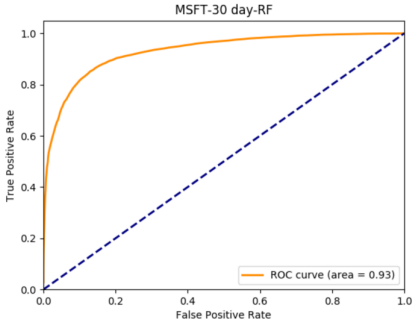
MSFT

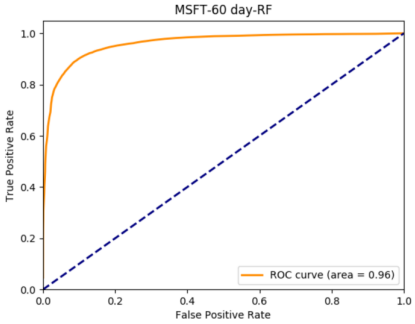

MSFT-3 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.73)

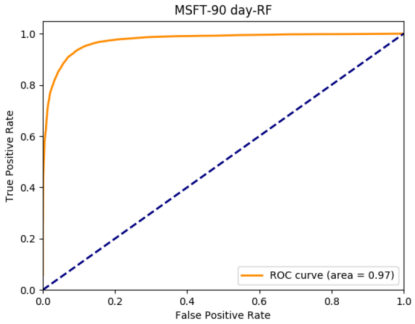

MSFT-5 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.80)

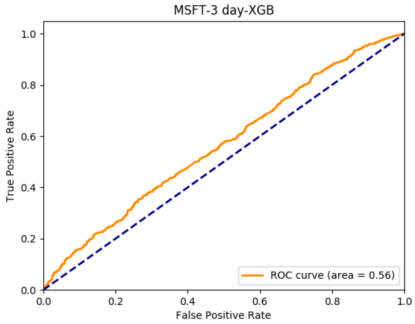

MSFT-10 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.85)

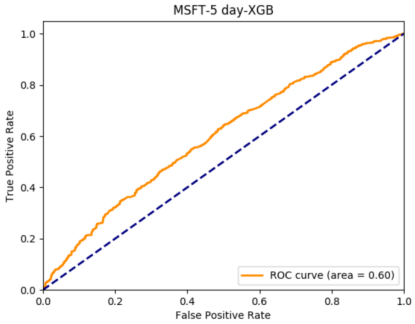

MSFT-15 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.89)

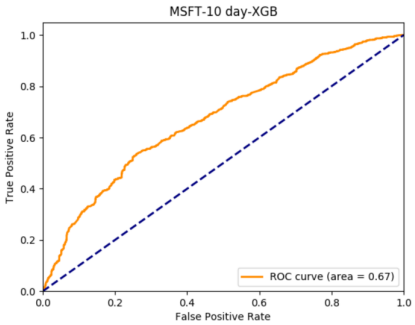

MSFT-30 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.93)

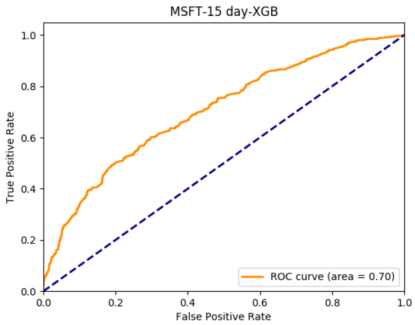

MSFT-60 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)

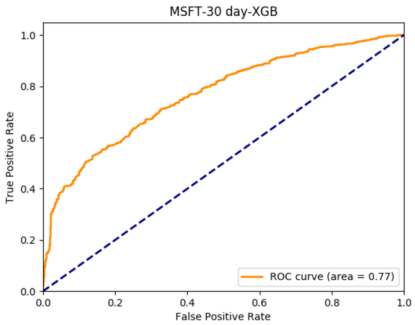

MSFT-90 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.97)

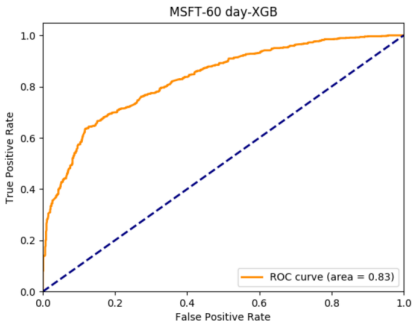

MSFT-3 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.56)

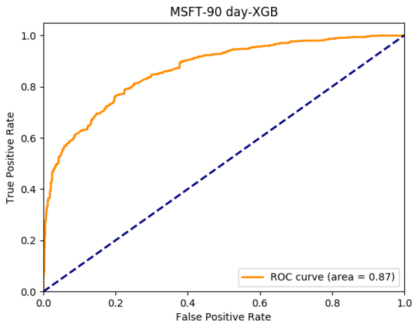

MSFT-5 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.60)


MSFT-10 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.67)

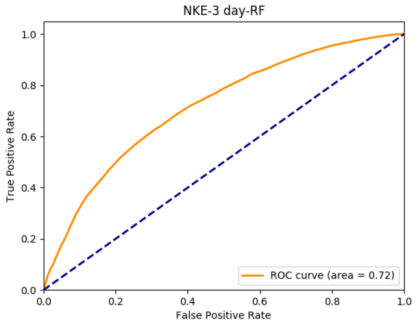

MSFT-15 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.70)

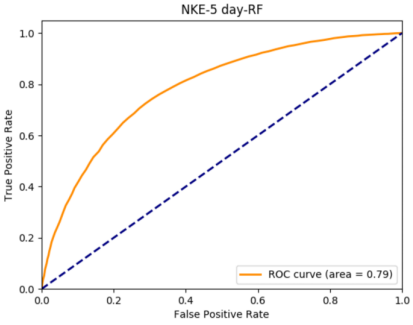

MSFT-30 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.77)

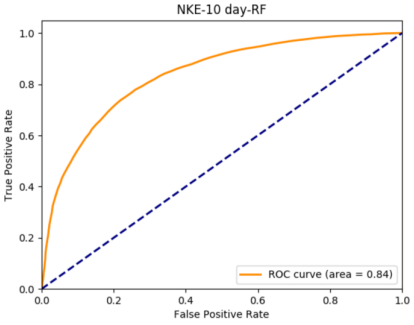

MSFT-60 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.83)

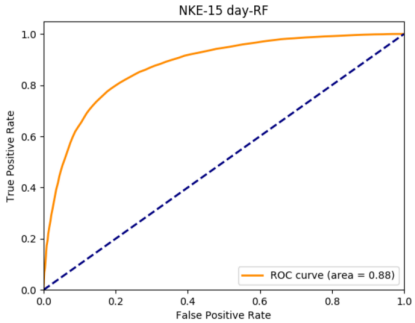

MSFT-90 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.87)

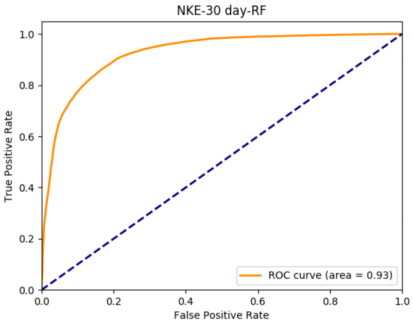
NKE

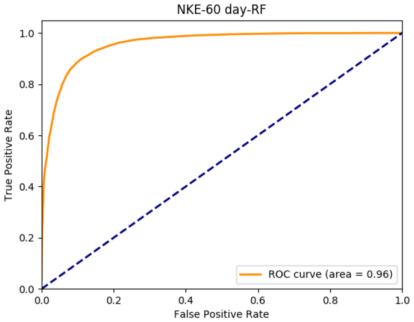

NKE-3 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.72)

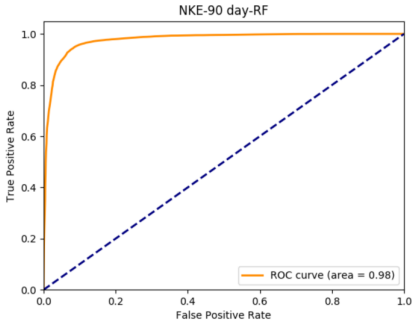

NKE-5 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.79)

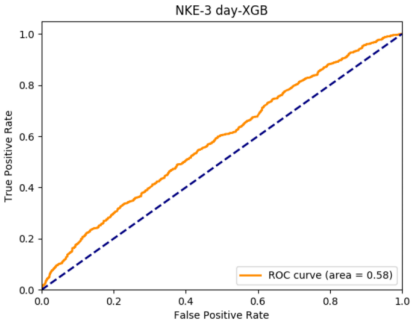

NKE-10 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.84)

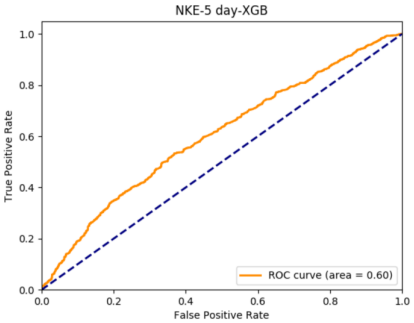

NKE-15 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.88)

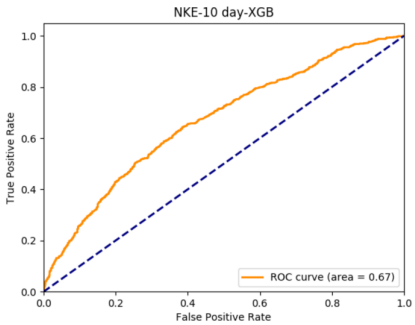

NKE-30 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.93)

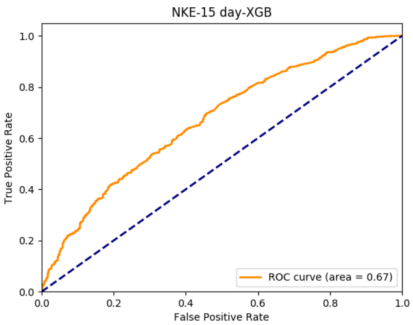

NKE-60 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)

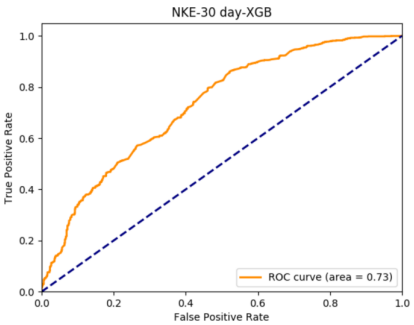

NKE-90 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.98)

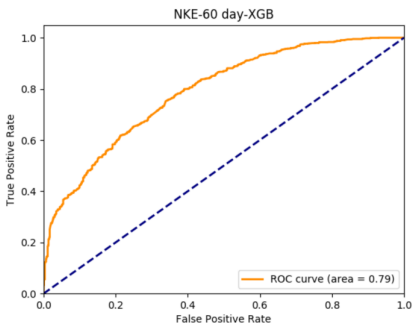

NKE-3 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.58)

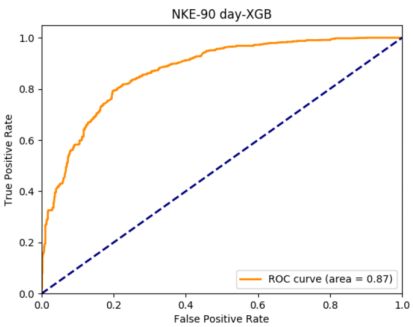

NKE-5 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.60)


NKE-10 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.67)

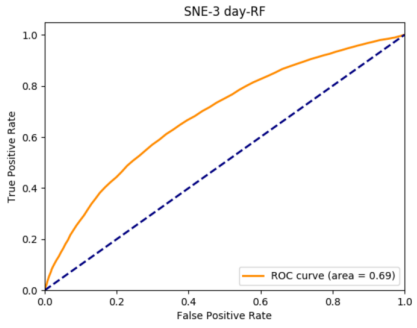

NKE-15 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.67)

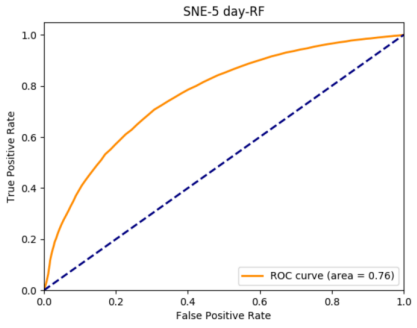

NKE-30 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.73)

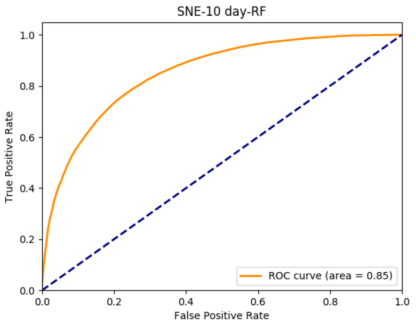

NKE-60 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.79)

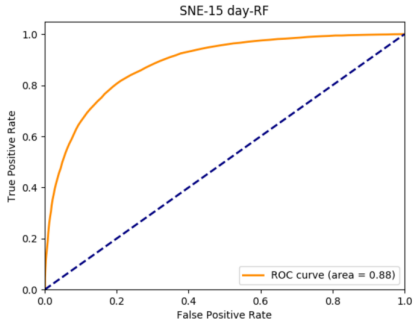

NKE-90 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.87)

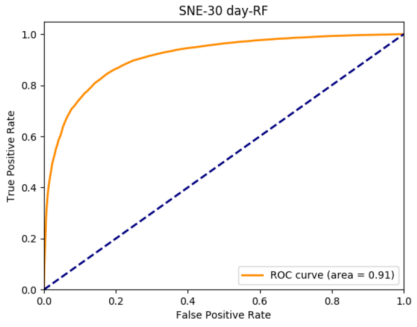
SNE

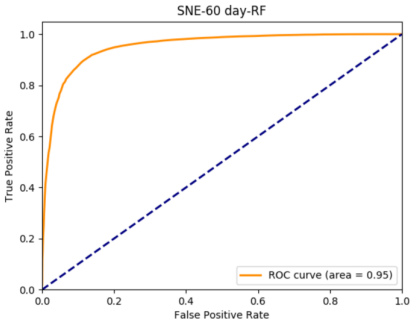

SNE-3 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.69)

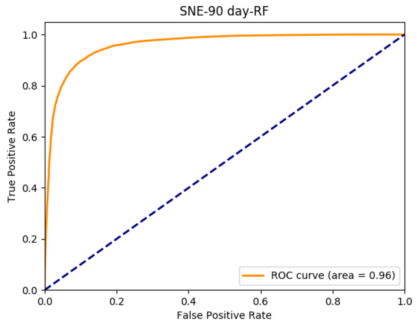

SNE-5 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.76)

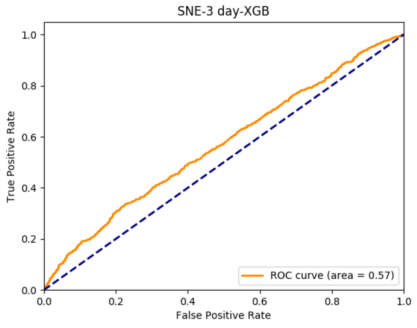

SNE-10 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.85)

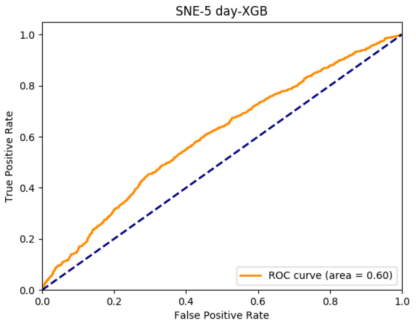

SNE-15 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.88)

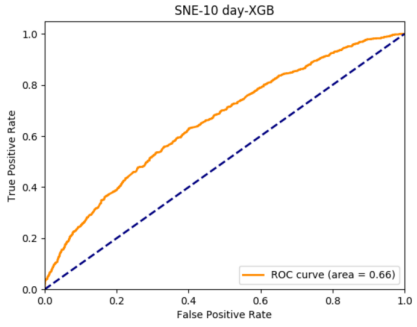

SNE-30 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.91)

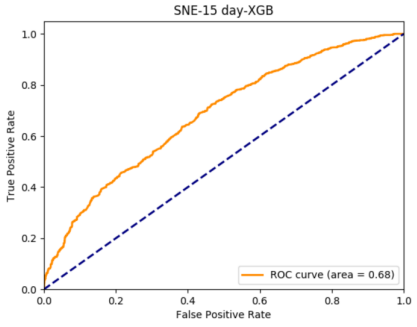

SNE-60 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.95)

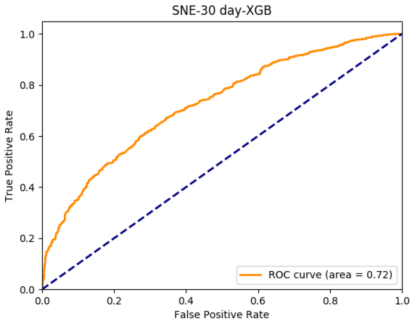

SNE-90 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)

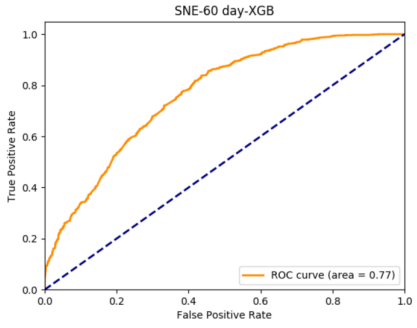

SNE-3 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.57)

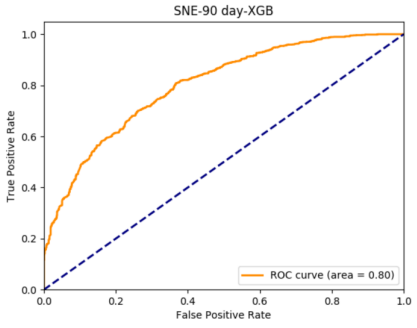

SNE-5 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.60)


SNE-10 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.66)

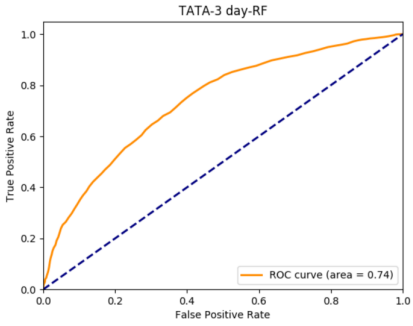

SNE-15 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.68)

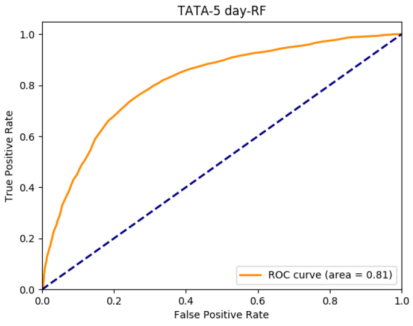

SNE-30 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.72)

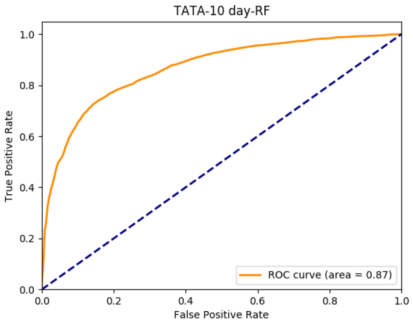

SNE-60 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.77)

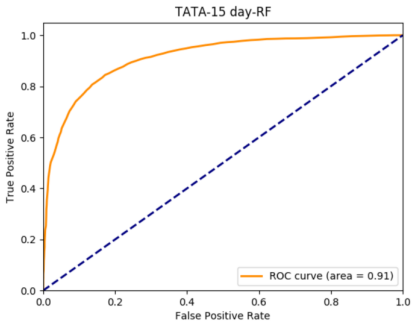

SNE-90 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.80)

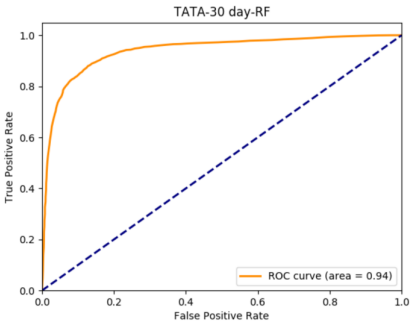
TATA

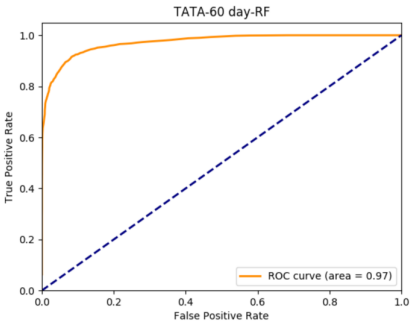

TATA-3 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.74)

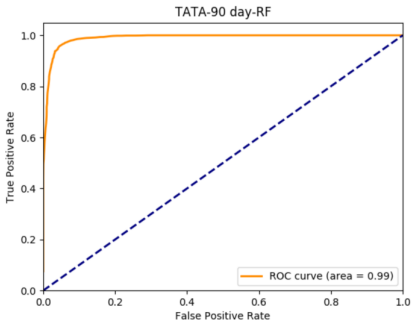

TATA-5 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.81)

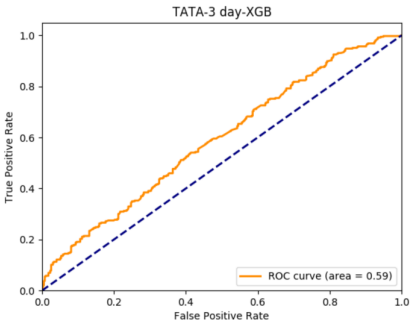

TATA-10 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.87)

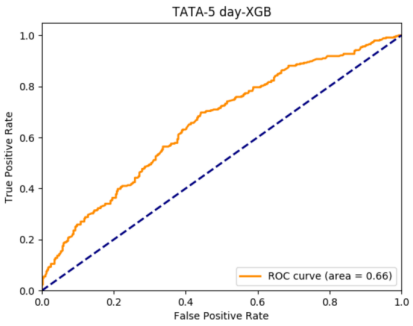

TATA-15 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.91)

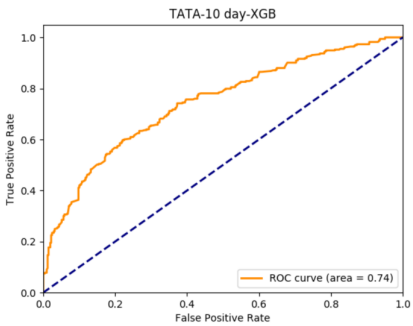

TATA-30 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.94)

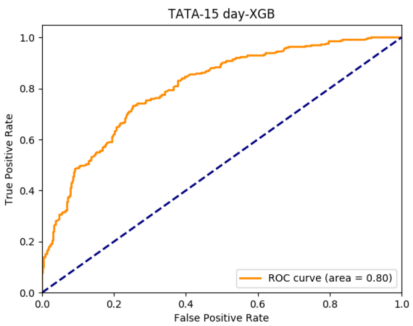

TATA-60 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.97)

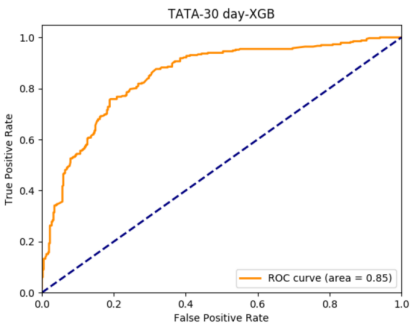

TATA-90 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.99)

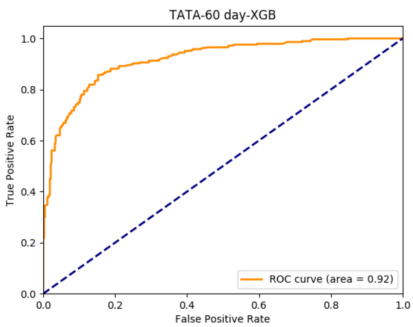

TATA-3 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.59)

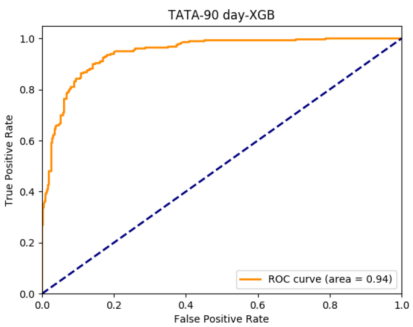

TATA-5 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.66)


TATA-10 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.74)

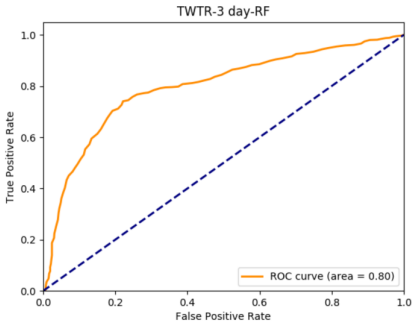

TATA-15 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.80)

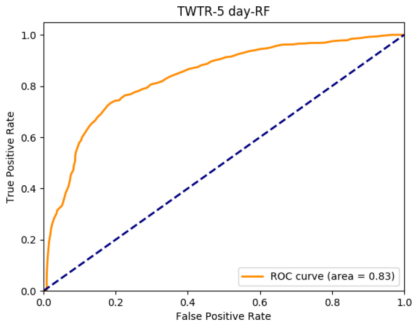

TATA-30 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.85)

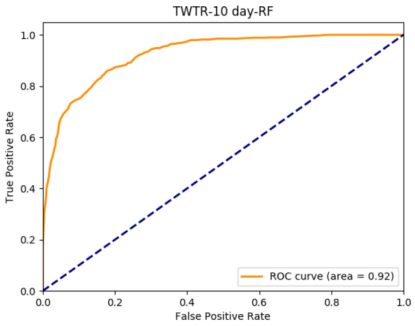

TATA-60 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.92)

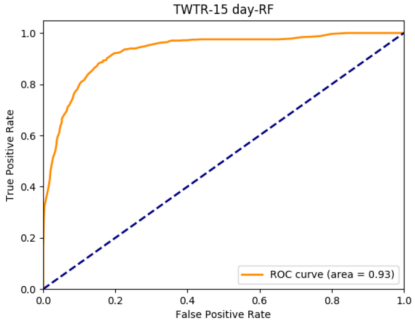

TATA-90 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.94)

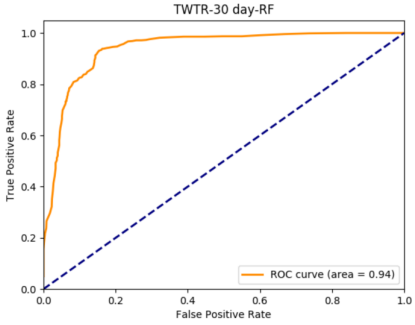
TWTR

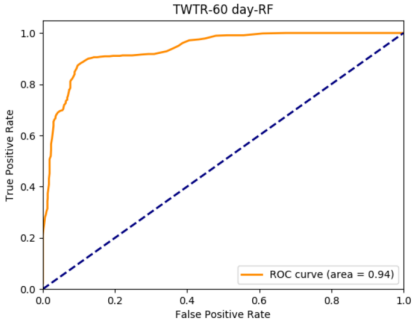

TWTR-3 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.80)

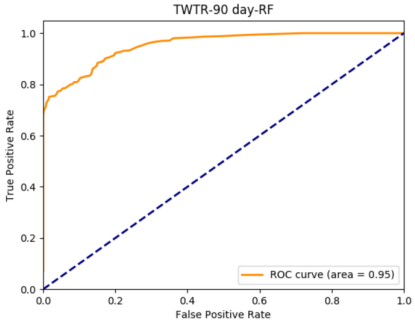

TWTR-5 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.83)

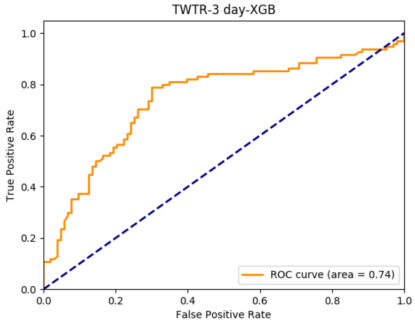

TWTR-10 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.92)

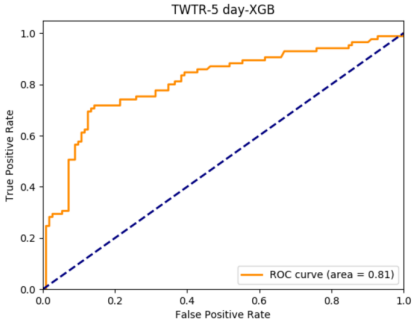

TWTR-15 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.93)

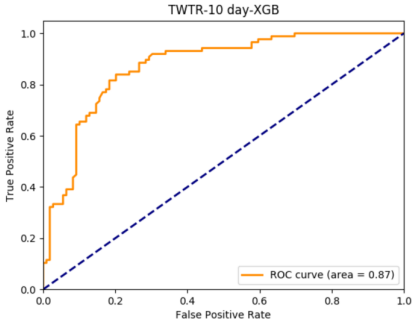

TWTR-30 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.94)

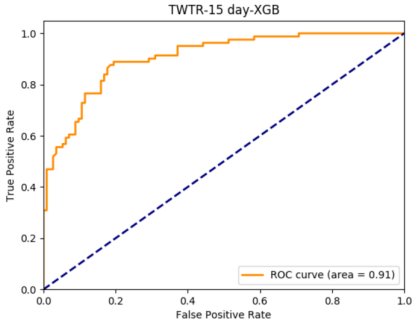

TWTR-60 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.94)

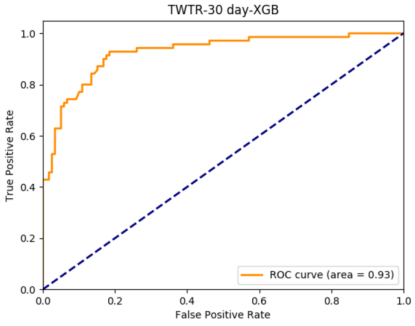

TWTR-90 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.95)

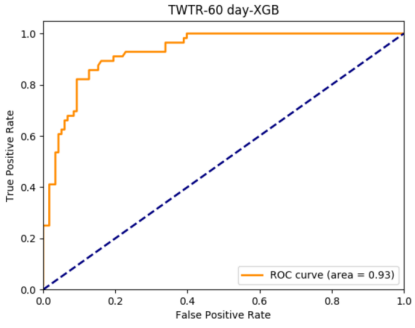

TWTR-3 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.74)

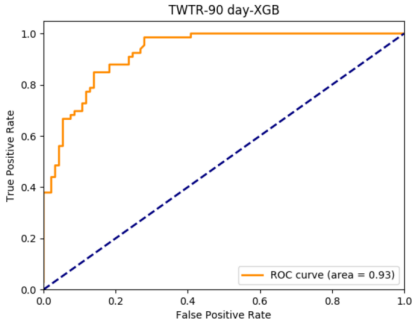

TWTR-5 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.81)


TWTR-10 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.87)

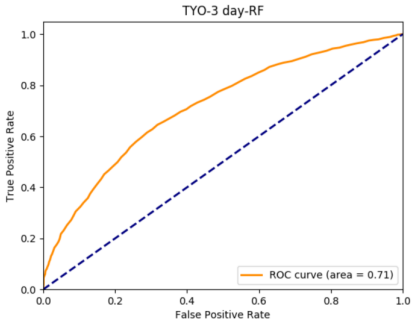

TWTR-15 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.91)

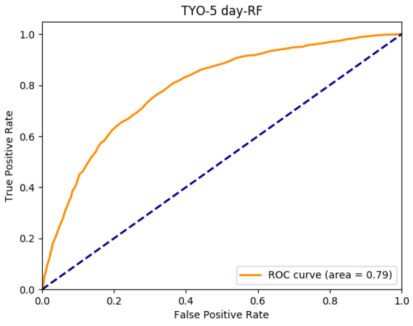

TWTR-30 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.93)

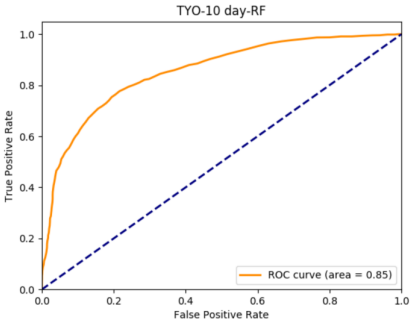

TWTR-60 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.93)

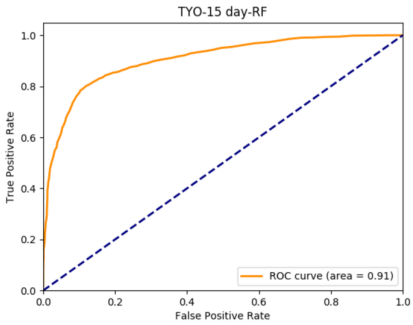

TWTR-90 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.93)

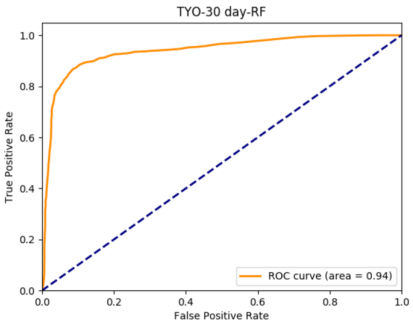
TYO

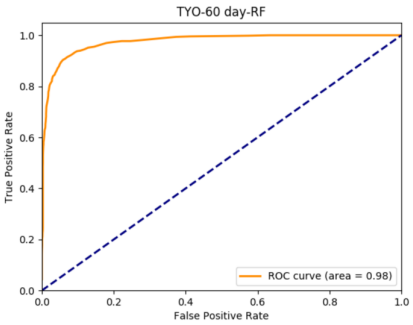

TYO-3 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.71)

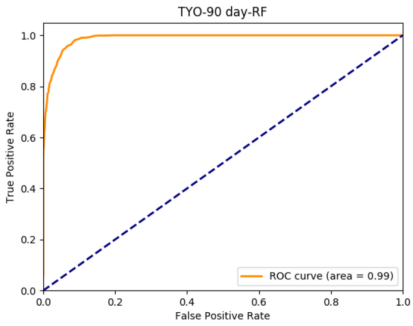

TYO-5 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.79)

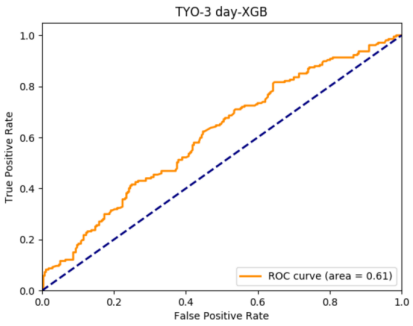

TYO-10 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.85)

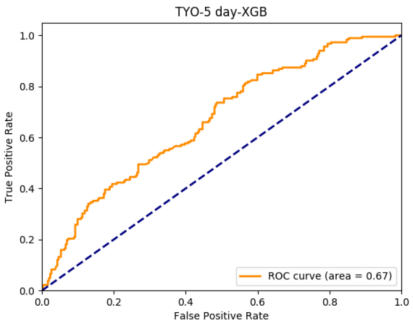

TYO-15 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.91)

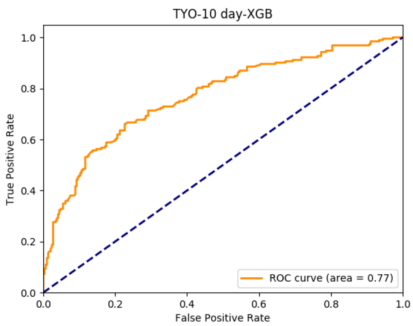

TYO-30 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.94)

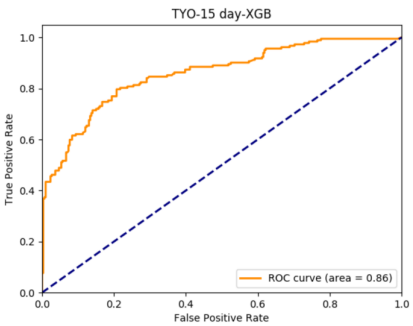

TYO-60 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.98)

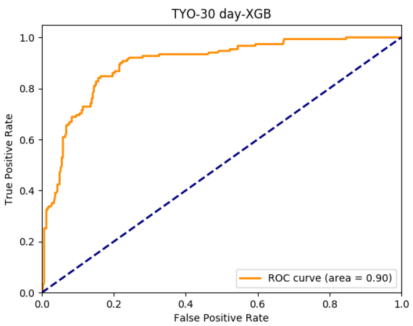

TYO-90 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.99)

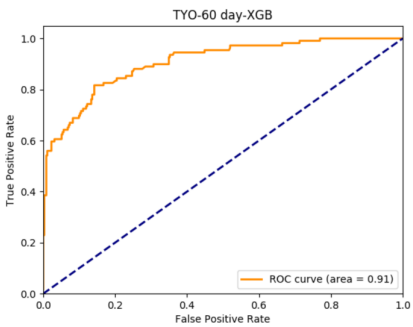

TYO-3 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.61)

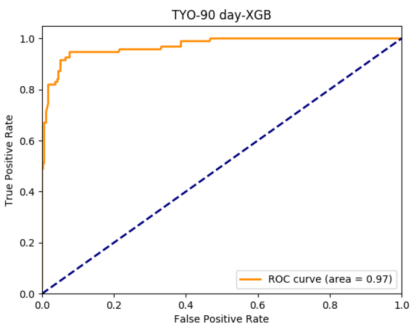

TYO-5 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.67)


TYO-10 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.77)


TYO-15 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.86)


TYO-30 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.90)


TYO-60 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.91)


TYO-90 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.97)

6. An Application of the ML Methods to Pharmaceutical Stocks: A Case Study

It may be an interesting exercise to investigate why certain stocks did not succumb to the aggravated economic crisis, even during the last round of global meltdown. The relevant candidates could easily be found among the pharmaceutical companies. Generally speaking, the health care industry comprising of the life science sectors such as pharmaceutical, biotechnology, and medical devices do not respond significantly to crisis or boom. These are well-known providers of inelastic commodities and services, that do not undergo visible changes in demand when subjected to income and price effects. These products are less exposed to fluctuations typically because health care requirements continue to remain unchanged regardless of economic prosperity or downturns. Indeed, during economic crisis, it is quite possible that people may be even more susceptible to stress and depression, leading to other forms of ailments. However, it cannot be neglected that during economic crisis, the internal readjustments within such companies generating purely supply side effects could lead to loss of production. It is also expected that research and development, which is at the core of innovations and inventions for the pharmaceutical industry get negatively affected. This may have considerable implications for profit and the outcome of stock prices. The principal conjecture is that the randomness or volatility does not affect the prediction accuracy significantly when the time window is expanded. Pharmaceutical stocks may indeed be resilient to shocks (Behner et al., 1998). It is therefore an empirically open question that we attempt to answer through the general procedures followed above.

From a machine learning and methodology point of view, it is an interesting problem to address. The reasons we want to address stocks of pharmaceutical companies are two. First, it is for the purpose of diversification – we wanted to make sure that we address stocks of different types of companies. And second, the low fluctuations in the stocks provides an interesting premise for automation methods for stock trading suggestions and strategies. The results confirm our choice of methods and the higher-than-usual accuracy in turn verifies the nature of the stocks of pharmaceutical companies. The direction predictability, is related to the stocks' variance over time. Naturally, the gains or losses of stocks which are stable would be easier to predict than stocks that are relatively noisy. A lower variance in the data implies a better predictive capability of ML classifiers, and from an economic point of view, it accounts for the stability. For these reasons, we have selected data for one year and have presented the results till $t = 30$ days. Our presumption prior to investigating this was that the convergence would be appreciable and that we'd be able to achieve good results for a shorter trading window. Our results confirm our hypothesis.

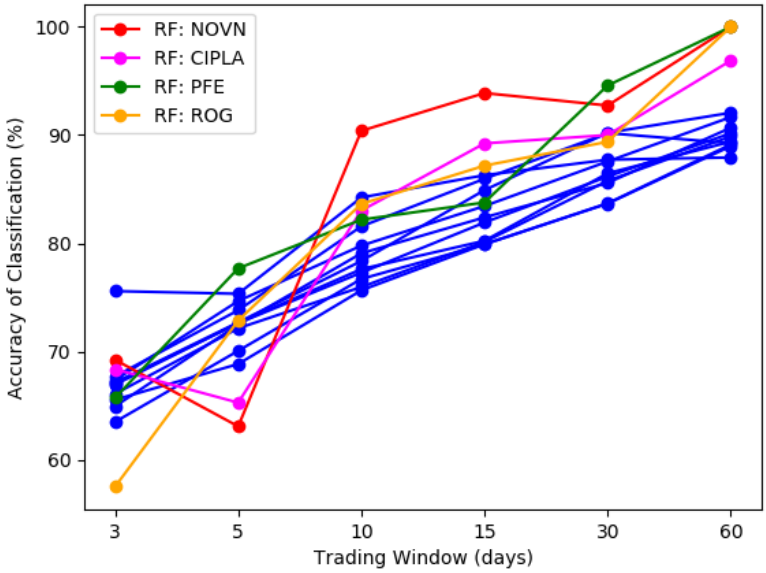


Figure 10: Comparison of the performance of the algorithm on stock data of pharmaceutical companies. The plots in blue represent the accuracies of ten other companies (the results of whose classification are presented in Appendix 1 of the main paper. Prediction accuracy in pharmaceutical stocks is distinctively better.

From Figure 10, it is observed that in the trading window between 10 and 60 day period, the method exhibits better accuracy. This is noteworthy as it establishes the resistance of pharmaceutical companies to market fluctuations. More specifically, the prediction accuracy achieved for the non-pharma organizations (10 data sets considered in our analysis) requires at least 60-day window to match the accuracy in prediction, as compared to the pharmaceutical companies that take only 10-15 day window to match the prediction. Typically, the exercise of predictive analysis in pharmaceutical stocks does not seem to require wide training windows. This is clearly in agreement with our conjecture.

As reported by Behner et al. (1998) (only a few such studies are available since the crisis), unless major policy changes are implemented or completely exogenous factors are in play (such as budget cuts in health care), the pharmaceutical stock prices are less exposed to financial crisis or elasticity. Therefore, understanding and accounting for volatility in pharmaceutical stocks should be easier as compared to other stocks we considered in this paper. Consequently, it is not too hard to comprehend the reason for the remarkable prediction accuracy achieved using shorter time windows. We have presented the results in Tables 11 and 11 and the ROC plots of the pharmaceutical companies after that.

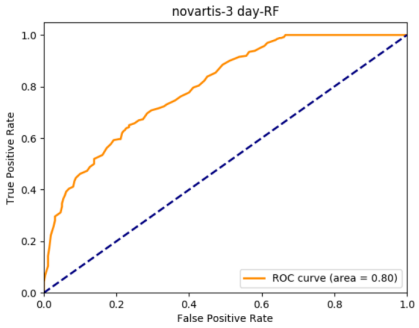
Table 10: RF Results Table for Pharmaceutical Stocks: results of Random Forests implemented on stocks of novartis, cipla, pfizer and roche.

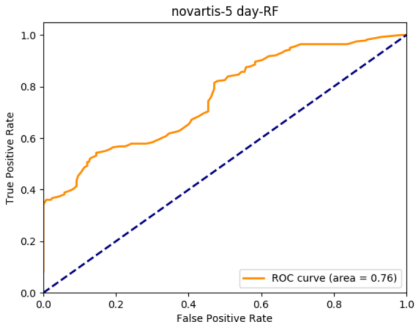
Company Name	Trading Window	Accuracy	Recall	Precision	Specificity	F-Score	Brier Score	AUC
novartis	3	68.46	0.75	0.66	0.62	0.70	0.33	0.79
	5	63.65	0.59	0.69	0.69	0.64	0.38	0.76
	10	90.00	1.00	0.81	0.83	0.89	0.10	0.99
	15	93.88	0.89	0.94	0.97	0.92	0.06	0.95
	30	92.73	0.92	0.83	0.93	0.87	0.07	0.97
cipla	3	69.27	0.72	0.72	0.66	0.72	0.29	0.73
	5	63.45	0.76	0.50	0.56	0.60	0.36	0.76
	10	82.64	0.93	0.79	0.70	0.85	0.19	0.92
	15	89.23	0.91	0.89	0.87	0.90	0.12	0.95
	30	90.43	0.91	0.91	0.90	0.91	0.11	0.92
pfizer	3	66.43	0.50	0.72	0.82	0.59	0.39	0.75
	5	79.11	0.77	0.74	0.81	0.75	0.21	0.91
	10	82.36	0.74	0.87	0.90	0.80	0.18	0.89
	15	83.02	0.71	0.95	0.96	0.81	0.17	0.95
	30	94.38	0.96	0.93	0.93	0.94	0.04	0.99
roche	3	55.00	0.60	0.61	0.48	0.60	0.45	0.61
	5	75.54	0.85	0.78	0.60	0.81	0.25	0.77
	10	83.09	0.93	0.84	0.63	0.88	0.15	0.89
	15	86.42	0.92	0.90	0.67	0.91	0.11	0.89
	30	90.00	1.00	0.90	0.20	0.95	0.10	0.80

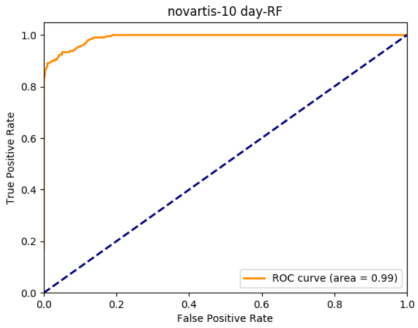
Table 11: XGB Results Table for Pharmaceutical Stocks: results of XGBoost implemented on stocks of novartis, cipla, pfizer and roche.

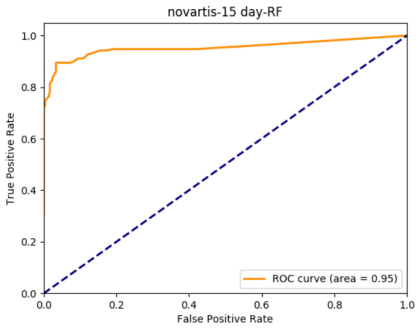
Company Name	Trading Window	Accuracy	Recall	Precision	Specificity	F-Score	Brier Score	AUC
novartis	3	63.46	0.73	0.61	0.54	0.67	0.37	0.74
	5	63.46	0.57	0.70	0.71	0.63	0.37	0.69
	10	90.20	0.95	0.83	0.87	0.89	0.10	0.97
	15	95.92	0.95	0.95	0.97	0.95	0.04	0.99
	30	90.91	0.92	0.79	0.91	0.85	0.09	0.98
cipla	3	65.45	0.67	0.69	0.64	0.68	0.35	0.74
	5	61.82	0.75	0.48	0.54	0.59	0.38	0.65
	10	84.91	0.93	0.82	0.75	0.87	0.15	0.95
	15	86.54	0.93	0.84	0.79	0.88	0.13	0.93
	30	89.36	0.88	0.92	0.91	0.90	0.11	0.91
pfizer	3	66.07	0.56	0.68	0.76	0.61	0.34	0.67
	5	76.79	0.74	0.71	0.79	0.72	0.23	0.83
	10	80.00	0.73	0.83	0.86	0.78	0.20	0.86
	15	84.91	0.74	0.95	0.96	0.83	0.15	0.94
	30	93.75	0.92	0.96	0.96	0.94	0.06	0.98
roche	3	58.93	0.72	0.62	0.42	0.67	0.41	0.61
	5	76.79	0.86	0.79	0.62	0.82	0.23	0.75
	10	85.45	0.95	0.85	0.67	0.90	0.15	0.94
	15	86.79	0.93	0.90	0.67	0.92	0.13	0.88
	30	89.58	1.00	0.89	0.17	0.94	0.10	0.78

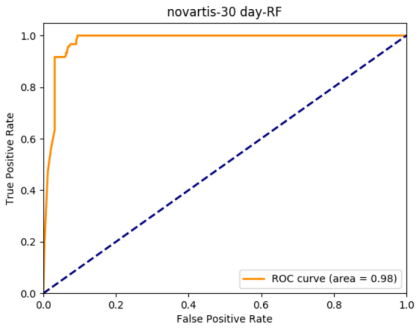
NOVN

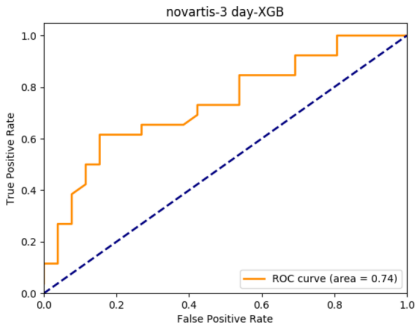

novartis-3 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.80)

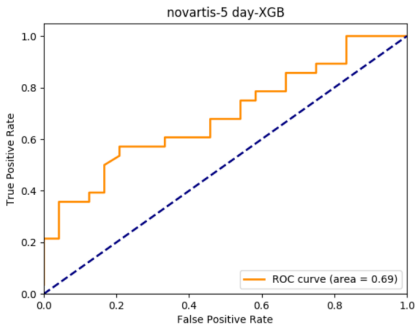

novartis-5 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.76)

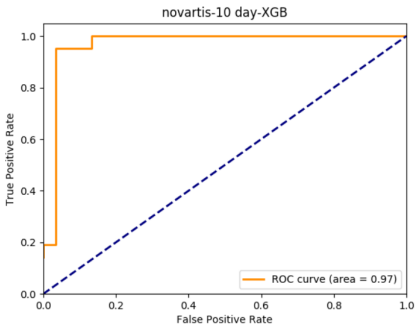

novartis-10 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.99)

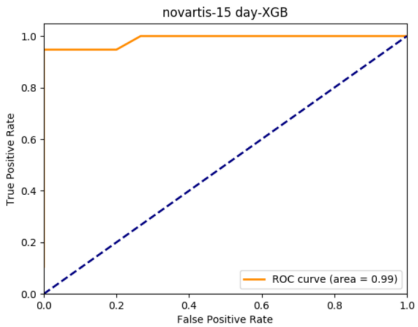

novartis-15 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.95)

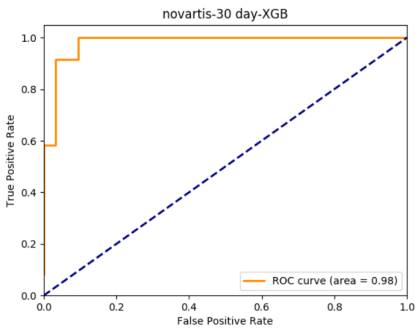

novartis-30 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.98)


novartis-3 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.74)

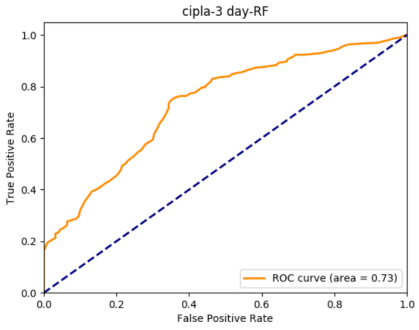

novartis-5 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.69)

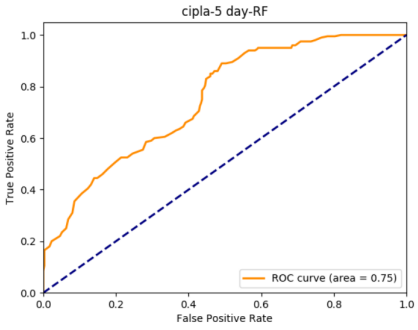

novartis-10 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.97)

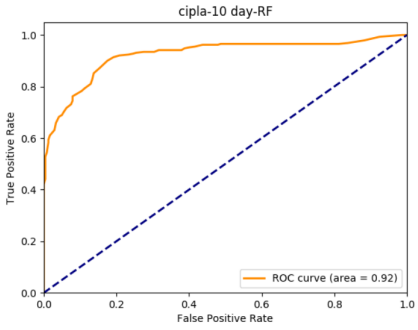

novartis-15 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.99)

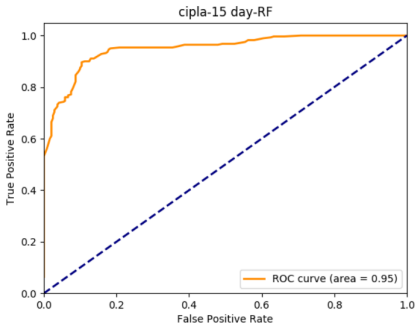

novartis-30 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.98)

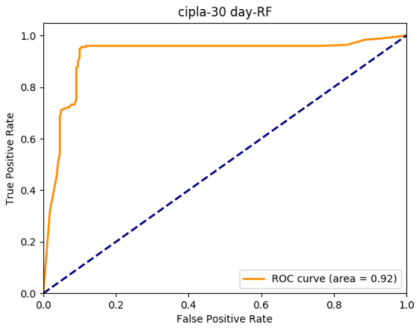
CIPLA

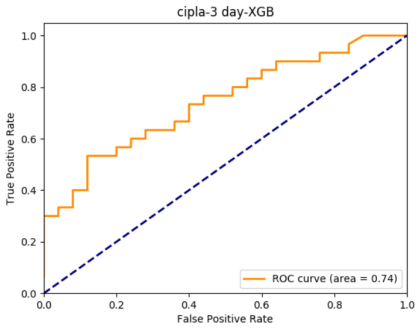

cipla-3 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.73)

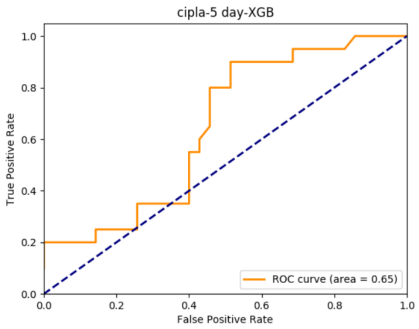

cipla-5 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.75)

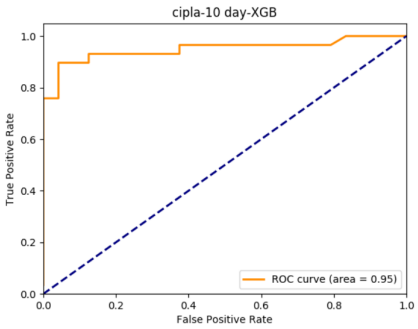

cipla-10 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.92)

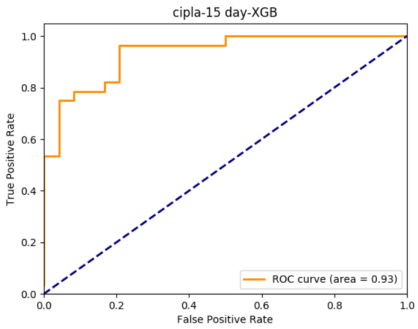

cipla-15 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.95)

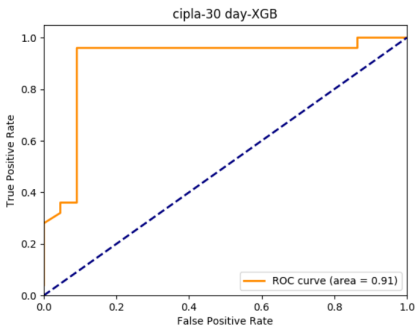

cipla-30 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.92)


cipla-3 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.74)

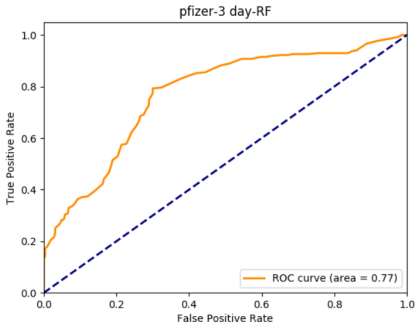

cipla-5 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.65)

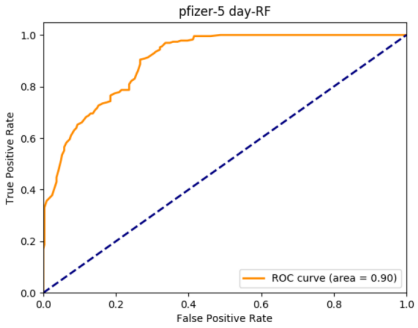

cipla-10 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.95)

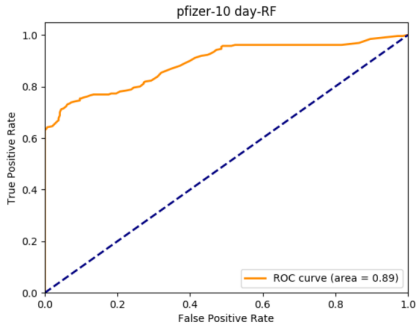

cipla-15 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.93)

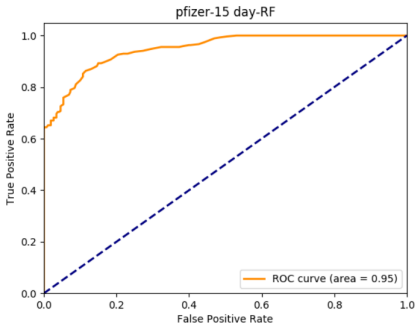

cipla-30 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.91)

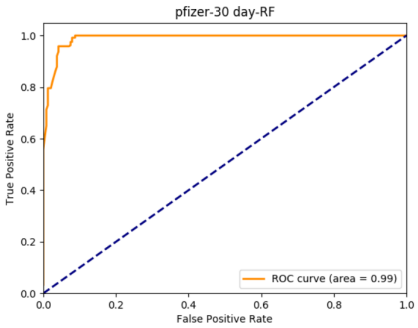
PFE

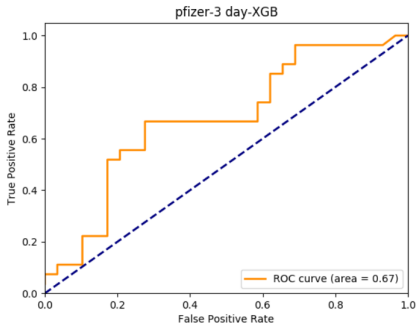

pfizer-3 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.77)

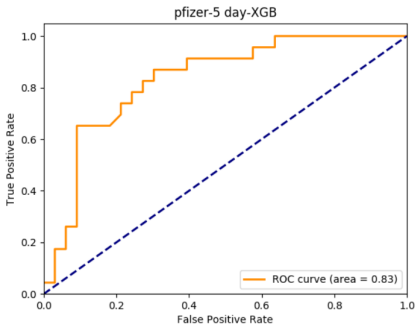

pfizer-5 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.90)

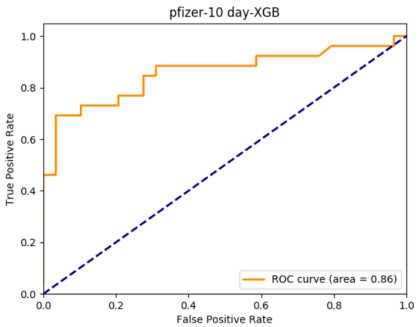

pfizer-10 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.89)

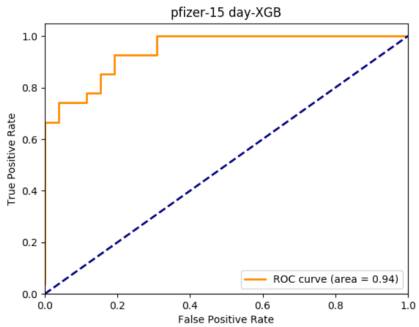

pfizer-15 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.95)

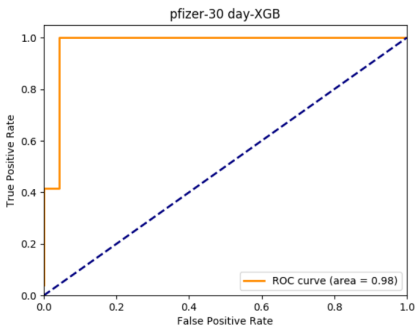

pfizer-30 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.99)


pfizer-3 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.67)

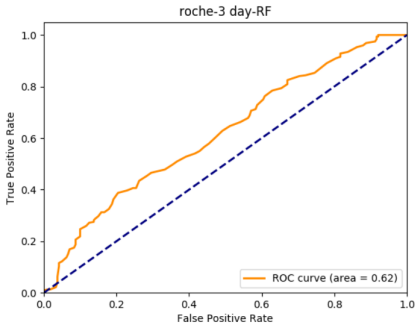

pfizer-5 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.83)

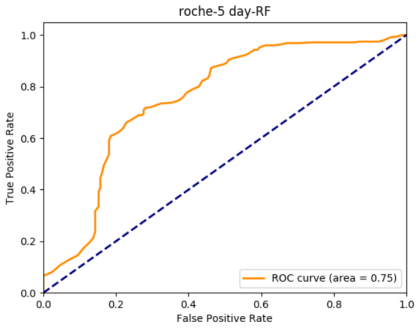

pfizer-10 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.86)

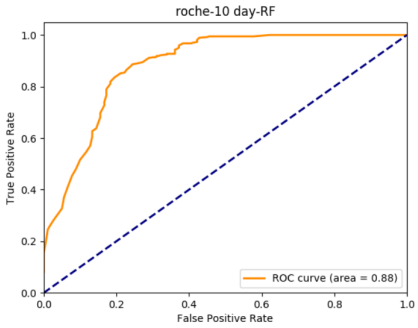

pfizer-15 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.94)

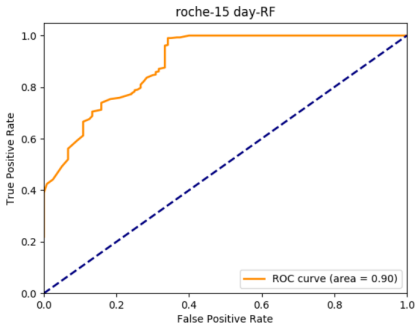

pfizer-30 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.98)

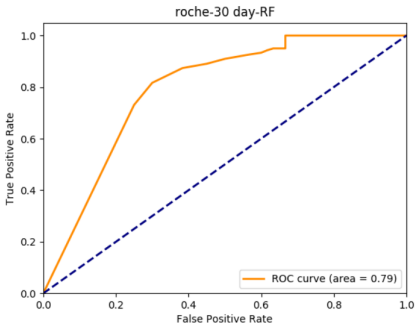
ROG

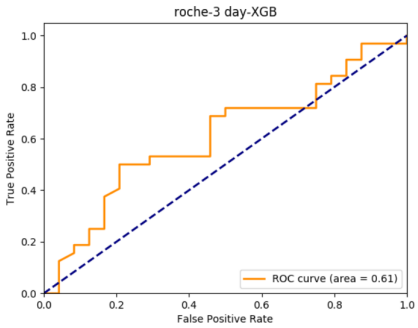

roche-3 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.62)

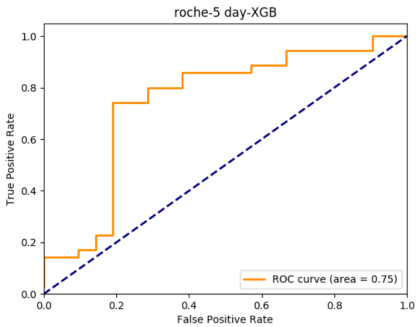

roche-5 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.75)

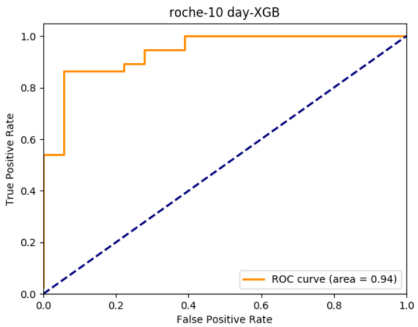

roche-10 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.88)

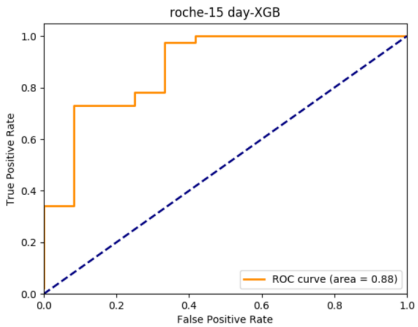

roche-15 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.90)

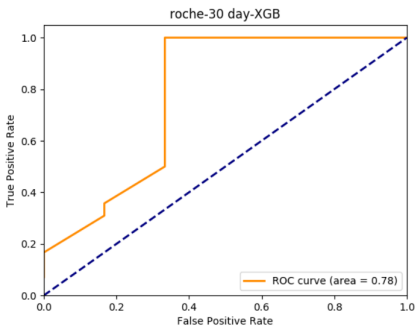

roche-30 day-RF
True Positive Rate
False Positive Rate
ROC curve (area = 0.79)


roche-3 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.61)


roche-5 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.75)


roche-10 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.94)


roche-15 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.88)


roche-30 day-XGB
True Positive Rate
False Positive Rate
ROC curve (area = 0.78)

7. Comparison of Performance

In this section, we present the ROC plots of another popular classifier used in literature: support vector machines (SVM). We present a representative sample to facilitate the understanding of the efficacy of SVM in Figure 11, which is not an exhaustive set of ROCs; we see that unlike the case of RF and XGB, SVM (with a linear kernel) does not exhibit an improvement in performance with the trading window. Furthermore, the ROC curves are very close to the 50% line.

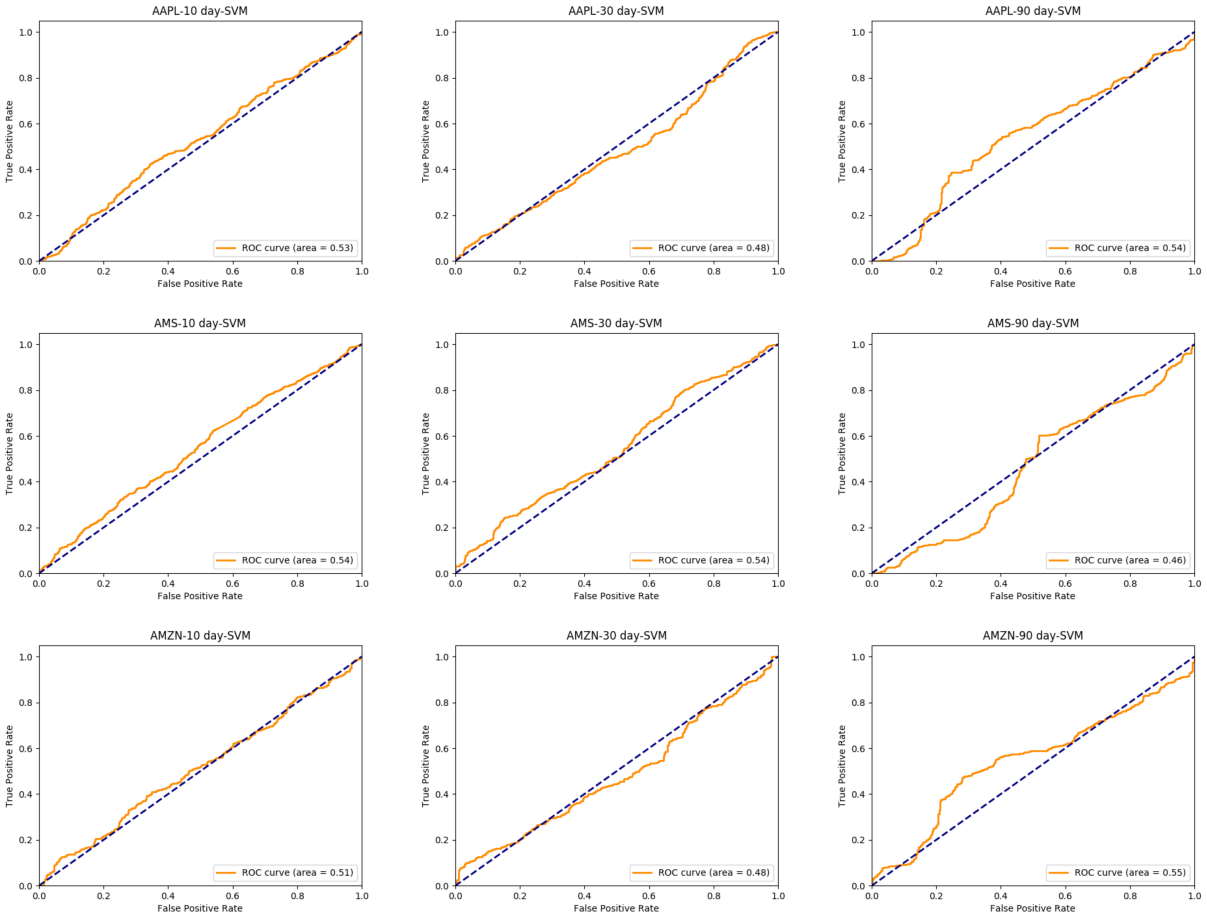


Figure 11: ROC curves of SVM applied to the dataset after preprocessing. From the graphs, it is evident that generally, the efficacy of SVM with a linear kernel is lower than that of RF and XGBoost.

8. References

Breiman, L. (2001), Statistics Department, University of California Berkeley, CA 94720. Random Forests.

Bylander, T. & Hanzlik, D. (1999) Estimating Generalization Error Using Out-of-Bag Estimates. AAAI-99 Proceedings.

Behner, P., Schwarting, D., Vallerien, S., Ehrhardt, M., Beever, C. & Rollmann, D. (2009). Pharmaceutical Companies in the Economic Storm Navigating from a Position of Strength. Technical Report: BooZ & Co Analysis.

Geurts, P., & Louppe, G. (2011). Learning to rank with extremely randomized tree. JMLR: Workshop and Conference Proceedings, 14, 49–61.

Horowitz, E., Sahni, S. & Anderson-Freed, S. (1992). Fundamentals of Data Structures in C, W. H. Freeman & Co., New York, NY, USA, ISBN = 0716782502

Quinlan, J. R. (1986). Induction of Decision Trees, MACH. LEARN 1, pp 81–106