Supervised Learning:

Next Day Rise/Fall Stock Price Predictor

Kaci Kus

Initial data set after cleaning and organization

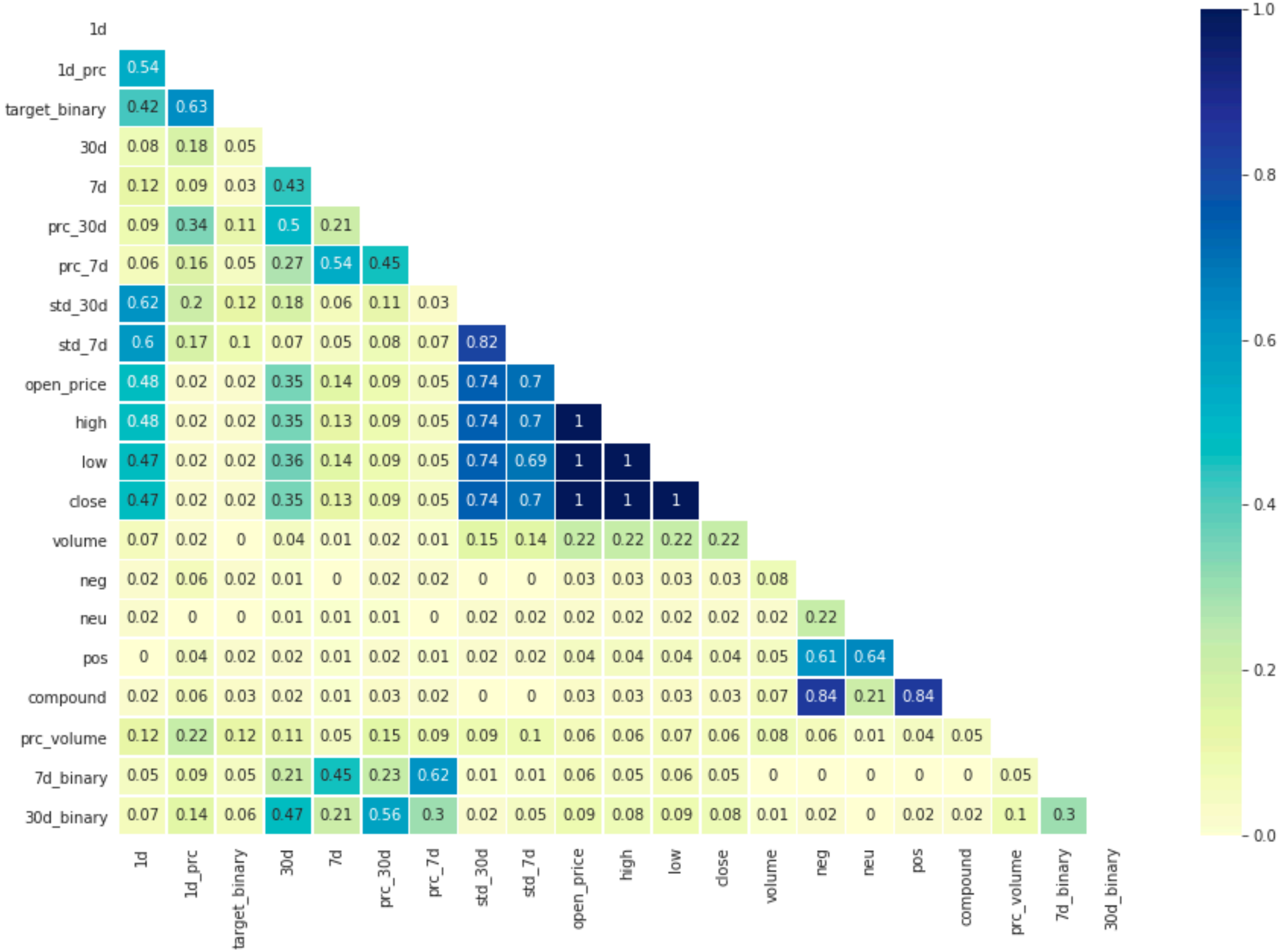
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14134 entries, 0 to 14133
Data columns (total 18 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   company      14134 non-null  object
 1   date         14134 non-null  object
 2   30d          14134 non-null  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
```

- Sentiment scores (neg, neu, pos, and compound) were averaged across all articles published in a day.
  - Excluding compound scores of zero
- Final data set has 14,124 observations

Target is binary:
0 : no change or decrease
1: positive next day change

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

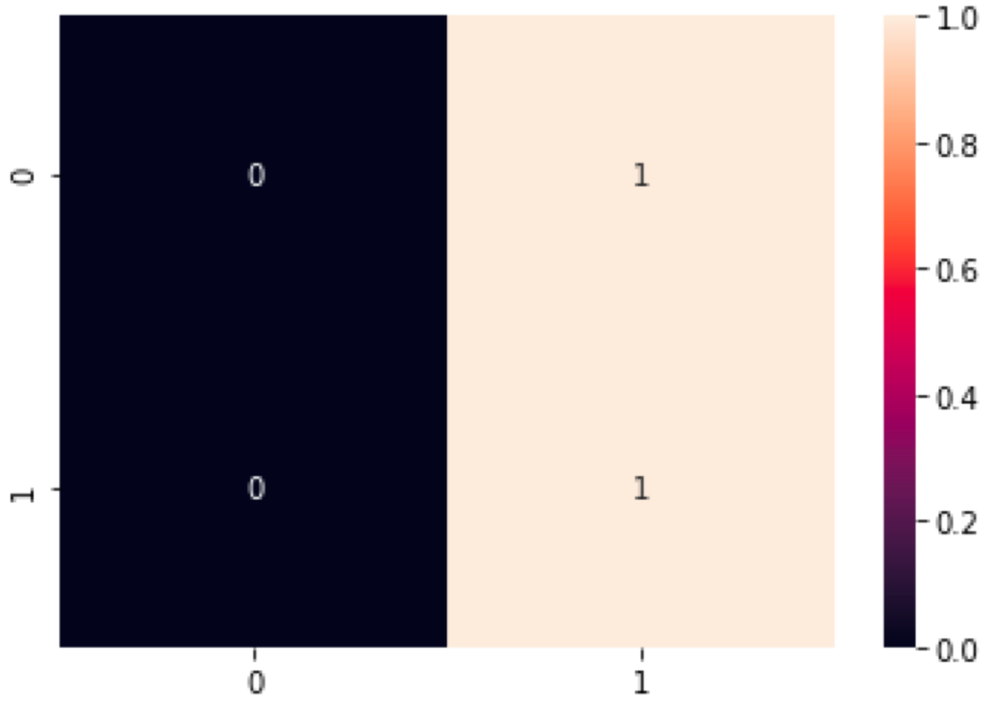
Initial correlation matrix



There is some collinearity that will need to be addressed, but for an initial model we will use all features as they are

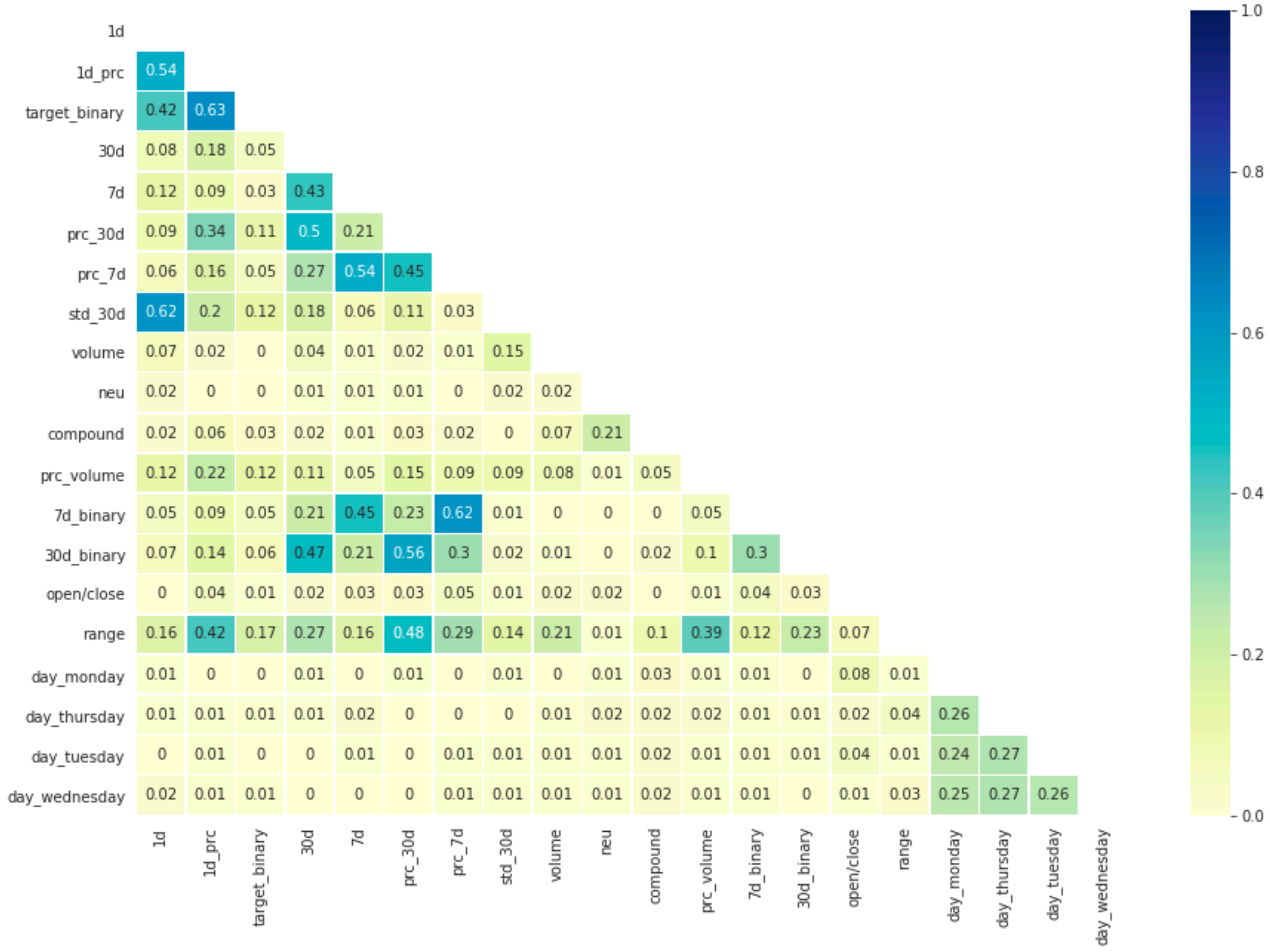
Base Logistic Regression

- Accuracy score is: 0.62
- Confusion matrix below



From what we know about the dataset, the base logistic regression model appears to simply be classifying everything as equal to 1, which indicates increasing the next day. This is a problem, because with this model we would never properly identify decreases the next day!

Feature engineering to add some new features, and take away the too strongly correlated features.



No features correlated > 0.7

Preparing the model

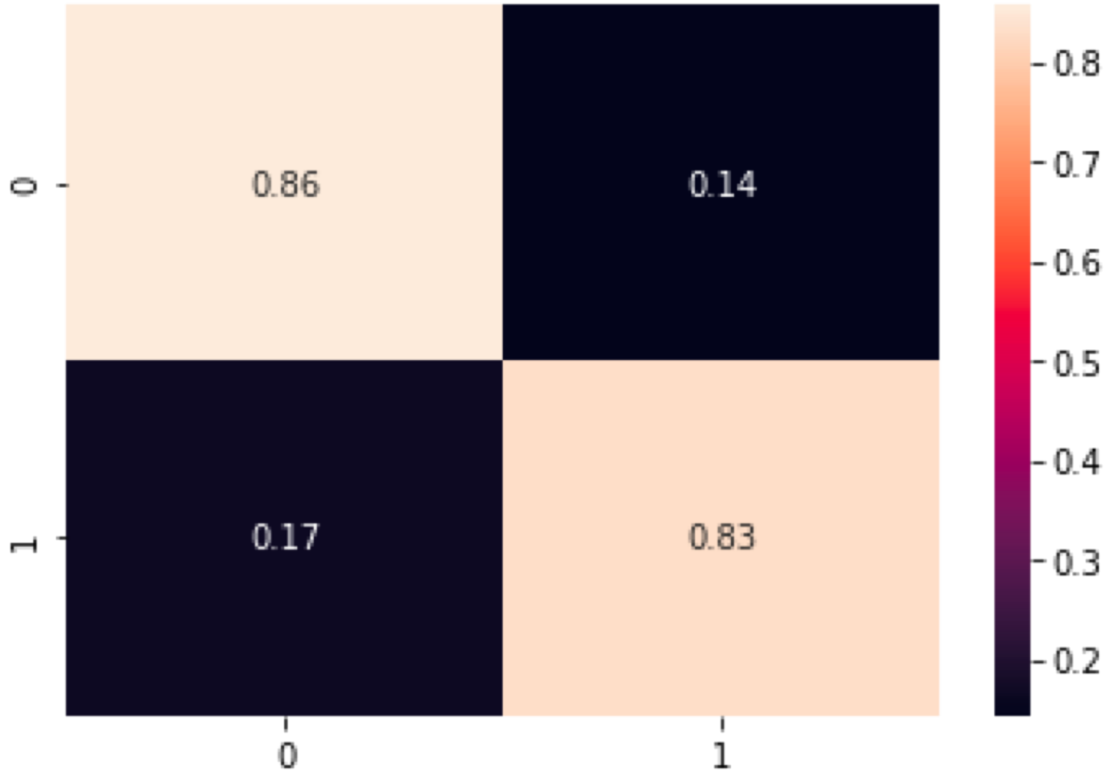
- Split into a training set and a test set
  - 20% of data moved to test set
- MinMaxScaler() used to scale the data from 0 to 1 to account for binary features (such as day of the week).
  - Scaler was fit to only the training feature set, and this scaler was then applied to the test feature set.
  - Target values were not scaled.
- GridSearchCV() used to identify optimal parameters for each model

Gradient Boosting after feature engineering

Accuracy: 0.84
Precision: 0.85
Recall: 0.83

From the confusion matrix, you can see that the accuracy for true positives (bottom right) and true negatives (top left) are still pretty good!

The false positive (top right) identification is also very much improved, and has changed from 100% to 14%



Other models with less optimal results

- Support vector machines model
- Random Forest
- K-Nearest Neighbor