
lululemon

INTRODUCTION

Lululemon is a Canadian athletic apparel retailer renowned for its high-quality yoga and activewear clothing.

Lululemon's major business lines encompass apparel, accessories, and footwear.

Lululemon's primary profit driver has been its apparel segment, fueled by innovative designs, and high-quality fabrics appealing to its core customer base.

Lululemon's optimistic outlook is driven by its continuous innovation in product development, expansion into new markets, online sales focus, and efforts to diversify its product offerings. And

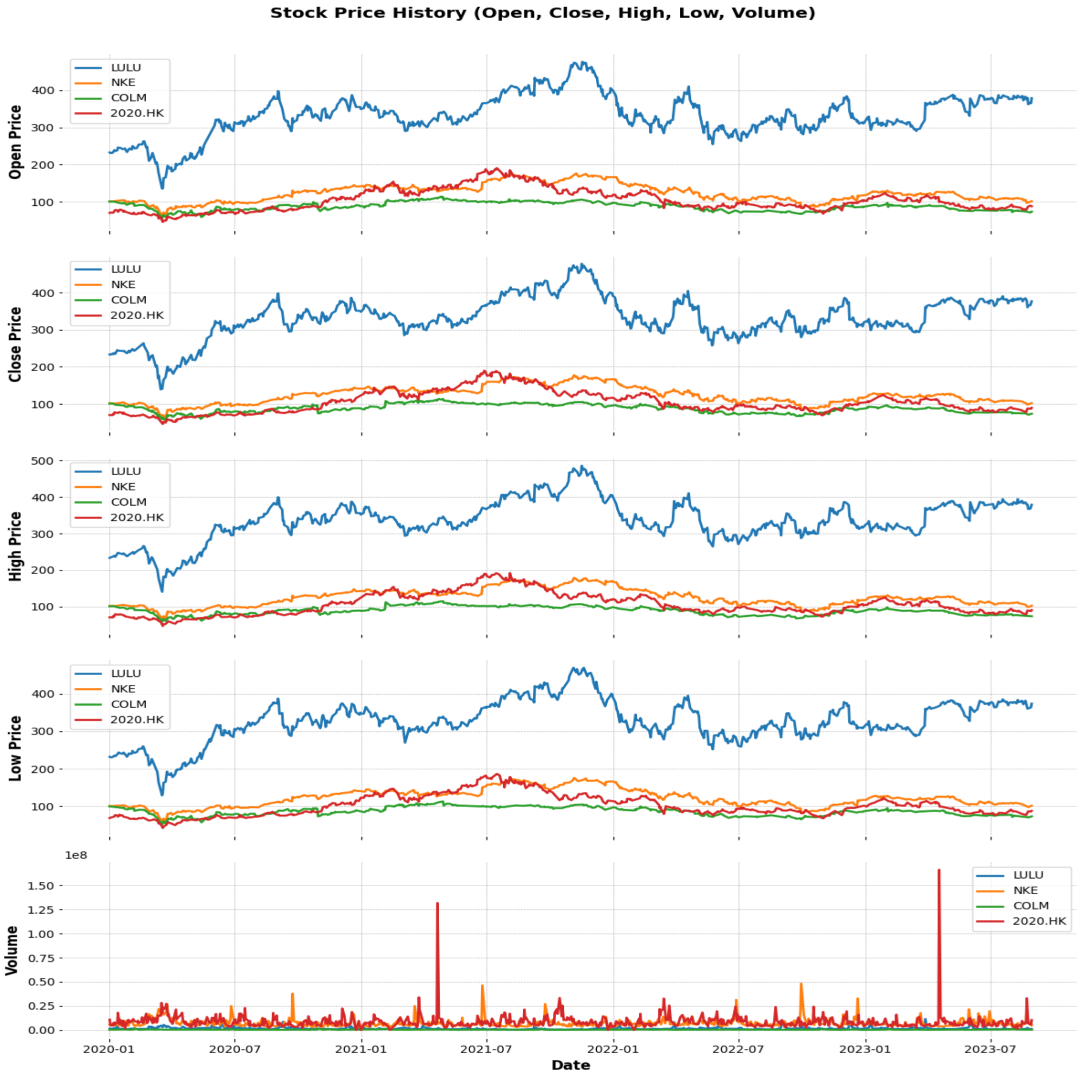
LULULEMON BUSINESS LINE

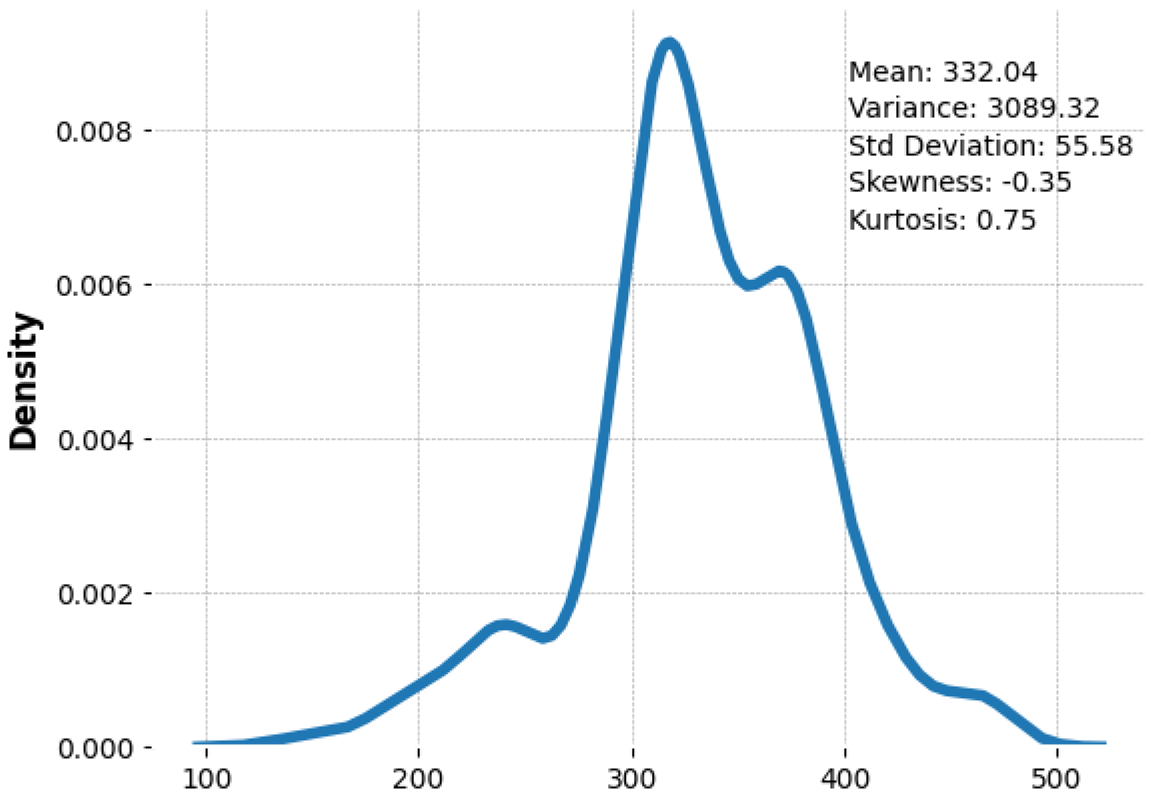
- Yoga and Athletic Apparel
- Men's and Women's Collections
- Athleisure
- Footwear
- Online Sales
- Retail Stores
- Fitness and Community Initiatives

Lululemon stock price history

Competitors

Stock price history comparison with competitors





Statistical Analysis of Lululemon

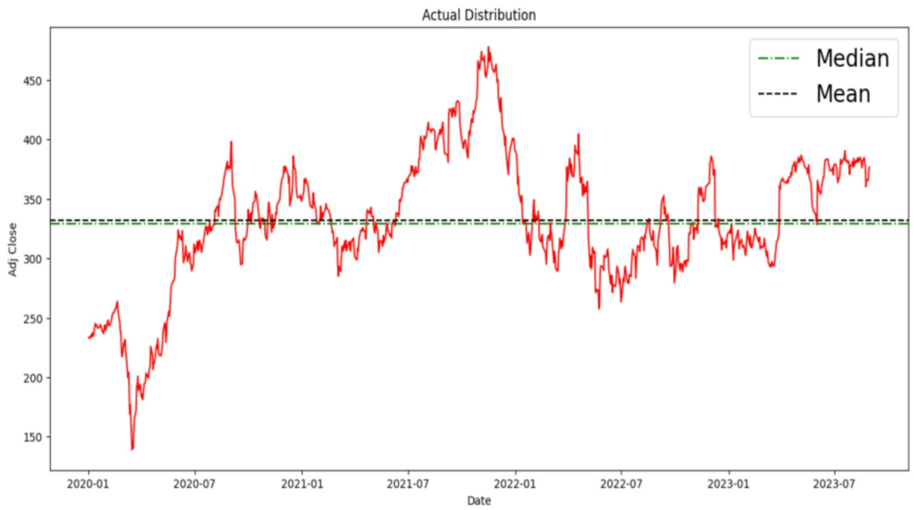
- Mean (Average) : 332.04
- Standard Deviation (Measure of Spread) : 55.58
- Skewness (Measures asymmetry of Data Distribution) : -0.35
- Kurtosis (Spikiness): 0.75

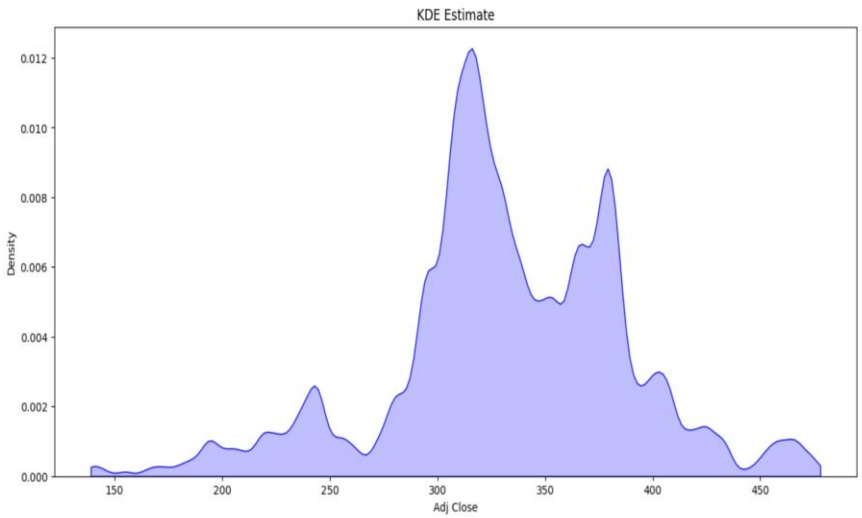
Kernel Density Estimation

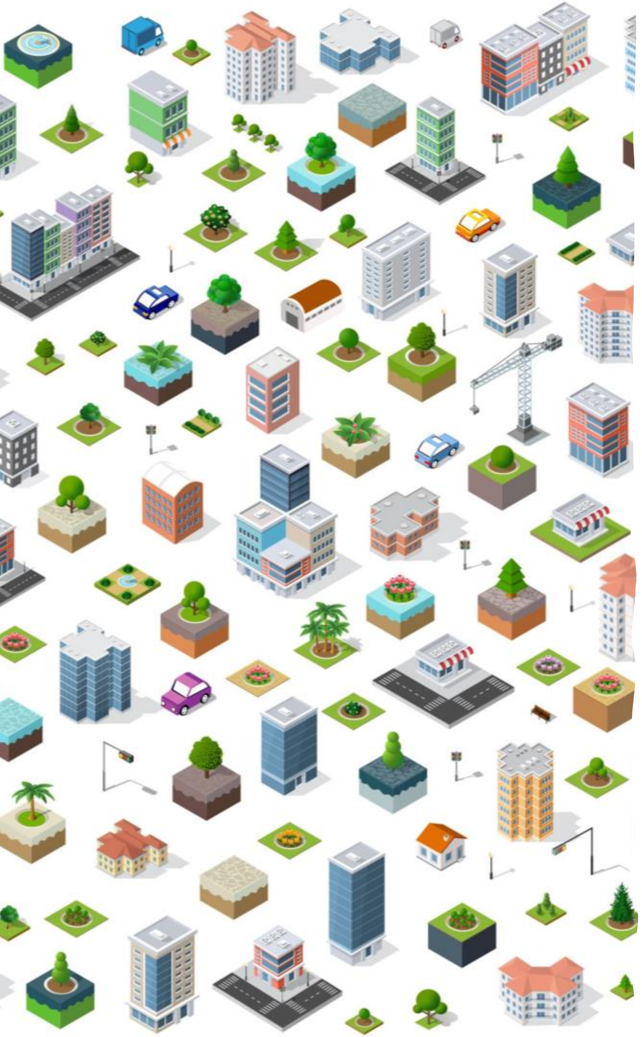
Kernel density estimation (KDE) is the application of kernel smoothing for probability density estimation, i.e., a nonparametric method to estimate the probability density function of a random variable based on kernels as weights. KDE answers a fundamental data smoothing problem where inferences about the population are made, based on a finite data sample.

LULULEMON– Actual Distribution vs KDE Estimate

- Multimodal distribution for Lululemon, The peak price in the period is close to $450, and lowest price is around $150.
- Due to the multimodal nature of the KDE estimate, the bandwidth parameter is lowered to capture different peaks.
- The mean value of the stock price during this period is around $335 and the median value is around $330.the







Model feature selection

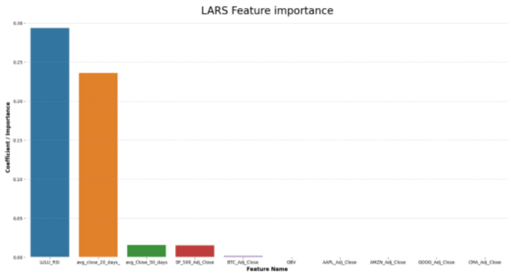
- To choose relevant features with high importance we ran various models like RandomForest, LARS, Ridge Regression, Decision Trees, and XGBoost.

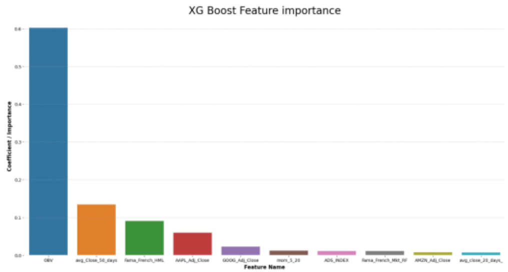
- While some of the features were consistently on the top for LULULEMON (like OBB factors), other features came up as important only in certain models

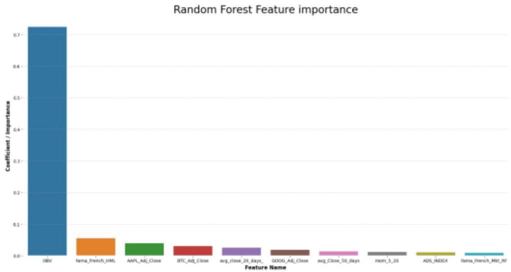
Features used in models:

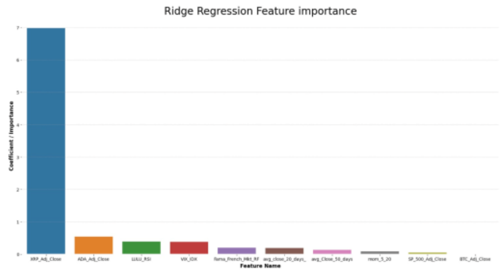
Feature Importance plots

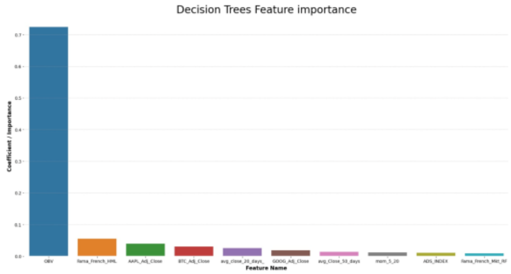
- While running different machine learning models, Momentum factors regularly stood out at the top.
- We decided to set a threshold and pick the top 3 feature from every model for a total of 8 features (set union)

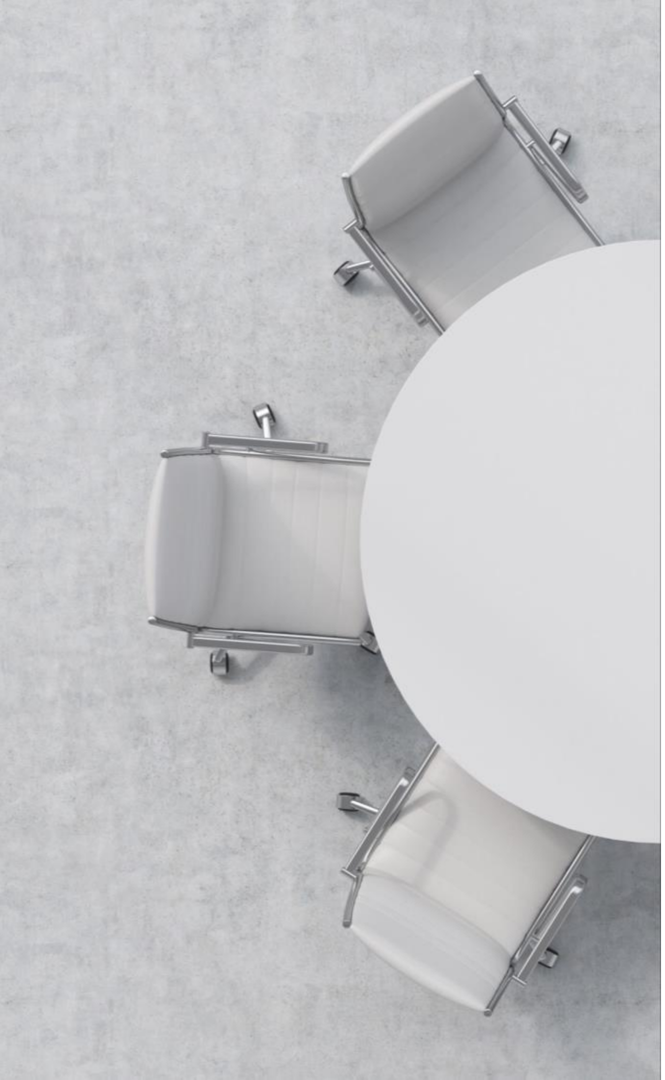












The 8 final features for training our models:

Applying Statistical Models on LULULEMON Stock

Ridge Regression Prediction

```
[24] # Define the model and fit the data
     ridge = Ridge(alpha=0.1)
     ridge.fit(X_train, y_train)

     # Make predictions on the test set
     y_pred = ridge.predict(X_test)

     # Evaluate the model performance using mean squared error
     mse = mean_squared_error(y_test, y_pred)
     rr_rmse = np.sqrt(mse)
     print('RMSE value is:', rr_rmse)
```

```
RMSE value is: 13.974289889082193
```

Random Forest Regression Prediction

[23]

```
# Define the model and fit the data
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_model.predict(X_test)

# Evaluate the model performance using mean squared error
mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(mse)
print('RMSE value is:', rf_rmse)
```

```
RMSE value is: 8.591772241036615
```

Gradient Boosting(XGBoost) Regression Prediction

```
# Fit an XGBoost model to the data
xgb_model = xgb.XGBRegressor(random_state=0, n_estimators=100).fit(X_train, y_train)

# Make predictions on the test set
y_pred = xgb_model.predict(X_test)

# Evaluate the model performance using mean squared error
mse = mean_squared_error(y_test, y_pred)
xgb_rmse = np.sqrt(mse)
print('RMSE value is:', xgb_rmse)
```

```
RMSE value is: 7.54052912599941
```

Lasso Regression Prediction:

```
[26] # Fit a LassoLarsCV model to the data
     lars = LassoLarsCV(cv=5).fit(X_train, y_train)

     # Make predictions on the test set
     y_pred = lars.predict(X_test)

     # Evaluate the model performance using mean squared error
     mse = mean_squared_error(y_test, y_pred)
     lasso_rmse = np.sqrt(mse)
     print('RMSE value is:', lasso_rmse)
```

```
RMSE value is: 13.94561792598584
```

Decision Tree Regression Prediction:

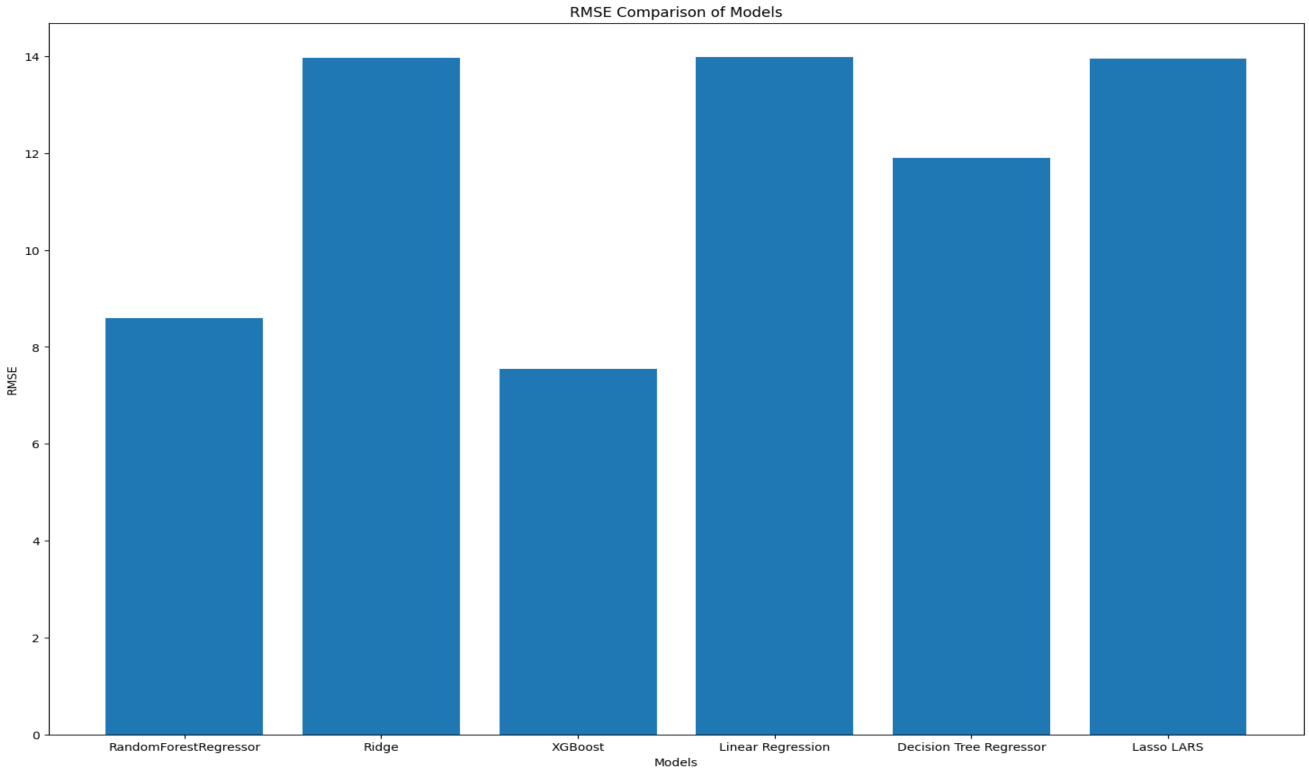
```
[27] # Fit a decision tree model to the data
     tree = DecisionTreeRegressor(random_state=0)
     tree.fit(X_train, y_train)

     # Make predictions on the test set
     y_pred = tree.predict(X_test)

     # Evaluate the model performance using mean squared error
     mse = mean_squared_error(y_test, y_pred)
     dt_rmse = np.sqrt(mse)
     print('RMSE value is:', dt_rmse)
```

```
RMSE value is: 11.909796425438934
```

Linear Regression Prediction:

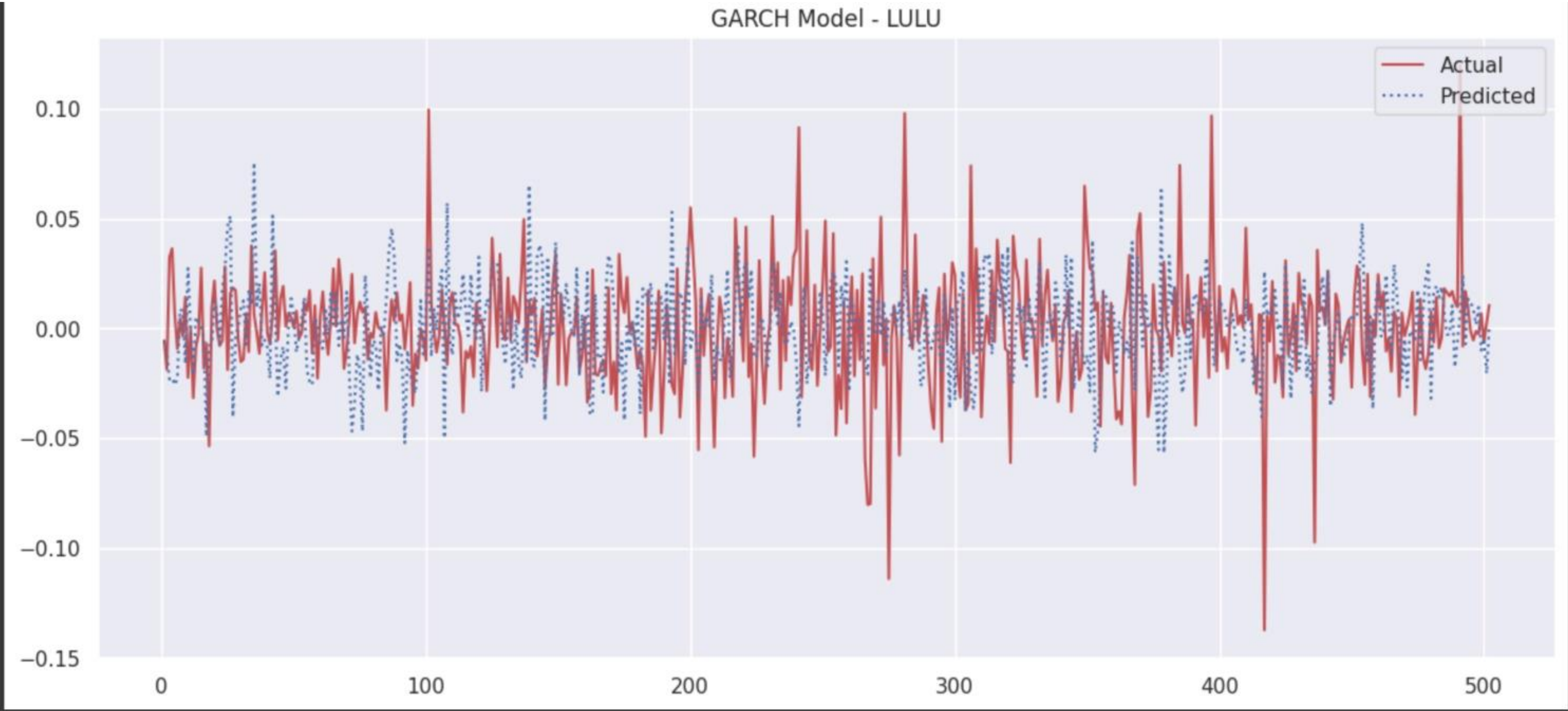
```
[28] from sklearn.linear_model import LinearRegression

     # Initialize the model
     linear_model = LinearRegression()

     # Train the model
     linear_model.fit(X_train, y_train)

     # Make predictions on the test set
     y_pred = linear_model.predict(X_test)

     # Evaluate the model performance using mean squared error
     mse = mean_squared_error(y_test, y_pred)
     lr_rmse = np.sqrt(mse)
     print('RMSE value is:', lr_rmse)

     RMSE value is: 13.988789429474924
```

Comparative Graph RMSE of all the models:



Benchmark: GARCH

- Generalized Autoregressive Conditional Heteroskedasticity
- It is a statistical model used to analyze the volatility of financial time series data

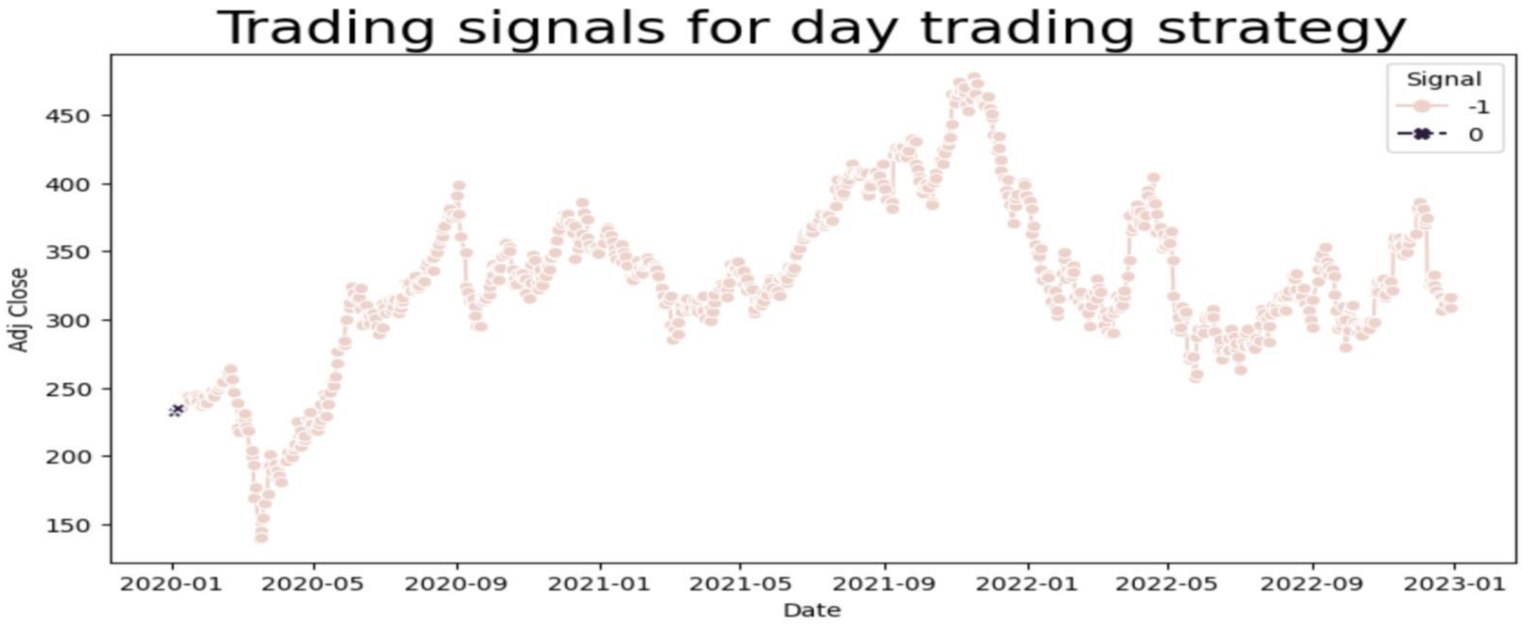



```
[9] RMSE = np.sqrt(np.mean((Y_GARCH - Y)**2))
    print('RMSE values is:', RMSE)
```

```
RMSE values is: 0.03355220861284629
```

Benchmark: Kalman Filter

- A mathematical algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, to estimate the underlying state of a system.
- Combine predictions based on the system model with measurements to obtain an optimal estimate of the system state.
- Two steps: prediction step & and update step



Thank You