15.0621 Final Project Report

Fiona Fan

Wellesley College ‘17

**Prediction of Action on Stock within Healthcare Sector by Mining ZZAlpha’s Stock Recommendations and Daily Stock Features**

1. **Abstract and Introduction**

Ever since the inauguration of Affordable Care Act, the healthcare industry has gained 1 million new jobs. Its momentum to grow has sustained over the past 5 years. The mandate, insurance exchanges and Medicaid expansions have given 12 million more Americans insurance coverage (NYTimes, 2015). Despite the pressure to cut costs, innovations that are transforming the industry, like mHealth and telemedicine, and relevant policies do and will sufficiently sustain the growth of the healthcare sector (Diamond, 2015; Bureau of Labor Statistics, 2014). Currently traded at a P/E of 40.33 and with a market cap of 110620 Billion, the healthcare sector is projected to grow over the next 3 years (MSFT, 2013).

The sustained growth of the healthcare sector gives me the motivation to look at which stocks to buy or sell within the sector. This project explores the right action to take (long, short or do nothing) on stocks in healthcare sector, given the daily stock attributes (opening price, closing price, adjusted closing price, lowest price, highest price and daily traded volume). More simply put, the posited question is whether one made the right decision of longing or shorting the stock based on recommendations made by ZZAlpha, a machine learning stock recommendation firm.

ZZAlpha is a firm specialized in recommending stocks recommendation. Their portfolios have historic annualized returns over 15%, exceeding publicly recognized benchmarks by 200 basis points. Risks are also well behaved (Pratt, 2015). Thus, I am trusting that the stock picks (whether to long or short a certain stock) by ZZAlpha can yield a profitable outcome that yields more return than the average market return.

1. **Data Source**

The dataset for this project is procured from UCI Machine Learning Repository, and donated by ZZAlpha Ltd. The dataset consists of recommendations made by ZZAlpha for various US traded stock portfolios the morning of each day during the 3-year period Jan 1, 2012 - Dec 31, 2014. The given dataset consists of txt files (one for each day) with each line in the following format:

Jan 04 2005\_006 Big\_100\_5\_LONG\_SHORT\_F.pdf, L, AA 0.959 =25.97/27.09, AMAT 0.950 =14.70/15.46, EBAY 0.930 =53.33/57.31, PFE 0.995 =19.84/19.95, UPS 0.980 =71.72/73.16, Avg of 5 = 0.963

The above indicates recommendations were made before market open on Jan 4, 2005. This portfolio was limited to the biggest 100 cap stocks and was of size 5. It was for 'L' or long recommendations. The five stocks recommended are shown by ticker, result, price at sale divided by price at purchase. The average for the five is shown.

1. **Data Cleaning**

By asking this question of whether one should long or short the stock, looking at each stock’s daily attributes, I only have to extract one feature from the given dataset, which is ZZAlpha’s recommendations, and get the other features (daily stock prices and volumes) from Yahoo Finance (MSFT, 2013).

The merged txt file is sized 115.7MB. Since multiple recommendations are made each day, much duplicate information is existent. After cleaning, 558 stocks were mentioned in the recommendations made from Jan 2012 to Dec 2014. To keep the dimensions of the data tractable, I randomly selected 50 stocks. Out of the 50, only 44 had data stream of historical prices from Yahoo available on Python Pandas (Appendix II). Thus the dataset is composed of 44 stocks in total. I used Matlab for matching the recommendation data and the historical stock features, ranked chronologically. The cleaned data is a 1X44 cell array with each cell a 755X11 matrix (*cell­\_array*). Each matrix stands for a single stock. There are 755 market days and 10 features included in the dataset. The 7 features used for prediction are *open* and *close* prices, daily *high* and *low* prices, adjusted close prices (*adj\_close*), percentage change in stock prices (*pctchange*, (close-open)/open)) and *volume*. The outcome variables are whether the recommendation made is *long* or *short*.

Since the data is time-series data, I will be training the models on 2012-2013 data and validate the model on 2014 data (*trainingset*, *validationset*).

1. **Descriptive Statistics.**
   1. **Predictor Examination**

Figure 1 shows a time series plot of adjusted close price, percentage change in daily price and daily traded volume for the 45 stocks in healthcare sector. The adjusted close prices show that most stocks are in an uprising trend (Figure 1a), consistent with the overall healthcare sector trend. The percentage changes in daily prices show the volatilities of the stocks. There are some spikes, indicating that major changes, like mergers and acquisitions, happened to the companies on that day (Figure 1b). The time series data of volumes traded show the traffic of the stocks. We can see from Figure 1c that there seems to be a general bump in traffic during May-Aug 2012, which can be explained by the favorable Supreme Court’s ruling towards ACA.

* 1. **Target Variables Examination**

Figure 2 shows ZZApla’s recommendation percentage distribution. We can see that long and short recommendations are made with similar frequencies. The mean percentage for long recommendation is 4.67% with a standard deviation of 3.84% and the mean for short recommendation is 4.3% with a standard deviation of 3.26%. Due to the small size of target data (longs and shorts) per stock, I decided not to oversample and instead try an array of cutoff values.

1. **Models**
   1. **Logistic Regression**

Since I have categorical outcome variables while my input variables are continuous, the first model I tried was logistic regression. I tried four types of link functions. Link function defines the relationship f(mu) = x\*b between the mean response mu and the linear combination of predictors x\*b. The four types of link functions that I tried were:

'logit' f(mu) = log(mu/(1-mu))

'probit' f(mu) = norminv(mu)

'comploglog' f(mu) = log(-log(1-mu))

'loglog' f(mu) = log(-log(mu))

For each link function, I ran the LR model through a range of cut-off values and chose the one that yielded minimum overall error, as defined by:

(#misclassification of 1 to 0 + #misclassification of 0 to 1)/total data points. Then I looped through the 44 stocks. Each stock has its own unique cut-off value. Figure 3 and 4 shows the ROC curves of LR model. We can see that, first, the different link functions don’t matter very much-- the ROC curves for different link functions look approximately the same. Second, LR model only gives more accurate predictions for a few stocks; for the majority of the stocks, LR model doesn’t work very well especially when predicting short recommendations.

The incapability of logistic regression can also be shown in Figure 5, which is a distribution of error rates and success rates. Success rates are defined as (#corrected identified target class/ total number of target class). The mean overall error rate is 50.3%, which is no different from random assignment. The mean success rate is 10.6% for long calls and 0.8% for short sells.

* 1. **Classification Tree**

The intuition behind using classification tree comes from the instinct that historically speaking, when a stock’s certain attributes fall into a specific range, the stock is more likely to be recommended for shorting or longing.

I started with a full tree (Figure 6 and 7). To better the performance of my decision tree model, I decided to try out different numbers of minimum number of leaves. As shown in Figure 8, I tried 1~35 for minimum numbers of leaves and calculated the cross validation error and resubstitution error as measures of performance of the tree. Cross validation error gives a good estimate of the predictive accuracy of the resulting tree by testing new trees on new data. Resubstitution error is the difference between the response training data and the predictions the tree makes of the response based on the input training data. Optimally both measures should be small. Combining Figure 8 and the complexity of the tree (when the minimum number of leaves in the tree is too high, the tree becomes too simplistic to have a meaning), I chose 3 as the minimum number of leaves for ‘long’ tree and 1 as the minimum number of leaves for ‘short’ tree for further analyses. The ‘long’ tree is the same as the full tree, and the new ‘short’ tree is shown in Figure 9.

Figure 10 shows the ROC curves of the optimized tree. Like logistic regression, the classification tree works well with some stocks but not for the majority. In total it seems to perform as good as or worse than random assignment.

To check this hypothesis, I looped over the 44 stocks. The average overall error rate is 20.54%, which is better than the yielded by logistic regression. The average success rate for long calls is 3.54%, which is worse than logistic regression, and 3.86% for short sells, which is better than logistic regression.

* 1. **Neural Net**

Since the relationship between the predictors and the outcome is not straight forward, I decided to use neural net to bypass the hidden relationship.

I first looped through 1~100 numbers of hidden layer on the first stock, ‘ABBX’. As it turns out, the minimum error rate of around 3% occurs when there’s only one hidden layer. Thus for application of neural net on the rest 43 stocks I am using only 1 hidden layer as well.

Next, I looped through all 44 stocks. Since in Matlab neural net operates on its own tool box so I couldn’t figure out how to plot multiple ROC curve on the same graph, I decided to pick a representative stock yielding minimum overall error rate, which is an indication of working well with the neural net model. The stock is BBIB. As shown in Figure and, the neural net model seems to work better for short recommendations than long ones. To testify for this hypothesis, I looped through the 44 stocks again. As it turned out, the average success rate for long call is around 5.1%, not as good as logistic regression but better than classification tree, and the average success rate for short call is around 6.2%, the best among the three. BBIB might not be a representative stock that works best with neural net, but the neural net model does seems to work better for short calls than for long calls. And the average overall error rate is 21.8%, a little worse than classification tree.

1. **Summary and Future Directions**

This project aims to use daily stock prices to classify the stock as a ‘short’, ‘long’ or ‘no action’. Logistic regression, classification tree and neural net were attempted to perform the task. Logistic regression performs best at classifying ‘long’ recommendations with a success rate of 10.6%, and neural net gives best prediction of ‘short’ recommendations with a success rate of 6.2%. Classification tree and neural net are around the same level of giving general predictions with an overall error rate of around 20%.

Overall, all three models give very poor prediction of the recommendations made by ZZAlpha with the highest success rate not exceeding 10%. One of the most salient reasons might be that daily stock features alone are not enough to capture the desirableness of a specific stock. Even though this is a time-series dataset, I am only using the time attribute to partition the data. My justification for not using time series for prediction lies in the fact that in a perfect market, the aggregate past performance of the overall market should bear no predicting power over the market’s future.

If I had more time (and wishfully more knowledge about machine learning), I would include more market features like S&P500 and NASDAQ index, ß of each stock, and dummy variables of the stock’s appearance and sentiment (positive or negative) in the news for prediction. I would also use a time window, instead of a time point, to mine the features in the past in search of similar attributes to predict outcome.

*Disclaimer: For project proposal I proposed using EEG data to predict eye blinks. In that proposal I suggested using one dataset to build the model and cluster the other one. If one of the clusters fit the model well then that cluster would refer to eye blinks. However, in trying to implement the model I built in the first dataset in the second dataset, I found that the second dataset is too raw for comparison (have to be filtered through signal processing), and I couldn’t find a comparable dataset on the internet. Thus I switched topic for my final project. I hope it’s not too much inconvenience!*

**Appendix I. Bibliography**

Diamond, D. (2015). Healthcare Jobs Just Grew At Fastest Pace Since 1991. Retrieved December 9, 2015, from http://www.forbes.com/sites/dandiamond/2015/06/05/hospitals-jobs-growth-is-suddenly-booming/

Employment Projections: 2014-24 Summary. (2015, December 8). Retrieved December 9, 2015, from http://www.bls.gov/news.release/ecopro.nr0.htm

Healthcare Sector Snapshot - NYTimes.com. (2015). Retrieved December 9, 2015, from http://markets.on.nytimes.com/research/markets/usmarkets/sectors.asp?sector=56

Healthcare Stocks Predictions & the Best Healthcare Stocks to Buy | InvestorPlace. (n.d.). Retrieved December 9, 2015, from http://investorplace.com/hot-topics/healthcare-stocks/#.VmhT--MwhE4

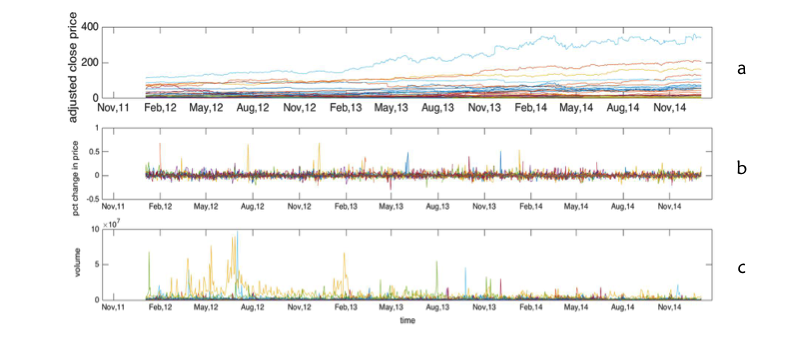
Microsoft Corp. (MSFT) (2013). Profile, business summary. *Yahoo!Finance*. Retrieved from  <http://finance.yahoo.com/q/pr?s=MSFT>.

Pratt, Kevin B. "Proof Protocol for a Machine Learning Technique Making Longitudinal Predictions in Dynamic Contexts." *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015.

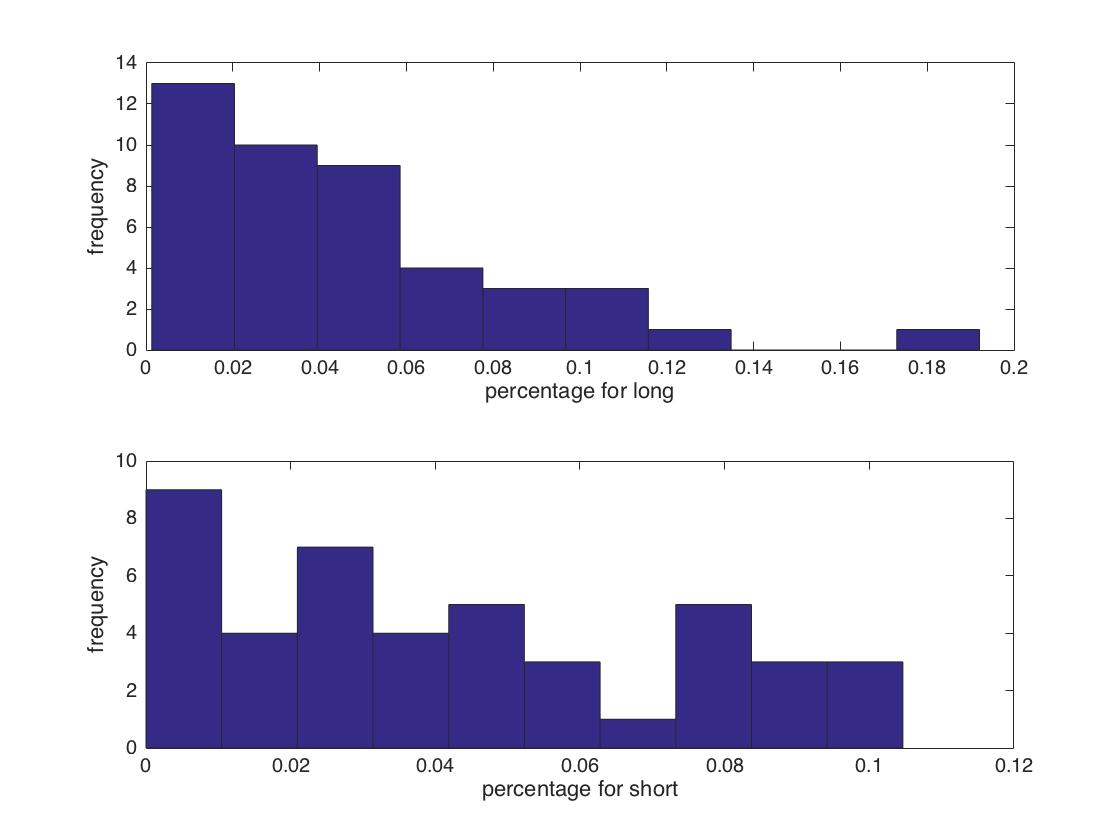
**Appendix II. Name of Tickers**

{'ABAX', 'ABC', 'ACUR', 'ADK', 'ALC', 'ALKS', 'AMLN', 'AMSG', 'ANAC', 'APRI', 'ARNA', 'AXGN', 'BIIB', 'CLDX', 'CLVS', 'CNMD', 'COO', 'CORT', 'CPIX', 'CPTS', 'DCTH', 'DGX', 'DXCM', 'ECYT', 'ELGX', 'ELOS', 'ENDP', 'ENZN', 'ETRM', 'EW', 'GALT', 'GTIV', 'GTXI', 'HMA', 'HWAY', 'INCY', 'INSY', 'IVC', 'KERX', 'LCAV', 'LH', 'MAKO', 'MAXY', 'MCK'};

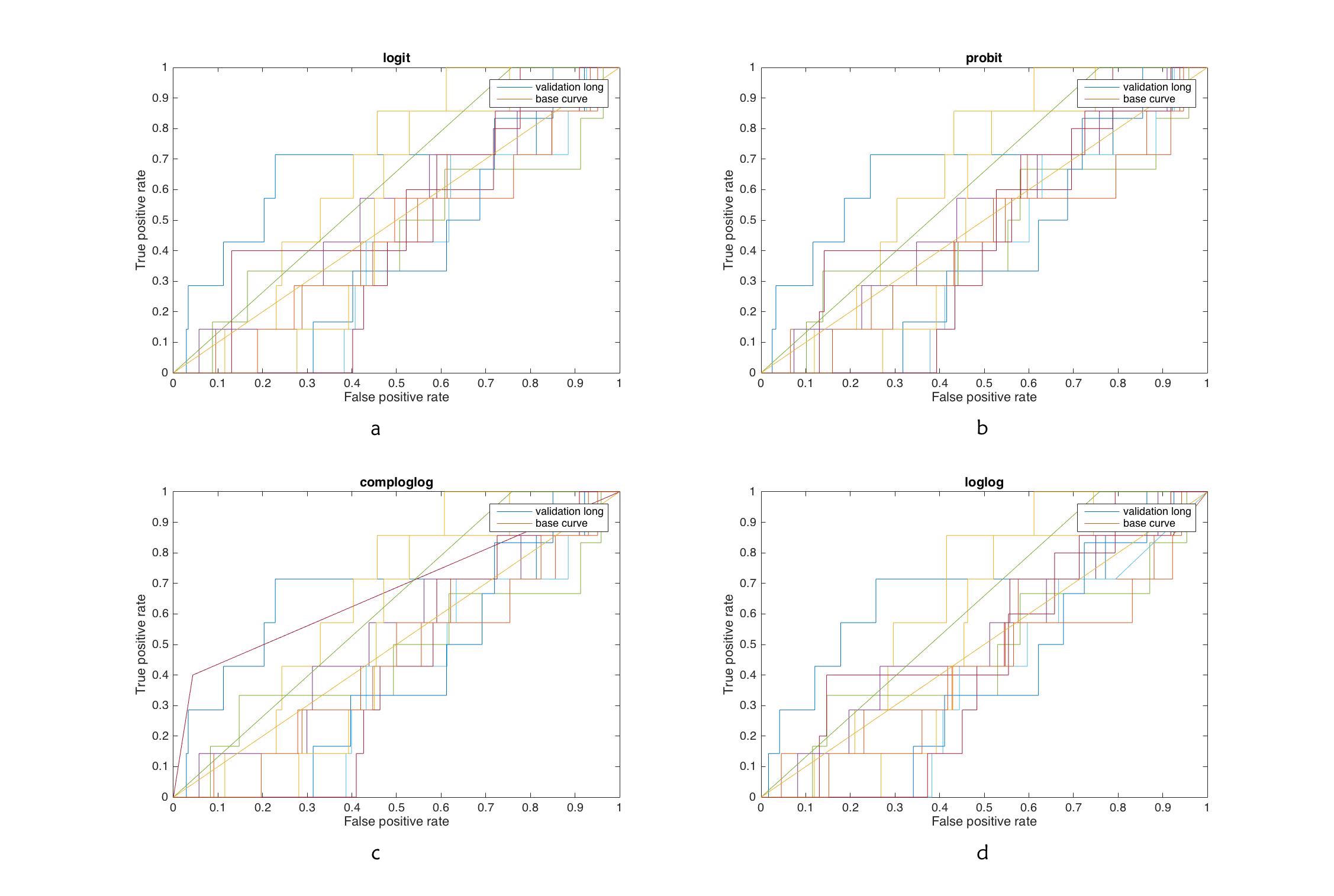
**Appendix III. Figures and tables**



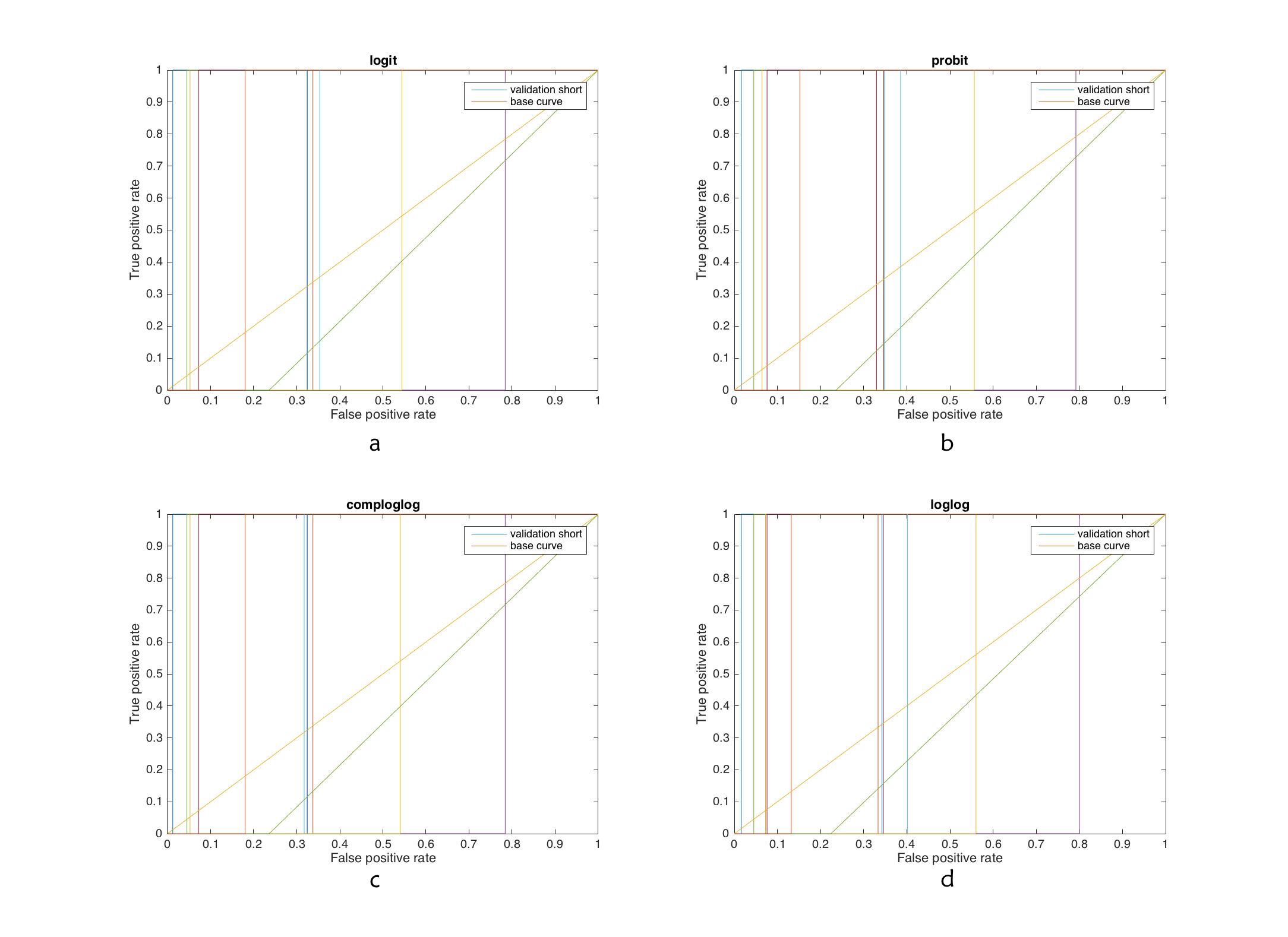
**Figure 1.** Time series plot for adjusted closing price, percent change in daily prices, and volume traded



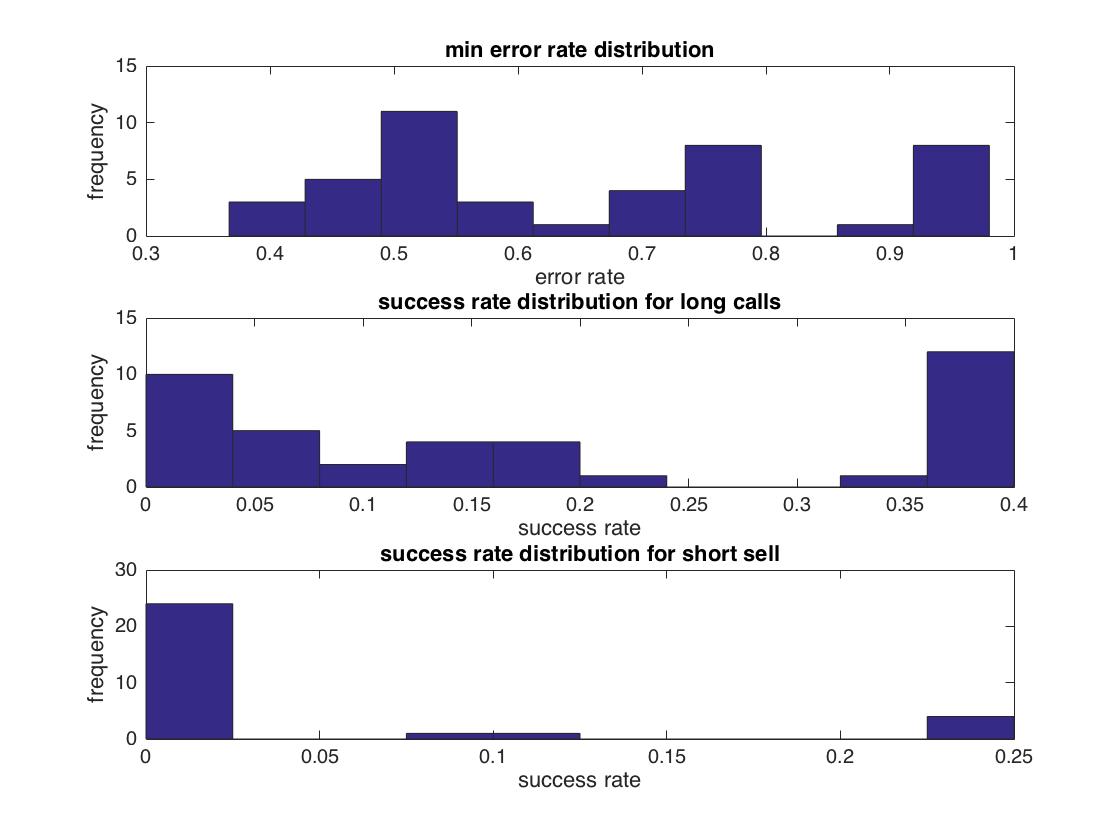
**Figure 2.** Average distribution of recommendations made by ZZAlpha as percentage of days across 755 trading days and 44 stocks. (a) Distribution of long recommendation. (b) Distribution of short recommendations.



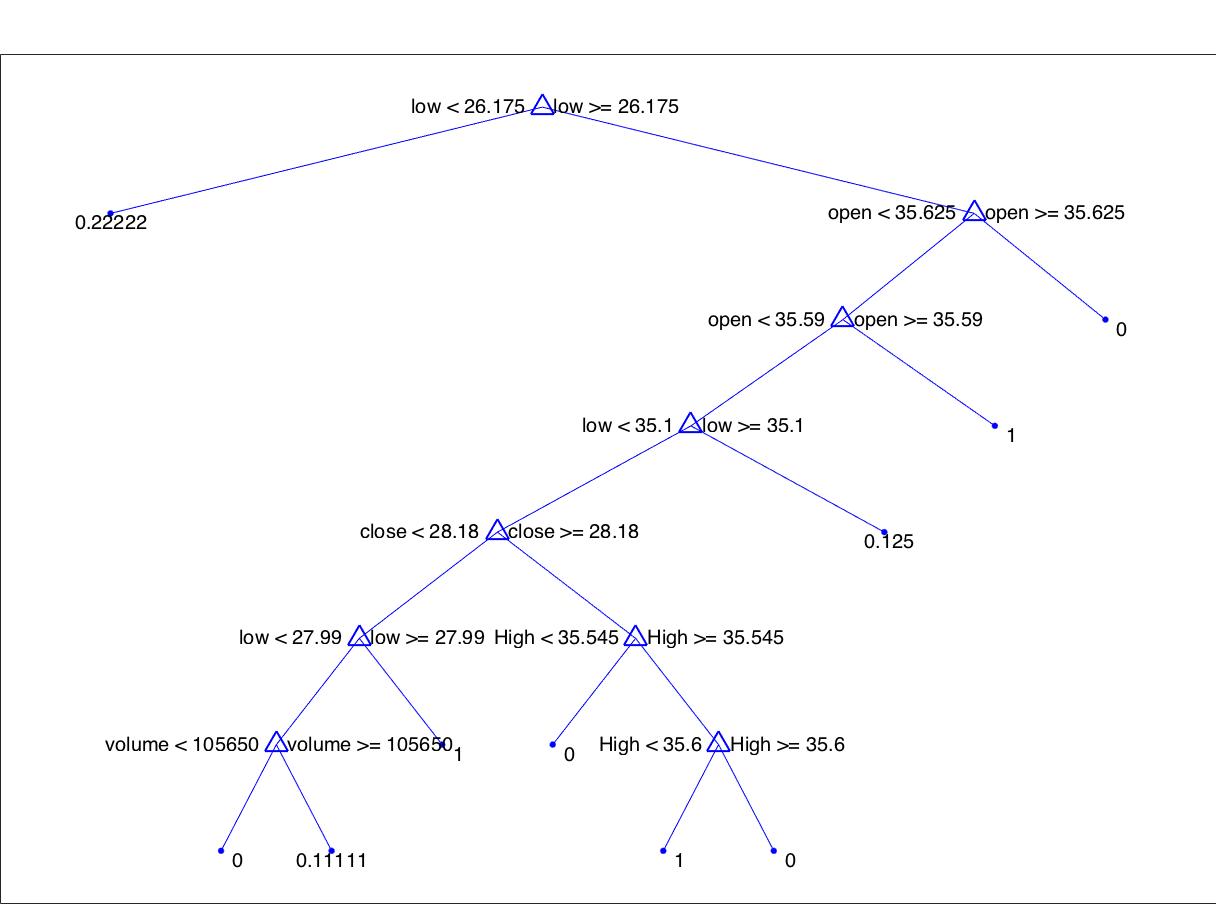
**Figure 3.** Logistic regression ROC curves for long call prediction over 44 stocks. (a) uses the logic model where f(mu) = log(mu/(1-mu)), (b) uses f(mu) = norminv(mu), (c) uses f(mu) = log(-log(1-mu)) and (d) uses f(mu) = log(-log(mu)).



**Figure 4.** Logistic regression ROC curves for short sell prediction over 44 stocks. (a) uses the logic model where f(mu) = log(mu/(1-mu)), (b) uses f(mu) = norminv(mu), (c) uses f(mu) = log(-log(1-mu)) and (d) uses f(mu) = log(-log(mu)).



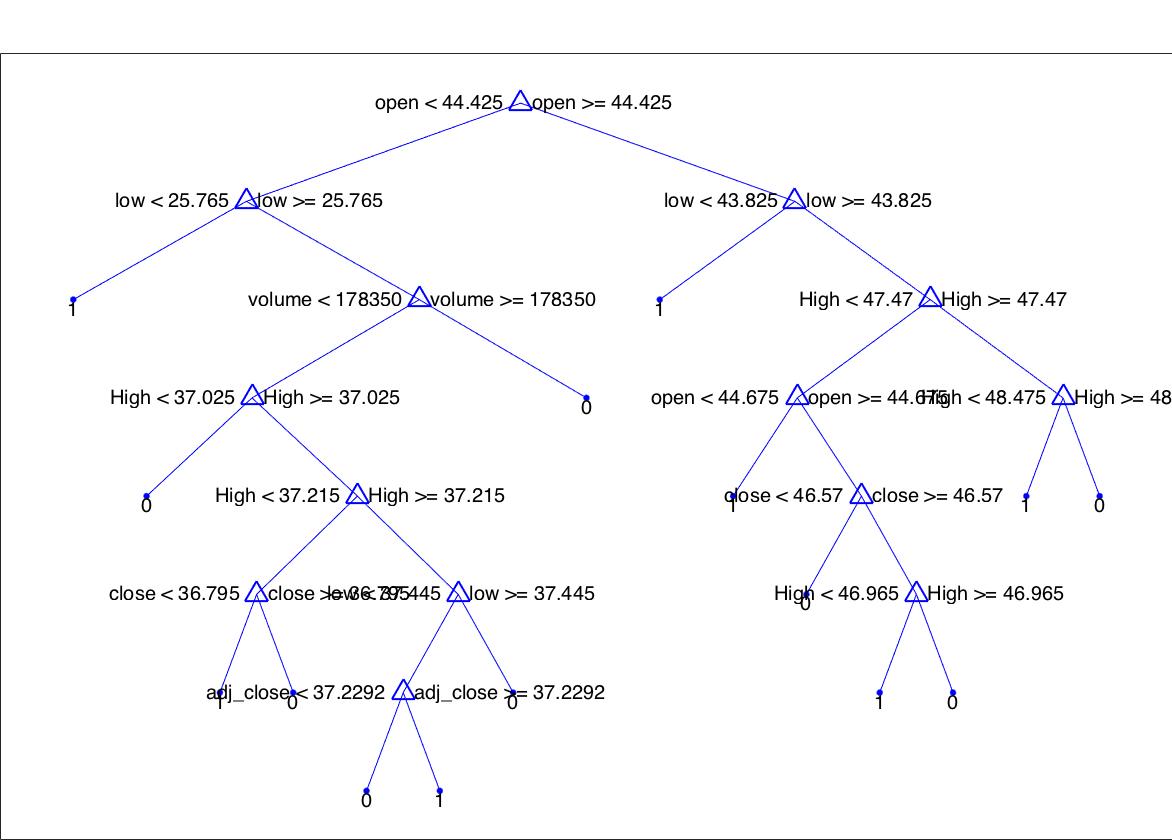
**Figure 5.** Overall error rate and success rate distribution over 44 stocks for logistic regression. (a) Overall error rate distribution. The error rates were chosen as the minimum after trying an array of cutoff values. (b) Distribution of success rates for long calls. (c) Distribution of success rates for short sells.



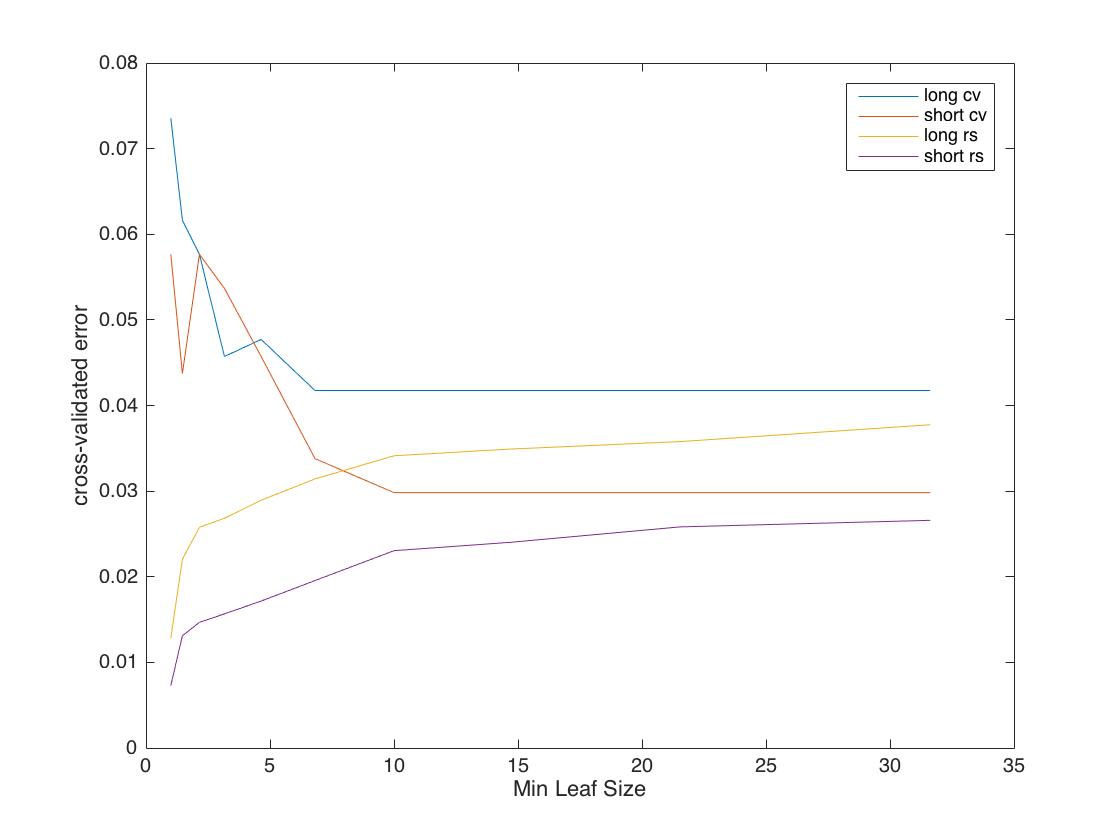
**Figure 6.** Full classification tree for long recommendations.



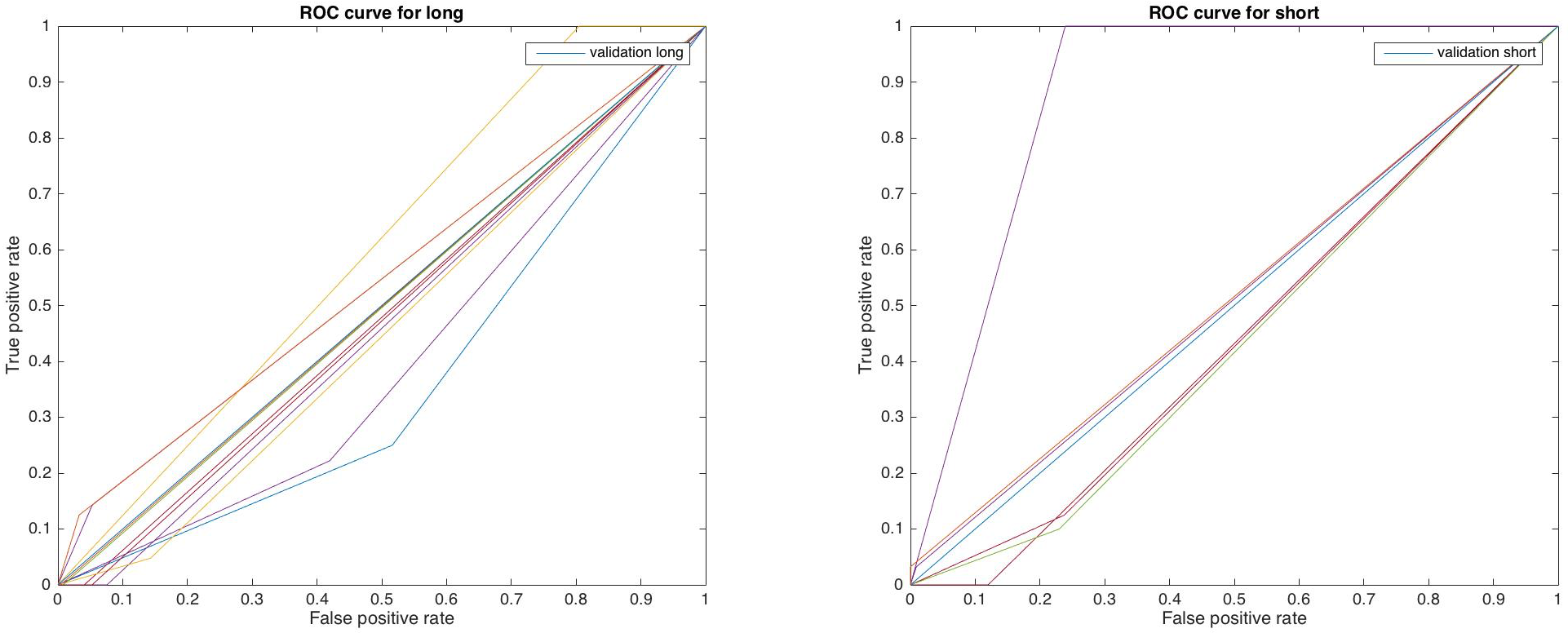
**Figure 7.** Full tree for short recommendations.



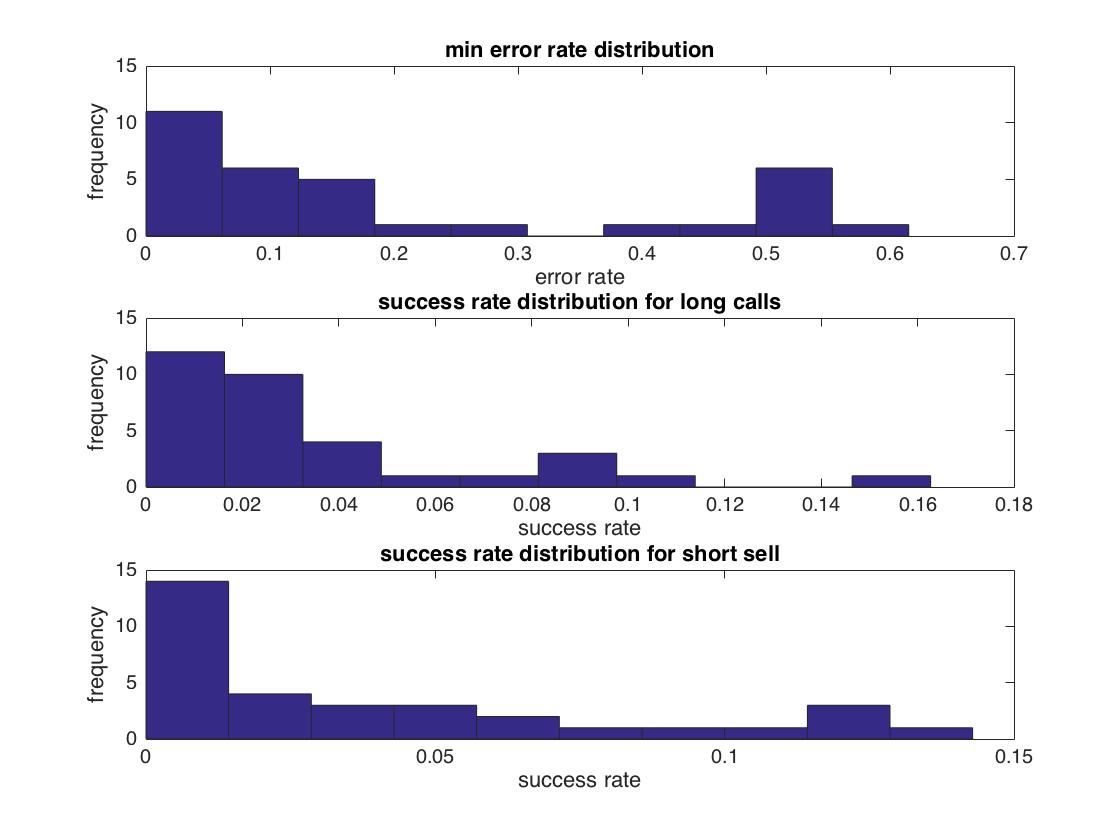
**Figure 8.** Final tree for short recommendations with minimum of 3 leaves in each node.



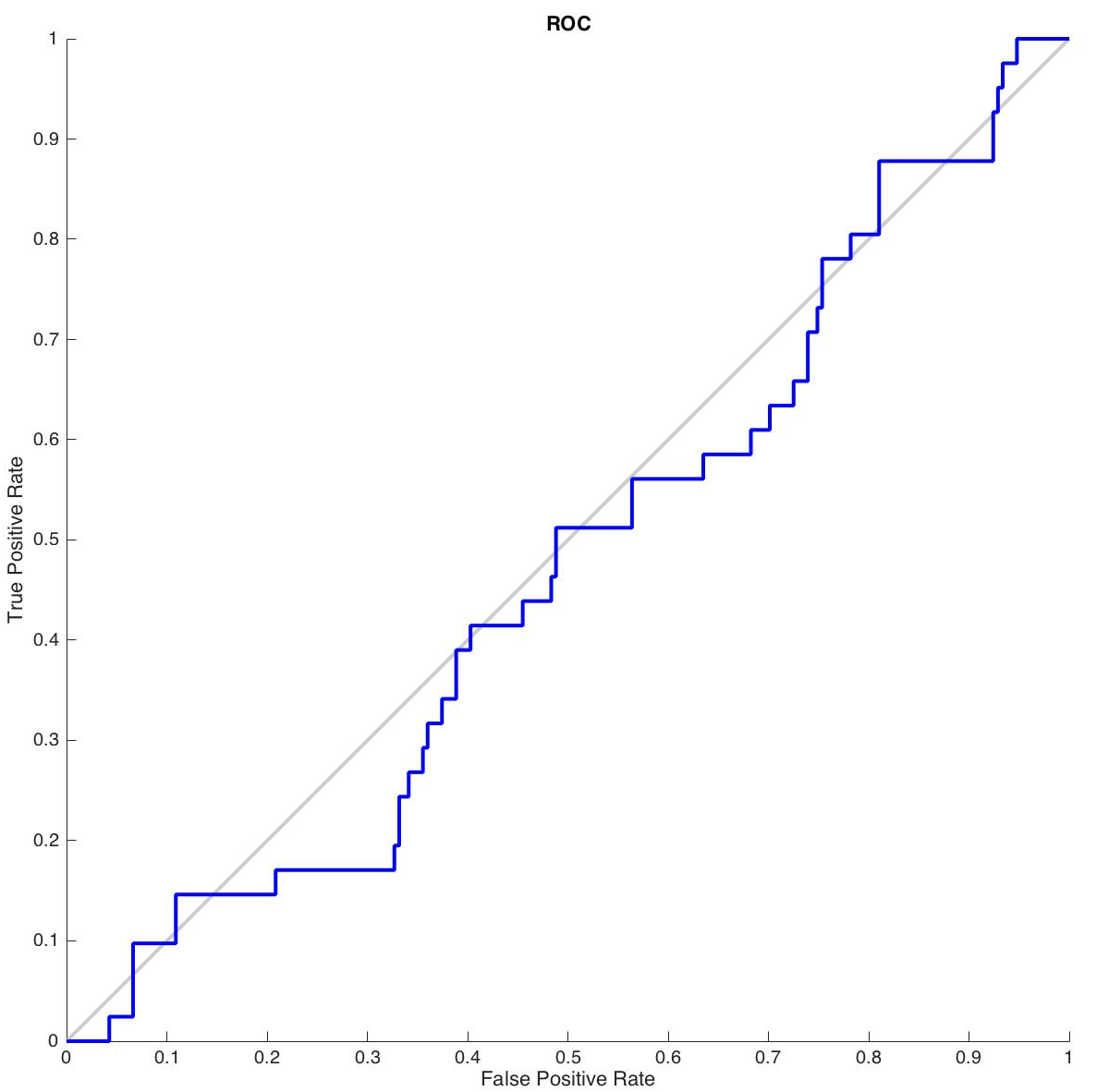
**Figure 9.** Cross validation error and resubstitution error for varying number of minimum leaves in each node.



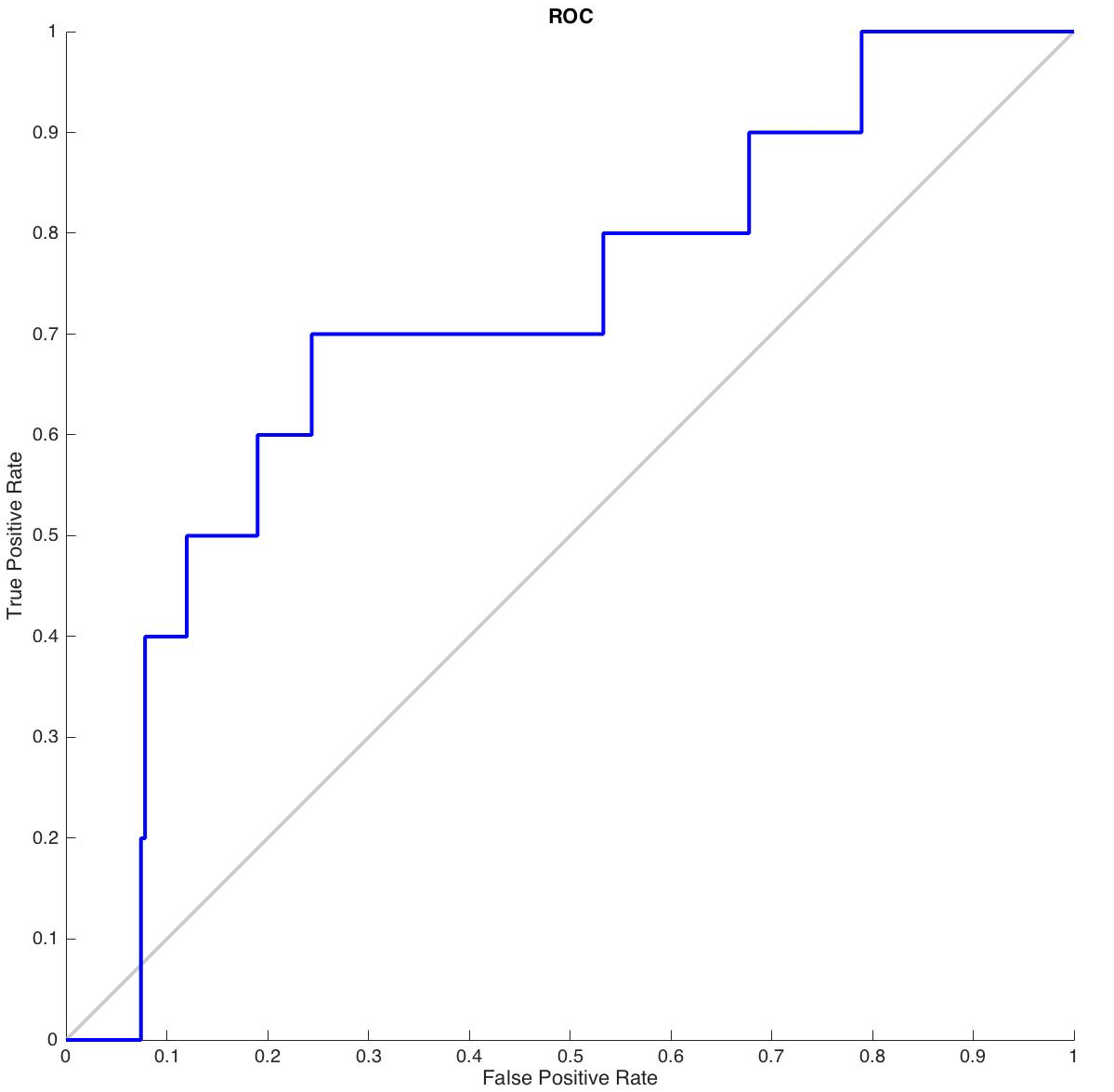
**Figure 10.** Decision Tree ROC curves for long and short recommendations across 44 stocks.



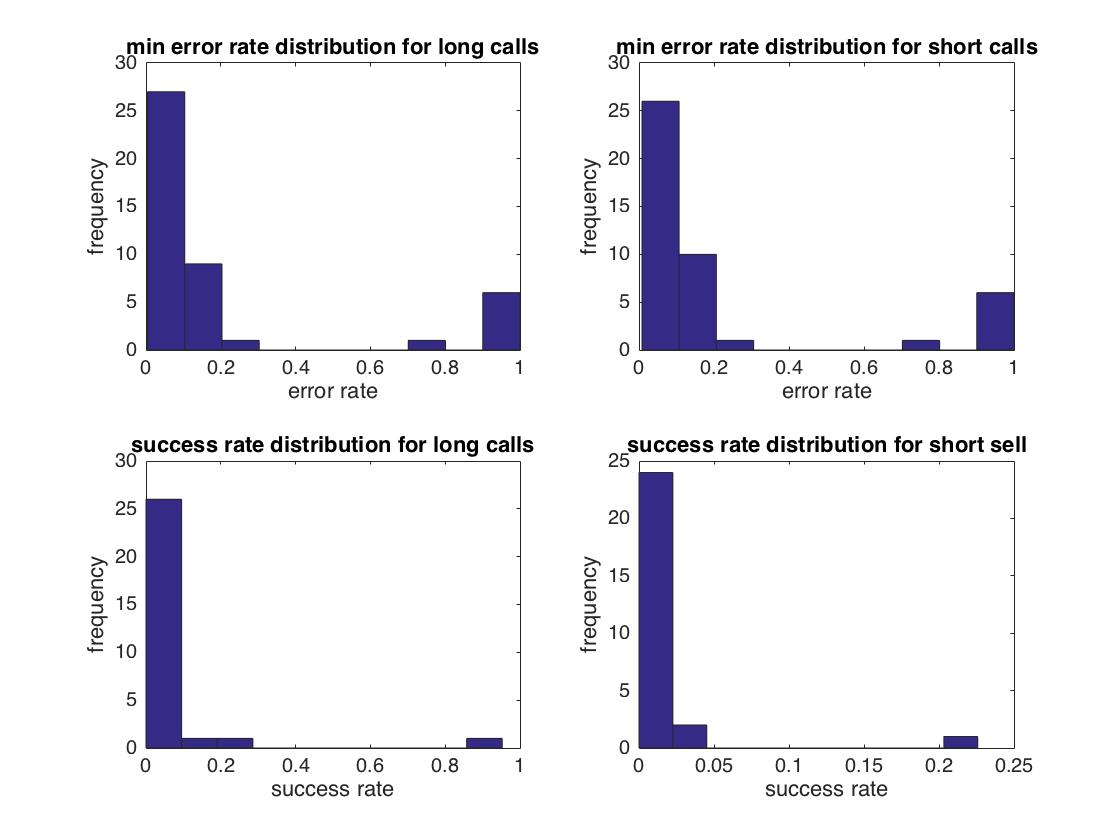
**Figure 11.** Overall error rate and success rate distribution over 44 stocks for classification tree. (a) Overall error rate distribution. The error rates were chosen as the minimum after trying an array of cutoff values. (b) Distribution of success rates for long calls. (c) Distribution of success rates for short sells.



**Figure 12.** Neural Net ROC curve for prediction of long recommendation for BIIB.



**Figure 13.** Neural Net ROC curve for prediction of short recommendation for BIIB.



**Figure 14.** Error rates and success rates distributions over 44 stocks for classification tree. (a) Error rate distribution for long and long recommendations. (b) Error rate distribution for short recommendations. The error rates were chosen as the minimum after trying an array of cutoff values. (c) Distribution of success rates for long calls. (d) Distribution of success rates for short sells.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Overall Error Rate | Success Rate (long) | Success Rate (short) |
| Logistic Regression | 50.3% | 10.6% | 0.8% |
| Classification Tree | 20.5% | 3.5% | 3.9% |
| Neural Net | 21.8% | 5.1% | 6.2% |

**Table 1.** Summary of three data mining models.

**Appendix IV. Matlab code (dataset available upon request)**

logistic\_regression.m

%

% tbl =array2table(cell\_array{1}(1:503,2:11),'VariableNames',{'open','low','high','close','volume','pctchange','adj\_close','action','long','short'});

% tbl(1:503,8)=array2table(cell\_array{1}(1:503,9));

%

model={'logit', 'probit', 'comploglog', 'loglog'};

stocks\_val=zeros(4,44);

stock\_vals=[];

for (b=1:4)

for (c=[1:44])

try

cutoff\_l=sum(cell\_array{c}(cell\_array{1}==1))/755;

cutoff\_s=sum(cell\_array{c}(cell\_array{1}==-1))/755;

clear mdl\_l mdl\_s

% trainingset=cell(size(cell\_array));

% validationset=cell(size(cell\_array));

% for i=1:44

% trainingset{1,i}=cell\_array{1,i}(1:503,:);

% validationset{1,i}=cell\_array{1,i}(504:755,:);

% end

pred=trainingset{1,c}(:,2:8);

resp\_train\_l=trainingset{1,c}(:,10);

resp\_train\_s=trainingset{1,c}(:,11);

%regression

mdl\_l = fitglm(pred,resp\_train\_l,'Distribution','binomial','Link',model{b});

mdl\_s = fitglm(pred,resp\_train\_s,'Distribution','binomial','Link',model{b});

[scores\_l\_val,ynewci\_l] = predict(mdl\_l,validationset{1,c}(:,2:8));

[scores\_s\_val,ynewci\_s] = predict(mdl\_s,validationset{1,c}(:,2:8));

scores\_l\_train = mdl\_l.Fitted.Probability;

scores\_s\_train = mdl\_s.Fitted.Probability;

rng('default');

[X\_train\_l,Y\_train\_l] = perfcurve(trainingset{1,1}(:,10),scores\_l\_train,1);

[X\_train\_s,Y\_train\_s] = perfcurve(trainingset{1,1}(:,11),scores\_s\_train,1);

[X\_val\_l,Y\_val\_l] = perfcurve(validationset{1,1}(:,10),scores\_l\_val,1);

[X\_val\_s,Y\_val\_s] = perfcurve(validationset{1,1}(:,11),scores\_s\_val,1);

figure(1);

subplot(2,2,b)

title (model{b});

% plot(X\_train\_l,Y\_train\_l);

% hold on;

% plot(X\_train\_s,Y\_train\_s);

% hold on;

plot(X\_val\_l,Y\_val\_l);

hold on;

% % plot(X\_val\_s,Y\_val\_s);

% hold on;

plot(X\_val\_s,X\_val\_s);

legend('validation long','base curve');

xlabel('False positive rate'); ylabel('True positive rate');

figure(2);

subplot(2,2,b)

title (model{b});

plot(X\_val\_s,Y\_val\_s);

hold on;

plot(X\_val\_s,X\_val\_s);

legend('validation short','base curve');

xlabel('False positive rate'); ylabel('True positive rate');

catch ME

end

multiple=0:0.1:15;

error=[];

correct\_l=[];

correct\_s=[];

for a=1:length(multiple)

%confusion matrices

[C\_train\_l,trash] = confusionmat(trainingset{1,1}(:,10),double(scores\_l\_train>cutoff\_l\*multiple(a)));

[C\_train\_s,trash] = confusionmat(trainingset{1,1}(:,11),double(scores\_s\_train>cutoff\_s\*multiple(a)));

[C\_val\_l,trash] = confusionmat(validationset{1,1}(:,10),double(scores\_l\_val>cutoff\_l\*multiple(a)));

[C\_val\_s,trash] = confusionmat(validationset{1,1}(:,11),double(scores\_s\_val>cutoff\_s\*multiple(a)));

error=[error (C\_val\_l(1,2)+C\_val\_l(2,1)+C\_val\_s(1,2)+C\_val\_s(2,1))/2/sum(sum(C\_val\_s))];

correct\_l=[correct\_l C\_val\_l(2,1)/sum(validationset{1,c}(:,10))];

correct\_s=[correct\_s C\_val\_s(2,1)/sum(validationset{1,c}(:,11))];

end

[min\_error\_rate,min\_idx] = min(error(:));

[max\_correct\_l, max\_idx\_l]=max(correct\_l(:));

[max\_correct\_s, max\_idx\_s]=max(correct\_s(:));

stocks\_val(1,c)=min\_error\_rate;

stocks\_val(2,c)=max\_correct\_l;

stocks\_val(3,c)=max\_correct\_s;

stocks\_val(4,c)=multiple (min\_idx);

end

stock\_vals=[stock\_vals stocks\_val];

figure (3)

subplot(3,1,1)

hist(stocks\_val(1,:))

title('min error rate distribution')

xlabel('error rate')

ylabel('frequency')

subplot(3,1,2)

hist(stocks\_val(2,:)/5)

title('success rate distribution for long calls')

xlabel('success rate')

ylabel('frequency')

subplot(3,1,3)

hist(stocks\_val(3,:))

title('success rate distribution for short sell')

xlabel('success rate')

ylabel('frequency')

end

% for j=1:length(predicted)

% if predicted(j,1)>cutoff\_long

% predicted(j,1)=1;

% elseif predicted(j,1)<cutoff\_short

% predicted(j,1)=-1;

% else predicted(j,1)=0;

% end

% end

% training=[cell\_array{1}(1:503,9) predicted(1:503,1)];

% validation=[cell\_array{1}(504:755,9) predicted(504:755,1)];

% [training\_confusion,order1]=confusionmat(training(:,1),training(:,2));

% [validation\_confusion, order2]=confusionmat(validation(:,1),training(:,2));

% A=[];

% for k=1:44

% tickers(2,k)=num2cell(nanmean(cell\_array{1,k}(:,6)));

% end

% [trash idx] = sort([tickers{2,:}], 'descend');

% tickers(:,idx)

% % random\_assign=nan(755,1);

%

% for k=1:44

% cell\_array{1,k}(:,10)=zeros(755,1);

% cell\_array{1,k}(:,11)=zeros(755,1);

% A=find(cell\_array{1,k}(:,9)==1);

% for j=1:length(A)

% cell\_array{1,k}(A(j),10)=1;

% end

% B=find(cell\_array{1,k}(:,9)<0);

% for l=1:length(B)

% cell\_array{1,k}(B(l),11)=1;

% end

% end

classification\_tree.m

cutoff\_l=sum(cell\_array{1}(cell\_array{1}==1))/755;

cutoff\_s=sum(cell\_array{1}(cell\_array{1}==-1))/755;

errorRates=[];

stock\_vals=[];

% t\_l=fitrtree(pred,resp\_train\_l,'PredictorNames',{'open' 'High' 'low' 'close' 'volume' 'pctchange' 'adj\_close'});

% % % rs\_error\_tl = resubLoss(t\_l); %resubstitution error, high bad low not necessarily good

% t\_s=fitrtree(pred,resp\_train\_s,'PredictorNames',{'open' 'High' 'low' 'close' 'volume' 'pctchange' 'adj\_close'});

% % % rs\_error\_ts = resubLoss(t\_s);

% view

% leafs = logspace(0,1.5,10);

% rng('default')

% N = numel(leafs);

% err\_l\_cv = zeros(N,1);

% err\_s\_cv = zeros(N,1);

% err\_l\_rs = zeros(N,1);

% err\_s\_rs = zeros(N,1);

% for n=1:N

% t\_l\_cv = fitctree(pred,resp\_train\_l,'CrossVal','On',...

% 'MinLeaf',leafs(n));

% t\_s\_cv = fitctree(pred,resp\_train\_s,'CrossVal','On',...

% 'MinLeaf',leafs(n));

% t\_l\_rs = fitrtree(pred,resp\_train\_l,...

% 'MinLeaf',leafs(n));

% t\_s\_rs = fitrtree(pred,resp\_train\_s,...

% 'MinLeaf',leafs(n));

% err\_l\_cv(n) = kfoldLoss(t\_l\_cv);

% err\_s\_cv(n) = kfoldLoss(t\_s\_cv);

% err\_l\_rs(n) = resubLoss(t\_l\_rs);

% err\_s\_rs(n) = resubLoss(t\_s\_rs);

% end

% plot(leafs,err\_l\_cv,leafs,err\_s\_cv,leafs,err\_l\_rs,leafs,err\_s\_rs);

% xlabel('Min Leaf Size');

% ylabel('cross-validated error');

% legend('long cv','short cv','long rs','short rs');

for (c=3:44)

try

clear OptimalTree\_l OptimalTree\_s

pred=trainingset{1,1}(:,2:8);

resp\_train\_l=trainingset{1,c}(:,10);

resp\_train\_s=trainingset{1,c}(:,11);

OptimalTree\_l = fitctree(pred,resp\_train\_l,'minleaf',1,'PredictorNames',{'open' 'High' 'low' 'close' 'volume' 'pctchange' 'adj\_close'});

OptimalTree\_s = fitctree(pred,resp\_train\_s,'minleaf',3,'PredictorNames',{'open' 'High' 'low' 'close' 'volume' 'pctchange' 'adj\_close'});

scores\_l\_train=predict(OptimalTree\_l,trainingset{1,c}(:,2:8));

scores\_s\_train=predict(OptimalTree\_s,trainingset{1,c}(:,2:8));

scores\_l\_val=predict(OptimalTree\_l,validationset{1,c}(:,2:8));

scores\_s\_val=predict(OptimalTree\_s,validationset{1,c}(:,2:8));

rng('default');

[X\_train\_l,Y\_train\_l] = perfcurve(trainingset{1,c}(:,10),scores\_l\_train,1);

[X\_train\_s,Y\_train\_s] = perfcurve(trainingset{1,c}(:,11),scores\_s\_train,1);

[X\_val\_l,Y\_val\_l] = perfcurve(validationset{1,c}(:,10),scores\_l\_val,1);

[X\_val\_s,Y\_val\_s] = perfcurve(validationset{1,c}(:,11),scores\_s\_val,1);

% figure(1);

% plot(X\_train\_l,Y\_train\_l);

% hold on;

% plot(X\_train\_s,Y\_train\_s);

% hold on;

% plot(X\_val\_l,Y\_val\_l);

% hold on;

% plot(X\_val\_s,Y\_val\_s);

% hold on;

% legend('training long','training short','validation long','validation short');

% xlabel('False positive rate'); ylabel('True positive rate');

%

%

% figure(2);

% subplot(1,2,1)

% title('ROC curve for long')

% plot(X\_val\_l,X\_val\_l);

% hold on;

% plot(X\_val\_l,Y\_val\_l);

% hold on;

% legend('validation long');

% xlabel('False positive rate'); ylabel('True positive rate');

%

% subplot(1,2,2)

% title('ROC curve for short')

% plot(X\_val\_s,X\_val\_s);

% hold on;

% plot(X\_val\_s,Y\_val\_s);

% hold on;

% legend('validation short');

% xlabel('False positive rate'); ylabel('True positive rate');

%

catch ME

end

% [~,~,~,bestlevel\_l] = cvLoss( OptimalTree\_l,...

% 'SubTrees','All','TreeSize','min')

% [~,~,~,bestlevel\_s] = cvLoss( OptimalTree\_s,...

% 'SubTrees','All','TreeSize','min')

% view(OptimalTree\_l,'mode','graph')

% view(OptimalTree\_s,'mode','graph')

multiple=0:0.1:15;

error=nan(1,44);

correct\_l=nan(1,44);

correct\_s=nan(1,44);

for a=1:length(multiple)

%confusion matrices

C\_train\_l = confusionmat(trainingset{1,c}(:,10),double(scores\_l\_train>cutoff\_l\*multiple(a)));

C\_train\_s = confusionmat(trainingset{1,c}(:,11),double(scores\_s\_train>cutoff\_s\*multiple(a)));

C\_val\_l = confusionmat(validationset{1,c}(:,10),double(scores\_l\_val>cutoff\_l\*multiple(a)));

C\_val\_s = confusionmat(validationset{1,c}(:,11),double(scores\_s\_val>cutoff\_s\*multiple(a)));

if sum(size(C\_val\_l))==4 &sum(size(C\_val\_s))==4

error=[error (C\_val\_l(1,2)+C\_val\_l(2,1)+C\_val\_s(1,2)+C\_val\_s(2,1))/2/sum(sum(C\_val\_s))];

correct\_l=[correct\_l C\_val\_l(2,1)/sum(sum(C\_val\_s))];

correct\_s=[correct\_s C\_val\_s(2,1)/sum(sum(C\_val\_s))];

end

end

[min\_error\_rate,min\_idx] = min(error(:));

[max\_correct\_l, max\_idx\_l]=max(correct\_l(:));

[max\_correct\_s, max\_idx\_s]=max(correct\_s(:));

stocks\_val(1,c)=min\_error\_rate;

stocks\_val(2,c)=max\_correct\_l;

stocks\_val(3,c)=max\_correct\_s;

stocks\_val(4,c)=multiple (min\_idx);

%

end

%

figure (4)

subplot(3,1,1)

hist(stocks\_val(1,:))

title('min error rate distribution')

xlabel('error rate')

ylabel('frequency')

subplot(3,1,2)

hist(stocks\_val(2,:))

title('success rate distribution for long calls')

xlabel('success rate')

ylabel('frequency')

subplot(3,1,3)

hist(stocks\_val(3,:))

title('success rate distribution for short sell')

xlabel('success rate')

ylabel('frequency')

% resubOpt = resubLoss(OptimalTree\_l);

% lossOpt = kfoldLoss(crossval(OptimalTree\_l));

% resubDefault = resubLoss(t\_l);

% lossDefault = kfoldLoss(crossval(t\_l));

% resubOpt,resubDefault,lossOpt,lossDefault

%

%

% view(t\_l,'mode','graph')

% Ynew = predict(tree,Xnew);

**neural\_net.m**

% c=13;

error\_l=[];

error\_s=[];

success\_l=[];

success\_s=[];

stocks\_val=[];

for c=1:44

pred=trainingset{1,c}(:,2:8);

resp\_train\_l=trainingset{1,c}(:,10);

resp\_train\_s=trainingset{1,c}(:,11);

x=pred.';%predictors

t=resp\_train\_l.';%target

setdemorandstream(391418381);

clear net testX testT;

net = patternnet(1);

% view(net);

[net,tr] = train(net,x,t);

% nntraintool

% plotperform(tr);

testX = validationset{1,c}(:,2:8).';

testT\_l = validationset{1,c}(:,10).';

testT\_s = validationset{1,c}(:,11).';

testY\_l = net(testX);

testY\_s = net(testX);

testIndices\_l = vec2ind(testY\_l);

testIndices\_s = vec2ind(testY\_s);

% plotconfusion(testT,testY)

[l,cm\_l] = confusion(testT\_l,testY\_l);

[s,cm\_s] = confusion(testT\_s,testY\_s)

error\_l=[error\_l l];

success\_l=[success\_l cm\_l(2,2)/sum(validationset{1,c}(:,10))];

error\_s=[error\_s s];

success\_s=[success\_s cm\_s(2,2)/sum(validationset{1,c}(:,11))];

% fprintf('Percentage Correct Classification : %f%%\n', 100\*(1-c));

% fprintf('Percentage Incorrect Classification : %f%%\n', 100\*c);

figure (1)

plotroc(testT\_l,testY\_l)

% plotroc(testT\_s,testY\_s)

hold on

[min\_error\_rate\_l,min\_idx\_l] = min(error\_l(:));

[min\_error\_rate\_s,min\_idx\_s] = min(error\_s(:));

[max\_correct\_l, max\_idx\_l]=max(success\_l);

[max\_correct\_s, max\_idx\_s]=max(success\_s);

% stocks\_val(1,c)=min\_error\_rate\_l;

% stocks\_val(2,c)=min\_error\_rate\_s;

% stocks\_val(3,c)=max\_correct\_l;

% stocks\_val(4,c)=max\_correct\_s;

end

figure (4)

subplot(2,2,1)

hist(error\_l)

title('min error rate distribution for long calls')

xlabel('error rate')

ylabel('frequency')

subplot(2,2,2)

hist(error\_s)

title('min error rate distribution for short calls')

xlabel('error rate')

ylabel('frequency')

subplot(2,2,3)

hist(success\_l)

title('success rate distribution for long calls')

xlabel('success rate')

ylabel('frequency')

subplot(2,2,4)

hist(success\_s)

title('success rate distribution for short sell')

xlabel('success rate')

ylabel('frequency')