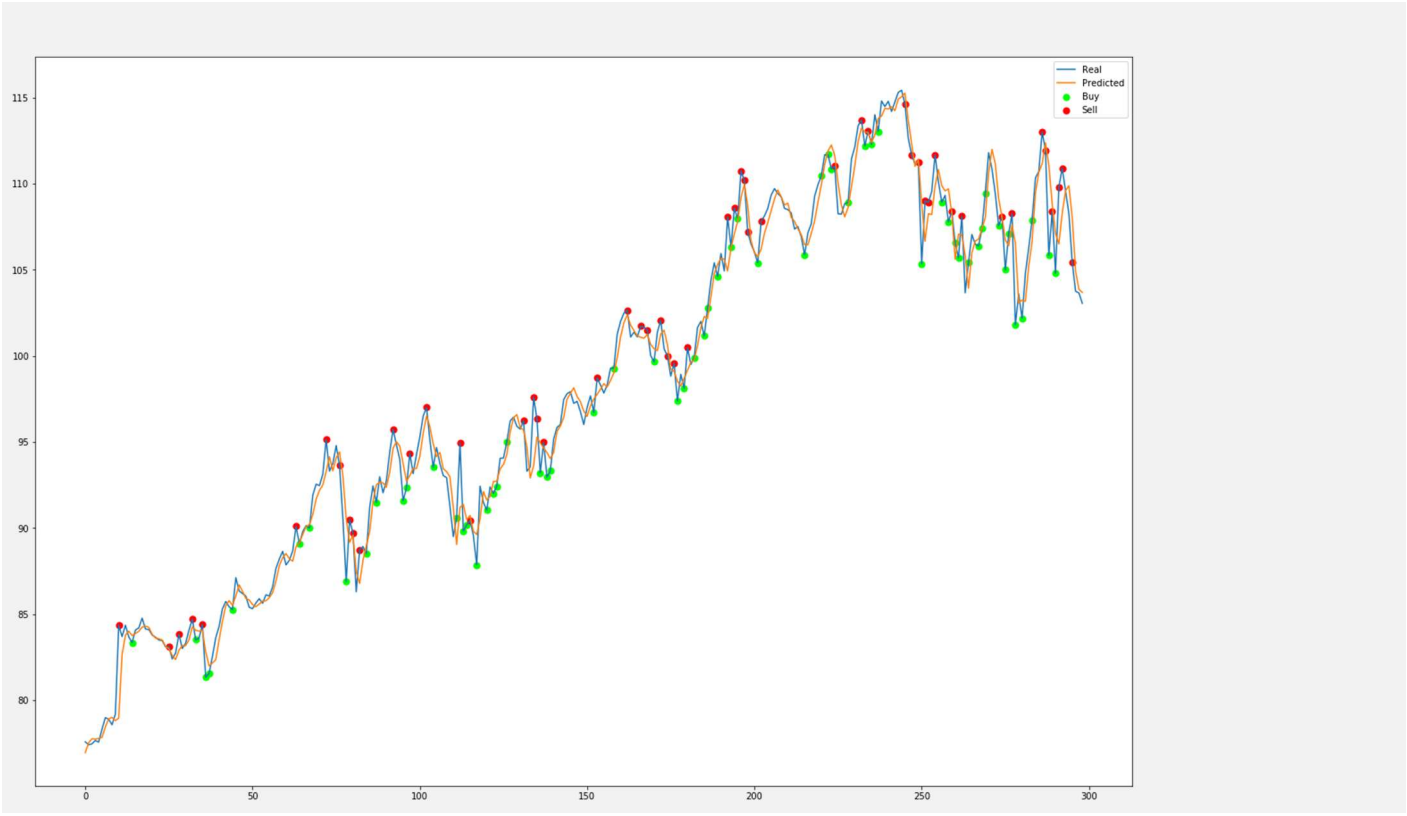
ディープラーニングを用いた株価予測

人間の投資家が成功できるなら、なぜ機械ができないのですか?

ヤクーブ・アーメド

従う

2019 年 10 月 12 日· 12 分読み取り



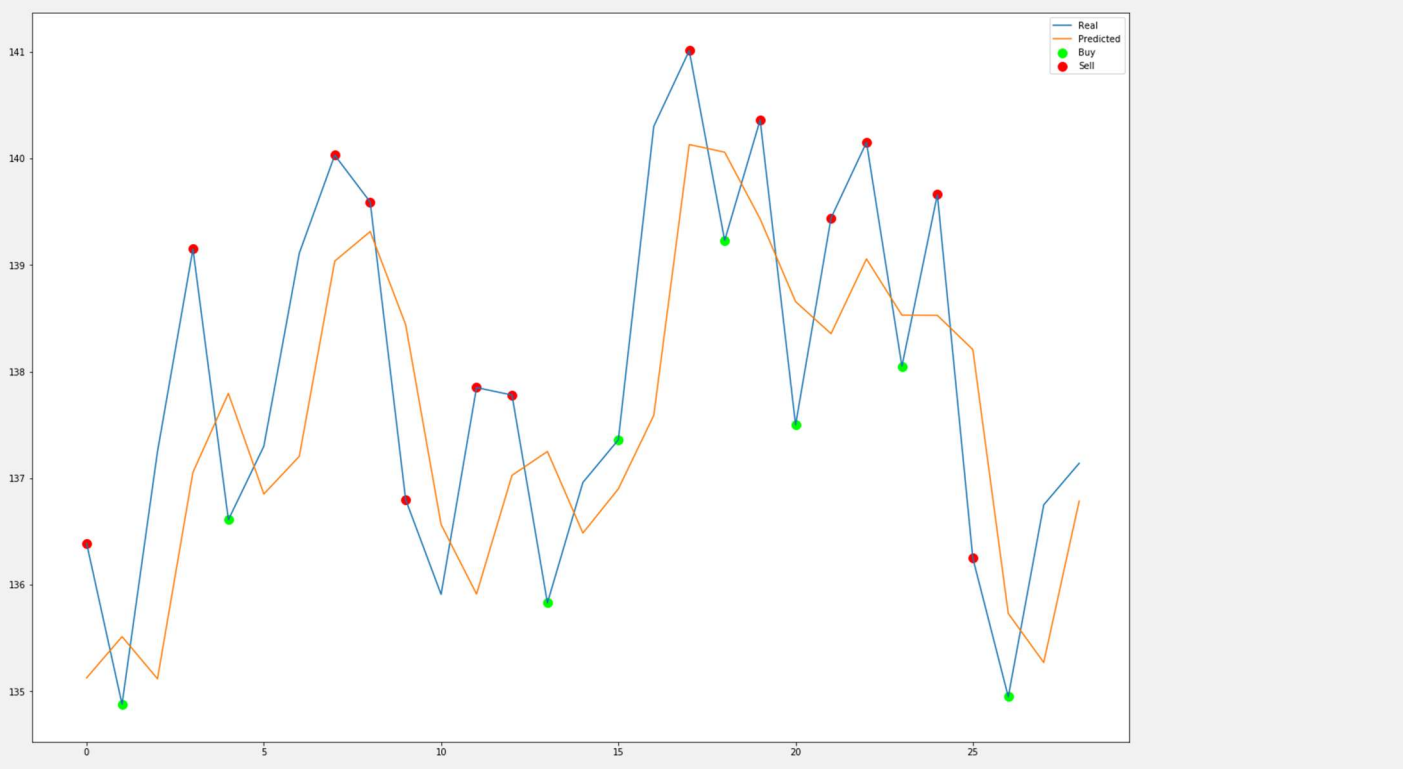
免責事項を追加したいのですが、このプロジェクトは完全に研究目的を意図しています!私はディープラーニングについて学んでいる学生で、プロジェクトは進行中の作業です、これほどナイーブなアルゴリズムにお金を入れないでください!

アルゴリズム取引は、株式市場とその周辺産業に革命をもたらしました。現在米国で起こっているすべての取引の 70%以上がボットによって処理されています[1]。電話に叫んで紙のシートを振る適した人々と満員の証券取引所の時代は過ぎ去りました.

これは、株式を取引するための独自のアルゴリズムを開発する方法を考えさせられました，または少なくとも正確にそれらを予測しようとします.

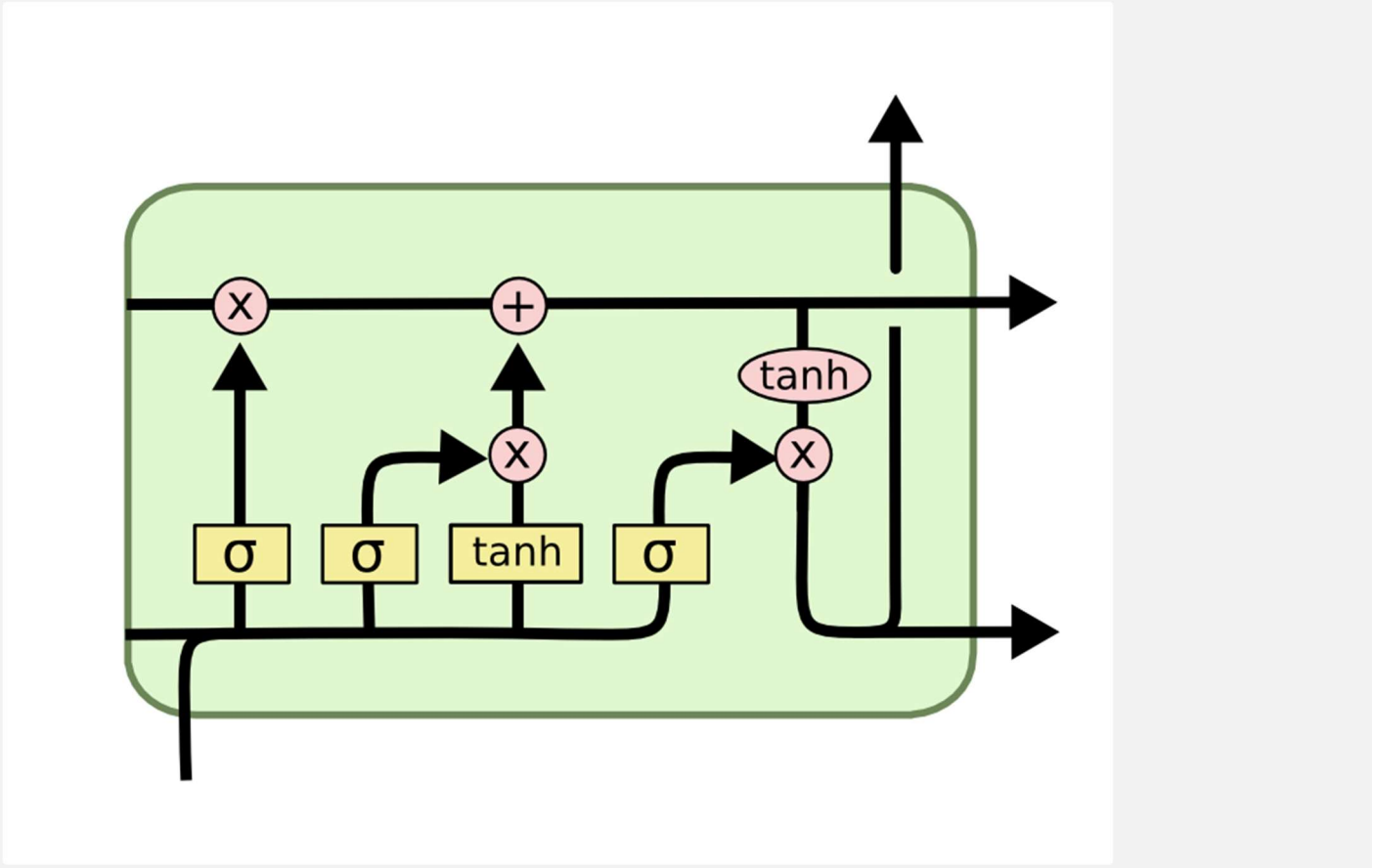
マシンは数字と素晴らしいです!

そして、結果は悪くありませんでした!



MSFT 株の取引アルゴリズム 9 月 — 2019 年 10 月

私は夏の間にニューラルネットワークと機械学習について多くのことを学びましたが、私が学んだ最新かつ適用可能な ML 技術の 1 つは LSTM セル[2]です。



LSTM セル。クレジット: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

長期記憶細胞は、より大きなニューラルネットワークのメモリを可能にするように設計されたミニニューラルネットワークのようなものです。これは、LSTM セル内の繰り返しノードを使用して実現されます。このノードには、1 の重みでエッジ ループバックが設定されており、feedfoward の反復処理のたびに、セルは前のステップの情報と以前のすべてのステップに保持できます。ループ接続の重みは 1 つなので、古いメモリは従来の RNN のように時間の経過とともに消えません。

LTSM と繰り返しニューラルネットワークは、過去を覚える能力のおかげで時系列データを扱うのが得意です。これらの繰り返しノードに古い状態の一部を格納することで、RNN と LSTM は、現在の情報と、ネットワークが 1、10、000 のステップ前に見た情報について推論することができます。さらに、私は LSTM セルの独自の実装を書く必要はありません。彼らはテンソルフローのケラスのデフォルトのレイヤーです。

だから私は私の計画を持っていた。LSTM と Keras を使用して株式市場を予測し、おそらくいくらかのお金を稼ぐことも。

コードに直接ジャンプしたい場合は、GitHub リポジトリ:)をチェックしてください。

データセット

株価履歴の良い点は、基本的にラベル付けされた事前に形成されたデータセットだということです。いくつかのグーグルの後、私は AlphaVantage と呼ばれるサービスを見つけました.彼らは過去 20

年間の NASDAQ 株の毎日の価格履歴を提供しました。これには、今日から 1999 年までさかのぼる、毎日のオープン、高、低、終値、取引量が含まれていました。さらに、サービス用の Python ラッパーが存在します。私はウェブサイトから私の無料の API キーを取得し、マイクロソフトの毎日の在庫履歴をダウンロードしました。

AlphaVantage の無料 API は 1 分間に 5 回の通話(および 1 日に最大 500 コール)しか許可しないため、データセットをダウンロードして CSV 形式で保存して、必要な頻度で使用できるようにしました。

過去 20 年以内に IPO 上場を果たしていた株式については、取引初日の取引の初日は、大量のボリュームのためにしばしば異常に見えました。この膨張した最大ボリューム値は、データを正規化するときにデータセット内の他のボリューム値がどのようにスケーリングされたかにも影響を与えたため、すべてのセットから最も古いデータポイントを削除することを選択しました。私はまた、モデルが取引がいつ起こったかについて何も知る必要がないので、日付を落とします、必要なのは十分に順序付けられた時系列データです。

また、使用する {*history_points*} の数、モデルがその予測を基にした在庫履歴の日数も追跡しています。したがって、history_points 50 に設定されている場合、モデルはトレーニングを行い、ちょうど翌日の予測を行うために過去 50 日間の在庫履歴を必要とします。

ここで、ネットワークの収束速度を向上させるために、データを正規化し、0 ~ 1 の範囲でスケーリングする必要があります[3] 。Sklearn はこれを行うことができる素晴らしい*前処理*ライブラリを持っています。

data_normalised 正規化された株価が含まれるようになりました。

ここで、モデルの使用に備えたデータセットを作成します。

ohlcv_histories リストは、ニューラルネットワークを訓練する際の x パラメータになります。リストの各値は、50 個のオープン、高、低、閉じ、ボリューム値を含む numpy 配列で、最も古いものから最新の値に向きます。これはスライス操作の中で見られるように、history_points パラメーターによって制御されます。 *history_points*

したがって、各 x 値に対して[i:i + history_points]株価(numpy スライスは[inclusive:exclusive])、y 値は単数[i + history_points]株価でなければならない。翌日の株価は非常に高い。

また、予測する価値を選択する必要があります。翌日の**オープン**値を予測することにしたので、データ内のすべての ohlcv 値の 0 番目の要素を取得する必要 *data_scaled* があります。

保持する y_normaliser と呼ばれる変数もあります。これは予測の最後に使用され、モデルは 0 から 1*normalised* の間の正規化された数値を吐き出し、データセットの正規化の逆を適用して実世界の値にスケールアップします。将来的には、これを使用して、モデルの実世界(正規化されていない)エラーを計算します。

その後、Keras でデータを操作するために、y 配列を *np.expand_dims()*を使用して 2 次元にします。そして最後に、私は後で結果をプロットするためのスケールされていない翌日のオープン値を保持します。

データを返す直前に、x の数が y の数をチェックします。

次の手順を実行して、csv ファイルを持つデータセットを取得できます。

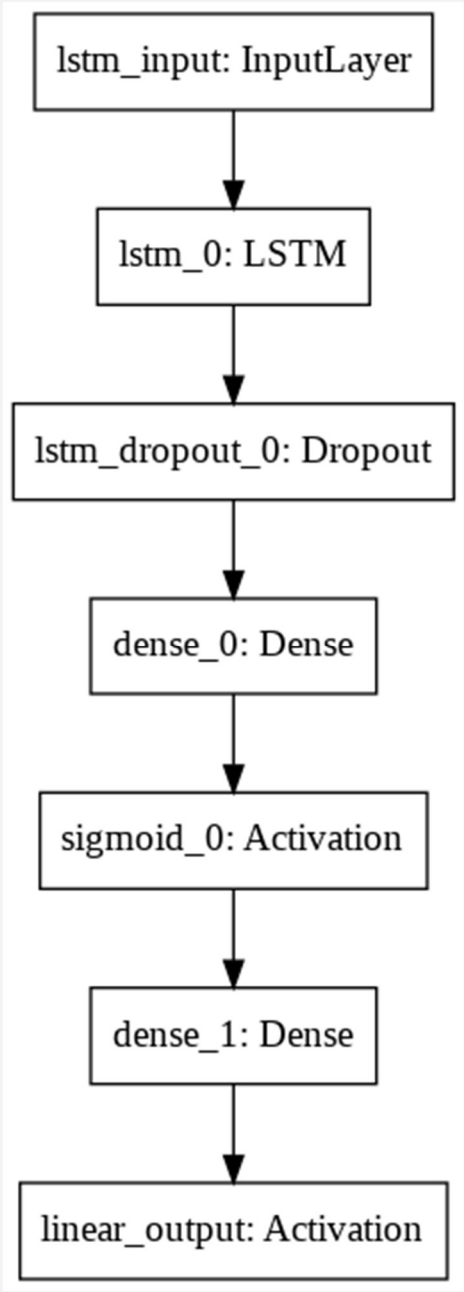
そして、それはそれです!データセットの準備ができました。

モデル

私はこのプロジェクトを始めたのはシーケンシャルな Keras コードの書き方しか知りませんが、私はより複雑なネットワーク構造を望み、最終的には各ブランチに異なるレイヤータイプを持つ 2 つの入力を特徴としていたので、機能的な API を学びました。

最初に思いついた最も基本的なモデルを見てみます。

これは私たちに次のようなモデルを与えます:



基本的なモデルアーキテクチャ

入力層は形状(history_points,5)を有し,各入力データ点は*[ohLCV×history_points]* × のような配列形状であるためである。このモデルには、最初のレイヤーに 50 個の LSTM セルがあり、オーバーフィットを防ぐためのドロップアウト層と、すべての LSTM データを結合するための密なレイヤーがあります。

このネットワークの重要な特徴は線形出力の活性化であり、モデルは最も高い重みを正確に調整できる。

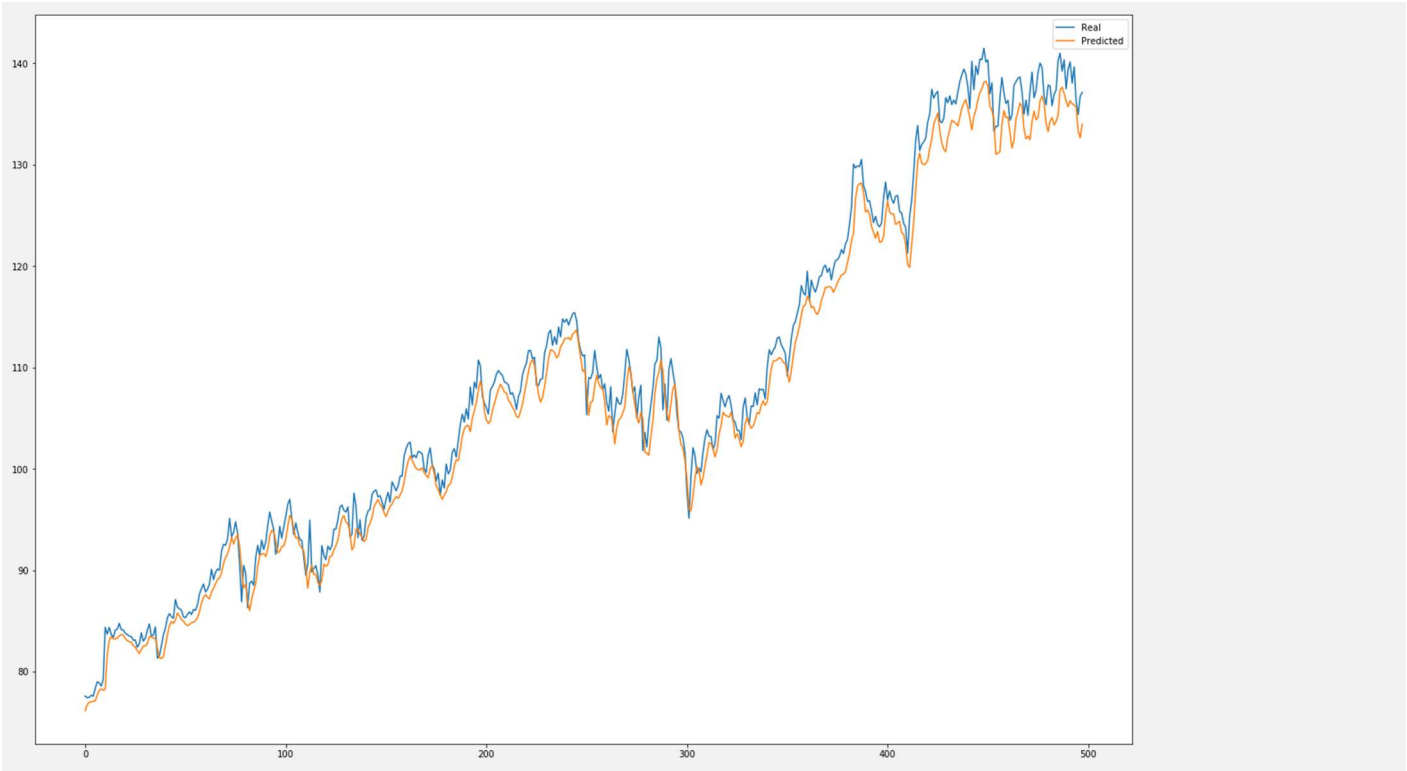
トレーニング

これが私がケラスを愛する理由です。文字通り 3 行のコードを使用すると、モデルがデータセットでどれだけうまく動作しているかを即座に知ることができます。私は 0.00029 の最終評価スコアを得ましたが、これは超低いようですが、これは*正規化された*データの平均二乗誤差であることを覚えておいてください。スケーリング後にこの値は大幅に上がるので、損失のための優れたメトリックではありません。

評価

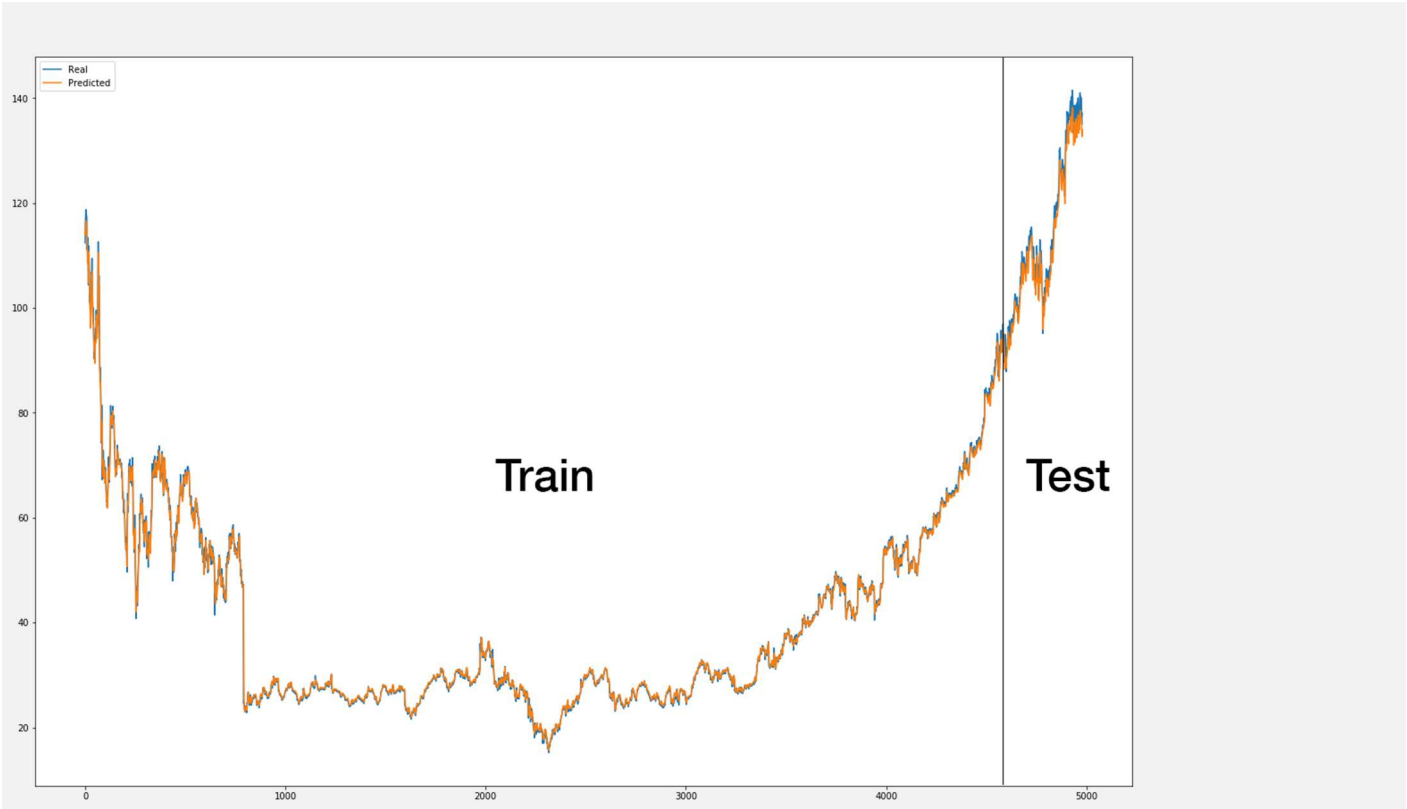
モデルをより正確に評価するために、実際の値と比較してテストセットをどのように予測するかを見てみましょう。まず予測値をスケールアップし、次に平均二乗誤差を計算しますが、次に、データセットに対する誤差をテストデータの「スプレッド」で割り、最大テストデータポイントから最小を引いた値を計算します。

これにより、調整後の平均平方誤差**が 7.17 になります**。それは良いですか?それは驚くべきことではありません、それは平均して予測されたラインが実際から 7%以上逸脱する意味です。グラフ上での見た目を見てみましょう。



過去 500 日間からの MSFT の実質および予測された毎日の開始株価、基本的なネットワーク

しかし、悪くない!予測値が実際の値よりも一貫して低い理由はわかりませんが、テストセットと列車セットの分割方法と関係があるのかもしれません。そして、完全なデータセット全体の結果:



1999 年以降の MSFT の実質および予測された毎日の開始株価、基本的なネットワーク

アルゴリズムがこのグラフ全体でどれほどうまく動作しているかを見分けるのは難しいですが、私たちが期待するように、列車セット全体のよりタイトなフィット感を見ることができます。

改善点

モデルをより複雑にし、データセットのサイズを大きくすることもできます。より複雑なモデルを作成しようとから始めましょう。

株式市場のアナリストが使用する一般的な指標は、*テクニカル指標[4]*です。テクニカル指標は株価の履歴に対して行われる数学の操作であり、伝統的に市場が変化する方向を特定するのに役立つ視覚的援助として使用されています。二次入力ブランチを通してこれらのテクニカル指標を受け入れるように、モデルを増強することができます。

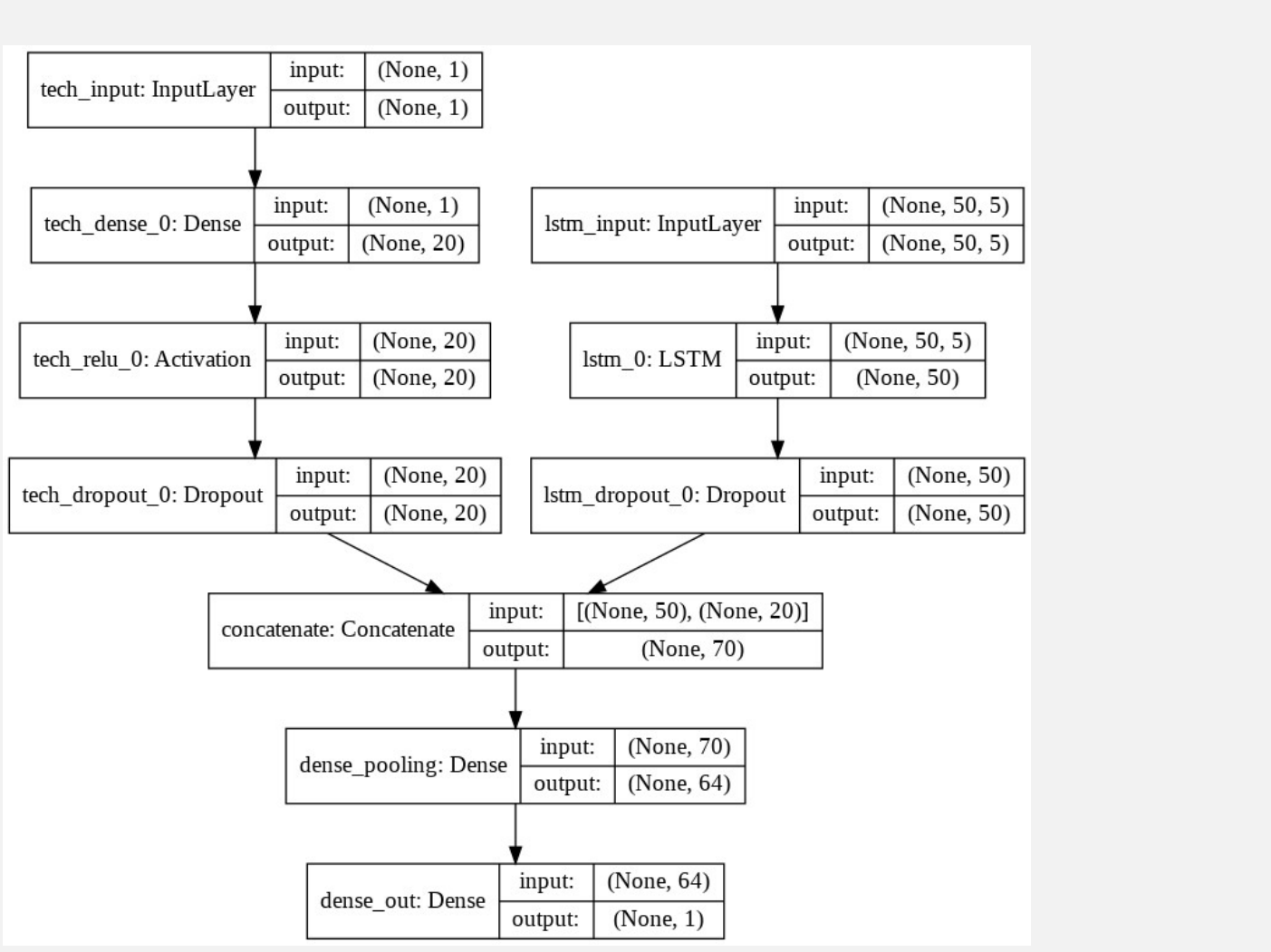
今のところ、単純な移動平均 SMA インジケータのみを私たちのネットワークへの余分な入力として使用しましょう。

株式の単純移動平均を計算するには、過去の n 時間ステップでの株式の*終値*の平均を取るだけ[5]。私たちはすでに価格履歴の固定タイムステップブロックを扱っているので、これは私たちにとって素晴らしい作品です。SMA をモデルに含めるには、データセット処理コードを変更する必要があります。

これは、ohlcv_histories と next_day_open_values 配列を定義した直後に発生します。50 価格のデータ ブロックごとにループし、3 番目の列の平均（終値）を計算し、その値を

technical_indicators リストに追加します。その後、リストはデータの残りの部分と同じ変換を行い、0 ~ 1 の値に収まるようにスケーリングされます。その後、返品明細書を変更して、テクニカル指標だけでなく、以前から返したものも返します。

次に、この新しいデータセットに一致するようにモデルを拡張します。古い LSTM 構造を使用できるようにしたいのですが、SMA テクニカルインディケータをミックスのどこかに組み込んでいます。SMA は時系列データではないため、LSTM を通過しないでください。代わりに、最終的な予測が行われる前にそれを混ぜる必要があります。それを究極の 64 節の高密度層に入力する必要があります。したがって、連結層と出力の 2 つの入力を持つモデルが必要になります。



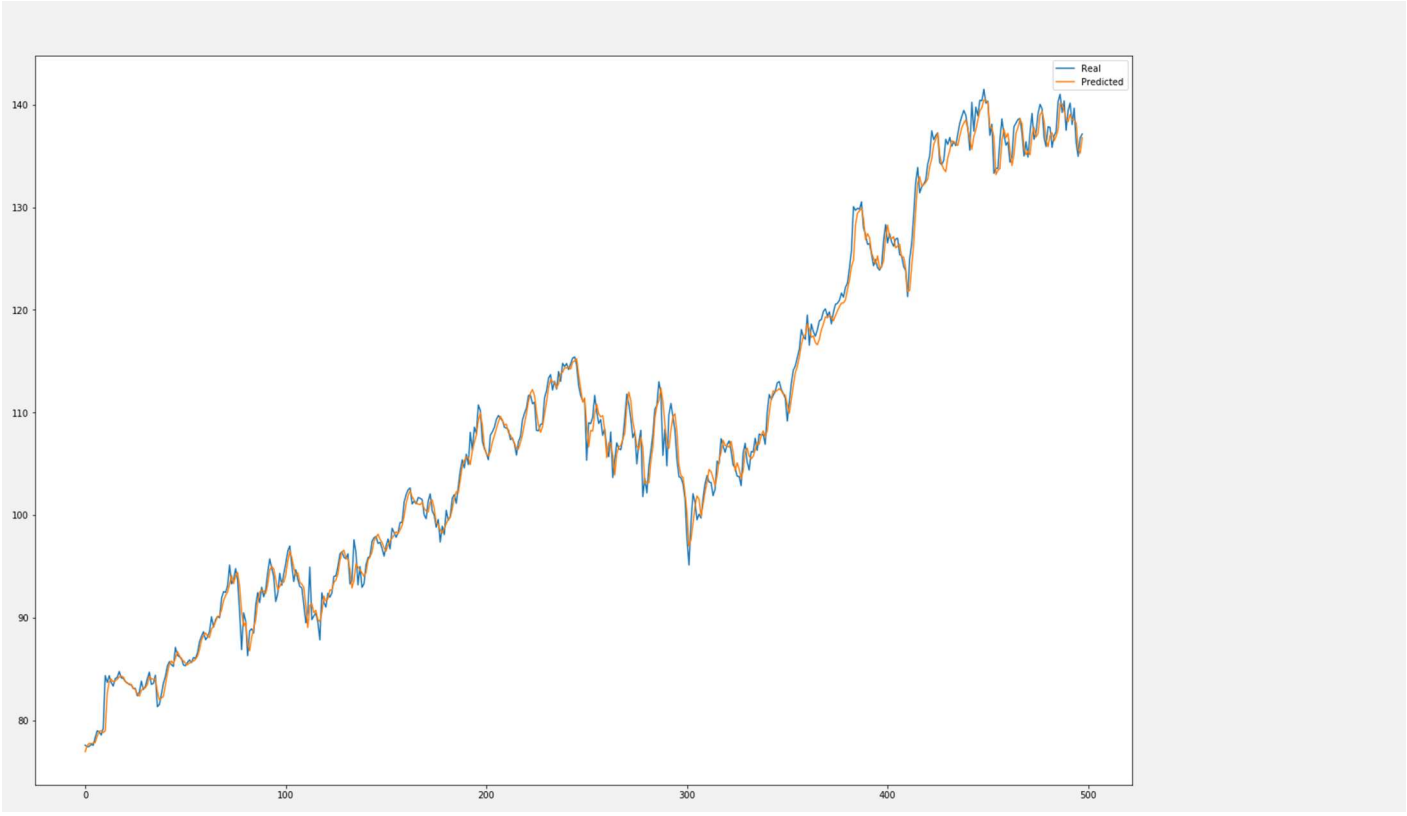
技術指標を使用する拡張後のモデルアーキテクチャ

tech_input レイヤーの入力図形として technical_indicators.shape[1] を使用した方法に注意してください。つまり、新しいテクニカル指標は、モデルを再コンパイルする際に適切に適合します。

評価コードも、このデータセットの変更に合わせて変更する必要があります。

モデルへの入力として[ohlcv,technical_indicators]のリストを渡します。この順序は、モデルの入力を定義した方法と一致します。

そして、我々は 2.25 の調整平均平方誤差を得る!はるかに低く、プロット時に予測がテストセットにかなり近いように見えます。



SMA を使用して、過去 500 日間からの MSFT の実質および予測された毎日の開始株価

このモデルは、一定の量で連続的にオフになっているという以前の問題に苦しんでいないようですが、同様に突然のジャンプをキャッチしないことに苦しんでいるように見えます。x 座標 120 の

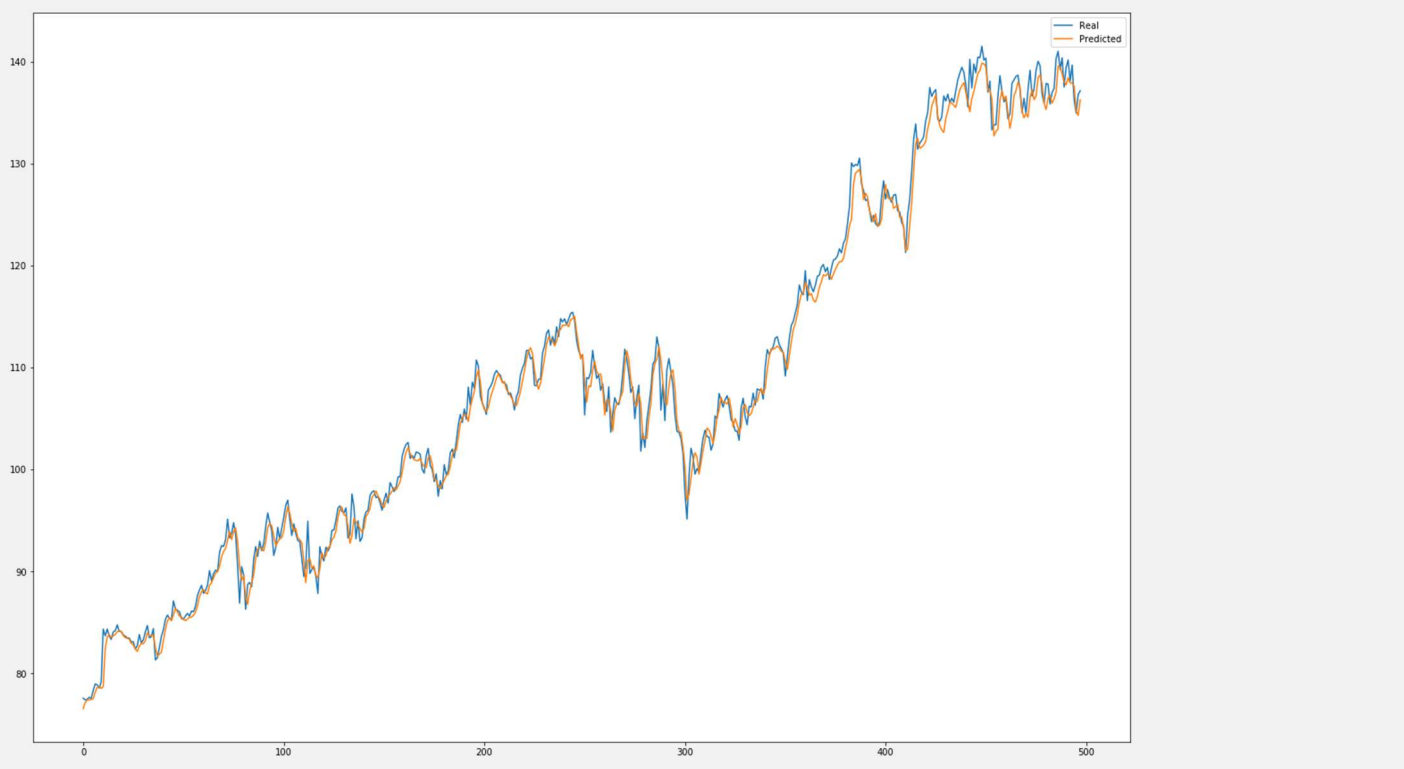
周りのように、実際の価格で大きなジャンプとディップが発生しますが、モデルはこれを効果的にキャプチャすることができません。しかし、それは良くなっています!そして、テクニカル指標が前進する可能性があるようです。

より高度なテクニカル指標を含めてみてみましょう: 移動平均収束発散.MACD は、12 周期 EMA[6]から 26 周期指数移動平均を差し引いて計算されます。EMA は次の式を使用して計算されます[7]:

$$
\begin{aligned}
&EMA = \text{Price}(t) \times k + EMA(y) \times (1 - k) \\
&\text{where:} \\
&t = \text{ today} \\
&y = \text{ yesterday} \\
&N = \text{ number of days in EMA} \\
&k = 2 \div (N + 1)
\end{aligned}
$$

テクニカル指標ループを更新して MACD 指標を含めるには、次の手順に従います。

SMA および MACD インディケータを使用すると、当社のモデルは 2.60 の調整平均平方誤差を達成します。SMA のみを使用する場合よりもわずかに高く、それはグラフに反映されます。



SMA と MACD を使用して、過去 500 日間からの MSFT の実質および予測された毎日の開始株価

SMA のみを使用する場合ほど正確に収まらないことがわかります。たぶん、私たちはモデルにより多くのデータを与えているので、それを理解するためにより多くのレイヤーが必要ですが、今のところは MACD テクニカルインディケータの使用を省略します。

また、より大きなデータセットを使用して実験することもできます。Microsoft の株式の株式予測に適用される手法がすべての株式に一般化できると仮定すれば、csv_to_dataset() 関数の結果をさまざまな株式履歴に組み合わせることができます。例として、AMZN、FB、GOOGL、MSFT、NFLX の株式履歴をトレーニングし、AAPL 株の結果をテストすることができます。

まず、あなたが使用したいすべての異なる株式のために、この記事の開始時に戻ってすべての方法から save_dataset メソッドを実行します。次に、データセットの作成方法は次のようになります。

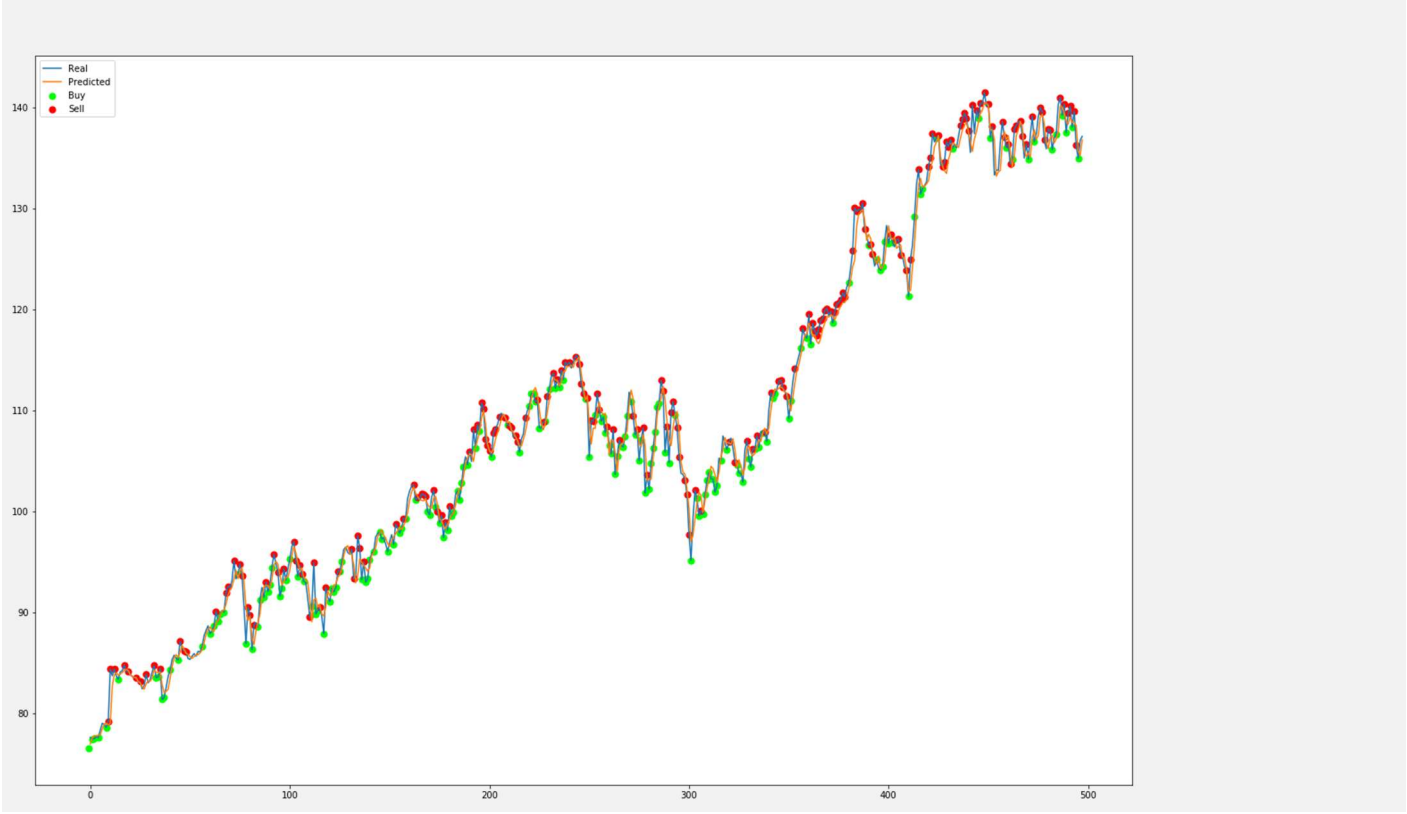
これは、基本的に、現在のディレクトリ内の各 csv ファイルについて、ファイル名が *test_set_name* されていない場合は、そのファイル名からデータセットを読み込み、データセット全体に追加します。次に、test_set_name csv ファイルを読み込み *test_set_name*、テスト データセットとして使用します。AMZN、NFLX、GOOGL、FB、MSFT の列車セットの株価を使用して、19854 年の列車サンプルを取得します。テストセットの AAPL ストックを使用して、4981 のテストサンプルを取得します。

SMA インジケータを使用して 50 エポックのためのこの新しい、より大きなデータセットでトレーニングした後、我々は 1 つの株式で訓練したときよりも高い **12.97** の調整された MSE 値を得る。株式予測が実際にはシンボル固有の場合です。異なる株式は、わずかに異なる方法で動作する可能性があります。ネットワークが異なる株式から学び、一般化することができたので、それらはすべて根本的に同様に振る舞うことは明らかですが、最も正確な予測では、その株式の歴史を訓練するだけであることが最善かもしれません。結局のところ、Microsoft データセットには、それを理解するのに十分なトレーニング データがあるようでした。

アルゴリズム

大丈夫っぽい株式予測アルゴリズムを武器に、私は株式の歴史を考えると、今日株式を売買することを決定するボットを作成する素朴な方法を考えました。本質的には、次の日の株式の開始価値を予測するだけで、しきい値を超えている場合は株式を購入します。別のしきい値を下回っている場合は、株式を売却します。この死んだ単純なアルゴリズムは、実際には、少なくとも視覚的には非常にうまく動作するように見えました。

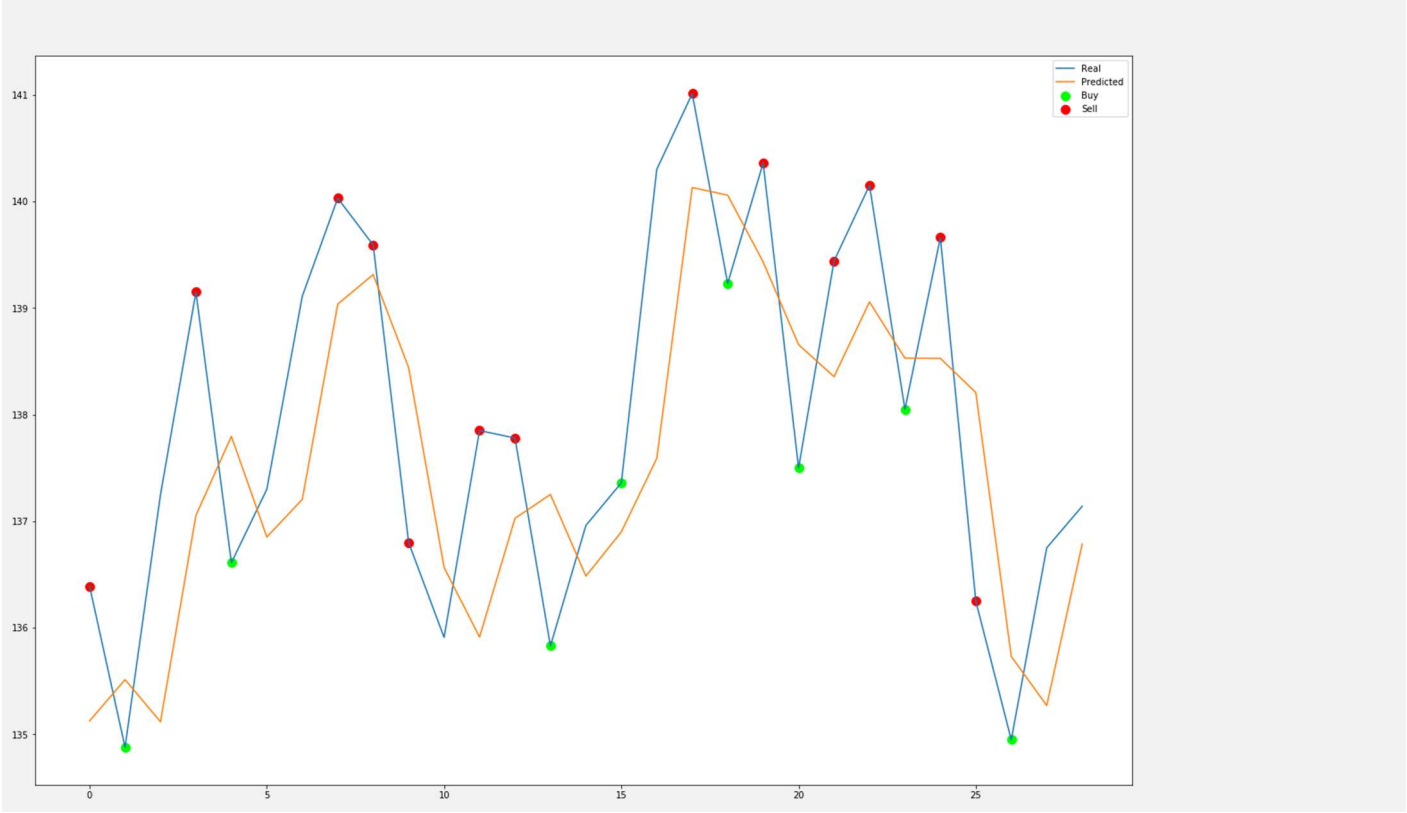
その後、取引をプロットします。



見栄えが良い!アルゴリズムは正しく低く買い、高く売っているようです。これは、すべてテストデータ、つまりネットワークがこれまでに見たことのないデータであることを覚えておいてください。このデータが実際に生きていたはずがない理由はありません、 そしてこれらの取引は実際に現実の!

これらの売買を考えると、各「買い」で 10 ドル相当の株式を購入し、各「売り」ですべての株式を売却すると、アルゴリズムは$38.47 を獲得していたでしょう。しかし、それは 500 日にわたってであることを覚えておいてください。アルゴリズムの収益を計算するコードはここにあります;

代わりに、同じ $10 購入金額としきい値レベル 0.2 を使用して、このアルゴリズムを 30 日にわたって試した場合、アルゴリズムは 1.55 ドルを獲得しているでしょう。しかし、それは何もないよりはましです!



過去 30 日間の MSFT 株式の取引アルゴリズム

結論

予測アルゴリズムの改善の余地はまだあると思います。すなわち、使用されるテクニカル指標、ハイパーパラメータ、買い/売りアルゴリズム/ハイパーパラメータ、モデルアーキテクチャ history_points、私は将来最適化したいと思います。

また、AlphaVantage で利用可能な各タイムステップに 1 つずつ、より多くの LSTM ブランチを持つことで、モデルにより多くのデータを提供することを考えてみたいと思います。

このプロジェクトの完全なコードは、私の GitHub で入手できます。問題ページにフィードバック/改善点を残してお気軽に!

私は、このプロジェクトをもう少し拡大し、数値データだけで株式を予測することで達成できることの限界を本当に押し広げるつもりです。私は私のブログでやっていることを最新の状態に保ちます！

参照

[1]: https://www.experfy.com/blog/the-future-of-algorithmic-trading

[2]: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[3]: https://jovianlin.io/why-is-normalization-important-in-neural-networks/

[4]: https://www.investopedia.com/terms/t/technicalindicator.asp

[5]: https://www.investopedia.com/terms/s/sma.asp

[6]: https://www.investopedia.com/terms/m/macd.asp

[7]: https://www.investopedia.com/ask/answers/122314/what-exponential-moving-average-ema-formula-and-how-ema-calculated.asp