UDACITY MACHINE LEARNING ENGINEERING NANODEGREE

Machine Learning for Stock Market Investing

Capstone Project / August 08 - 2019

Carlos Viejo

Definition

Project Overview

Stock market analysis prediction is one of the most challenging fields to estimate due to the multi-dimensionality and complexity of the inputs. There can be many factors involved in the calculations varying from multiple company/competitor interplays, variability, public perception, rational and irrational behaviour of traders and many more, the combination of all these aspects together make price volatile and challenging to predict and a perfect candidate for this project.

Data scientists and **Financial theorists** for over half a century have been employed to make sense of the stock market and its variations in order to increase return on investments, but it has been an overwhelmingly difficult challenge for humans to solve due to the complexity and massive amount of inputs.

Thanks to the advancements in machine learning algorithms and its applications, the field has evolved to combine multiple inputs like never before. State of the art approaches join information from the organization historical price trends, technical data, social media trends, news and classic techniques utilizing non-deterministic solutions that can "learn" what is going on from millions of data inputs, these techniques can utilize a mix of neuronal network models that rely on long short term memory units and natural language understanding. unearthing patterns and insights we didn't see before, using them to make unerringly accurate predictions.

By building a system that can accurately prognosticate stock market variations can be used to successfully predict a stock's future price, maximizing investor's gains.

The solution will use statistical figures to identify trends on the market, the datasets are obtained from

Quandl

https://www.quandl.com/ is a platform source for financial, economic, and alternative datasets, serving investment professionals

Yahoo! Finance

https://ca.finance.yahoo.com/ is a platform that provides It provides financial news, data and commentary including stock quotes, press releases, financial reports, and original content.

The data that we will use is **from J.P. Morgan Chase & Co.** we are planning to use a total of 10 years of data were 7 years will be used for training and 3 years for validation and testing, the data can be obtained directly with the following link. https://ca.finance.yahoo.com/quote/JPM?p=JPM

Problem Statement

Due to the multi-dimensionality and complexity of the problem the main objective of this capstone project is to provide a **machine learning model that utilizes state of the art algorithms, historical stock market data of a public trade company to predict future trends.**

We will start simple utilizing one single attribute and expand to increased complexity by incorporating more inputs, moving from simple algorithms like linear regression to advanced techniques like neuronal networks using LSTM cells.

To make machine learning predictions quantifiable, measurable and replicable, we will define the datasets utilized and provide access to them, select a well-known performance metric to quantify improvements over a baseline model and provide the codebase (hosted on GitHub) to replicate the results.

The goal is to create a stock market prediction model the task involved are the following:

1. **Select a dataset from the financial sites described**
2. **Train a time series regression that can predict the stock price for n days in the future**
3. **Submit the trained model for hosting in order to provide predictions**
4. **Invoke the endpoint for predictions, returning value of the stock the next day**

Metrics

After our machine learning model has been trained is quite important to assess how well the model it is able to capture patterns and predict, in order to diagnostic our model we will utilize evaluation metrics and residual diagnostics

The model will use:

MAE: mean absolute squared error, this metric is typically used in regression problems and works quite well as an indicator of performance

MAE = Average of All absolute errors

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |x_i - x|$$

Equation 1: Mean absolute error

Where:

- n = the number of errors,
- Σ = summation symbol (which means “add them all up”),
- |xi – x| = the absolute errors.

Analysis

Data Exploration

The **J.P. Morgan Chase & Co.** contains 10 years of historical data from **2009-01-02** to **2019-07-19** this information is aggregated in daily bases with a total amount of records of 2654 rows containing 7 columns of information in regards to the stock value, the datasets contain the following fields and variable types.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2654 entries, 0 to 2653
Data columns (total 7 columns):
Date         2654 non-null object
Open         2654 non-null float64
High         2654 non-null float64
Low          2654 non-null float64
Close        2654 non-null float64
Adj Close    2654 non-null float64
Volume       2654 non-null int64
dtypes: float64(5), int64(1), object(1)
memory usage: 145.2+ KB
```

Table 1: dataset information and variable types

Most of the variables contained in the datasets are floats numeric type. with the exception of the date this shouldn't be a concern because the date variable will not be considered to train the model, the selected target variable is Close price and we will be predicting the stock price one day into the future, quick statistical analysis of the variables shows most of the values within acceptable ranges for the model, pre-processing steps will be required for standardization of the data, to increase accuracy on the model and avoid impacts due to data variable ranges.

	Open	High	Low	Close	Adj Close	Volume
count	2,654.00	2,654.00	2,654.00	2,654.00	2,654.00	2,654.00
mean	62.13	62.72	61.53	62.13	55.72	26,770,442.20
std	25.80	25.91	25.69	25.79	27.44	21,861,503.66
min	15.37	17.29	14.96	15.90	12.46	3,680,000.00
25%	41.50	41.94	41.03	41.53	33.33	13,436,550.00
50%	56.62	57.00	56.14	56.67	49.27	19,304,750.00
75%	78.75	79.44	78.30	79.25	73.62	32,481,600.00
max	119.13	119.33	118.08	118.77	115.30	217,294,200.00

Table 2: dataset statistical descriptions

In regards to data integrity and validation, there are not NaNs values in the dataset or any other significant problem that can be observed

```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

Table 3: dataset integrity and validation on error

The final representation of the raw dataset and the description of each of the fields that will be used to generate more features and train a machine learning model can be observed in the following table and figure

	Date	Open	High	Low	Close	Adj Close	Volume
0	2009-01-02	31.19	31.64	30.47	31.35	24.56	32494900
1	2009-01-05	30.73	30.77	29.08	29.25	22.92	44069400
2	2009-01-06	29.79	30.42	29.51	29.88	23.41	44216300
3	2009-01-07	29.15	29.40	28.00	28.09	22.01	42156500
4	2009-01-08	27.90	27.95	26.86	27.22	21.33	52075800

Table 4: the dataset, variable visualization by row

The columns **Open** and **Close** represent the starting and final price at which the stock is traded on a particular day.
High, **Low** represent the maximum and minimum price of the share for the day.
Volume is the number of shares bought or sold in the day and Turnover (Lacs) is the turnover of the particular company on a given date. We will consider **Close** price as the target variable

Data Visualization

In order to better understand the distributions of the data before building any type of model we will be using a mix of **scatter plots**, **histograms** and **time-series** plots to extract potentially valuable insights

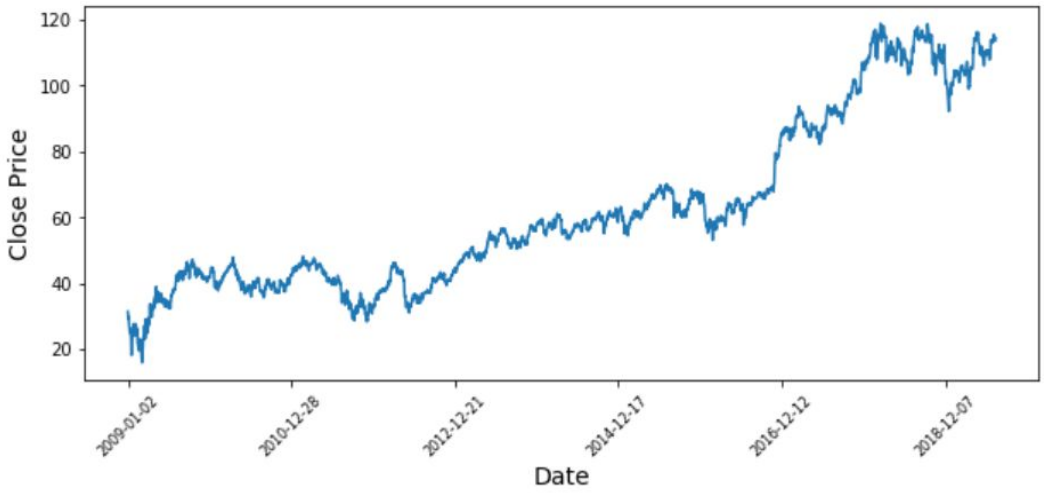

Figure 1: time-series representation of the data from 2009

From the analysis of figure one it's clear that there is trending upwards trend that can influence the model training, it's extremely recommended that the data is scaled and normalized in a certain way before passing the variables as an input to the model, we also can observe there is some type of seasonality of the data that can influence the training, for this reason, is recommended to split the datasets in a careful way.

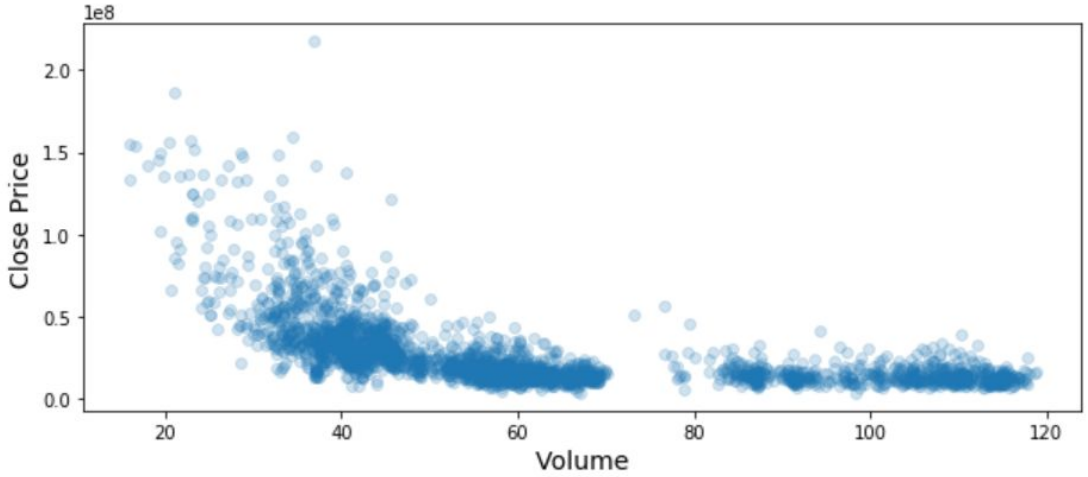

Figure 2: Correlation between the volume of transactions and close price

Figure 2 shows the correlation between the close price of the stockpile and the volume of the stock movement over the same period of time, this correlation seems to be a negative

non-linear correlation that can be exploited by the model to increase its accuracy, it's recommended to create some of the features in regards to volume.

The volume feature follows a log-normal distribution some type of standardization is also recommended in order to avoid impacts due tue it's raw distribution potentially skewing the results of the predictions, figure 3 display the histograms of distributions for the value feature

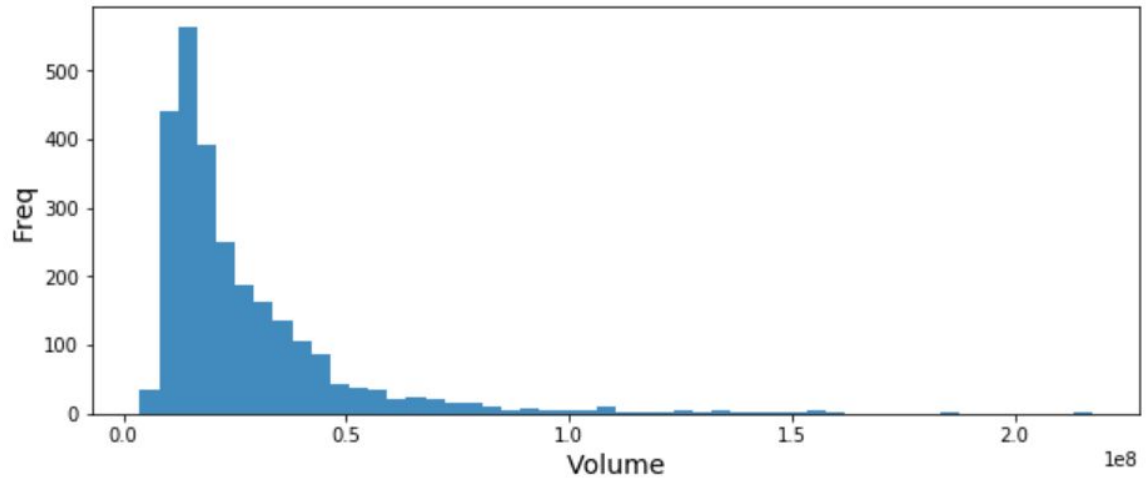

Figure 3: Histogram analysis and distribution of stock volume

We can summarize the previous analysis in the following 3 steps that need to be implemented before moving to train with the selected dataset:

1. Normalize all of the inputs before training the model
2. Utilize a more sophisticated CV strategy that can account for seasonality
3. Build some of the features around the stock volume variable

Algorithms and Techniques

Our **machine learning model** will be built utilizing neuronal networks, the network units will be Long Short Term Memory Cells known as LSTM, the model will be built utilizing

Keras and tensorflow as a backend, the architecture of the final model will be defined upon multiple iterations and tests in a loop with hyperparameters modifications in order to improve model performance more descriptions on the implementation section.

Our final architecture utilizes LSTM because they widely use for sequence prediction problems and proven to be extremely effective in this field. The reason they work so well is that LSTM is able to store past information that is important and forget the information that is not. LSTM has three gates:

- The input gate: The input gate adds information to the cell state
- The forget gate: It removes the information that is no longer required by the model
- The output gate: Output Gate at LSTM selects the information to be shown as output

This makes LSTM a perfect fit for time-series predictions in comparison to other techniques. To increase the model capabilities to learn and be more accurate at prediction time, we will use a multiple time series cross-validation strategy splitting the datasets into multiple groups and sets based on time, to achieve this results we have to build a custom time series generator that works in the following way:

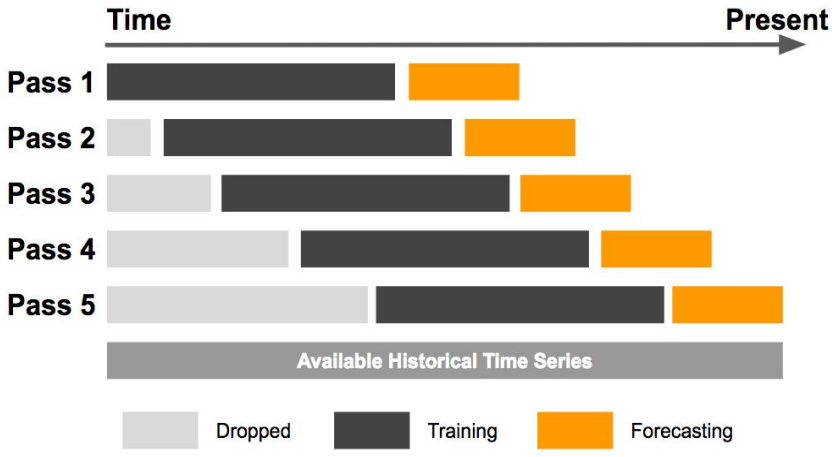

Figure 4: Cross-validation strategy, multiple folds

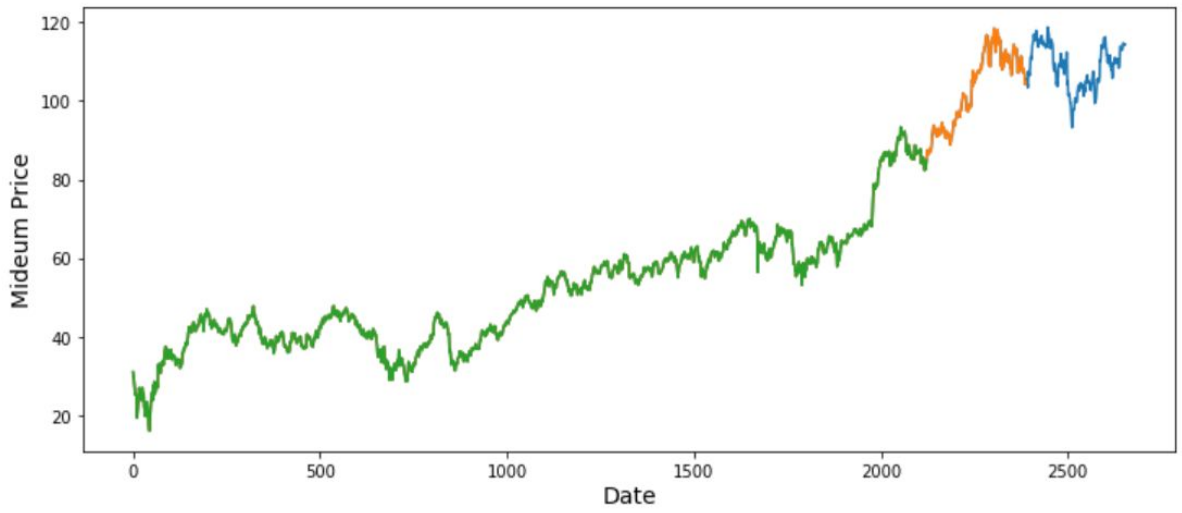

Figure 5: Cross-validation strategy, train, validation and test dataset

The model has the following parameters that can be tuned to optimize the predictions:

Training parameters:

- Train length: Number of epochs, steps per epoch
- Solver optimization function: the mathematical model used to optimize the model
- Learning rate: How fast the model can learn
- Learning rate decay: Dynamic learning rate once the model learning stalls
- Drop out amount: % of the cells being randomly dropped
- Regularization: beta 1 and beta 2

Neuronal network architecture

- Number of layers: The number of layers in the model
- Number of units: How many cells will be used in each layer
- Layer types (LSTM, Bi-directional, Directional, Dropout, Dense)

Generator parameters

- Steps back: how much information to select back tom time t
- Steps to predict in the future: how many steps in the future will be predicted
- Sample steps: Every how many points a sample is selected
- Batch size: How many points to look at once every single training step

- Start and end indexes: datasets indexes to sample data from

During training both of the datasets are loaded in the generator in order to train and validate the model performance at the same time, the training is completed utilizing the Adam optimizer with learning rate decay, we use 30 days back of data and 1 day into the future as target to build multiple batches to train the model.

Benchmark

To create a potential benchmark of the performance of the model we decide to utilize the persistence algorithm it uses the value at the previous time step (t-1) to predict the expected outcome at the next time step (t+1).

- Simple: A method that requires little or no training or intelligence.
- Fast: A method that is fast to implement and computationally trivial to make a prediction.
- Repeatable: A method that is deterministic, meaning that it produces an expected output given the same input.

In order to compute the benchmark the following function was developed and tested against the validation generator

```
def evaluate_naive_method():
    batch_maes = []
    for step in range(val_steps):
        samples, targets = (val_generator[step])
        preds = samples[:, -1, 0]
        mae = np.mean(np.abs(preds - targets))
        batch_maes.append(mae)
    return (np.mean(batch_maes))

mean_absolute_error = evaluate_naive_method()
print(mean_absolute_error)
```

Figure 6: Code snippet for implementation of the evaluate_naive_method

The results of this simple benchmark are quite good and is well know that utilizing the previous day stock price value can

The current MAE of the benchmark model is close to 0.01202, this value is calculated using the results of the normalized datasets

MAE One day in the future = 0.01202

In cases that we want to use the same model to predict two days in the future the performance start diminishing, this is where we believe out more sophisticated implementation will demonstrate the superiority against the benchmark mode.

MAE Two days in the future = 0.01650

MAE Four days in the future = 0.02501

MAE Seven days in the future = 0.03271

Methodology

Data Preprocessing

The preprocessing steps are completed to prepare the data for the machine learning model training we can classify this steps in to major classes:

Data standardization: this step is looking to standardize the numeric fields utilizing data normalization techniques this ensures that the model does not suffer from the wide range of values presented in the raw data by standardizing the datasets we improve the performance of the training algorithms

Feature Engineering: this process utilize domain expertise in the field and insights extracted from the exploratory data analysis phase to build new features to enhance the model performance this performance can be measured as reduction in the model MAE.

All the rpe-processing steps are completed in the following order:

Feature Engineering Steps:

- Differences between selected time steps, in this case used [5,10,20,40,80] steps
- Moving Average for selected time steps, in this case used [5,10,20,40,80] steps
- Calculate multiple ratios between features

- Remove errors created by time offsets

Data Standardization Steps:

- Normalize data utilizing Min-Max scaler
- Convert data to np array for easy processing on data generator
- Generating datasets using the custom time-series generator

	160	161	162	163	164
Date	2009-08-21	2009-08-24	2009-08-25	2009-08-26	2009-08-27
Open	0.16	0.17	0.17	0.17	0.16
High	0.16	0.17	0.17	0.16	0.16
Low	0.16	0.17	0.17	0.17	0.16
Close	0.17	0.16	0.17	0.17	0.17
Adj Close	0.12	0.12	0.12	0.12	0.12
Volume	0.18	0.17	0.15	0.13	0.12
VDiff_5	0.43	0.42	0.44	0.43	0.42
VDiff_10	0.44	0.49	0.45	0.47	0.47
VDiff_20	0.54	0.55	0.54	0.52	0.49
VDiff_40	0.49	0.52	0.53	0.51	0.51
VDiff_80	0.44	0.41	0.47	0.37	0.46
VDiff_160	0.62	0.57	0.56	0.55	0.51
MA_5	0.15	0.15	0.16	0.16	0.16
MA_10	0.14	0.14	0.15	0.15	0.15
MA_20	0.12	0.12	0.12	0.13	0.13
MA_40	0.07	0.07	0.08	0.08	0.08
MA_80	0.05	0.05	0.06	0.06	0.06
MA_160	0.00	0.00	0.00	0.00	0.00
Open/Close	0.41	0.68	0.51	0.56	0.48

Figure 6: Data preview after standardization and feature engineering steps completed

During inference for the model or prediction time the previous steps need to be applied to the dataset, if this is avoid unexpected results can manifest

Implementation

The model was implemented (trained and deployed) utilizing **Amazon SageMaker** full stack of AWS products that collaborate with SageMaker, S3 Storage, Model Creation, Training, Validation and Deployment to construct an end to end solution.

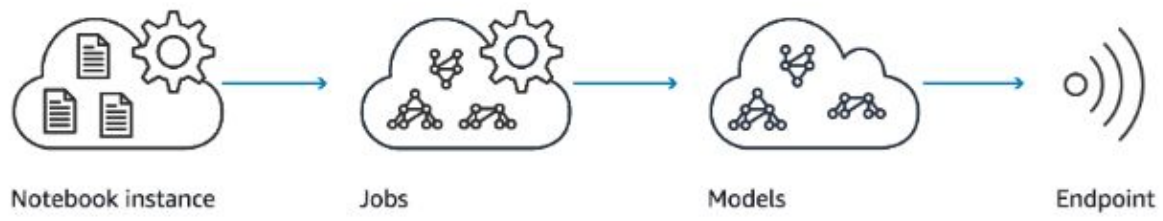

Figure 7: High level description of the end to end implementation

The majority of the development was completed utilizing **Jupyter Lab**, the source control system utilized was **Github**, the SRC files necessary for the training on the model in the cloud were created with **Jupyter Lab**

The JupyterLab development was divided in the following steps:

- Importing datasets for the model: Data is imported directly into the notebook from amazon S3, once the data is downloaded is loaded into a Pandas DataFrame
- Exploratory data analysis and visualizations
- Feature engineering
- Data standardization and validation
- Cross validation, training validation and test datasets

- Model benchmark and metric analysis
- Model architecture and parameter selection
- Model train and validation, monitoring of training and validations loss
- Visualization and metric analysis
- If model is not improving return to feature engineering step to enhance
- Train the model in the cloud and deploy as endpoint

The model was constructed utilizing **Keras** with **TensorFlow** as a Backend, the final solutions architecture has:

Input Layer: Dimensions (None, 16)

Layer 1: 32 LSTM Cells / Recurrent DropOut 0.01 / Drop Out 0.01

Layer 2: 16 LSTM Cells / Recurrent DropOut 0.01 / Drop Out 0.01

Layer 3: 8 Dense Fully Connected

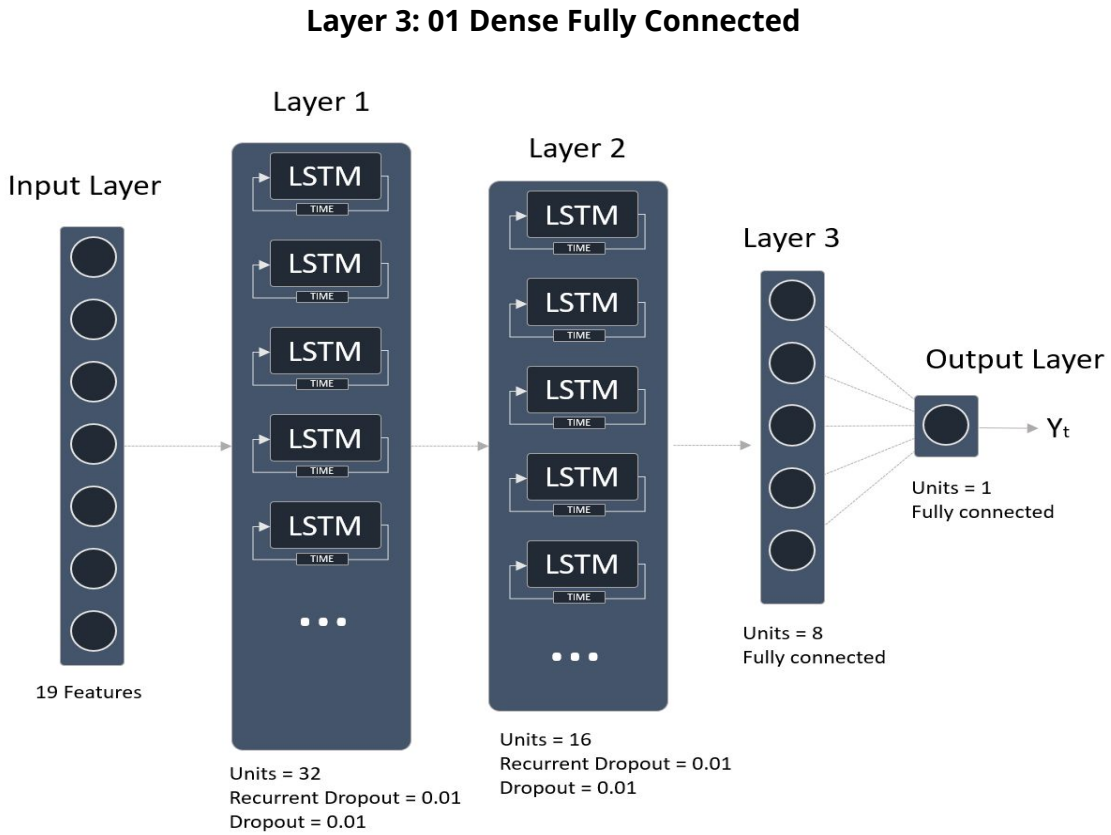

Figure 8: High level description of the model architecture

Refinement

After multiple iterations the model seems to get closer to the benchmark only in cases that we are trying to predict the stock market value for more than one day into the future, in cases that we are trying to beat the previous days prediction is quite challenging, the final implementation was completed in **Keras** utilizing a **TensorFlow** backend, the model has an overall denormalized train MAE of 0.018 predicting 4 days in the future vs 0.025 on the benchmark model

The final model summary is described in the following table

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_5 (LSTM)                (None, None, 32)          6784
_________________________________________________________________
lstm_6 (LSTM)                (None, 16)                3136
_________________________________________________________________
dense_5 (Dense)              (None, 16)                272
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 17
=================================================================
Total params: 10,209
Trainable params: 10,209
Non-trainable params: 0
_________________________________________________________________
```

Figure 9: Model output summary from Keras

There were three major key modifications to reach this final stage of the model from its original naive architecture (benchmark model utilizing utilize the persistence algorithm) this baseline mode was improved utilizing the following techniques

- More sophisticated architecture designed to operate with time-series and consider time, in this case the model was enhanced with **LSTM cells**
- The LSTM cells were upgraded from a directional type to **Bi-Directional LSTM** enhancing the amount of signal that can extract from the sequences

- Hyperparameter tuning and iterations with the architecture

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_23 (Bidirectio (None, None, 48)          8640
_________________________________________________________________
bidirectional_24 (Bidirectio (None, 16)                3648
_________________________________________________________________
dense_31 (Dense)             (None, 4)                 68
_________________________________________________________________
dense_32 (Dense)             (None, 1)                 5
=================================================================
Total params: 12,361
Trainable params: 12,361
Non-trainable params: 0
_________________________________________________________________
```

Figure 10: Model output summary from Keras after more improvements

The following figure represents the training loss over time for the validation and train datasets, earlier stopping was used in order to minimize overfitting.

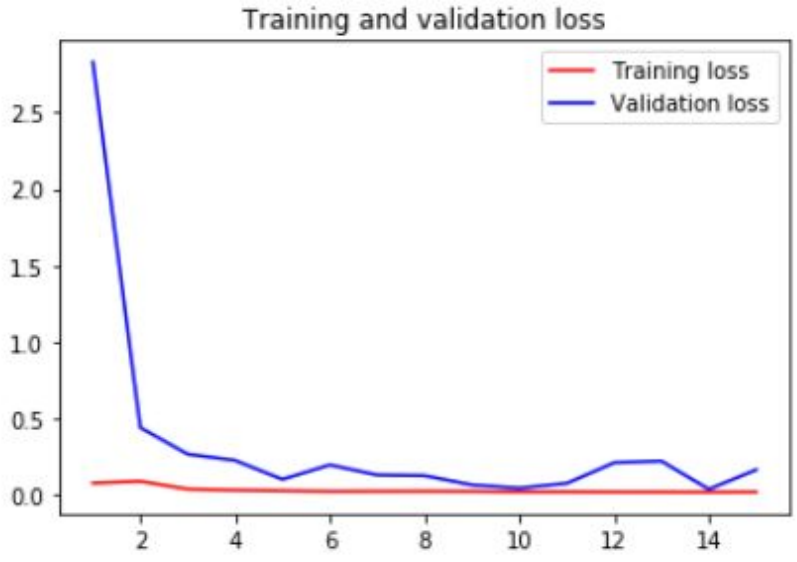

Figure 10: Training and validation steps

Results

Model Evaluation and Validation

The final architecture and combinations of selected parameters are the result of multiple iterations and validations with different architectures, and hyperparameters, multiple of this test resulted in poor performance due to extreme overfitting of the model to the dataset and in some cases underfitting.

During this time and thanks to a robust cross validation strategy, that started by isolation datasets to be used as sequences for training, validation and testing. I was confident that not much information was leaked into the model.

The following list describes each of the selection process completed to reach the final architecture

- The final selected amount of layers for the model is **3, two LSTM layers and one fully connected dense layer**, any other number of layers cause massive overfitting reducing the performance of the model
- The first LSTM layer has **32 units**, and the second only **16 units**, higher number of layers also resulted in overfitting so we found this to be and optimal number of them
- The LSTM layers have two types of dropout, the first one is recurrent dropout and the second is general dropout, this numbers are set quite low into the model with **numbers around 0.001% probability of dropout**, any bigger number cause the model to drop performance due to not having enough information, **only over 2654 points**
- The third layer is a fully connected layer without dropout to act as a buffer between the output layer, by not having this layer in between the model performance decreased
- The model was trained with multiple sequences created by selecting **batches of size 32**, bigger numbers cause the model to drop performance
- The sequences selected for the mode look back **30/120 days on the past** and **one day to seven days into the future for targets**, smaller numbers also seems to

impact the model performance, looking back into the past also influence the model performance due to high variability of the data over time

- The final **selected training rate was 0.001** using smaller numbers increase the convergence time of the model to much, this value is quite good taking into account that we utilize a **learning rate decay of 0.0005**
- The model was trained for **20 epoch** more than that is not necessary there is not much information even if we are using 10 years of data

To verify the robustness of the model we relied on the validations sets, of data that is weeks away into the future than the training datasets, this is extremely beneficial for time series were leakage of data is quite common

There was also a test dataset to evaluate the final performance of the model in data never seen before, not even in the validation set, helping to make the final evaluation reliable

Justification

The results of the stock prediction model utilizing tensorflow, demonstrate that it can surpass the benchmark stock prediction model (benchmark model has been trained to predict the next days stock value by using the previous day value) this is only true when we predict for multiple days into the future, at least more than one day. Another huge benefit of the model that can be useful to extract trends of positive and negative movement based on a comparison of the previous few days, something that is not possible with the simple naive approach.

The result can be summarized in the following image that demonstrate the lost in performance as we try to predict more into the future, the benchmark model and the Keras/LSTM model are compared side by side.

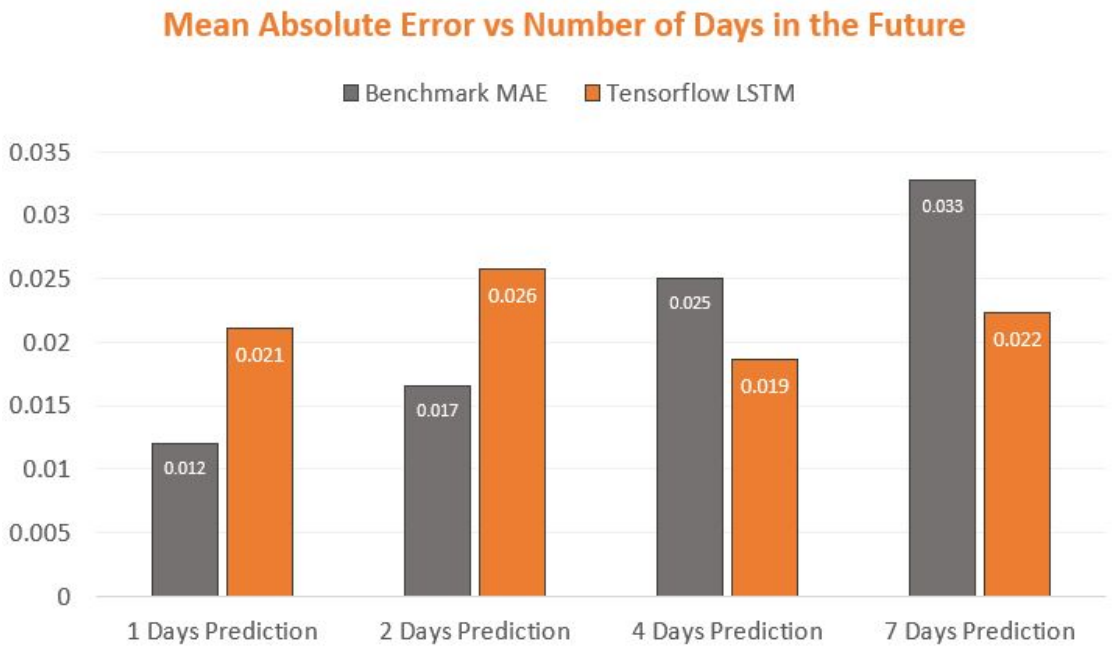

Figure 11: Model MAE comparison vs # of days predictions in the future

Conclusion

Reflection

Inorder to complete the training process and the creation of a machine learning model to predict stock value into the future the following steps were completed:

1. Find an initial problem to solve and develop a strategy around
2. Collect all information or data about the problem, try to establish the target and a baseline
3. Create a benchmark model in order to test consecutive improvements on the model
4. Train the model and iterate with new features, parameters and architecture to enhance ot performance
5. Deploy the model using an estimator
6. Serve predictions

I found the steps numbers 5 and 6 the most complex ones due to specific and detailed configurations that Cloud services required in order to deploy models, this is a great opportunity to improve myself in regards to this type of technologies and streamline the process for future models.

Is quite important to notice that there is a huge

Improvement

There can be multiple ways to enhance the model, in future iterations, but this will require to enhance the datasets utilizing data from multiple fields that are capturing data or have some kind of influence on the stock value that we are interested in, for example potential enhancements will be to add data from competitors, social media, news, and world economic indicators, there is also important to to additional effort on feature engineering and collecting more historical data for the stock value.

The top recommendations to enhance the model in future iterations will be:

- Add a more diversified datasets with multiple source of information
- Add more historical data, the model was trained with 10 years of data potentially load all data available
- Implement automatic hyperparameter tuning
- Utilize NLP and other techniques to extract more insights from the datasets
- Accelerate the training time using GPU

One of the last improvements opportunities for anyone working on these types of models is to increase the amount of domain expertise in regards to the field and stock markets in general, this is a huge plus that will provide more context and insight to improve the model performance.