Dividend Champion Predictor

Connor DePalma
ECE Dept.
Stevens Institute of Technology
Somerville MA, United States
cdepalma@stevens.edu

Tim Demetriades
ECE Dept.
Stevens Institute of Technology
Staten Island NY, United States
edemetri@stevens.edu

Dan Pelis
ECE Dept.
Stevens Institute of Technology
Hoboken NJ, United States
dpelis@stevens.edu

Abstract—**Predicting the performance of a stock is not an easy task. It is something people have spent years working on, with countless machine learning models having been created to try to assess whether a certain stock is a good or bad investment. Dividends, which are payments made to its shareholders, can be used to determine how well a company is performing. Dividend champions are companies that have been paying out increasing dividends for over 25 years. This makes these companies worthy investments, while at the same time if a company loses its status as a Dividend Champion it may be a good idea to avoid investing. As there are currently no machine learning models that specifically focus on Dividend Champions, we proposed to create a model that can predict whether a company will lose its status as a Dividend Champion based on historical metrics of current and past Dividend Champions to determine if a company is worth investing in. Since the data was time series data, several time series orientated methods were used and compared, including ARIMA, Adaptive Filtering, and an RNN with LSTM cells. Preliminary results for the model were promising, with the model doing a relatively good job at predicting future dividend values and therefore predicting if a company will lose its champion status.**

I. Introduction

Dividend Champions are companies that have kept a streak of increasing their dividends for a minimum of 25 years. These stocks are held in high regard to those who practice the investing strategy known as ”Dividend Snowball”, which relies on the compounding effect caused by reinvesting dividends. As a result, it is generally smart and less risky to invest in Dividend Champions as they have a proven track record of performing well even during times of recession. The purpose of titles such as ”Champion” indicates to investors that these companies have and will continue to provide a reliable stream of dividends for many years. Therefore, we propose a system that will analyze the historical data of the current and past Dividend Champions to predict whether a company will lose its titles in the near future.

The Dividend Investing Resource Center (dripinvesting.org) is a centralized hub for research and information about dividend investing. The source provides monthly reports including the list of the current Dividend Champions and many metrics such as their share price and dividend yield. These reports can be used to create a reliable list of Dividend Champions dating back to 2010, including the companies which held the Champion title but later lost their status. From this list the Yahoo Finance API can be called and accurate quarterly data can be queried and organized. The prepossessing steps above result in a robust dataset of past and present Dividend Champions along with their metrics, which can be then be used in a variety of regression or classification algorithms.

The dataset described above allows itself to be used for either classification or regression. However, since the features consist of time series data it is intuitive to perform regression on the set. With time series data there are several different methods that can be used to create successful models. One such method is the ARIMA model. The ARIMA (AutoRegressive Integrated Moving Average) model is one of the simplest ways to conduct time series forecasting. By using a linear combination of past values and past forecast errors along with differencing it is able to accurately predict future values, such as future dividends. Because of its simplicity, ARIMA models were used as a baseline to compare with more complex machine learning models, such as adaptive filtering and RNNs.

Another approach to time series analysis are Adaptive Filters. This process treats time series data as a signal and fits a filter's coefficients to optimize a given cost function on that signal. The Adaptive Filtering process does not create a generalized model, therefore a new filter needs to be fit for every stock, however it is very easy for the filter to adapt as new data becomes available. Adaptive Filters are relatively simple algorithms, therefore the only parameters that need to be tuned for each signal are the cost function, learning rate, and number of time steps to take into account.

Lastly, one more method that is more complex but can work the best in some scenarios is Recurrent Neural Networks (RNNs). RNNs are great at modeling sequential data due to their looping architecture that allows signals to travel in both directions, essentially providing memory to the network. However, a simple RNN is usually not particularly accurate due to the common issue of vanishing gradients. The vanishing gradient problem causes these RNNs to have short-term memory, meaning they have trouble remembering earlier data and as a result will make its predictions mostly from the most recent data. There is a solution though, in the form of LSTMs and GRUs. The Long Short Term Memory Network (LSTM) and Gated Recurrent Unit (GRU) variations of the classic RNN algorithm will help avoid the problem of vanishing gradients and help improve results.

Models for these three methods, ARIMA, adaptive filtering, and RNNs with LSTM cells, were created and compared to see which produced the best results with the data provided.

As for the results, the team plotted the predictions of each model with the true values from the test samples to visually determine the performance of the models. It can be seen that for the companies that lost champion status, the model does a pretty good job at predicting the following dividend value given the previous two dividend values. The mean squared error (MSE) was also calculated and was relatively low for each model. After making an adjustment for lag, the RNN model produced even better results. These results made it easy to determine if a company was to lose its status as a Dividend Champion.

Many of the solutions to similar problems the team found relating to stock forecasting use various models, such as multiple linear regression, support vector regression, and neural networks. However, the majority of these models focus on predicting the future price of stocks. While this may be useful in determining if a stock is worth investing in, we believe that focusing on Dividend Champions can potentially be a superior determiner in whether to invest in a stock. Additionally, the use of methods such as RNNs with LSTM cells allows for long-term memory, meaning the model is able to use not just recent metrics but a long sequence of metrics for its forecasting, resulting in higher accuracy.

II. Related Work

Being that Dividend Champions are generally well-established and profitable companies, they are generally less of a risk to invest in and therefore desirable to many people. This is why people started tracking Dividend Champions, such as Dave Fish who started the spreadsheets that contain a list of all the Dividend Champions along with Contenders (companies who have increased their dividends for the past 10 years) and Challengers (5 years) and is the source of our dataset.

Although these Dividend Champions have been tracked for over 10 years, from what we have found there are no machine learning models that have been created to make predictions based on these lists. While many machine learning models have been created to forecast stock prices and predict whether or not to invest in a stock based on such metrics, there are currently no models that are able to predict whether or not a Dividend Champion is going to lose its status, which can be an important metric in determining whether or not to invest in a stock.

Many of the related work found focuses on predicting the future price of a stock. For instance, David Enke, Manfred Grauer, and Nijat Mehdiyev introduced a way to use multiple regression and type-2 Fuzzy Clustering to create prediction models for forecasting stock market prices 1. Multiple regression is similar to regular linear regression, although instead of there only being one independent variable there are several (different metrics can influence the price of a stock), while Type-2 Fuzzy Clustering uses type-2 fuzzy sets to make predictions. In their paper, the three explain how they created a hybrid of the models to predict stock price levels.

Support Vector Regression (SVR) is another machine learning technique that can be used for regression. SVR is similar to the popular technique of Supper Vector Machines (SVM), although instead of trying to minimize the number of points within the decision boundaries, SVR tries to maximize this. Bruno Henrique, Vinicius Sobeiro, and Herbert Kimura proposed the use of SVR models to predict stock prices for large and small capitalisations and in three different markets 2.

The use of neural networks is also very common with both regression and classification tasks. Neural networks are made up of multiple nodes and layers, with each node behaving similarly to multiple linear regression, and the network tries to find correlations between many different inputs. However, unlike with multiple regression, neural networks can handle non-linear data as a result of the use of non-linear activation functions. Donglin Chen and Dissanayaka Seneviratna attempted to use feed forward back propagation neural networks (BPNN) to conduct one step ahead forecasting on stock price indices 3.

A common trend among all of these solutions that were found is that they all focus on forecasting stock prices. While this is certainly a good way to predict whether a stock is a worthy investment, our solution introduces the idea of Dividend Champions, focusing specifically on metrics such as the dividend yield to predict whether or not a company will lose its champion status, which can potentially be a better determiner if a certain stock is a good or bad investment.

III. Our Solution

A. Description of Dataset

Being that there is no publicly available dataset containing an organized list of Dividend Champions and their historical data, a dataset needed to be created for this experiment. Originally, the dataset was to be comprised of Dividend Aristocrats, which are stocks with a 25 year streak of increasing dividends while also being part of the S&P 500 Stock Index. However, being that these conditions make the Aristocrat list quite short (around 65), it was more efficient to study the Champions list which provides many additional companies for training, as the champions are not required to be part of the S&P 500.

The Dividend Investing Resource Center is an online resource that provides **monthly** reports on the current Dividend Champions, as well as some other Dividend related groupings. These reports contain the list of the current companies maintaining the title of "Dividend Champion" and the stock metrics (such as stock price and dividend yield) at the time of the report. Due to format inconsistencies and incomplete data, we found the best time period to use these reports is from 2010-2020. From these reports two lists can be compiled, one containing the current dividend champions and one tracking companies who were Champions at some point within the time period, but lost their status. Without this resource the team would have needed to iterate through a multitude of data (likely through a financing API) to find companies that maintained increasing dividends for 25 years only to lower payouts in a subsequent year in order to find the companies that lost their status as Dividend Champions.

An example of a company, Washington Real Estate Investment Trust, that lost status can be seen in Figure 1 below. As shown, this company had increasing dividends for several years, but then had a significant reduction in their dividend payout for a sustained period of time starting around July 2012 for long enough for there to be an overall negative change from the previous year. This is exactly what constitutes a loss of Champion status.

The lists were separated to enable the dataset to be used for either regression or classification implementations. They were then used to collect accurate historical data through the Yahoo Finance API. This API provides robust historical data on a given stock and offers the ability to collect various stock metrics. For a regression implementation the distributed dividends of the companies were collected and processed.

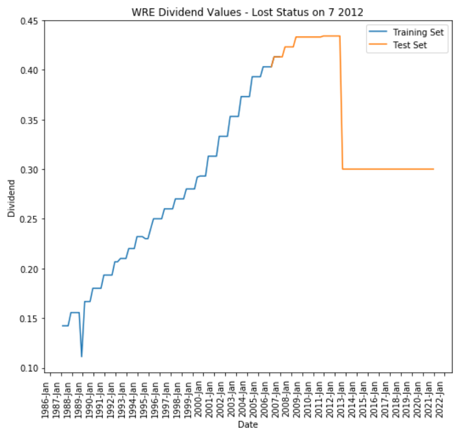

Fig. 1: Lost Status

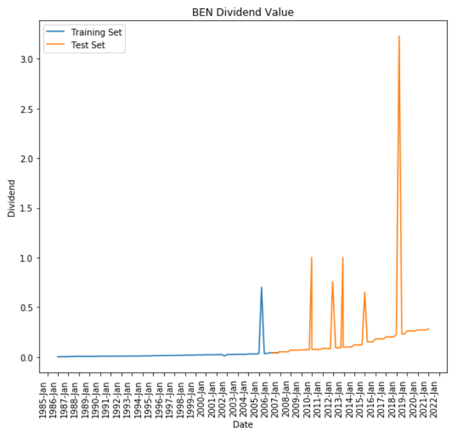

Fig. 2: Kept Status With Noise

Being that dividend distributions are up to the discretion of the company this metric can be quite inconsistent. For instance the company could choose to not pay dividends for a given period, which is usually uncommon for Dividend Champions since they are generally stable and successful companies and since it would cause them to lose their status as Dividend Champions. However, it does sometimes occur, and this introduces missing values into the dataset. To combat this, zeros needed to be added in these missing periods, since zero is an accurate representation of the dividend amount distributed for that period.

A source of anomalies is a type of dividend called *Special Dividends*. These are unscheduled one-time dividends which are usually larger than the normal dividend stream a company puts out. In terms of the dataset, these special dividends would produce spikes that act as noise and would hurt a model's ability to find a trend in the regular dividend stream, therefore they needed to be removed. Figure 2 contains a visual representation of the dividend values for Franklin Resources, Inc., which has several noisy samples as shown by the large spikes. This sample should not automatically remove it's Champion status, as it was likely an adjustment or additional dividend provided by Franklin Resources, Inc. to its shareholders. The noisy samples were removed by utilizes a filter that is sensitive to large "edges" (samples with a large difference in value) in data commonly seen in image processing applications. The filter allowed for the simple removal of the anomalous samples, as shown in figure 3. By decreasing the impact of the noisy Special Dividends, the team is ensuring that the generated models will not accidentally predict the loss of Champion status.

Finally, dividends do not have to be paid on a quarterly basis. It is common to find companies that pay their dividends on a monthly basis, meaning these dividends need to be combined into a quarterly yield to be properly compared with the other companies.

B. *Machine Learning Algorithms*

1) ARIMA: AutoRegressive Integrated Moving Average (ARIMA) models are one of the simplest, and as a result most commonly used, time series forecasting methods. They work

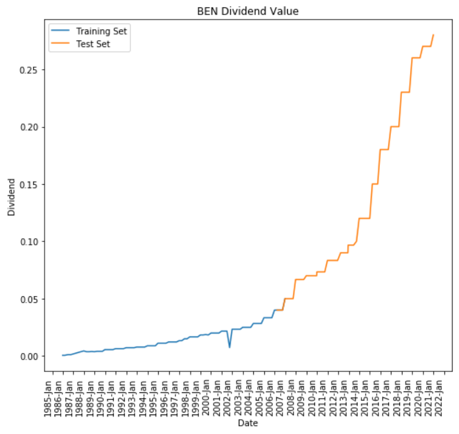

Fig. 3: Kept Status With Noise Removed

on the basis of previous values carrying inherent information that can be used to predict future values. ARIMA models consist of three parts - an AutoRegressive part, a Moving Average part, and a Integrated part. Starting with the AutoReressive (AR) part, these are models that depend on past lagged values to estimate future values. The evolving variable of interest is regressed on its own prior values. As for the Moving Average (MA) part, these are models that depend on past forecast errors to make predictions. It specifies that the regression error is actually a linear combination of error terms whose values occurred at various times in the past. Lastly, as for the Integrated part, this essentially just means the model looks at a differenced version of the data in an effort to reduce non-stationarity and allow the model to better fit the data. The formula for the general ARIMA model 4 is made up of a bias term, terms for the AR part, and terms for the MA part:

$$\hat{y}_t = \mu + \phi_1 y_{t-1} + \ldots + \phi_p y_{t-p} - \theta_1 e_{t-1} - \ldots - \theta_q e_{t-q}$$

Fig. 4: ARIMA General Forecasting Formula

The order of the model is determined by three parameters: **p**, which is the order of the AR part, **q**, which is the order of the MA part, and **d**, which is the number of differencing applied to the data. The **p** parameter can be determined from the partial autocorrelation (PACF) plot. The **q** parameter can be determined from the autocorrelation (ACF) plot. Finally, the **d** parameter can be determined by testing for stationarity such as using the Augmented Dickey-Fuller (ADF) test. For simplicity, all of these parameters were set to 1, which seemed to produce the best results.

2) Adaptive Filtering: Adaptive Filters (AF) are commonly used in image and signal processing, however they have proven to perform well on times series data. Adaptive Filters are computational devices that attempt to model the relationship between signals. This is done by adjusting the coefficients of the filter to optimize a given cost function, as such these filters are usually referred to by their cost functions. There are various types of Adaptive Filters, one popular example are Least-mean-squares (LMS) filters which are fit to minimize the least mean square of the error signal.

The parameters chosen when creating an Adaptive Filer include, the cost function, the learning rate (μ), and the number of time steps used for prediction (n). As discussed above the cost function is used to tune the coefficients of the filter. For this Normalized Least-mean-squares (NLMS) was chosen, as it provides more stable and optimal results when compared to vanilla LMS. For the NLMS algorithm the normal learning rate is replaced with the following equations4:

$$\mu + \frac{2\epsilon}{(||x(k)||^2)}$$

According to theory μ= 1 should provide the best convergence, however a μbetween 0 and 2 could be chosen. The original learning rate chosen was 0, however this is chosen just to instantiate the Filter object. An optimization algorithm was then run to find the optimal μgiven the other parameters. Finally from manual tests, the best number of time steps used was set to 5. Note that the number of time steps was set to 1 in the comparison section to match the other models.

3) LSTM: Recurrent Neural Nets (RNNs) use connections along a temporal sequence to process variable length input sequences. These RNNs keep an internal hidden state, often referred to as memory, to keep track of the information they have already seen. These neural networks create recurrent connections by using the output of the preceding step as the input for the current step. Given the temporal nature of RNNs, they are a common solution applied to time series analysis problems. A limitation of RNNs comes when the model is applied to a long time series (usually hundreds of time steps). This long input sequence translates to a large amount of steps in the RNN, which becomes a problem during gradient descent since the algorithm must propagate back through these steps. This causes the recurrent weight that connects the layers to become very small leading to the values sent through the network to get smaller with each step. This issue is known as the **Vanishing Gradient** problem.

Due to the transformations the data goes through when traversing an RNN, some information is lost at each time step, which causes the RNN state to essentially forget the beginning of the sequence. Long Short-Term Memory (LSTM) networks, a variant of RNNs, address this and the vanishing gradient with the use of long-term memory, known as *cell state*. An LSTM cell is similar to a regular cell except that it splits the state into two vectors - the hidden (short-term) state and the cell (long-term) state. In doing so, it allows the network to learn what to remember and store in the cell state and what to forget. The LSTM can then preserve this for as long as it is needed (and forget it when it's not needed) and extract it when it's needed. As a result, while LSTMs may be more complex than regular RNNs, they are able to remember much more information, producing superior performance and thus have made regular RNNs almost irrelevant.

The design of the teams' LSTM model 5 consisted of an arbitrary number of LSTM layers and hidden units, which were later tuned after experimentation. The dropout stages were added between the LSTM layers to statistically mask weights and reduce overfitting throughout the network. Additionally, the LSTM layers were also chosen to be bidirectional, which allows them to preserve both past and future information. Since LSTMs were used there was no need to choose activation functions. LSTMs by definition use a tanh activate function for cell state and a sigmoid activation function for cell output. The model ends with a single Dense layer with an output of size one, representing the stocks dividend yield for the succeeding time period.

C. Implementation Details

1) ARIMA: As mentioned previously, several types of models were created so they could be compared. ARIMA models are one of the simplest models to use when dealing with time series data and as a result were used as a sort of baseline to compare with more complex machine learning models such as adaptive filtering and RNNs to see if they were even necessary to produce better results. ARIMA models were chosen compared to standard ARMA models as the original dividend data was not stationary, so applying a 1 period difference (d = 1) would make the dividend data stationary and allow for better results. The ADF test was used to determine that a 1 period difference would make the data stationary, as after applying the 1 period difference the p-value from the test went from almost 1 to under 0.05, allowing us to reject the null hypothesis and assert that the data is now stationary. As for the other 2 ARIMA parameters (p and q), these were determined based on the PACF and ACF plots, respectively. PACF and ACF plots were created using the differenced dividend data for several companies, and from these it was determined that the best values for these parameters would be 1 for the majority of the companies, so an ARIMA model of order (1, 1, 1) was created for each company's dividend data, which is good for simplicity sake as higher order models are more complex and only necessary if they perform better.

An ARIMA model was fit to each company's dividend data. To do this, first the data was split into 70 percent training data and 30 percent test data. The training data was then used to fit an ARIMA model to each company's dividend data, and then forecasting was done using the test data to determine the performance of the model. Additionally, the MSE for each model was calculated so they could be compared. The following figures contain predictions from the ARIMA models

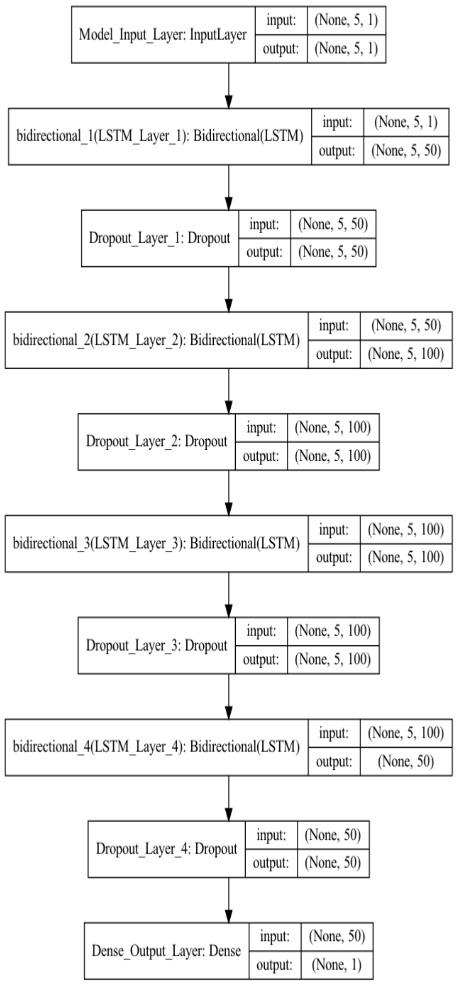
Model_Input_Layer: InputLayer | input: (None, 5, 1) | output: (None, 5, 1)
bidirectional_1(LSTM_Layer_1): Bidirectional(LSTM) | input: (None, 5, 1) | output: (None, 5, 50)
Dropout_Layer_1: Dropout | input: (None, 5, 50) | output: (None, 5, 50)
bidirectional_2(LSTM_Layer_2): Bidirectional(LSTM) | input: (None, 5, 50) | output: (None, 5, 100)
Dropout_Layer_2: Dropout | input: (None, 5, 100) | output: (None, 5, 100)
bidirectional_3(LSTM_Layer_3): Bidirectional(LSTM) | input: (None, 5, 100) | output: (None, 5, 100)
Dropout_Layer_3: Dropout | input: (None, 5, 100) | output: (None, 5, 100)
bidirectional_4(LSTM_Layer_4): Bidirectional(LSTM) | input: (None, 5, 100) | output: (None, 50)
Dropout_Layer_4: Dropout | input: (None, 50) | output: (None, 50)
Dense_Output_Layer: Dense | input: (None, 50) | output: (None, 1)

Fig. 5: LSTM Model

(each of order (1, 1, 1)), one for a company that maintained its Champion status 6 and another for a company that lost its status 7. As can be seen in the figures, the models were able to do an excellent job at forecasting the dividend values.

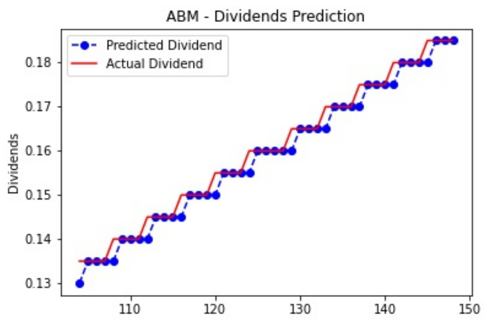

Fig. 6: ARIMA Forecasting - Champion Status Kept

2) Adaptive Filtering: The Adaptive Filter model was implemented using a Python module names ”padasip”, which simplifies adaptive signal processing tasks. Unfortunately, the module does not offer methods similar to GridSearch that would find the optimal filter type and number of time steps for a given input sequence. As mentioned in the previous section the number of time steps used was manually tuned to 5 and the type of filter chosen was a Normalized Least-mean-squares filer.

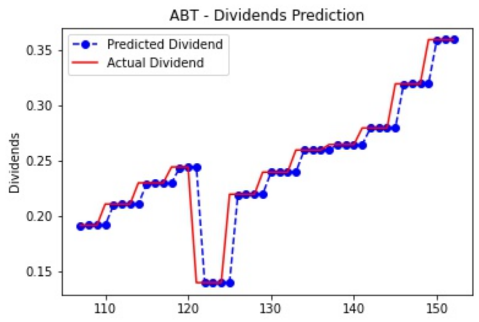

Fig. 7: ARIMA Forecasting - Champion Status Lost

Before an Adaptive Filter can be fit the training data of the stock must be formatted based on the chosen number of time steps. A function from the ”padasip” module provides functionality for converting a 1D time series to a matrix of time steps. Additionally a target function must be defined. In signal processing this may be be some linear equations, however for time series the target function is simply set to the input one time step ahead.

The Adaptive Filter implementation begins by instantiating an Adaptive Filter with a learning rate of zero. This filter is then passed to a tuning algorithm to find the best learning rate from 0.01 to 1. Once the best rate it found a new NLMS Filter is defined with the desired parameters and it is fit to the training data of a given stock. As the name of the filter suggests, the training process will return Squared Error along with the fit filter and training predictions. The error is later converted into Mean Squared Error for comparison with the other models.

The results of the Adaptive Filter fit by implementation described above can be seen in figure 8. The first plot shows the predicted values split by train and test layered on top of the actual dividend data for the ABT stock. It is clear that the filter is able to make predictions very close to the actual values, however it can be observed in this example and other instances that large spikes will not be tracked closely. The filter will pick up on the direction change of the signal but will not predict as drastic of movements as the actual signal. Since the movement of the signal is still predicted, this does not render the filter useless. This would only be an issue if the filer was trying to predict exact values far into the future. The second plot of figure 8 shows the squared error of the ABT filter along with its MSE on the test set. The low MSE of this plot matches with the accurate tracking shown in the first plot.

The results of these filters do vary between experiments due to the stochastic nature of the weight initialization. When instantiating a filer class the weights can be set to zeros or random numbers. Testing has shown that random initialization

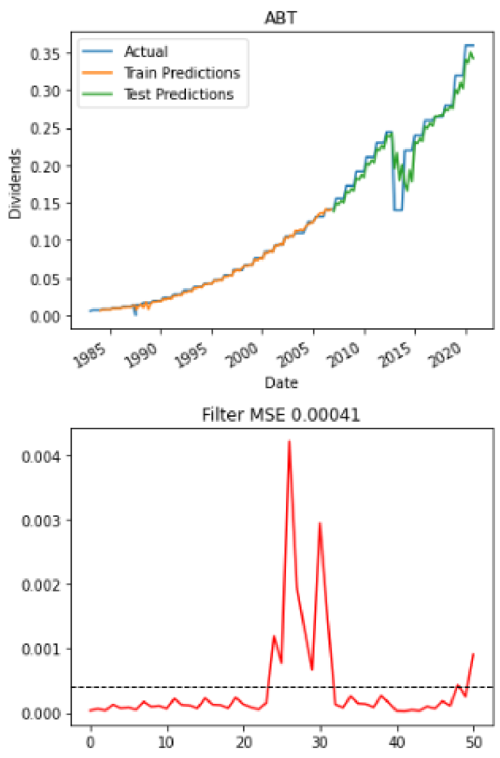

Fig. 8: Adaptive Filter Prediction With Lag 5 and Squared Error

yields better results, but the results vary greatly between experiments. Initiating weights at zero yield overall worse results, however they are much more consistent.

3) LSTM: As mentioned in the previous section, the team also has an LSTM model 5 as a candidate algorithm that could provide a solution to the proposed problem. Similar to the dataset creation and clean up process, the model was written in Python3 using Jupyter Notebooks. This environment allowed for easy testing and debugging of the model by allowing for inline visualizations and intermediary outputs. The model was written using Google's Keras library, which contains high-level classes and methods that wrap TensorFlow (Google's core AI framework) functions, objects, etc. The use of Keras allowed for the team to quickly test different LSTM architectures and tune hyper-parameters.

The hyper-parameters were tuned through validation testing of the model. This involved using a subset of the training samples (*not* testing) as validation for the accuracy of the model at each epoch, known as the validation set. Since a regressive model was chosen, ideally the predictions would align closely with the testing samples themselves. This meant that a simple loss function like Mean-Squared-Error (MSE) would suffice for this implementation. The validated results were based off of the MSE measurements and helped the team decide whether a parameter change helped or hindered the model.

The teams results for the LSTM algorithm implementation are visually based. Since the intention of the model was to ingest a handful of samples and predict the next sample, the best way to test the performance of the model is to simply plot the prediction alongside the true values of the test samples. The Washington Real Estate Investing Trust, the same company shown in Figure 1, was pulled out as an example since it was known that the company lost its status. In Figure 9 it can been seen that the model does a decent job of predicting the next dividend value given two previous dividends.

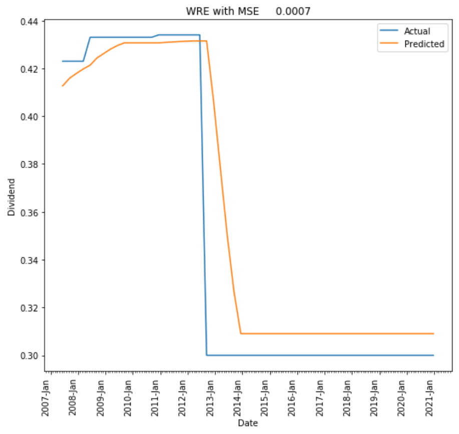

Fig. 9: Model Prediction Example

This resulted in a relatively low MSE, however there is lag introduced by the LSTM based model that would need to be compensated for in order to provide more accurate predictions. An example of the same results for the Washington Real Estate Investing Trust are shown in Figure 10 but with a lag adjustment of a single time step. It is can easily be seen how this improves the model prediction, to the point where the team could possibly use these results to quickly determine that this company would be on the verge of losing status around July 2012.

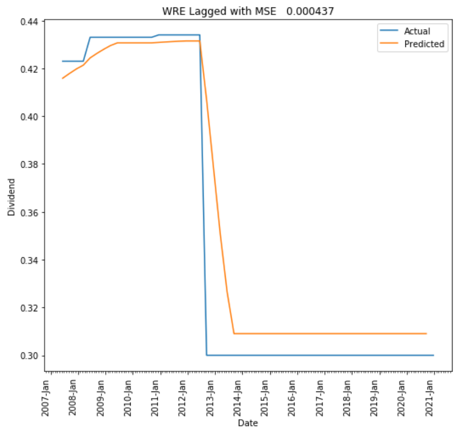

Fig. 10: Model Prediction Example With Lag Adjustment

IV. Comparison

The three methods the team used to create models, ARIMA, adaptive filtering, and RNNs all performed relatively well with

	ARIMA	AD	RNN
MGEE	0.00003	0.01609	0.00003
TGT	0.00053	0.06771	0.00041
BEN	0.00009	0.01621	0.00009
CBSH	0.00003	0.01126	0.00003
PBI	0.00091	0.01185	0.00134
ABT	0.00039	0.01785	0.00042
WRE	0.00033	0.02700	0.00044
HP	0.00641	0.09980	0.00626

TABLE I: MSE Comparison for Different Models

forecasting the dividends. After fitting a model to the training data, each method was able to produce models that could make accurate predictions compared to the test data. The mean squared error (MSE) values for all of the models produced was relatively low. In particular, the MSE values for the ARIMA and RNN models were the lowest and also very similar. Both of these methods were able to generate predictions that were very close to the actual dividend values.

However, while still low, the adaptive filtering models produced predictions with higher MSE compared to the other two models. This may make it seem like this model would be less ideal to forecast with, but on closer examination of the curves produced from the three methods, the adaptive filtering models' curves may actually be better at predicting a drop in dividend yield. For example, looking at the curves in this figure 11, it can be seen that even though the curves from the ARIMA and RNN models fit the actual curve better, the shape of the curve is better represented by the curve the adaptive filtering model created. The curve may be lower, but it does a better job at showing the trend of the dividend, which is useful when trying to determine when a drop will happen and as a result when a company will lose Champion status.

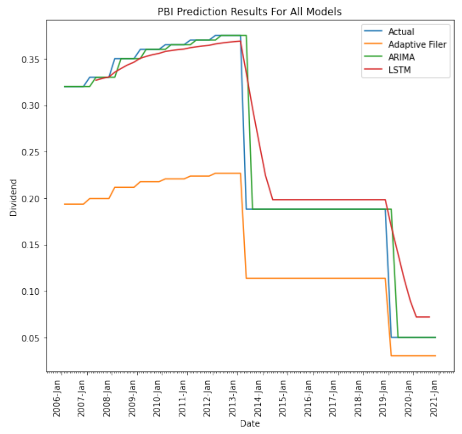

Fig. 11: PBI Prediction Results for All Models

V. Future Directions

If given additional time, several things could have been done to further improve the performance of the models created. For instance, more data could have been gathered and metrics other than dividend yield could have been used. The hyper-parameters of the models could have also been tuned further with cross-validation to find more optimal values. For example, while ARIMA (1, 1, 1) models were mostly used, different order models could have been found that may have produced better results. Additionally, ensembling, combining the predictions from multiple models, could have been done to produce even better results.

Better models would produce more accurate predictions. They would also allow for forecasting farther into the future, which would be useful in determining when and/or if a drop was coming. This could be used to determine if a company is going to lose Champion status even if it was to happen way in the future, which would allow for better and smarter investments.

VI. Conclusion

The experiments described above resulted in three types of models that were able to track the movement of dividend yields well. However, there still lies the issue that these models only predict one time step into the future. A Dividend Champion's title is determined based on annual yield, therefore to be effective the models need to predict at least four time steps in the future to show a steady trend.

The experiments also yielded that the best of the three models for this problem would be the Adaptive Filters. It was found that although these models had the highest MSE values, they also tracked the trends of their respective stocks the closest, which was the desired effect of this research. The exact values of the predictions are not crucial as long as the predicted trend follows the truth.

Further improvements to these models would be in the form of some type of GridSearch. The best cost function for the Adaptive Filters was manually found to be NLMS. It could be beneficial to perform a search to find the best cost function for a given time series. This process could also be applied to finding the optimal number of time steps to use for a prediction. Additionally ensemble methods could be applied with multiple Adaptive Filers using different cost functions.

References

[1] David Enke, Manfred Grauer, Nijat Mehdiyev, Stock Market Prediction with Multiple Regression, Fuzzy Type-2 Clustering and Neural Networks, Procedia Computer Science, Volume 6, 2011, Pages 201-206, ISSN 1877-0509, https://www.sciencedirect.com/science/article/pii/S1877050911005035

[2] Bruno Miranda Henrique, Vinicius Amorim Sobreiro, Herbert Kimura, Stock price prediction using support vector regression on daily and up to the minute prices, The Journal of Finance and Data Science, Volume 4, Issue 3, 2018, Pages 183-201, ISSN 2405-9188, https://www.sciencedirect.com/science/article/pii/S2405918818300060

[3] Chen, D. and Seneviratna, D. (2014) Using Feed Forward BPNN for Forecasting All Share Price Index. Journal of Data Analysis and Information Processing, 2, 87-94. doi: https://www.scirp.org/(S(351jmbntvnsjt1aadkposzje))/journal/paperinformation.aspx?paperid=51486

[4] Cejnek, M (2017) Padasip - open source library for adaptive signal processing in language Python Studentská tvůrčí činnost https://stc.fs.cvut.cz/pdf17/6563.pdf