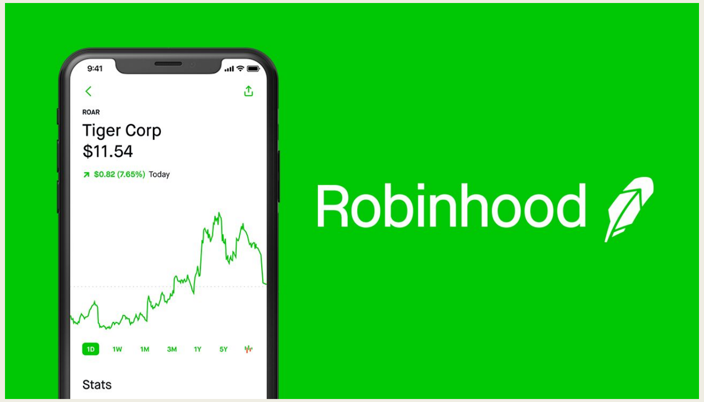
SUPERVISED LEARNING:

Next Day Rise/Fall Stock Price Predictor

Kaci Kus
September 25, 2020

Motivation

- Stock price prediction is a topic of interest for companies and investors
  - *Buy low, sell high*
- Difficult to predict and highly volatile due to changes in:
  - *Politics*
  - *Leadership*
  - *New releases*
  - *Investor sentiment*



Short-term → Medium-term → Long-term

Days to weeks | Months to a year | 5 year

Short-term prediction

- Next day stock price prediction can:
  - *Help day/swing traders decide when to buy/sell*
- Recently tech company stocks skyrocketed in August
  - *Tesla grew 81% in 20 days!*
- Two ways to handle prediction model
  - *Classification (rise/fall)*
  - *Regression (predict actual $ price)*

Tesla Stock Price

Goal

- Use supervised learning to predict whether any companies closing stock price will INCREASE or DECREASE the next day based on that day's news headlines and historical stock price data.

- Stretch Goal:
  - *Test basic regression model*

OUTLINE

- Data set
- Base model
- Feature engineering
- Models on augmented data
- Results
- Future work

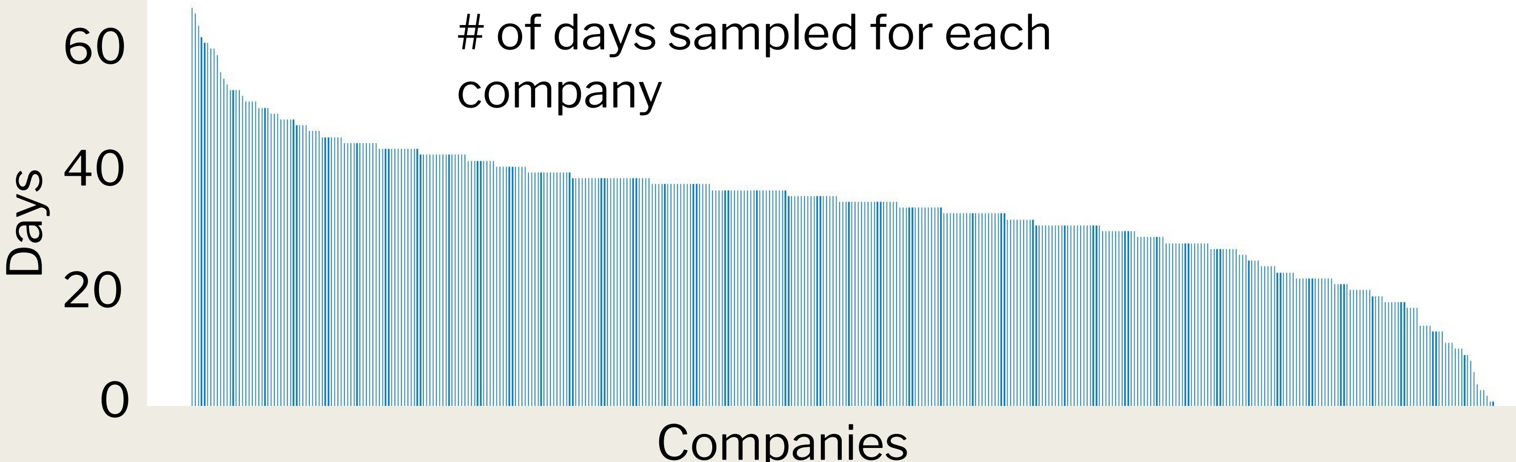
Data set: Collection

API requests to get most recent 100 news headlines for companies from S&P 500 list

- Sentiment analysis using NLTK library
- Scores of 0 were removed, and remaining scores were averaged for each day

Gather historical stock price data from Yahoo Finance

- Use the yfinance Python library
- Also collected next-day stock price to use as target variable



Feature data set

- 14,134 Observations
- 18 columns
- No missing data

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14134 entries, 0 to 14133
Data columns (total 18 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   company      14134 non-null   object
 1   date         14134 non-null   object
 2   30d          14134 non-null   float64
 3   7d           14134 non-null   float64
 4   prc_30d      14134 non-null   float64
 5   prc_7d       14134 non-null   float64
 6   std_30d      14134 non-null   float64
 7   std_7d       14134 non-null   float64
 8   open_price   14134 non-null   float64
 9   high         14134 non-null   float64
 10  low          14134 non-null   float64
 11  close        14134 non-null   float64
 12  volume       14134 non-null   float64
 13  neg          14134 non-null   float64
 14  neu          14134 non-null   float64
 15  pos          14134 non-null   float64
 16  compound     14134 non-null   float64
 17  prc_volume   14134 non-null   float64
dtypes: float64(16), object(2)
memory usage: 2.0+ MB
```

Target data

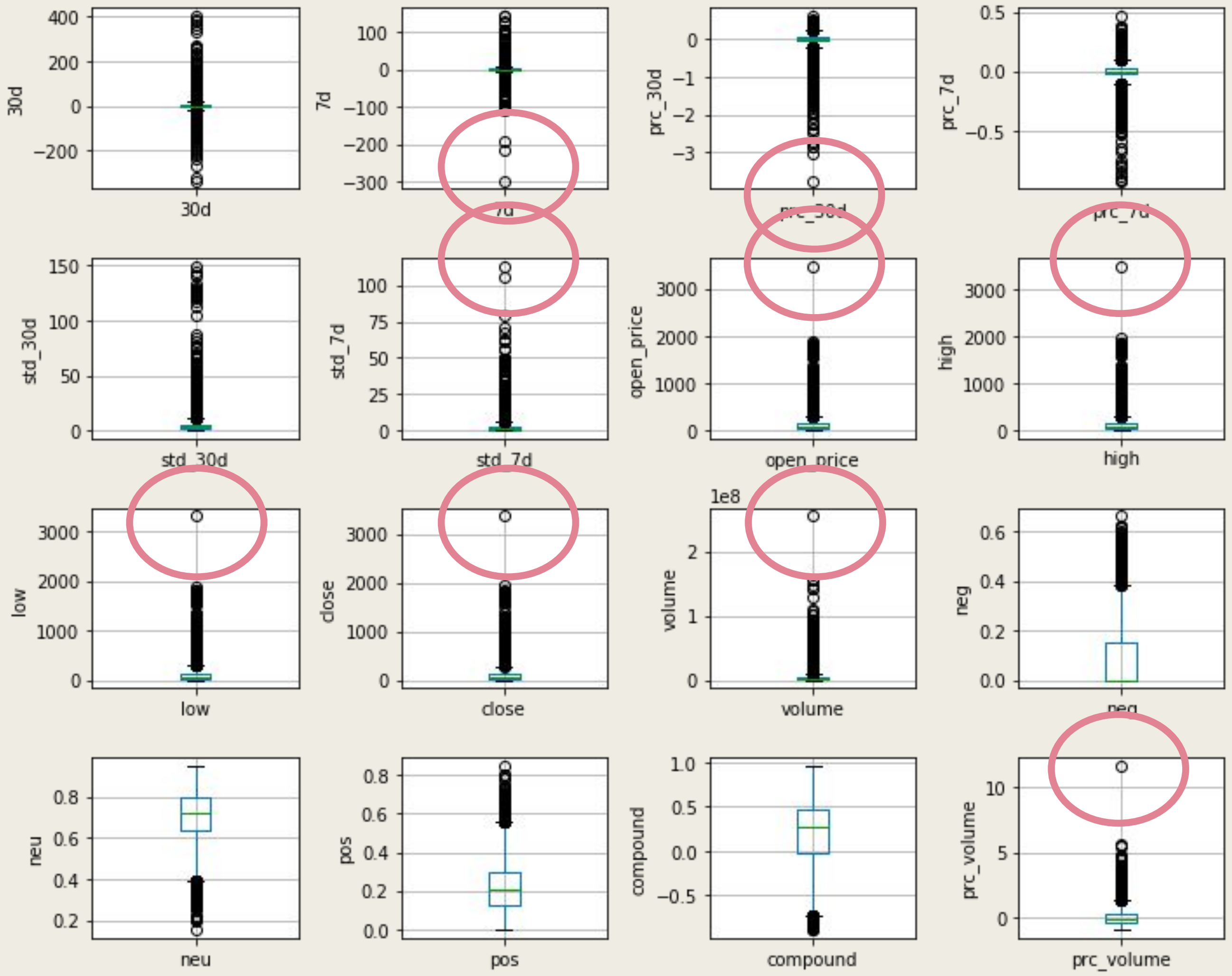
- 14,134 Observations
- 2 columns
- Add column for binary target

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14134 entries, 0 to 14133
Data columns (total 2 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   1d       14134 non-null   float64
 1   1d_prc   14134 non-null   float64
dtypes: float64(2)
memory usage: 331.3 KB
```

Outlier removal

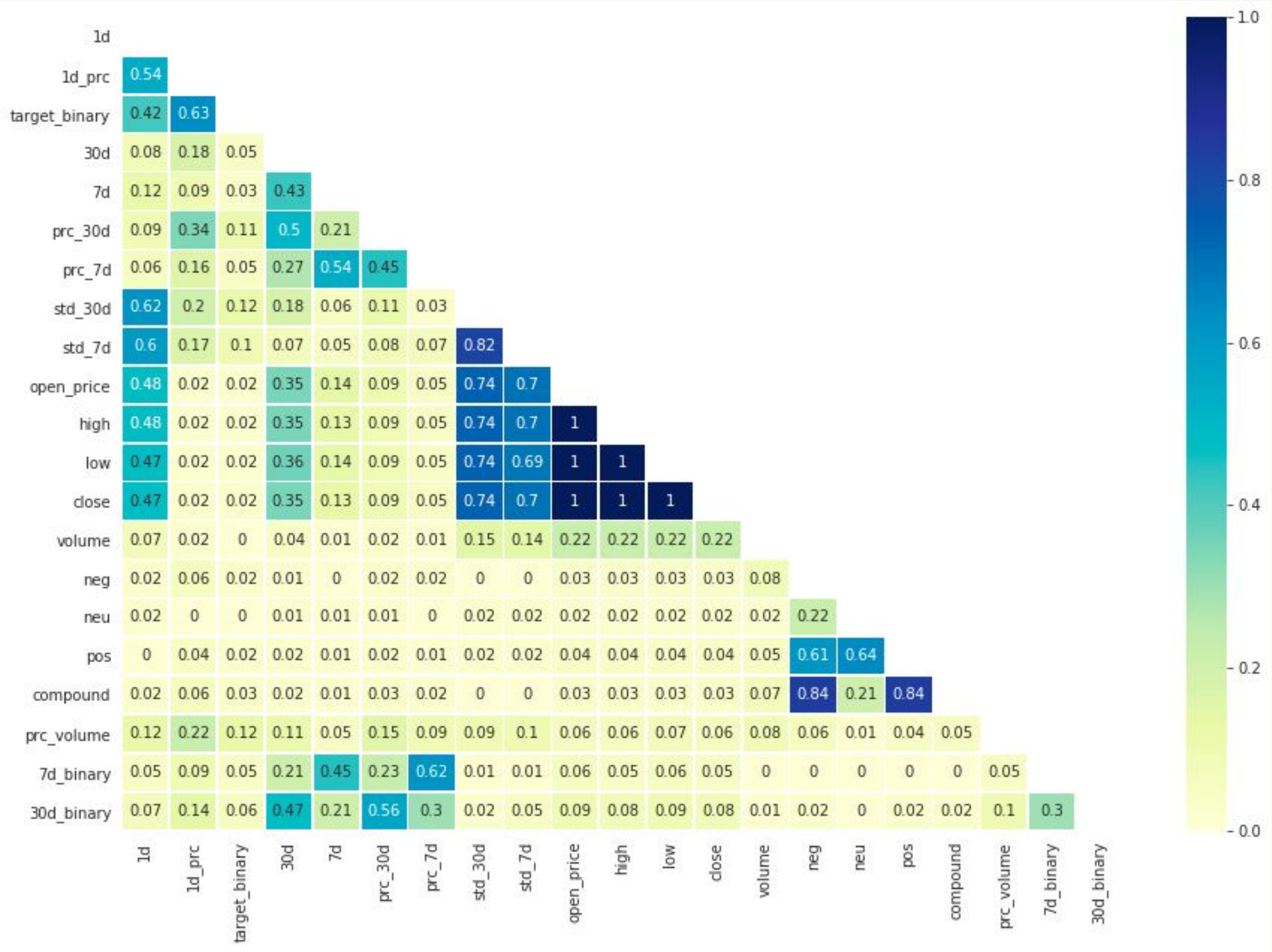
- 6 outliers stand out

Boxplots of continuous variable distribution



Initial correlation matrix

- There collinearity
- Std_30d is strongly correlated with std_7d, close, low, high, and open price
  - Drop std_7d because it has weaker correlation with target
- Open, high, low, and close are very strongly correlated
  - Try to combine into new variables
- Neg and pos are strongly correlated with compound sentiment score
  - Drop them from feature set



TARGET IS BINARY:
0 : NO CHANGE OR DECREASE
1: POSITIVE NEXT DAY CHANGE

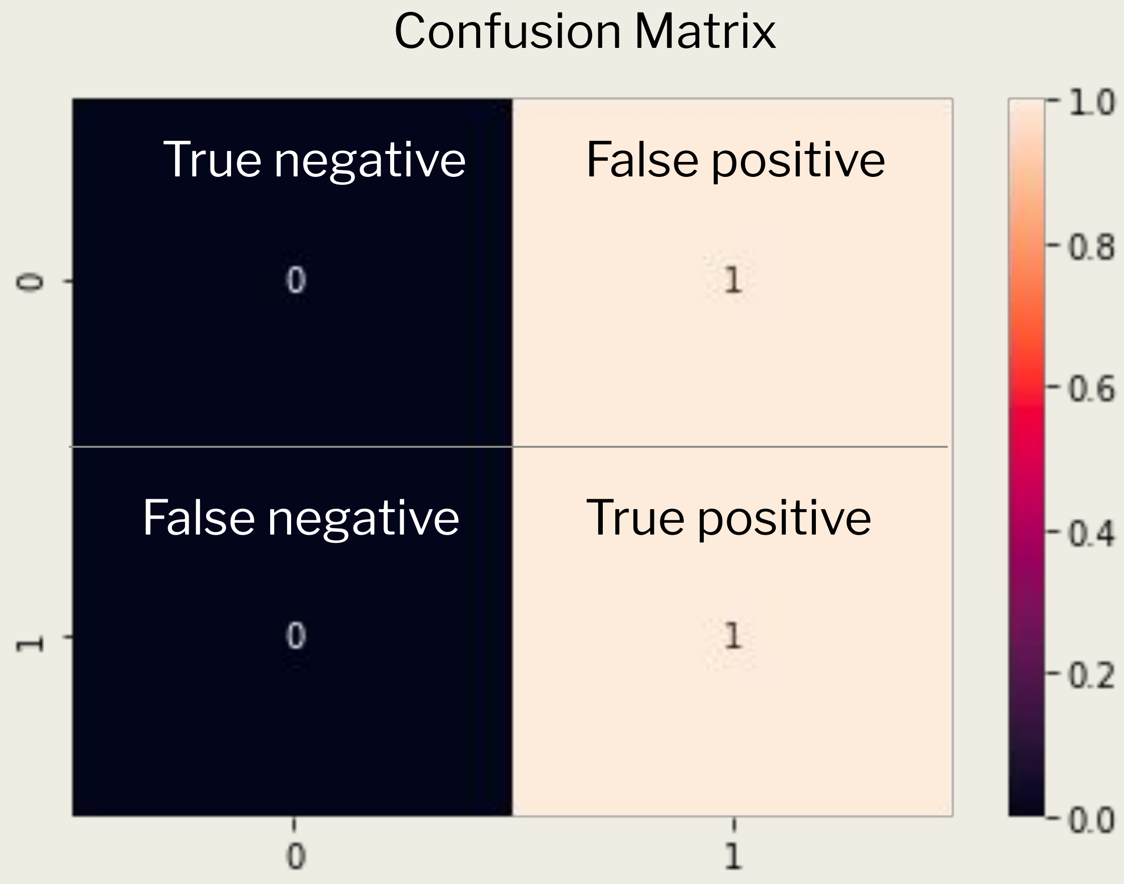
- 8944 observations == 1
- 5190 observations == 0

Baseline Logistic Regression Model

- 0.7/0.3 train-test-split
- Accuracy: 0.62
- Assigning everything to be a next day increase (1)

- n next day increase observations/ total observations = 2,628/4,239
- = 0.62

Not a very good model!

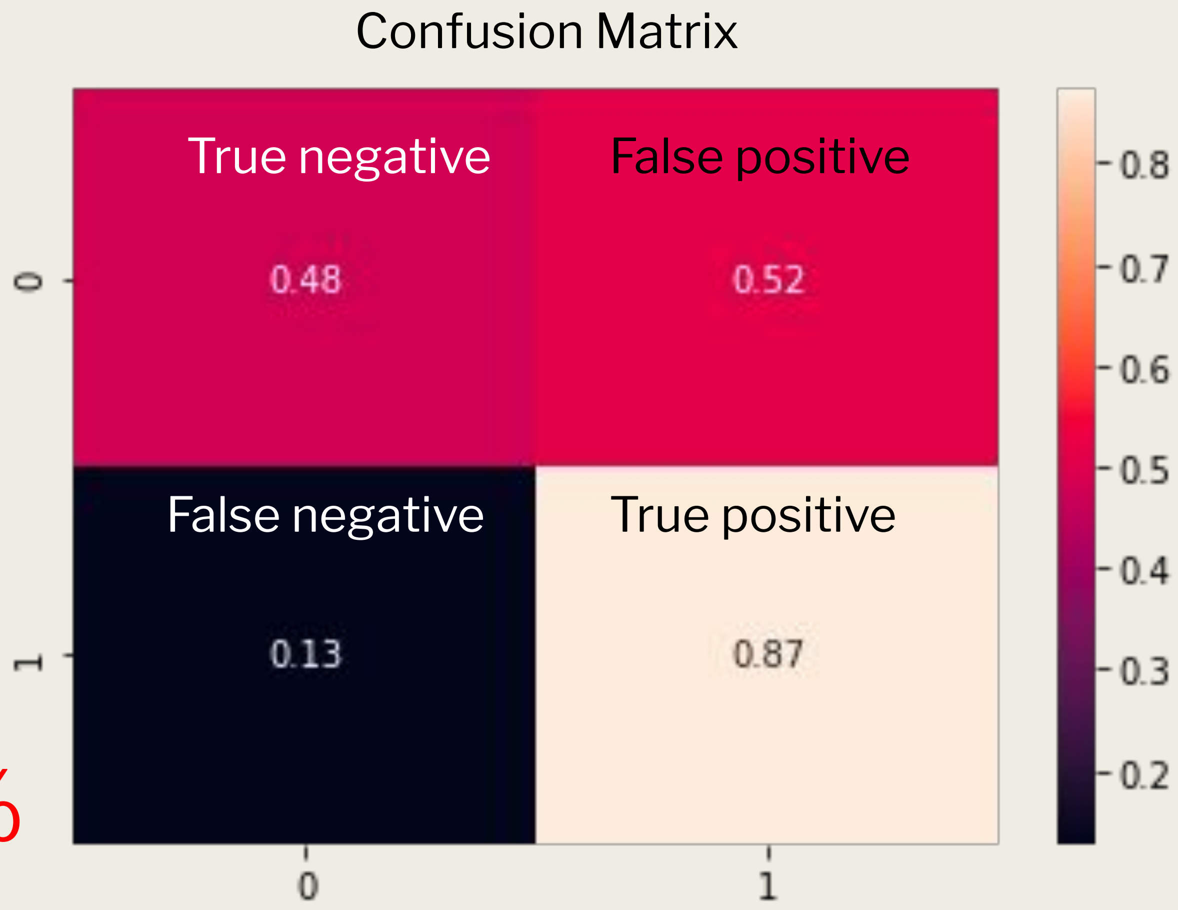


Baseline Gradient Boosting Model

- GridSearchCV() used to find optimal parameters
- Learning rate: 0.1
- Max depth: 9
- N_estimators: 500
- Subsample: 0.7

- Accuracy: 0.72
- Precession: 0.73
- Recall: 0.87

Slightly better accuracy, but it is still scoring >50% false positives, which is too risky!

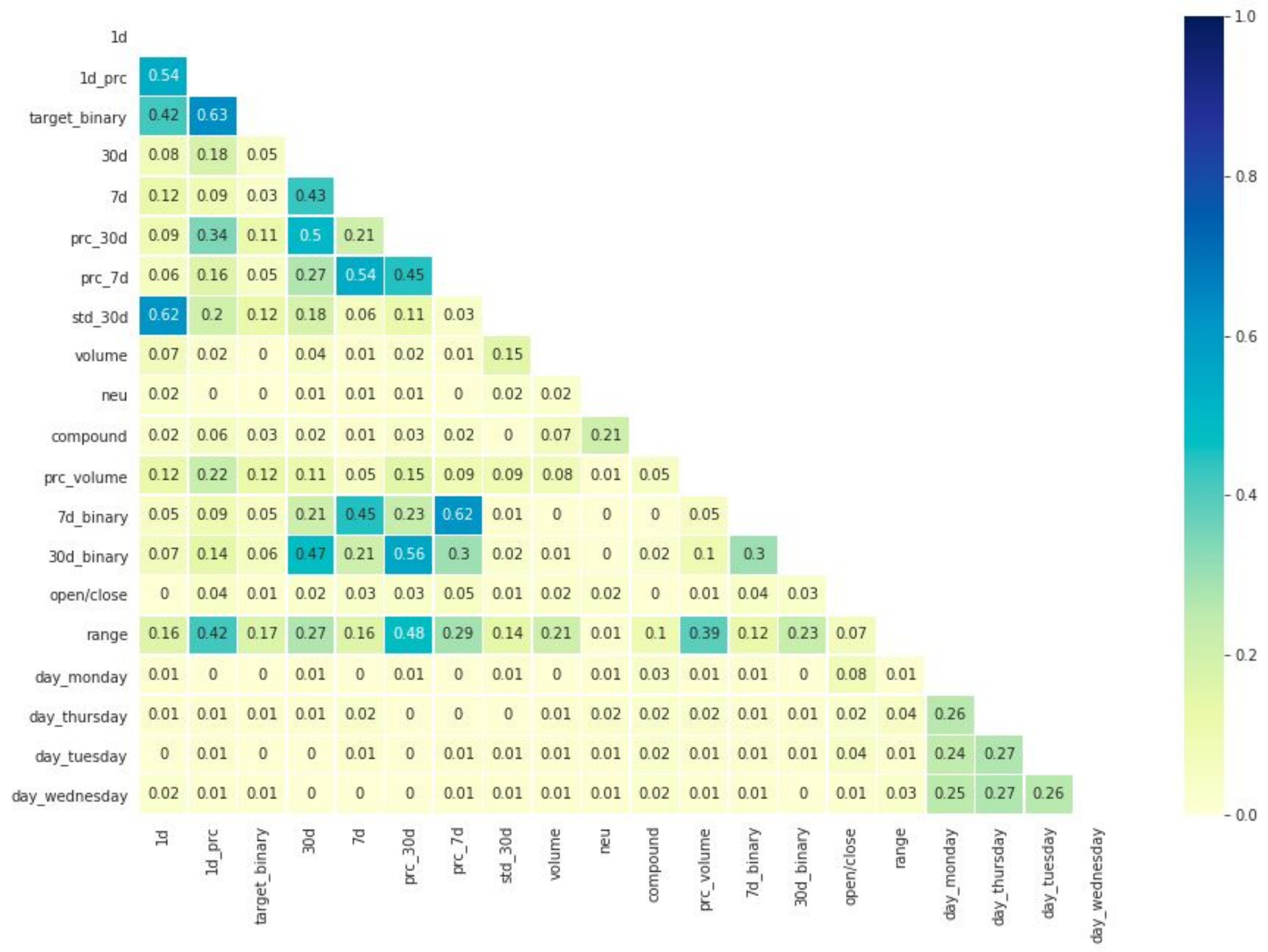


IMPROVING THE MODEL

- MinMaxScaler() used to scale the data from 0 to 1 to account for binary features (such as day of the week).
  - *Scaler was fit to only the training feature set, and this scaler was then applied to the test feature set.*
  - *Target values were not scaled.*
- GridSearchCV() used to identify optimal parameters for each model
- Reduce collinearity of variables

Feature engineering

- Convert date to day of week categorical variable
- Create open : close price ratio variable
- Calculate daily range : close price ratio.
- Drop neg, pos, std_7d, and date

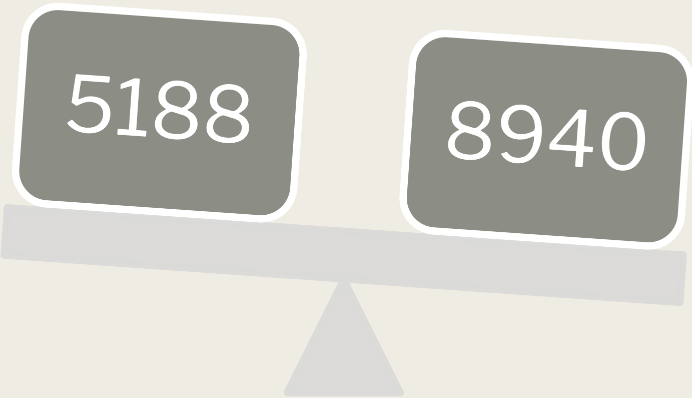


No features correlated > 0.7

Balancing target data

Decrease/
no change:
0

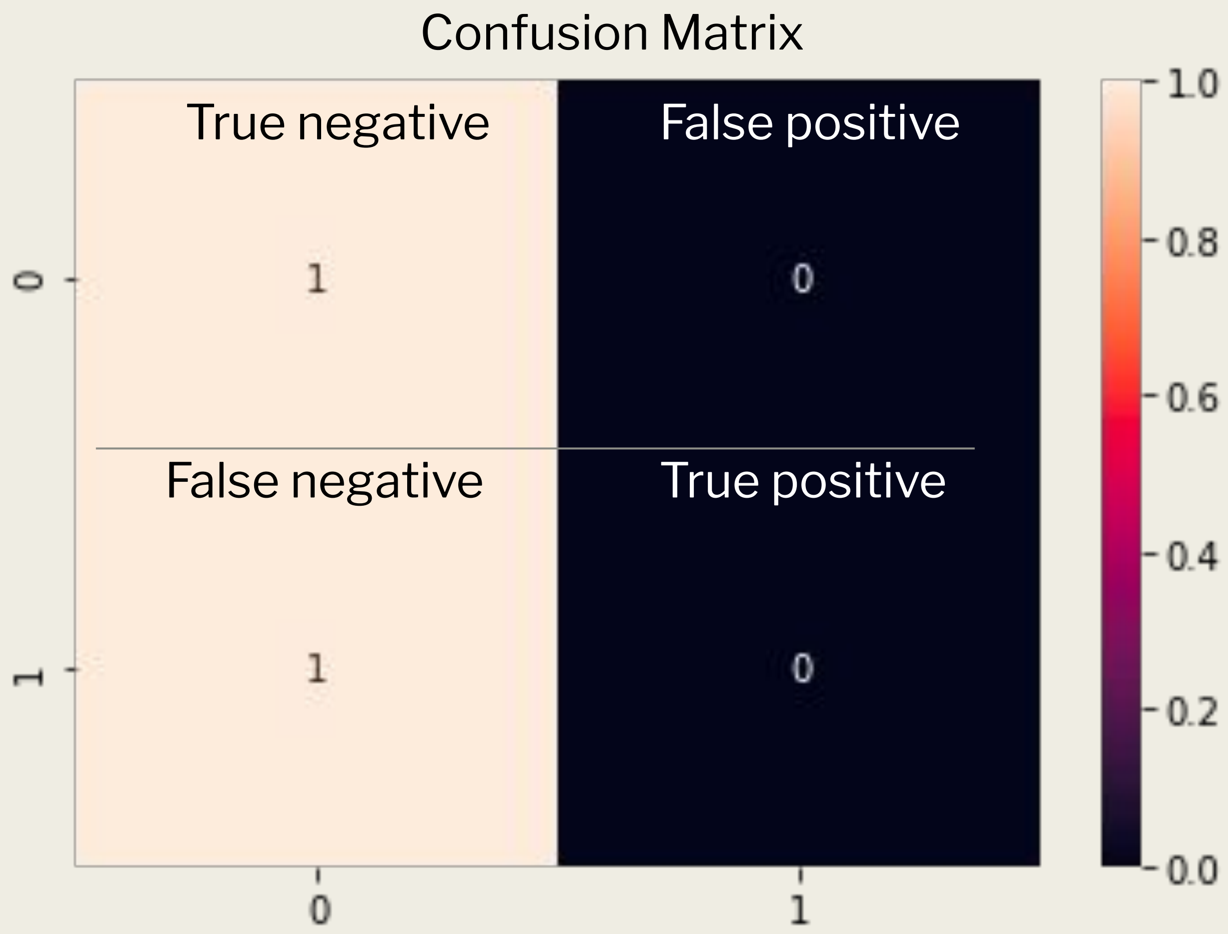
Increase: 1



- Impact of unbalanced target variable:
  - *Contributing to over-predicting false-positives*
- Solution:
  - *Re-sample the observations with target == 0*
  - *Random sampling with replacement to create 8940 observations with target == 0*

Logistic Regression Model after feature engineering

- Still a very poor estimator
- Predicting everything to be a next day decrease
- Accuracy: 0.5

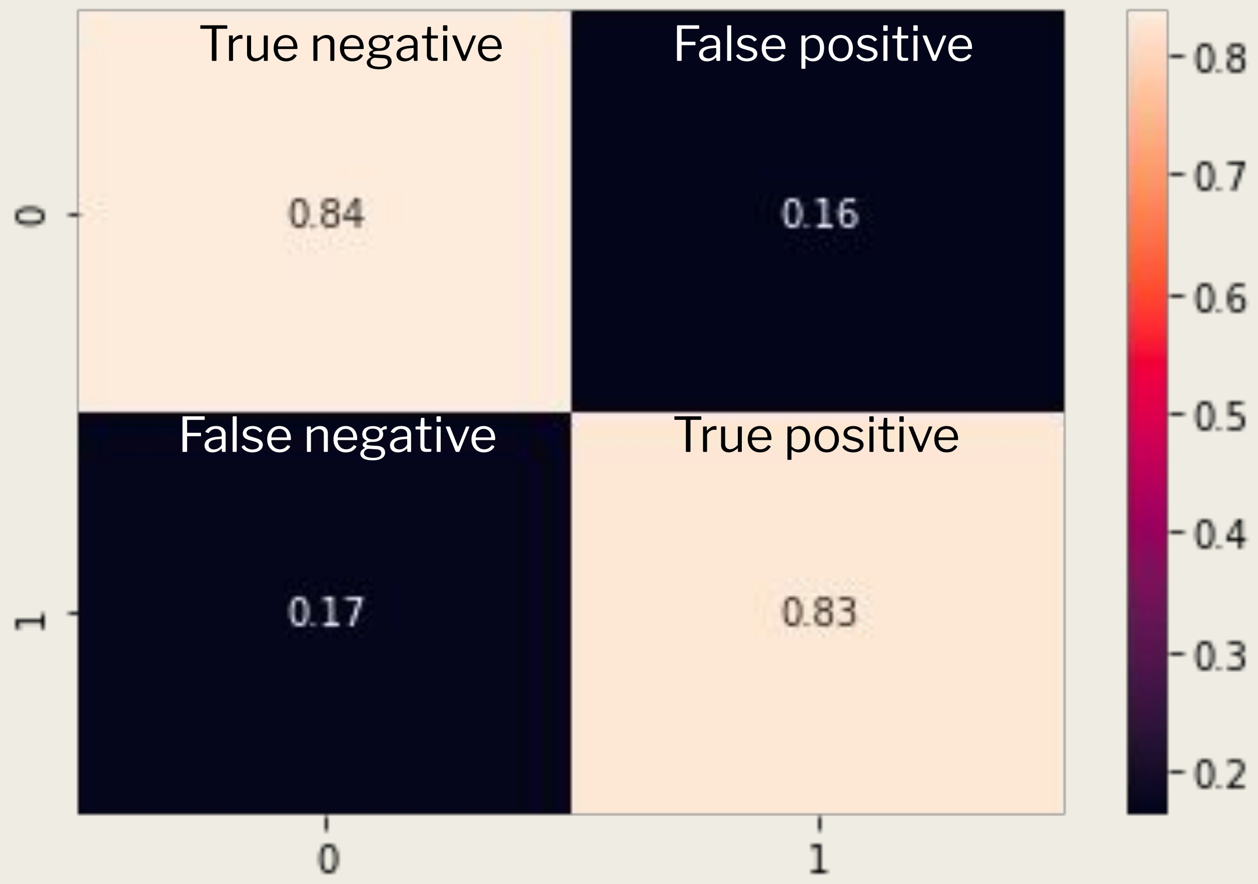


Gradient Boosting Model after feature engineering

- Accuracy: 0.83
- Precision: 0.83
- Recall: 0.82

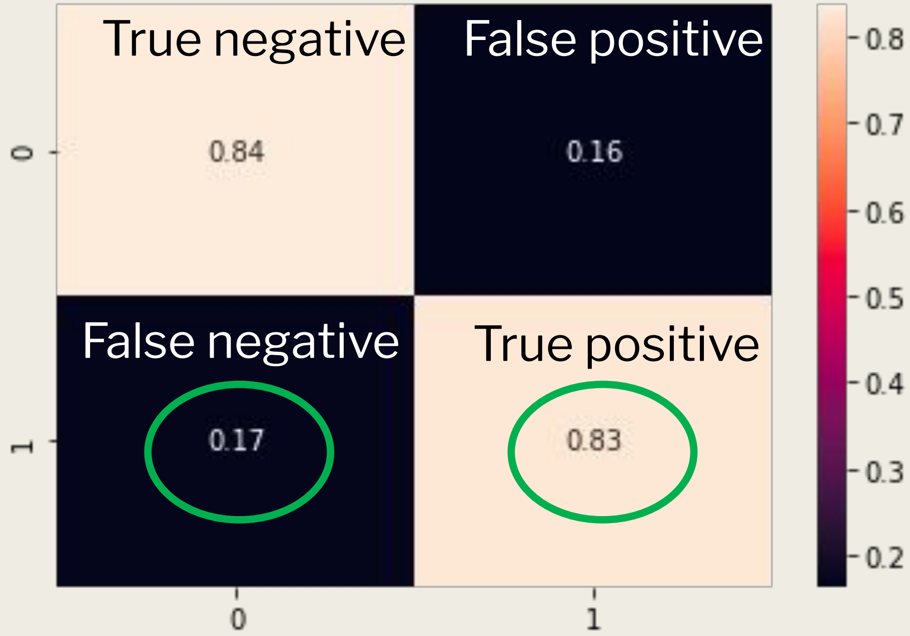
Significant improvement in performance!

Confusion Matrix

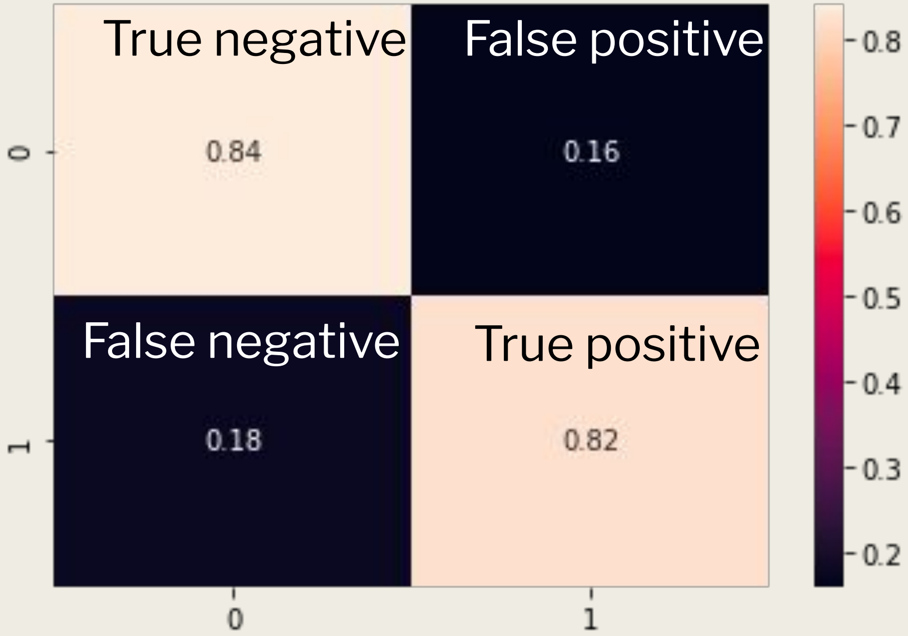


Is Sentiment Analysis of Headlines Significantly Impacting Results?

With sentiment analysis



Without Sentiment analysis



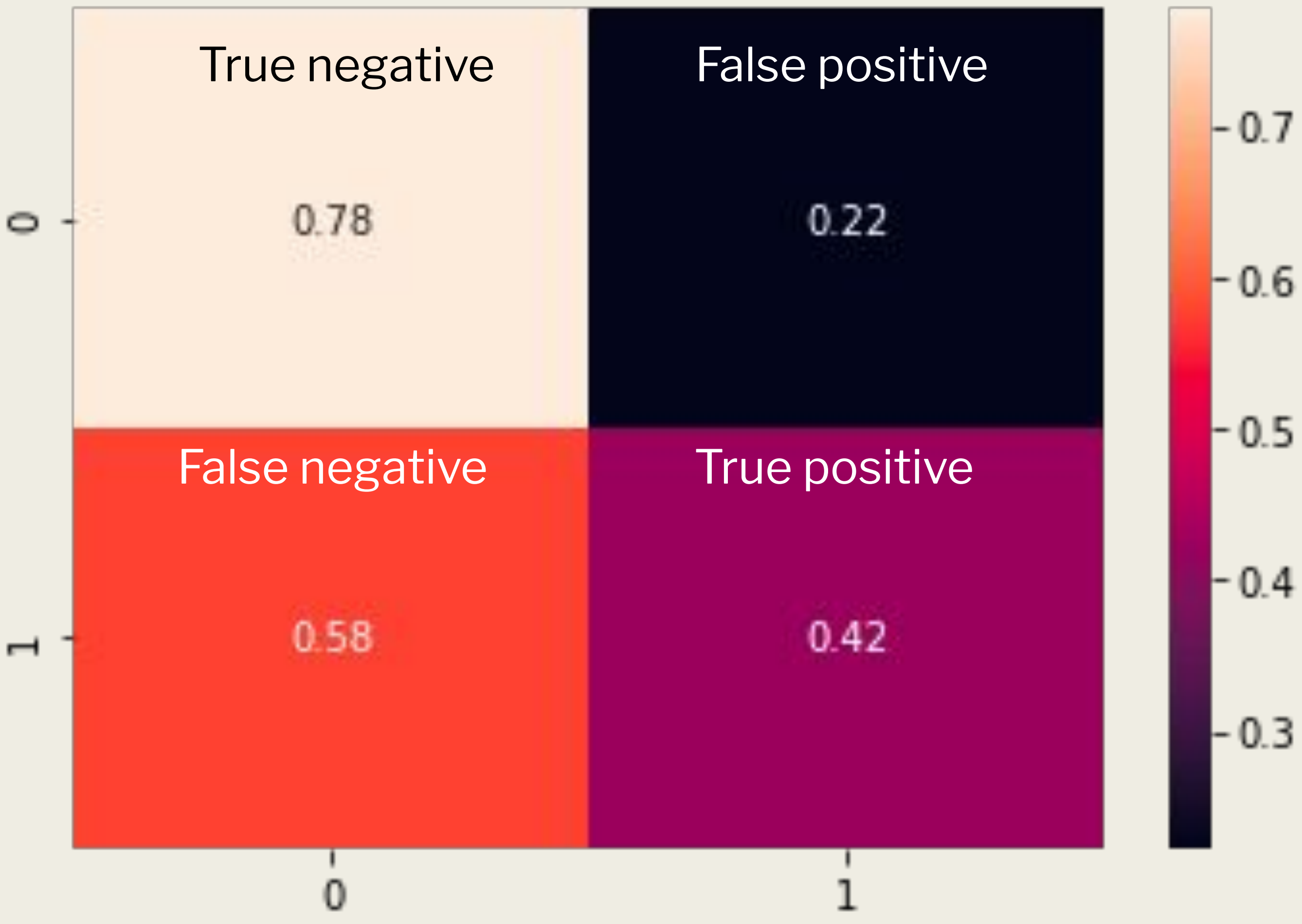
- Sentiment analysis is not contributing heavily to the model performance
- Sentiment analysis does improve false negative and true positive scores
- Worth keeping and attempting to improve in the future

Support Vector Machines Model

- GridSearchCV()
  - *C = 10*
  - *Gamma = 1*
  - *Kernel = 'rbf'*
- Accuracy: 0.6
- Precision: 0.66
- Recall: 0.42

All around weaker performance than gradient boosting model

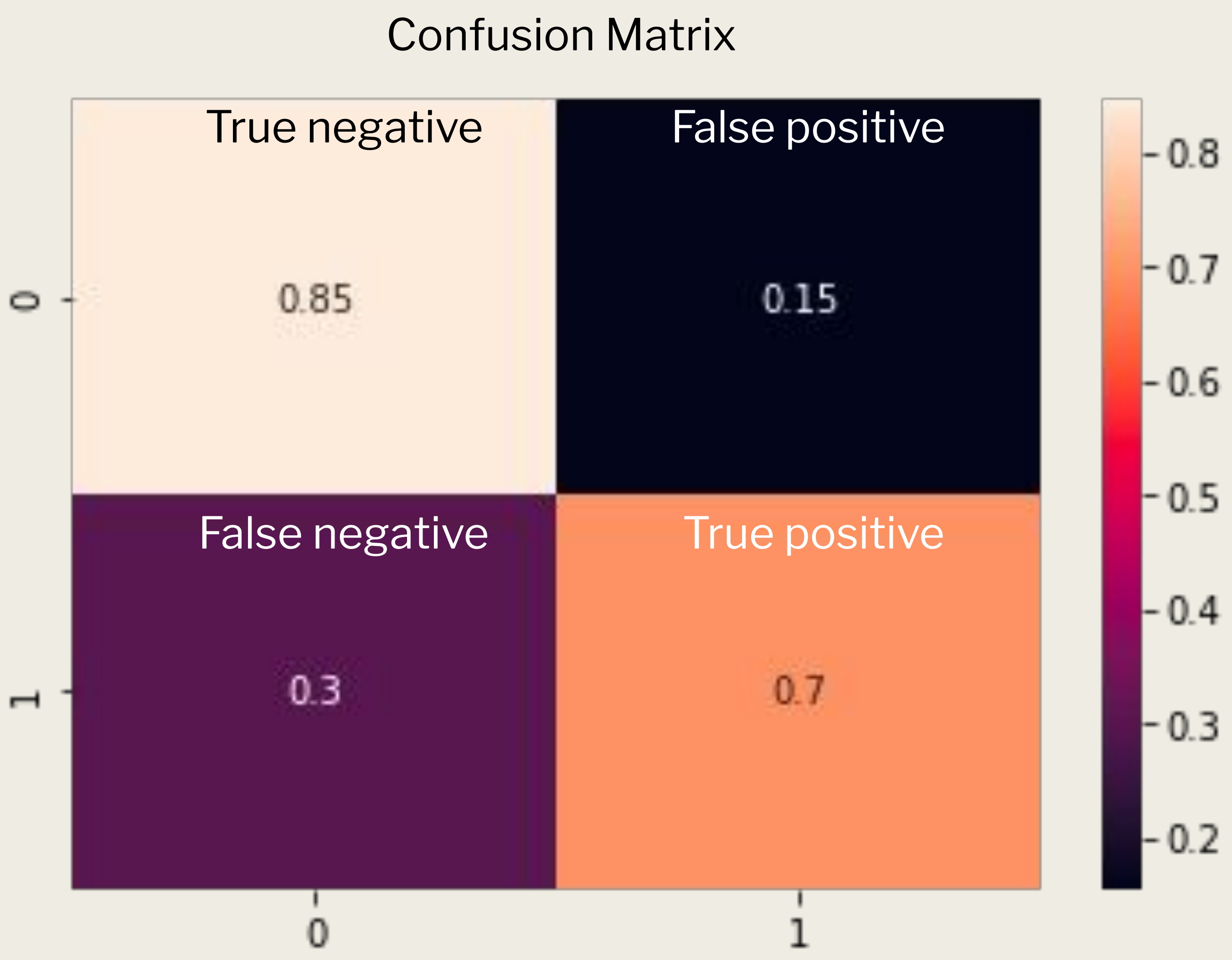
Confusion Matrix



Random Forest Classifier

- GridSearchCV()
  - *Criterion: 'gini'*
  - *Max depth: 15*
  - *Min samples split: 10*
  - *n_estimators: 300*
- Accuracy: 0.77
- Precision: 0.82
- Recall: 0.70

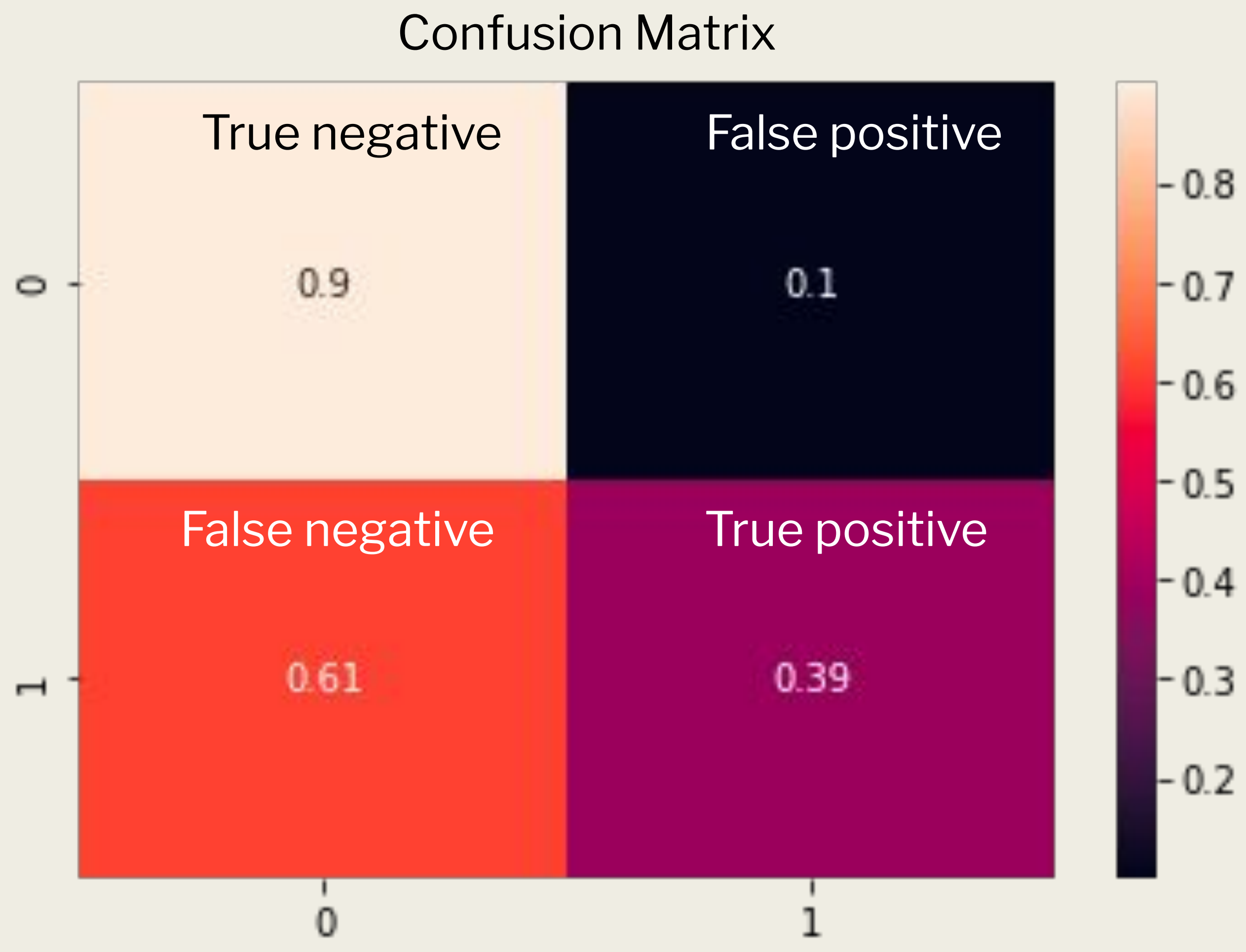
Slightly better and slightly worse than gradient boosting model



K-Nearest Neighbors model

- K = sqrt(n)
  - *N = 10,730*
  - *K = 105*
- Accuracy: 0.64
- Precision: 0.79
- Recall: 0.39

Really good at identifying next-day decreases, not so good at identifying increases

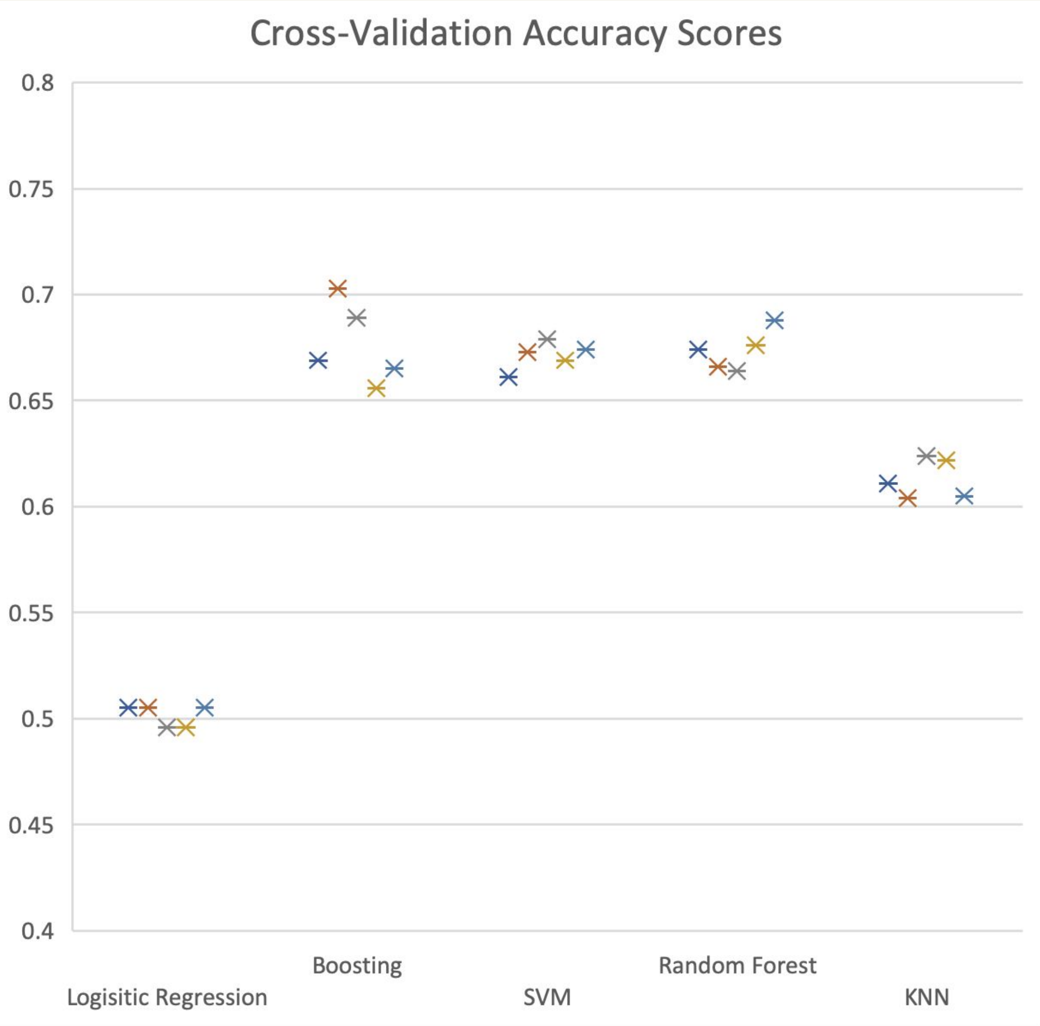


Results

	Logistic Regression	Gradient Boosting	Support Vector Machines	Random Forest	K-NN
Accuracy	0.5	0.83	0.6	0.77	0.64
Precision	0	0.83	0.66	0.82	0.79
Recall	0	0.82	0.42	0.7	0.39
True positive	0	0.83	0.42	0.7	0.39
True negative	1	0.84	0.78	0.85	0.9
False positive	0	0.16	0.22	0.15	0.1
False negative	1	0.17	0.58	0.3	0.61

Cross-validation

- All models seem to be relatively consistent.
- Gradient Boosting Model has the greatest variation, but still only has a range of 0.05, which is not too bad.



Results

- KNN Model:
  - *Really good for identifying next day decreases*
  - *Conservative, risk-averse model*
- Gradient Boosting Classifier:
  - *All around best performance*

Bonus Model: Regression

- OLS regression model:
  - *Adj. R-squared: 0.2*
  - *Very poor performance*

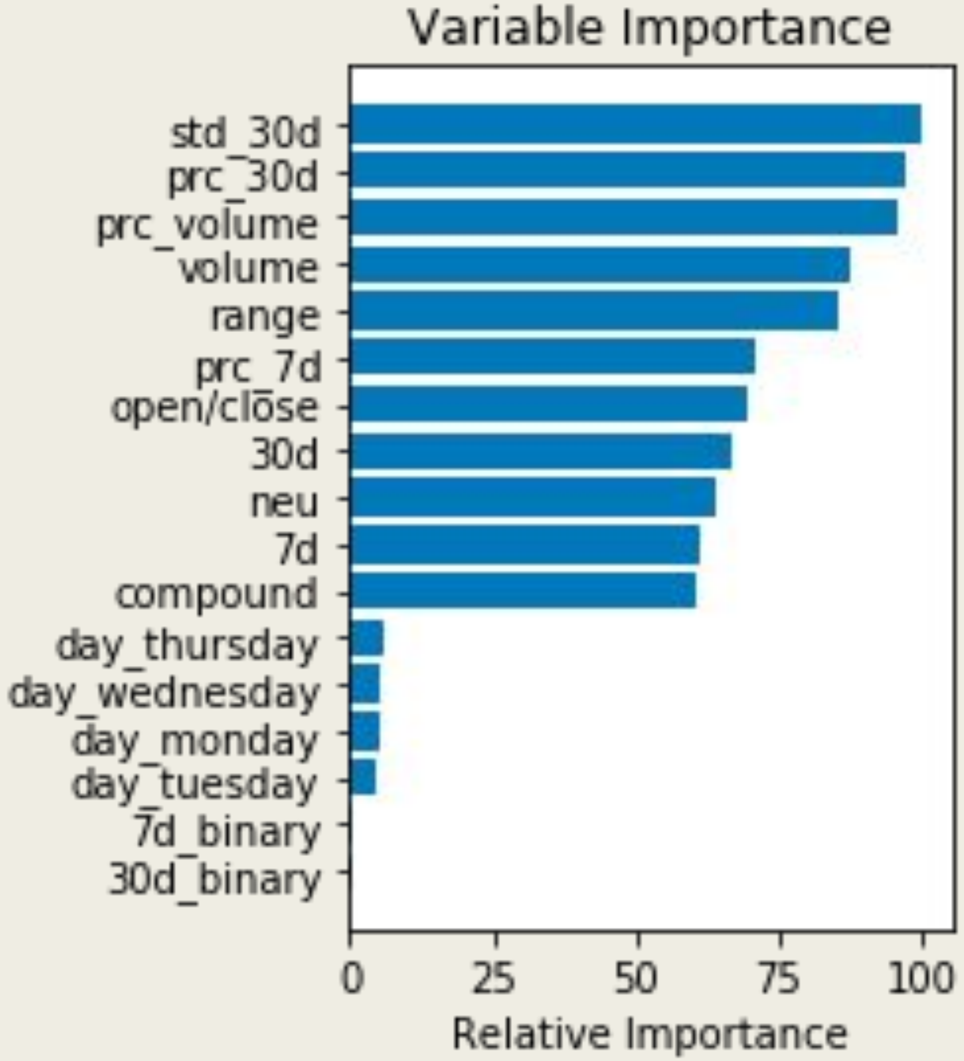
Future work:

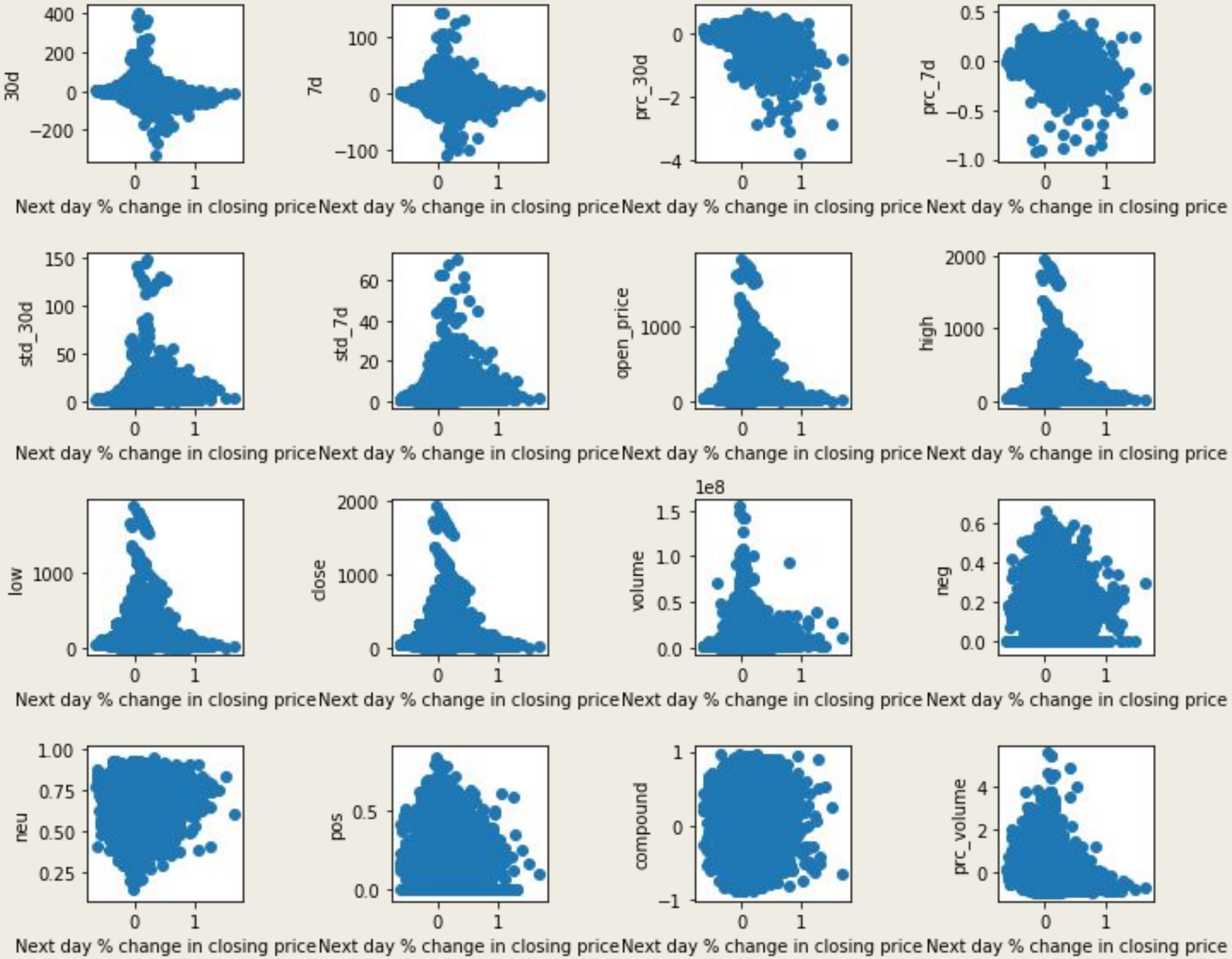
- Improve regression model
- More feature engineering
  - *Make more specialized dictionary for sentiment analysis*
- Pull additional stock history data
  - *Running averages*

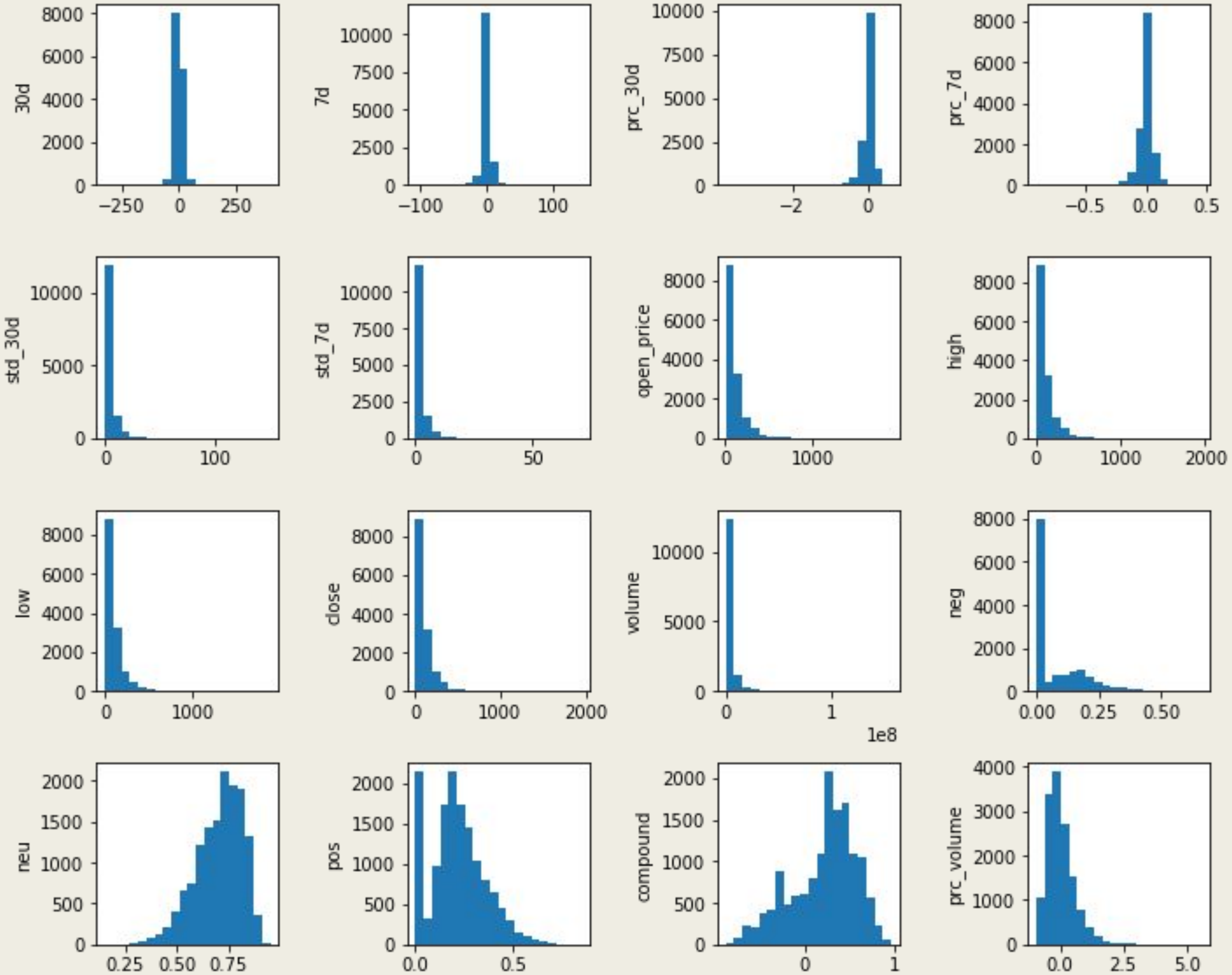
QUESTIONS?

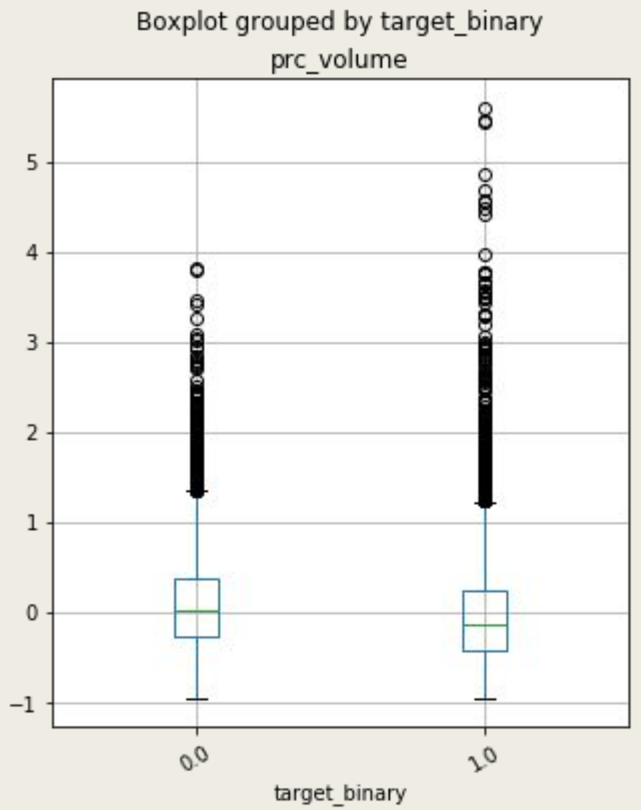
BACKUP SLIDES

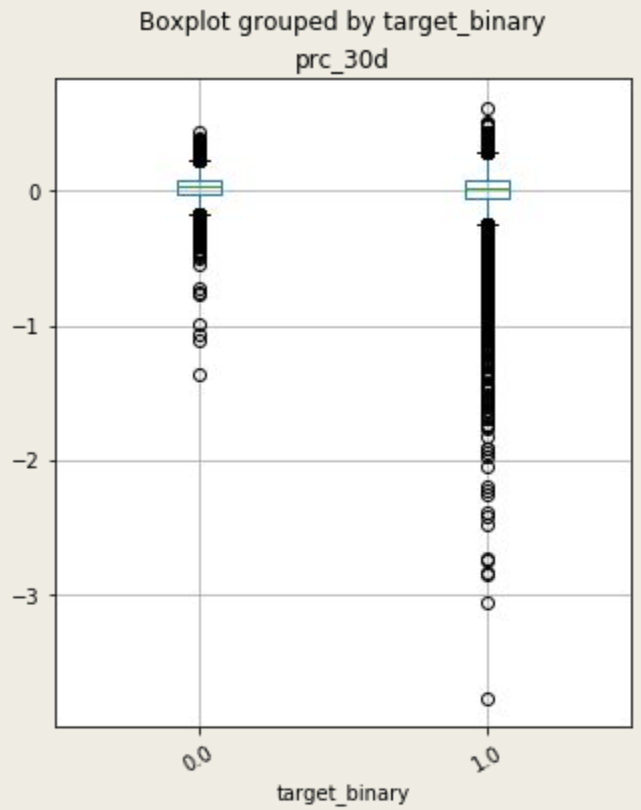
Feature importance of gradient boosting model with augmented features

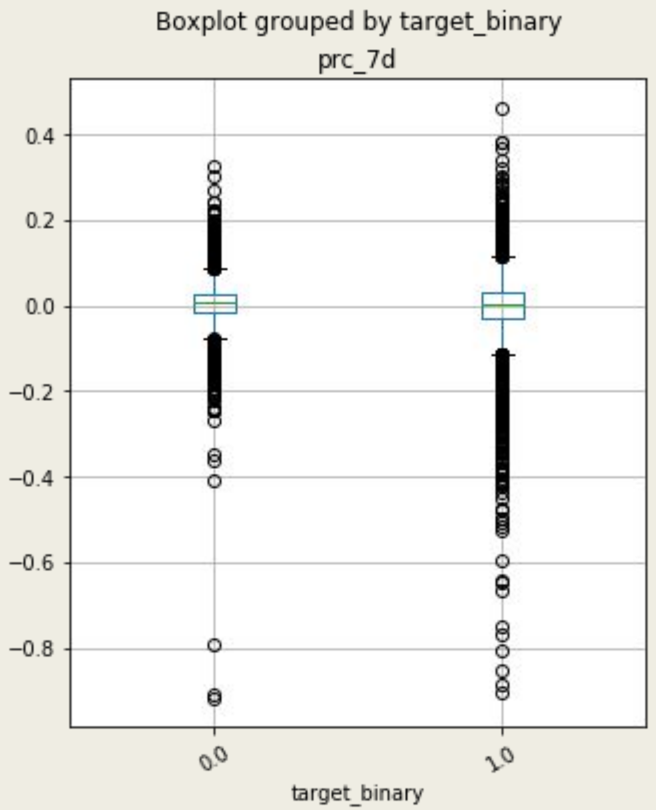



30d
7d
prc_30d
prc_7d
std_30d
std_7d
open_price
high
low
close
volume
1e8
neg
neu
pos
compound
prc_volume
Next day % change in closing price


30d
7d
prc_30d
prc_7d
std_30d
std_7d
open_price
high
low
close
volume
1e8
neg
neu
pos
compound
prc_volume


Boxplot grouped by target_binary
prc_volume
5
4
3
2
1
0
−1
0.0
1.0
target_binary


Boxplot grouped by target_binary
prc_30d
0
−1
−2
−3
0.0
1.0
target_binary


Boxplot grouped by target_binary
prc_7d
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0.0
1.0
target_binary