Wall Street Watcher

Will be presented to you by the Data Survivors

Team

Ben Montague

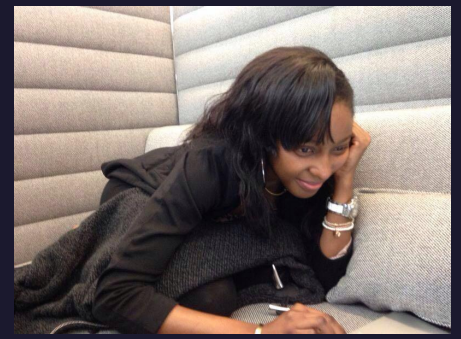
Data Survivor
Website Killer

D’Aurelia Harvey

Data Survivor
Program Killer

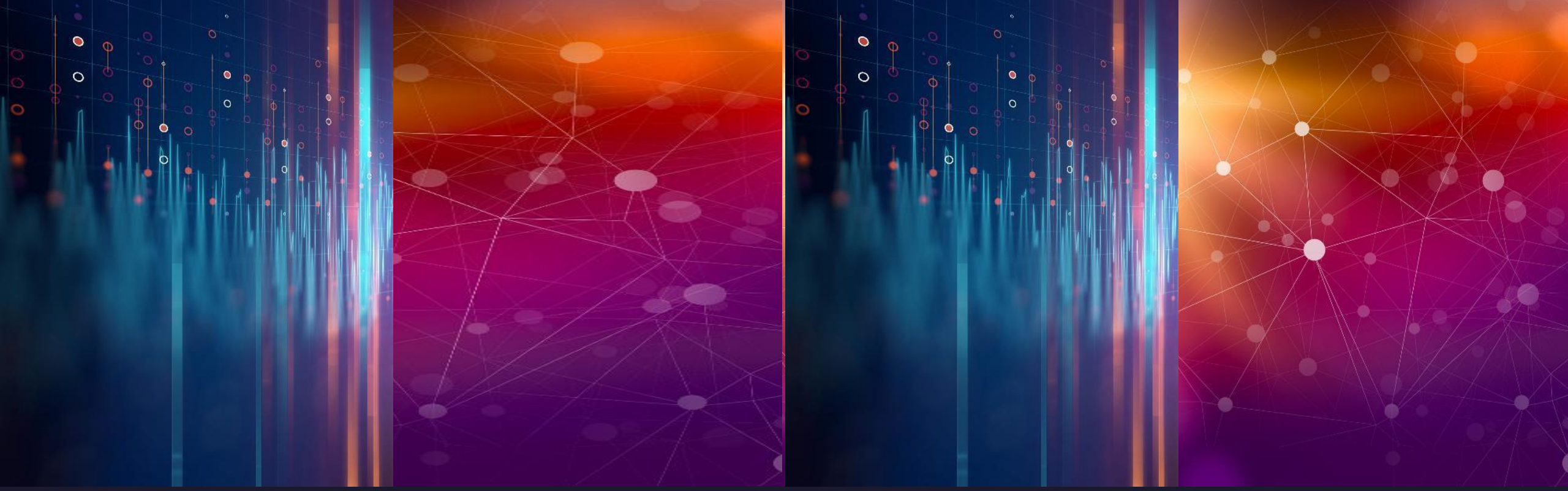
Natalie Cowart

Data Survivor
Homework
Killer



Dieneba Diaby

Data Survivor
Machine Killer



Introduction

- *We will be explaining the financial dashboard we created using Yahoo Finance API, to determine when to buy, sell, or hold your stocks.*

Overview

- Data Collection & Cleanup
- Analysis
- Summary

Data Collection & Cleanup

Web Scraping with BeautifulSoup

```python
import pandas as pd
import yfinance as yf
from splinter import Browser
from bs4 import BeautifulSoup
from webdriver_manager.chrome import ChromeDriverManager


executable_path = {'executable_path': ChromeDriverManager().install()}
browser = Browser('chrome', **executable_path, headless=True)

while True:
    url = 'https://stockanalysis.com/stocks/'
    browser.visit(url)
    element = browser.find_by_name('perpage').first
    element.select('10000')
    html = browser.html
    soup = BeautifulSoup(html,'html.parser')

    table = soup.find('table', {'class' : 'symbol-table index'})

    symbol = []
    company_name = []
    industry = []
    market_cap= []
    for row in table.find_all('tr')[1:]:
        symbol.append(row.find_all('td')[0].text)
        company_name.append(row.find_all('td')[1].text)
        industry.append(row.find_all('td')[2].text)
        market_cap.append(row.find_all('td')[3].text)

    stocks_df = pd.DataFrame(list(zip(symbol, company_name,industry,market_cap)),
            columns =['Symbol', 'Company','Industry','Market Cap'])

    stocks_df.to_csv('Output/stock_list.csv',index=False)

    print("Stock List Created")
```

- Used BeautifulSoup to scrape tables from StockAnalysis.com and Wikipedia.com
- Collected all tickers for the stock market as well as a list of S&P tickers
- Created 2 CSV files to pull into Tableau

Yahoo Finance in Python

```
history = pd.DataFrame(columns = ['Symbol','Date','Open','High','Low','Close','Volume','Dividend

for i in symbol:
    try:
        data = yf.Ticker(i).history(period = '1wk')
        df = pd.DataFrame(data)
        df['Date']=df.index
        df['Symbol'] = i
        history = history.append(df)
        print(f"----------{i} complete----------")
    except:
        pass

history.to_csv('Output/1wk_stock_history_test.csv',index=False)
```

- Used the Yahoo Finance library (yfinance) to collect ticker history
- Looped through the tickers collected on StockAnalysis.com
- Created CSV with data to pull into Tableau

Machine Learning

```
def get_clean_data (df, start_date, end_date):
    features = df.copy()
    features = features.drop(['formatted_date'], axis=1)
    #creating features as stated above
    features['volume'] = features['volume'].shift(1)
    features['SMA'] = features['adjclose'].rolling(window=20).mean().shift(1)
    features['Std_20'] = features['adjclose'].rolling(window=20).std().shift(1)
    features['Band_1'] = features['SMA'] - features['Std_20']
    features['Band_2'] = features['SMA'] + features['Std_20']
    features['ON_returns'] = features['close'] - features['open'].shift(-1)
    features['ON_returns'] = features['ON_returns'].shift(1)
    features['ON_returns_signal'] = np.where(features['ON_returns']<0, 'up', 'down')
    features['dist_from_mean'] = features['adjclose'].shift(1) - features['SMA']
```

```
comb_features = comb_features.drop('ON_returns', axis=1) #dropping original categorical columi
comb_features = comb_features.drop('close', axis=1) #not really needed this value since we ha
###Create return column to predict
comb_features['stock_move'] = np.where(comb_features['adjclose']-
                                       comb_features['adjclose'].shift(-1)<0, "Buy", "Sell")
features_clean = comb_features.dropna() #Dropping Nan values
features_clean = features_clean[:-1] #Drop last row which do not have any stock signal
features_clean.tail()
return features_clean
```

- used a classification approach for our Machine learning models to predict predict the market move to determine the best time to buy vs sell

Splitting and Scaling our Data

```
#convert stock_move to binary
features['stock_move'] = np.where(features['stock_move'] == 'Sell', 0, 1)
# Split Data
X = features.drop(['high', 'low', 'stock_move'], axis=1)
y = features['stock_move']


#test train split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


# Scale Our Data

#Scale the features
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

- Used train split to split our data. the test size was 30% while the train size was 70% of the data
- Scaled the data with scaler.fit

Setting up models for fitting and prediction

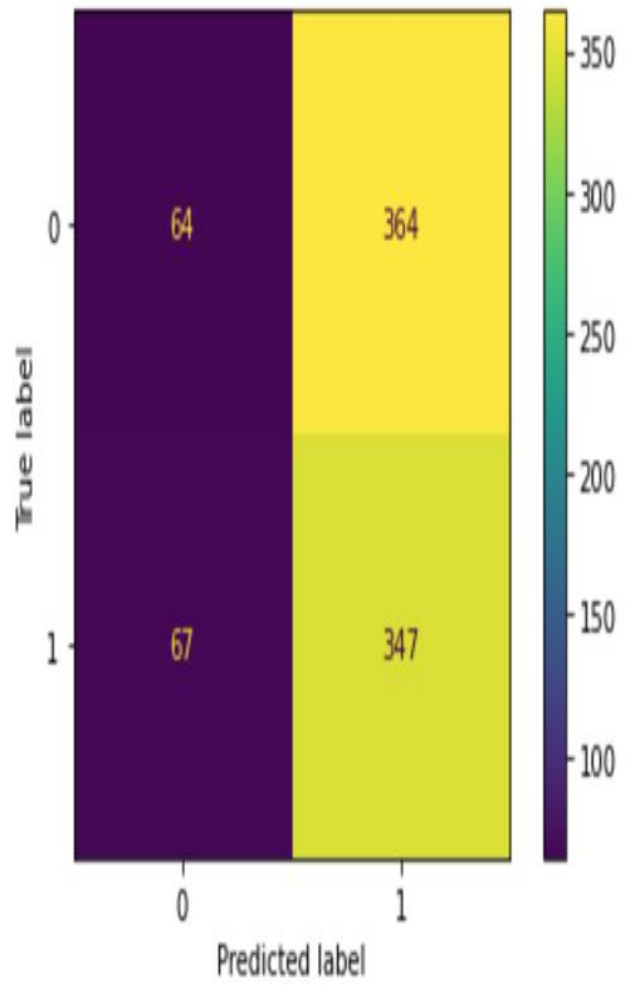
```
def get_models():
    models = list()
    models.append(LogisticRegression(solver='liblinear'))
    models.append(DecisionTreeClassifier())
    models.append(SVC(gamma='scale', probability=True))
    models.append(GaussianNB())
    models.append(KNeighborsClassifier())
    models.append(AdaBoostClassifier())
    models.append(BaggingClassifier(n_estimators=100))
    models.append(RandomForestClassifier(n_estimators=100))
    models.append(ExtraTreesClassifier(n_estimators=100))
    return models
# collect out of fold predictions form k-fold cross validatio
def get_out_of_fold_predictions(X, y, models):
    meta_X, meta_y = list(), list()
    # define split of data
    kfold = KFold(n_splits=10, shuffle=True)
    # enumerate splits
    for train_ix, test_ix in kfold.split(X):
        fold_yhats = list()
        # get data
        X_train, X_test = X[train_ix], X[test_ix]
        y_train, y_test = y.iloc[train_ix], y.iloc[test_ix]
        meta_y.extend(y_test)
        # fit and make predictions with each sub-model
        for model in models:
            model.fit(X_train, y_train)
            yhat = model.predict_proba(X_test)
            # store columns
            fold_yhats.append(yhat)
        # store fold yhats as columns
```

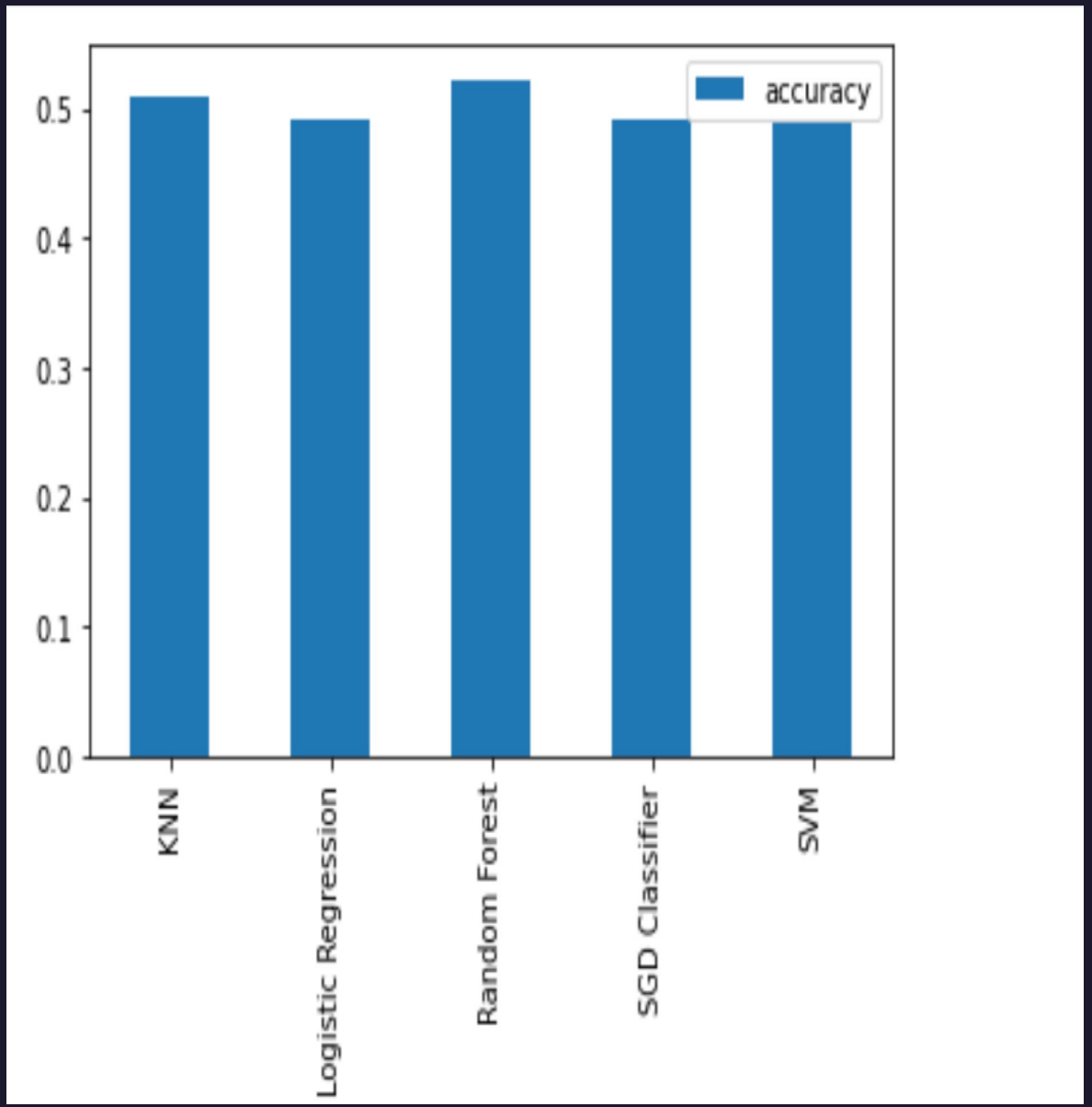
- Stocks were predicted using trained models that

  were compared with a chart (next slide) and a confusion matrix was also used to showcare the performance of our combined algorithms.

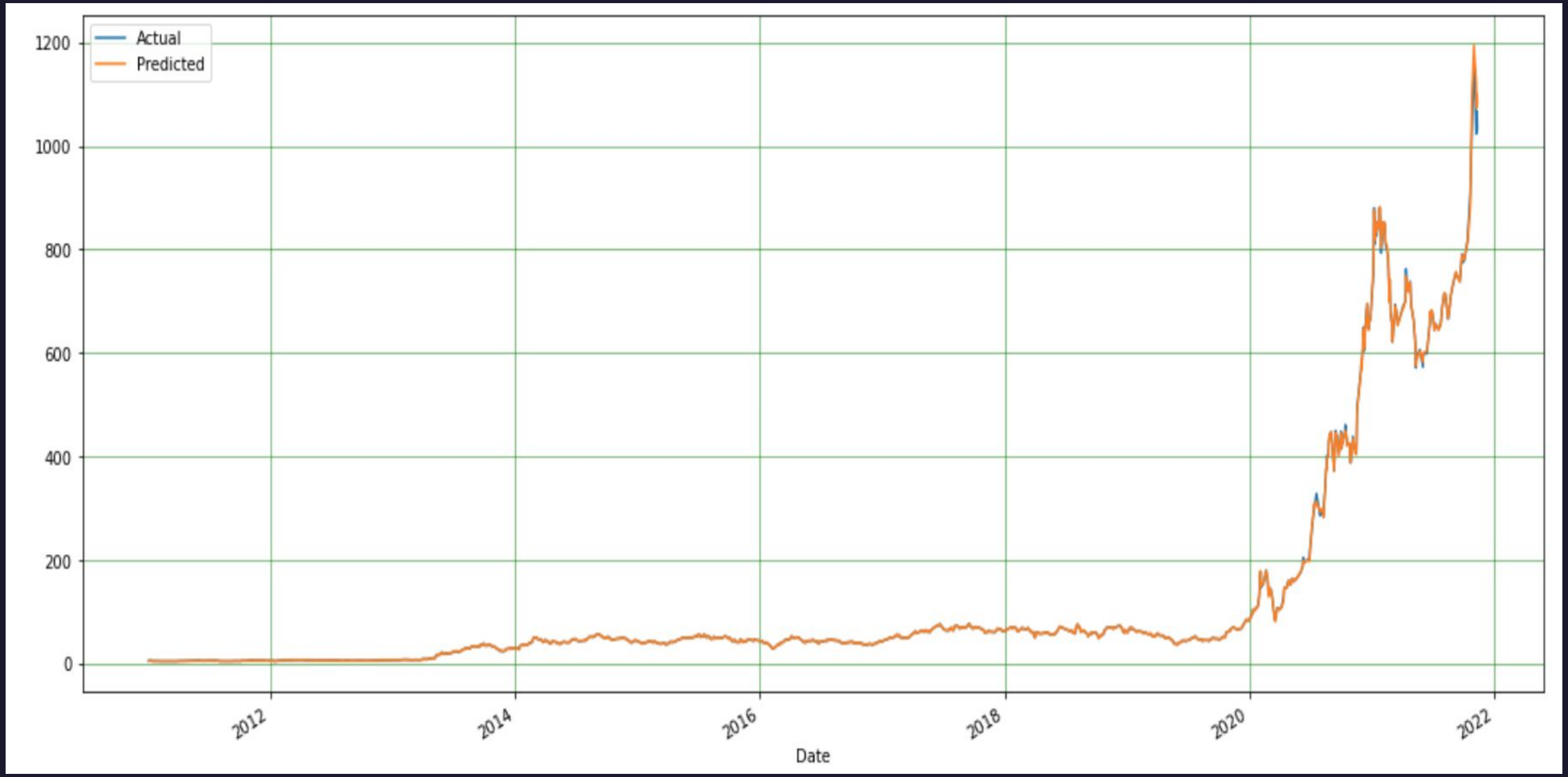
- Looped into the models to make the predictions then stack them to get a super learner algorithm

Accuracy score of the model is: 0.48812351543942994





Actual vs predicted charts



Analysis with tableau

Summary

The Wall Street Watcher dashboard allows entry level investors to review each stock ticker over time and make educated trades based on the stock's relative volatility. By using this dashboard, users are better equipped to take strong positions on stocks over time.

Thank You

- Questions and Answer Time

- References:
  - https://www.yahoofinanceapi.com/
  - https://stockanalysis.com/stocks/
  - https://en.wikipedia.org/wiki/S%26P_500