Project 10K

Annual financial report and financial statements analysis using different machine learning techniques

Team 10K: Thomas Mitrevski, Paul Gill, Yvette Ly, Racim Badsi

PROPOSAL:

- Through sentiment analysis, investors can quickly understand if the tone of annual (10K) and quarterly (10Q) are positive, negative, or litigious etc. The overall sentiment expressed in these reports can then be used to help investors decide if they should invest in a company.

- On the other hand, many would argue that nothing beats the numbers, and many investors would prefer to look at a company’s financials before they invest.

Story:

- Our client wants to know if he should focus on the financials or the sentiment of the reports or both?

Hypothesis (Revenue Predictions on SEC Filings):

- Is it possible, using sentiment analysis, to predict revenue increases or decreases of certain stocks using a company’s financial reports from the last 10 years?

- OR is a Time Series Analysis of financial statements (balance sheets, income statements) a better predictor?

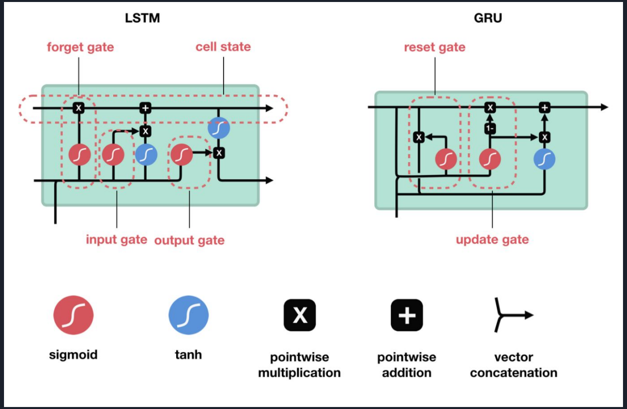
Motivation and Summary for Sentiment Analysis

- We want to look at the contents of 10-K and 10-Q statements to determine if the sentiment contained in the text will predict a shift in the prices during the week after the statement is released.

- If we can show that the sentiment contained in in a 10-K statement affects the price in a fashion different from the normal price movement, we may be able to take advantage of that in an algorithmic trading signal in the future.

Sentiment Analysis Model Summary

- We used a RNN in order to generate the signals that originated from the 10-K and 10-Q statements submitted to the SEC
- We generated a fairly deep model based off of a research paper created by Sander Blomme in 2020
- This paper found that the structure of the model succeeded best when using an embedding layer to create the initial vectors, as well as a bidirectional GRU instead of LSTM in order to develop a more comprehensive understanding of text and faster performance.
- Several Batch Normalization and Dropout layers were also included in order to increase performance and prevent overfitting

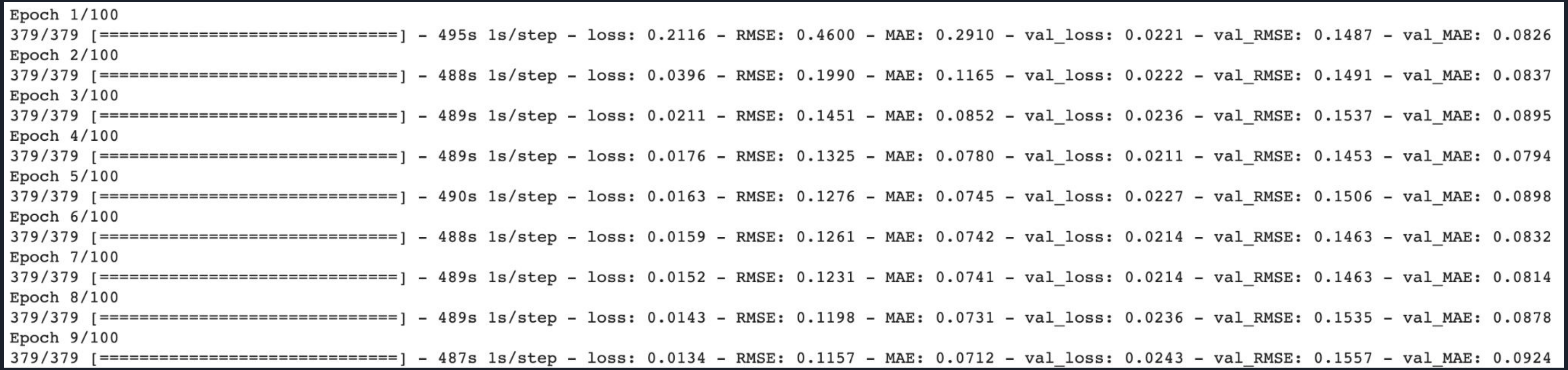


Data Cleanup

- SEC filings were retrieved from the EDGAR site over a period of multiple days
- SEC filings were cleaned using regular expressions and a technique noted by Jörg Hering in 2016
- From there, the text from sections 1, 1a, 7, and 7a were extracted as these contain the most relevant business text. These sections deal with a description of the company's business, any business risk factors, management's discussion of financial condition, and discussions of Market Risk.
- The prices were retrieved for the period of 3 months before and 1 month after using the Yahoo Finance API
- Fetching and the cleanup of data was way more difficult and took much longer than expected
- Once the SEC filings were retrieved and cleaned, n-grams were created from the texts in order to provide context for the words in the documents
- The n-grams were then tokenized in order to pass into the RNN

Model Training

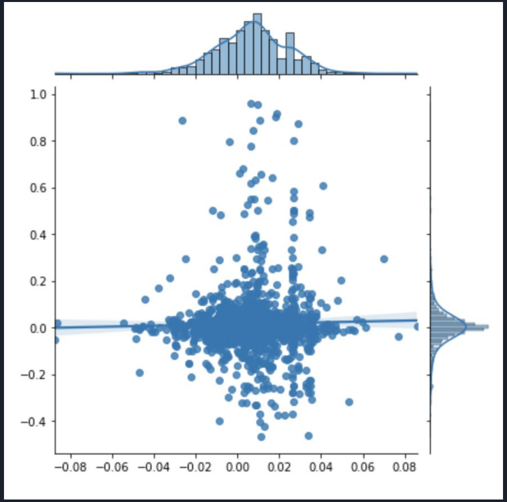
- The model was trained using the tokens from the SEC filings as well as the corresponding SEC data
- The RNN was designed to optimize for a regression was run to evaluate MSE and ran for 9 epochs
- Training took approximately 1 hour on a CoLab GPU. Cloud computing needed to be used and data size needed to be reduced as original average epoch time on a Mac was 8 hours.

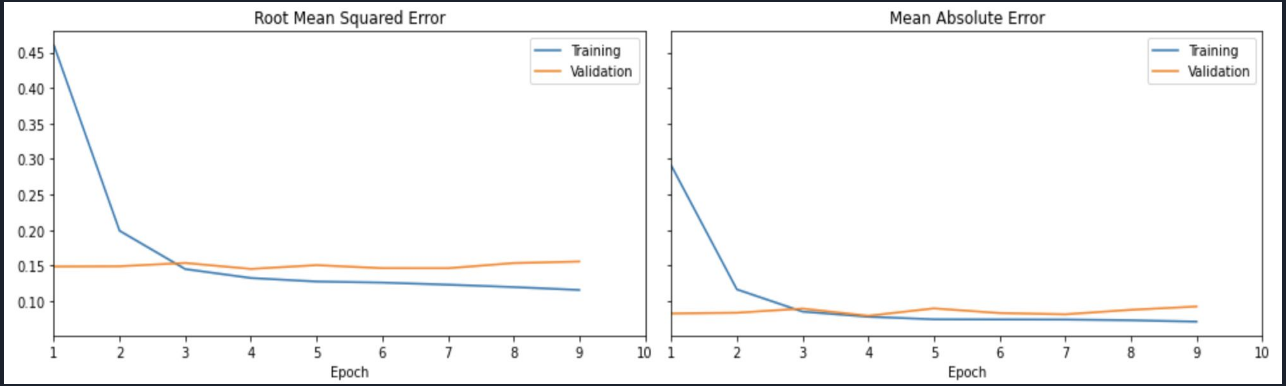


```
Epoch 1/100
379/379 [==============================] - 495s 1s/step - loss: 0.2116 - RMSE: 0.4600 - MAE: 0.2910 - val_loss: 0.0221 - val_RMSE: 0.1487 - val_MAE: 0.0826
Epoch 2/100
379/379 [==============================] - 488s 1s/step - loss: 0.0396 - RMSE: 0.1990 - MAE: 0.1165 - val_loss: 0.0222 - val_RMSE: 0.1491 - val_MAE: 0.0837
Epoch 3/100
379/379 [==============================] - 489s 1s/step - loss: 0.0211 - RMSE: 0.1451 - MAE: 0.0852 - val_loss: 0.0236 - val_RMSE: 0.1537 - val_MAE: 0.0895
Epoch 4/100
379/379 [==============================] - 489s 1s/step - loss: 0.0176 - RMSE: 0.1325 - MAE: 0.0780 - val_loss: 0.0211 - val_RMSE: 0.1453 - val_MAE: 0.0794
Epoch 5/100
379/379 [==============================] - 490s 1s/step - loss: 0.0163 - RMSE: 0.1276 - MAE: 0.0745 - val_loss: 0.0227 - val_RMSE: 0.1506 - val_MAE: 0.0898
Epoch 6/100
379/379 [==============================] - 488s 1s/step - loss: 0.0159 - RMSE: 0.1261 - MAE: 0.0742 - val_loss: 0.0214 - val_RMSE: 0.1463 - val_MAE: 0.0832
Epoch 7/100
379/379 [==============================] - 489s 1s/step - loss: 0.0152 - RMSE: 0.1231 - MAE: 0.0741 - val_loss: 0.0214 - val_RMSE: 0.1463 - val_MAE: 0.0814
Epoch 8/100
379/379 [==============================] - 489s 1s/step - loss: 0.0143 - RMSE: 0.1198 - MAE: 0.0731 - val_loss: 0.0236 - val_RMSE: 0.1535 - val_MAE: 0.0878
Epoch 9/100
379/379 [==============================] - 487s 1s/step - loss: 0.0134 - RMSE: 0.1157 - MAE: 0.0712 - val_loss: 0.0243 - val_RMSE: 0.1557 - val_MAE: 0.0924
```

Model Evaluation and Discussion

- We found a RMSE of .1557
- We also calculated a spearman correlation and this showed a 77% p-value that predicted score was influencing the actual price difference
- This is not sufficient because this value shows there is a high likelihood that the price change of a stock was influenced by things other than the sentiment contained in the 10-K or 10-Q release.





Postmortem

- Extreme difficulty retrieving and cleaning the data. The only reason we had enough time is because we started the project early.
- The text we received from the SEC site was UNICODE encoded, and a lot of extra characters had to be dropped in order to better process the data
- The model took an extremely long amount of time to train, and it was very difficult to run multiple iterations.
- We made arbitrary decision to truncate length of sentences and how many words we could look back on in order to speed up training, and these could probably be optimized.
- We could also change the size of the embeddings and add more data in order to further optimize the model.

Motivation and Summary for Time Series Analysis

- We want to look at the contents of 10-K and 10-Q statements to determine if the contents of the Income Statement, Balance Sheet and Statement of Cash Flows will predict a shift in the prices on the dates the statements are published.
- If we can show that the Key Performance Indicators (KPIs) contained in the Financial Statements affect the price in a fashion different from the normal price movement, we may be able to take advantage of that in an algorithmic trading signal in the future.

ROADBLOCK

- Because raw data from the 10-K and 10-Q SEC filings is extremely robust, we chose to retrieve data from SimFin
- By using SimFin Python API, we were able to download and use not just the raw financial data but the KPIs of publicly traded companies as well. For this particular machine learning model, we performed a time series analysis from the information derived from the Income Statement, Balance Sheet and Statement of Cash Flows.

Time Series Analysis Model Summary

- We downloaded datasets to include the Income Statements, Balance Sheets, Cash-Flow Statements, and Share Prices of companies traded in the US. Then used quarterly data for financial reports, and daily data for share-price.
- With a SimFin+ subscription, we were able to load the derived figures & ratios dataset. This dataset contains pre-calculated fundamental ratios which we then used as our signals.
- We used the SimFin tutorial presented by Hvass Laboratories as a guide for our models.
- Since some of the signals have a lot of missing data which causes problems in the statistical analysis, we removed all signals that have more than 25% missing data. Examples include, R&D/Revenue, Inventory Turnover, etc.
- From the remaining data, we used the mean-log returns of 1-3 year periods in the dataset. This is because data from the financial statements will be exactly the same for 3 months at a time, it cannot be used to predict short-term stock-returns.
- We removed outliers by “Winsorization” of the data.
- We looked at the linear correlation between the signals and stock-returns, to roughly assess which signals might be the best predictors for stock-returns.
- We split the dataset according to stock-tickers, so a ticker belongs to either the training- or test-set, but not both. We used 80% of all the tickers in the training-set, and 20% in the test-set.

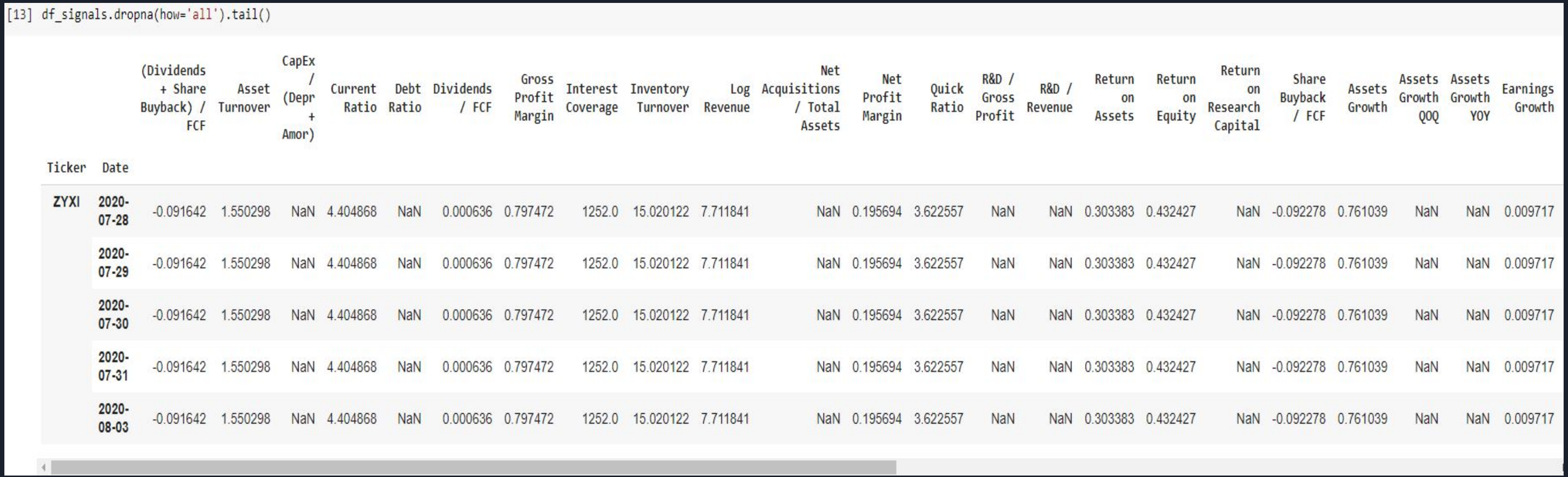
Signals

1. Financial Signals - Current Ratio, Debt Ratio, Net Profit Margin, Return on Assets, etc.

2. Growth Signals - Earnings Growth, FCF Growth, Sales Growth, etc.

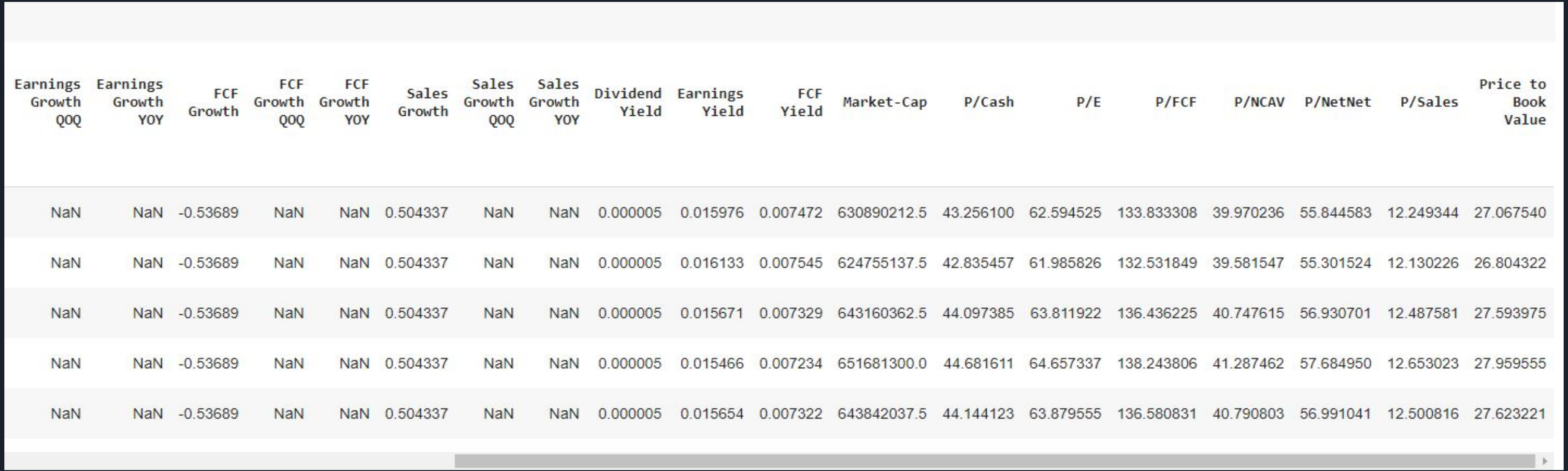
3. Valuation Signals - P/E, P/ Sales, P/Cash, etc.

Combined Signals Dataframe



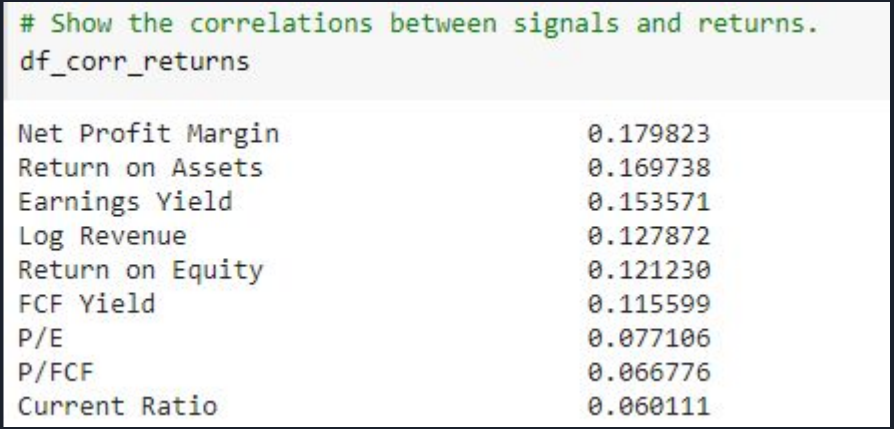
```
[13] df_signals.dropna(how='all').tail()
```

Ticker	Date	(Dividends + Share Buyback) / FCF	Asset Turnover	CapEx / (Depr + Amor)	Current Ratio	Debt Ratio	Dividends / FCF	Gross Profit Margin	Interest Coverage	Inventory Turnover	Log Revenue	Net Acquisitions / Total Assets	Net Profit Margin	Quick Ratio	R&D / Gross Profit	R&D / Revenue	Return on Assets	Return on Equity	Return on Research Capital	Share Buyback / FCF	Assets Growth	Assets Growth QOQ	Assets Growth YOY	Earnings Growth
ZYXI	2020-07-28	-0.091642	1.550298	NaN	4.404868	NaN	0.000636	0.797472	1252.0	15.020122	7.711841	NaN	0.195694	3.622557	NaN	NaN	0.303383	0.432427	NaN	-0.092278	0.761039	NaN	NaN	0.009717
	2020-07-29	-0.091642	1.550298	NaN	4.404868	NaN	0.000636	0.797472	1252.0	15.020122	7.711841	NaN	0.195694	3.622557	NaN	NaN	0.303383	0.432427	NaN	-0.092278	0.761039	NaN	NaN	0.009717
	2020-07-30	-0.091642	1.550298	NaN	4.404868	NaN	0.000636	0.797472	1252.0	15.020122	7.711841	NaN	0.195694	3.622557	NaN	NaN	0.303383	0.432427	NaN	-0.092278	0.761039	NaN	NaN	0.009717
	2020-07-31	-0.091642	1.550298	NaN	4.404868	NaN	0.000636	0.797472	1252.0	15.020122	7.711841	NaN	0.195694	3.622557	NaN	NaN	0.303383	0.432427	NaN	-0.092278	0.761039	NaN	NaN	0.009717
	2020-08-03	-0.091642	1.550298	NaN	4.404868	NaN	0.000636	0.797472	1252.0	15.020122	7.711841	NaN	0.195694	3.622557	NaN	NaN	0.303383	0.432427	NaN	-0.092278	0.761039	NaN	NaN	0.009717



Earnings Growth QOQ	Earnings Growth YOY	FCF Growth	FCF Growth QOQ	FCF Growth YOY	Sales Growth	Sales Growth QOQ	Sales Growth YOY	Dividend Yield	Earnings Yield	FCF Yield	Market-Cap	P/Cash	P/E	P/FCF	P/NCAV	P/NetNet	P/Sales	Price to Book Value
NaN	NaN	-0.53689	NaN	NaN	0.504337	NaN	NaN	0.000005	0.015976	0.007472	630890212.5	43.256100	62.594525	133.833308	39.970236	55.844583	12.249344	27.067540
NaN	NaN	-0.53689	NaN	NaN	0.504337	NaN	NaN	0.000005	0.016133	0.007545	624755137.5	42.835457	61.985826	132.531849	39.581547	55.301524	12.130226	26.804322
NaN	NaN	-0.53689	NaN	NaN	0.504337	NaN	NaN	0.000005	0.015671	0.007329	643160362.5	44.097385	63.811922	136.436225	40.747615	56.930701	12.487581	27.593975
NaN	NaN	-0.53689	NaN	NaN	0.504337	NaN	NaN	0.000005	0.015466	0.007234	651681300.0	44.681611	64.657337	138.243806	41.287462	57.684950	12.653023	27.959555
NaN	NaN	-0.53689	NaN	NaN	0.504337	NaN	NaN	0.000005	0.015654	0.007322	643842037.5	44.144123	63.879555	136.580831	40.790803	56.991041	12.500816	27.623221

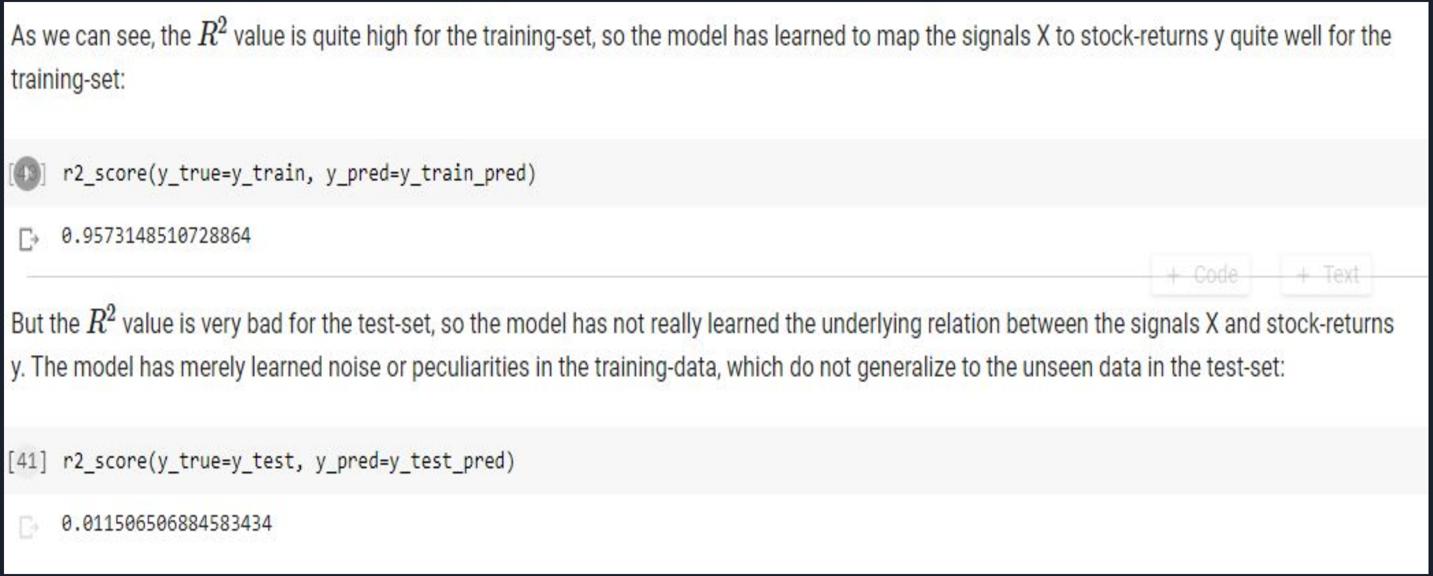
Correlation: Signals vs Returns



```
# Show the correlations between signals and returns.
df_corr_returns
```

```
Net Profit Margin                    0.179823
Return on Assets                     0.169738
Earnings Yield                       0.153571
Log Revenue                          0.127872
Return on Equity                     0.121230
FCF Yield                            0.115599
P/E                                  0.077106
P/FCF                                0.066776
Current Ratio                        0.060111
```

REGRESSION MODEL & RESULTS



As we can see, the R^2 value is quite high for the training-set, so the model has learned to map the signals X to stock-returns y quite well for the training-set:

```
r2_score(y_true=y_train, y_pred=y_train_pred)
```

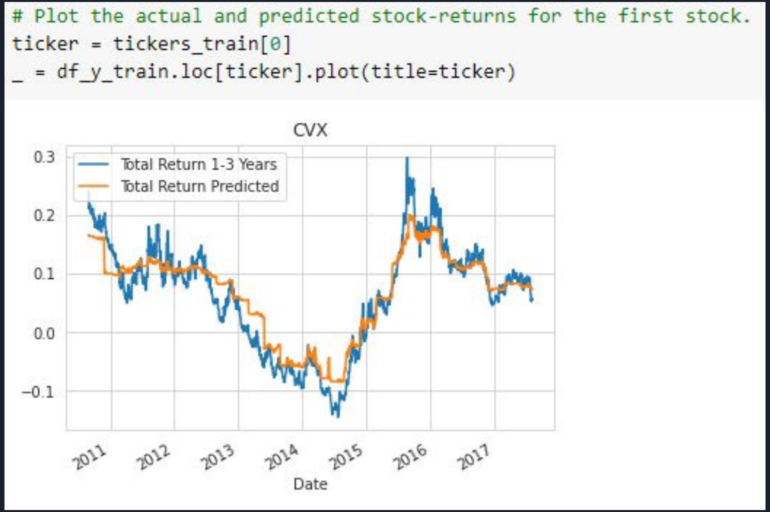
```
0.9573148510728864
```

But the R^2 value is very bad for the test-set, so the model has not really learned the underlying relation between the signals X and stock-returns y. The model has merely learned noise or peculiarities in the training-data, which do not generalize to the unseen data in the test-set:

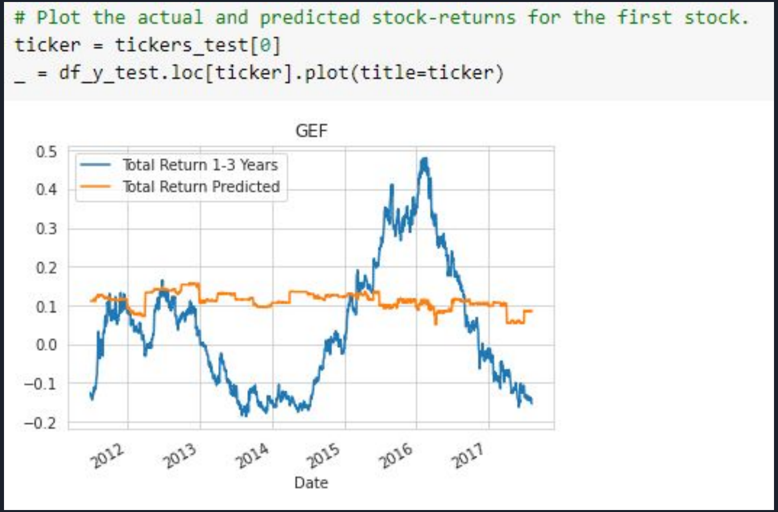
```
[41] r2_score(y_true=y_test, y_pred=y_test_pred)
```

```
0.011506506884583434
```

Actual and Predicted stock-returns plots for CVX and GEF



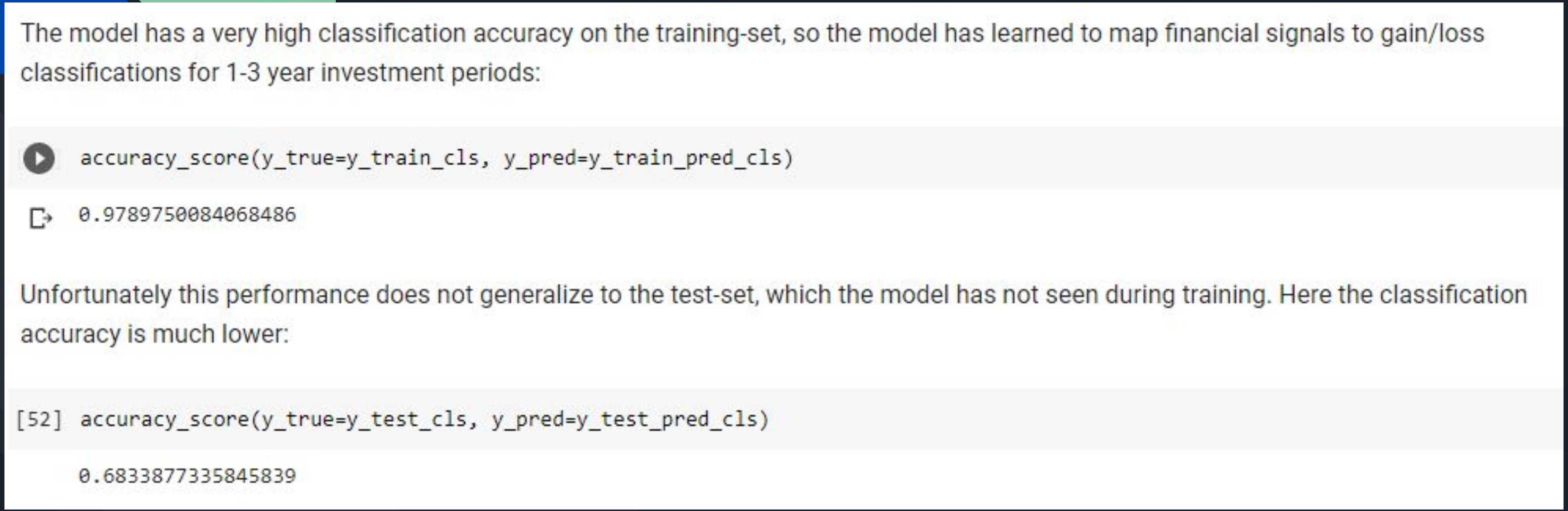
```
# Plot the actual and predicted stock-returns for the first stock.
ticker = tickers_train[0]
_ = df_y_train.loc[ticker].plot(title=ticker)
```



```
# Plot the actual and predicted stock-returns for the first stock.
ticker = tickers_test[0]
_ = df_y_test.loc[ticker].plot(title=ticker)
```

Train set vs Test Set

CLASSIFICATION



The model has a very high classification accuracy on the training-set, so the model has learned to map financial signals to gain/loss classifications for 1-3 year investment periods:

```
accuracy_score(y_true=y_train_cls, y_pred=y_train_pred_cls)
```

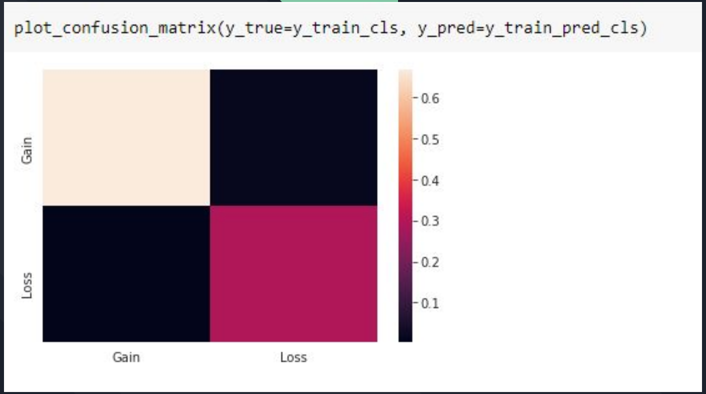
```
0.9789750084068486
```

Unfortunately this performance does not generalize to the test-set, which the model has not seen during training. Here the classification accuracy is much lower:

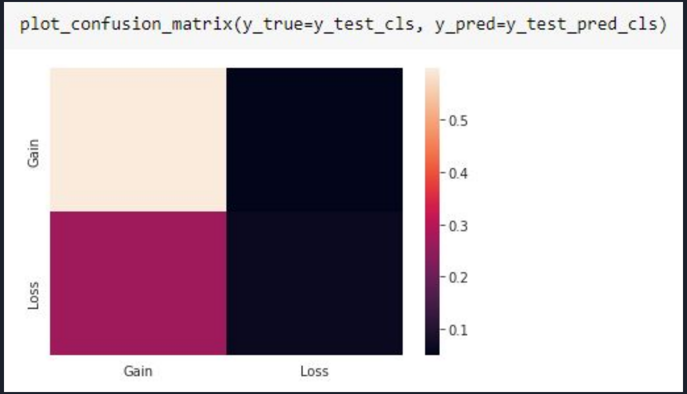
```
[52] accuracy_score(y_true=y_test_cls, y_pred=y_test_pred_cls)
```

```
0.6833877335845839
```

Confusion Matrix



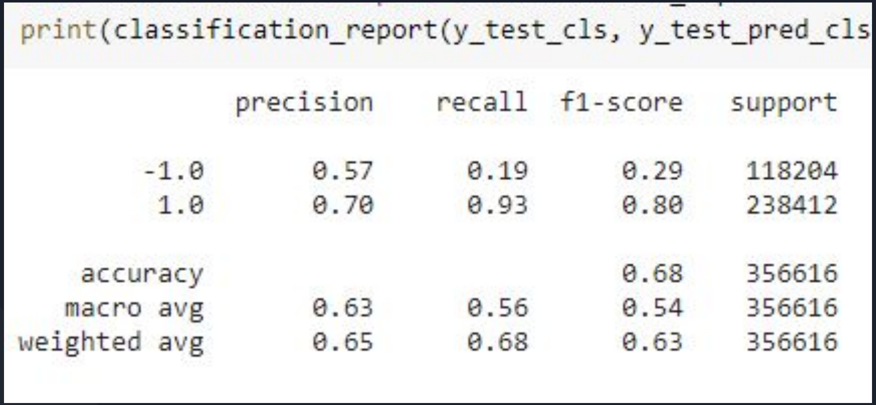
```
plot_confusion_matrix(y_true=y_train_cls, y_pred=y_train_pred_cls)
```



```
plot_confusion_matrix(y_true=y_test_cls, y_pred=y_test_pred_cls)
```

Train Set vs Test Set

Classification Report



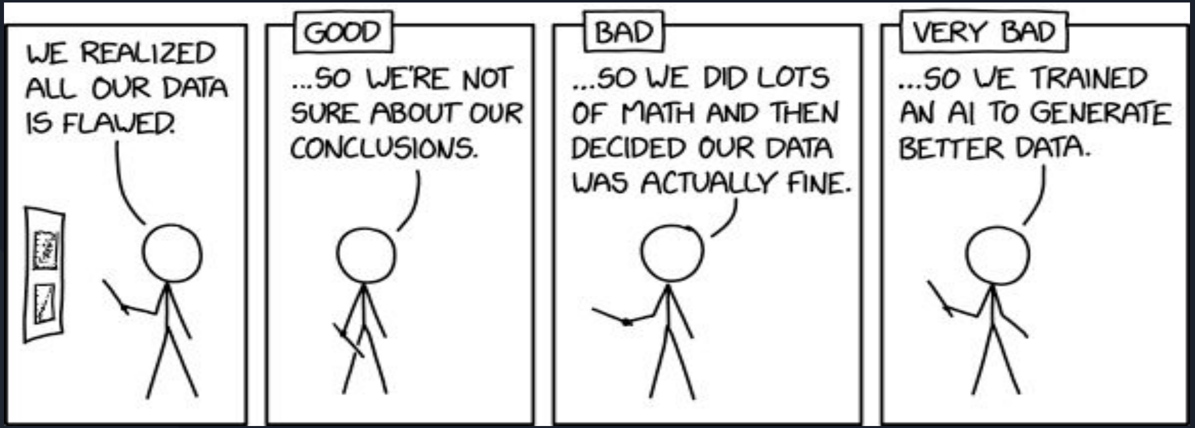
```
print(classification_report(y_test_cls, y_test_pred_cls

              precision    recall  f1-score   support

        -1.0       0.57      0.19      0.29    118204
         1.0       0.70      0.93      0.80    238412

    accuracy                           0.68    356616
   macro avg       0.63      0.56      0.54    356616
weighted avg       0.65      0.68      0.63    356616
```

CONCLUSIONS


WE REALIZED ALL OUR DATA IS FLAWED.
GOOD
...SO WE'RE NOT SURE ABOUT OUR CONCLUSIONS.
BAD
...SO WE DID LOTS OF MATH AND THEN DECIDED OUR DATA WAS ACTUALLY FINE.
VERY BAD
...SO WE TRAINED AN AI TO GENERATE BETTER DATA.