Gradient Boosted Machines

(For Stocks)

Project Location

- https://github.com/AIClubUA/Weekly-Meetings/tree/master/September%202018/Week%20of%209-17-2018/gbm-for-stocks

Disclaimer:

- Do NOT start trading based off of this presentation
- This is for demonstration purposes only and does not account for the numerous considerations to take into account when formulating a trading strategy
- The primary focus of this is showcasing how easy it is to turn a .csv of data into useful and interesting insight with a low barrier to entry

Problem Framing

- Given a bunch of stock market data, can we predict if the stock will go up or down the next day?
- This is a classification problem
  - Will be using 3 classes
    - Up
    - Down
    - Neutral

Data

- We will be using .csv data sourced from Yahoo Finance
- Can download manually from here:
  - https://finance.yahoo.com/quote/TSLA/history?p=TSLA
- Or use:
  - pip install yapywrangler (on command line)

```
import yapywrangler as yp

stocks = ['FB', 'TSLA', 'BAC']
data = yp.securityData(stocks, end='2010-01-01', save=True, epoch=False)
```

Data Snippet

	A	B	C	D	E	F
1	date	open	high	low	close	volume
2	1.54E+09	290.04	295	288.13	293.09	2139507
3	1.54E+09	288.76	297.33	286.52	295.2	6763600
4	1.54E+09	288.02	295	285.18	289.46	6340300
5	1.54E+09	281.44	292.5	278.65	290.54	10015400
6	1.54E+09	279.47	282	273.55	279.44	9170000
7	1.54E+09	273.26	286.03	271	285.5	14283500
8	1.54E+09	260.1	268.35	252.25	263.24	22491900
9	1.54E+09	284.8	291.17	278.88	280.95	7480800
10	1.54E+09	285.05	286.78	277.18	280.74	7720800
11	1.54E+09	296.94	298.19	288	288.95	8350500
12	1.54E+09	302	305.31	298.6	301.66	5375100
13	1.54E+09	302.26	304.6	297.72	303.15	7216700
14	1.54E+09	310.27	311.85	303.69	305.01	7447400
15	1.54E+09	318.41	318.88	311.19	311.86	7649100

Data – Inputs and Labels

- Inputs
  - Our inputs will be the 10 previous trading days
  - Just using the closing price
- Labels
  - Generated by looking through past data
  - If “tomorrow” saw > 1% growth
    - Label = 2
  - If < 1%
    - Label = 1
  - Else, Label = 0

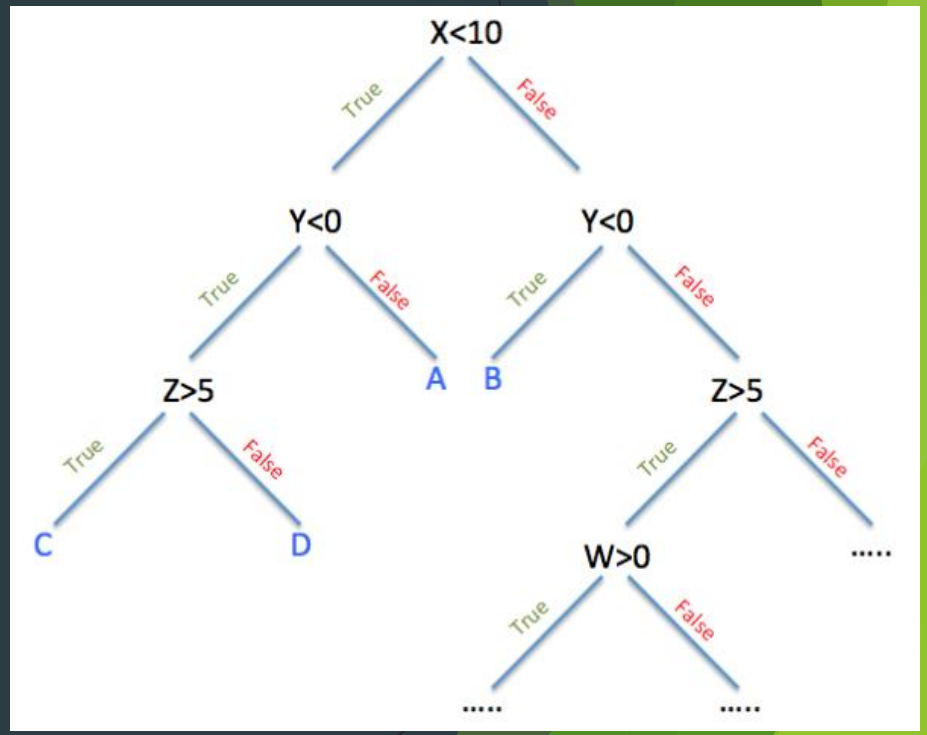
	A	B	C	D	E	F
1	date	open	high	low	close	volume
2	1.54E+09	290.04	295	288.13	293.09	2139507
3	1.54E+09	288.76	297.33	286.52	295.2	6763600
4	1.54E+09	288.02	295	285.18	289.46	6340300
5	1.54E+09	281.44	292.5	278.65	290.54	10015400
6	1.54E+09	279.47	282	273.55	279.44	9170000
7	1.54E+09	273.26	286.03	271	285.5	14283500
8	1.54E+09	260.1	268.35	252.25	263.24	22491900
9	1.54E+09	284.8	291.17	278.88	280.95	7480800
10	1.54E+09	285.05	286.78	277.18	280.74	7720800
11	1.54E+09	296.94	298.19	288	288.95	8350500
12	1.54E+09	302	305.31	298.6	301.66	5375100
13	1.54E+09	302.26	304.6	297.72	303.15	7216700
14	1.54E+09	310.27	311.85	303.69	305.01	7447400

Model – Gradient Boosted Machine

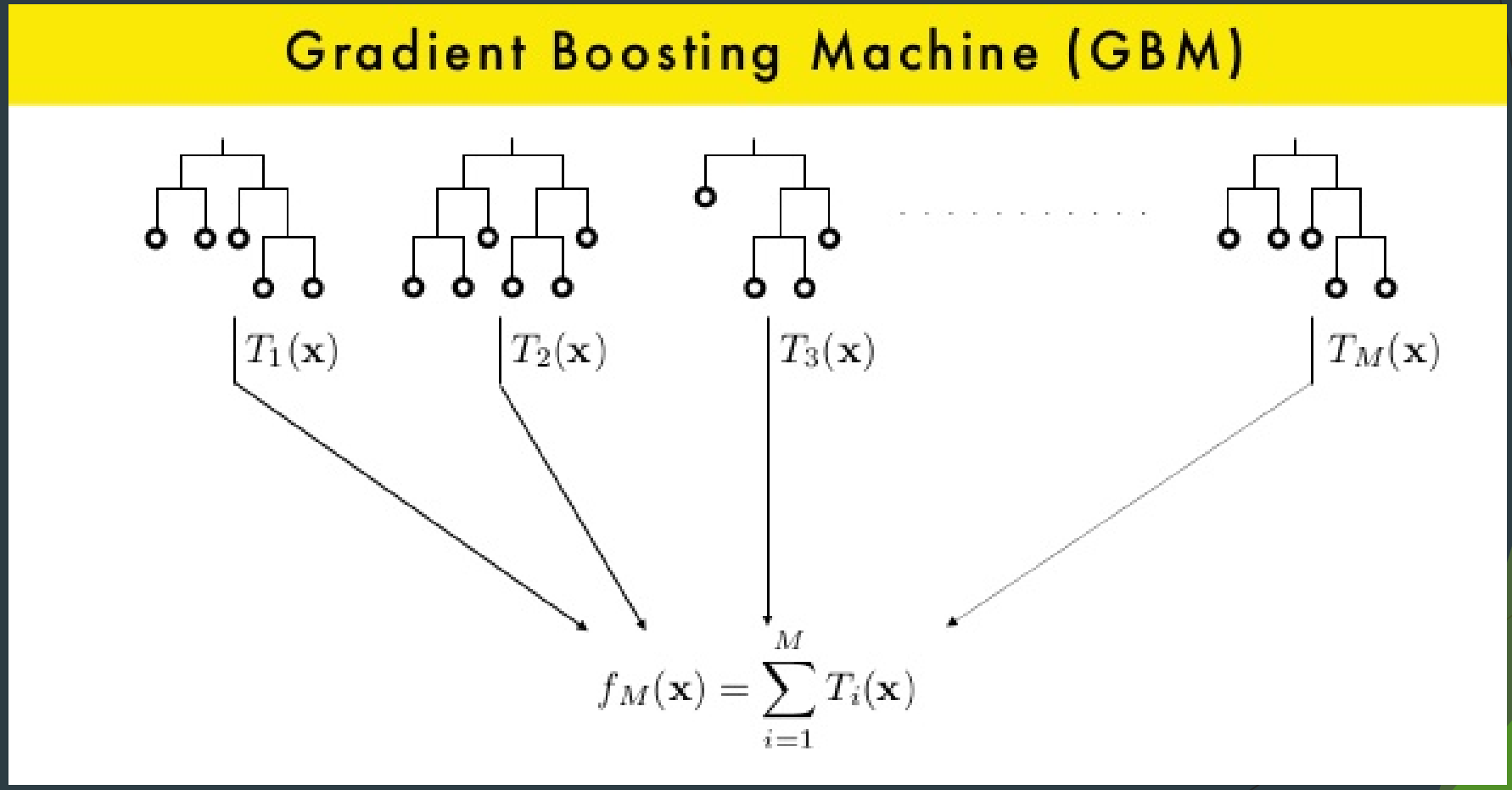
- Why GBM?
  - Neural Nets are awesome, but they require tons and tons of data to learn
  - Without tons of data, GBMs perform much much better
  - Very simple implementation
    - (we will go into tuning at a later meeting)

Model – Gradient Boosted Machine

- What is a GBM model?
  - GBMs are generally built on top of decision trees
  - GBMs combine multiple decision trees
    - Each tree can learn different features
    - Tree A is 90% confident, Tree B is 70% confident, chose Tree A
- Single Decision Tree →



Model – Gradient Boosted Machines



Preprocessing – Creating Inputs

- First, we will want to isolate the “close” column
- Then we will want to create groups of data points
  - Doing groups of 11, will rip off one to make the label
  - This leaves 1:10 - label:input dimensions
- Next we will normalize each group
  - We do this so we can pool together all stocks, even ones trading at different prices
  - We do this as opposed to creating a model for each stock

Preprocessing – Creating Labels

- Once we have the normalized groups of 11:
  - Compare most recent day to next most recent
  - 0^{th} index to 1^{st} index

```
for brick in grouped_data:
    if brick[0]*1.01 > brick[1]:
        labels.append(2)
    elif brick[0]*0.99 < brick[1]:
        labels.append(1)
    else:
        lablels.append(0)

    inputs.append(brick[1:])
```

Training/Fitting the Model

- Want to merge data from each stock
  - Shuffle the data as well
- Also want to do a test_train split
  - Create a validation set to see how well we perform on unseen data
- Fit the model
- Gauge performance
- Make predictions

Core GBM Implementation

```
from sklearn.ensemble import GradientBoostingClassifier
gbm = GradientBoostingClassifier( learning_rate=0.01, # [OK]
                                  n_estimators=3600, # [OK]
                                  min_samples_split=10, # [OK]
                                  min_samples_leaf=50, # [OK]
                                  max_depth=14, # [OK]
                                  max_features='sqrt',
                                  subsample=0.9, # .8 [OK]
                                  random_state=10) # [OK]

gbm.fit(X_train, y_train)
score = gbm.score(X_test, y_test)
print("Validation/Testing Score:", score)

preds = gbm.predict(y_train)
```

Final Considerations

- Improve accuracy and effectiveness of model
- 1) Improve the Data
  - 90% of the time, this is the most effective route
  - “Garbage in, garbage out”
  - Upsampling/Downsampling
    - Want balanced dataset
- 2) Improve the Model
  - Grid Searching
  - Great how to: https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/
  - XGBoost is another wildly popular and effective algorithm
    - Will cover this later on