Predicting Uniqlo Stock Prices with Machine Learning

Objective Statement

The goal of this project is to produce a machine learning model that can predict the closing price of Uniqlo stock. While stock price prediction is always an important task, now more than ever, in such turbulent times, when financial wellness is so difficult to attain, it is **essential** that inidividuals use data-driven methodologies like this to make investments that will maintain their financial stability.

Data Loading and Preparation

In [68]:

```
#import necessary libraries
import pandas as pd
import numpy as np
```

In [69]:

```
#read in data
uniqlo_stock = pd.read_csv("uniqlo_data.csv").dropna()
uniqlo_stock.head()
```

Out[69]:

	Date	Open	High	Low	Close	Volume	Stock Trading
0	2016-12-30	42120	42330	41700	41830	610000	25628028000
1	2016-12-29	43000	43220	42540	42660	448400	19188227000
2	2016-12-28	43940	43970	43270	43270	339900	14780670000
3	2016-12-27	43140	43700	43140	43620	400100	17427993000
4	2016-12-26	43310	43660	43090	43340	358200	15547803000

In [70]:

```
uniqlo_stock.drop(['Date'], axis=1, inplace=True)

uniqlo_stock.head()
```

Out[70]:

	Open	High	Low	Close	Volume	Stock Trading
0	42120	42330	41700	41830	610000	25628028000
1	43000	43220	42540	42660	448400	19188227000
2	43940	43970	43270	43270	339900	14780670000
3	43140	43700	43140	43620	400100	17427993000
4	43310	43660	43090	43340	358200	15547803000

For simplicity of model training, date will not be considered in this analysis.

In [71]:

```
uniqlo_stock.shape
```

Out[71]:

```
(1226, 6)
```

The next step in our data preparation process is to standardize the data so that there is no bias introduced into our results by differences in the scale of each variable.

In [72]:

```
Y = uniqlo_stock.loc[:, 'Close']
```

In [73]:

```
#standardizing X

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

high = uniqlo_stock.loc[:,'High'].values

low = uniqlo_stock.loc[:,'Low'].values

open_price = uniqlo_stock.loc[:,'Open'].values

volume = uniqlo_stock.loc[:,'Volume'].values

trading = uniqlo_stock.loc[:,'Stock Trading'].values


std_high = scaler.fit_transform(high.reshape((-1,1)))

std_low = scaler.fit_transform(low.reshape((-1,1)))

std_open = scaler.fit_transform(open_price.reshape((-1,1)))

std_volume = scaler.fit_transform(volume.reshape((-1,1)))

std_trading = scaler.fit_transform(trading.reshape((-1,1)))


std_high = pd.DataFrame(std_high, columns=['std_high'])

std_low = pd.DataFrame(std_low, columns=['st_low'])

std_open = pd.DataFrame(std_open, columns=['std_open'])

std_volume = pd.DataFrame(std_volume, columns=['std_volume'])

std_trading = pd.DataFrame(std_trading, columns=['std_trading'])




##creating final X

X = pd.concat([std_high, std_low, std_open, std_volume, std_trading], axis=1, join='inner'
)

X.head()
```

Out[73]:

	std_high	st_low	std_open	std_volume	std_trading
0	0.745638	0.781205	0.773944	-0.284260	0.079879
1	0.827051	0.859773	0.855357	-0.675024	-0.342215
2	0.895658	0.928053	0.942322	-0.937387	-0.631107
3	0.870960	0.915894	0.868310	-0.791818	-0.457589
4	0.867301	0.911217	0.884037	-0.893136	-0.580826

In [74]:

```
X.shape
```

Out[74]:

```
(1226, 5)
```

Model Creation and Testing

Now that the data are prepared for analysis, let us create several regression models and test their out-of-sample fit on our prepared data.

In [75]:

```
#importing linear regression model
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import make_scorer
```

In [76]:

```
#create and test regression model

stock_lr = LinearRegression()

from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor

stock_rf = RandomForestRegressor()
stock_dt = DecisionTreeRegressor()



lr_score = np.mean(cross_val_score(stock_lr, X, Y, scoring=make_scorer(mean_squared_error
)))

rf_score = np.mean(cross_val_score(stock_rf, X, Y, scoring=make_scorer(mean_squared_error
)))

dt_score = np.mean(cross_val_score(stock_dt, X, Y, scoring=make_scorer(mean_squared_error
)))



print(f'Linear Regression Average MSE: {lr_score}')
lr_loss = lr_score ** 0.5
print(f'The linear regression model, on average, predicts closing price within {lr_loss:.5
} units (Currency details are not available in the dataset.).')


print(f'Random Forest Average MSE: {rf_score}')
rf_loss = rf_score ** 0.5
print(f'The random forest model, on average, predicts closing price within {rf_loss:.6} un
its (Currency details are not available in the dataset.).')


print(f'Decision Tree Average MSE: {dt_score}')
dt_loss = dt_score ** 0.5
print(f'The decision tree model, on average, predicts closing price within {dt_loss:.6} un
its (Currency details are not available in the dataset.).')
```

```
Linear Regression Average MSE: 73449.84305212226
The linear regression model, on average, predicts closing price within 271.02
units (Currency details are not available in the dataset.).
Random Forest Average MSE: 10282628.128559563
The random forest model, on average, predicts closing price within 3206.65 un
its (Currency details are not available in the dataset.).
Decision Tree Average MSE: 9759537.840053095
The decision tree model, on average, predicts closing price within 3124.03 un
its (Currency details are not available in the dataset.).
```

Note: Model testing is done here by finding mean squared error via five-fold cross validation. This allows for better model evaluation than a metric like R-squared, which is a less direct, and thus less useful, measure of a model's predictive ability.

As we can see from the results above, the linear regression algorithm is by far the best model with which to predict Uniqlo closing stock price. Let us now see if we can improve this model with hyperparameter tuning.

Hyperparameter Tuning

In [77]:

```
#see model parameters
stock_lr.get_params()
```

Out[77]:

```
{'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'normalize': False}
```

In [78]:

```
#trying diffferent paramater combinations
from sklearn.model_selection import GridSearchCV


param_grid = {
    'copy_X': [True, False],
    'fit_intercept': [True, False],
    'n_jobs': [1,-1],
    'normalize': [True, False]
}

grid_search = GridSearchCV(estimator=stock_lr, param_grid = param_grid)

grid_search.fit(X, Y)
```

Out[78]:

```
GridSearchCV(estimator=LinearRegression(),
             param_grid={'copy_X': [True, False],
                         'fit_intercept': [True, False], 'n_jobs': [1, -1],
                         'normalize': [True, False]})
```

In [79]:

```
grid_search.best_params_
```

Out[79]:

```
{'copy_X': True, 'fit_intercept': True, 'n_jobs': 1, 'normalize': True}
```

In [80]:

```
#tuned model

tuned_lr = LinearRegression(normalize=True)

tuned_lr.fit(X, Y)

tuned_lr_score = np.mean(cross_val_score(tuned_lr, X, Y, scoring=make_scorer(mean_squared_
error)))

print(f'Tuned Regression Average MSE: {tuned_lr_score}')
tuned_lr_loss = tuned_lr_score ** 0.5
print(f'The linear regression model, on average, predicts closing price within {tuned_lr_l
oss:.5} units.')
```

```
Tuned Regression Average MSE: 73449.84305212248
The linear regression model, on average, predicts closing price within 271.02
units.
```

In [81]:

```
#finding max percent error of model

#finding lowest price in dataset
lowest_closing_price = np.min(Y)

#finding percent of this number to get conservative estimate of model error

percent_error = (tuned_lr_loss / lowest_closing_price) * 100

print(f'The average margin of error for the tuned regression model is {percent_error:.3}%
 of the lowest closing price in the dataset.')
```

```
The average margin of error for the tuned regression model is 1.98% of the lo
west closing price in the dataset.
```

Conclusion

The hyperparameter tuning step produced a model that is virtually identical in out-of-sample fit to the default linear regression model. This model can predict closing price, on average, within 1.98% of the lowest closing price available in the training data.

While the results of this project are quite impressive, this project could still be modified to produce more useful results. In particular, the dataset used here was last updated four years ago, so a more applicable model would be attained with more recent data.

Finally, given the utility of linear regression in predicting the Uniqlo closing stock price, the results of this analysis suggest that linear regression should be considered for future closing stock price prediction projects.

References:

Previous Coursework

Classmate Consultation

Dataset Source (https://www.kaggle.com/daiearth22/uniqlo-fastretailing-stock-price-prediction)

Making Heatmaps (https://matplotlib.org/stable/gallery/images_contours_and_fields/image_annotated_heatmap.html)

Multicolinearity Discussion (https://stattrek.com/multiple-regression/multicollinearity.aspx#:~:text=There%20are%20two%20popular%20ways,factor%20for%20each%20inc

More Multicolinearity Discussion (https://statisticsbyjim.com/regression/multicollinearity-in-regression-analysis/#:~:text=Fortunately%2C%20there%20is%20a%20very,VIF%20for%20each%20independent%20variable

Links in Markdown (https://markdownmonster.west-wind.com/docs/_4xs10gaui.htm)

Computing VIF with Python (https://www.geeksforgeeks.org/detecting-multicollinearity-with-vif-python/)

Copying a DataFrame (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.copy.html)

One Hot Encoder Documentation (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html)

Pandas DropNA (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html)

Pandas One-Hot Encoding (https://stackabuse.com/one-hot-encoding-in-python-with-pandas-and-scikit-learn/)

Discussion of Dropping a Dummy (https://datascience.stackexchange.com/questions/27957/why-do-we-need-to-discard-one-dummy-variable)

Skewed Data/Data Preparation (https://towardsdatascience.com/top-3-methods-for-handling-skewed-data-1334e0debf45)

Linear Regression Documentation (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)

LASSO Regression Documentation (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html)

Assuming Normality of Y in Linear Regression (https://thestatsgeek.com/2013/08/07/assumptions-for-linear-regression/)

Numpy Histogram Documenation (https://numpy.org/doc/stable/reference/generated/numpy.histogram.html)

Seaborn Color Palattes (https://seaborn.pydata.org/tutorial/color_palettes.html)

Sklearn Standard Scaler (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html)

Sklearn K-fold (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html)

Sklearn MSE (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html)

Sklearn Cross-Validation (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html)

Sklearn Linear Regression (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)

Markdown Syntax Reference (https://www.math.mcgill.ca/yyang/regression/RMarkdown/example.html)

Make Scorer Sklearn (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.make_scorer.html)

Reshaping Pandas Series (https://www.geeksforgeeks.org/how-to-reshape-pandas-series/)

K-Fold Cross Validation Discussion (https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f)

More K-Fold Cross Validation Disucssion (https://machinelearningmastery.com/k-fold-cross-validation/)

Standarization for Regression (https://medium.com/@kylecaron/introduction-to-linear-regression-part-2-standardization-and-regression-diagnostics-a15cb27944b1)

Decision Tree Regressor (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html)

Random Forest Regressor (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html)

Random Forest Regression Discussion (https://neptune.ai/blog/random-forest-regression-when-does-it-fail-and-why)

Standardization Discussion (https://statisticsbyjim.com/regression/standardize-variables-regression/)

Standardization of Y (https://stats.stackexchange.com/questions/278566/if-you-standardize-x-must-you-always-standardize-y)

Example Regression with this Dataset (https://www.kaggle.com/fillerink/basic-linear-regression-with-housing-prices)

Seaborn Heatmap Tips (https://heartbeat.fritz.ai/seaborn-heatmaps-13-ways-to-customize-correlation-matrix-visualizations-f1c49c816f07)

Definition of MSE (https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/mean-squared-error/#:~:text=The%20mean%20squared%20error%20tells,to%20remove%20any%20negative%20signs.)

Pandas Set Index (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.index.html?highlight=index#pandas.DataFrame.index)

Pandas Index (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.index.html?highlight=index#pandas.DataFrame.index)

Sklearn Cross Validation General Discussion (https://scikit-learn.org/stable/modules/cross_validation.html)

Stock Market Prediction Reference (https://www.kaggle.com/getting-started/94106)

Reference to Data Used (https://www.kaggle.com/learn-forum/167685)

Sample Project with Dataset (https://www.kaggle.com/mehmetkasap/stock-price-prediction-lstm)

Stock Market Data Definitions (https://analyzingalpha.com/open-high-low-close-stocks#:~:text=In%20stock%20trading%2C%20the%20high,total%20amount%20of%20trading%20activity.)

Linear Regression Guide Sklearn (https://realpython.com/linear-regression-in-python/)

Linear Regression Hyperparameter Tuning (https://stackoverflow.com/questions/60454618/is-it-possible-to-tune-the-linear-regression-hyperparameter-in-sklearn)

Grid Search Documentation (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)

Grid Search Tutorial (https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/)