From Regression to Neural Networks:

Evaluating AI Models for Real-World Financial Trading Strategies

Open Source Quantitative Finance Research Conference

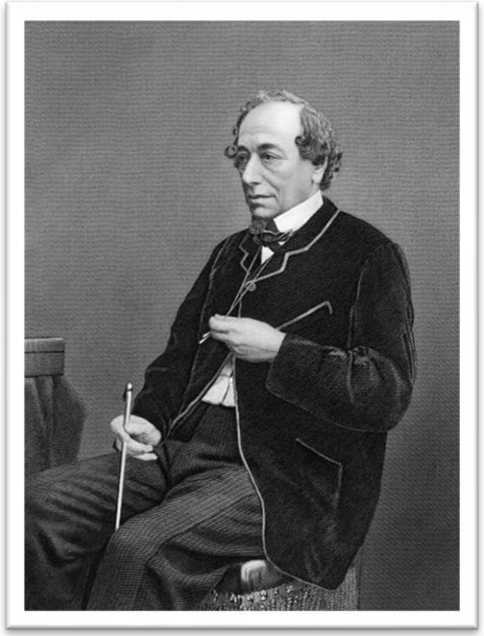
Chicago, IL – April 11-12, 2025

Davide Pandini

PhD, CMT, MFTA, CFTe, CSTA

Disclaimer

"… There are three types of lies: lies, damn lies and statistics …"



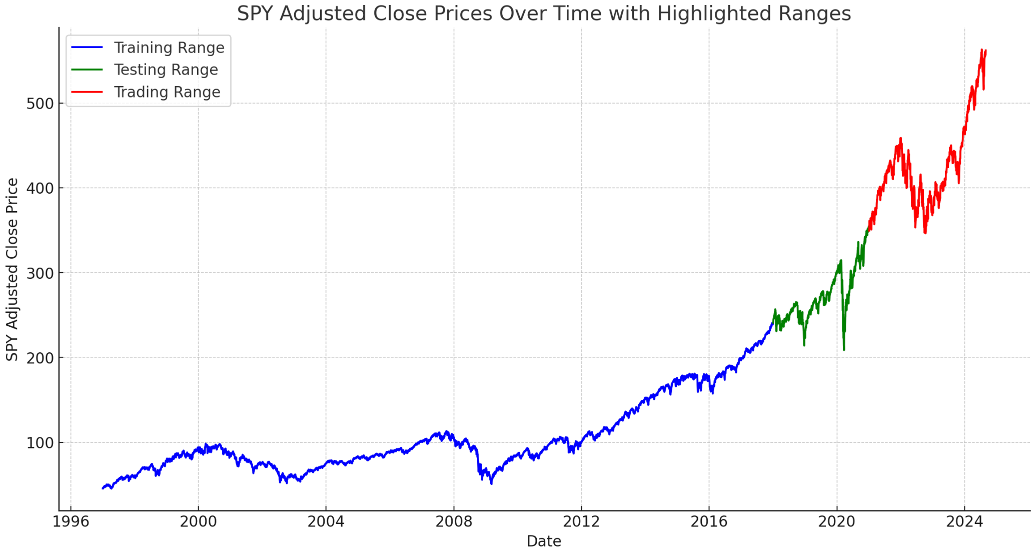
Benjamin Disraeli (1804-1881)
Prime Minister of Great Britain from 1874 to 1880

Prediction is very difficult … especially if it's about the financial markets!

"… *Prediction is very difficult, especially if it's about the future!* …"

Niels Bohr (1885-1962)
Danish physicist, Nobel Prize in Physics (1922)

Data analysis and backtesting



S&P 500 (SPY) historical data span and market cycles: 27 years and 8 months (Jan 1997 – Aug 2024) with 6,923 observations

Backtesting strategy

Training Range: from Jan 01, 1997, till Dec 31, 2017 (5,250 observations, ~76%)

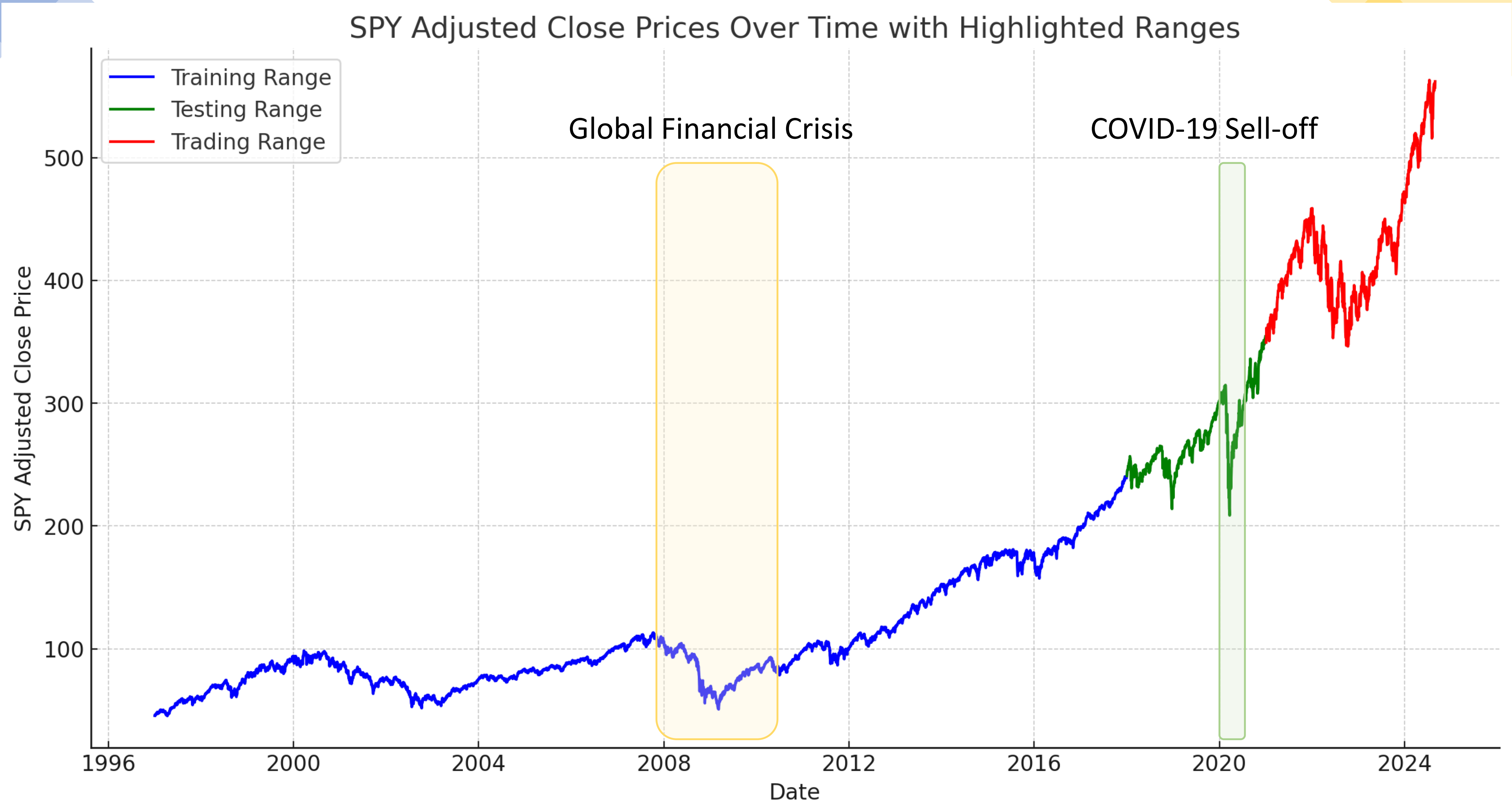
Trading Range: from Jan 01, 2021, till Aug 31, 2024 (919 observations, ~13%)

1 Jan. 2018 – 31 Dec. 2020

1 Jan. 1997 – 31 Dec. 2017

1 Jan. 2021 – 31 Aug. 2024

Testing Range: from Jan 01, 2018, till Dec 31, 2020 (754 observations, ~11%)

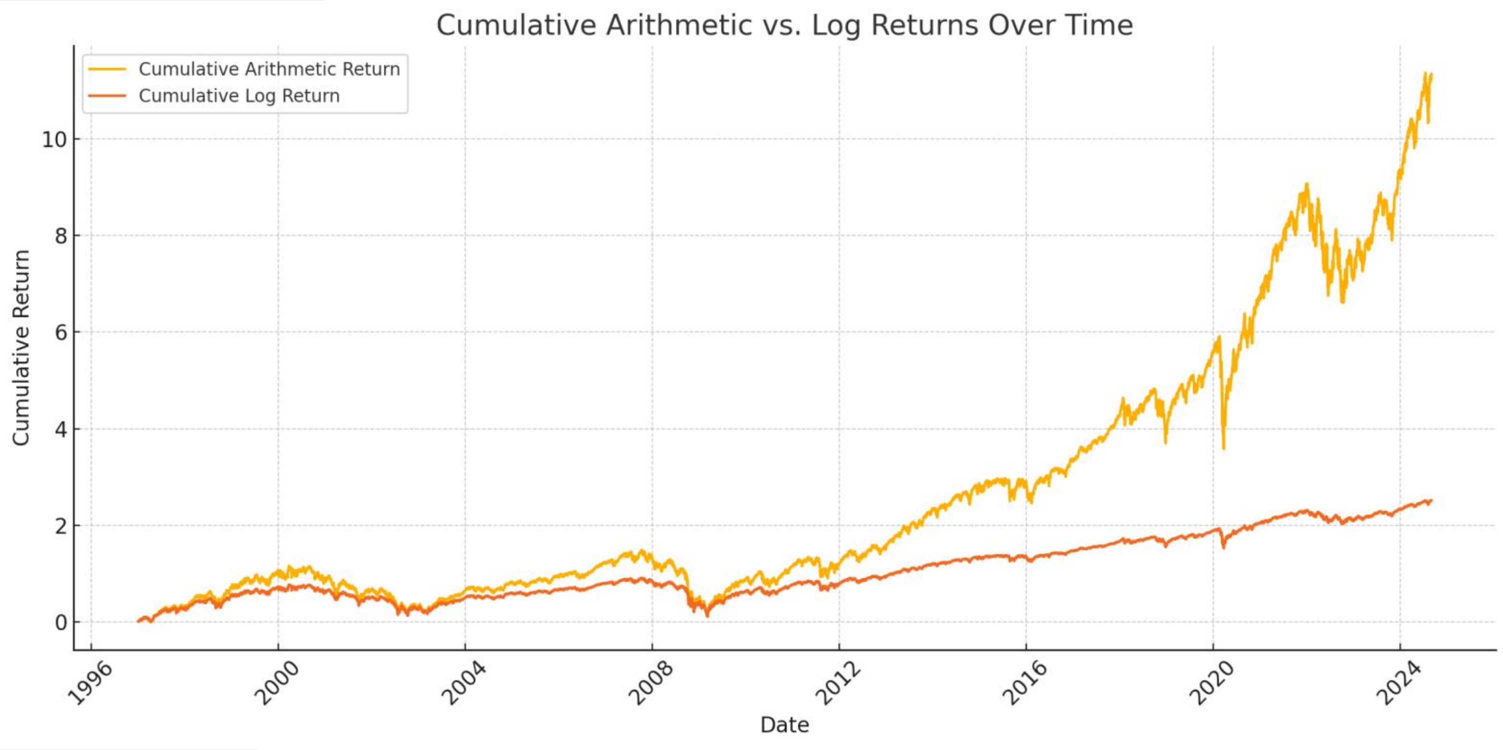

SPY Adjusted Close Prices Over Time with Highlighted Ranges
Training Range
Testing Range
Trading Range
Global Financial Crisis
COVID-19 Sell-off
SPY Adjusted Close Price
100
200
300
400
500
1996
2000
2004
2008
2012
2016
2020
2024
Date

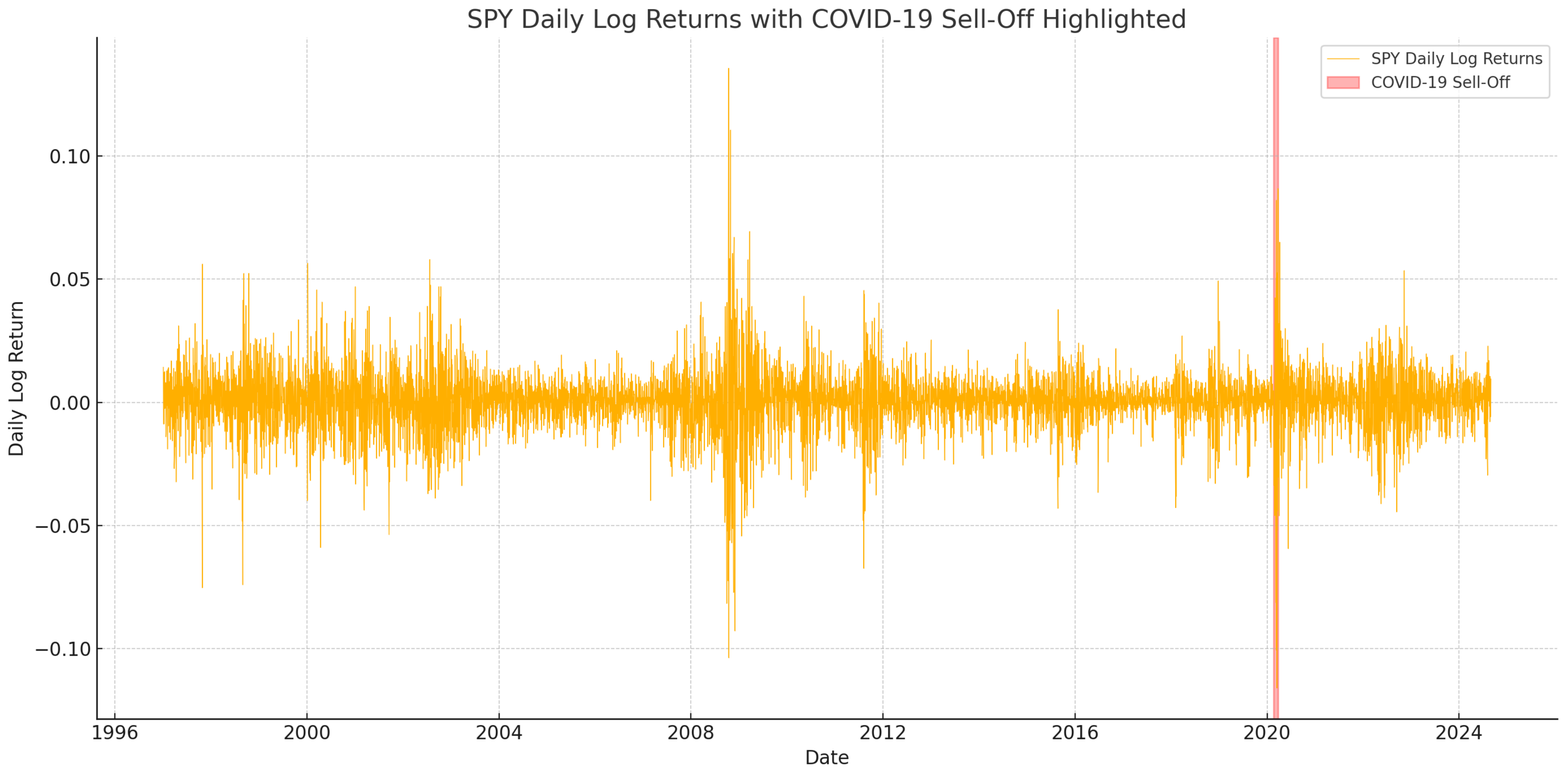
Capturing market events

Feature	2008 Global Financial Crisis	2020 COVID-19 Sell-off
Cause	Housing bubble, financial instability	Global pandemic and lockdowns
Market Decline Speed	Gradual, multi-month decline	Sharp, 1-month drop
Depth of Decline	~57% (S&P 500)	~34% (S&P 500)
Recovery Time	~5 years	~5-6 months
Policy Response	Bailouts, QE, TARP, regulatory changes	Rapid fiscal and monetary stimulus
Economic Impact	Deep recession, high unemployment	Sharp but brief recession
Investor Sentiment	Prolonged caution	Rapid rebound
Long-term Impact	Regulatory reforms, cautious investing	Digital transformation, resilience

Backtesting window selection

- Include both bear and bull markets in backtesting for holistic strategy evaluation to avoid skewed results if only one market regime is considered
- Consistent backtesting window sizes for reliable performance assessment
- Log daily returns (for stationarity and normality) highlighting volatility clusters (e.g., GFC and COVID crash)
- Distribution of cumulative returns to illustrate returns behavior and growth over time (e.g., long bull market post 2009)

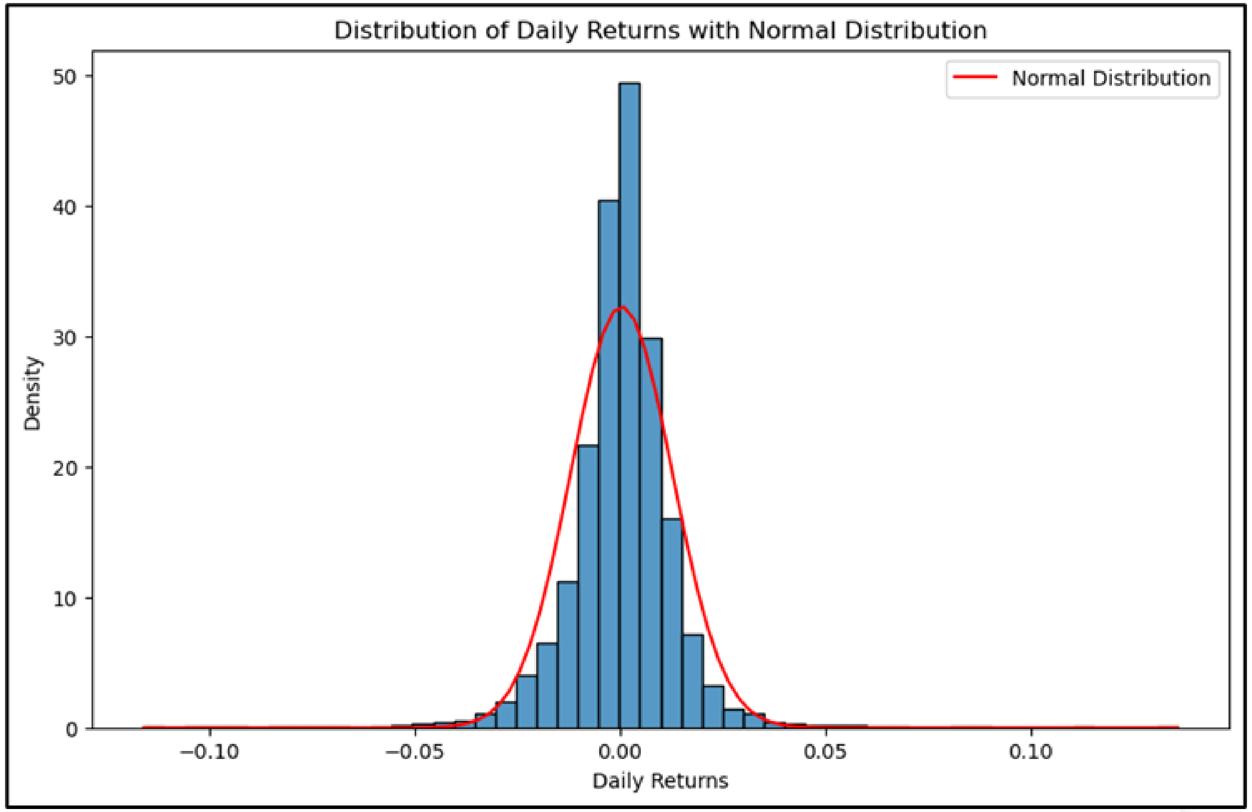




SPY Daily Log Returns with COVID-19 Sell-Off Highlighted
SPY Daily Log Returns
COVID-19 Sell-Off
Daily Log Return
0.10
0.05
0.00
−0.05
−0.10
1996
2000
2004
2008
2012
2016
2020
2024
Date

Jarque-Bera normality test

- The JB normality test statistic (30144) is larger than the critical value (i.e., six), then the null hypothesis of the log daily returns normal distribution is rejected
- The p-value from the chi-squared distribution with df=2 of the JB test statistic is ~0.0 then the null hypothesis of normality is rejected with 95% of statistical confidence
- The Jarque-Bera test confirms that the log daily returns do not follow a normal distribution



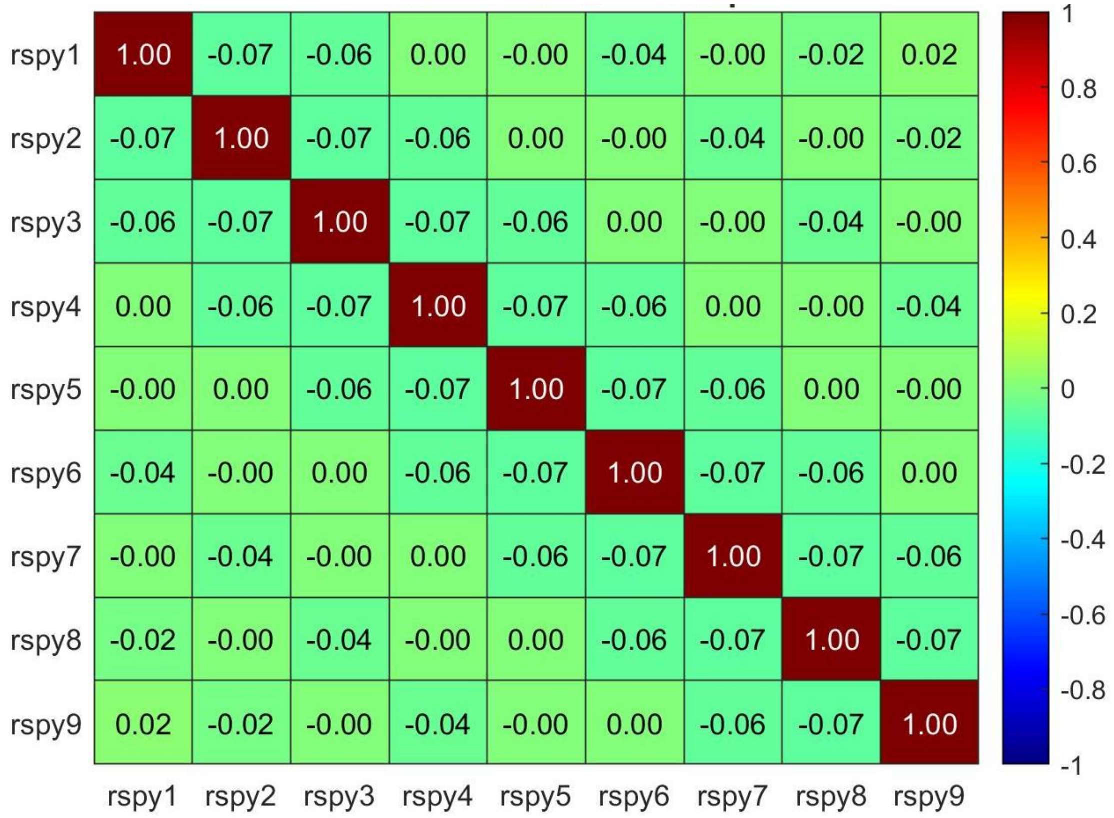
	log daily returns
skewness	-0.276214
excess kurtosis	10.201022
jarque-bera	30144.39
p-value (df=2)	< 2.2E-16

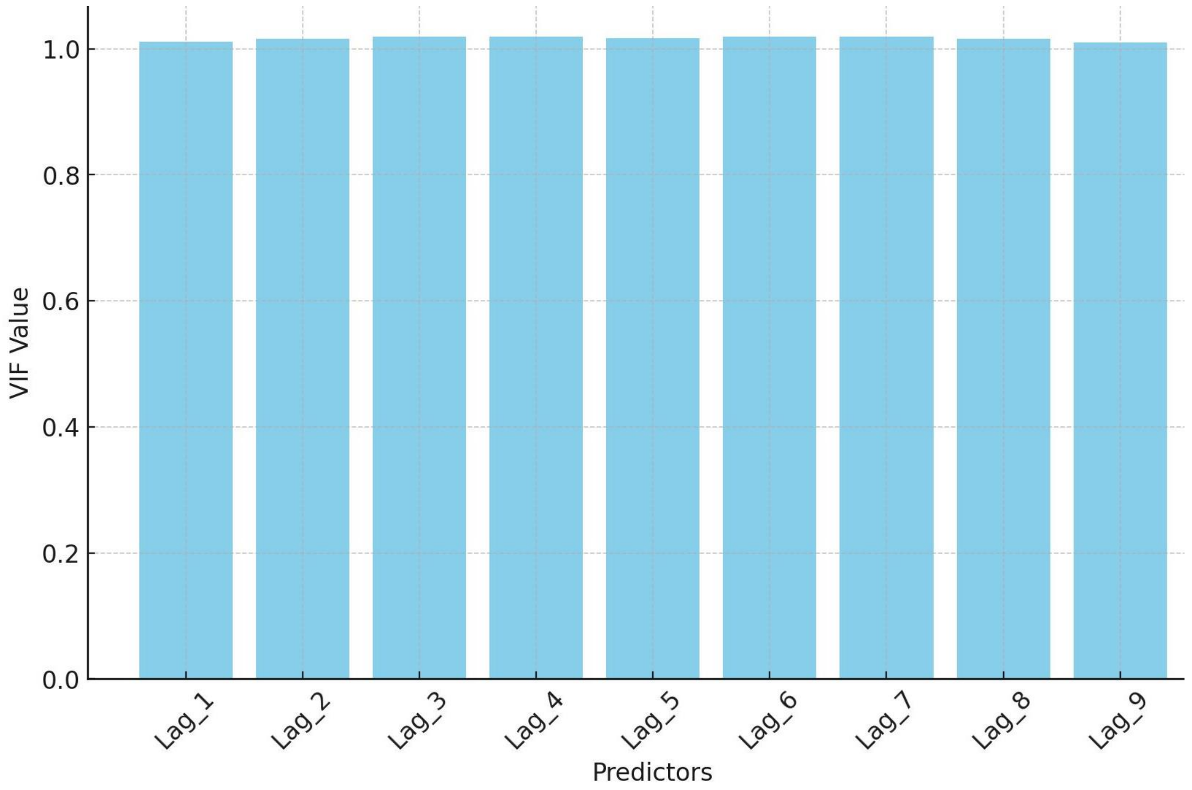
$$JB = n \cdot \left(\frac{S^2}{6} + \frac{(K-3)^2}{24} \right)$$

Feature selection methodology

- **S&P 500 lagged 9-day returns** chosen as features to capture short-term momentum/reversion
- **Temporal Dependencies**: Lagging captures short-term memory in financial data, where recent movements influence near-term future movements
- **Mean Reversion & Patterns**: If markets exhibit mean reversion or momentum, lagged returns may help detect these patterns
- **Practical Precedent**: Box-Jenkins methods in the 1970s formalized using lags in ARIMA models. Nobel laureates like Robert F. Engle (ARCH models) contributed to understanding when lagged data can hold predictive power
- **Widespread Use**: Traders use lagged indicators (e.g., yesterday's return, last week's trend) to inform strategies

Multicollinearity assessment





Feature selection techniques

Method	Description	Type
Selection by Filtering (SBF)	Evaluates each feature independently for relevance	Filter
Recursive Feature Elimination (RFE)	Iteratively trains models, removing the least important features each round	Wrapper
LASSO Regression	Performs feature selection integrated into model training via regularization; Penalizes and sets less important feature coefficients to zero	Embedded
Principal Component Analysis (PCA)	Transforms original correlated features into fewer uncorrelated components, preserving most of the data's variance	Feature Extraction

Selected features for modeling

- A parallel set of models is built using PCA components (likely a few principal components capturing most variance from the 9 lags) to compare against the SBF approach

- By testing both SBF vs. PCA we evaluate if dimensionality reduction improves model performance or efficiency

- This addresses whether to focus on interpretability (keeping actual lag features SBF) or potential accuracy gains (using abstract components PCA)

Method	Explanatory variables selected			
SBF	rspy_(t-1)	rspy_(t-2)	rspy_(t-5)	
RFE	rspy_(t-1)	rspy_(t-2)	rspy_(t-5)	rspy_(t-7)
LASSO	rspy_(t-1)	rspy_(t-2)		

SBF offers a balanced middle ground: more features than LASSO to avoid underfitting and less features than an all-inclusive approach

Aspect	SBF (Selected Best Features)	PCA (Principal Component Analysis)
Approach	Selects the most relevant features from the original dataset.	Transforms the original features into a set of uncorrelated components.
Interpretability	High (features retain their original meaning).	Low (principal components are combinations of original features).
Dimensionality Reduction	Limited (retains original features with potential multicollinearity).	Significant (reduces to fewer uncorrelated components).
Best Used For	When feature interpretability is essential and multicollinearity is not a major issue.	When reducing dimensionality and multicollinearity is important.
Assumptions	No assumptions about data structure.	Assumes linear relationships among features.
Risk	May miss complex patterns by focusing only on a subset of features.	Loss of interpretability and risk of ignoring non-linear relationships.

Forecasting models benchmark

Model	Strengths	Weaknesses	Best Use Cases
MLR	Simple, fast, interpretable	Limited to linear relationships	Small/moderate-sized datasets Interpretability crucial
XGBoost	High accuracy Handles complex interactions	Interpretability issues Tuning complexity	Structured data Predictive accuracy
SVM	High-dimensional Nonlinear boundaries	Complex tuning Slower training	Nonlinear Moderate-sized datasets
ANN	Flexible nonlinear modeling	Overfitting risk Low explainability	Hidden nonlinear patterns Adequate data
DNN	Learns deep/complex patterns	High computational needs Severe overfitting	Large datasets Complex patterns (images)

Parameter	First Method	Second Method
`initialWindow`	168 (train on first 168 observations initially).	48 (train on first 48 observations initially).
`horizon`	82 (test on 82 observations following the training period).	12 (test on 12 observations following the training period)
`fixedWindow`	Keeps training window fixed at 168.	Keeps training window fixed at 48.
`skip`	No skips between time slices.	Skips 12 observations after each testing set.

Model evaluation

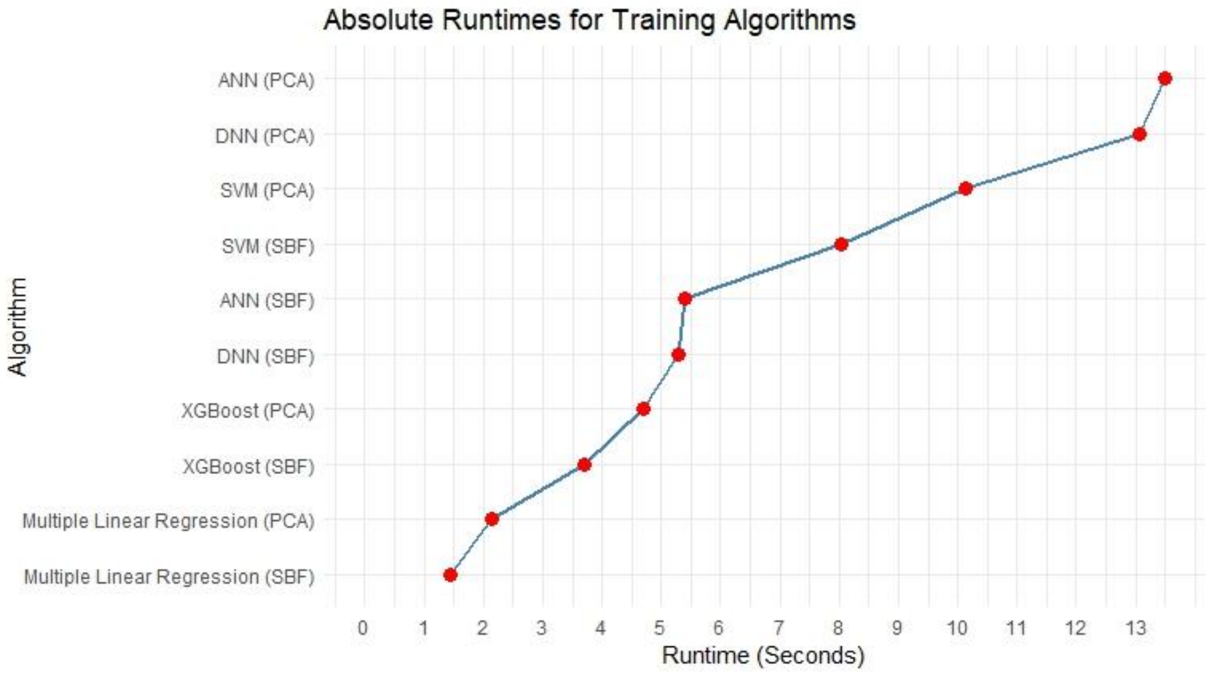
- Training vs. Testing: All models were trained on the 1997–2017 data and then evaluated in 2018–2020 test data to check generalization
- This was a supervised regression task, where models learned to predict next-day returns from lagged returns
- In walk-forward (time-series) cross-validation, models are re-trained and tested on expanding/rolling windows, preserving time order

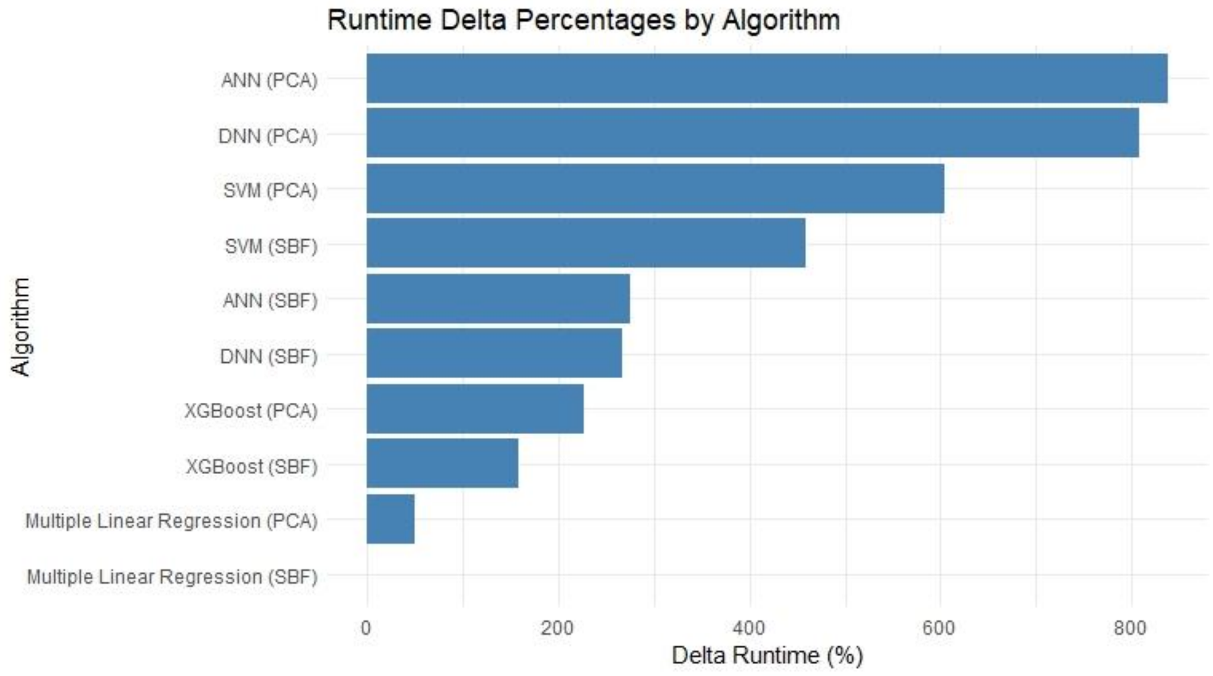
Forecasting accuracy metrics

- The numerical results are computed on the training set
- Models are benchmarked against a random walk baseline

		Scale Dependent		Scale Independent	
Model	**Predictor Set**	**RMSE**	**MAE**	**MAPE**	**MASE**
Multiple Linear Regression	SBF	0.01470	0.00895	113.02260	0.68060
	PCA	0.01483	0.00899	113.70940	0.68393
Extreme Gradient Boosting (XGBoost)	SBF	0.01457	0.00889	114.24760	0.67645
	PCA	0.01629	0.00932	112.33880	0.70934
Support Vector Machine (RBF)	SBF	0.01477	0.00896	125.63300	0.68151
	PCA	0.01473	0.00896	130.73850	0.68150
Artificial Neural Network (ANN)	SBF	0.01467	0.00893	112.88810	0.67978
	PCA	0.01484	0.00899	114.67790	0.68409
Deep Neural Network (DNN)	SBF	0.01471	0.00895	112.90700	0.68062
	PCA	0.01482	0.00899	113.93390	0.68362

Forecasting model training runtime





Forecasting model training runtime

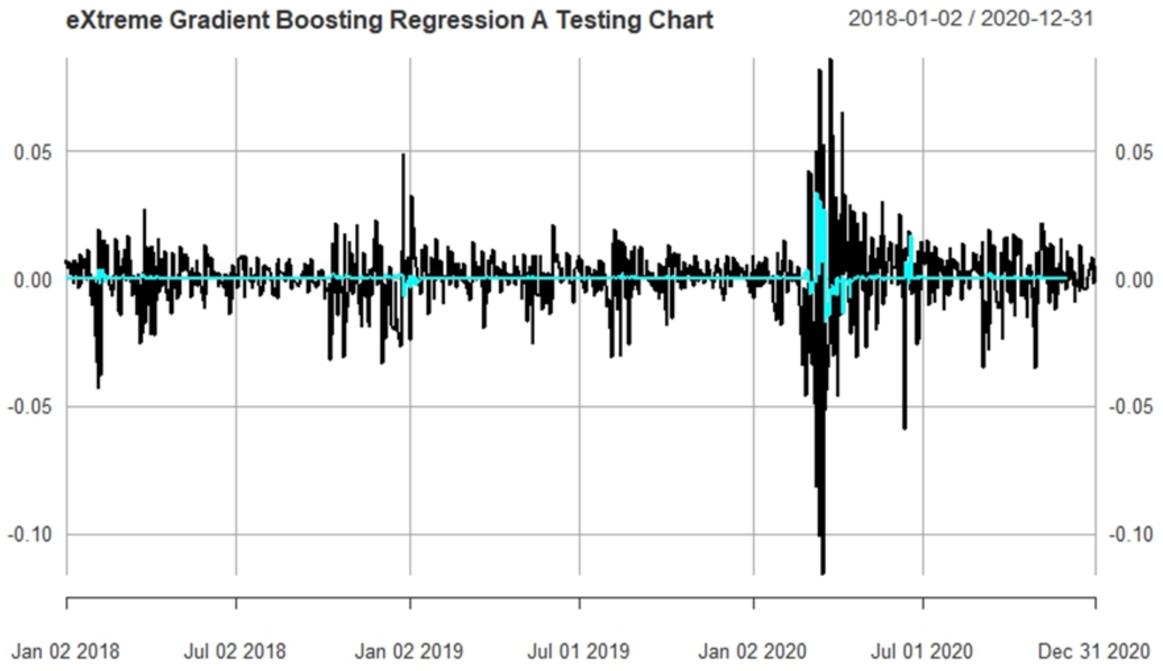
Algorithm	Runtime (Seconds)	Delta Runtime (Percent)
Multiple Linear Regression (SBF)	1.32160	Baseline
Multiple Linear Regression (PCA)	2.18499	65.30%
XGBoost (SBF)	3.60908	173.10%
XGBoost (PCA)	4.45782	237.30%
SVM (SBF)	8.86284	570.60%
SVM (PCA)	10.55498	698.70%
ANN (SBF)	5.38184	307.20%
ANN (PCA)	14.60274	1004.90%
DNN (SBF)	5.00221	278.50%
DNN (PCA)	15.25054	1053.90%

Forecasting accuracy takeaways

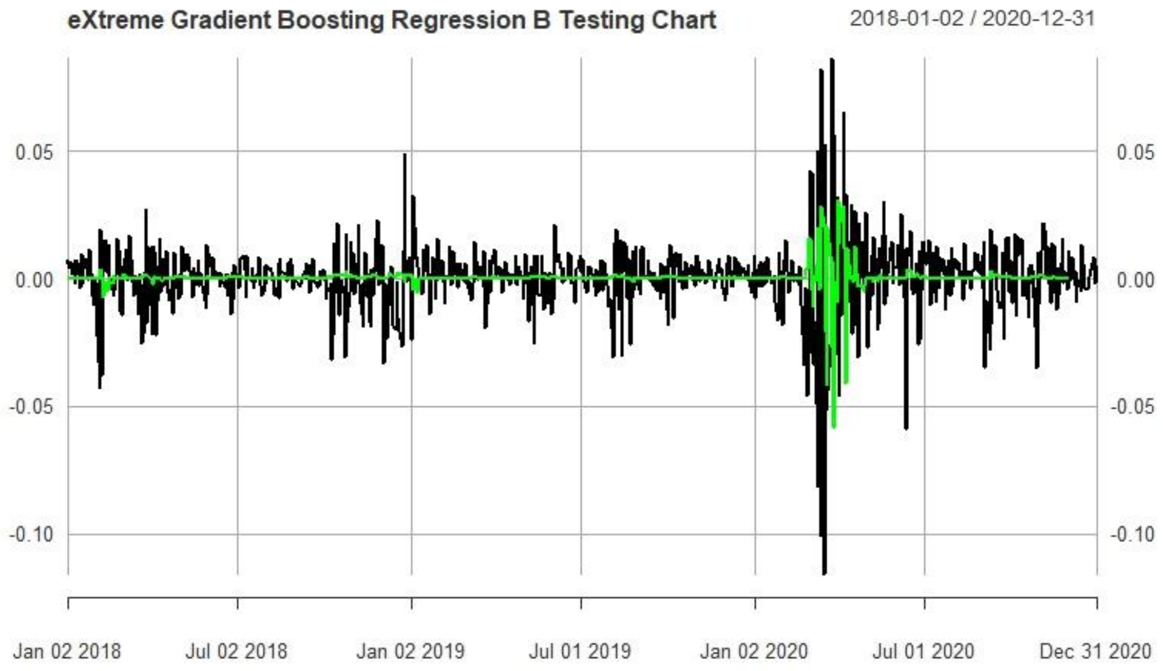
- Selection by Filtering (SBF) yields better or comparable forecasting accuracy than Principal Component Analysis (PCA) in the training range
  - This suggests that when the predictors show low multicollinearity the dimensionality reduction of PCA is not significantly beneficial
- The forecasting accuracy metrics (RMSE, MAE, MAPE, MASE) were relatively close across all five models (Multiple Linear Regression, XGBoost, SVM, ANN, DNN) and both predictor sets
- The traditional Multiple Linear Regression (MLR) model achieved comparable levels of forecasting accuracy to the more advanced AI-driven models, but with a much lower runtime

Forecasting accuracy: testing set

SBF

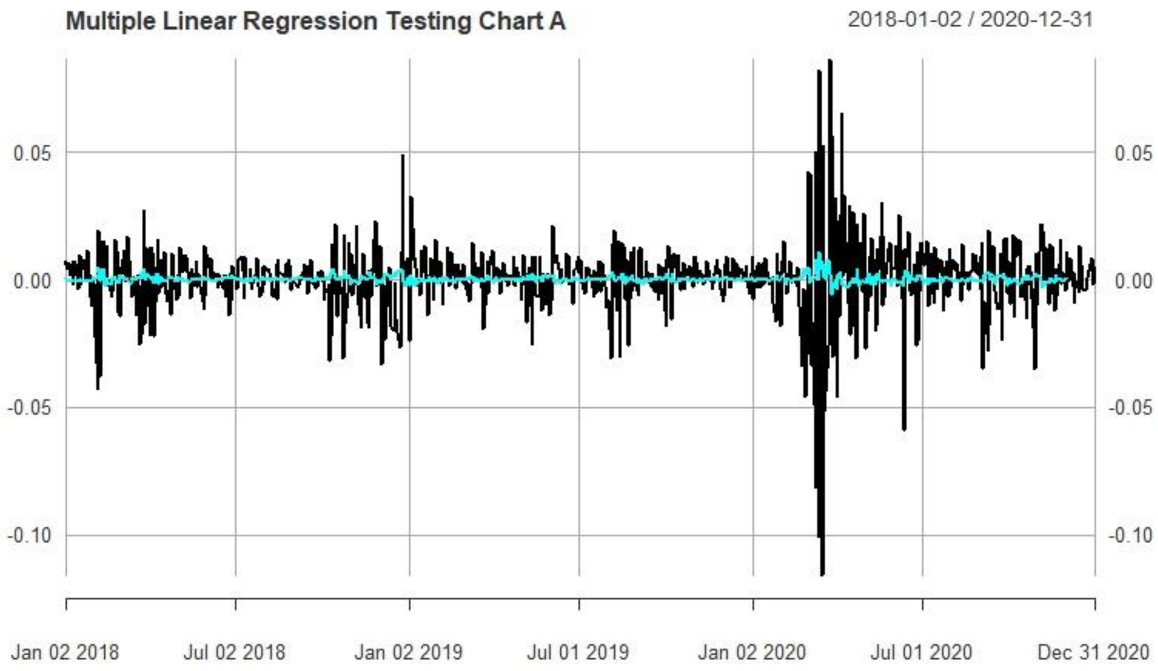


PCA

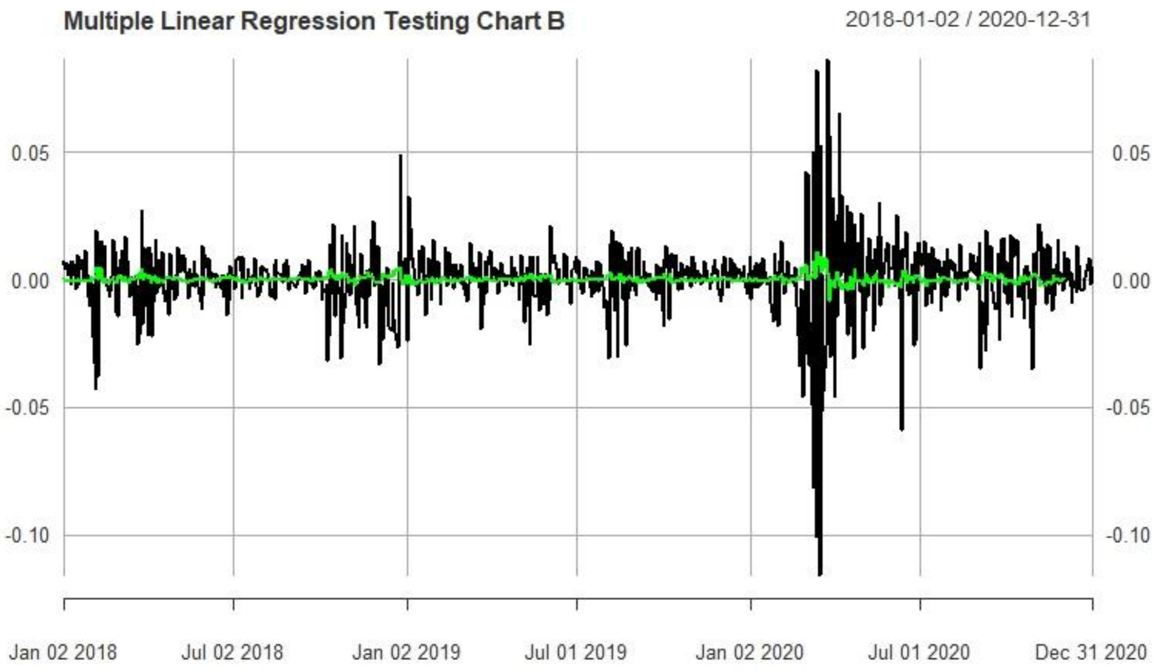


Forecasting accuracy: testing set

SBF



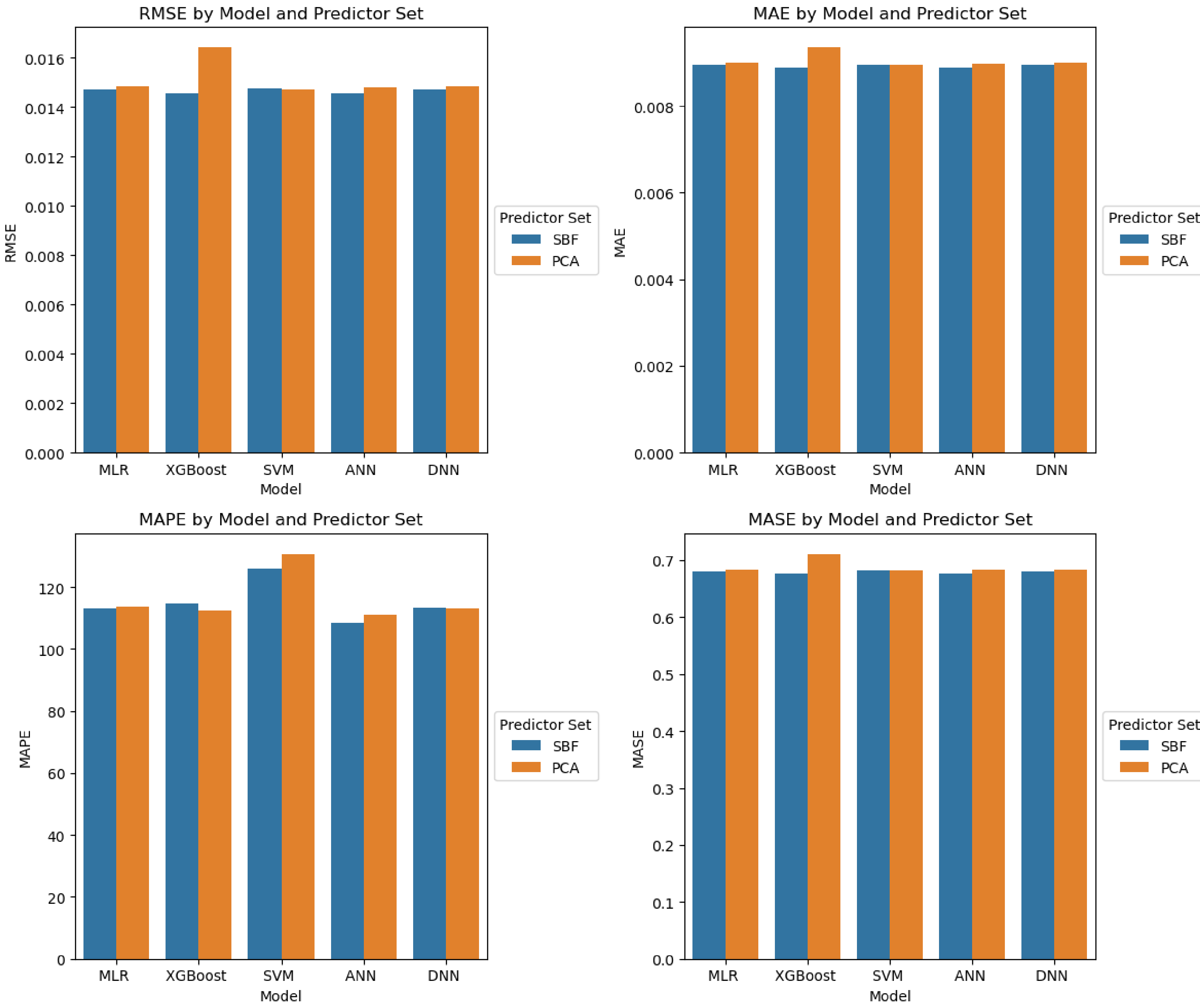
PCA



Forecasting accuracy metrics: testing set

		Scale Dependent		Scale Independent	
Model	**Predictor Set**	**RMSE**	**MAE**	**MAPE**	**MASE**
MLR	SBF	0.014703	0.008945	113.027546	0.680603
	PCA	0.014831	0.008989	113.709691	0.683929
XGBoost	SBF	0.014457	0.008856	111.008100	0.673793
	PCA	0.015934	0.009278	122.203322	0.705939
SVM	SBF	0.014767	0.008951	125.562054	0.681048
	PCA	0.014728	0.008957	130.755548	0.681513
ANN	SBF	0.014703	0.008945	113.111680	0.680602
	PCA	0.014828	0.008987	112.700635	0.683763
DNN	SBF	0.014776	0.008967	112.215480	0.682268
	PCA	0.014832	0.008989	114.179355	0.683921

Forecasting accuracy metrics: testing set



Trading strategy: design and implementation

- Signal generation: Look at the predicted next-day return direction relative to prior prediction
  - A **buy signal** is triggered when a model predicts a shift from a negative to a positive return, suggesting an upward movement
  - **A sell signal** is issued when a shift from a positive to a negative return is forecasted, indicating a potential downturn
  - **A hold** signal is maintained when no significant change in return direction is predicted, implying no trading action
- The strategy tries to capture market reversals by acting when the model predicts a change in trend direction

Benefits of simplicity

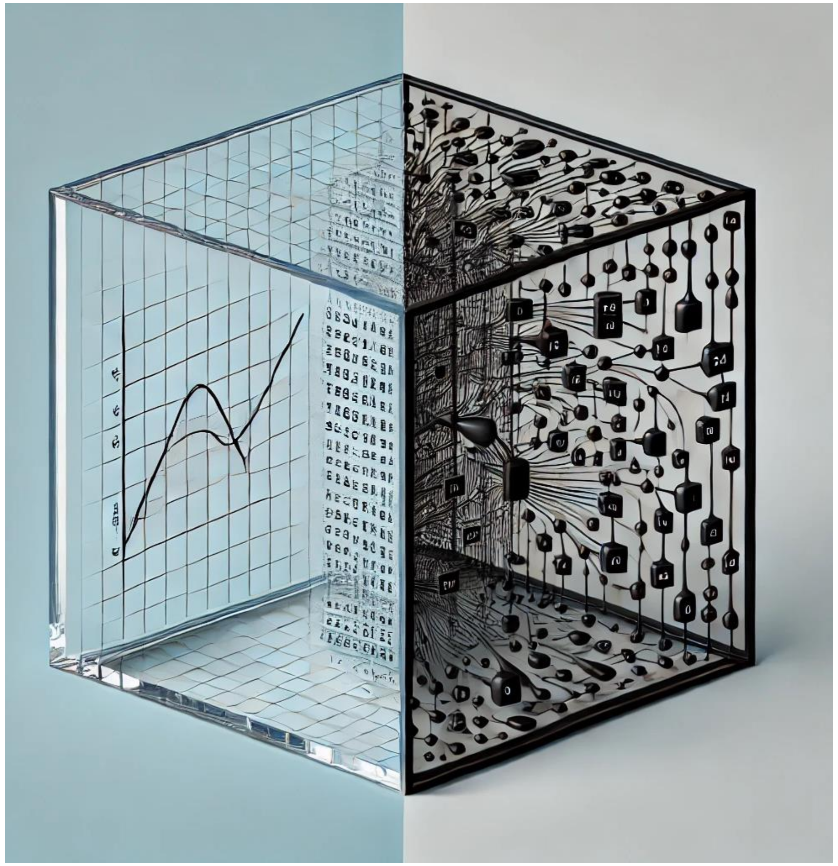
- Less risk of overfitting strategy rules to past
- Easier to attribute performance to model vs. complicated rule synergy
- Lower chance of capturing noise: A simple reduces noise and overfitting
- Focus on key patterns: Basic reversal targets fundamental market behavior (trend changes) which are relatively stable phenomena
- Ease of interpretation: Traders can understand why a signal happened

Trading strategy: design and implementation

- **Avoiding Look-Ahead Bias**
  - Shift Signals by One Day: Any signal generated on day t (based on info up to t) is executed at the open of day t+1. This ensures the model is not trading on information it could not have known (like using day t’s closing price to trade on day t – which is impossible in real time)
  - By shifting forward, any inadvertent peeking at future data when backtesting is eliminated. This simulates how yesterday’s predictions are used for today’s trades
  - Prevent inflated performance due to hindsight
- **Uniform Application**
  - All models' signals (buy/sell/hold) are generated the same way, so differences in results are due to model forecasts, not strategy differences

Data requirements

- **Amount of Data**: ML algs typically require large amounts of data to train effectively. If your dataset is limited, a NN might not perform better than a simpler model
- **Feature Engineering**: Multiple regression models can benefit significantly from carefully engineered features, potentially matching or exceeding the performance of a ML alg
- **Interpretability**: Multiple regression models are more interpretable, allowing for better understanding and validation of the relationships captured
- **Black Box Nature of NNs**: The complexity of NNs can make it difficult to interpret the model's decisions, which is a disadvantage in fields where understanding the model is crucial



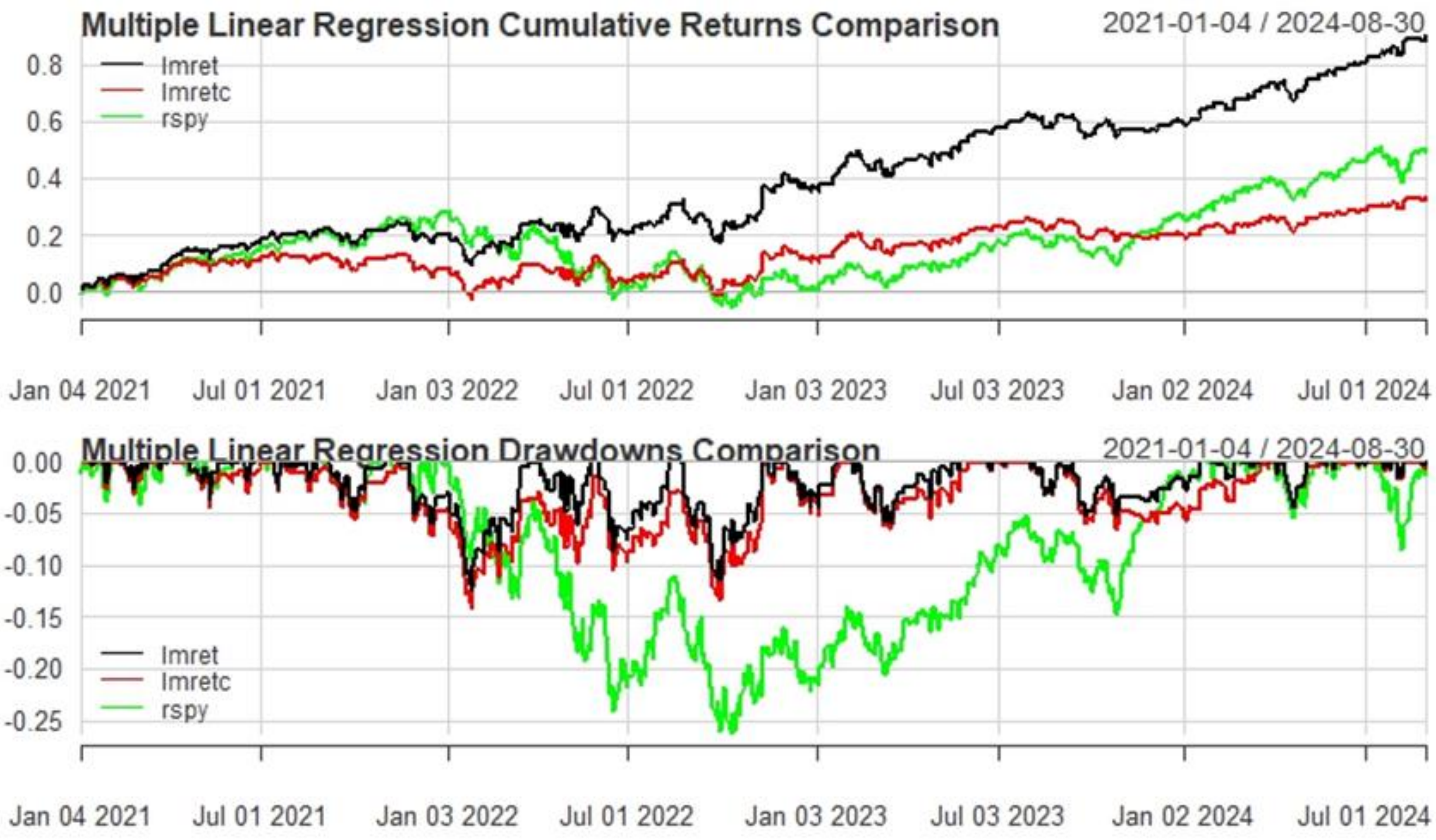
Computational resources and mixed results

- **Training Time**: NNs require more computational power and time to train
- **Optimization Challenges**: They can be more sensitive to hyperparameters and require more careful tuning
- **Empirical Evidence**: Studies comparing linear models and NNs for financial forecasting have produced mixed results. In some cases, simpler models perform just as well or better

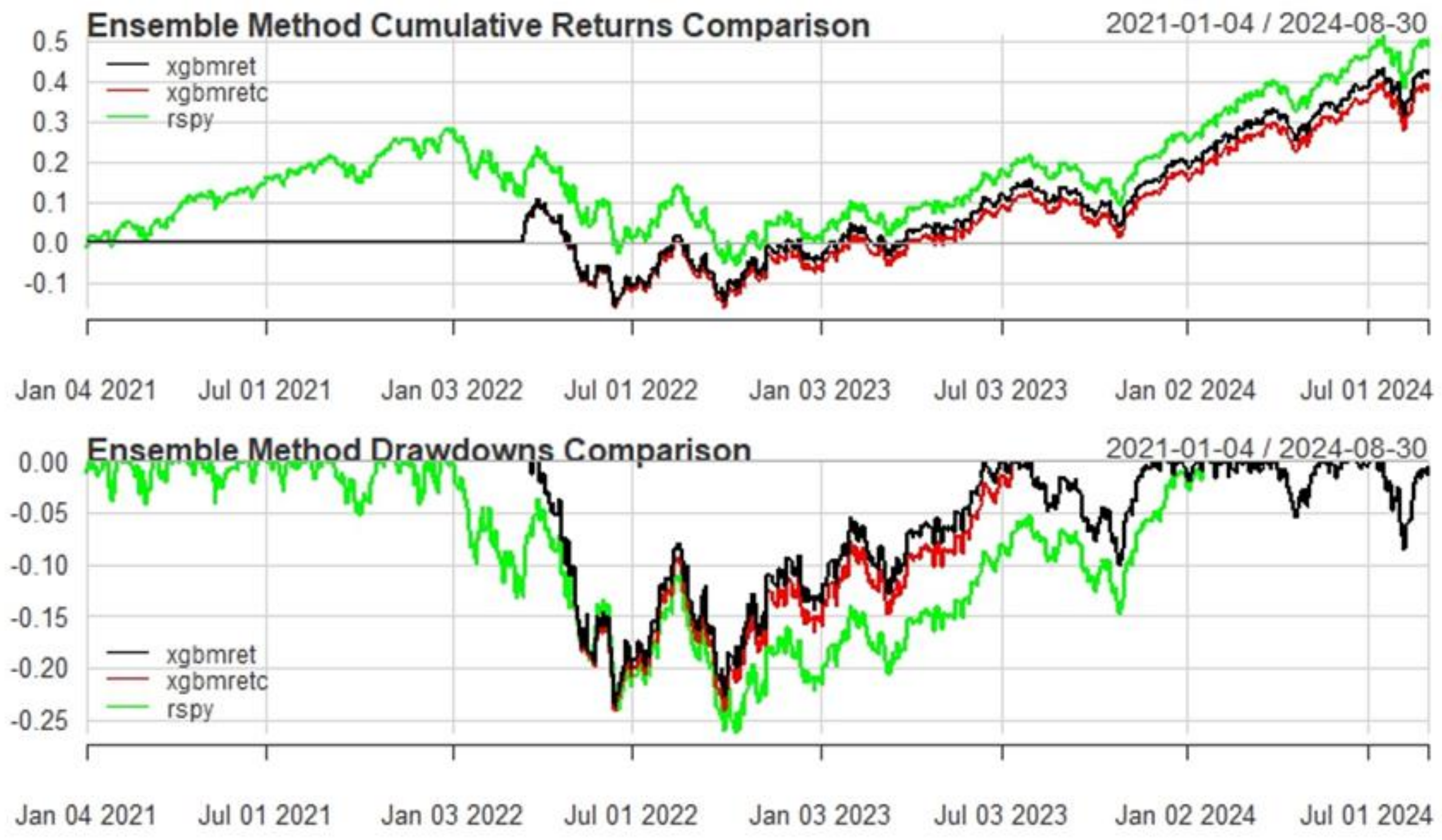
Trading strategy: design and implementation

- After models are trained and tested, each model is used (without retraining) to generate predictions on the **Trading Range (2021–2024)**
- XGBoost had best forecast accuracy, so one might expect it to generate the best trading returns
- Although XGBoost achieved highest forecasting accuracy, for completeness all models are evaluated in trading
- The results will show if best forecast accuracy translates to best trading performance (not always the case, because small accuracy differences might not matter or might be offset by other factors like consistency or variance of errors)
- Benchmark Buy-and-Hold on SPY for the same period to compare passive vs. model-driven approach

MLR

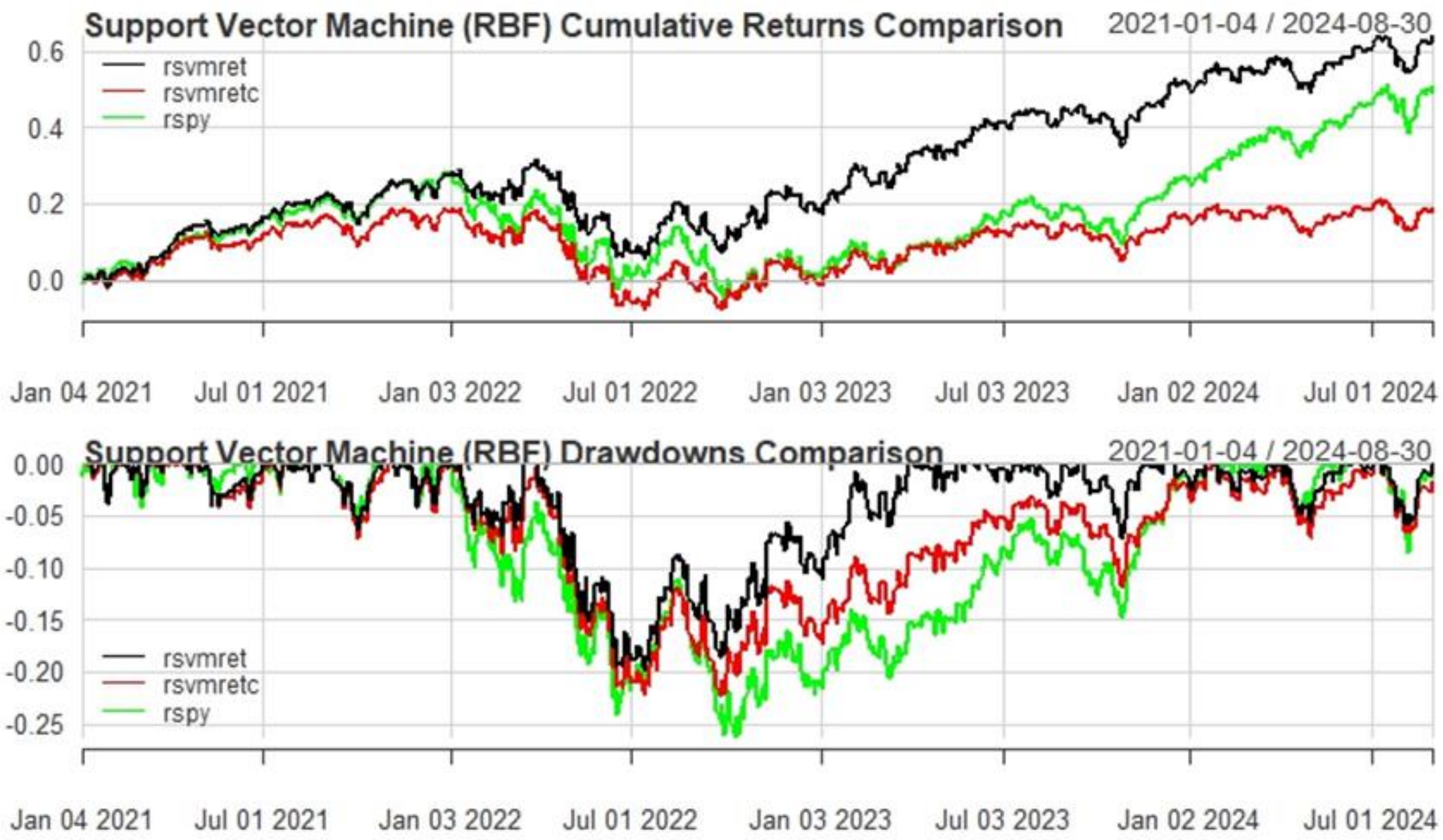

Multiple Linear Regression Cumulative Returns Comparison
2021-01-04 / 2024-08-30
lmret
lmretc
rspy
0.8
0.6
0.4
0.2
0.0
Jan 04 2021
Jul 01 2021
Jan 03 2022
Jul 01 2022
Jan 03 2023
Jul 03 2023
Jan 02 2024
Jul 01 2024
Multiple Linear Regression Drawdowns Comparison
2021-01-04 / 2024-08-30
0.00
-0.05
-0.10
-0.15
-0.20
-0.25
lmret
lmretc
rspy
Jan 04 2021
Jul 01 2021
Jan 03 2022
Jul 01 2022
Jan 03 2023
Jul 03 2023
Jan 02 2024
Jul 01 2024

XGBoost

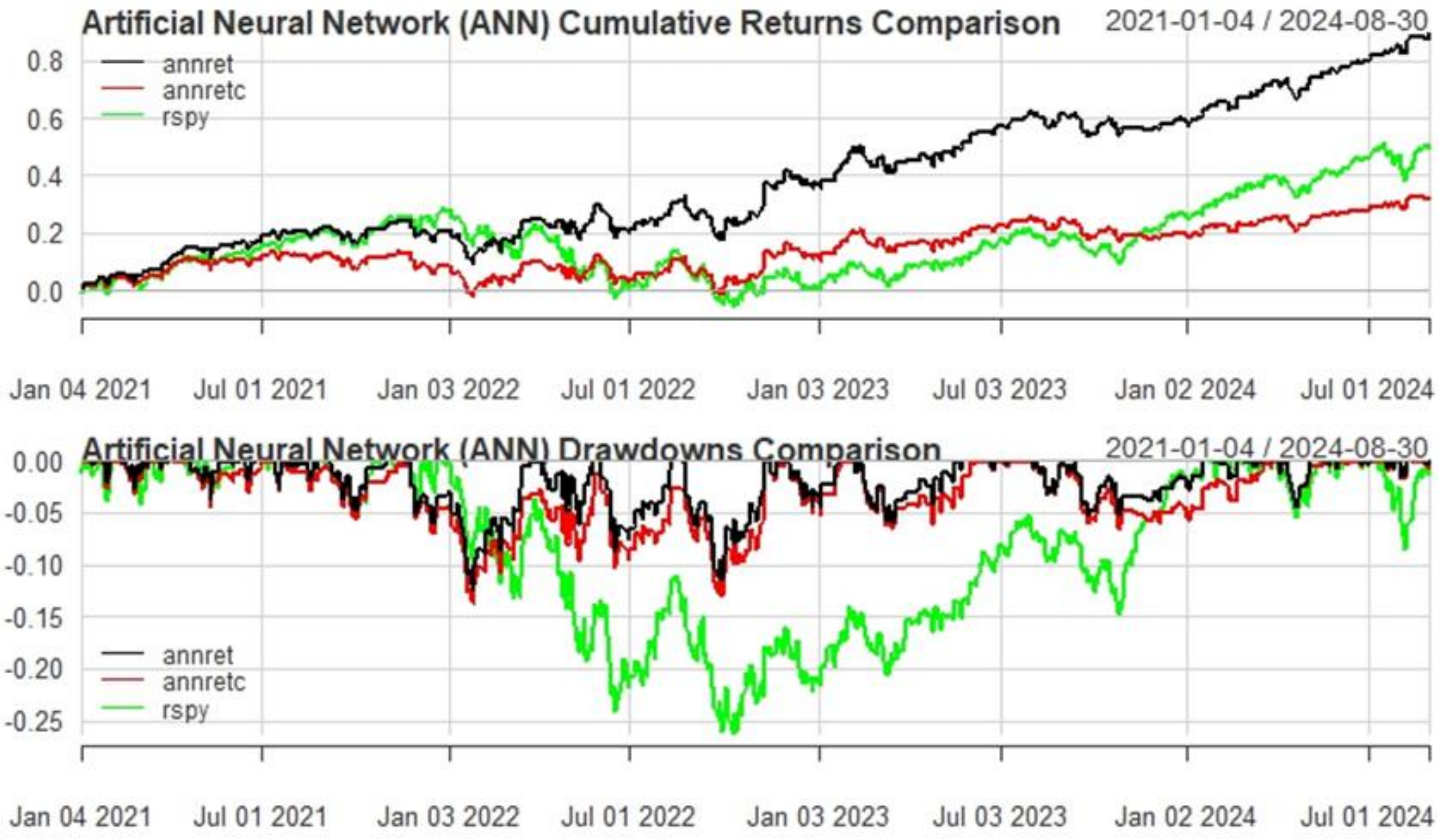


Ensemble Method Cumulative Returns Comparison
2021-01-04 / 2024-08-30
0.5
0.4
0.3
0.2
0.1
0.0
-0.1
xgbmret
xgbmretc
rspy
Jan 04 2021
Jul 01 2021
Jan 03 2022
Jul 01 2022
Jan 03 2023
Jul 03 2023
Jan 02 2024
Jul 01 2024
Ensemble Method Drawdowns Comparison
2021-01-04 / 2024-08-30
0.00
-0.05
-0.10
-0.15
-0.20
-0.25
xgbmret
xgbmretc
rspy
Jan 04 2021
Jul 01 2021
Jan 03 2022
Jul 01 2022
Jan 03 2023
Jul 03 2023
Jan 02 2024
Jul 01 2024

SVM

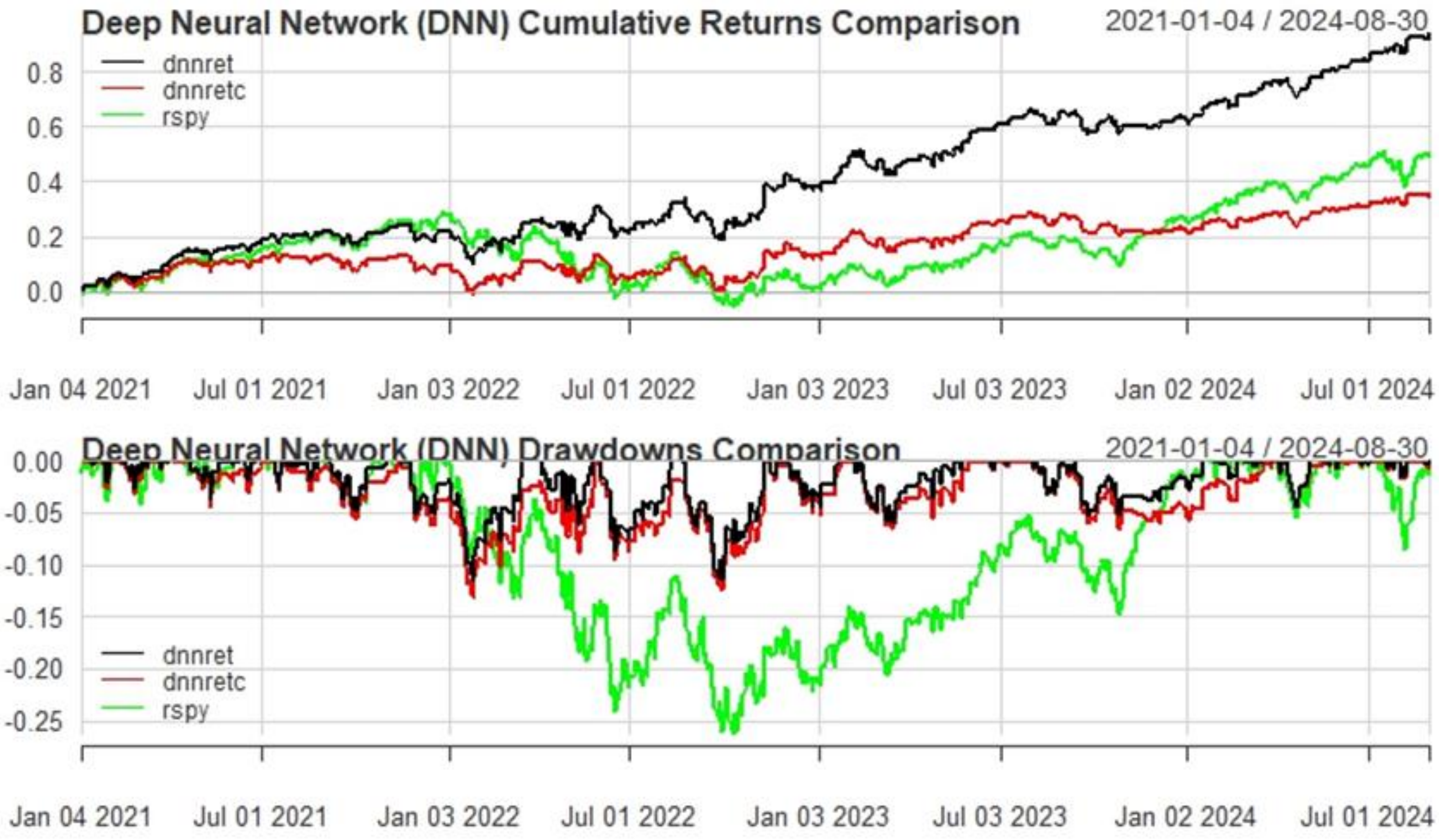

Support Vector Machine (RBF) Cumulative Returns Comparison
2021-01-04 / 2024-08-30
rsvmret
rsvmretc
rspy
0.6
0.4
0.2
0.0
Jan 04 2021
Jul 01 2021
Jan 03 2022
Jul 01 2022
Jan 03 2023
Jul 03 2023
Jan 02 2024
Jul 01 2024
Support Vector Machine (RBF) Drawdowns Comparison
2021-01-04 / 2024-08-30
0.00
-0.05
-0.10
-0.15
-0.20
-0.25
rsvmret
rsvmretc
rspy
Jan 04 2021
Jul 01 2021
Jan 03 2022
Jul 01 2022
Jan 03 2023
Jul 03 2023
Jan 02 2024
Jul 01 2024

ANN



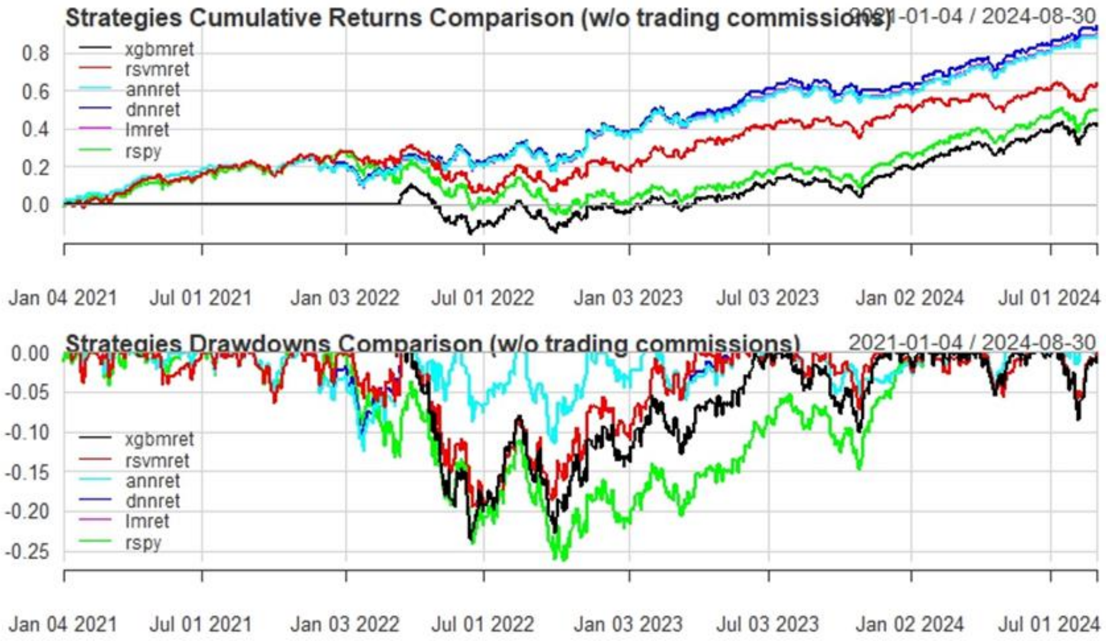
Artificial Neural Network (ANN) Cumulative Returns Comparison
2021-01-04 / 2024-08-30
annret
annretc
rspy
0.8
0.6
0.4
0.2
0.0
Jan 04 2021
Jul 01 2021
Jan 03 2022
Jul 01 2022
Jan 03 2023
Jul 03 2023
Jan 02 2024
Jul 01 2024
Artificial Neural Network (ANN) Drawdowns Comparison
2021-01-04 / 2024-08-30
0.00
-0.05
-0.10
-0.15
-0.20
-0.25
annret
annretc
rspy
Jan 04 2021
Jul 01 2021
Jan 03 2022
Jul 01 2022
Jan 03 2023
Jul 03 2023
Jan 02 2024
Jul 01 2024

DNN



Deep Neural Network (DNN) Cumulative Returns Comparison
2021-01-04 / 2024-08-30
dnnret
dnnretc
rspy
0.8
0.6
0.4
0.2
0.0
Jan 04 2021
Jul 01 2021
Jan 03 2022
Jul 01 2022
Jan 03 2023
Jul 03 2023
Jan 02 2024
Jul 01 2024
Deep Neural Network (DNN) Drawdowns Comparison
2021-01-04 / 2024-08-30
0.00
-0.05
-0.10
-0.15
-0.20
-0.25
dnnret
dnnretc
rspy
Jan 04 2021
Jul 01 2021
Jan 03 2022
Jul 01 2022
Jan 03 2023
Jul 03 2023
Jan 02 2024
Jul 01 2024

Comparative analysis



Trading results

Model	Trading Condition	Annualized Return	Annualized Std Dev	Annualized Sharpe	Maximum Drawdown
Multiple Linear Regression	w/o commissions	0.19260	0.13080	1.47300	0.12318
	w/ commissions	0.08200	0.13100	0.62590	0.14087
Extreme Gradient Boosting (XGBoost)	w/o commissions	0.10320	0.14160	0.72860	0.23630
	w/ commissions	0.09560	0.14180	0.67420	0.24092
Support Vector Machine (RBF)	w/o commissions	0.14510	0.15040	0.96470	0.19880
	w/ commissions	0.04850	0.15090	0.32150	0.22297
Artificial Neural Network (ANN)	w/o commissions	0.19150	0.13070	1.46460	0.12318
	w/ commissions	0.08100	0.13100	0.61810	0.13877
Deep Neural Network (DNN)	w/o commissions	0.19920	0.13070	1.52420	0.11479
	w/ commissions	0.08800	0.13100	0.67190	0.13157
Buy-and-Hold	w/o commissions	0.11890	0.16820	0.70690	0.26215

Why did MLR do well?

- MLR vs. AI Models: Surprisingly, Multiple Linear Regression's trading performance was among the top
- It nearly matched deep learning in return and Sharpe, and equally minimized risk (volatility, drawdown)
- The S&P 500 might have a lot of linear auto-correlation structure (e.g., mild momentum or mean reversion that a linear model can catch). The additional complexity of non-linear models did not add much extra predictive power (consistent with the similar accuracy metrics)
- MLR, being stable, might not overfit, thus giving consistent signals, whereas more complex models could occasionally predict spurious reversals
- Trading frequency: If MLR was a bit less aggressive in flipping signals than some AI models (like XGBoost may react to slight changes), it might have fewer trades, hence lower cost impact and steadier performance
- Cost-aware strategy design is crucial!

Practical implications for traders

Complex Models vs Simpler: If transaction costs are non-trivial, the complexity might not pay off. A simpler, perhaps slower-moving model (like regression) might yield nearly as good net performance with fewer trades

Interpretability and Trust: Traders might prefer MLR or simpler models if they perform similarly, because understanding why a signal is generated is easier (no black-box). This can be important in high-stakes financial decisions

Market Regime Consideration: In calm trending times, the simple model might do just fine. The AI models might prove their worth in highly volatile or complex pattern times, but those are not so frequent. So day-to-day, simpler could be more robust

When to prefer simpler models

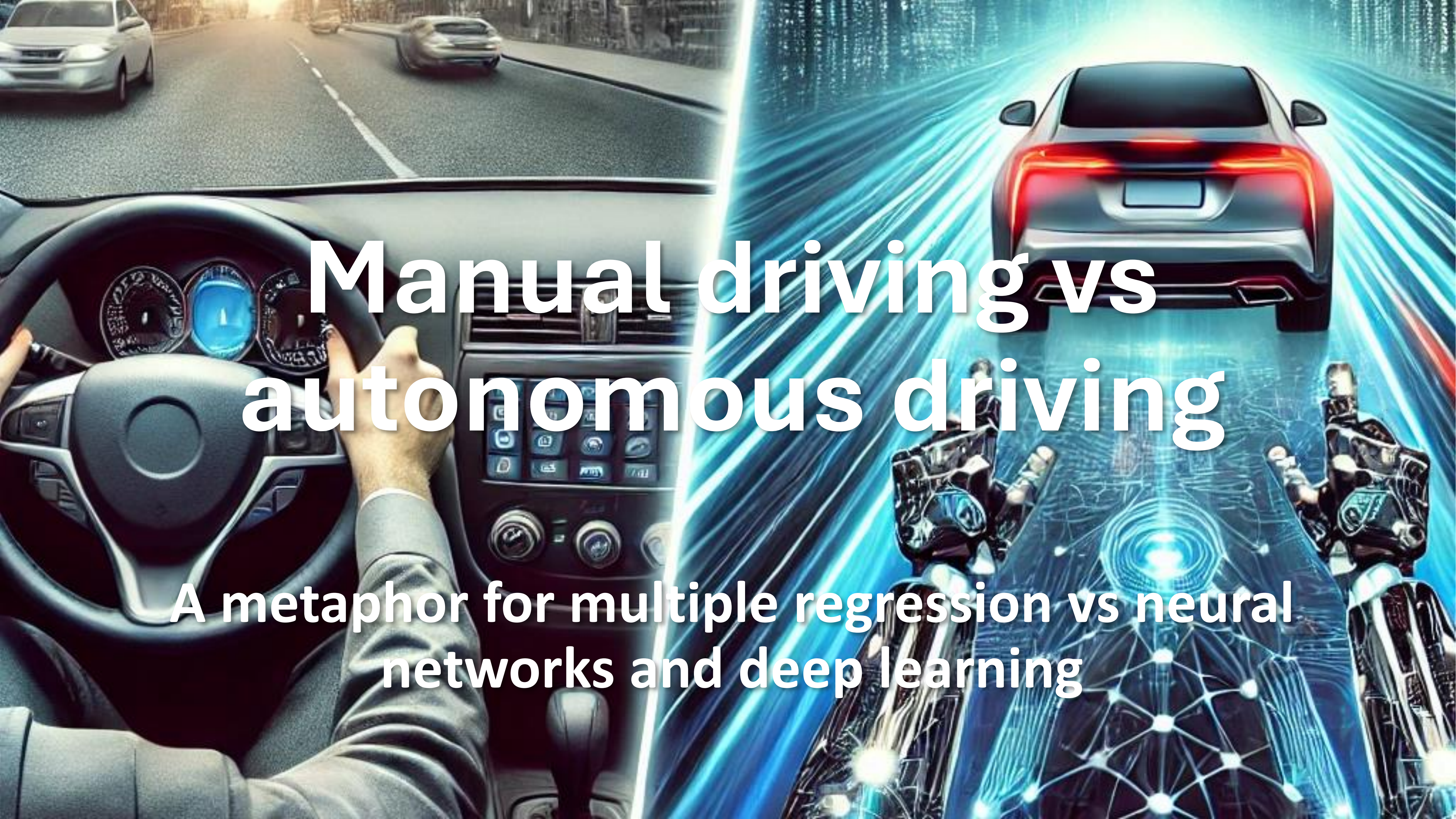
- Computational resources are limited (we see large runtime differences)
- Need for quick decisions (low latency, where simpler models shine)
- Market is efficient enough that linear models capture most signals
- Interpretability or regulatory necessity
- The strategy must be easily adjusted/communicated

R packages

Library	Usage / Purpose
caret	Core engine for train, tuning, and resampling - allows for model comparison and cross-validation
corrplot	Creates visual correlation matrices (heatmaps), helping spot multicollinearity among variables
forecast	Provides time series forecasting functions and 'accuracy()' for calculating RMSE, MAE, MAPE, etc.
kernlab	Implements kernel-based machine learning methods (SVM), used with the RBF kernel (svmRadial)
neuralnet	Builds and trains feedforward neural networks (ANN / DNN), specifying hidden layers, activation functions, etc.
PerformanceAnalytics	Performance and risk analysis tools for returns, drawdowns, charts, Sharpe ratios, and annualized metrics
quantmod	Simplifies financial data extraction from sources like Yahoo! and provides quick charting and transformations
tseries	Time-series tools (e.g., jarque.bera.test for normality checks), common in finance and econometrics
xgboost	Trains Extreme Gradient Boosting models (gradient-boosted decision trees) for structured data
writexl	Exports data frames to Excel (.xlsx) files, useful for saving outputs or performance logs
car	Contains additional regression diagnostics, including vif() for checking multicollinearity
ggplot2	Creating data visualizations (bar charts, line charts, etc.)

Final takeaway

- Traders and financial analysts should evaluate the trade-off between model complexity and practical benefits
- Simpler approaches often work surprisingly well and are easier to manage, whereas advanced models need to prove their worth through tangible improvements in predictive power or trading outcomes before displacing the trusty linear regression



Manual driving vs autonomous driving

A metaphor for multiple regression vs neural networks and deep learning


We're on a mission to
say… THANK YOU
Sweet Home Chicago!

R code snippets

R code snippets

```
# Univariate Filters
sbfctrlt <- sbfControl(functions=lmSBF)
sbft <- sbf(rspy~rspy1+rspy2+rspy3+rspy4+rspy5+rspy6+rspy7+rspy8+rspy9,data=rspyt,sbfControl=sbfctrlt)

# Recursive Feature Elimination
rfectrlt <- rfeControl(functions=lmFuncs)
rfet <- rfe(rspy~rspy1+rspy2+rspy3+rspy4+rspy5+rspy6+rspy7+rspy8+rspy9,data=rspyt,rfeControl=rfectrlt)

# Predictor Features Selection Embedded Methods
lassot <- train(rspy~rspy1+rspy2+rspy3+rspy4+rspy5+rspy6+rspy7+rspy8+rspy9,data=rspyt,method="lasso")
predictors(lassot)

# eXtreme Gradient Boosting Regression training
xgbmta <- train(rspy~rspy1+rspy2+rspy5,data=rspyt,method="xgbTree")
xgbmtb <-
train(rspy~rspy1+rspy2+rspy3+rspy4+rspy5+rspy6+rspy7+rspy8+rspy9,data=rspyt,method="xgbTree",preProcess="pca")
```

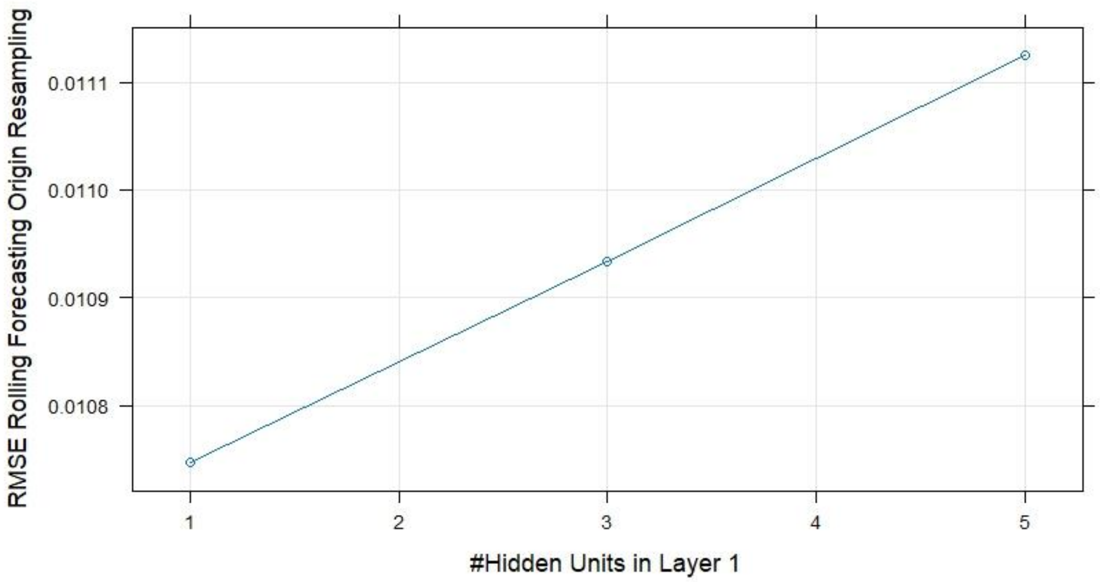
R code snippets

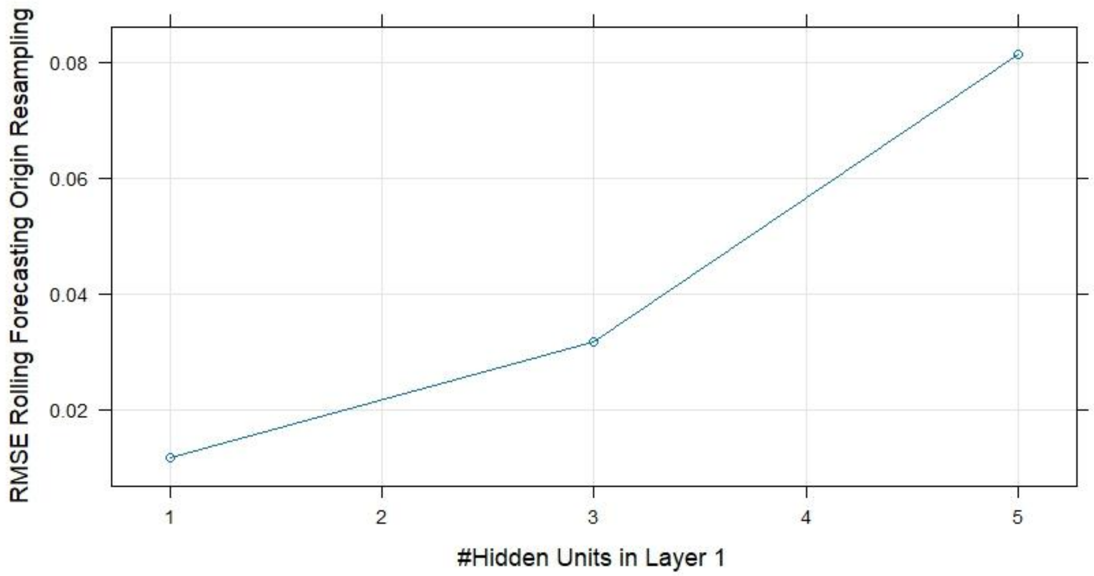
```
# Intermediate testing step as newdata needs to be same length as training range
xgbmpa <- predict.train(xgbmta,newdata=rspyp)
xgbmpb <- predict.train(xgbmtb,newdata=rspyp)

# Limited to testing range
xgbmdfa <- cbind(index(rspyp),as.data.frame(xgbmpa))
xgbmla <- xts(xgbmdfa[,2],order.by=as.Date(xgbmdfa[,1]))
xgbmfa <- window(xgbmla,start="2018-01-01")
xgbmdfb <- cbind(index(rspyp),as.data.frame(xgbmpb))
xgbmlb <- xts(xgbmdfb[,2],order.by=as.Date(xgbmdfb[,1]))
xgbmfb <- window(xgbmlb,start="2018-01-01")
```

R code snippets

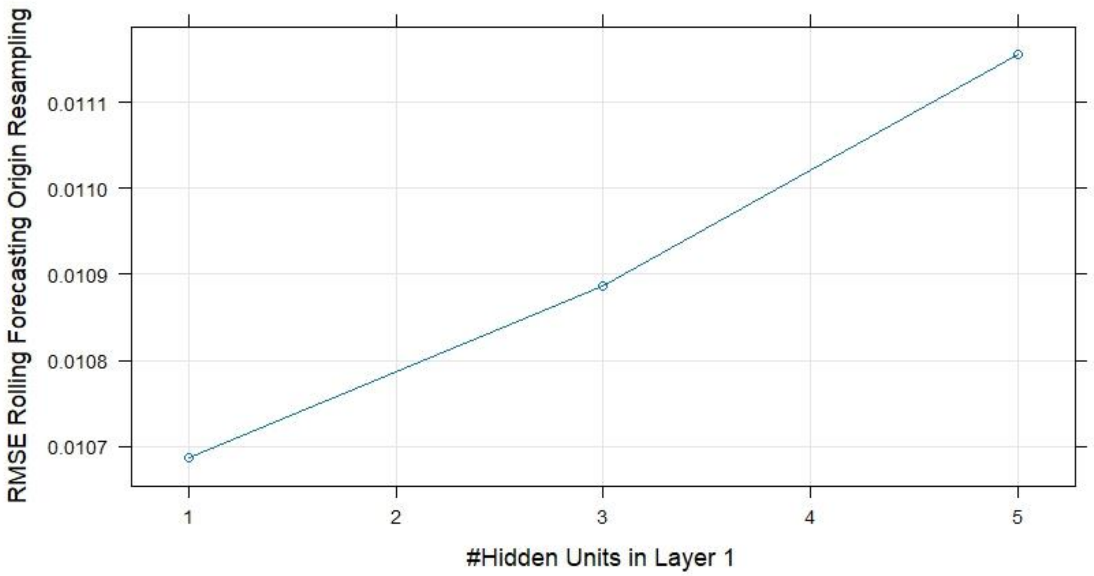
```
# Artificial Neural Network Regression training
annta <-
train(rspy~rspy1+rspy2+rspy5,data=rspyt,method="neuralnet",trContr
ol=tsctrlt)
anntb <-
train(rspy~rspy1+rspy2+rspy3+rspy4+rspy5+rspy6+rspy7+rspy8+rspy9,
data=rspyt,method="neuralnet", preProcess="pca",trControl=tsctrlt)
```

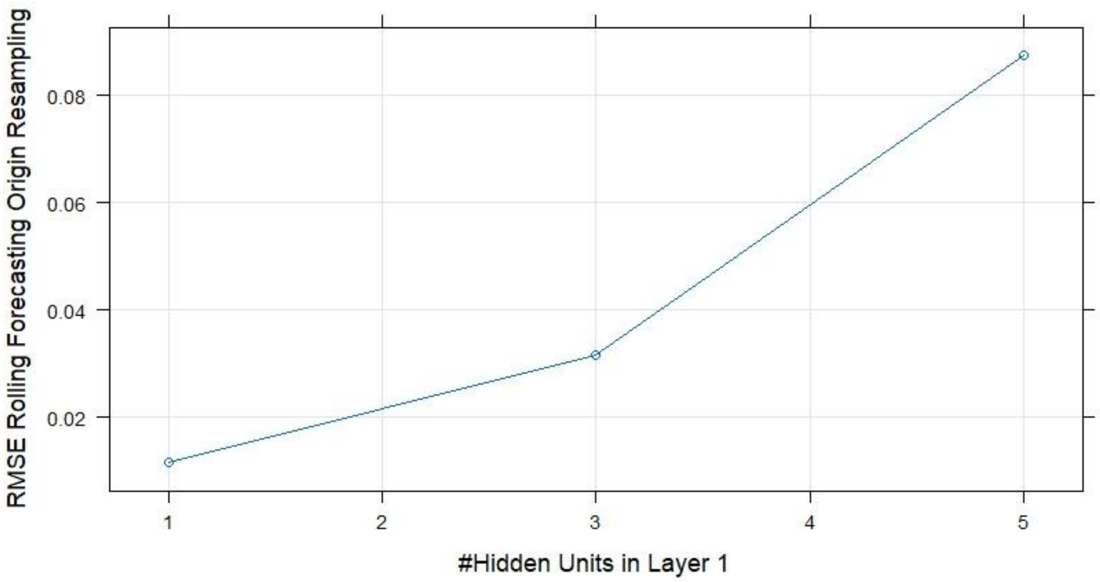




R code snippets

```
# DNN Regression training
dnnta <- train(rspy~rspy1+rspy2+rspy5,data=rspyt,method="neuralnet",trControl=tsctrlt)
dnntb <- train(rspy~rspy1+rspy2+rspy3+rspy4+rspy5+rspy6+rspy7+rspy8+rspy9,data=rspyt,method="neuralnet",preProcess="pca",trControl=tsctrlt)
```





R code snippets

```
# Deep Neural Network Regression trading signals
dnnsig <- Lag(ifelse(Lag(dnns)<0&dnns>0,1,ifelse(Lag(dnns)>0&dnns<0,-1,0)))
dnnsig[is.na(dnnsig)] <- 0
# Deep Neural Network Regression trading positions
dnnpos <- ifelse(dnnsig>1,1,0)
for(i in 1:length(dnnpos)) {
  dnnpos[i] <- ifelse(dnnsig[i]==1,1,
                   ifelse(dnnsig[i]==-1,0,
                          dnnpos[i-1]))}
dnnpos[is.na(dnnpos)] <- 0

# Multiple Linear Regression Method Trading Strategy Performance Comparison
lmret <- lmpos*rspys[,1]
lmretc <- ifelse(
  (lmsig==1|lmsig==-1) & lmpos!=Lag(lmpos),
  (lmpos*rspys[,1])-0.001,
  lmpos*rspys[,1])
lmcomp <- cbind(lmret,lmretc,rspys[,1])
colnames(lmcomp) <- c("lmret","lmretc","rspy")
table.AnnualizedReturns(lmcomp)
charts.PerformanceSummary(lmcomp,main="Multiple Linear Regression Method Daily Returns Comparison")
```