Predicting next day stock movement

Andrew Foote, Evgueni Freiman

Our Goal

The goal of our project is to try and predict next day stock movement (up or down) with an accuracy greater than 50%. We limited our target stocks to big US company stocks.

We chose this topic because traditionally getting a strong model for stock market movement is very difficult… otherwise everyone would be rich! Since the project doesn't focus on creating a super strong model but rather learning new tools and exploring we thought it would be fun to have a difficult topic.

Data Sets Used

We used three datasets. Two datasets are from Kaggle, one is from Google Dataset Search

Daily price history dataset #1:
1.8gb, 19 million rows*
3457 unique stocks
From 1999 to 2022

Daily price history dataset #2:
719mb, 14 million rows*
7195 unique stocks
From 1999 to 2017

*after preliminary cleanup and transformation that will be described later

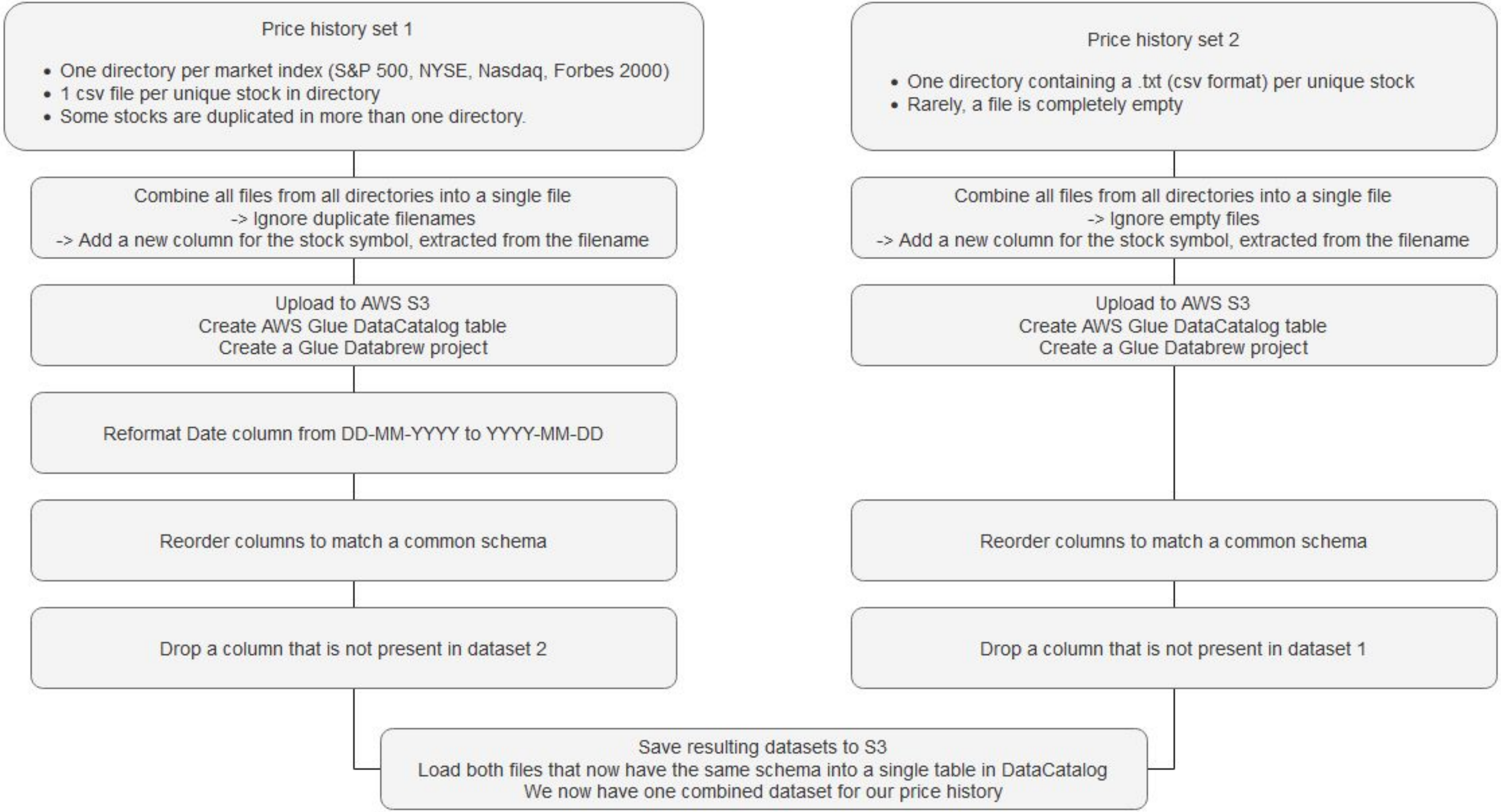
The first dataset has fewer companies but more recent prices. The second dataset has more companies but less recent prices. The two sets have some amount of overlap.

Dataset #3: data about S&P companies, notably their industrial sector (18.7kb)

Data Integration

Primary Goal: combine the two price history sets into one set so that we can then analyze and transform the data further.

Secondary goal: integrate the data from dataset #3. This is done after combining and partially cleaning the price history data.


Price history set 1
• One directory per market index (S&P 500, NYSE, Nasdaq, Forbes 2000)
• 1 csv file per unique stock in directory
• Some stocks are duplicated in more than one directory.
Price history set 2
• One directory containing a .txt (csv format) per unique stock
• Rarely, a file is completely empty
Combine all files from all directories into a single file
-> Ignore duplicate filenames
-> Add a new column for the stock symbol, extracted from the filename
Combine all files from all directories into a single file
-> Ignore empty files
-> Add a new column for the stock symbol, extracted from the filename
Upload to AWS S3
Create AWS Glue DataCatalog table
Create a Glue Databrew project
Upload to AWS S3
Create AWS Glue DataCatalog table
Create a Glue Databrew project
Reformat Date column from DD-MM-YYYY to YYYY-MM-DD
Reorder columns to match a common schema
Reorder columns to match a common schema
Drop a column that is not present in dataset 2
Drop a column that is not present in dataset 1
Save resulting datasets to S3
Load both files that now have the same schema into a single table in DataCatalog
We now have one combined dataset for our price history

	Symbol	Date	Low	High	Open	Close	Volume
0	A	1999-11-18	28.612302780151367	35.765380859375	32.54649353027344	31.473533630371094	62546380
1	A	1999-11-19	28.47818374633789	30.75822639465332	30.713518142700195	28.880544662475586	15234146
2	A	1999-11-22	28.65700912475586	31.473533630371094	29.551143646240234	31.473533630371094	6577870
3	A	1999-11-23	28.612302780151367	31.205293655395508	30.400571823120117	28.612302780151367	5975611
4	A	1999-11-24	28.612302780151367	29.998212814331055	28.701717376708984	29.372318267822266	4843231
...	...	...	...	...	...	...	...
3340412	WFC	2022-12-06	42.65999984741211	43.849998474121094	43.68000030517578	43.400001525878906	25961400
3340413	WFC	2022-12-07	42.439998626708984	43.34000015258789	43.09000015258789	42.4500007629394 5	24114900
3340414	WFC	2022-12-08	42.11000061035156	42.88999938964844	42.709999084472656	42.58000183105469	17161400
3340415	WFC	2022-12-09	42.31999969482422	42.91999816894531	42.33000183105469	42.5	16022700
3340416	WFC	2022-12-12	42.11000061035156	42.59749984741211	42.599998474121094	42.59749984741211	3501355

Match on Symbol col.

	Symbol	Name	Sector
0	MMM	3M Company	Industrials
1	AOS	A.O. Smith Corp	Industrials
2	ABT	Abbott Laboratories	Health Care
3	ABBV	AbbVie Inc.	Health Care
4	ACN	Accenture plc	Information Technology
...	...	...	...
500	XYL	Xylem Inc.	Industrials
501	YUM	Yum! Brands Inc	Consumer Discretionary
502	ZBH	Zimmer Biomet Holdings	Health Care
503	ZION	Zions Bancorp	Financials
504	ZTS	Zoetis	Health Care

Combining the combined price history set and the Sector column from dataset #3.

Records with Symbols that did not exist in dataset #3 were filtered out.

Data Cleaning

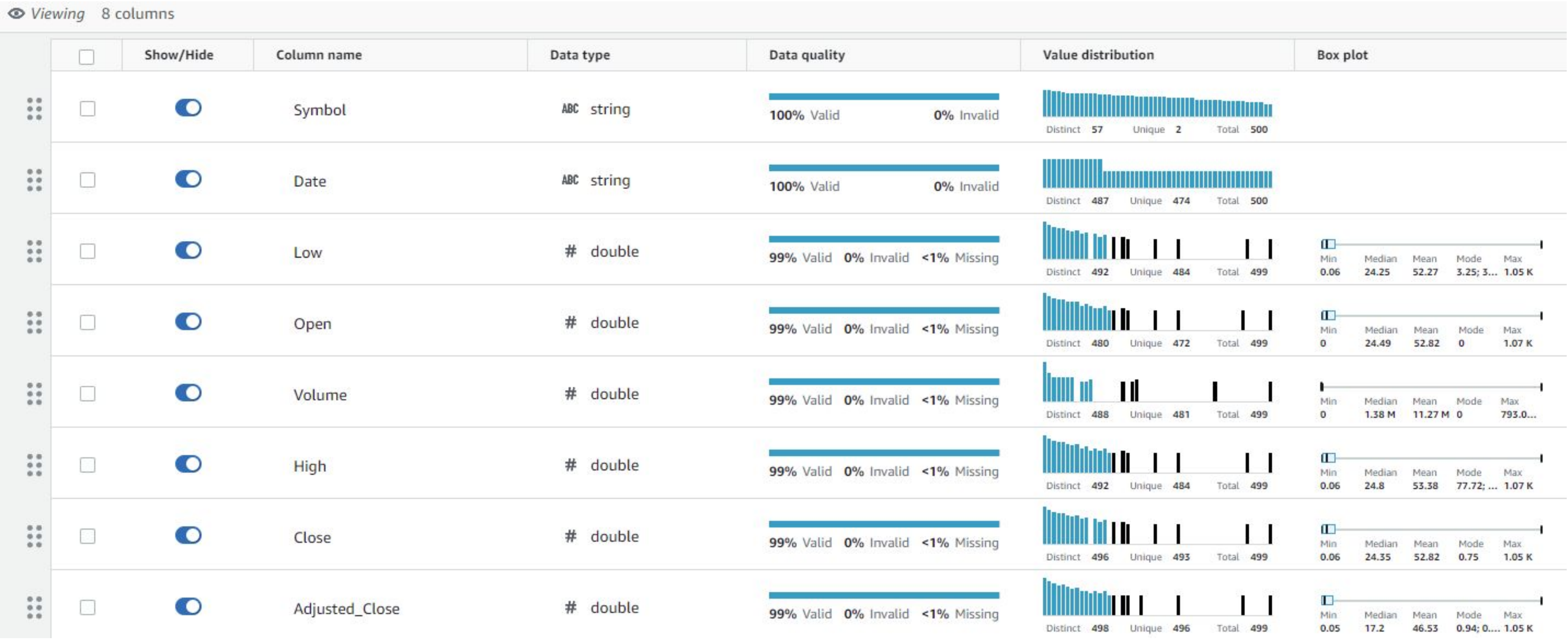
- Skip duplicate files in dataset 1 (mentioned in integration part)
- Skip empty files in dataset 2 (mentioned in integration part)
- Rarely, Volume values are formatted not as integers but as decimals ending with ".0". We reformatted these as integers.
- Rarely, a row has null values for everything but the Symbol and Date. These rows were deleted.
- More on next slide

Data Cleaning cont.

- Price history set 1 and 2 have some overlap, so combining them resulted in duplicate records.
- However, the values for a given stock on a given day differed between the two sets.
- To resolve this, duplicates were determined based on the Symbol and Date values only. The values from price history set 1 were given priority because set 1 covers a wider date range (up to 2022).

	Symbol	Date	Low	High	Open	Close	Volume
0	A	1999-11-18	28.612303	35.765381	32.546494	31.473534	62546380.0
3340417	A	1999-11-18	27.002000	33.754000	30.713000	29.702000	66277506.0
1	A	1999-11-19	28.478184	30.758226	30.713518	28.880545	15234146.0
3340418	A	1999-11-19	26.872000	29.027000	28.986000	27.257000	16142920.0
2	A	1999-11-22	28.657009	31.473534	29.551144	31.473534	6577870.0
...	...	...	...	...	...	...	...

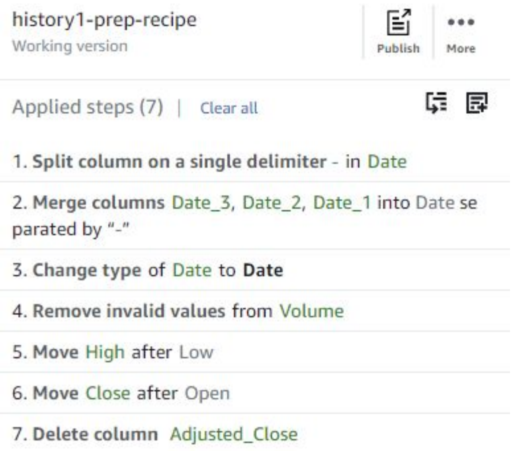
Some screenshots from Glue DataBrew...



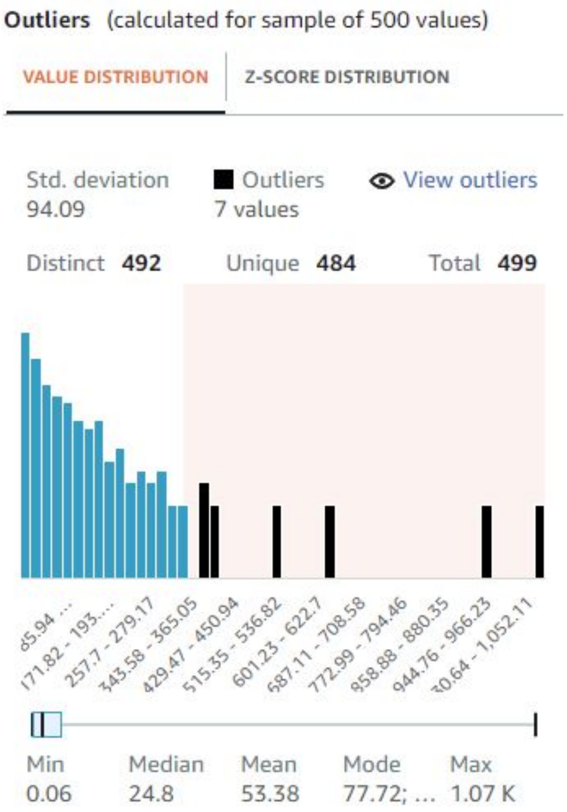
Viewing 8 columns

Show/Hide	Column name	Data type	Data quality	Value distribution	Box plot
	Symbol	ABC string	100% Valid 0% Invalid	Distinct 57 Unique 2 Total 500	
	Date	ABC string	100% Valid 0% Invalid	Distinct 487 Unique 474 Total 500	
	Low	# double	99% Valid 0% Invalid <1% Missing	Distinct 492 Unique 484 Total 499	Min 0.06 Median 24.25 Mean 52.27 Mode 3.25; 3... Max 1.05 K
	Open	# double	99% Valid 0% Invalid <1% Missing	Distinct 480 Unique 472 Total 499	Min 0 Median 24.49 Mean 52.82 Mode 0 Max 1.07 K
	Volume	# double	99% Valid 0% Invalid <1% Missing	Distinct 488 Unique 481 Total 499	Min 0 Median 1.38 M Mean 11.27 M Mode 0 Max 793.0...
	High	# double	99% Valid 0% Invalid <1% Missing	Distinct 492 Unique 484 Total 499	Min 0.06 Median 24.8 Mean 53.38 Mode 77.72; ... Max 1.07 K
	Close	# double	99% Valid 0% Invalid <1% Missing	Distinct 496 Unique 493 Total 499	Min 0.06 Median 24.35 Mean 52.82 Mode 0.75 Max 1.05 K
	Adjusted_Close	# double	99% Valid 0% Invalid <1% Missing	Distinct 498 Unique 496 Total 499	Min 0.05 Median 17.2 Mean 46.53 Mode 0.94; 0.... Max 1.05 K

Schema tab where we can see the schema (this is not our final schema), and data distribution for a 500 row sample. <1% Missing is not 0%. It means there is a small amount of values missing.



Glue DataBrew recipe used to integrate price history set 1. All steps that we do in DataBrew are saved as recipes. These recipes can be automatically re-applied later if new data is added.



Detecting outliers. There are different options to deal with outliers if desired.

Data Transformation

- The Volume column has a wide range of values from 0 to over 2 billion for some outliers. Some stocks always have a lot of volume and some always have little. We wanted to emphasize the relative fluctuations in Volume rather than the absolute value.
- We applied logarithmic scaling on the Volume column to reduce the spread.
- Then we rescaled the Volume to be between 0 and 1 and applied a normal distribution to the values in the column.

# Volume	# Volume_outlier_rescaled	# Volume_outlier_rescaled_norm...
Distinct 492 Unique 487 Total 500	Distinct 492 Unique 487 Total 496	Distinct 492 Unique 487 Total 496
Min 0 Median 1.46 M Mean 13.64 M Mode 0 Max 1.44 B	Min 5.99 Median 14.2 Mean 13.94 Mode Max 21.08	Min 0 Median 0.54 Mean 0.53 Mode Max 1
72800	11.1954712360754	0.344787547671874
77400	11.2567420614788	0.348847000109466
80600	11.2972539304025	0.351531083504315
81200	11.3046705280588	0.352022464601865
82300	11.3181263885765	0.352913972516418
89127	11.3978175997823	0.358193853756050
91463	11.4236897998278	0.359907996904101
100400	11.5169174881847	0.366084727252674
100800	11.5208936365650	0.366348163972000
100800	11.5208936365650	0.366348163972000
101680	11.5295859078104	0.366924063863392
103800	11.5502212516644	0.368291243044633
110944	11.6167808504465	0.372701099183213
111900	11.6253608962632	0.373269563663266
113100	11.6360276640692	0.373976282338691
113600	11.6404387872349	0.374268537985154
122700	11.7174976326778	0.379374013811722
124800	11.7344677368991	0.380498355298783
124902	11.7352847107775	0.380552483288413
126200	11.7456232310728	0.381237454171985

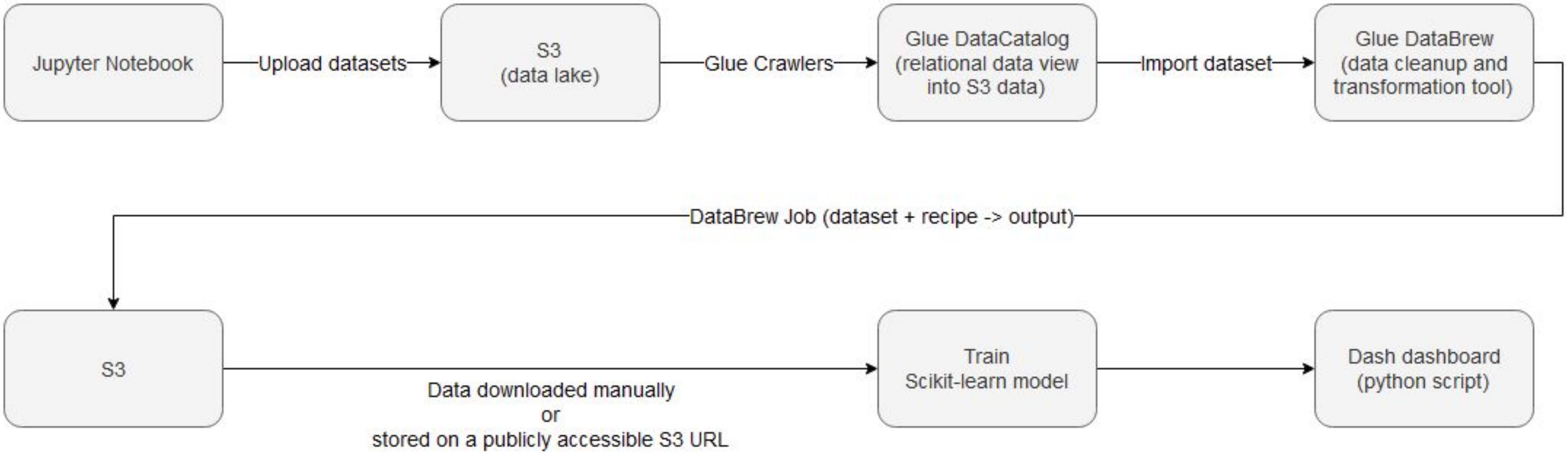
Data Transformation cont.

- Added a column called ‘Label’ that has the value ‘0’ if the stock went down in price for that day, and ‘1’ if the price stayed the same or went up. This column is used to train the ML model.
- Added a column ‘WeeklyDiff’ that holds the change in price over the previous week.
- Added columns ‘DayOfWeek’ and ‘Month’.
- For training the ML model we encoded the Sector column values as integers rather than strings.

	Symbol	Date	Low	High	Open	Close	Volume	VolumeScaledNormalized	WeekDiff	Sector	Label	DayOfWeek	Month
10000	AAL	2022-05-27	17.340000	18.209999	17.450001	18.129999	27615600	0.753878	0.350000	0	1	4	5
10001	AAL	2022-05-31	17.510000	18.219999	17.700001	17.870001	31158300	0.759189	1.379999	0	0	1	5
10002	AAL	2022-06-01	16.980000	18.100000	18.070000	17.290001	32637900	0.761230	1.610001	0	0	2	6
10003	AAL	2022-06-02	17.180000	17.510000	17.270000	17.459999	23337700	0.746473	0.540001	0	1	3	6
10004	AAL	2022-06-03	16.090000	17.049999	17.000000	16.219999	46267600	0.776584	1.959999	0	0	4	6
...	...	...	...	...	...	...	...	...	...	...	...	...	...
100095	ADSK	2000-04-11	10.656250	11.281250	11.000000	10.703125	2308400	0.644680	-0.296875	2	0	1	4
100096	ADSK	2000-04-12	9.593750	10.968750	10.718750	9.734375	4786800	0.676769	-0.296875	2	0	2	4
100097	ADSK	2000-04-13	9.484375	10.531250	9.875000	9.750000	2879600	0.654408	-1.312500	2	1	3	4
100098	ADSK	2000-04-14	8.953125	9.726563	9.421875	8.953125	3041200	0.656810	-1.578125	2	0	4	4
100099	ADSK	2000-04-17	8.875000	9.453125	8.890625	9.359375	3780400	0.666383	-2.265625	2	1	0	4

A sample subset of our final dataset looks like this, before passing it to the ML script. The ML script will drop some of the columns like Date that should not be included in the training data.

Overall data pipeline



Feature extraction

For selecting the data features that the ML model is trained on, we had to drop some columns that were either not relevant or allowed the model to “cheat” as we discovered when testing the model.

We dropped columns Open, Close, Low, High because this information is not available for a future day that we are trying to predict.

We dropped Date because it meaningless when trying to predict the future price of a stock.

We selected VolumeScaledNormalized, WeekDiff, Sector, DayOfWeek, Month for the training features.

ML & Data Visualization

Once data was ready to be ran through the model we chose 3 different models to run

1) Random Forest Classifier
2) Decision Tree Classifier
3) XGBoost Classifier

Main Issues:

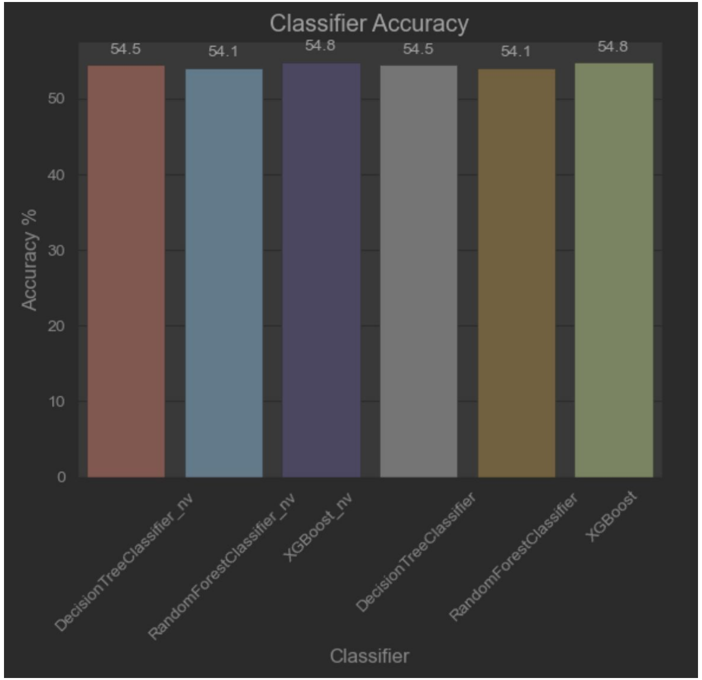
First runs resulted in a very high accuracy of ~64% which was very high (Issue was resolved by manipulating the weeklyDiff variable to not include the current day's difference)

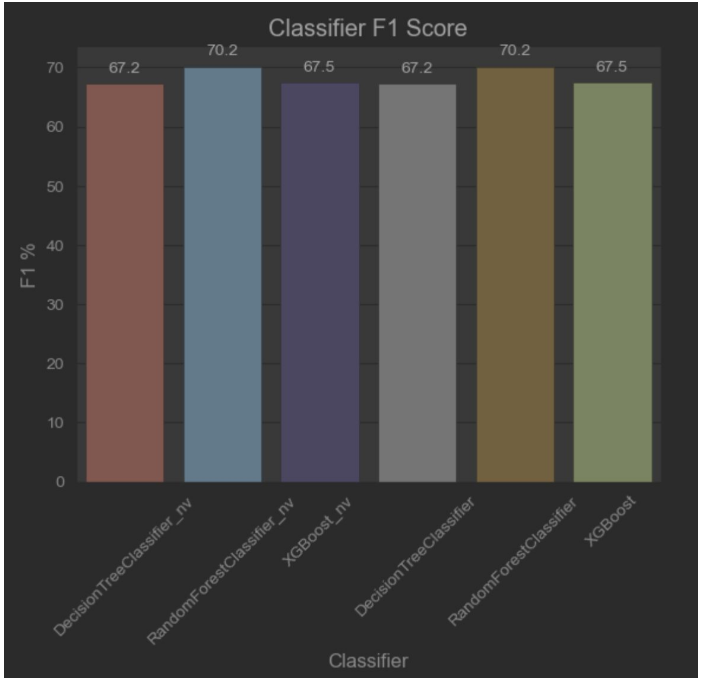
We forgot to remove the closedDiff, had to troubleshoot the 100% accuracy

ML & Data Visualization (Cont.)

Process for running 6 models with sklearn and xgboost

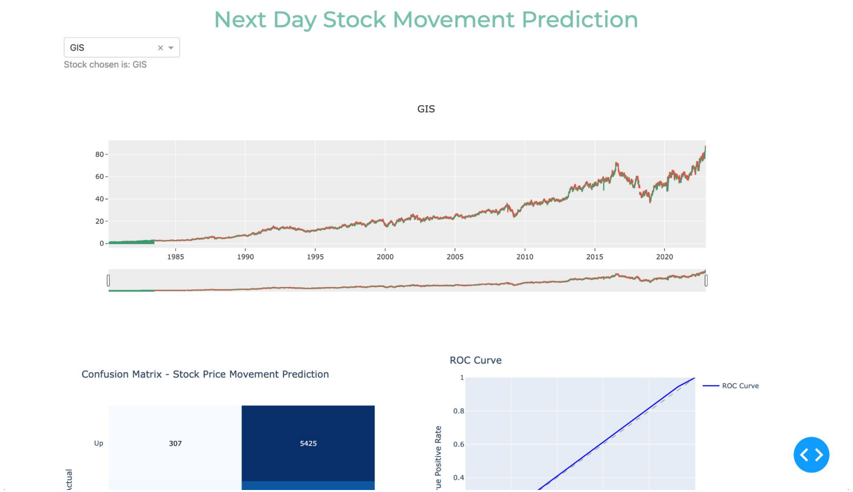
```
import xgboost as xgb
xgb_model = xgb.XGBClassifier(objective="binary:logistic", random_state=42)
xgb_model.fit(X_train, y_train)

pred = xgb_model.predict(X_test)

xgb_acc_nv = metrics.accuracy_score(y_test,pred)
xgb_f1_nv = metrics.f1_score(y_test,pred)

print(metrics.accuracy_score(y_test,pred))
print(metrics.f1_score(y_test,pred))

confusion = metrics.confusion_matrix(y_test, pred)

plt.figure(figsize=(9,9))
sns.heatmap(confusion, annot=True, linewidths=.5, square = True, cmap = 'Blues_r');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(rf_acc_nv)
plt.title(all_sample_title, size = 15);
plt.show()
```





We compared the accuracy and F1 scores of each model we tested and decided on the XGBoost model as it performed the best all-round but only marginally

Dashboard



We created a dash dashboard to showcase the model we built along with displaying the price history of the stock selected. The dashboard can run the model on entire stocks and compares it the actual movement of up or down

Having never used dash it works a lot like Javascript having callback functions that allow the figures to update in real time on the page.

Unlike the graphs in the Jupyter Notebook where we used, matplotlib and seaborn. We used Plotly since it is the creator of dash and works very well together

To finish off the presentation we'll showcase the dashboard