451 Financial Engineering: Programming Assignment 1

Prepared by Logan MacPhail

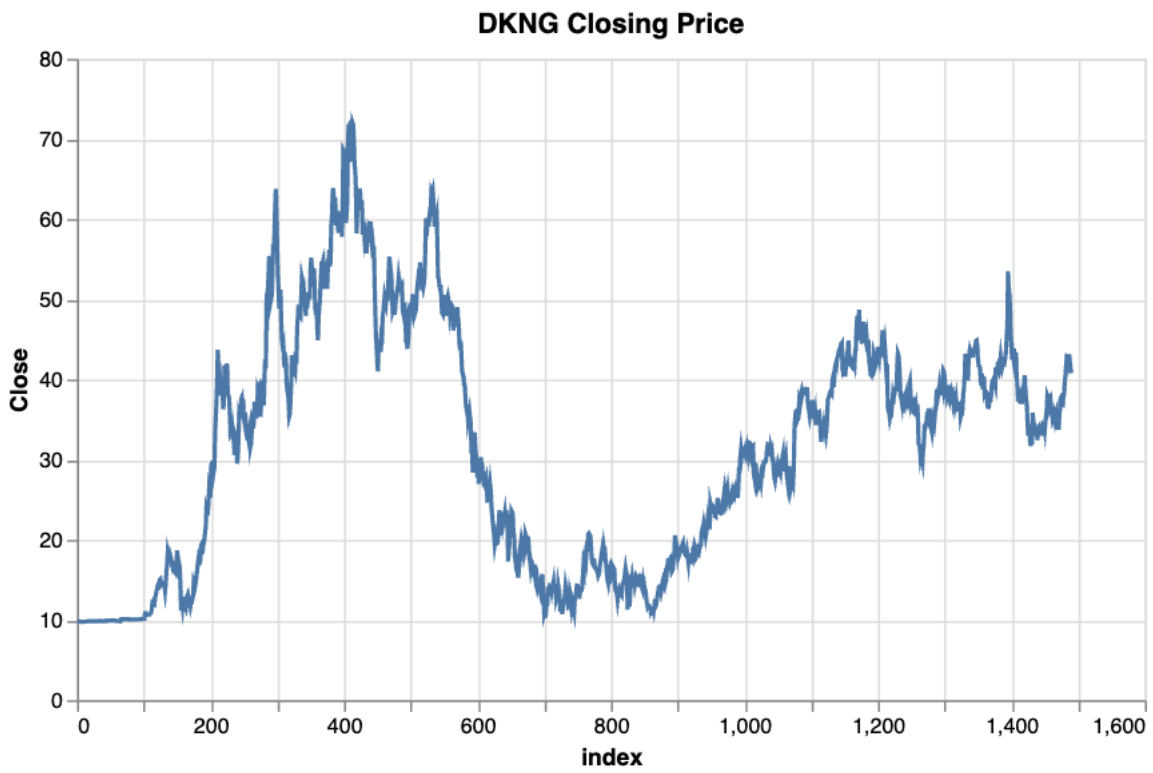
July 6, 2025

1. Problem Description

This programming assignment focuses on predicting the direction (up or down) of returns for an individual stock the following day. For this assignment, the stock being used is DraftKings Inc. (DKNG). DraftKings is an online gambling company that started in 2012 and offers sportsbook and daily fantasy sports services. To predict the direction of the DKNG stock price, several variables and model types will be evaluated for classification. Those variables include lags of DKNG's opening, closing, high, and low prices, as well as the daily trading volume.

2. Data Preparation and Pipeline

The data used for this assignment comes from the *yfinance* module (yf) in Python, which is part of the Yahoo Finance package that contains daily stock prices for most publicly traded stocks on the New York Stock Exchange. The closing, opening, high, and low prices, as well as the daily trading volume, were imported using yf. . Given that DKNG started in 2012, the entire stock's trading history was imported. Below is a look at the closing price of DKNG over time.



After importing the price and volume variables for the DKNG stock, several other custom metrics were created using these variables, including HML, which is the high price minus the low price. OMC was also created, which is the opening price minus the closing price. Lagged versions of each variable were also created, representing those same variables on previous days, extending back as far as three days prior. For example, *CloseLag1* represents the closing price for DKNG the previous day. Finally, an exponential moving average of the closing prices was created, looking back at the past two, four, and eight day averages.

To determine the direction of the stock, the *LogReturn* variable was created by taking the natural log of the closing price divided by the one-lagged version of the closing price, representing the percentage change in the closing price from the previous day to the current day. The response variable *Target* was created as a one-hot encoded variable, where '1' signifies an upward

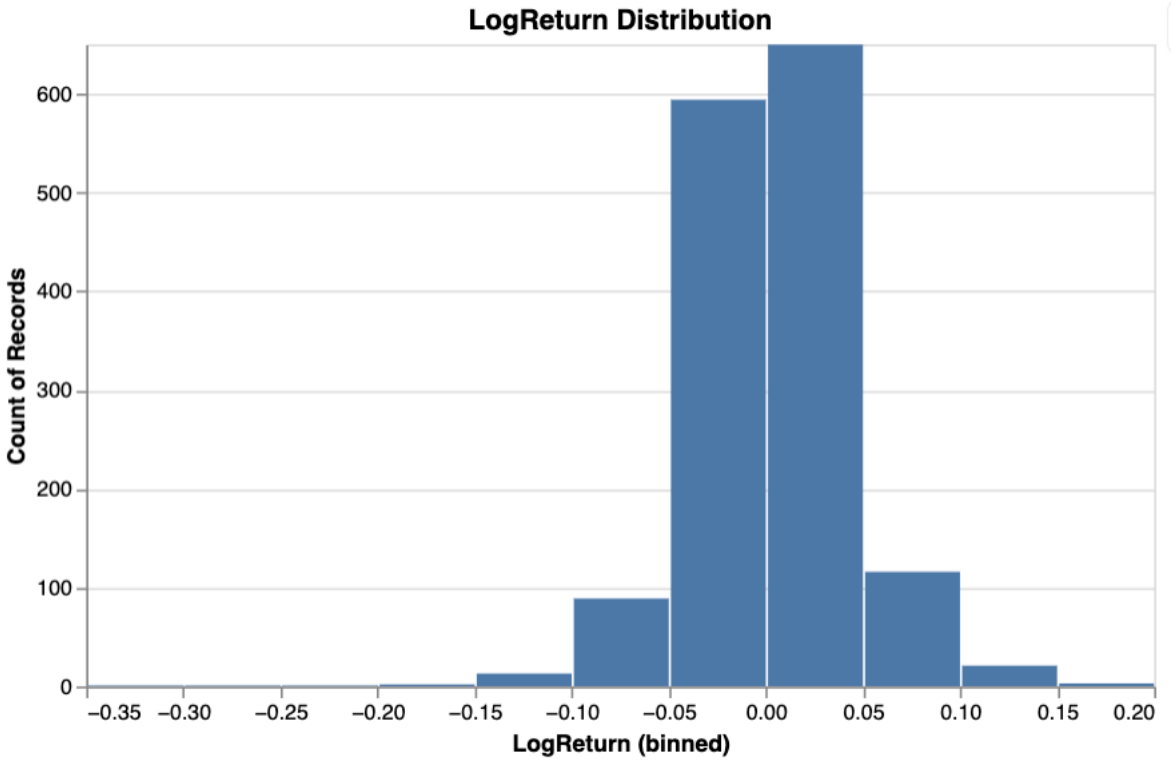
direction (i.e., *LogReturn* greater than zero) and '0' represents a downward direction (i.e., *LogReturn* less than zero). Not including *LogReturn* and *Target*, and the current day's prices, there are a total of 15 variables that can be used in the model, potentially.

1. **CloseLag1:** Lag-one daily closing price
2. **CloseLag2:** Lag-two daily closing price
3. **CloseLag3:** Lag-three daily closing price
4. **HMLLag1:** Lag-one high minus low daily prices
5. **HMLLag2:** Lag-two high minus low daily prices
6. **HMLLag3:** Lag-three high minus low daily prices
7. **OMCLag1:** Lag-one open minus closing daily prices
8. **OMCLag2:** Lag-two open minus closing daily prices
9. **OMCLag3:** Lag-three open minus closing daily prices
10. **VolumeLag1:** Lag-one daily volume
11. **VolumeLag2:** Lag-two daily volume
12. **VolumeLag3:** Lag-three daily volume
13. **CloseEMA2:** Exponential moving average across two days
14. **CloseEMA4:** Exponential moving average across four days
15. **CloseEMA8:** Exponential moving average across eight days

Before feature selection was started, each of the variables was standardized to a mean of 0 and standard deviation of 1 using the StandardScaler function from Sci-Kit Learn's preprocessing module.

3. Research Design

The first thing evaluated was the distribution of *LogReturn* to determine whether *Target* was balanced. *LogReturn* represented the response variable on a continuous scale, which provided a clearer understanding of the distribution. Below is a look at the distribution for *LogReturn* and *Target*.



Target	count
i32	u32
1	718
0	773

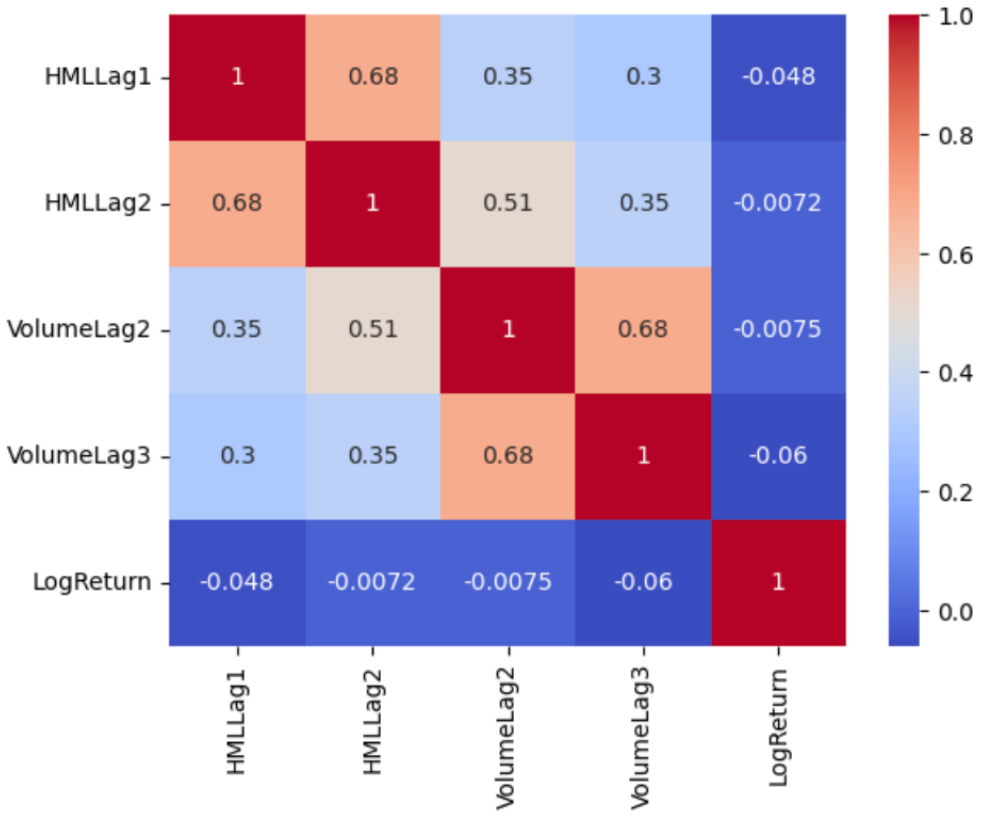
Outside of a few outliers, seemingly when the stock dropped precipitously around index 600, the response variable seems relatively balanced.

Since the dataset is a time series, it cannot be randomly split into training and test datasets. The TimeSeriesSplit function from Sci-kit Learn splits a dataset into training and test splits, while maintaining the sequential order of the time series without any data leakage. TimeSeriesSplit does not create separate datasets, but instead designates which portion of the entire dataset will be used for training and testing with indices. TimeSeriesSplit was used to create five splits of training and test sets.

For feature selection, a logistic regression model was used along with every possible subset of the 15 variables. The Akaike information criterion (AIC) of each model was recorded, and the top 10 performing models were evaluated to determine which variables showed up the most often. The following four features were the variables that showed up the most often

1. **HMLLag1:** Lag-one high minus low daily prices
2. **HMLLag2:** Lag-two high minus low daily prices
3. **VolumeLag2:** Lag-two daily volume
4. **VolumeLag3:** Lag-three daily volume

The correlation matrix shows that each of the variables are somewhat correlated with each other, but not very correlated with the *LogReturn* variable, so the model is not likely to be very accurate.



4. Models

With the data now ready for modeling, four different model types —Logistic Regression, Support Vector Machine, XGBoost, and Light GBM —will be evaluated to determine which one best fits the data. To compare the models against each other, a combination of accuracy on the test dataset, along with overall AUC, precision, and recall on the complete dataset, will be used to determine the best model. Each model will start with a baseline fit, and then the optimal hyperparameters will be selected by using a randomized search over an array of potential settings, using the RandomizedSearchCV from Sci-Kit Learn. Once the optimal hyperparameters have been determined, a final model will be fit with those settings and evaluated on the test dataset.

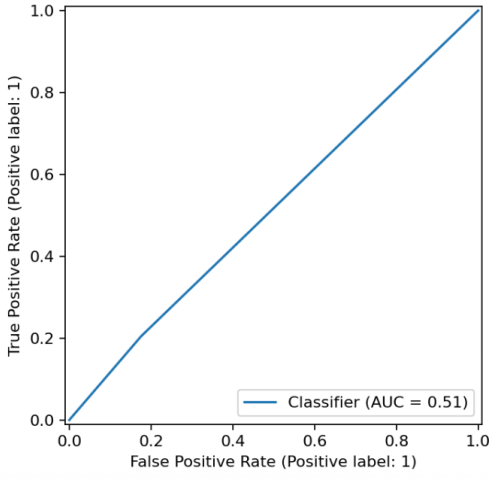
5. Results

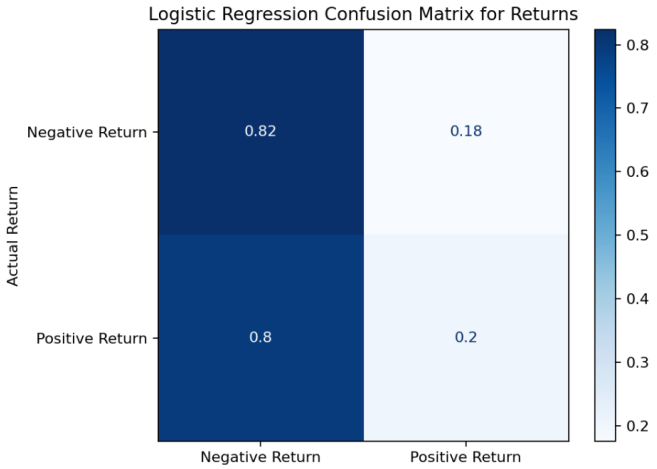
Logistic Regression

The first model that was fit was the Logistic Regression. The initial baseline model had a mean accuracy of 49.8% with a 5.6% standard deviation. The hyperparameters that were searched on were penalty, C, and solver. Below were the optimal hyperparameters determined by the random search:

- **Penalty:** L2
- **C:** 1.20
- **Solver:** Newton-CG

The model ended up with a 50.9% accuracy on the test dataset, so slightly better than the initial baseline model. Below is a look at the ROC curve and confusion matrix for the logistic model.





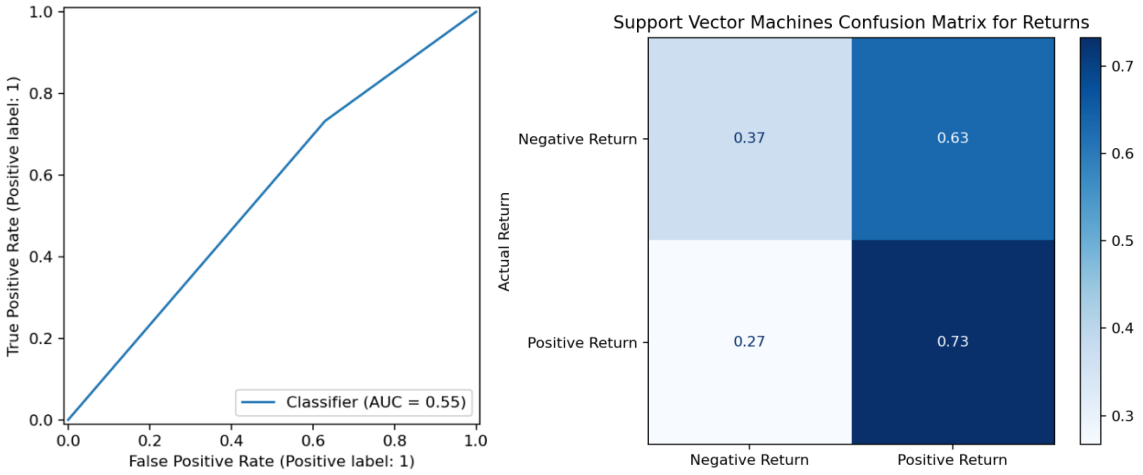
The AUC for the ROC curve is only .51, barely better than chance level. We can see from the confusion matrix that the model is overly biased in predicting negative returns. Only 20% of the days with positive returns were predicted correctly.

Support Vector Machines

The next model that was fit was a support vector classifier. The initial model was fit with a radial basis function kernel and balanced class weights. The baseline support vector model had an accuracy of 51.5% with a 1.9% standard deviation, so the initial support vector model was more accurate than the tuned logistic regression model. The hyperparameters that were searched on were C and Gamma. Below were the optimal hyperparameters determined by the random search:

- **C:** 0.3
- **Gamma:** Scale (automatically calculated using 1 / (number of features * variance of input data)

The hyperparameter tuning increased the accuracy to 51.9% on the test dataset. Below is a look at the ROC curve and confusion matrix for the support vector model.



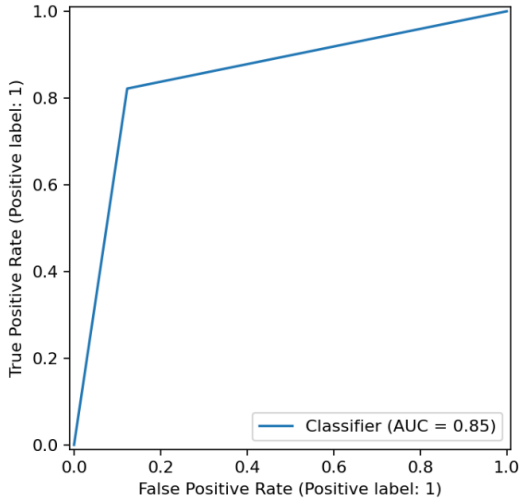
The AUC of 0.55 is slightly better than the Logistic model; however, we can see that the support vector model is biased in the opposite direction of the logistic model, in that it predicts positive returns too often. Only 37% of the days with negative returns were predicted correctly.

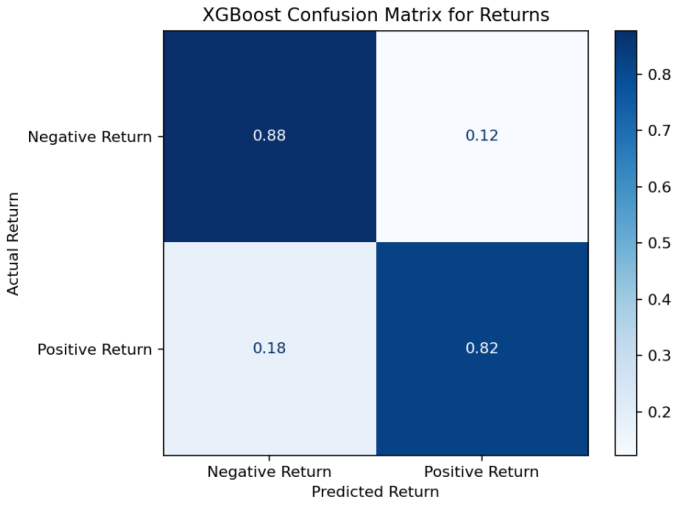
XGBoost

The next model that was fit was an XGBoost classifier. The initial model was fit with 1000 estimators. The baseline model had an accuracy of 49.6% with a 5.5% standard deviation, similar to the logistic regression model. The hyperparameters that were searched on were max depth, min child weight, subsample, learning rate, and number of estimators. Below were the optimal hyperparameters determined by the random search:

- **Max Depth:** 9
- **Min Child Weight:** 9
- **Subsample:** .504
- **Learning Rate:** 0.089
- **Number of Estimators:** 273

The hyperparameter tuning increased the accuracy to 49.9% on the test dataset. Below is a look at the ROC curve and confusion matrix for the support vector model.





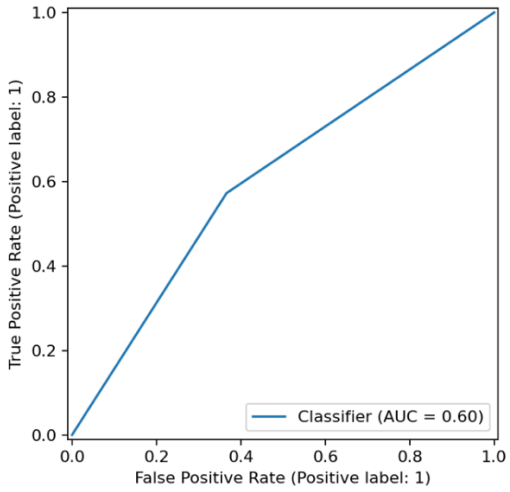
Despite a sub 50% testing accuracy, the AUC and confusion matrix for the XGBoost model are significantly better than the logistic or support vector models. This may suggest that the XGBoost model has overfit the training data, given the poor performance on the test data.

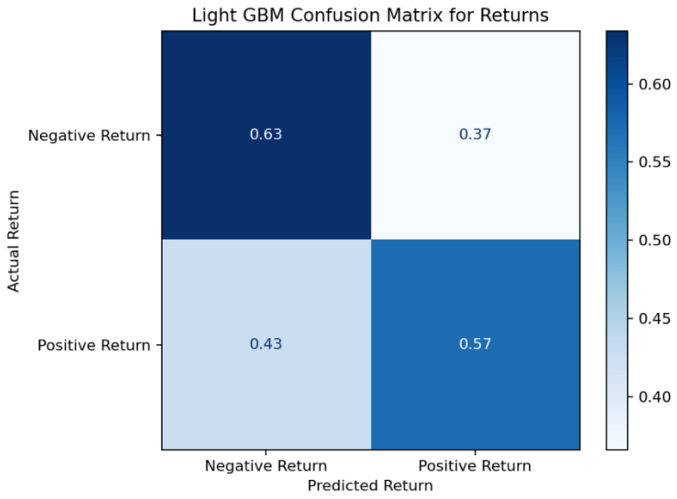
Light GBM

The final model that was fit was a Light GBM classifier. The baseline model had an accuracy of 47.5% with a 3.7% standard deviation, which was the lowest of the baseline models. The hyperparameters that were searched on were max depth, number of leaves, min data in leaf, learning rate, and bagging fraction. Below were the optimal hyperparameters determined by the random search:

- **Max Depth:** 4
- **Number of Leaves:** 109
- **Min Data in Leaf:** 226
- **Learning Rate:** 0.045
- **Bagging Fraction:** 0.859

The hyperparameter tuning increased the accuracy to 51.3% on the test dataset. Below is a look at the ROC curve and confusion matrix for the support vector model.





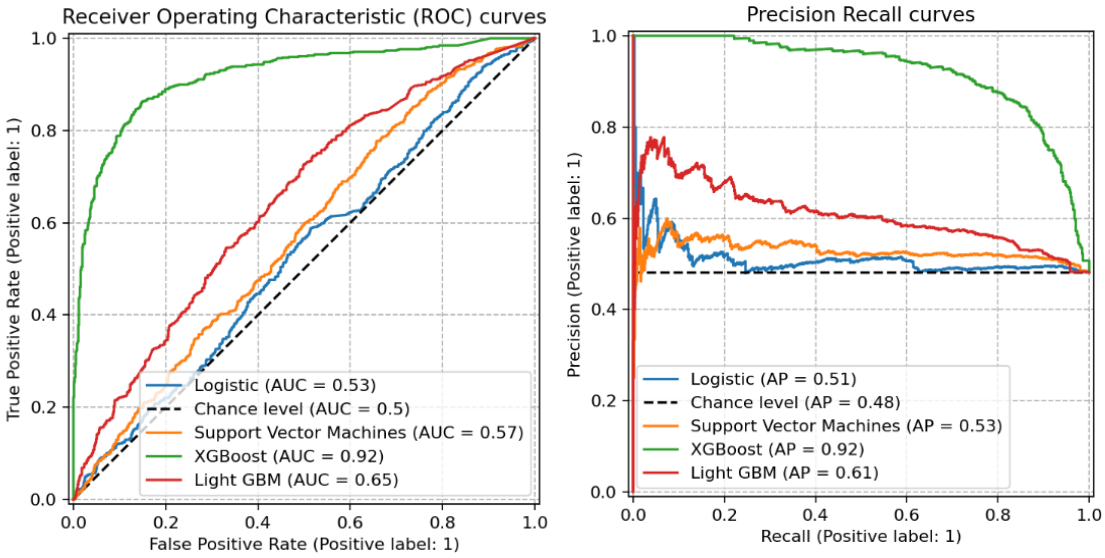
The AUC for the Light GBM model was better than the logistic or support vector models, but still well below the XGBoost model. Like the XGBoost model, the confusion matrix is well

balanced between predicting negative or positive returns, however, the Light GBM precision score for both classes was around 60%, while the XGBoost precision scores were around 85%.

Model Comparison

Each of the models performed better in certain areas compared to the others. The logistic model was the worst of the four models. It was overly biased in predicting negative returns, and the accuracy on the test dataset was the 3rd best. The support vector model was the second worst in terms of precision and recall on the entire dataset, but had the best test accuracy of all the models. The XGBoost was the opposite in that it was vastly superior to the other models in precision and recall, but had the worst test accuracy of all the models, likely indicating the model may have overfit the training data. The Light GBM was a balance between the support vector and XGBoost. It had better precision and recall than the support vector, but not as good as the XGBoost, but it was more accurate on the test dataset compared to the XGBoost, although not as good as the support vector model. The table and charts below show the final results for each of the models.

Model	Precision	Recall	AUC	Test Accuracy
Support Vector	0.56	0.55	0.55	**51.9%**
Light GBM	0.60	0.60	0.60	51.3%
Logistic	0.52	0.51	0.51	50.9%
XGBoost	**0.85**	**0.85**	**0.85**	49.9%



6. Conclusion

None of the models generalized well to the test data. The top accuracy on the test data was 51.9% using a support vector classifier. The XGBoost performed well overall, but given its lower accuracy on the test dataset, there's a reasonable belief that the model overfit the training dataset. The support vector and logistic models are biased in opposite directions, so there's potential to ensemble the models together to improve the accuracy, which may prove beneficial given that the support vector had the highest test accuracy.

There are potential options to explore to improve the accuracy of predicting the direction of DKNG's stock price. It may be worth looking for stocks in the same industry, such as FanDuel (FLUT) and other sports/gambling related businesses on the NYSE to try and correlate DKNG's movement with theirs. We may also want to try other techniques such as sequential modeling with Neural Networks to determine the optimal window to evaluate DKNG's stock movement. We defaulted to lagging to the past 3 days, but maybe the optimal window is further back. We may also want to monitor news about the gambling industry in general, especially governmental regulation.