Data Science Capstone project: Clever Money

Felipe Eduardo Trujillo Preuss

August 2021

Outline

- Executive Summary
- Introduction
- Methodology: Data collection, Data Wrangling, EDA
- Predictive Analysis
- Model Iteration
- Conclusion

Executive Summary

This project is about predicting stock market returns with a different angle, applying XGBoost algorithm & Long Short Term Memory (LSTM) over a S&P 500 Stock data to create a model and predict the return direction and future values. Also iterate over the model with Principal component Analysis to extract important features and increase the accuracy.

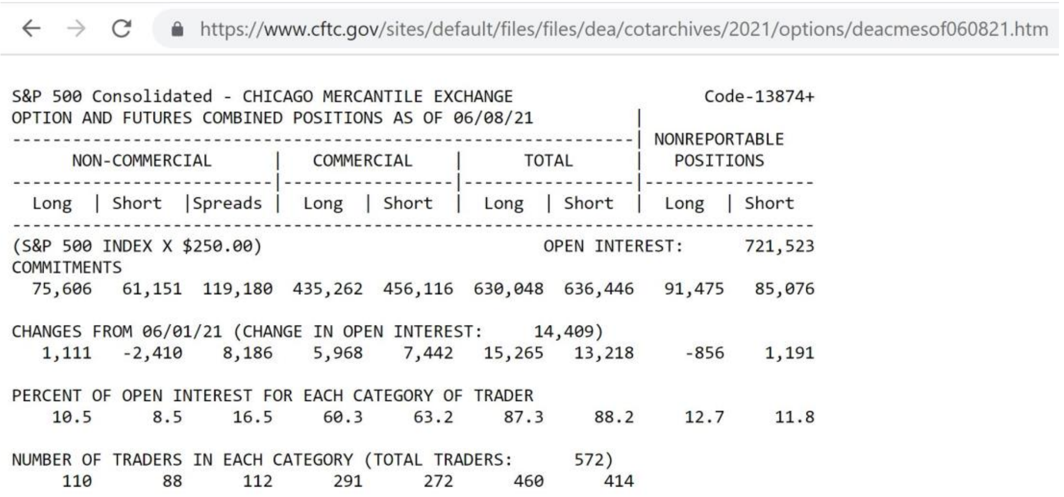
Introduction

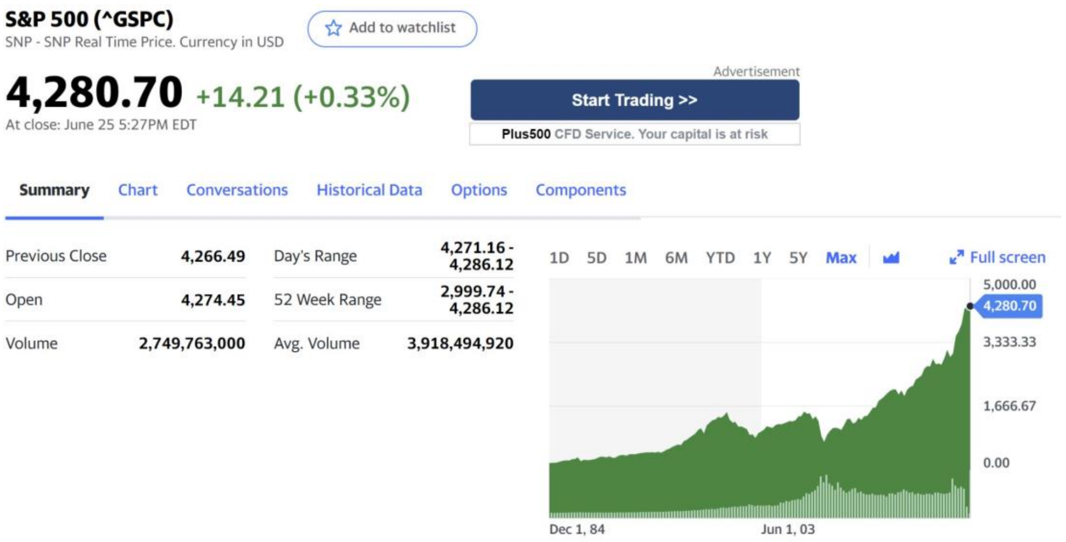
- Investing in the STOCK or CRYPTO market is a coin flip - “will the market go up or down this week?”
- Institutional investors (“CLEVER MONEY”) are the big hedge funds and investment banks who are able to move markets considerably. Based on fundamental analysis (macro-economic landscape, political climate) & long-term horizon for trades (years)
- Retail Investors (“DUMB MONEY”) are small players who are often on the losing side of a trade. Based on technical analysis (short term indicators, chart patterns) & Short term horizon for trades (days)

- Can we predict future stock market returns by analyzing investor positioning alone?

Data collection

- S&P 500 data
  - Data: Closing prices every Tuesday from June 2010 to June 2021
  - Source: Yahoo Finance
  - Format: pandas dataframe
- Commitment of traders reports
  - Data: Long/Short Commitments for S&P 500 from June 2010 to June 2021
  - Reports get released every Tuesday
  - Source: Commodity Futures Trading Commission (CFTC) website
  - Format: static html (python web-scraping)



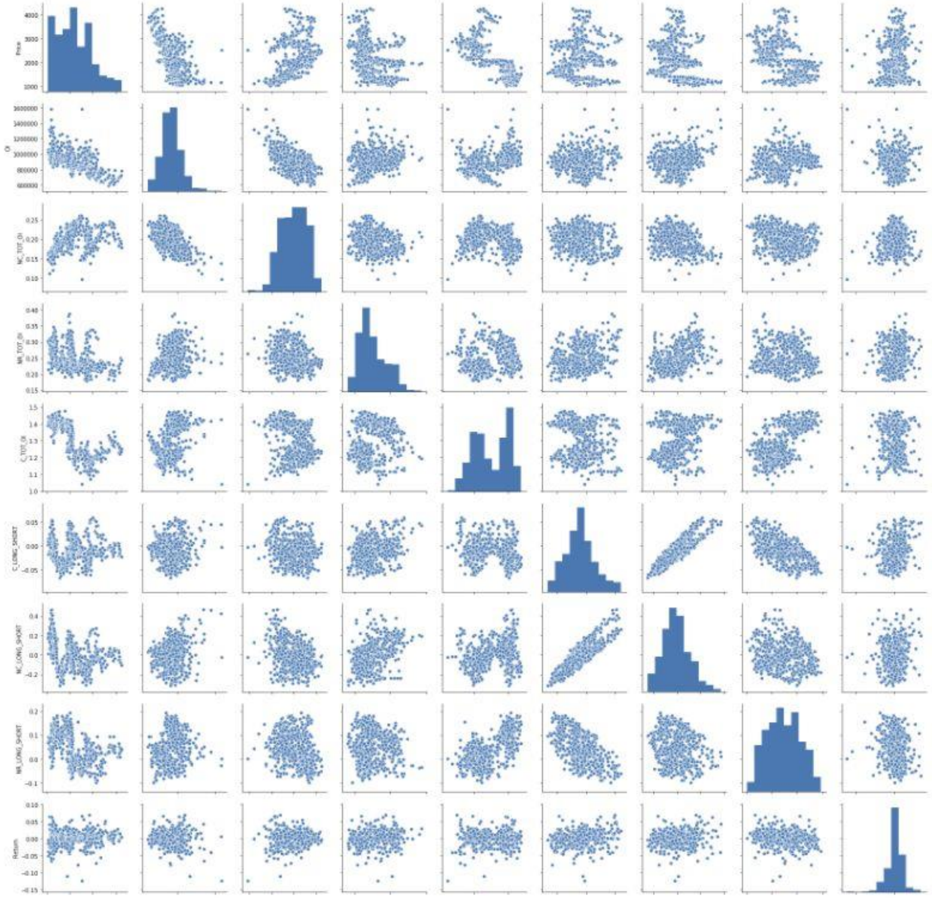


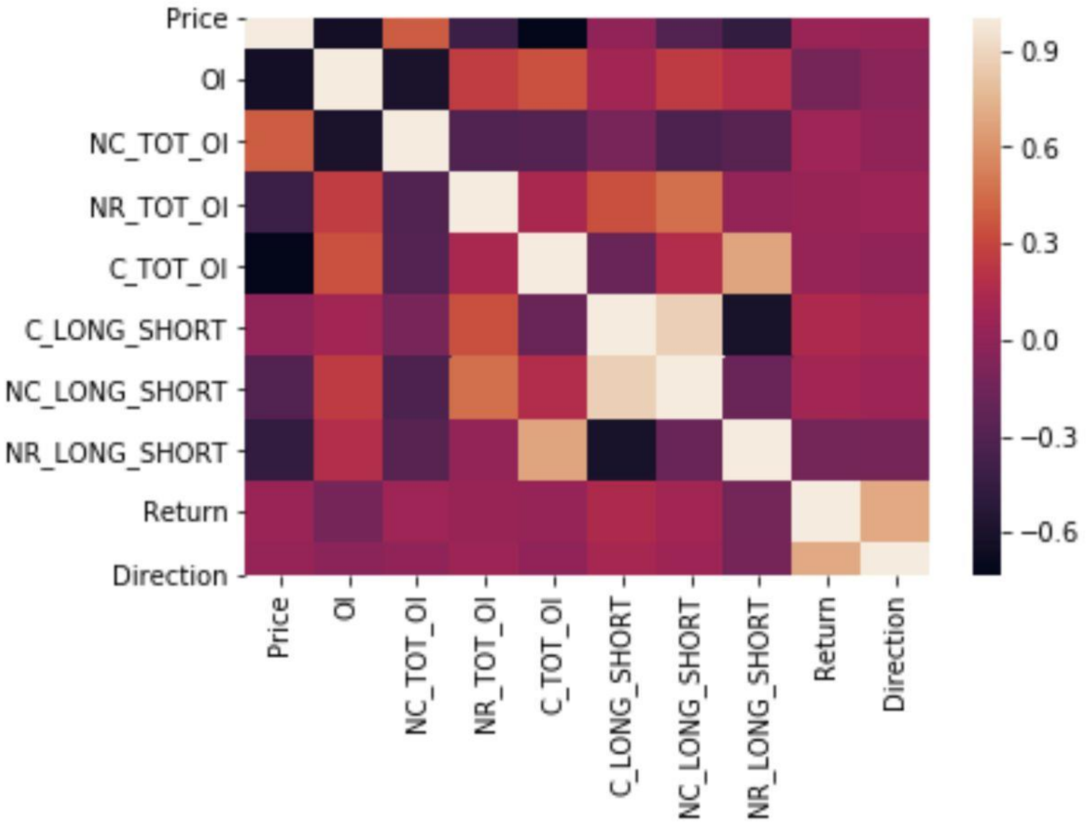
Data wrangling – Quality Adjustment

- There were some public holidays (e.g. Christmas, 4th July) on Tuesday in the USA which meant S&P 500 prices came back as null.
- Commitment of traders reports would also be preponed on these weeks.
- To correct for this, S&P 500 prices on the previous days were manually inserted into the dataframe. I.e. If 4th of a July was a Tuesday, the Monday 3rd of July closing price would be considered instead
- Commitment of traders reports were also manually inputted on these exception dates.

```
for date in dates:
    date_str = date.strftime('%Y/%m/%d')
    price = 0
    try:
        price=sandpclosing[date]
    except:
        print('Price Exception for '+date_str)
    url = "https://www.cftc.gov/sites/default/files/files/dea/c
    commitments = [0,0,0,0,0,0,0,0,0,0]
    try:
        page = urlopen(url)
    except:
        print('URL Exception for '+date_str)
    else:
        html = page.read().decode("utf-8")
```

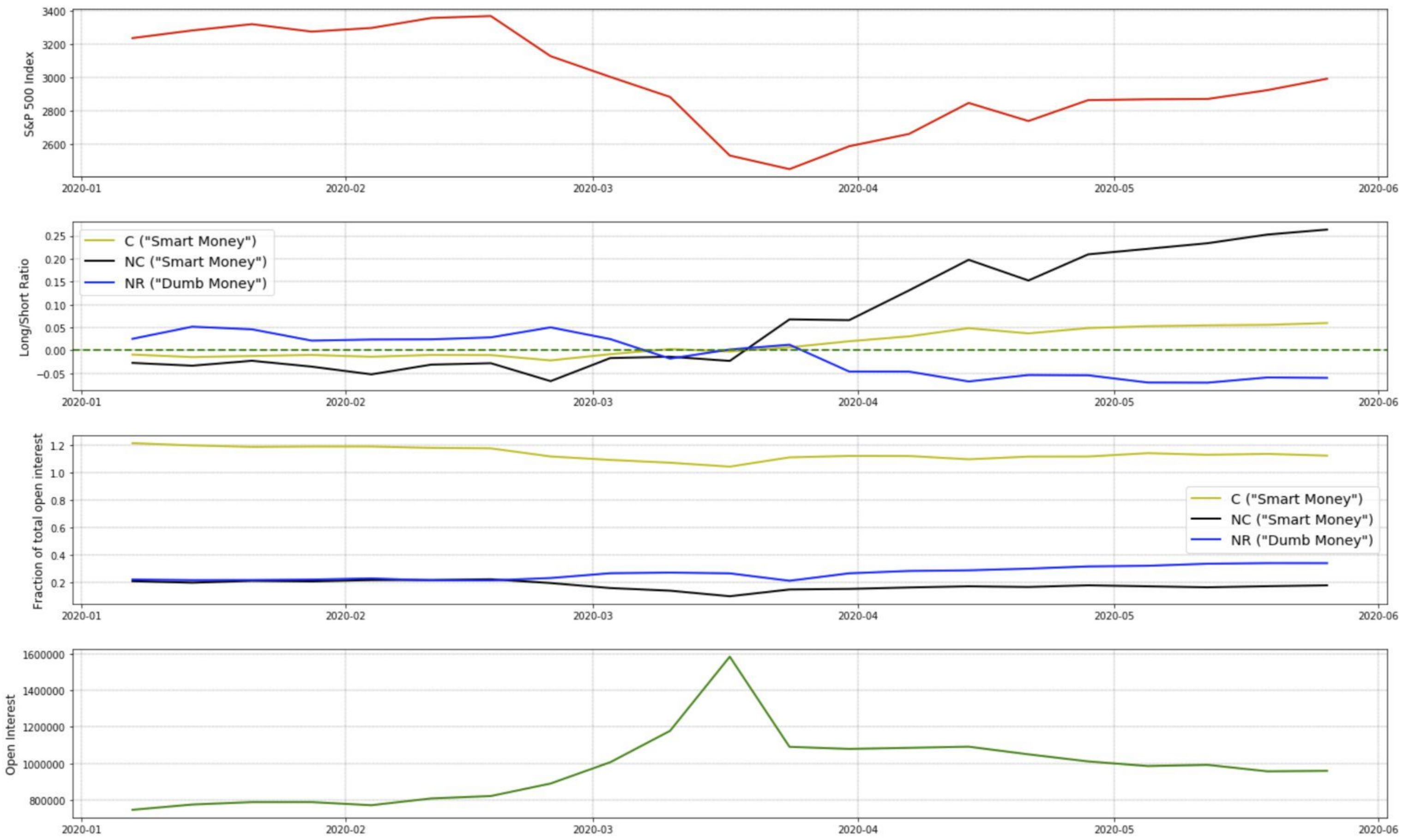
EDA with data visualization - exploration





EDA with data visualization - exploration

- Evolution analyzing the covid crash



Data Normalization

- Due to the skew and varying magnitudes in some of the distributions seen through data exploration, features and output were normalized
- The number of outliers in the dataset is minimal, so we expect min-max normalization to yield good results.

```
# - - - - - - - - - - -
# FEATURE SCALING
# - - - - - - - - - - -

# split data into X (features) and Y (output)
X = df[['OI','NC_TOT_OI','NR_TOT_OI','C_TOT_OI','NC_LONG_SHORT','NR_LONG_SHORT','C_LONG_SHORT']]
Y = df['Return']

# normalize X data
X_normalizer = MinMaxScaler().fit(X)
X_scaled = X_normalizer.transform(X)

# normalize Y data
Y_normalizer = MinMaxScaler().fit(Y.values.reshape(-1,1))
Y_scaled = Y_normalizer.transform(Y.values.reshape(-1,1))
```

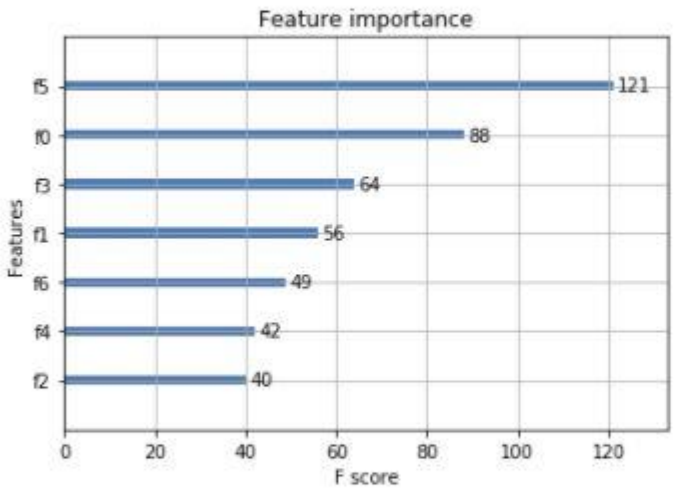
Predictive analysis (Direction)

- Predict direction of this week’s return using gradient boosted trees (XGBoost algorithm)
- Model justification:
  - Gradient boosted trees have the benefit of starting with a poor model and iterating multiple times.
  - XGBoost is a specific implementation which is designed to be more performant and less prone to overfitting
  - One viable alternative considered was a support vector machine (SVM)
- Model performance indicator:
  - Accuracy (percentage of correct “up” or “down” weeks)
  - we simply want to know if the model guesses the right direction, we are not necessarily concerned with true positives vs false positives

Predictive analysis (Direction)

- XGBoost relies on several hyperparameters, of which 3 are tuned using GridSearchCV provided by scikit-learn.
- The optimal XGBoost classifier is then used to predict the test data. Note the test data is arbitrarily chosen as April 2019 until June 2021.
- Training and test accuracy are around 60%
- Feature importance suggests the open interest (‘OI’) and Commercial long-short ratio (‘C_LONG_SHORT’) are the two most important features in determining the direction of return

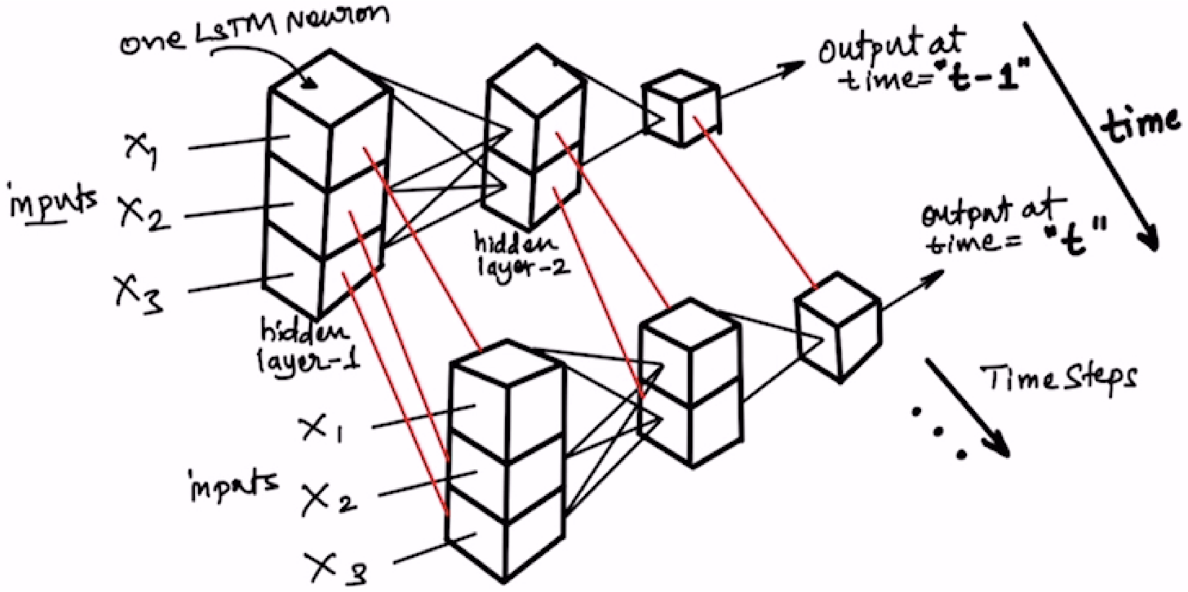
```
Training accuracy: 0.6109
Test accuracy: 0.6000
```




```
params = {
    "learning_rate" : [0.1, 0.2, 0.3],
    "max_depth" : [3, 5, 7],
    "min_child_weight" : [1, 3, 5]
}

cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)

clf = xgboost.XGBClassifier(use_label_encoder=False, objective = 'binary:logistic',eval_metric='error')
grid_search = GridSearchCV(clf, param_grid=params, scoring='accuracy', cv=cv)
grid_search.fit(X_train, Y_train)

clf = grid_search.best_estimator_
score = cross_val_score(clf, X_train, Y_train, cv=cv)

y_pred = clf.predict(X_test)
print('Training accuracy: {0:0.4f}'. format(score.mean()))
print('Test accuracy: {0:0.4f}'. format(accuracy_score(Y_test, y_pred)))

xgboost.plot_importance(clf)
plt.figure(figsize = (16, 12))
plt.show()
```

Predictive analysis (Future)

- Long Short Term Memory (LSTM) is an artificial recurrent neural network (RNN)
- STM is the most appropriate RNN for time-series analysis
- Model performance indicator
  - Mean squared error (MSE)
  - Mean absolute error (MAE)



Predictive analysis (Future)

- Data is first split up into test and train sets. The training set covers June 2010 to April 2019, and test set covers April 2019 until June 2021.
- We are basing next week's return off 5 previous week's data
- The optimizer and learning rate was chosen after trial and error
- Model fitting uses 20% validation split and we set shuffle=False since we need to maintain ordering of our data.

```
# - - - - - - - - - - - - - - - - - - - - -
# SPLIT DATA INTO TEST/TRAIN FOR LSTM
# - - - - - - - - - - - - - - - - - - - - -

# use n_past values to determine next week's return
n_future = 1
n_past = 5

X_train, Y_train, X_test, Y_test = [],[],[],[]
for i in range(n_past, len(X_scaled)-n_future+1):
    if i < test_start:
        X_train.append(X_scaled[i-n_past:i,:])
        Y_train.append(Y_scaled[i+n_future-1:i+n_future])
    else:
        X_test.append(X_scaled[i-n_past:i,:])
        Y_test.append(Y_scaled[i+n_future-1:i+n_future])

X_train, Y_train, X_test, Y_test = np.array(X_train), np.array(Y_train), np.array(X_test), np.array(Y_test)
```

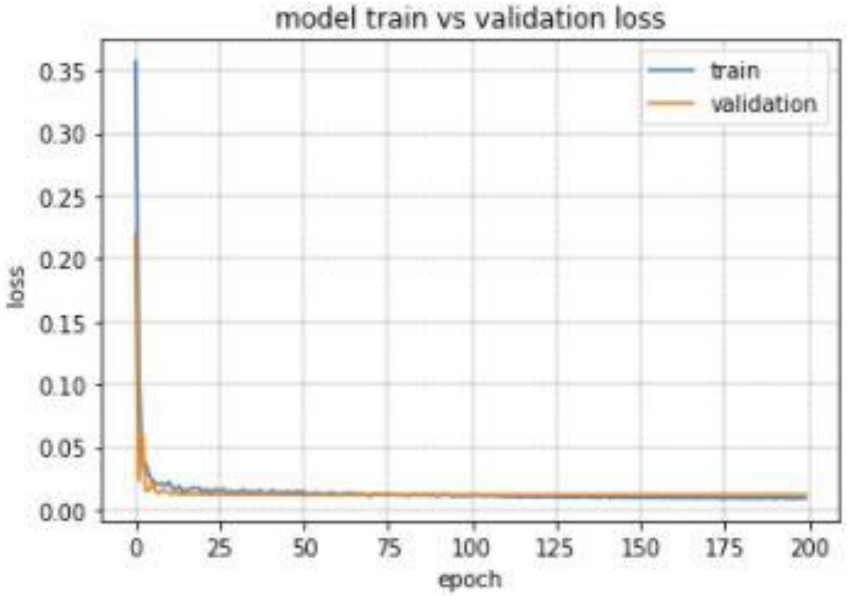
```
# - - - - - - - - - - - - - - - -
# CREATE LSTM
# - - - - - - - - - - - - - - - -

np.random.seed(1234)
tf.random.set_seed(1234)
model = Sequential()
model.add(LSTM(64, activation='relu', input_shape=(X_train.shape[1],X_train.shape[2]), return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(32, activation='relu', return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(16, activation='relu', return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(1))

model.compile(tf.keras.optimizers.Adam(learning_rate=5e-3), loss='mse')
model.summary()

history = model.fit(X_train,Y_train,validation_split=0.2,epochs=200,batch_size=64,verbose=1,shuffle=False)
```

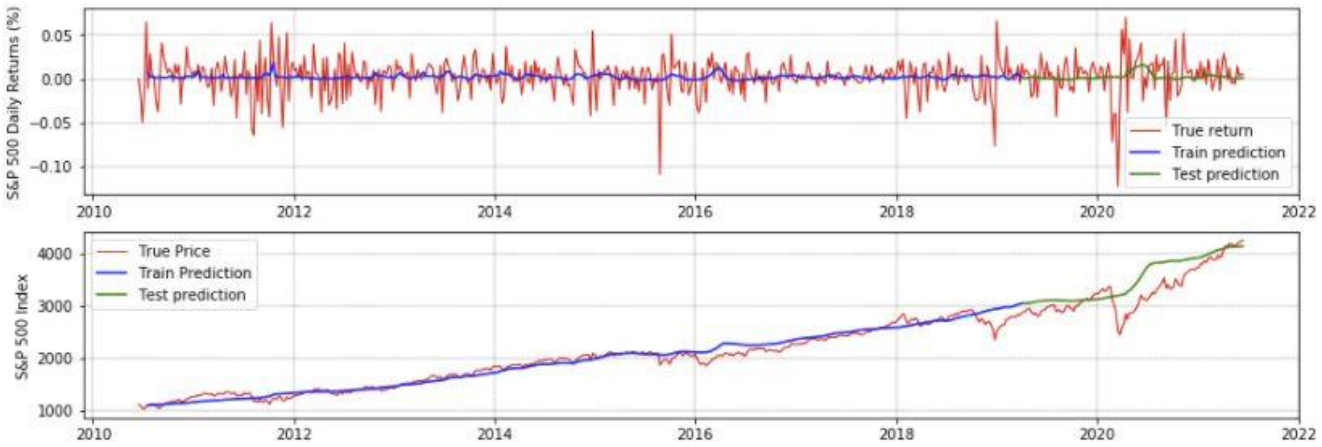
Predictive analysis (Future)

- Training performance:
  - MSE: 0.0003
  - Accuracy: 60%

- Test performance:
  - MSE: 0.0006
  - Accuracy: 67%

- Predicted returns are smoothed relative to true returns



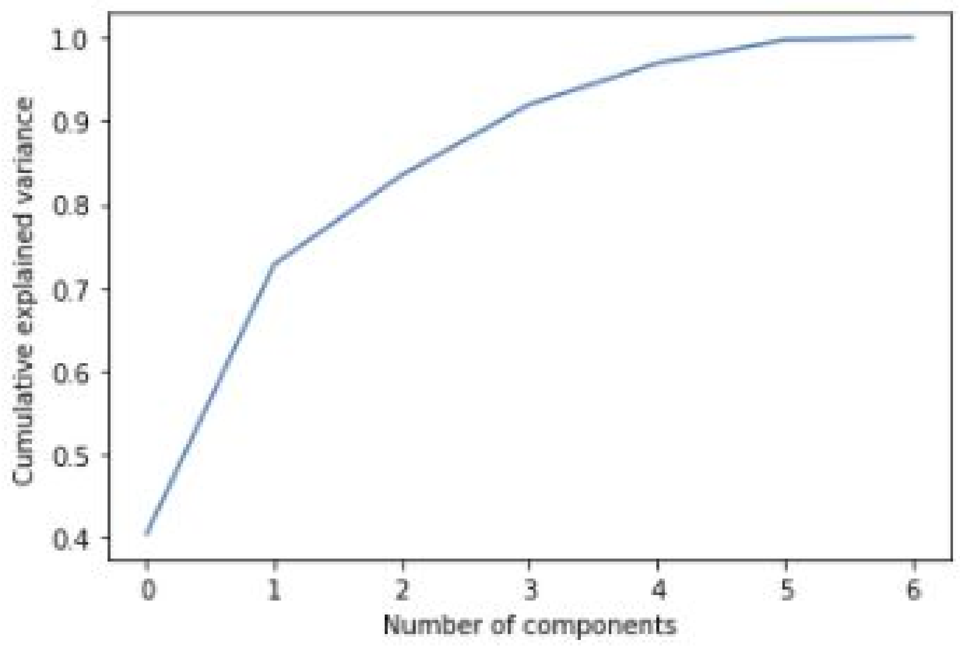


Model Iteration

- We shall use Principal Component Analysis (PCA) to extract the most important features which best determine return.

- According to PCA analysis, 4 features describe 92% of the cumulative variance.

- This suggests we can reduce our features from 7 to 4.



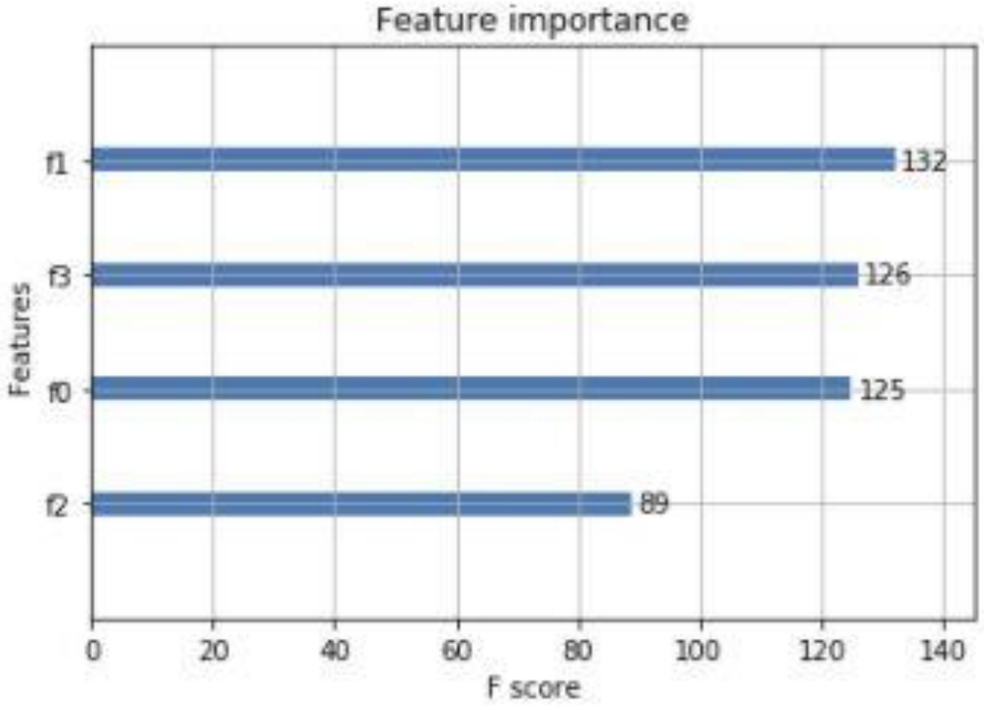

```
# - - - - - - - - - - - - - - -
# PCA COMPONENT ANALYSIS
# - - - - - - - - - - - - - - -
pca = PCA(n_components=7)
pca.fit(X_scaled)

plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of components')
plt.ylabel('Cumulative explained variance')
```

Model Iteration – Effect

- Training accuracy increased from 61% to 68%
- Test accuracy increased from 60% to 67%
- relative feature importance is most even compared

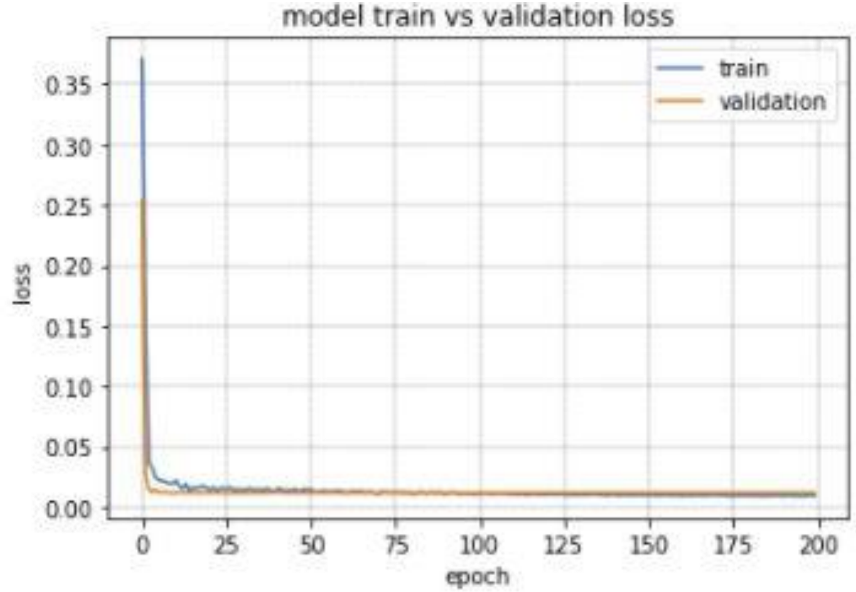
```
pca = PCA(n_components=4)
X_scaled = pca.fit_transform(X_scaled)
X_scaled = MinMaxScaler().fit_transform(X_scaled)
```

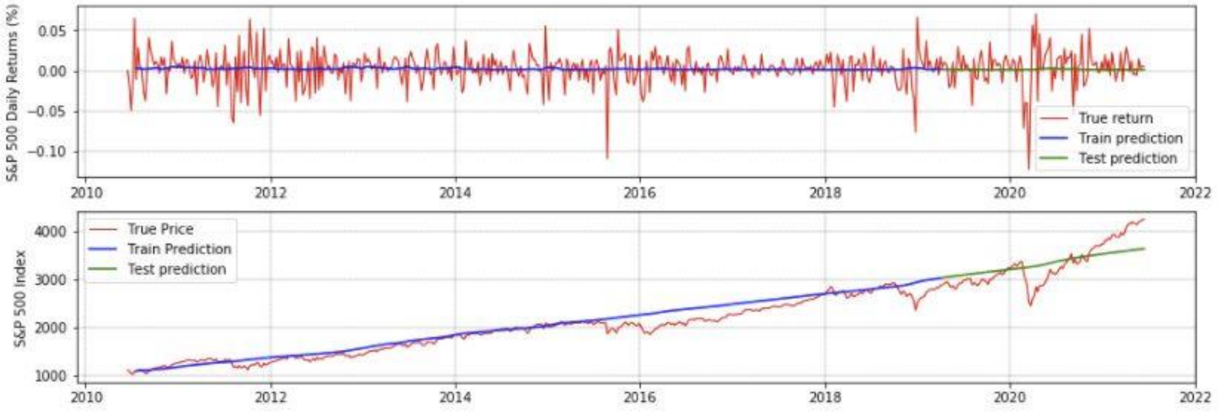
```
Training accuracy: 0.6826
Test accuracy: 0.6783
```



Model Iteration – Effect

- Training performance:
  - MSE: 0.0004 (increased from 0.0003)
  - Accuracy: 62% (increased from 60%)
- Test performance:
  - MSE: 0.0006
  - Accuracy: 66% (decreased from 67%)
- Minor improvements in accuracy, but slightly increased errors for training.





CONCLUSION

- We can predict stock market returns using investor positioning
- Using XGBoost, we were at least 60% and as much as 68% accurate in predicting direction of a week's returns
- Using LSTM, we were able to achieve similar accuracy, but more importantly infer long term trends in price based purely off investor sentiment.
- Limitations: Investor sentiment, LSTM was not great at predicting short term price fluctuation