Amazon Stock Movement Prediction

The Epic B Team

Parag Saxena

School of Data Science, the University of North Carolina at Charlotte

GITHUB link:

https://github.com/ParagDataEngineerScientist/DSBA6156Shared/blob/main/AmazonStockEpicB_6156.ipynb

Contents

1. Introduction

Stock Market is one of the most researched areas when it comes to AI and Machine Learning. There have been many different types of trading and investing that traders and investors implement. Trading is usually treated as riskier than investing.

Mainly, there are three different types of trading strategies:

Day Trading : A trader must close the trade by close of the stock market on that particular day.

Positional Trading : A trader executes a trade which they look to close in future from 1 day to 1 month of the period roughly .

Buy Today Sell Tomorrow (BTST) : Where a trader Buy or Sells a stock one day and closes the position the following day.

Our goal is to mostly help the third type of traders that work on BTST. We are mainly working on the Stock price movement of Amazon stock, and our assumption is that our machine learning model will generalize to other stocks as well. We have used data from Yahoo Finance which has features: Date, Open, High, Low, Close Volume and Adjusted-close.

2. Problem Statement

Ever since the beginning of the stock market, investors and analysts have aimed to predict stock prices accurately, as it is one of the most well-liked investments because of its huge profit margin. Though it can result in huge profits, there is an equal amount of risk involved in it. But to be profitable, we don't need to predict the exact price, instead, we just need to know whether it will be higher or lower than the price that is today to invest and make profits out of it. Therefore, the main target is to predict the stock price direction of Amazon whether the next day's closing price will be higher than the opening price.

3. Motivation

We are always amazed by the huge number of people who have an active investment in the stock market. There are many of them who have incurred losses, and only a few have made significant profits. Usually, the investors/traders who have their portfolio in stock markets are also employed in other jobs, so they can't actively devote their time to closely monitor their investments and trades. It will be beneficial for them to have a tool which enables them to predict the next day's stock movement. It will not only save time, but also can result in profitability.

4. Survey and Review of related research

Similar studies have tried two broad approaches for Amazon stock prediction using data ranging from 5 to 15 years.

- One approach was trying to predict the actual stock price value for the following day using Regression -
    - A Linear Regression method came up with an overall accuracy of 52%
    - Algorithm using Support Vector Machines (SVR) had an accuracy of 53%

- Another approach was to predict the direction of the stock movement whether it would increase or decrease the following day based on current day's close.
    - Algorithm using Support Vector Machines (SVC) had an accuracy of 55%
    - Long Short-Term Memory (LSTM) – Accuracy-based selection approach had a max accuracy of 56%

5. Open questions in the domain

- What is the accuracy number that research companies would target in order to generate long-term profits?
- How to rank indicators by the duration in which they have a higher effect on prediction, for example, short-term, long term or medium-term?
- In the case of stock movement prediction, how to improve the prediction by combining classical machine learning approaches with time series?

6. Approach summary

Pre-Processing and EDA

We have used the given data to calculate our Target and encoded it as 0 and 1. 0 means stock went down the next day, and 1 means that stock went up. We have used low, high, close and open prices to calculate many to of the stock indicators. We have calculated 40+ indicators based on the features given. Some of the important calculated features are MA (moving averages from 1 week to 100 days), EMA (exponential moving averages from 1 week to 100 days, MACD (Moving Averages Convergence and Divergence), RSI (Relative Strength Index) and many others.

After creating features and targets, we divided our data into test and train splits with a test size of 10%.

Model Building and Evaluation

We have used three approaches in model building, in the first approach we used all the Classification models that we learned and combined them to create a stacking classifier. We used

RandomSearchCV and GirdSeearchCV for each of the models, with cross-validation, k=5 using K-Folds. This step took a huge amount of time to come up with the best hyperparameters, but ROC-AUC accuracy was only 51%.

In the second approach, we tried the same hyper-tuning approach with backtracking and Time Series cross-validation split with k=5. This model was faster to train and test but gave a ROC-AUC accuracy of 52%, with stacking.

In the third approach, we investigated only the third-best performing algorithm and tuned them, and added features like a month, day and year by decomposing date as we were not adding information related to date previously. After using a stacking classifier on this combination, we got an accuracy of 57% which outperformed the other studies we found on Amazon stock price movement.

7. Background

As the purpose of this project, we are trying to predict Amazon stock price movement. Our aim is to correctly classify the stock direction for the next day to help BTST traders. We have studied a good number of approaches which are available on the internet and have taken inspiration from them to come up with our unique solution to address the same problem. we have seen that for other stocks accuracies are higher using the same approaches but due to volatility and not a predictive trend in the case of Amazon stock, the accuracies of all the approaches and models are just a little better than a random guess.

8. Overall Framework details of algorithms and methods

Approach-1 Algorithms used:

XGBoost , Random Forest, Logistic Regression, GaussianNBF, Support Vector Machine, XGRFClassifier.

Created scikit-learn pipeline for each of the seven classifiers using

- Scaling the features(Min Max)
- RandomSearchCV
- GridSearchCV

Example of Pipeline:

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
steps = [('rf',rf)]
pipeline_rf = Pipeline(steps)

criterion = ['gini','entropy','log_loss']
max_depth = np.arange(2,20)
min_samples_split = np.arange(2,5)
min_samples_leaf = np.arange(1,5)
n_estimators = np.arange(50,5000,50)
class_weight = ['balanced','balanced_subsample']
parameters = {
    'rf__criterion':criterion,
    'rf__max_depth':max_depth,
    'rf__min_samples_split':min_samples_split,
    'rf__min_samples_leaf' : min_samples_leaf,
    'rf__n_estimators' : n_estimators,
    'rf__class_weight': class_weight }

pipeline_rf_cv_rm = RandomizedSearchCV(pipeline_rf,parameters,n_iter=100,verbose=2,cv=5,n_jobs=-1)
results_rf_random = pipeline_rf_cv_rm.fit(X_train,Y_train)
print('Best Score RS %s'% results_rf_random.best_score_)
print('Best Parameters %s' %results_rf_random.best_params_)

print(pipeline_rf_cv_rm.best_estimator_.get_params()['rf'])
```

Approach-2 Algorithms used:

For the second approach we used Random Forrest Classifier with backtracking instead of K-fold cross-validation. By using step size of 750 and set a threshold as .6 to classify the target as 1 or 0. Following code snippet shows the implementation of this approach.

```
[ ] i = 1000
    step = 750

    train = data.iloc[0:i].copy()
    test = data.iloc[i:(i+step)].copy()
    rfc.fit(train[predictors], train["Target"])
    preds = rfc.predict(test[predictors])
```

```
    preds = rfc.predict_proba(test[predictors])[:,1]
    preds = pd.Series(preds, index=test.index)
    preds[preds > .6] = 1
    preds[preds<=.6] = 0
```

+ Code + Text

```
[ ] predictions = []
    # Loop over the dataset in increments
    for i in range(1000, data.shape[0], step):
        # Split into train and test sets
        train = data.iloc[0:i].copy()
        test = data.iloc[i:(i+step)].copy()

        # Fit the random forest model
        rfc.fit(train[predictors], train["Target"])

        # Make predictions
        preds = rfc.predict_proba(test[predictors])[:,1]
        preds = pd.Series(preds, index=test.index)
        preds[preds > .6] = 1
        preds[preds<=.6] = 0

        # Combine predictions and test values
        combined = pd.concat({"Target": test["Target"],"Predictions": preds}, axis=1)

        predictions.append(combined)
```

Approach-3 algorithms used:

Finally used Logistic Regression, XGBRFClassifier and Bernoulli's Naive Bayes and used stacking classifier to merge the results of all these algorithms. We used Bayesian optimization for XGBRFClassifier, and for stacking classifiers we used Logistic Regression with default setting as its default applies ridge regression. And Bernoulli-NB was performing best in default settings and doesn't need hyperparameter tuning .

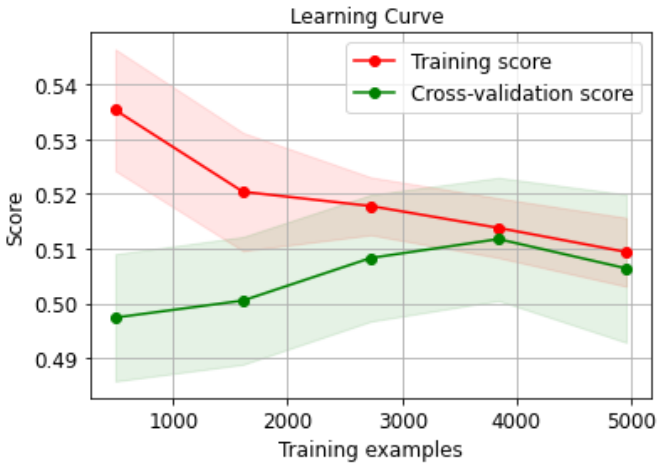
```
from sklearn.ensemble import StackingClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
lr = LogisticRegression(penalty='l2')
level0 = list()
level0.append(('bayes', BernoulliNB()))
level0.append(('xgbf',XGBRFClassifier(n_estimators=5000,max_depth=12,learning_rate=0.001,gamma=3.9,colsample_bynode=0.2,colsample_bytree=0.54,reg_alpha
level0.append(('lrf',LogisticRegression()))

clf= StackingClassifier(estimators =level0, final_estimator = lr,cv=10,n_jobs=-1)
```

9. Results

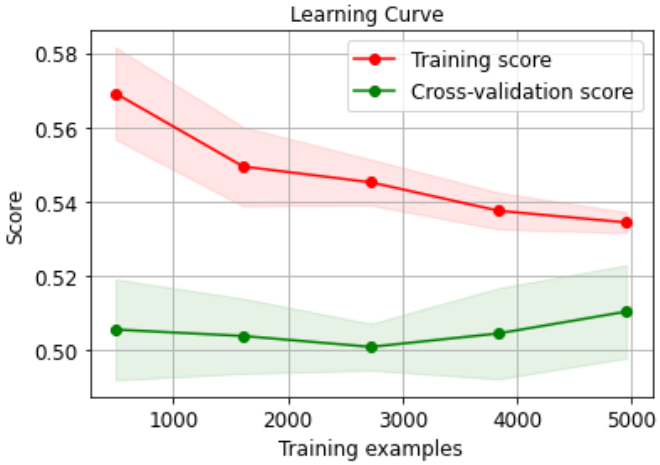
Accuracies and Cross-validation results for all the algorithms tested:

```
LogisticRegression() :
Training ROC :  0.5136037468120659
Validation ROC :  0.5319008904374758
Training Accuracy :  0.5093393289519198
Validation Accuracy :  0.5063201703445606
SVC(probability=True) :
Training ROC :  0.5437082798146556
Validation ROC :  0.5223964382500967
Training Accuracy :  0.5375302663438256
Validation Accuracy :  0.5290021293070073
BernoulliNB() :
Training ROC :  0.5186364073169392
Validation ROC :  0.5692895857530004
Training Accuracy :  0.5172950536146662
Validation Accuracy :  0.5637969415408439
RandomForestClassifier(n_estimators=2000) :
Training ROC :  1.0
Validation ROC :  0.46195799457994585
Training Accuracy :  1.0
Validation Accuracy :  0.4752710027100271
XGBRFClassifier() :
Training ROC :  0.5844267139868913
Validation ROC :  0.5436653116531166
Training Accuracy :  0.5541335178139052
Validation Accuracy :  0.533778552071235
LGBMClassifier() :
Training ROC :  0.8471733342173166
Validation ROC :  0.4661730545876887
Training Accuracy :  0.7632307160152196
Validation Accuracy :  0.4826413085559427
XGBClassifier() :
Training ROC :  0.6778406166404802
Validation ROC :  0.5042295780100658
Training Accuracy :  0.6150121065375302
Validation Accuracy :  0.5188975996902827
```

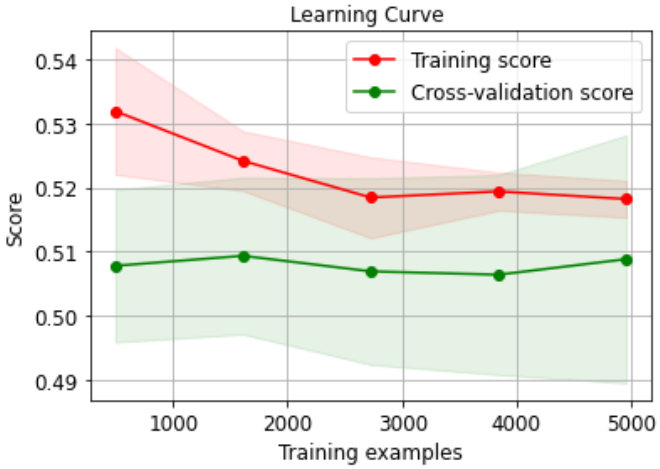
Logistic Regression:



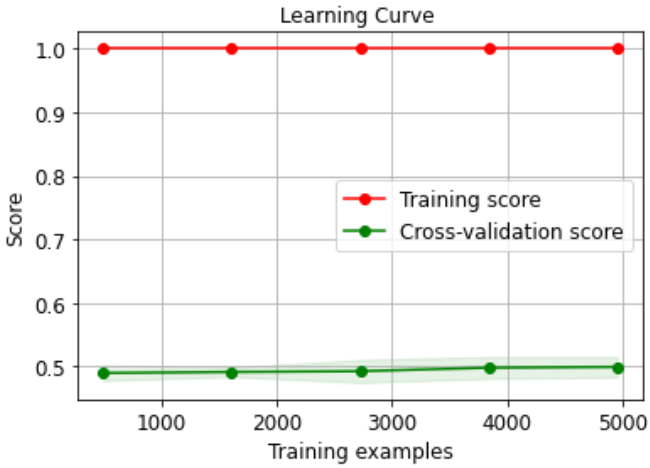
SVM:



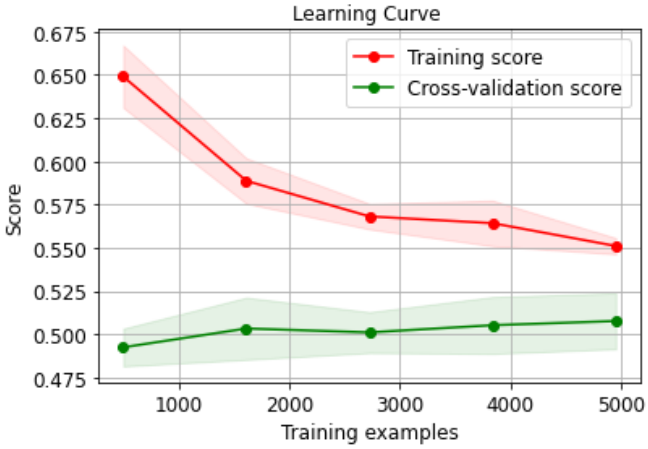
BernoulliNB:



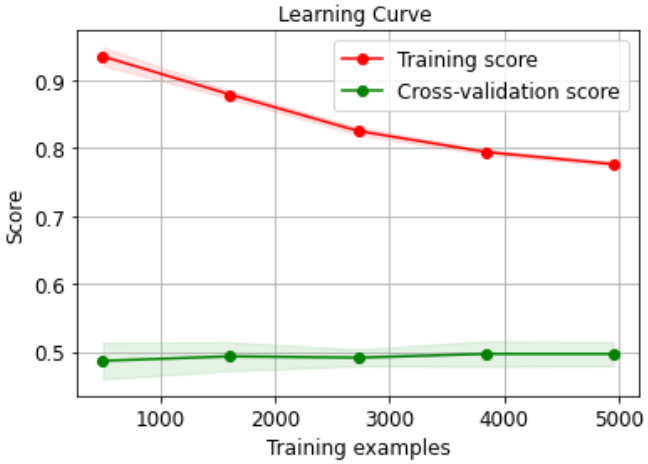
Random Forrest:



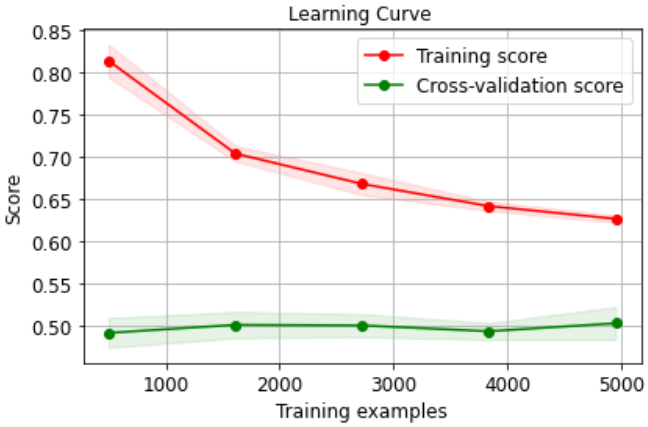
XGBRFClassfiier:



LightGBM:



XGBClassifier:



10. Conclusion

Achieved AUC-ROC score of 57% (slightly better than the similar studies available online), using more than 40 features based on opening and close price of amazon stock.
Implemented around 8 classifiers including Boosting, Bagging and Stacking Classifiers to sum up the results of other heterogeneous classifiers.

We are looking to apply NLP-TEXT Analytics to add feature for sentiment analysis by web scrapping News/Twitter/ other social-media. This feature can take three categories for positive, neutral and negative sentiment(-1,0,1).

11. Learnings

As your project is predicting Amazon Stock price movement, it's a project that many financial organizations may have been working on.

Major learnings from our project are:

1. Domain expertise is required at the time of feature engineering. We have spent a lot of time in feature engineering guessing that some of the features might be more important.
2. Highly rated Algorithms like Random Forrest or XGboost may not always give an optimal best performance even after rigorous Hyper-parameter tuning.
3. Aiming for high accuracy or AUC-ROC score is not a good idea, before understanding the business objective of the problem.
4. We realized the correct way of using Cross-Validation, the biggest drawback of cross validation is that there might be a huge difference in accuracy score for out of sample data.
5. We were also able to comprehend, how the ML code pipelines are created for real world projects

12. References

https://medium.com/@nutanbhogendrasharma/predict-amazon-inc-stock-price-with-machine-learning-ee05afb5f419

https://www.kaggle.com/code/aenes95/amazon-stock-price-and-movement-prediction

https://binalkagathara.medium.com/stock-predictor-using-amazon-forecast-d3a146f3a7ba

https://finance.yahoo.com/quote/AMZN?p=AMZN&.tsrc=fin-srch