


Stock Market Prediction

 ... reviewing machine learning literature about security price prediction and risk quantification in the market

Machine Learning Algorithms for financial Asset Price Forecasting

- Goal : compares and contrast implementations of modern Machine Learning algorithms on high performance computing (HPC) infrastructures versus the traditional and highly popular Capital Asset Pricing Model (CAPM) on U.S equities data
- Link : <https://arxiv.org/pdf/2004.01504v1.pdf> (03/2020)
- Code : -
- Credible source: -

Model

Capital Asset Pricing Model (CAPM):	Model Assumptions
<p>developed in the 1960's building on the earlier work of Markovitz and his mean-variance and market portfolio models. It's a one-factor model that assumes the market risk or beta is the only relevant metric required to determine a expected rate of return for any asset or project.</p> <p>The CAPM provides an equilibrium relationship between the investments expected return and market beta:</p> <div>$E[r_i] = r_f + \beta_i(E[r_M] - r_f).$</div>	<div><div>1. One-period investment model.</div><div>2. Risk averse investors: This assumption was initially developed by Markovitz and asserts that all are rational and risk averse actors in the sense that when choosing between financial portfolios inv aim to optimize the following:<div><div>(a) Minimize the variance of the portfolio returns.</div><div>(b) Maximize the expected returns given the variance.</div></div></div><div>3. Zero transaction costs: There are no taxes or transactional costs.</div><div>4. Homogenous information: All investors have homogenous views and information regarding the distributions of all security returns.</div><div>5. Risk free rate of interest: All investors can lend and borrow at the at a specified risk free rate of</div></div>
Machine Learning Algorithms:	<div><div>• To solve a non-linear multivariate time series problem, by training historical data and make pr</div><div>• Uses Classification and regression testing to test the training data</div><div>• By including optimization and numerical analysis that reduces the problem of overfitting, whic</div></div>

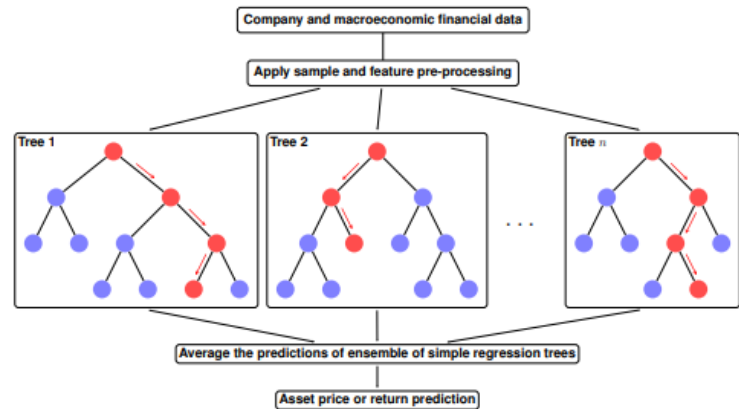
Gradient Boosting Tree Algorithms:

1. first pre-process the data in a process known as sample and feature bagging
2. Then These base learners are aggregated through a function such as an average to produce asset price prediction
3. The final function is estimated through the minimization of the pre-defined loss function descr

$$\hat{f}(X) = \arg \min_{f(X)} \mathcal{L}(f(X), y)$$

the below:

FIGURE 1: EXAMPLE OF GRADIENT BOOSTED TREE ARCHITECTURE



Regulations and Financial Constraints:

1. Transparency and Explainability
2. Risk Management

Solutions:

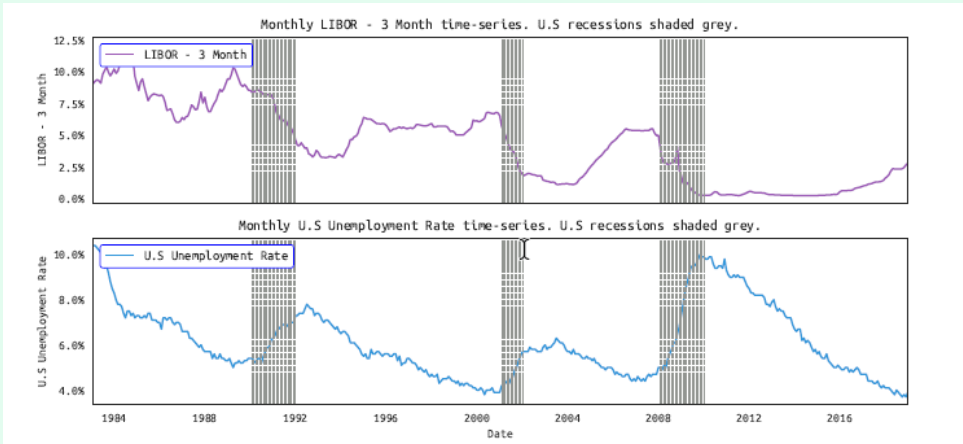
1. Develop domain agnostic software packages and tools to explain trained ML models, such a
2. Develop point forecasts models to Bayesian techniques which allows the end user to accoun

Experiment 1

- article : Machine Learning Algorithms for financial Asset Price Forecasting

Training data

- All publicly traded U.S stocks available on the Wharton Research Data Services (WRDS) cloud, from 1983 to the start of 2019 covering a time horizon of 30 years - this equated to 782 stocks
- From the WRDS cloud, data was pulled from the Center for Research in Security Prices (CRSP) and the Standard & Poor's (S&P) Global Market Intelligence databases
- Data was extracted for monthly and annually asset prices, accounting financial statements and macroeconomic factors, examples below:



The study attempts to predict the annual returns of the Machine Learning algorithms shown below, v

ML Algorithm	Optimization /
NGBoost [100].	Grid Search.
XGBoost.	HyperOpt imp Tree of Parzen 50 trials [110].
Catboost [111].	HyperOpt imp Tree of Parzen 50 trials [110].
LightGBM [112].	HyperOpt imp Tree of Parzen 50 trials [110].
Shallow Feed-Forward Neural Network (Shallow FNN). [109]	HyperOpt imp Tree of Parzen 100 trials [110]
Deep Feed-Forward Neural Network (Deep FNN) [109].	HyperOpt impl Parzen Algori trials [110].

--	--

Results:

Optimized Model	Mean Squared Error (MSE)
CAPM	1.6001
NGBoost	0.3572
XGBoost	0.3280
Catboost	0.3125
LightGBM	0.3131
Shallow FNN	0.3628
Deep FNN	0.3531

NGBoost: Natural Gradient Boosting for Probabilistic Prediction

- Goal : This paper present the Natural Gradient Boosting (NGBoost), an algorithm for generic probabilistic prediction via gradient boosting. And show how the Natural Gradient is required to correct the training dynamics of our multiparameter boosting approach
- Link : <https://arxiv.org/abs/1910.03225v4> (June 9,, 2020)
- Code : <https://github.com/stanfordmlgroup/ngboost>

Model

Approach	Description	Advantages	Analysis and
----------	-------------	------------	--------------

<p>NGBoost: is a new boosting algorithm, which uses Natural Gradient Boosting, a modular boosting algorithm for probabilistic prediction.</p>	<p>This algorithm consists of base learner, parametric probability distribution, and scoring rule</p> <div data-bbox="289 220 1092 317"> </div> <p>Figure 2: NGBoost is modular with respect to choice of base learner, distribution, and scoring rule.</p> <ol style="list-style-type: none"> Base Learner: It takes inputs x and outputs are used to form the conditional probability. Those base learners use scikit-learn's Decision Tree for a tree learner and Ridge regression for a linear learner. Parametric Probability Distribution: It's a conditional distribution, that is formed by an additive combination of base learner outputs. Scoring Rule: takes a predicted probability distribution and one observation of the target feature to produce a score to the prediction, where the true distribution of the outcomes gets the best score in expectation. This algorithm uses MLE (Maximum Likelihood Estimation) or CRPS (Continuous Ranked Probability Score). <p>Pseudo Code:</p> <div data-bbox="289 638 954 1297"> <p>Algorithm 1 NGBoost for probabilistic prediction</p> <p>Data: Dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$.</p> <p>Input: Boosting iterations M, Learning rate η, Probability distribution with parameter θ, Proper scoring rule S, Base learner f.</p> <p>Output: Scalings and base learners $\{\rho^{(m)}, f^{(m)}\}_{m=1}^M$.</p> <p>$\theta^{(0)} \leftarrow \arg \min_{\theta} \sum_{i=1}^n S(\theta, y_i)$ {initialize to marginal}</p> <p>for $m \leftarrow 1, \dots, M$ do</p> <p> for $i \leftarrow 1, \dots, n$ do</p> <p> $g_i^{(m)} \leftarrow \mathcal{I}_S \left(\theta_i^{(m-1)} \right)^{-1} \nabla_{\theta} S \left(\theta_i^{(m-1)}, y_i \right)$</p> <p> end</p> <p> $f^{(m)} \leftarrow \text{fit} \left(\left\{ x_i, g_i^{(m)} \right\}_{i=1}^n \right)$</p> <p> $\rho^{(m)} \leftarrow \arg \min_{\rho} \sum_{i=1}^n S \left(\theta_i^{(m-1)} - \rho \cdot f^{(m)}(x_i), y_i \right)$</p> <p> for $i \leftarrow 1, \dots, n$ do</p> <p> $\theta_i^{(m)} \leftarrow \theta_i^{(m-1)} - \eta \left(\rho^{(m)} \cdot f^{(m)}(x_i) \right)$</p> <p> end</p> <p>end</p> </div> <ul style="list-style-type: none"> In each iteration m, the algorithm calculates, for each example i, the natural gradient of the scoring rule S with respect to the predicted parameter The output of the fitted base learner is the projection of the natural gradient on to the range of the base learner class The scaling factor is chosen to minimize the overall true scoring rule loss along the direction of the projected gradient in the form of a line search For very large datasets computational performance can be easily improved by simply randomly sub-sampling mini-batches within the fit() operation. 	<ul style="list-style-type: none"> NGBoost is flexible, scalable, and easy-to-use. handles classification, regression, survival problems No expert knowledge of deep learning, Bayesian statistics, or Monte Carlo methods is required to use NGBoost. employs the natural gradient to correct the training dynamics of multiparameter boosting 	<ol style="list-style-type: none"> Boosting distributic family of covariate Computa standard number o observati <div data-bbox="1458 655 1495 770"> <p>Figur purpo exam the va exam the er</p> </div>
---	--	---	---

Experiment

Training data	Results
---------------	---------

- The datasets were generated from the UCI Machine Learning Repository.
- For all datasets, we hold out a random 10% of the examples as a test set
- From the other 90% we initially hold out 20% as a validation set to select M (the number of boosting stages) that gives the best log-likelihood, and then retrain on the entire 90% using the chosen M
- For all experiments, NGBoost was configured with the Normal distribution, decision tree base learner with a maximum depth of three levels, and log scoring rule
- For the Year MSD dataset we use a mini-batch size of 10%,

NGBoost matches or exceeds the performance of existing methods for probabilistic prediction while offering additional benefits in flexibility, scalability, and usability.

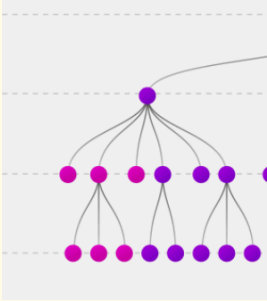
Table 3. Comparison of point-estimation performance on regression benchmark UCI datasets as measured by RMSE. Although not optimized for point estimation, NGBoost still offers competitive performance. Bolding is as in Table 1.

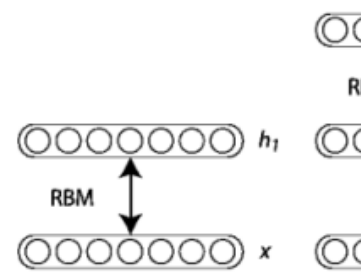
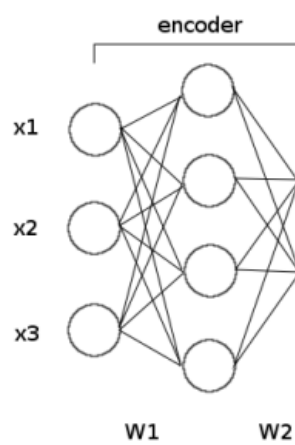
Dataset	N	NGBoost	Elastic Net	Random Forest	Gradient Boosting	GAMLSS	Distributional Forest
Boston	506	2.94 ± 0.53	4.08 ± 0.16	2.97 ± 0.30	2.46 ± 0.32	4.32 ± 1.40	3.99 ± 1.13
Concrete	1030	5.06 ± 0.61	12.1 ± 0.05	5.29 ± 0.16	4.46 ± 0.29	6.72 ± 0.59	6.61 ± 0.83
Energy	768	0.46 ± 0.06	2.75 ± 0.03	0.52 ± 0.09	0.39 ± 0.02	1.43 ± 0.32	1.11 ± 0.27
Kin8nm	8192	0.16 ± 0.00	0.20 ± 0.00	0.15 ± 0.00	0.14 ± 0.00	0.20 ± 0.01	0.16 ± 0.00
Naval	11934	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Power	9568	3.79 ± 0.18	4.42 ± 0.00	3.26 ± 0.03	3.01 ± 0.10	4.25 ± 0.19	3.64 ± 0.24
Protein	45730	4.33 ± 0.03	5.20 ± 0.00	3.60 ± 0.00	3.95 ± 0.00	5.04 ± 0.04	3.89 ± 0.04
Wine	1588	0.63 ± 0.04	0.58 ± 0.00	0.50 ± 0.01	0.53 ± 0.02	0.64 ± 0.04	0.67 ± 0.05
Yacht	308	0.50 ± 0.20	7.65 ± 0.21	0.61 ± 0.08	0.42 ± 0.09	8.29 ± 2.56	4.19 ± 0.92
Year MSD	515345	8.94 ± NA	9.49 ± NA	9.05 ± NA	8.73 ± NA	NA ± NA	NA ± NA

Predicting online user behavior using deep learning algorithms

- Goal : This paper proposes a robust classifier to predict buying intentions based on user behavior.
- Link : <https://arxiv.org/pdf/1511.06247v3.pdf> (May 27, 2016)
- Code : -

Model

Approach	description	Type	Equation/pseudo code
Non-Negative Matrix Factorization (NMF) Used to reduce dimensionality	<ul style="list-style-type: none"> It's a class of unsupervised learning algorithms, such as Principal Components Analysis (PCA) or learning vector quantization (LVQ) that factorizes a data matrix subjected to constraints Given a non-negative matrix V (containing the training data), NMF learns non-negative matrix factors, W and H applied to high dimensional problems with sparse data, like image recognition and text analysis; used in this example to compress data into feature subset Drawback: Lack of an optimal method to compute the factor matrixes and stopping criteria to find the ideal number of features to be selected 	Advanced	$V = W H$
Classifiers Decision Trees	<ul style="list-style-type: none"> Extremely easy to visualize and interpret, as can be represented graphically Considered as a white-box models; as by observing a decision tree, one can clearly understand all the intermediate steps of the classification process, such as which variables are used, by what order, etc. Unlike other methods which can't be interpreted directly Extremely fast in classifying new data Drawbacks: <ol style="list-style-type: none"> The process of building an optimal decision tree is considered to be NP-hard Tend to overfit the data unless it uses some regularization methods Trees tend to prefer variables with more categories over the other; which may cause the classifier to incorrectly consider some variables more important than others 	Baseline	
Random Forest	<ul style="list-style-type: none"> The Random Forest creates a group of decision trees using randomization, and each tree classifies the input individually Each tree is grown in an independent, random way The risk of overfitting is considerably reduced here The final classifier is efficient and capable of dealing with large data sets, missing data, and outliers. <p>Drawback: Random forests have the same bias towards variables with many categories as decision trees.</p>	Advanced	

<p>Deep Learning Methods</p> <p>Refers to a wide class of ML techniques while using many layers of non-linear processing</p> <p>The Hidden layers are referred to as deep neural networks (DNNs) which are generally trained by the Back-propagation algorithm</p> <p>Introduces two approaches</p>	<p>Deep Learning Methods</p> <ul style="list-style-type: none"> Proposed by Hinton in 2006 as an unsupervised learning algorithm for a class of generative models It's composed of a stack of restricted Boltzmann machines (RBMs) It updates the hidden units in parallel given the visible units using the $P(v)$ equation $P(v) = \frac{\sum_h e^{-Energy(v,h)}}{Z} = \frac{e^{-FreeEnergy(v)}}{Z}$ $= \frac{1}{Z} \sum_{h_1} \sum_{h_2} \dots \sum_{h_k} e^{\beta(v) - \sum_i \gamma_i(v,h_i)} = \frac{1}{Z} \sum_{h_1} \sum_{h_2} \dots \sum_{h_k} e^{\beta(v)} \prod_i e^{-\gamma_i(v,h_i)}$ $= \frac{e^{\beta(v)}}{Z} \sum_{h_1} e^{-\gamma_1(v,h_1)} \sum_{h_2} e^{-\gamma_2(v,h_2)} \dots \sum_{h_k} e^{-\gamma_k(v,h_k)}$ $= \frac{e^{\beta(v)}}{Z} \prod_i \sum_{h_i} e^{-\gamma_i(v,h_i)}$ <ul style="list-style-type: none"> The learning algorithm makes effective use of unlabeled data, it can be interpreted as a probabilistic generative model The over-fitting problem here can be reduced by the generative pre-training step Drawback : they are hard to train and very sensitive to learning parameters like weights initialization. 	
	<p>An autoencoder is a neural network with a single hidden layer and where the output layer and the input layer have the same size</p> <ul style="list-style-type: none"> When using an autoencoder to encode data, we calculate the vector $y = s(Wx + b)$ And when we use it to decode and reconstruct back the original input, we calculate $z = s(WT y + b')$ The weights of the decoder are the transpose of the encoder We want to optimize W, b, and b' so that the reconstruction is as similar to the original input as possible with respect to some loss function: $E(t, z) = \frac{1}{2}(t - z)^2$ <ul style="list-style-type: none"> The weight matrix of the decoding stage is the transpose of weight matrix of the encoding stage in order to reduce the number of parameters to learn Once an autoencoder layer has been trained, a second autoencoder can be trained using the output of the first autoencoder layer 	

A Comparative Analysis of Forecasting Financial Time Series Using ARIMA, LSTM, and BiLSTM

- Goal : Machine and deep learning-based algorithms are the emerging approaches in addressing prediction problems in time series. These techniques have been shown to produce more accurate results than conventional regression-based modeling. It has been reported that Artificial Recurrent Neural Networks (RNN) with memory, such as Long Short-Term Memory (LSTM), are superior compared to Autoregressive Integrated Moving Average (ARIMA) with a large margin. This paper tests the LSTM and the BiLSTM models against ARIMA and compares them.
- Link : <https://arxiv.org/abs/1911.09512> (Nov, 21, 2019)
- Code : -

Background Knowledge:

The Recurrent Neural Networks (RNNs) are an extension of the conventional Feed-Forward neural networks with the ability of managing variable-length sequence inputs. Unlike the conventional Feed-Forward neural networks, which are not generally able to handle sequential inputs and all their inputs (and outputs) must be independent of each others, the RNNs models provide some gates to store the previous inputs and leverage sequential information of the previous inputs. This special RNNs memory is called recurrent hidden states and gives the RNNs the ability to predict what input is coming next in the sequence of input data. However, due to RNNs' memory limitations, the length of the sequential information is limited to only a few steps back.

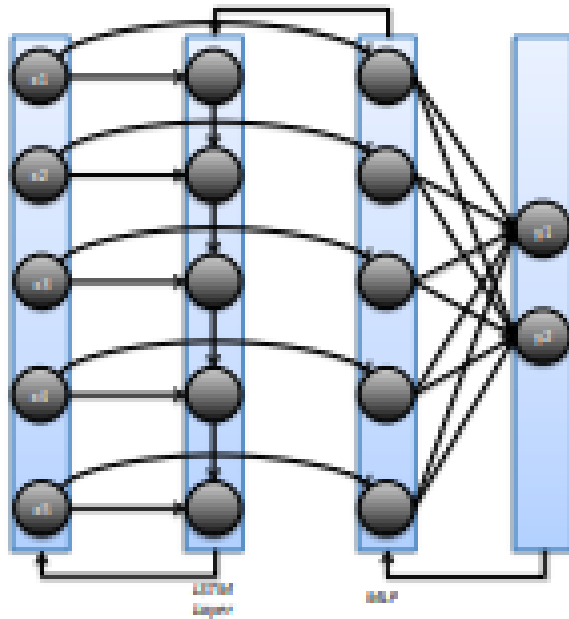
RNNs Challenges:

- "vanishing gradients" which happens when the information about the input or gradient passes thorough a lot of layers, it will vanish by the time when it reaches to the end or beginning layer. This problem makes it hard for RNNs to capture the long-term dependencies, and the training of RNNs will be extremely challenging. As the training algorithm assigns smaller values to the weight matrix and thus the RNN model stops learning.

2. “exploding gradients,” which refers to the cases in which information about the input or gradient passes through a lot of layers, it will accumulate and result in a very large gradient when it reaches to the end or beginning layer. This problem makes RNNs hard to train. The training algorithm assigns higher values to the weight matrix without any reasons

Models

Models	Description
<p>ARIMA:</p> <p>An autoregressive integrated moving average, is a statistical analysis model that uses time series data to predict future trends.</p> <p>This acronym is descriptive, capturing the key aspects of the model itself. Briefly, they are:</p> <ul style="list-style-type: none"> • AR: <i>Autoregression</i>. A model that uses the dependent relationship between an observation and some number of lagged observations. • I: <i>Integrated</i>. The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary. • MA: <i>Moving Average</i>. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations. 	<ul style="list-style-type: none"> • The model's goal is to predict future securities or financial market in the series instead of through actual values. • Each component in ARIMA functions as a parameter with a standard would be ARIMA with p, d, and q, where integer values substitute model used. The parameters can be defined as: <ul style="list-style-type: none"> • p: the number of lag observations in the model; also known as • d: the number of times that the raw observations are differenced • q: the size of the moving average window; also known as the <pre> 2 from pandas import read_csv 3 from pandas import datetime 4 from matplotlib import pyplot 5 from statsmodels.tsa.arima.model import ARIMA 6 from sklearn.metrics import mean_squared_error 7 from math import sqrt 8 # load dataset 9 def parser(x): 10 return datetime.strptime('190'+x, '%Y-%m-%d') 11 series = read_csv('shampoo-sales.csv', header=0, parse_dates=[0], dayfirst=True, squeeze=True) 12 series.index = series.index.to_period('M') 13 # split into train and test sets 14 X = series.values 15 size = int(len(X) * 0.66) 16 train, test = X[0:size], X[size:len(X)] 17 history = [x for x in train] 18 predictions = list() 19 # walk-forward validation 20 for t in range(len(test)): 21 model = ARIMA(history, order=(5,1,0)) 22 model_fit = model.fit() 23 output = model_fit.forecast() 24 yhat = output[0] 25 predictions.append(yhat) 26 obs = test[t] 27 history.append(obs) 28 print('predicted=%f, expected=%f' % (yhat, obs)) 29 # evaluate forecasts 30 rmse = sqrt(mean_squared_error(test, predictions)) 31 print('Test RMSE: %.3f' % rmse) 32 # plot forecasts against actual outcomes 33 pyplot.plot(test) 34 pyplot.plot(predictions, color='red') 35 pyplot.show() </pre>
<p>Long Short-Term Memory (LSTM) Models:</p> <ol style="list-style-type: none"> 1. LSTM: special type of RNN models is the Long Short-Term Memory (LSTM) networks, through which the relationships between the longer input and output data are modeled. These RNN-based models, called feedback-based models, are capable of learning from past data, in which several gates into their network architecture are employed in order to remember the past data and thus build the prospective model with respect to the past and current data. Hence, the input data are traversed only once (i.e., from left (input) to right (output)). 	<ul style="list-style-type: none"> • The LSTM-based models are an extension for RNNs, which are at very clean way. • It essentially extends the RNNs' memory to enable them keep and extend the ability of remembering information over a longer period by deleting information from their memories • It captures important features from inputs and preserves this information • The decision of deleting or preserving the information is made based on the training process. Hence, an LSTM model learns what to delete and what to preserve • It consists of three gates: forget, input, and output gates: <ol style="list-style-type: none"> 1. Forget Gate: A sigmoid function is usually used for this gate to make a decision about what information to be removed from the LSTM memory. This decision is essentially made based on the output of this gate is f_t, a value between 0 and 1, where 0 indicates deleting the whole value. The output is determined by: $f_t = \sigma(W_{f_h}[h_{t-1}], W_{f_x}[x_t], b_f)$ 2. Input Gate: This gate makes the decision of whether or not the new information should be stored in the memory. This gate consists of two layers: 1) a sigmoid layer, and 2) a "tanh" layer. The sigmoid layer creates a vector of new candidates to be updated, and the tanh layer creates a vector of new candidates to be updated. $c(t)$ represents the candidate vector.



(c) LSTM

$$i_t = \sigma(W_{i_h}[h_{t-1}], W_{i_x}[x_t], b_i)$$

$$\tilde{c}_t = \tanh(W_{c_h}[h_{t-1}], W_{c_x}[x_t], b_c)$$

layers provides an update for the LSTM memory in which the current value is updated through multiplication of the old value represented in the below:

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$$

3. **Output Gate:** This gate first uses a sigmoid layer to make the output. Then, it performs a non-linear tanh function to map the output to a value between 1 and 1.

$$o_t = \sigma(W_{o_h}[h_{t-1}], W_{o_x}[x_t], b_o)$$

$$h_t = o_t * \tanh(c_t)$$

representation as a value between 1 and 1.

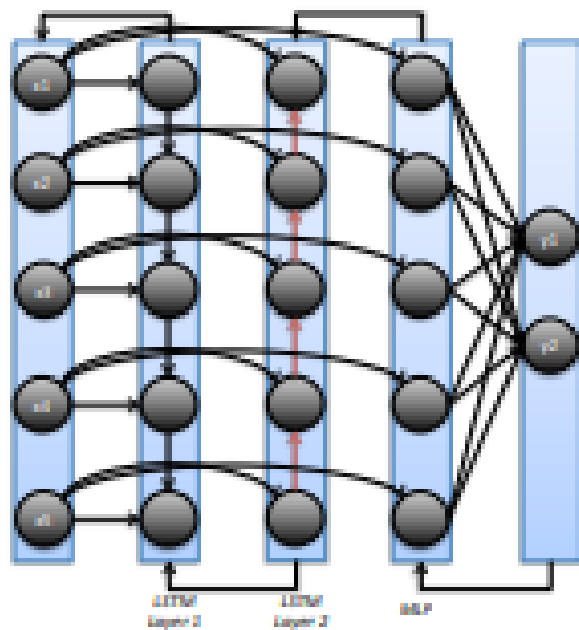
Long Short-Term Memory (LSTM) Models:

2. BiLSTM: Bidirectional LSTM enable additional training by traversing the input data twice (i.e., 1) left-to-right, and 2) right-to-left). In an BiLSTM model the given input data is utilized **twice for training** (i.e., first from left to right, and then from right to left).

In which the desired model is trained not only from inputs to outputs, but also from outputs to inputs. Given the input sequence of data, a BiLSTM model first feed input data to an LSTM model (feedback layer), and then repeat the training via another LSTM model but on the reverse order of the sequence of the input data.

- BiLSTMs is an extension of the described LSTM models in which in one round, an LSTM is applied on the input sequence (i.e., forward layer) and in the second round, an LSTM is applied on the reverse sequence (i.e., backward layer). Apply long-term dependencies and will improve the accuracy

Applying the LSTM twice train their models differently than LSTMs by feeding the input sequence twice.



(d) BiLSTM

Experiment

Training data

- time series of some stock data for the period of Jan 1985 to Aug 2018 were extracted from the Yahoo finance Website. The data included 1) Nikkei 225, 2) NASDAQ, 3) Hang Seng Index (HSI), 4) S&P 500, 5) Dow Jones, and 6) IBM Stock data
- The "Adjusted Close" variable was chosen as the only feature of financial time series to be fed into the ARIMA, LSTMs and its variation, BiLSTM models
- The data set was divided into training and test where 70% of each data set were used for training and 30% of each data set was used for testing the accuracy of models

TABLE I
THE TIME SERIES DATA STUDIED.

Stock	Observations		Total
	Train 70%	Test 30%	
N225.monthly	283	120	403
IXIC.daily	8,216	3,521	11,737
IXIC.weekly	1,700	729	2,429
IXIC.monthly	390	168	558
HSI.monthly	258	110	368
GSPC.daily	11,910	5,105	17,015
GSPC.monthly	568	243	811
DJI.daily	57,543	24,662	82,205
DJI.weekly	1,189	509	1,698
DJI.monthly	274	117	391
IBM.daily	1,762	755	2,517
Total	84,093	36,039	120,132

Assessment

- Root-Mean-Square-Error (RMSE) to assess the prediction performances. RMSE measures the differences between actual and predicated values. The formula for computing RMSE is:
Where N is the total number of observations,
 y_i is the actual value;
 \hat{y}_i is the predicated value

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2}$$

- The percentage of reduction in RMSE, as a measure to assess the improvement that can be calculated as:

$$Changes\% = \frac{New\ Value - Original\ Value}{Original\ Value} * 100$$

	<p>Loss vs. Batch Steps (Epoch = 1):</p> <ul style="list-style-type: none"> • ran developed scripts on the data and captured the loss values when the learning model fetches the next batch of data. • where the y-axis and x-axis represent the loss value and batch steps

Repository

topics	title (link)	goal	training data	algorithm	output
Fund valuation	Machine Learning Algorithms for financial Asset Price Forecasting https://arxiv.org/abs/2004.01504	<p>The prediction and forecasting of asset prices and returns remains one of the most challenging and exciting problems for quantitative finance and practitioners alike.</p> <p>This paper explores the performance of Machine Learning (ML) algorithms and techniques that can be used for financial asset price forecasting.</p>		<ul style="list-style-type: none"> • NG Boost • XG Boost • Catboost • LightGBM • Shallow FNN • Deep FNN 	<ul style="list-style-type: none"> • supervised • time series data
user behavior with the system, based on their activity	Predicting online user behavior using deep learning algorithms https://arxiv.org/abs/1511.06247v1?utm_source=top.caibaojian.com/53836	<p>This paper proposes a robust classifier to predict buying intentions based on user behavior.</p>	six months of records of user interaction with an e-commerce website	<ul style="list-style-type: none"> • Non-negative Matrix Factorization • logistic regression • Deep Belief Networks • Stacked Denoising 	<ul style="list-style-type: none"> • unsupervised • recommendation

				single auto-encoders	
	<p>Show or Suppress? Managing Input Uncertainty in Machine Learning Model Explanations</p> <p>(https://arxiv.org/abs/2101.09498)</p>	<p>We have highlighted and investigated how attribution explanation uncertainty can impact explanations in machine learning, particularly in terms of user trust, confidence, and decision-making.</p>	<p>UCI wine quality dataset</p>	<ul style="list-style-type: none"> Local Interpretability Model - Agnostic Explanations (LIME) 	<ul style="list-style-type: none"> supervised learning evaluation at the boundary of user interaction
	<p>Understanding user behavior at three scales; The AGoogleADay story</p> <p>(https://research.google/pubs/pub44006/)</p>	<p>How people behave is the central question for data analytics, and a single approach to understanding user behavior is often limiting. The way people play, the ways they interact, the kinds of behaviors they bring to the game, these factors all ultimately drive how our systems perform, and what we can understand about why users do what they do. I suggest that looking at user data at three different scales of time and sampling resolution shows us how looking at behavior data at the micro-, meso-, and macro levels is a superb way to understand what people are doing in our systems, and why. Knowing this lets you not just understand what's going on, but also how to improve the user experience for the next design cycle</p>	<p>Game user interactions dataset</p>	<ul style="list-style-type: none"> Looking at data at a micro-meso-and macro-levels 	<ul style="list-style-type: none"> unsupervised learning
<p>difficult holdings to value (difficult securities to price)</p>	<p>A Machine Learning Framework for Stock Selection</p> <p>(https://arxiv.org/abs/1806.01743)</p>	<p>This paper demonstrates how to apply machine learning algorithms to distinguish "good" stocks from the "bad" stocks.</p>	<p>Chinese stock data ranging from 2012.08.08 to 2013.03.08</p>	<ul style="list-style-type: none"> Logistic Regression Random Forest Deep Neural Network Genetic Algorithm 	<ul style="list-style-type: none"> supervised learning validation at the boundary of user interaction to volatility

				ratio
Cost-Sensitive Portfolio Selection via Deep Reinforcement Learning (https://arxiv.org/abs/2003.03051)	In this paper, considering the challenges of portfolio selection and its dynamic nature, we formulate portfolio selection as a Markov Decision Process (MDP), and propose a cost-sensitive portfolio policy network (PPN) to address it via reinforcement learning.	S&P 500	<ul style="list-style-type: none"> Markov Decision Process 	reinforcement learning
Reinforcement Learning for Portfolio Management (https://arxiv.org/abs/1909.09571)	This paper investigates the effectiveness of reinforcement learning agents on sequential portfolio management.	<ul style="list-style-type: none"> S&P 500 EURO STOXX 50 	<ul style="list-style-type: none"> Deep Soft Reinforcement Q-Neural Network (DQRN) Mixture of Score Machines (MSM) 	reinforcement learning
Unsupervised learning for economic risk evaluation in the context of Covid-19 pandemic (https://arxiv.org/abs/2011.13350)	In this work we propose a system that generates clusters among Colombian regions based on Covid19 new incidences forecasts, health, geographic, demographic and economic variables. Hence, such clusters generate geographical regions with similar impact due to the pandemic in several dimensions.	<ul style="list-style-type: none"> National Administrative Department of Statistics (DANE) National Planning Department (DNP) Colombia's Institute of Hydrology, Meteorology and Environmental Studies (IDEAM) National Institute of Health (INS) 	<ul style="list-style-type: none"> neural network PCA k-means k-means 	unsupervised learning

	<p>Statistical Arbitrage Risk Premium by Machine Learning (https://arxiv.org/abs/2103.09987)</p>	<p>We coin the expected return of an asset's factor residual risk as its Statistical Arbitrage Risk Premium (SARP). The challenge in empirically estimating SARP is finding the peers for each asset and constructing the replicate portfolios. We use the elastic-net, to project each stock's past returns onto that of every other stock. We say a stock has high (low) Statistical Arbitrage Risk (SAR) if it has low (high) R-squared with its peers.</p>	<ul style="list-style-type: none">• CRSP (Center for Research in Security Prices) daily and monthly data• Fama - French data from Kenneth French's	<ul style="list-style-type: none">• elastic-net	<ul style="list-style-type: none">• linear regression
--	---	--	---	---	---