Forecasting ETFs with Machine Learning Algorithms

By Jim Kyung-Soo Liew and Boris Mayster

Email: kliew1@jhu.edu & boris.mayster@jhu.edu

Johns Hopkins Carey Business School

Version 1.3

http://etfprediction.pythonanywhere.com/

Date: January 14, 2017

In this work, we apply cutting edge machine learning algorithms to one of the oldest challenges in finance: Predicting returns. For the sake of simplicity, we focus on predicting the direction (e.g. either up or down) of several liquid ETFs and do not attempt to predict the magnitude of price changes. The ETFs we use serve as asset class proxies. We employ approximately five years of historical daily data obtained through Yahoo Finance from January 2011 to January 2016. Utilizing our supervised learning classification algorithms, readily available from Python's Scikit-Learn, we employ three powerful techniques: (1) Deep Neural Networks, (2) Random Forests, and (3) Support Vector Machines (linear and radial basis function). We document the performance of our three algorithms across our four information sets. We segment our information sets into (A) past returns, (B) past volume, (C) dummies for days/months, and a combination of all three. We introduce our "gain criterion" to aid in our comparison of classifiers' performance. First, we find that these algorithms work well over the one-month to three-month horizons. Short-horizon predictability, over days, is extremely difficult, thus our results support the short-term random walk hypothesis. Second, we document the importance of cross-sectional and intertemporal volume as a powerful information set. Third, we show that many features are needed for predictability as each feature provides very small contributions. We conclude, therefore, that ETFs can be predicted with machine learning algorithms but practitioners should incorporate prior knowledge of markets and intuition on asset class behavior.

Machine learning and artificial intelligence algorithms have come home to roost. These algorithms, whether we like it or not, will continue to permeate our daily lives. Nowhere is this more evident than in their current uses in self-driving cars, spam filters, movie recommendation systems, credit fraud detection, geo-fencing marketing campaigns, etc. The usage of these algorithms will only expand and deepen going forward. Recently, Stephen Hawking issued a forewarning -- *"the automation of factories has already decimated jobs in traditional manufacturing, and the rise of artificial intelligence is likely to extend this job destruction deep into the middle classes"*[1] Whether we agree or disagree with regard to the virtues of automation, the only way to better utilize its potentials and evade its dangers is to gain a deeper knowledge and appreciation of these algorithms. Moreover, quite nearly upon us is the next big wave called the Internet of Things (IoT) whereby increasingly more devices and common household items will be interconnected and have the ability to stream tera-bytes of data. As our society is deluged with data, the critical question that emerges is whether machine learning algorithms contribute a net benefit or extract a net cost to society.

While the future looms large for machine learning and artificial intelligence, one pocket of their development appears to have been deliberately left behind. Namely, in finance, and more so in hedge funds that attempt to predict asset prices to generate alpha for their clients. The reason is clear: one trader's gain in applying a well-traded learning algorithms is another's loss. This edge becomes a closely guarded secret, in many cases, defining the hedge fund's secret sauce. In this work, we investigate the benefits of applying machine learning algorithm to this corner of the financial industry which academic researchers have left unexamined.

We are essentially interested in understanding whether machine learning algorithms can be applied to predicting financial assets. More specifically, the goal is to program an AI to throw off profits by learning and besting other traders, and possibly other machines. This goal has been rumored to have already been achieved by the hedge fund Renaissance Technology's Medallion fund. The Medallion fund, cloaked in mystery to all but a few insiders, has generated amazing performance over an extended time period. Renaissance Technology clearly has access to enough brain-power to be on the cutting edge of any machine learning implementation. And as much as others have tried to replicate their success, Medallion's secret sauce has yet to be cracked. In this work, we attempt to unravel several potential research paths in the attempt to shed light on how machine learning algorithms could be employed to trade financial market assets. We employ machine learning algorithms to build and test models on prediction of asset price direction for several well-known and highly liquid ETFs.

Fama(1965) states that empirical evidence supports the notion that stock prices follow a random walk. Markets are efficient, or at a minimum, at least semi-strong form efficient. All publicly-available information is reflected in the stock prices and the pricing mechanism is extremely quick and efficient in processing new information sets. Attempting to gain an edge is nearly impossible, especially when one tries to process widely accessible public information. Investors are therefore better off holding a well-diversified portfolio of stocks. Clearly, Fama would side with those that have little faith in the ability of these machine learning algorithms to process known publicly available information and with such information gain an edge by trading.

[1] Price, R, Dec. 2, 2016, "Stephen Hawking: Automation and AI is going to decimate middle class jobs", Business Insider - Tech Insider:
http://www.businessinsider.com/stephen-hawking-ai-automation-middle-class-jobs-most-dangerous-moment-humanity-2016-12

Many prior researchers have documented evidence that asset prices are predictable. Banz (1981) documents the size effect, in that market capitalization predicts returns. Jegadeesh and Titman (1993), Asness (1995), and Rouwenhorst (1998) show that past prices help predict returns. Fama French (1992, 1993, 1995) show that fundamental factors book-to-market and size affect returns. Liew and Vassalou (2000) document predictability of fundamental factors and link these factors to future real GDP growth. Keim (1983) document seasonality in returns with more pronounced performance in January. For more recent evidence on international predictability, see Asness, Moskowitz, and Pedersen (2013). Whether the predictability stems from suboptimal behavior along the reasoning of Lakonishok, Shleifer and Vishny (1994), limits to arbitrage by Shleifer and Vishny (1997), or for some unidentified risk based explanation by Fama and French (1992, 1993), it nonetheless appears predictability exists in markets.

We employ the most advanced machine learning algorithms, namely, Deep Neural Networks (DNN), Random Forest (RF), and Support Vector Machines (SVM). Our results are generally similar across the algorithms employed, with a slight advantage for RF and SVM over DNN. We report results for the three distinct algorithms and are interested in predicting ten ETFs price changes. These ETFs were chosen for their popularity as well as liquidity, their historical data being sourced from Yahoo Finance. As we are interested in predicting the change in prices over varying future periods, we employ daily data. The horizons that we attempt to predict range from trading days, weeks, and months.

We test several information sets to determine which sets are most important in predicting across differing horizons. Our information sets are based on (A) prior prices, (B) prior volume, (C) dummies for days of the week and months of the year, and (ABC) all of our information sets. We find that (B) volume is very important for predicting across the 20 days to 60 days horizon. Additionally, we document that each feature has very low predictability so we recommend model builders should use a wide range of features guided by financial experiences and intuition. Our methodology was constructed to be robust and allow for easy switching and testing different information set specifications and securities.

The next section describes the procedures employ and assumptions made in this work with focus on applying best practices when applicable. Afterwards, we discuss the machine algorithms employed. Then, we move into the details of our methodology and present our main results. Finally, we present our thoughts on implementation, weaknesses in our approach, implications, and conclusions.

Procedure with Embedded Best Practices

Machine learning algorithms are extremely powerful and most can easily overfit any data set. In machine learning parlance, overfitting is known as “high variance.” Fitting an overly complex model on the training set does not perform well out of sample or in the test set. Prior to the introduction of cross validation, researchers would rely on their best judgement as to whether a model was over-fit. Currently, the best practices for building a machine learning predictive model is based on the holdout cross-validation procedure.

We therefore employ the holdout cross-validation procedure in our paper. We split the data into three components: the training set, the validation set, and test set. Best-practices state that we should construct the model on the training set and validation set only and use the test set once. Repeatedly using the training set and validation set on that part of the data that does not contain the test set is known as hold out cross-validation. While cross-validation is readily employed in many other fields, some criticize the use in financial time-series data. Nonetheless, we believe our results are still interesting since we are investigating the foundational question of whether using machine learning algorithms work in modelling changes in prices.

The Irksome Stability Assumption

Arguably the most famous cross-sectional relationship is the CAPM model which states that the expected returns on any security is equal to the risk-free rate of returns plus a risk-premium. The CAPM's risk-premium defined as the security's beta multiplied by the excess return on the market. Market observability has been changed by Roll's (1976) critique, however generally speaking the theoretical CAPM has become mainstay in academics as well as in practice. To estimate beta students are taught to run a time-series linear regression of excess return of a given security on excess return on the market. The covariance of a security and the market provides for the fundamental building block to which the CAPM has been constructed. The importance of allowing both cross-sectional relationship to be captured in the algorithm builds on the CAPM intuition.

In this work, however, breaking from some finance tradition we view predictability disregarding the time-series structure. We make the irksome stability assumption that there is a stable relationship between predicting price changes and the many features employed across our information sets. That is, similar to modeling credit card fraud and email spam, we assume that the relationship between the response and features is independent of time. For example, we allow our algorithms to capture the relationship that maps the feature input matrix (X) to the output responses (y). With that said, the features are always known prior to the change in prices.

To sum, we incorporate the current best-practices of balancing the overfitting or high variance problem with the underfitting or high bias problem. This is accomplished by separating the training sample and performing k-fold cross validation on the train sample and employ the test sample only once. In this work, we adhere to this best practices when applicable.

We attempt to use machine learning algorithms to answer the following questions: (1) What is the optimal prediction horizon for ETFs? (2) What are the best information set for such prediction horizons?

Since we have a dependent variable (y) as future price movements, either up or down, in this work we are dealing with a supervised learning problem. The true value of the dependent variable is known apriori. We can also test the accuracy of our forecasts. Accuracy is measured by the percentage of time the model predicts correction over the total number of predictions. While there are a vast number of algorithms from which we could chose, we restrict our analysis to the three following powerful and popular algorithms:

(1) Deep Neural Networks
(2) Random Forests
(3) Support Vector Machines

Deep Neural Networks

Deep neural networks are defined by neural networks with more than one hidden layer. Neural networks are comprised of perceptrons first introduced by Rosenblatt (1957) who built on prior the work of McCulloch and Pitts (1943). McCulloch and Pitts introduced the first concept of a simplified brain cell: the McCulloch-Pitts (MCP) neuron. Widrow and Hoff (1960) improved upon Rosenblatt (1957) by introducing a linear activation function, therefore allowing the solutions to be cast in the minimization of the cost function. With the cost function being defined as the sum of squared errors, with errors in the context of supervised machine learning algorithm, defined as the predicted or hypothesized value minus the true value. The advantage of this setting allows for only the change of the activation function to yield different techniques.

Setting the activation function to either the Logistic or Hyperbolic tangent allows us to arrive at the multi-layered neural network. If the networks have more than one hidden layer, we arrive at our deep artificial neural network see Raschka (2015).

The parameters or weights in our deep neural network setting are determined by gradient descent. The process consists of initializing the weights across the neural network to small random numbers, then forward propagating the weights throughout the network. At each node the weights and input data are multiplied and aggregated then sent through the prespecified activation function. Prior layers are employed as input into the next layer, and repeated until the output layer.. Once the errors have been forward propagated throughout the network, backward propagation by Rumelhart, Hinton, Williams (1986) is employed to adjust the weights. Weights are adjusted until some maximum number of iterations have been met or some minimum limit of error has been achieved. It should be noted that the re-emergence of neural networks can be attributed to backward propagation contribution, which allowed for a much quicker convergence to the optimal parameters. Without backward propagation, this technique would have taken too long for convergence and would have been much less popular.

In our analysis, we employ a deep neural network algorithm. That is, a neural network with more than one hidden layer. Between the input and output layers contain the hidden layers. We employ two and three hidden layer neural networks and thus a deep neural network in this work. Recently, a deep learning neural network beat the best human campion in Go, showing that this algorithm can be employed in surprising ways.

Random Forests:

Random forest introduced by Breiman (2001) has become extremely popular as a machine learning algorithm. Much of this popularity stems from it's quick speed and it's ease of use. Unlike the deep neural network and support vector machines, the random forest classifier does not require any standardization nor normalization of input features in the preprocessing stage. By taking the raw data of features and responses and specifying the number of trees in the forest, random forest will return a model quickly and many times outperforms even the most sophisticated algorithms.

Decision trees can easily overfit the data which many have tried to overcome by making the decision tree more robust. Limiting the depth of the tree and number of leaves in the terminal nodes are some methods employed in attempt to reduce the high variance problem. Random forest takes a very different approach to gain robustness in resultant predictions. Given that decision trees can easily overfit the data, random forests attempt to reduce such overfitting along two dimensions. The first is bootstrapping with replacement the row samples used in a given decision tree. The second is a subset of features that are randomly sampled without replacement at each node-split with the objective of maximizing the information gain for this subsample of features. Parent and child nodes are examined and features selection are chosen that provides for the lowest impurity of the child node. The more homogeneous are the elements within the child split the better the branch is at separating the data.

Many statisticians were irked with random forests when it was initially introduced because it only provided a limited number of features. At that time, the model building intuition was to employ as much data as possible and to avoid limiting the feature space. By limiting the feature space, each tree has slightly different variations and thus the average across the many trees, also known as bagging the trees within the forest, provides for a very robust prediction which easily incorporates the complexity in the data. Random forest continues and will continue to gain even more appeal with the added benefit of allowing researchers to see the features that are most important for given random forest prediction model. We will present a list of the most important feature per ETFs later in this work and our results show the complexity of predicting across our ETF asset classes.

Support Vector Machines (SVM)

SVM by Vapnik (1995) attempts to separate the data by finding supporting vectors that provide for the largest separation between groups or maximizing the margin. Margin is defined as the distance between the supporting hyperplanes.

One of the main advantages of this approach is that SVM generates separations that are less influenced by outliers and potentially more robust vis-a-vis alternative classifiers. Additionally, SVM allows for the option to apply the radial basis function which allows for nonlinear separation by leveraging the kernel-trick. The kernel-trick casts the data into a higher dimension. In this higher dimension, linear separation occurs when projecting the data back down into the original dimensional space the results are non-linear separations.

Methodology

As mentioned earlier, we have chosen widely used and liquid ETFs from various asset classes. The cross-section of ETFs allows us to include cross-asset correlation to boost predictive power. Presumably investors make their decisions depending on their risk preferences as well as the ability to hold a well diversified portfolio of asset. While our list of ETFs is not exhaustive, it does represents well-known ETFs with which most practitioners and Registered Investment Advisors (RIAs) should be well acquainted. The list of ETFs is as follows:

<u>ETF Opportunity Set</u>

● SPY - SPDR S&P 500;	US Equities Large-Cap
● IWM - iShares Russell 2000;	US Equities Small-Cap
● EEM - iShares MSCI Emerging Markets;	Global Emerging Markets Equities
● TLT - iShares 20+ Years;	US Treasury Bonds
● LQD - iShares iBoxx $ Invst Grade Crp Bond;	US Liquid Investment Grade Corporate Bonds
● TIP - iShares TIPS Bond;	US Treasury Inflation Protected Securities
● IYR - iShares US Real Estate	Real Estate
● GLD - SPDR Gold Shares;	Gold
● OIH - VanEck Vectors Oil Services ETF;	Oil
● FXE - CurrencyShares Euro ETF;	Euro

We test predictability of ETF returns on a set of varying horizons. While it is common knowledge that stock prices and thus ETF prices follow a random walk on the shorter horizons, making shorter-term predictability very difficult, longer-horizons may be driven by asset class linkages and attention. Asset classes ebb and flow in and out of investors' favor. With this intuition, we attempt to predict the direction of price moves, not the magnitude. Thus we cast our research into a supervised classification problem. The returns are calculated employing adjusted closing prices, adjusted for stock-splits and dividends, for the given time periods as measured in trading days: 1, 2, 3, 5, 10, 20, 40, 60, 120 and 250 days using following formula:

$$r_{t-n,t}^{ETF} = P_t^{ETF} / P_{t-n}^{ETF} - 1$$

For each horizon of *n* days and each ETF, we examine four data set combinations as explanatory information sets. We employ the term information set as the set of features based on the following explicit definitions. Note that for any given asset's change in price we allow for it's own information as well as the other ETFs' information to influence the sign of the price change over the given horizon. We define our four information sets A, B, C, and ABC as follows:

- Information Set A - Previous *n* days return and *j* lagged *n* days return, where *j* is equivalent to previous horizon (i.e. for 20 days horizon, number of lagged returns will be 10) for all ETFs.

$$X_A = \{r_{t-n,t}, r_{t-n-1,t-1}, \ldots, r_{t-n-j,t-j}\}$$

- Information Set B - Average volume for *n* days and *j* lagged average volume for *n* days, where *j* is equivalent to previous horizon for all ETFs.

$$X_B = \{v_{t-n+1,t}, v_{t-n,t-1}, \ldots, v_{t-n-j+1,t-j}\}$$

- Information Set C - Day of the week and month dummy variables.
- Information Set ABC - A, B and C combined.

We concentrate our presentation of results on Information Set ABC, but the other information sets shed insight on the drivers of the predictions across our three algorithms. Apriori, we believe that past returns will be the most beneficial in terms of future return predictions and volume will be useful to boost the results. Volume has been shown by many to capture the notion of investors' "attention." Higher volumes are typically associated with more trading activity. If trading releases information, then those ETFs with higher volume of trading should be adjusting more quickly to their true values. Note that we implicitly assume the dollar volume of trading is approximately equal across ETFs and concern our study with share volume. Clearly, the ETF prices are not equal at any given time. However, the intuition is clear, more relative trading volume is an important feature for predictability across ETFs. We would suspect that that volume and prior returns should work in tandem. However, we find that volume in isolation works very well, that is, B works well even without A -- a rather surprising result!

Dummy variables are assumed to boost the performance of algorithms on shorter-time periods and to be insignificant on longer horizons. It is important to notice, that A and ABC data sets are equivalent in number of observations, while B and C are not equivalent to A and have one less observations. That occurs since we need n+1 days of adjusted closing prices to compute returns. We only need n days, however, to compute average volume. We are employing the daily volume for that day.

Dependent variable is defined as 1 if *n* days return is equal or greater than 0 and 0 otherwise:

$$Y = \left\{ 1,\ if\ \ r_{t,t+n} \geq 0;\ 0\ otherwise \right\}$$

Next, we employ cross-validation. We divide data sets and corresponding dependent variable into training and test sets. Division is done randomly in following proportion -- 70% training set and 30% test set. Training set will be used to train model and test set for estimation of predictive power of algorithms. Following best practice procedures, we use our training set and perform holdout cross-validation. Training set is separated into the training set and the validation set. Once the k-fold cross validation has been performed and the optimal hyper-parameters have been selected, we employ this model only once on our test set. Many textbooks recommend 10-fold cross validation in the training set, however we used only 3-fold cross validation given that larger cross-validation tests would take even longer time to generate results. We exhaustively searched for the best hyperparameters for each of our algorithm[2]. The possible values for hyperparameters for each algorithm are as follows:

Hyper-parameter Search Space[3]

- Deep Neural Nets (DNN)
  - Alpha (L2 regularization term) - $\{0.0001,\ 0.001,\ 0.01,\ 0.1,\ 1.0,\ 10.0,\ 100.0,\ 1000.0\}$
  - Activation Function - $\{rectified\ unit\ linear\ function,\ logistic\ sigmoid\ function,\ hyperbolic\ tan\ function\}$ [4]
  - Solver for weight optimization - $stochastic\ gradient\ descent$
  - Hidden layers - $\{(100,\ 100),\ (100,\ 100,\ 100)\}$

- Random Forest (RF)
  - Number of decision trees - $\{100,\ 200,\ 300\}$
  - Function to measure quality of a split - $\{Gini\ impurity,\ information\ gain\}$

- Support Vector Machine (SVM)
  - C (penalty parameter of the error term) - $\{0.0001,\ 0.001,\ 0.01,\ 0.1,\ 1.0,\ 10.0,\ 100.0,\ 1000.0\}$
  - Kernel - $\{linear,\ radial\ basis\ function\}$

The best estimator is decided upon accuracy of all possible combinations of hyperparameters values listed above. After the best estimator is found we use test set data to see how algorithm performs. Scoring for test performance is based on accuracy. Accuracy defined by how well the predicted outcome by the model compares to the actual outcome. In order to estimate performance of each algorithm we introduced our gain criteria. This criteria shows whether explanatory data set explains dependent variable better than randomly generated noise.

Introduce Gain Criteria

[2] In case of DNN and SVM we also standardize the data using training set prior to training and testing estimator. Standardization computed as the ratio of the demeaned feature divided by the standard deviation of that feature. Thus, in the training set each feature has zero mean and standard deviation one, however, in the test set the features' mean and standard deviation varies from zero and one.
[3] All other parameters have default values. For more information see http://scikit-learn.org/
[4] f(x) = max(0, x), f(x) = 1 / (1 + exp(-x)) and f(x) = tanh(x) respectively

Gain criteria is computed as the difference between the accuracy of the model given the input information set and the accuracy of the model given noise data. We define noise as creating random data from uniform random distribution bounded by zero and one and replacing the original input data with this simulated noise data in the modeling process. We replaced this noise directly into the input feature data space, which preserves the shape of the actual data. By re-running the same code as we previously did to obtain accuracy score for original data and subtracting the result from score obtained by testing best estimator with actual data we compute the gain. Formally:

$$Gain_n^{ETF} = \ Actual\ Data\ Test\ Score_n^{ETF} - Noise\ Test\ Score_n^{ETF}$$

Using this score, it is easy to compare the performance of each algorithm and choose the one with highest explanatory power.

Results

First, we compare the test scoring accuracy (Figure 1) and prediction gain criteria accuracy (Figure 2) of each algorithm on different horizons for ABC data set. In Figure 1 test score accuracy increase as the prediction horizon increase. We would expect such a pattern given that some of our ETFs examined has had a natural positive drift over the sample period examined. The gain criteria would adjust for such positive drifts. Notice that the gain criterion ranges from 0 to 35%, compared to that of the scoring accuracy which ranges from 0 to 100%

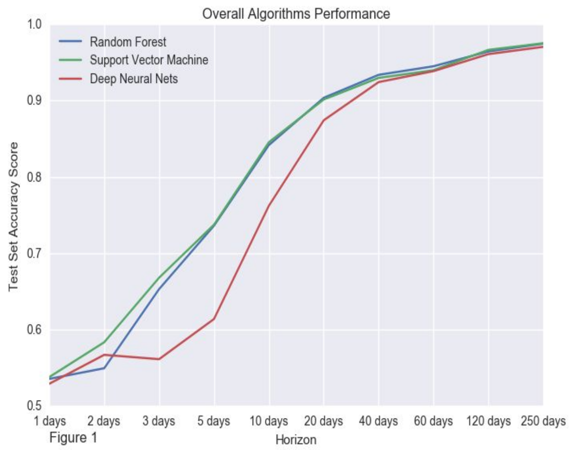

Figure 1

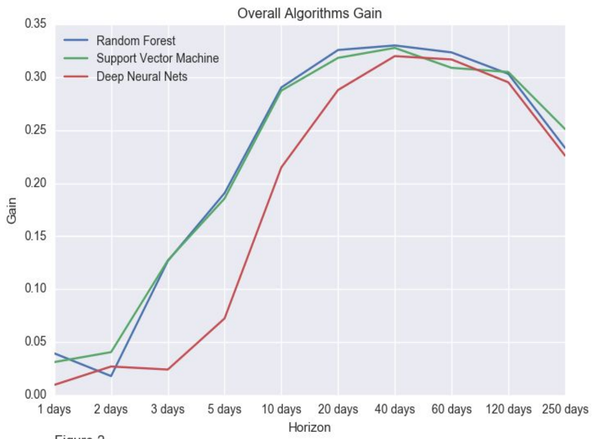

Figure 2

RF (blue line) and SVM (green line) shows a close results on all horizons in terms of test set accuracy and gain criterion, while DNN (red line) converges to the other algorithms at forty days or more. The reason for this may be due to the insufficient number of hidden layers or default values of other hyperparameters of DNN classifier. Overall, we see that even though test set accuracy on average is increasing with horizon, gain criterion peaks at forty days and steadily falls thereafter. It is caused by the fact that for some of ETFs, such as SPY, IWM, IYR and GLD, we see strong trend toward increases or decreases on the longer horizons (see Table A1), which increases the test set accuracy score, but lowers the gain criterion score.

Not surprisingly, the predictive power of algorithms increases with the forecast horizon. The result for gain criterion was anticipated, as from preprocessing data examination suggested that noise level will be increasing with horizon, thus decreasing gain. However, gain in predictive power for 120 and 250 days is still high, which raises the questions of why such long-term predictions using only technical analysis still have good results. In order to understand this phenomenon, more analysis is required with introduction of fundamental data, which is outside the scope of this paper.

Next step in our analysis is to see how algorithms performed in more details. Using gain criterion we compare performance of algorithms for each ETF from our list (See Appendix). Generally we see two patterns in gain behaviour with increase in returns time period. First is peak at ten to forty days level and fall at consequent horizons, second is increased up to twenty to sixty days and plateau with fluctuations or slight rising trend to 250 days. We believe the reasons for such behaviour are the same as described above.

The examples are SPY and EEM respectively:

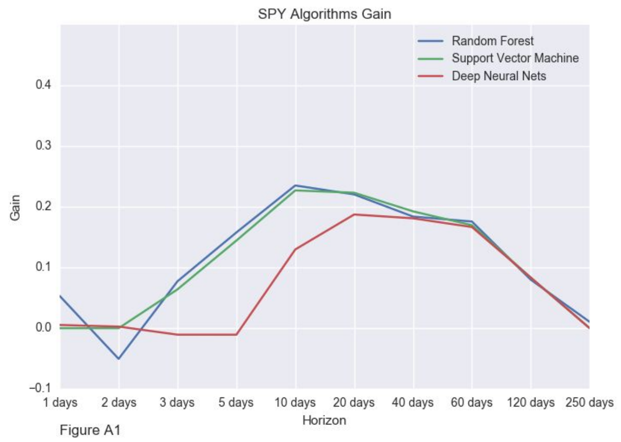

Figure A1

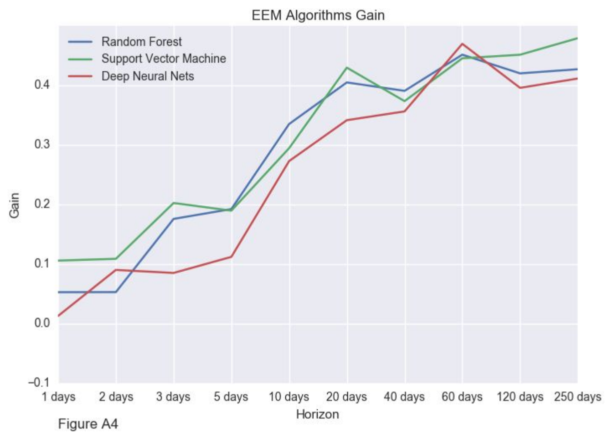

Figure A4

Since overall DNN performance was lower than those of two other on three to twenty days horizons, it is not surprising that on single ETF level it was showing the same pattern. Nonetheless, for some ETFs (i.e. Russell 2000 (IWM) and US Treasury Bonds 20yrs (TLT)) DNN algorithm was able to catch up to the rest at twenty days. SVM and RF show close results on all of ETFs, but the later seems to produce less volatile results with respect to horizon. We find that horizons of ten to sixty days to be most interesting across all ETFs. While for some instances 3, 5, 120 and 250 days periods also draw attention and deserve more rigorous analysis, our goal is to compare algorithms performances and to estimate possibility and feasibility of meaningful predictions rather than to investigate the specifics of prediction of individual ETF.

Next, we develop a deeper understanding of the explanatory variables and their significance in terms of predictive power. For that purpose we examine average test scores and gain criterion for A, B and C data sets for each algorithm and compare them to ABC results.

Let's start with RF displayed in Figure 3 & Figure 4. Volume (data set B) is effectively explains all the results obtained in previous sections. That was a surprising and unexpected result. Returns (data set A) have decent predictive power, however strong at the horizon of forty days and longer show the same performance as volume and combined data sets. Calendar dummies (data set C) however seem to explain a small portion of daily returns and monthly (20 days) returns. We assume it is because dummies are for day of the week and month. Nonetheless, predicatory power of this set is negligible and clear contribution can only be seen at one day horizon.

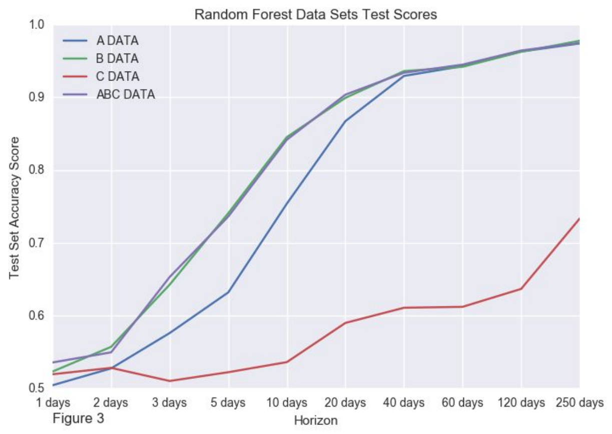

Figure 3

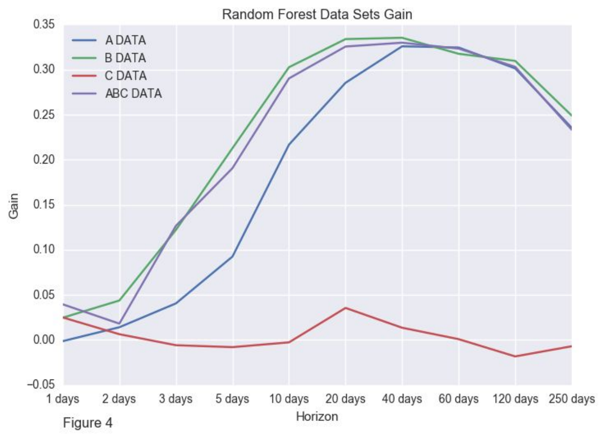

Figure 4

SVM results (Figure 5 & Figure 6) have the same pattern as RF. However, overall performance as well as gain for returns data set is closer to those of combined data set in comparison with RF. What is more interesting, combined data set seem to outperform individual data sets on one to three days horizons. Also, calendar dummy variables seem to yield better result, but it is still not big enough to be significant and doesn't add any predictive power to combined data set.

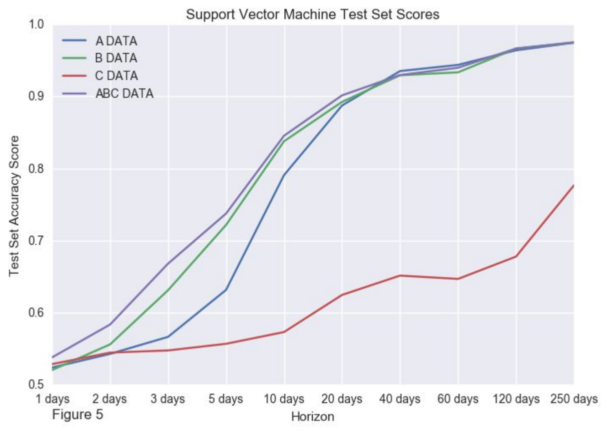

Figure 5

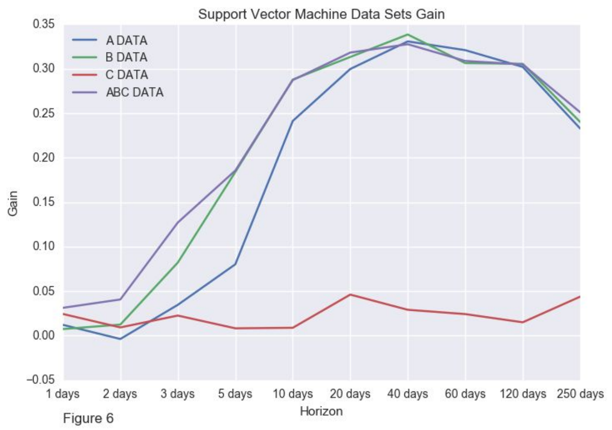

Figure 6

As mentioned earlier, DNN (Figure 7 & Figure 8) struggle to show competitive results on horizons less than twenty to forty days. One to five days horizons predictions have effectively no predictive power. Apart from other algorithms, DNN benefits from combination of data sets. However, gradation of data sets with respect to gain is the same as for previous algorithms, as well as patterns of change in gain and test score with varying horizon.

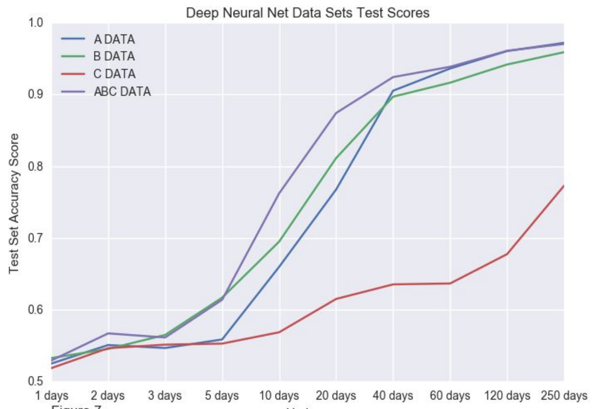

Figure 7

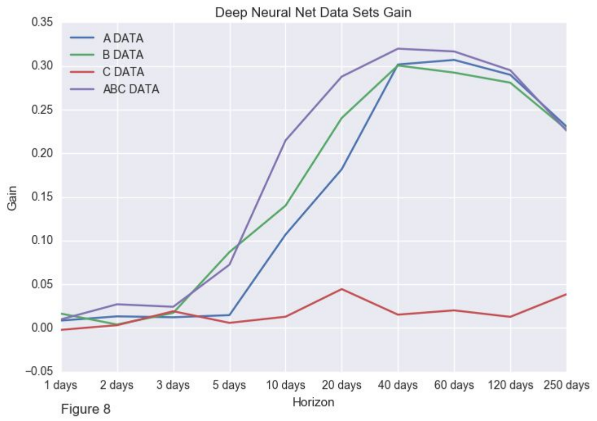

Figure 8

The results of DNN on ten to sixty days horizons suggest that there is possibility of improvement of algorithm prediction with combination of data sets, which is not the case for other algorithms. Overall, we see poor ability to predict short term returns for all algorithms. The solution to boost results might be ensembling of algorithms, but such analysis is beyond the scope of this paper.

As mentioned earlier, we find horizons from ten through sixty days most interesting in terms of predictive power. Thus we will examine performance of the algorithms on these time periods in more detail using Receiver operator characteristic (ROC). That will allow to compare algorithms from different angle. Based on ROC, we can compute another measure for algorithm comparison - ROC area under curve (AUC). We generated ROC curves for horizon ten to sixty days (See Appendix). The results are following the same pattern as in all previous sections. For example, let see the ROC graphs for EEM (Figure A14, Figure A24 & Figure A34). We also calculated AUC for each of selected horizons, ETF and algorithm (Table A2 through Table A4).

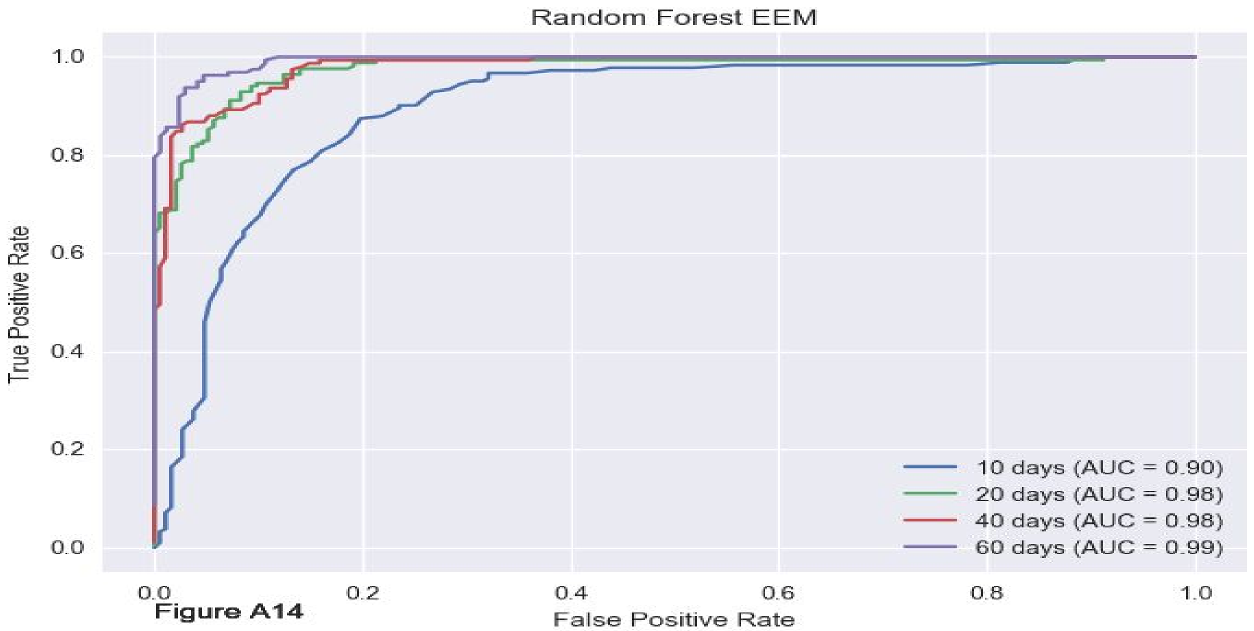

Figure A14

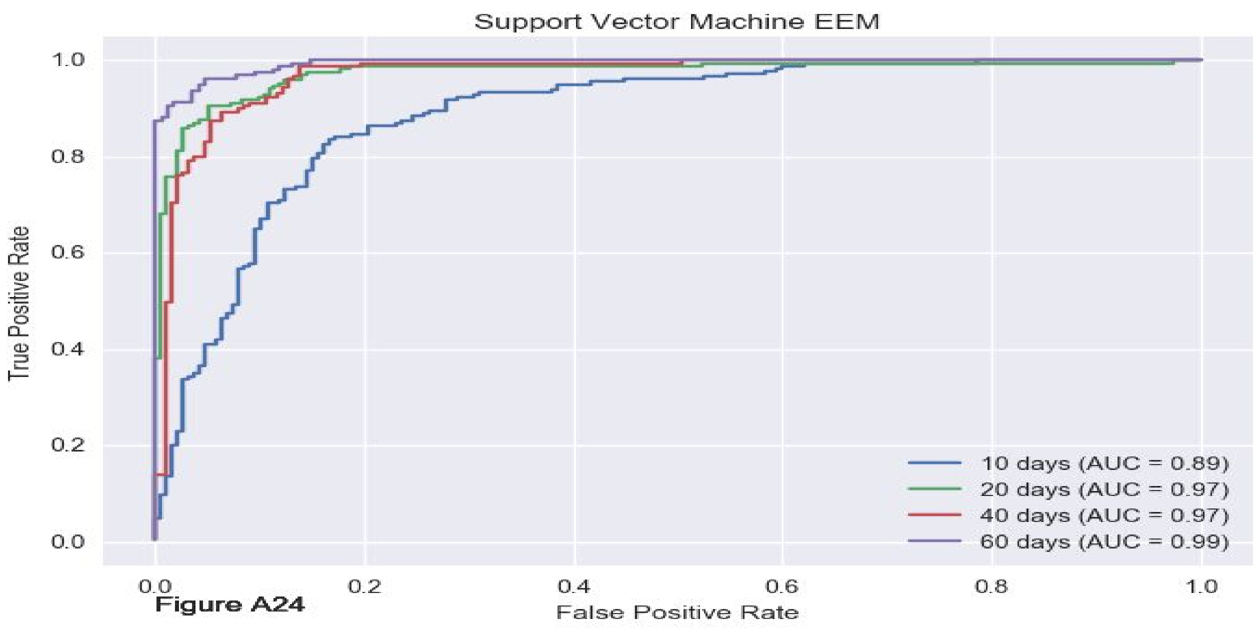

Figure A24

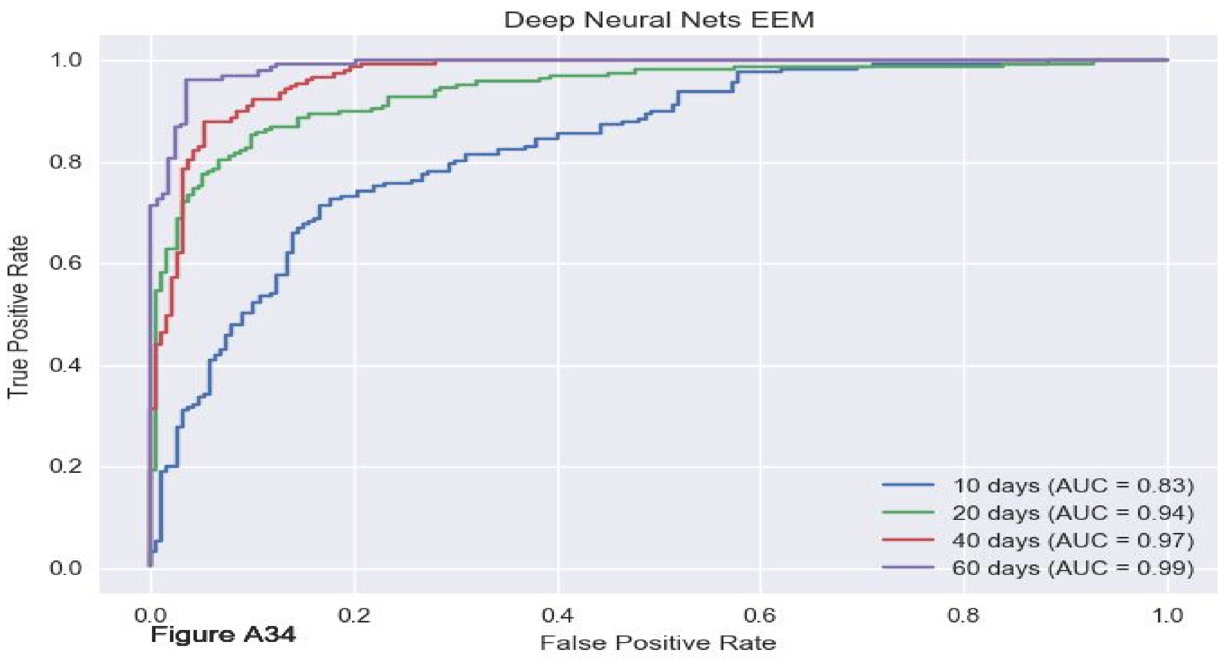

Figure A34

Longer horizon ROC curves have almost ideal form and AUC close to one, which suggest high predictive ability with little mistake. Altogether, we can conclude that predictions for this ETFs are possible.

Feature Importance with Random Forest

We also try to shed some light on what data drives the performance of algorithms by assessing feature importance with random forests. As previously discussed, volume is a good predictor by itself, but in combination with returns is more powerful. However, it is unclear which features are actually driving the performance of the algorithms. In case of SVM, it is only possible interpret weights of each feature if kernel is linear. For DNN, it is hard to explain and grasp what is the relationships in hidden layers. But for RF we can compute and interpret the importance of each feature in easy way.

We decided to examine twenty days RF features importance for ETFs (Figure 9). The results show that there is no single feature which would explain most of returns. Note that on graph below, features ares sorted in descending order for each ETF. As one can see, the pattern is same for all ETFs, meaning that almost all information in data set is contributing and useful for prediction, With features importance structured this way, we see that there is not a single factor which contributes more than 1.6%. However, data set also contains lagged variables The question which immediately arises is what if group or groups of features (i.e. volume of SPY and returns on EEM) are more beneficial, and we can drop them.

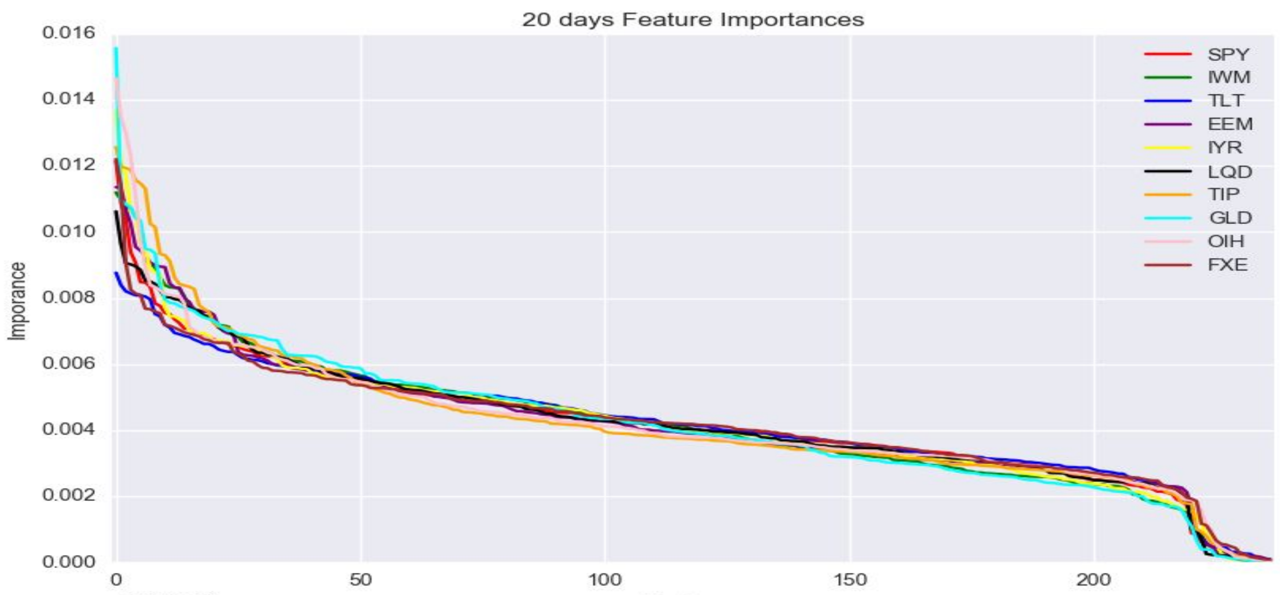

Figure 9

Table 1. 10 Features with highest importance by ETF

	Equity			Fixed Income			Commodities and Real Estate			
	SPY	IWM	EEM	TLT	TIP	LQD	IYR	GLD	OIH	FXE
1	Lag10_Vol_GLD	Vol_TLT	Lag1_Vol_OIH	Lag10_Vol_SPY	Lag6_Vol_LQD	Lag10_Vol_TIP	Lag3_GLD	Lag10_Vol_SPY	Vol_OIH	Lag1_Vol_TIP
2	Lag4_Vol_IWM	Lag10_Vol_GLD	Lag1_SPY	Lag9_Vol_SPY	Lag7_Vol_LQD	Lag10_Vol_GLD	GLD	Lag1_Vol_TIP	Lag2_Vol_OIH	Vol_TIP
3	Lag9_Vol_EEM	Lag10_Vol_EEM	SPY	Lag7_Vol_SPY	Lag5_Vol_LQD	Lag4_Vol_LQD	Lag2_GLD	Lag3_Vol_LQD	Lag3_Vol_OIH	Lag1_OIH
4	SPY	Lag9_Vol_EEM	Vol_OIH	Lag8_Vol_SPY	Lag10_Vol_LQD	Lag7_Vol_LQD	Lag9_Vol_IWM	Lag4_Vol_TIP	Lag1_Vol_OIH	Lag2_Vol_TIP
5	Lag3_Vol_IWM	Lag1_Vol_TLT	Lag9_Vol_GLD	Lag6_Vol_IWM	Lag8_Vol_LQD	Lag5_Vol_LQD	Lag1_GLD	Lag2_Vol_TIP	Lag10_FXE	Lag5_Vol_OIH
6	Lag10_Vol_OIH	Lag8_Vol_LQD	Lag7_Vol_GLD	Lag4_Vol_SPY	Lag9_Vol_LQD	Lag7_LQD	Lag10_Vol_IYR	Vol_TIP	Lag5_Vol_OIH	Lag5_Vol_TIP
7	Lag10_Vol_EEM	Lag1_Vol_IWM	Lag5_Vol_TIP	Lag1_Vol_FXE	Lag7_Vol_IYR	Vol_LQD	Lag7_LQD	Lag8_Vol_SPY	Lag9_Vol_EEM	Lag1_Vol_LQD

8	Vol_SPY	Lag8_Vol_GLD	Lag10_Vol_GLD	Lag6_Vol_SPY	Lag3_Vol_TIP	Lag8_Vol_LQD	Lag9_Vol_LQD	GLD	Lag4_Vol_OIH	Vol_LQD		
9	Lag6_Vol_EEM	Lag9_Vol_TIP	Lag6_Vol_TIP	Lag5_Vol_SPY	Lag3_Vol_LQD	Lag4_LQD	Lag10_Vol_LQD	Lag10_GLD	Lag6_Vol_OIH	Lag8_Vol_GLD		
10	Vol_TLT	Lag9_Vol_GLD	Lag8_Vol_GLD	Lag5_Vol_IWM	Vol_LQD	Lag1_Vol_LQD	Lag10_Vol_GLD	Lag5_Vol_OIH	Vol_EEM	Lag2_OIH		

For that purpose we grouped returns and volumes for each ETF and summed up importance within each group (Figure 10). Volume is more important than returns, which confirms the difference of results for A and B data set. One of the reasons for such behaviour might be a relationship between volume and returns; we assume that might be the result of relationship obtained by Cheng, Hong, Stein (2001) and Chordia, Swaminathan (2000) in a sense that past volume is a good predictor of future returns skewness and patterns. Calendar dummies show small to no influence on predictions, as expected from data sets results.

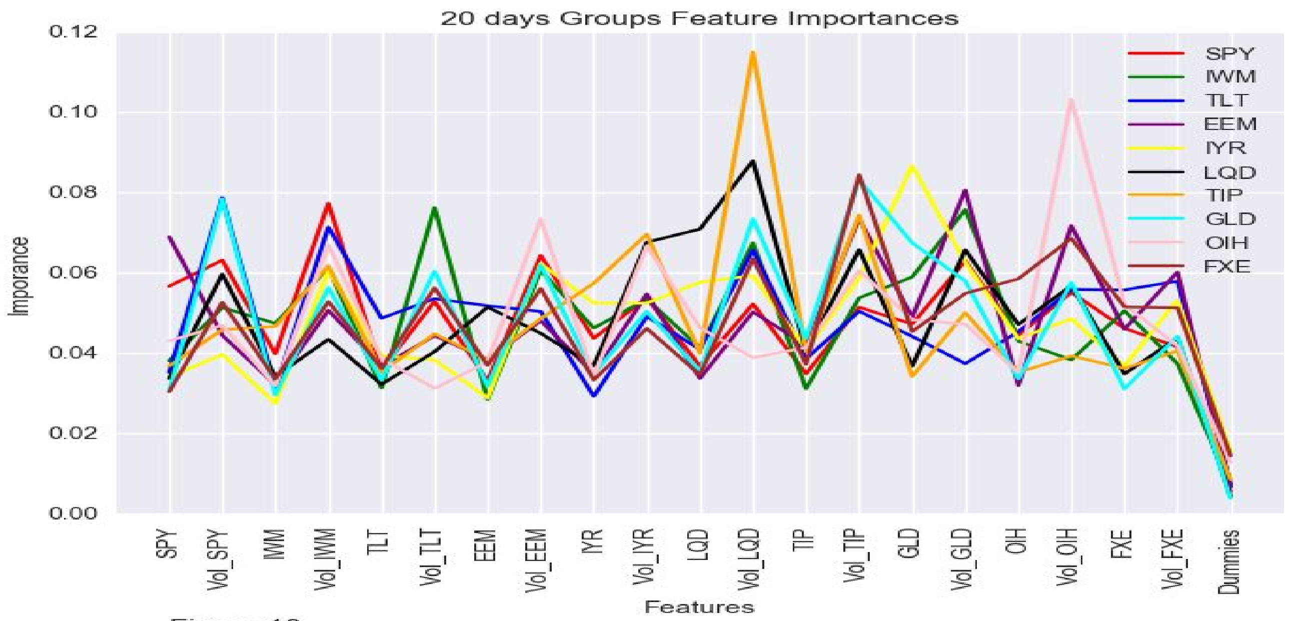

Figure 10

Conclusions

In this work we examined three popular machine learning algorithms ability to predict ETFs returns. While we restricted our initial analysis to only the direction of the future price movements, we still procured valuable results. First, machine learning algorithms do a good job of predicting price changes at the ten days to sixty days horizon. Not surprisingly, these algorithms fail to predict returns in the short-term horizons five days or shorter. We introduce our “gain” measure to help assess the efficacy across algorithms and across horizons. We also segmented our input features variables into different information sets so as to cast our research in the framework of the efficient markets hypothesis. We find that the volume information set (B) works extremely well across our three algorithm. Moreover, we find that the most important predictive features varies depending on the ETFs that’s being predicted. Financial intuition helps understand the prediction variables with complex relationship embedded within the prediction of S&P500, as proxied by

SPY, requiring more diverse set of features compared to the complexity of the top feature set needed to explain GLD or OIH.

In practice, the information set could be vastly extended to include other important features such as social media along the lines of Liew and Badavari (2016) who have identified the social media factor. Additionally, the forecasting time horizons could have been even further extended beyond one trading year as well as shortened to examine intraday performance. However, we leave this more ambitious research agenda to future work.

One interesting application is to use several different horizon models launched at staggered time within the day, thereby gaining slight diversification benefits to the resultant portfolio of strategies.

In sum, we hope that our application of machine learning algorithm motivates others to move this body of knowledge forward. These algorithms possess great potential in their applications to the many problems in finance.

Bibliography

Asness, C., 1995, “The Power of Past Stock Returns to Explain Future Stock Returns,” Applied Quantitative Research.

Asness, C., Moskowitz, T., and Pedersen, L., 2013, “Value and Momentum Everywhere”, The Journal of Finance 68(3), 929-985.

Breiman, L., 2001, “Random Forest,” Machine Learning. 45 (1): p.5-32. DOI: 10.1023/A:1010933404324

Chen, J., Hong, H. and Stein J.C., 2001, “Forecasting crashes: trading volume, past returns, and conditional skewness in stock prices”, Journal of Financial Economics 61(3), 345-381.

Chordia T. and Swaminathan, B., 2000, “Trading Volume and Cross-Autocorrelations in Stock Returns”, The Journal of Finance 55(2), 913-935.

Fama, E. and French, K., 1992, “The Cross-Section of Expected Stock Returns,” Journal of Finance 47, 427-465.

Fama, E. and French, K., 1993, “Common Risk Factors in the Returns on Stocks and Bonds,” Journal of Financial Economics 33, 3-56.

Fama, E. and French, K., 1995, “Size and Book-to-Market Factors in Earnings and Returns,” Journal of Finance 50, 131-155.

McCulloch, W.S. and Pitts, W., 1943, “A Logical Calculus of the Ideas Immanent in Nervous Activity.” The bulletin of mathematical biophysics, 5(4):115-133.

Jegadeesh, N., and Titman, S., 1993, “Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency,” Journal of Finance 48, 65-91.

Lakonishok, J., Shleifer, A., and Vishny, R., 1994, “Contrarian Investment, Extrapolation, and Risk,” The Journal of Finance 49(5), 1541-1578.

Keim, D., 1983, “Size-Related Anomalies and Stock Return Seasonality: Further Empirical Evidence”, Journal of Financial Economics 12(1), 13-32.

Liew, J., and Vassalou, V., 2000, “Can Book-to-Market, Size, and Momentum Be Risk Factors That Predict Economic Growth,” Journal of Financial Economics, 57, 221-245.

Liew, J., and Budavari, T., 2017, “The ‘Sixth’ Factor - Social Media Factor Derived Directly from Tweet Sentiments“, forthcoming Journal of Portfolio Management

Raschka, S., 2015, *Python Machine Learning*, PACKT Publishing, ISBN 978-1-78355-513-0

Roll, R., 1976, “A Critique of the Asset Pricing Theory’s Tests,” Journal of Financial Economics, 4, 129-176.

Rosenblatt, F., 1957, “The Perceptron, a Perceiving and Recognizing Automaton.” Cornell Aeronautical Laboratory.

Rouwenhorst, K. 1998, “International Momentum Strategies,” The Journal of Finance 53, 267-284.

Shleifer, A., and Vishny, R., 1997, “The Limits of Arbitrage,” The Journal of Finance 52 (1), 35-55.

Vapnik, V., 1995, The Nature of Statistical Learning Theory, Springer-Verlag, ISBN 0-387-98780-0

Widrow, B. and Hoff, T., 1960, “Adaptive ‘Adaline’ neuron using chemical ‘memistors’.” Number Technical Report 1553-2. Stanford Electron. Labs. Stanford, CA, October.

Appendix

Table A1. Proportion of upward price movements in test set

	1 day	2 days	3 days	5 days	10 days	20 days	40 days	60 days	120 days	250 days
SPY	0.584	0.609	0.629	0.607	0.603	0.678	0.741	0.779	0.896	0.969
IWM	0.562	0.612	0.581	0.575	0.568	0.617	0.670	0.688	0.764	0.932
TLT	0.504	0.559	0.536	0.532	0.557	0.579	0.589	0.618	0.573	0.661
EEM	0.507	0.516	0.544	0.529	0.495	0.468	0.457	0.488	0.479	0.490
IYR	0.520	0.593	0.624	0.634	0.578	0.617	0.557	0.618	0.747	0.917
LQD	0.541	0.604	0.568	0.586	0.624	0.631	0.693	0.733	0.684	0.745
TIP	0.469	0.559	0.525	0.516	0.551	0.579	0.598	0.585	0.524	0.349
GLD	0.493	0.524	0.512	0.505	0.514	0.424	0.382	0.370	0.205	0.047
OIH	0.496	0.532	0.525	0.503	0.514	0.474	0.529	0.533	0.556	0.484
FXE	0.488	0.484	0.501	0.503	0.481	0.457	0.437	0.406	0.403	0.411
Mean	**0.516**	**0.559**	**0.555**	**0.549**	**0.548**	**0.552**	**0.565**	**0.582**	**0.583**	**0.601**

For each horizon and ETF we compute proportion of upward movements in price in our test set. One can see strong upward trend with increase of horizon for SPY, IWM and IYR and downward trend for GLD.

Table A2. Random Forest ROC AUC

	1 days	2 days	3 days	5 days	10 days	20 days	40 days	60 days	120 days	250 days
SPY	0.579	0.563	0.704	0.791	0.878	0.951	0.967	0.980	0.999	0.996
IWM	0.516	0.553	0.727	0.804	0.919	0.953	0.967	0.977	0.998	0.986

TLT	0.516	0.514	0.682	0.804	0.906	0.974	0.986	0.992	0.992	0.999
EEM	0.506	0.618	0.745	0.801	0.896	0.975	0.980	0.993	0.991	0.990
IYR	0.518	0.582	0.697	0.819	0.934	0.951	0.992	0.986	1.000	0.981
LQD	0.539	0.504	0.672	0.843	0.935	0.981	0.980	0.992	0.995	0.993
TIP	0.496	0.563	0.721	0.807	0.923	0.962	0.983	0.980	0.991	0.999
GLD	0.536	0.586	0.708	0.792	0.934	0.945	0.982	0.995	0.974	0.997
OIH	0.482	0.560	0.691	0.793	0.907	0.972	0.996	0.987	0.988	1.000
FXE	0.571	0.553	0.662	0.804	0.928	0.962	0.980	0.994	0.996	1.000

Table A3. Support Vector Machine ROC AUC

	1 days	2 days	3 days	5 days	10 days	20 days	40 days	60 days	120 days	250 days
SPY	0.521	0.440	0.755	0.817	0.895	0.959	0.966	0.986	0.997	0.987
IWM	0.542	0.548	0.722	0.825	0.918	0.944	0.957	0.982	0.998	0.953
TLT	0.547	0.587	0.726	0.794	0.907	0.977	0.970	0.975	0.988	0.999
EEM	0.558	0.600	0.775	0.786	0.889	0.974	0.970	0.993	0.985	0.990
IYR	0.478	0.618	0.702	0.816	0.927	0.953	0.993	0.987	0.999	0.967
LQD	0.527	0.497	0.668	0.819	0.937	0.971	0.974	0.993	0.998	0.978
TIP	0.532	0.600	0.709	0.820	0.933	0.955	0.971	0.978	0.983	1
GLD	0.542	0.621	0.752	0.822	0.927	0.936	0.986	0.992	0.977	0.987
OIH	0.506	0.578	0.737	0.801	0.903	0.966	0.998	0.973	0.988	1
FXE	0.515	0.619	0.695	0.827	0.921	0.959	0.971	0.991	0.991	0.996

Table A4. Deep Neural Nets ROC AUC

	1 days	2 days	3 days	5 days	10 days	20 days	40 days	60 days	120 days	250 days
SPY	0.469	0.524	0.509	0.615	0.782	0.922	0.965	0.982	0.994	0.996

IWM	0.462	0.524	0.527	0.649	0.812	0.927	0.960	0.979	0.995	0.987
TLT	0.545	0.589	0.547	0.602	0.803	0.959	0.981	0.988	0.990	0.999
EEM	0.563	0.585	0.573	0.611	0.835	0.940	0.969	0.989	0.987	0.985
IYR	0.441	0.506	0.588	0.716	0.843	0.937	0.980	0.984	0.999	0.974
LQD	0.510	0.493	0.538	0.642	0.875	0.956	0.968	0.992	0.996	0.981
TIP	0.565	0.532	0.520	0.658	0.839	0.945	0.975	0.978	0.988	0.999
GLD	0.450	0.611	0.619	0.646	0.856	0.929	0.978	0.991	0.972	1
OIH	0.496	0.523	0.544	0.645	0.844	0.932	0.992	0.980	0.990	1
FXE	0.530	0.560	0.582	0.658	0.843	0.925	0.967	0.992	0.993	0.996

ETF Algorithms Gain

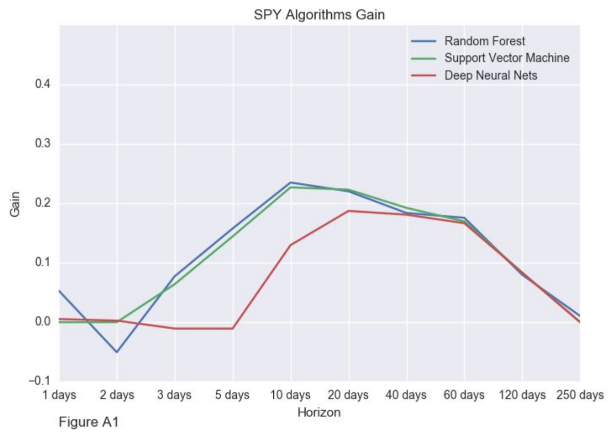

Figure A1

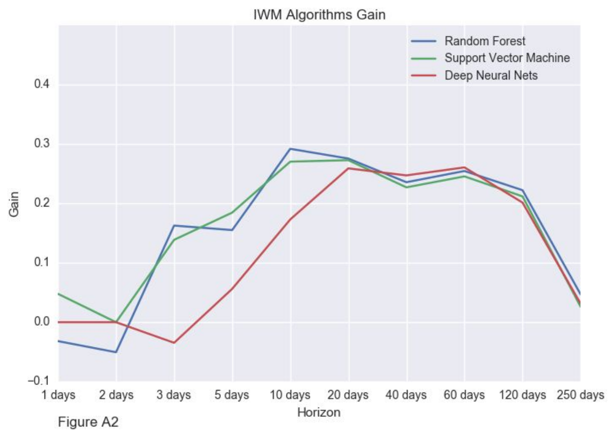

Figure A2

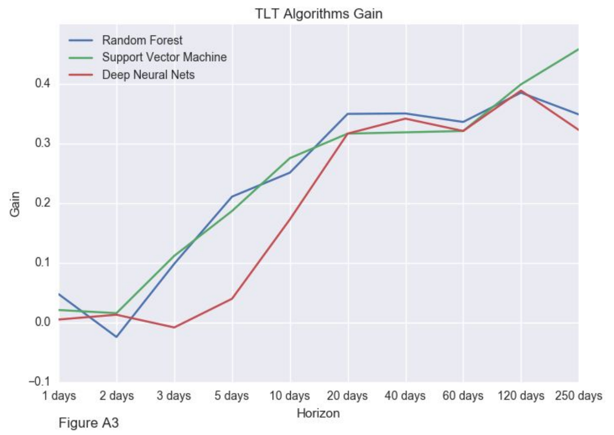

Figure A3

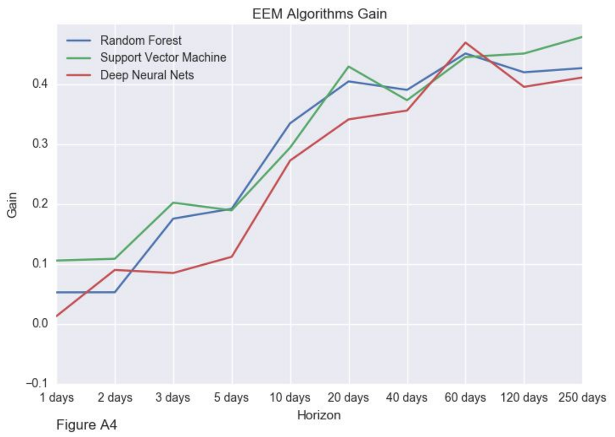

Figure A4

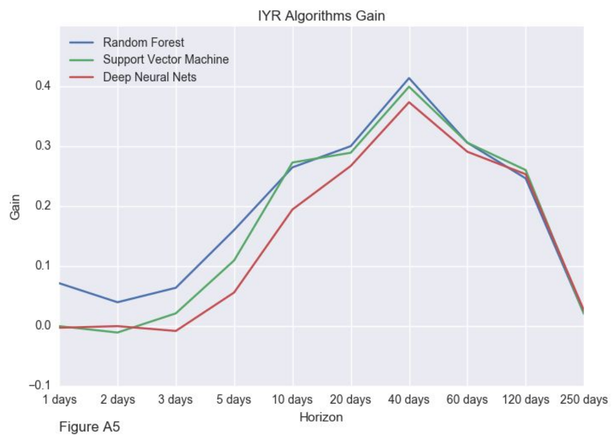

Figure A5

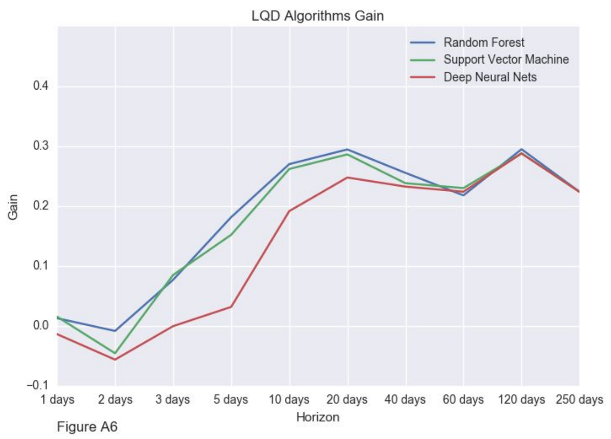

Figure A6

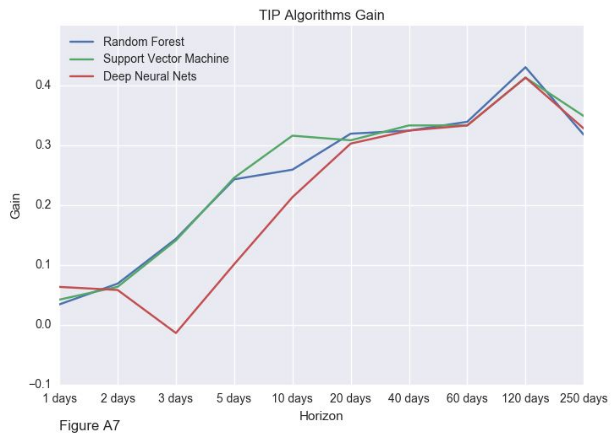

Figure A7

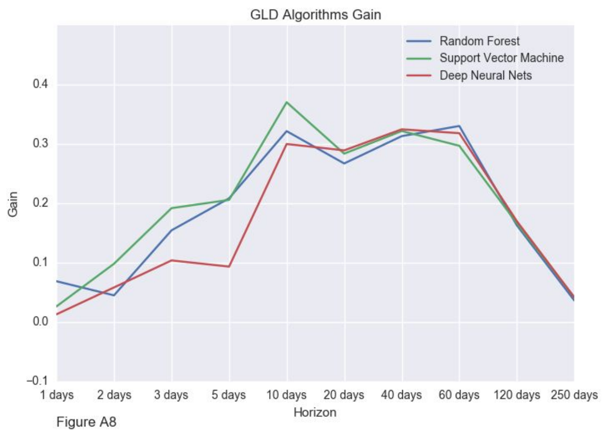

Figure A8

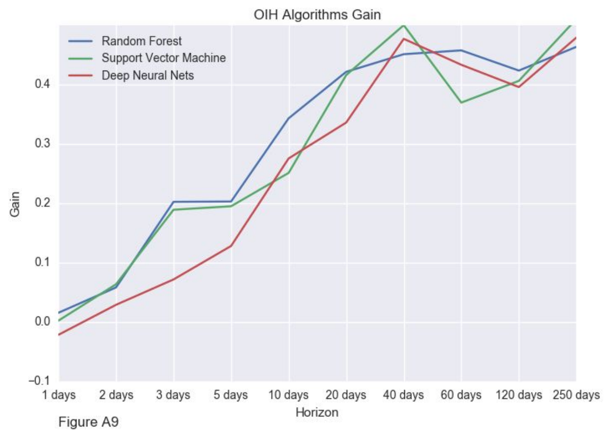

Figure A9

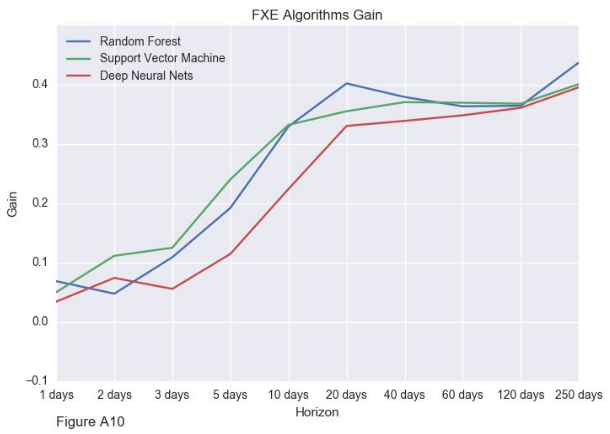

Figure A10

ETFs ROC

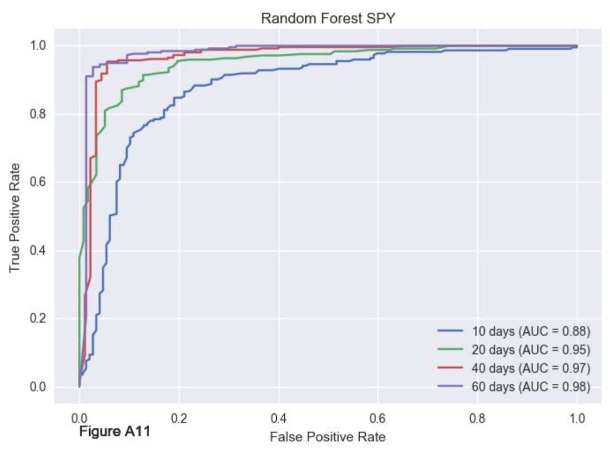

Figure A11

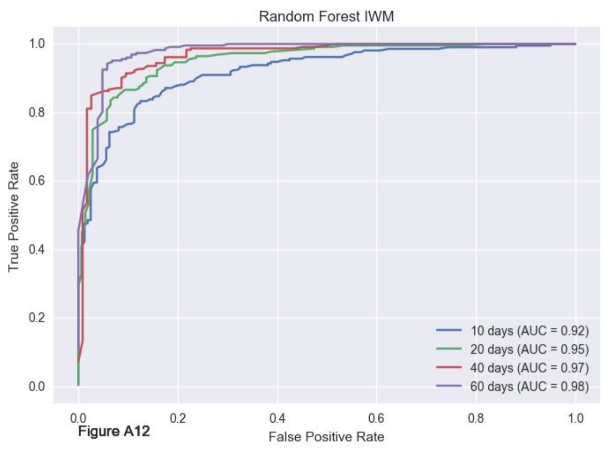

Figure A12

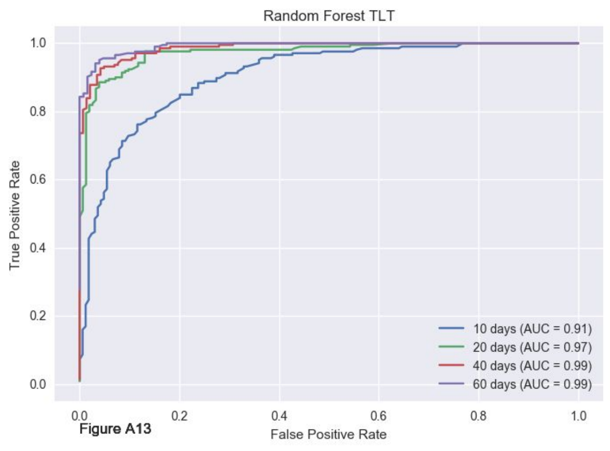

Figure A13

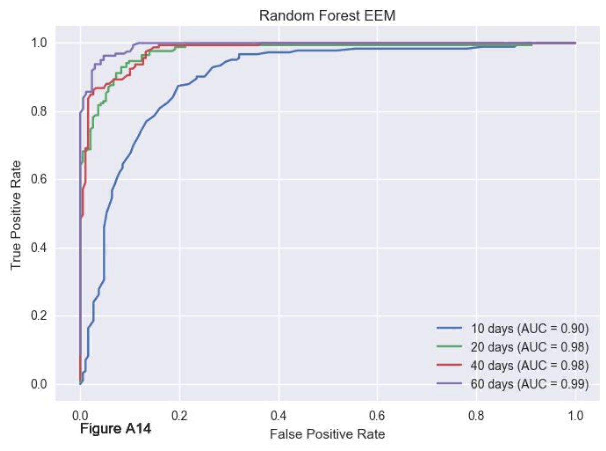

Figure A14

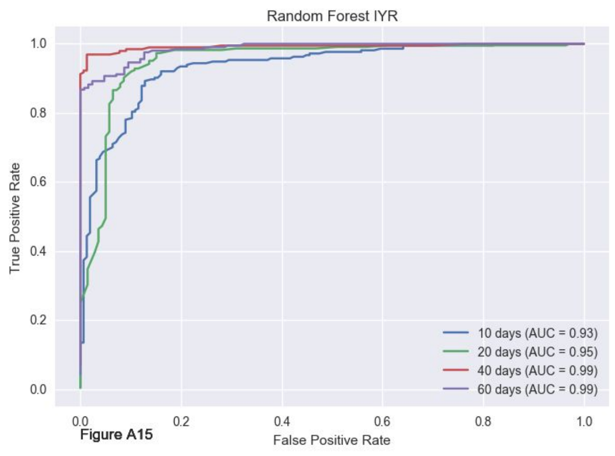

Figure A15

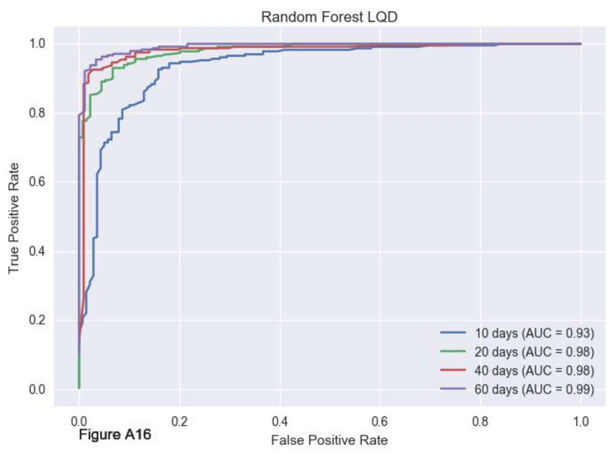

Figure A16

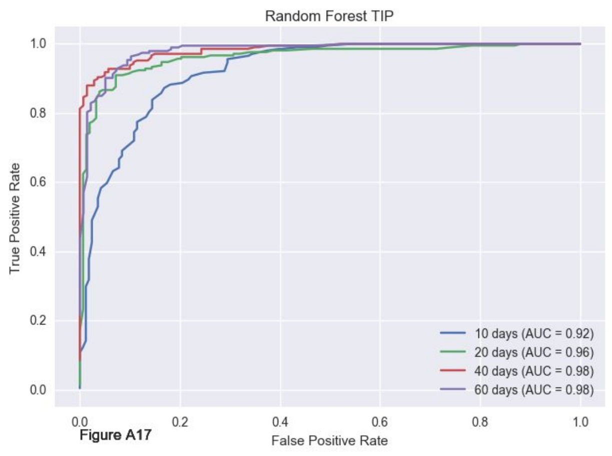

Figure A17

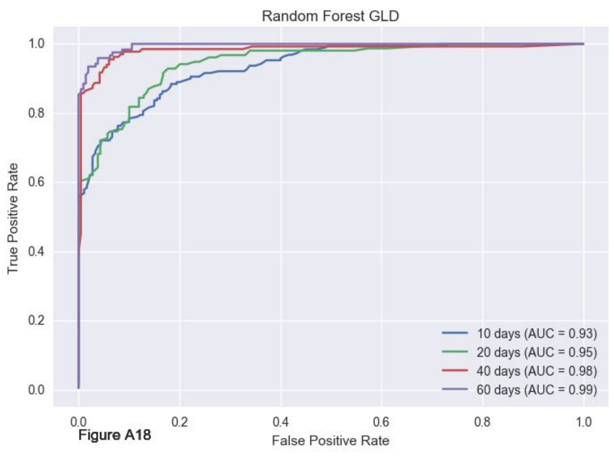

Figure A18

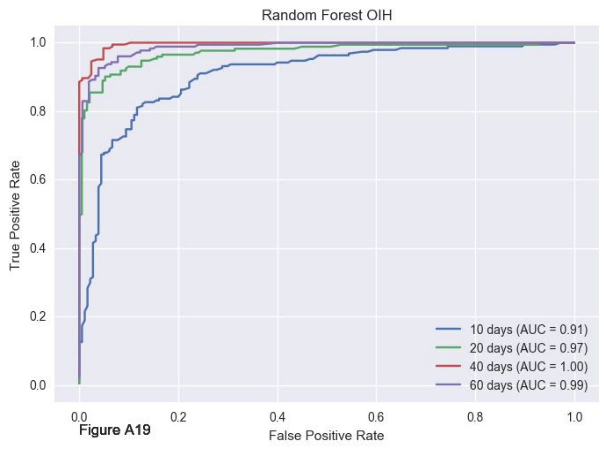

Figure A19

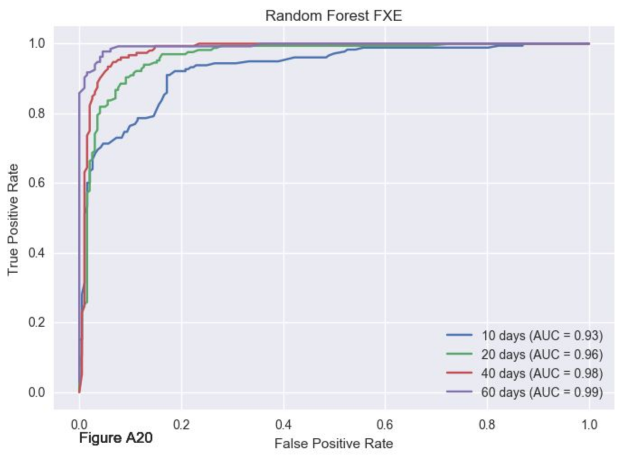

Figure A20

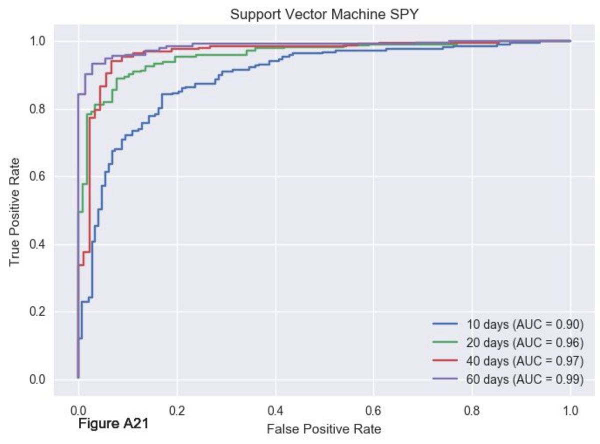

Figure A21

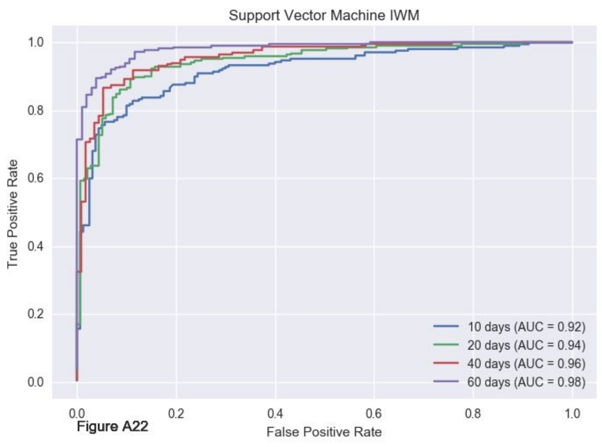

Figure A22

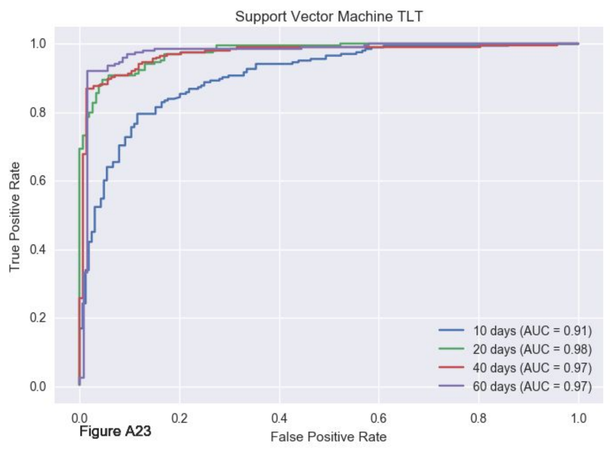

Figure A23

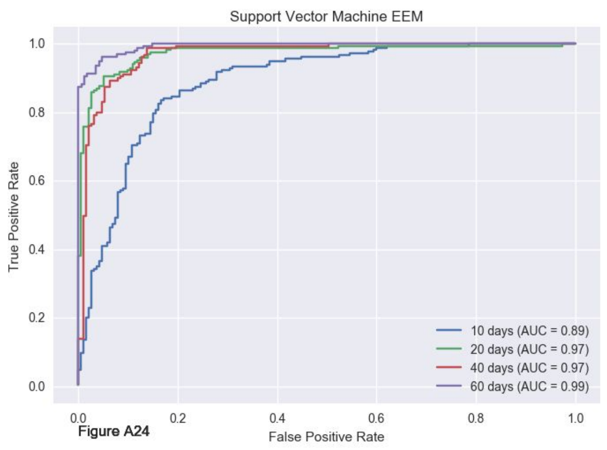

Figure A24

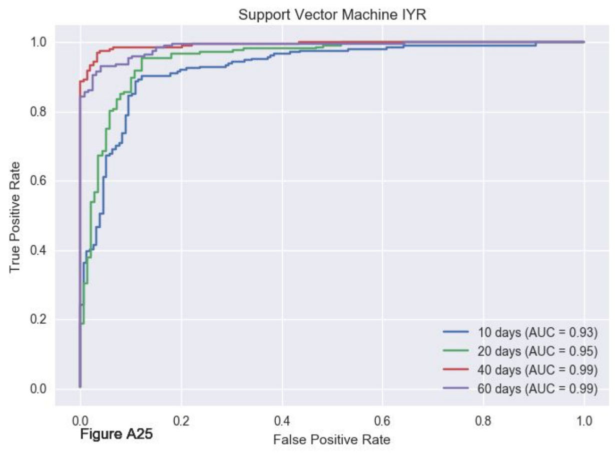

Figure A25

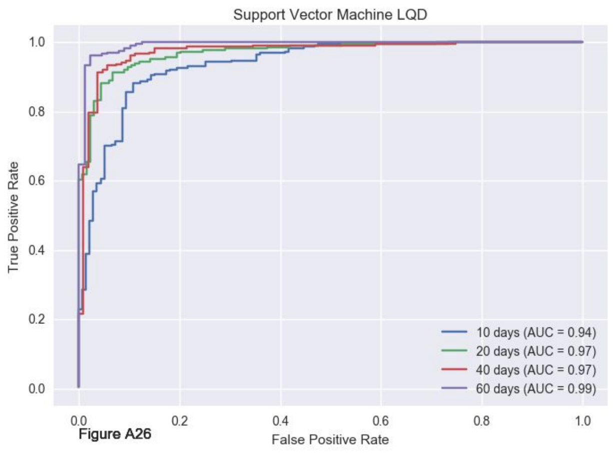

Figure A26

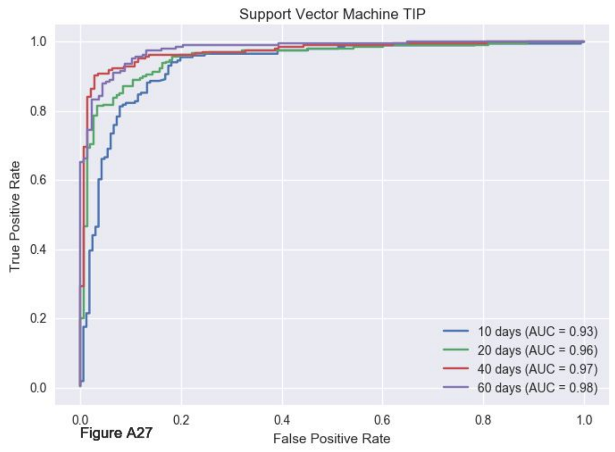

Figure A27

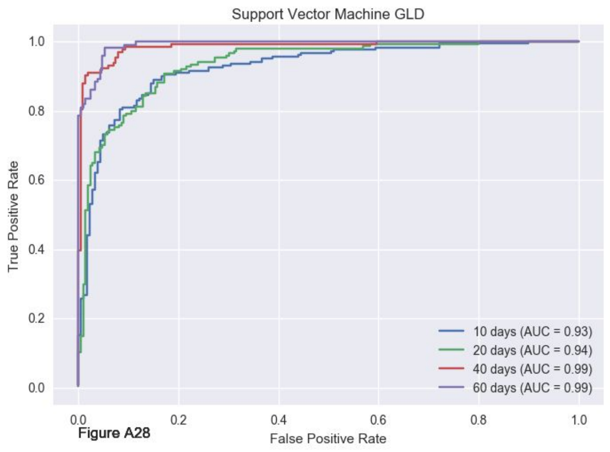

Figure A28

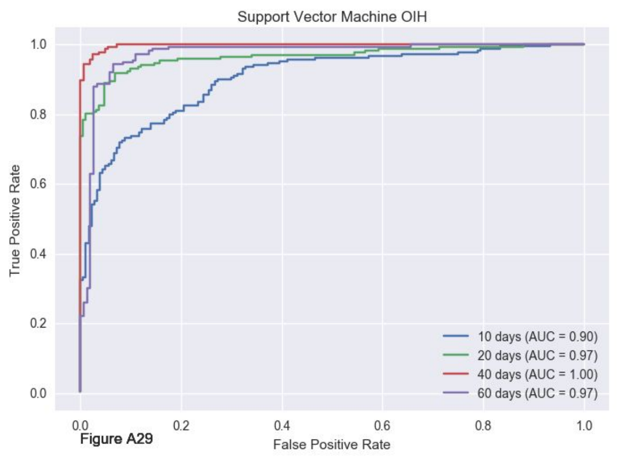

Figure A29

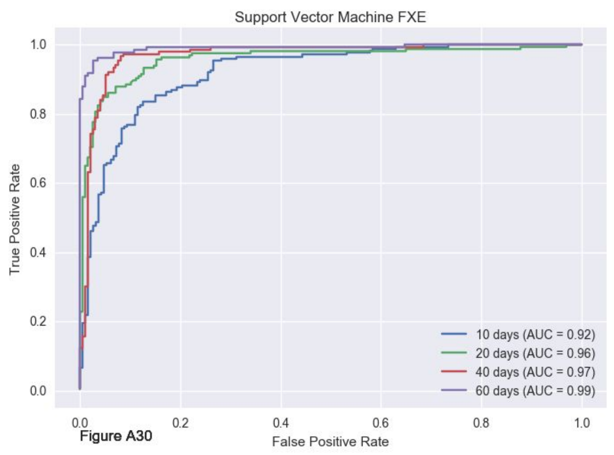

Figure A30

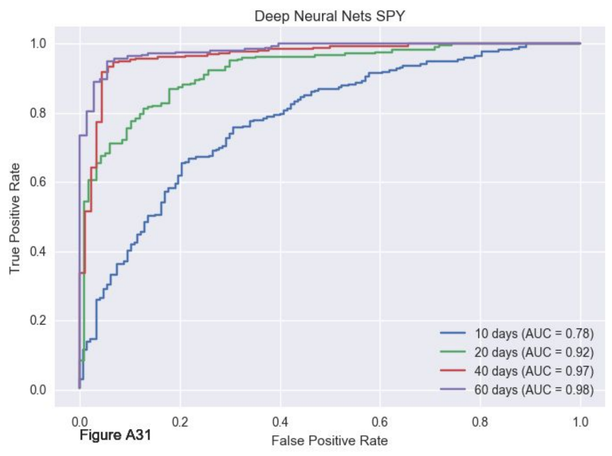

Figure A31

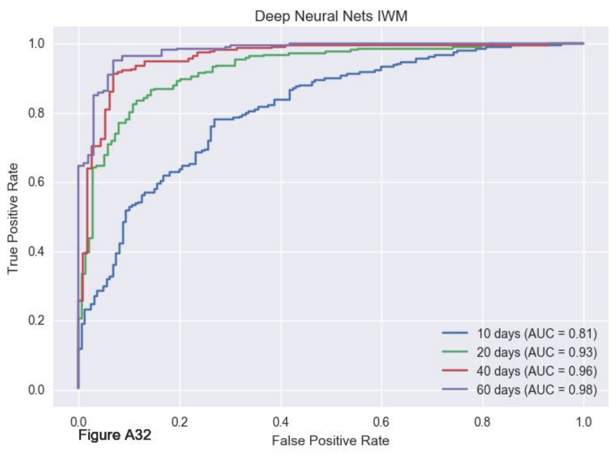

Figure A32

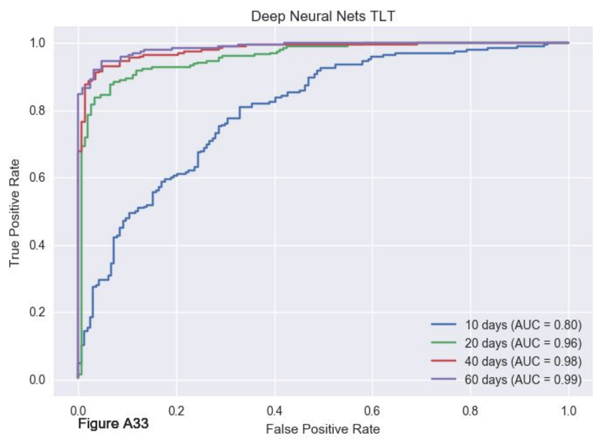

Figure A33

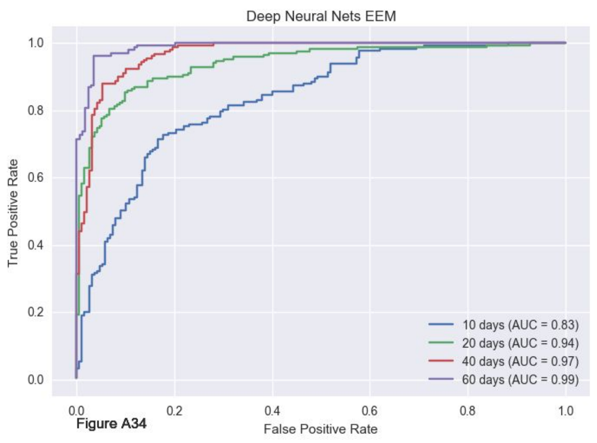

Figure A34

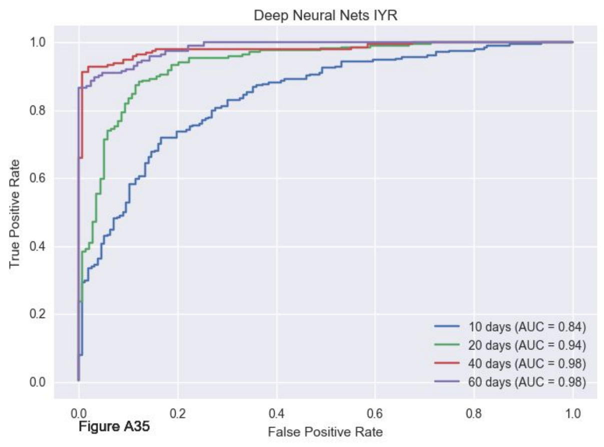

Figure A35

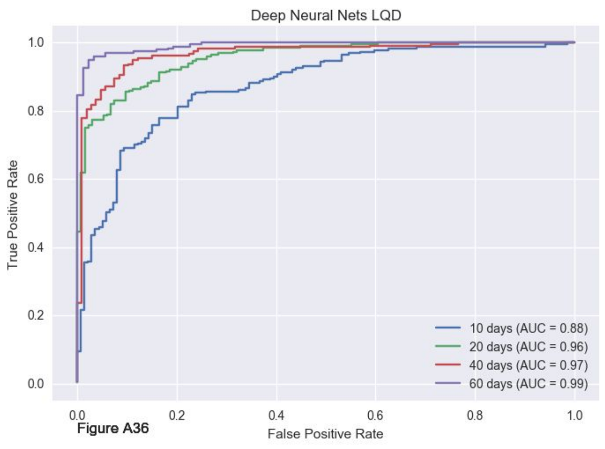

Figure A36

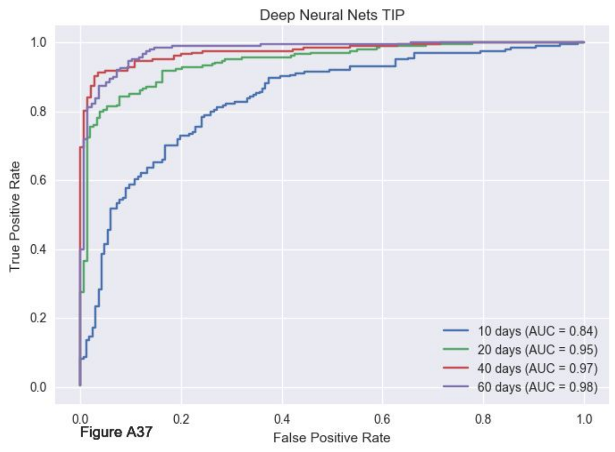

Figure A37

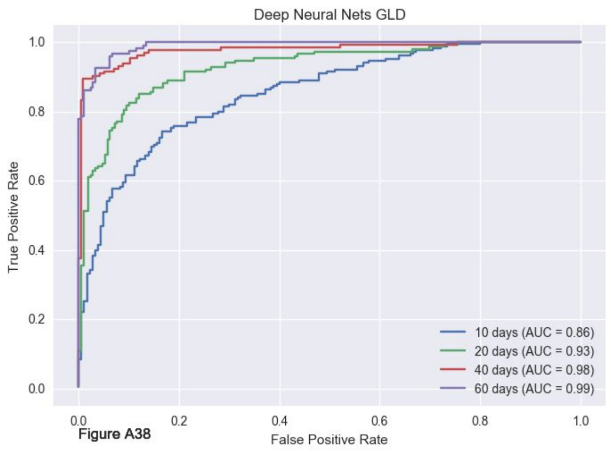

Figure A38

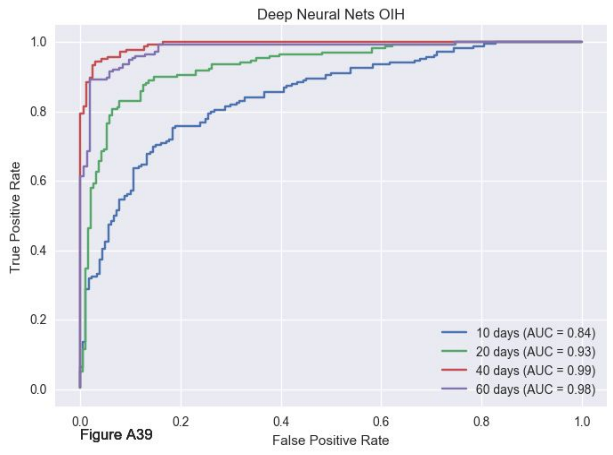

Figure A39

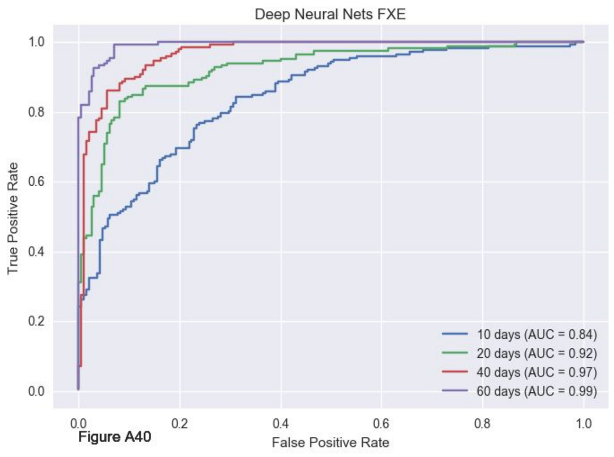

Figure A40