Analyzing Idiosyncratic Volatility – Evidence from Machine Learning

David Cao

November 17, 2023

Abstract

Ever since Ang et al (2006) found that idiosyncratic volatility has a negative impact on returns in data, the relationship between these two factors has been an ongoing controversy. The linear regression models they favor to analyze this puzzle explain about 5% of the variation in returns. This paper uses a neural network model to capture more of the variation in returns, and analyze the role that idiosyncratic volatility plays in predicting asset returns. Fascinatingly, my findings show that while idiosyncratic volatility does improve regression based forecasting, it does not add value to neural network forecasting. This suggests that idiosyncratic volatility impacts forecasting by reflecting an important feature of common risk factors that traditional models have yet to detect. Any unique information contained in volatility is not the reason it's important to forecasting. This methodology can be applied to evaluate the information composition of other controversial variables.

1 Introduction

Idiosyncratic volatility is a controversial risk factor in predicting stock returns. Within theoretical discussion, there are two dominant paradigms for how markets price idiosyncratic risk. In the first one based on the traditional capital asset pricing model (CAPM), investors are assumed to diversify idiosyncratic risk out of their portfolios. Under this assumption, investors are not exposed to idiosyncratic risk and it has no relationship with asset pricing or returns. In practice, however, many investors can not or will not diversify their portfolios. There are a lot of small investors–e.g. ordinary households–that do not have enough capital to hold stocks in multiple diverse assets. Even for larger investors, market frictions and other factors make it costly to diversify their portfolios completely.

Accomodating this fact, Merton (1987) presents an extension of CAPM that models the frictions preventing diversification. Within this second paradigm, investors remain exposed to idiosyncratic risk, and should demand compensation for that risk, leading to higher returns. The idea of higher risk yielding higher return is nice and intuitive, and most asset pricing theories agree with this premise.

The controversy emerges when analyzing empirical data and the actual relationship between the two variables. In a landmark paper, Ang et al (2006) performs a cross sectional analysis of one-month returns against an estimate for idiosyncratic volatility, and finds abnormally low returns for stocks with high idiosyncratic volatility. Their findings condradict both the relationship expected with CAPM and the more popular relationship proposed by Merton, presenting an anomaly.

In response to their results, much of the followup research attempts to reconcile these findings with the accepted theory. Barberis and Huang (2008) explore lottery preference of investors: the idea that investors favor stocks that experience extreme positive returns. Using a model based on cumulative prospect theory, they demonstrate that positively skewed assets

would have greater value assigned to them than expected utility or CAPM models would indicate. Bali et al (2010) and Boyer et al (2010) both perform an empirical analysts testing this hypothesis, using their own representations of skewness. Both of their findings show a negative coefficient of skewness on returns, though they disagree on how this impacts the sign of the volatility coefficient. Bali finds the coefficient reverses to a positive relationship, while Boyer finds that it becomes insignificant.

Huang et al (2009) develops another popular explanation based on market frictions: the anomaly tends to disappear over longer or shorter time frames, suggesting it may be a reflection of the return reversal effect or slow price movements. They use the previous month's returns as a variable to capture the return reversal effect, and use multiple measures of estimated volatility. When past returns is not in the regression, they find these volatility measures generally have negative coeffcients; controlling for past returns renders them insignifcant.

These two represent the most popular explanations for the puzzle, but there are plenty of other theories. To evaluate the overall set of explanations for the puzzle, Hou and Loh (2014) perform a decomposition analysis over a list of candidate variables. They find that variables based on lottery preferences capture 10-25% of the puzzle, variables based on market frictions capture 3-24% of the puzzle, and variables based on other explanations capture 5-10% of the puzzle. Collectively, the set of variables accounts for 29-54% of the coefficient of idiosyncratic volatility on returns, leaving most of the puzzle unexplained.

Other scholars dispute the existence of the puzzle, and their research confound matters further. Such papers argue there are flaws in the original paper's methodology, most often in how they estimate idiosyncratic volatility. Fu (2009) observes that their estimate assumes volatility is a random walk rather than a time varying value. Their own metric estimates volatility on EGARCH models, and they find a strong positive relation between this estimate and returns. Several studies would since adopt this method of estimating volatility and confirm their results.

Later research questions the validity of these results, showing Fu's estimate contains a look-ahead bias by including returns from month t–which is not available to traders. Guo (2014) demonstrates with simulated data that this bias can manifest a positive relationship between returns and volatility that does not exist in the data. Using a variant of the estimator that only includes data up to month $t-1$, they find that idiosyncratic risk has a negligible effect on asset returns. Overall, this conversation in the literature demonstrates that minor changes in methodology can reverse the volatility-returns relationship or cause it to disappear, adding yet more uncertainty to the debate.

However, as one overviews the literature, they might notice that all the research uses similar approaches to each other, and grapples with the same fundamental limitations. For one popular framework, portfolio analysis draws a broad picture of the impact of a single independent variable on a dependent one, with some ability to control for other variables. This occures by sorting firms in quintile or decile portfolios along idiosyncratic volatility, then reporting their value-weighted returns.. The researcher can see broad trends in returns based on volatility, without strong accumptions of the underlying structure.

The framework's weakness lies in how it controls for other variables. For example, if they're controlling for size, they would start by sorting firms into portfolios along firm size, then divide those portfolios into sub-portfolios sorted by volatility. This is effective for one or two variables, but breaks down for several variables at once. Portfolio analysis depends on having a larage number of firms in each portfolio, and each control variables shrinks that number of firms by another order of magnitude. One could reduce the number of divisions to compensate, but this sacrifices resolution. Ang (2006) handles this by performing portfolio analysis for each control variable one by one, but that approach would not be able to capture the effect of controlling for all of them at once; if anyone tried, they would only show just how sparse data is over multiples dimensions.

Complementing this, Fama-Macbeth's (1973) regression method fits the data to a linear model, offering a more refined analysis with easy to interpret parameters. This method can typically control for variables by including them in the regression, and seeing if the coefficient of volatility is subsumed. Most often, researchers will run the regression multiple times with different sets of control variables. Adding more variables does not impact the usable data, so it scales better to more complex environments.

Unlike portfolio analysis, Fama-Macbeth regression imposes a rigid structure on the underlying relationships between variables, and this limits the information it can capture from each variable. For many risk factors, the linear relationship does not inform their full impact on returns; their relationship is composed more of features such as nonlinearities or interaction effects with other factors. Linear models such as Fama-Macbeth do not intrinsically capture this information, and can only include these features if the researcher explicitly specifies them through an appropriate term. Ultimately the framework can approximate most important features, but this requires researchers to understand the underlying model in advance; if the relationship is too subtle or complex, then the model isn't likely to match the relationship.

Moreover, this framework expects parameters to be constant across firms. Lorek and Willinger (2009) demonstrate with empirical evidence that coefficients show considerable firm-specific variability, and fitting them as if equal loses a lot of performance.. Based on their own history or circumstances, different firms may be more sensitive to certain risk factors than other firms, and have their own parameters as a result.

Since the structure of linear regression restricts what it can capture, each study has a limited view of the relationship between idiosyncratic volatility and stock returns–using the most generous metrics, each paper's model explains about 5% of the variation in their dependent variable. Minor changes in methodology–such as different estimates of volatility–can shift which part of the bigger picture is viewed and draw contradictory conclusions of the same subject.

Much like the proverb of the blind men and the elephant, each paper is working with only a small part of the subject, and collectively misses too much to piece them together into a cohesive whole. A different approach would be needed to gain the bigger picture.

My paper aims to use a machine learning approach to evaluate the relationship between idiosyncratic volatility and stock returns, in an effort to shed light on the issue from a different framework. While there are many types of machine learning techniques, the ones I'm interested in are supervised methods, matching input variables to outputs with little to no parametrization. Such techniques are well suited to simulating or forecasting systerms with complex, nonlinear relationships between their variables. This is a much better description of the system that stock prices exist in, making machine learning models popular for forecasting applications.

After an evaluation of different methodologies in this category, I find that neural network methods work best at modeling this relationship, and its strengths complement the traditional methods nicely. By constructing a web of interconnected hidden variables, these methods can extract predictive features without requiring researchers to know them in advance. This enables them use information in the data that traditional methods wouldn't be able to access. In cases where complex or esoteric features are more important to forecasting, machine learning models will do a better job representing the underlying model.

To my knowledge, machine learning hasn't been used to study the idiosyncratic volatility puzzle. Though there is a large body of literature on machine learning in economics, the focus of this work is on its application in forecasting. Rather than using it as a tool for research, the endgoal is either to develop the algorithm or to evaluate it. The main reason for this is that machine learning models are difficult to interpret. In contrast to traditional methods, most machine learning methods do not include simple to interpret parameters, and do not make their structure obvious to any observer. For neural networks especially, the web of hidden variables is nearly impossible to comprehend.

I resolve this issue by starting off with a simple research question–is idiosyncratic volatility more valuable to prediction than a white noise variable? Framing it this way ensures I would only need to evaluate the model's performance, rather than any elements of its structure. I construct two datasets–one with the full set of control variables/systematic risk factors alongside idiosyncratic volatility, and one where the latter is replaced with white noise. The performance of neural network methods is sensitive to the number of input variables, so using white noise is preferred to removing the variable. By training the neural network over both datasets, I can compare how well they forecast over novel data and whether the neural network values the information in idiosyncratic volatility. I perform the same procedure within the cross sectional framework, to confirm the general literature's findings about the value of volatility.

My main contribution is finding that idiosyncratic volatility does not add much to prediction within the neural network framework, while it substantially improves prediction in the standard cross sectional framework. In addition, the neural network can explain over 90% of the variation in returns, while the cross-section is closer to 5%. This gap shows just how important the differences are between the two model frameworks and the kinds of information they can capture. Namely, returns cannot be predicted well using only information on the linear relationship between each risk factor and returns. The vast majority of the explanatory power of these variables is contained in more complex interactions and features.

The nature of these differences also give strong implications about the information composition of the idiosyncratic volatility variable, and exactly what makes it valuable to prediction. Idiosyncratic volatility isn't just white noise–else it would have no value in either framework, no matter the details in methodology. Nor does it offer valuable unique information–the neural network would certainly pick up on that. But recall that linear models can capture more complicated features, as long as they are explicitly specified. If some nonlinear feature of the dataset's control variables is important, not accounting for it will hurt the model's performance. But by

creating a term to encapsulate the feature, the model can draw a linear connection between that term and the output variable. This way, specifying features expands the set of information that linear models can use from the same dataset.

Neural networks are not receptive to the same assistance. By design, they will attempt to extract every feature of the data provided, testing each of them for importance. So if a researcher or analyst attempts to include a term to represent some complexity, the neural network will already be testing that feature. Information stored in complex features are already available to the neural network–specifying those features is redundant. For its performance to improve, any new term has to offer valuable unique information.

Giving how the models' performances changed, adding idiosyncratic volatility does not show the behavior of adding a new variable. It shows the behavior of specifying a feature. While the variable certainly contains information not found in common risk factors, that component doesn't appear to offer predictive value. Instead, it impacts prediction through a component that reflects features of common risk factors. For traditional models, adding idiosyncratic volatility presents features important to the relationship between common risk factors and returns, matching the linear model closer to the underlying model and letting it utilize important information it would not otherwise be able to capture.

Through this process, I find that even though machine learning models are difficult to interpret, how they respond to new variables can offer insight on the information composition of variables. Paired with traditional models, they can demonstrate what makes a variable important to forecasting and why–does the variable offer anything unique, or does its composition reflect an important undetected feature of the other variables? Using linear models alone, it can be difficult to tell if a variable's contribution reflects any new information, as the conflicting volatility literature demonstrates. Making use of machine learning techniques can isolate out overlapping information, answering this question.

The remainder of the paper proceeds as follows. Section 2 describes the most relevant machine learning models, and details how I evaluate each model and choose one to use for analysis. Section 3 describes the data. Section 4 lays out the methodology for goodness of fit testing. Section 5 presents the results, and Section 6 offers an in depth discussion of my findings. Section 7 summarizes and concludes.

2 Model Evaluation

Before using the data, I must first choose a machine learning method to analyze it through. I began by selecting a number of candidate models, as well as a selection of benchmark models. Candidate models are chosen among well-understood machine learning models, that are suited for forecasting continuous variables. Benchmark models are chosen based on common traditional techniques used in the literature. These are included to establish whether the candidate models actually improve prediction over the existing literature's methods, and ensure this investigation is worthwhile.

Once these were chosen, I simulated several datasets, and evaluated how well each model forecasted over each dataset. It is crucial to avoid using the real dataset when choosing the model. If I performed this analysis over real data first, then it's probably that one model will coincidentally fit the data better than others through no merit of its own. Evaluating them over multiple datasets allows me to judge their average performance over similar data, so erroneous conclusions are less likely.

That being said, data simulations cannot capture all patterns present in the real data, and a variety of features and wrong assumptions can render the simulated results a poor reflection of the real performance. Hence the conclusions need to be validated over real data. If the results over both show major discrepancies, then I would have to reevaluate the data simulation approach.

Models

I choose Ordinary Least Squares (OLS) as the first benchmark model. This is the baseline model for linear regression, predicting output variables by a linear combination of input variables; more formally, it fits the model $r_{i,t} = \beta' \mathbf{x}_{i,t-1} + \epsilon_{i,t}$. In economics, this is the framework for most parametric models, refined through carefully specifying its input variables and the scope of accepted inputs–among other possible modifications.

In its unmodified state, OLS using pooled data can only capture linear relationships between variables. As such, I expect it to perform the least well out of all models, which all have ways to estimate other patterns in the data. This is also the reason I include it in this test–its simplicity makes it valuable as a lower benchmark for forecasting performance. More sophisticated models will show improvement only if the patterns they're designed to capture exist within the data. If there is no real improvement, then the model doesn't work for the data.

The procedure in Fama-Macbeth (1973) is chosen as another benchmark model. This is another linear regression model, computed in a way that corrects for cross sectional correlations. For every time period in the data, a linear regression is performed over all available firms, to obtain a set of estimates $\hat{\beta}_t$. Then these estimates are averaged over, to obtain $\hat{\beta} = \frac{1}{T}\Sigma_t \hat{\beta}_t$. Such a method allows it to more accurately fit to each cross section within the data, without interference from other cross sections/time periods.

That being said, ultimately Fama-Macbeth uses the same information set and constraints as the pooled OLS model. I would expect this to result in similar, possibly slightly improved forecasts to the pooled OLS model in this application.

Fama-Macbeth is used incredibly frequently in the economic field, and as described in the introduction, is the go-to approach when studying the idiosyncratic volatility puzzle. The vast majority of research on the topic approach the puzzle by hypothesizing a plausible explanation,

constructing variables to represent it, and performing Fama-Macbeth regression to determine its relationship with either returns or IVOL, and how much of the coefficient of IVOL on returns can be explained by the candidate variable. Hence when choosing among cross sectional regression methods, this was the most fitting option, enabling me to use the methods of previous literature as a benchmark to compare other methods to.

ARIMA–developed by Box and Jenkins (1970)–is chosen primarily to represent a time series model among the benchmarks. Models such as these aim to capture time series relationships between a variable and past information, by using past outputs as another input. In this application, it fits the model $r_{i,t} = \beta' \mathbf{x}_{i,t-1} + \beta_1 r_{i,t-1} + \theta_1 \epsilon_{i,t-1} + \epsilon_{i,t}$. These are still easily understood linear models, and thus are popular in economics research, both for forecasting future outcomes and for drawing research conclusions. However, their performance scales drastically with the length of time that data covers–over more limited scopes or with sparse/missing information, the model is not suitable for use.

By expanding the scope of the model to time series forecasting, ARIMA has access to a larger pool of information to make each individual forecast. My implementation also fits over each firm's time-series, rather than pooled data, so it's capable of capturing firm specific parameters. Both of these factors make it likely to perform better than other parametric models over simulated data. Over real data, the method is far more sensitive to missing observations, and its performance in the simulations are not likely to be reflected.

The Fama-French Three-Factor model is an extension of the CAPM model, based on the efforts of Fama and French (1993) to more comprehensively capture systematic risk. In particular, it represents systematic risk through three variables: market risk premium $R_m - r_f$, SMB–the excess return of small caps over big caps, and HML–the excess return of value stocks over growth stocks. Due to these properties, this model is generally only used for prediction over diversified portfolios. It's still useful for firm-level analysis–its variables are borrowed and incorporated in

other models to represent systematic factors. In addition, being able to represent systematic risk means variation not explained by those factors can proxy for idiosyncratic risk–the standard method for estimating idiosyncratic volatility involves first estimating the Fama-French model, then taking the standard deviation of the residuals.

Fama-French does support firm specific parameters, but its restriction to systematic factors makes it use a more limited information set than any other model. Limiting its scope means it will likely perform better than OLS and Fama-Macbeth, but worse than ARIMA; how much so is tough to predict.

For an upper benchmark model, I use a semiparametric estimation method from Xiao (2010) for firm-specific coefficients. This method of kernel regression models coefficients as a function of firm-specific factors–in fact, its assumptions about the underlying structure of the data is the framework I used when constructing a data simulation method. After all, one of my primary concerns is that not every firm has the same response to exposure to risk factors, and thus are likely to have different parameters. This model simulates that idea nicely. Given the data simulation is based directly on this regression method, it will probably forecast over the simulated data better than any other model, including the machine learning models.

Note however that I do not intend to apply this model to real data, and it's disqualified as a final model candidate. I know which factor the coefficients are dependent on in the simulated data, but that's not information I can be certain of within the real data. Over simulated data where coefficients depend on one known variable, its performance will not be representative of its performance over real data, where determining coefficients is a higher dimensional problem over potentially unknown variables. It remains useful to see how this performs as an upper benchmark, since its performance represents the upper limit for performance in capturing firm-specific coefficients. Comparing the performance of each candidate method to the performance of this method should illustrate how well or poorly they can capture the same feature.

Among machine learning methods, two categories of models fit forecasting the best: decision tree methods and neural network methods. Decision trees operate by dividing data into branches based on common patterns, and continuing to divide the data up to some level of refinement. The result is a technique that can be applied to both classification and regression, using a comprehensible set of rules to match inputs to a classification or output. Sorting the data into increasingly fine groups leads to decision trees fitting extremely well to their training data, but also creates a tendency to capture erroneous patterns, and perform less well out-of-sample.

The variants that I choose to evaluate both address different potential flaws in the basic decision tree algorithm, albeit gaining complexity and losing comprehensibility in the process. The Random Forest method (Ho 1995) assumes that on average, any single decision tree will get the right results, and wrong predictions can be attributed to random noise. It resolves this the same way one typically handles standard error–law of averages. The method fits multiple decision trees, and takes the average of their predictions as the predicted output. If their assumption is correct for the dataset, then the majority of trees should give the correct prediction for any given input, ensuring any random errors caused by overfitting have minimal impact on any forecast.

A technique called gradient boosting (Friedman 2001) instead assumes that any errors in forecasting will be consistent across estimations, and wrong predictions need to be corrected for. But thanks to that consistency, the error itself is predictable. So to improve its estimate, gradient boosted trees work incrementally–after estimating an initial model, it runs the data and obtains the model's residuals, and fits another tree to the residuals. Doing this repeatedly should cause the final model to converge to an optimal decision tree, overcoming any sources of consistent errors.

Based on performance in general forecasting applications, I anticipate these models to perform best and most consistently on both real and simulated data. However, it's much tougher to predict which of the two will perform better compared to the other. Both make different assump-

tions about how basic decision trees perform on the data–random error or consistent error–and it's difficult to determine ahead of time which assumption is more correct for a particular dataset.

Neural network methods are well suited to nonparametric problems. These methods operate by constructing a network of three layered components–input layer, hidden layers, and output layer. Between the input and output layers, the neural network builds each hidden layer as an arbitrary number of cells. Each cell is connected to a random set of cells in the previous layer–that being the input layer in the case of the first hidden layer–and derive their value from some nonlinear function of the connected cells. Weights assigned to each of these connections determines the impact of any particular cell; the process of training the neural network is the process of continuously adjusting these weights until the output matches the data. This complicated structure allows them to capture and mimic nearly any complex relationship between inputs and outputs, making them useful in contexts where an explicitly defined algorithm is impractical. Conversely, the complexity of the network and its connections makes all such models a black box.

Recurrent Neural Networks (Rumelhart et al 1986) in particular show a lot of potential in modeling time series data, and is included as a candidate for this reason. In an RNN, information is processed one time period at a time, and cells are set up to observe previous values they've taken. By letting each cell use their previous states as inputs, RNNs can capture sequential and time series relationships and patterns. Hence they work particularly well in applications such as time series prediction and text analysis, where the sequence contains most of the meaning.

LSTM networks (Hochreiter, Schmidhuber 1997) are a variant of RNNs designed to address the vanishing gradients issue–the issue that older observations decay and vanish over the time horizon in RNNs, even for major events that should have a long lasting or permanent impact. To this end, each cell retains information it comes across–more similar to memory than observation–and is trained to decide under what conditions it will drop that information. In cases where

there are large gaps between important, impactful events, LSTM networks can remember them for longer and continue to factor them in predictions.

The data simulation method does not focus on the impact of big events, so RNN and LSTM are likely to perform similar to each other. However, major events can have long term impacts on stock returns in real data, persisting past the point when input variables revert to normal. So over real data, I would intuit that LSTM is better suited to prediction.

Both decision tree methods and neural network methods show consistently good performance in application, and it's difficult to anticipate which of the candidate machine learning models will perform the best.

Methodology

For the Monte Carlo design, I set up a data generating process to generate 240 time periods, 8000 firms, and 8 input variables $\mathbf{x}_{i,t-1}$ with one output variable $r_{i,t}$; this approximately matches the size and scope of the real dataset. Half of these variables follow a moving average, while the other half are distributed independently from previous time periods–this introduces a feature where only some variables have moving-average properties. More importantly, each firm is assigned a value z_i that is constant across time, and the coefficient for all input variables is a function of this firm-specific value $\beta(z_i)$. Output is then defined as a linear combination of these variables $r_{i,t} = \beta(z_i)'\mathbf{x}_{i,t-1} + \epsilon_{i,t}$, where $\epsilon_{i,t}$ is an error term following a standard normal distribution. For more details, please refer to Appendix B.

This data generating process focuses on simulating firm specific coefficients based on observable variables. It is possible to include other features such as more overt nonlinearity or cross-firm correlation, but it's also theoretically possible to account for them with a broad enough set of terms in a linear model. Hence the standard models in literature can contextualize them–the same does not apply to firm specific parameters, so I'm more interested in how each model

performs with this feature.

Each dataset is divided such that the first 180 time periods are used as training data, while the last 60 are used as a validation set. The training set is used to calibrate each benchmark and candidate model, then each model is tasked with forecasting over the validation set. Comparing the forecasts to the actual outputs, I compute mean squared error (MSE) and mean absolute deviation (MAD) measures to evaluate out of sample performance. While mean squared error is a reliable and more commonly used performance measure, it does have issues of excessively weighing outliers. Hence I make use of mean absolute deviation to validate its results.

Results

Table I

Evaluation of Benchmark and Candidate Models Over Monte Carlo Simulation

The table reports the mean MSE and the mean MAD of the selected models evaluated over 10 simulated datasets. Each dataset is generated using a Monte Carlo simulation, possessing 8 input variables and spanning across 240 time periods and 8000 firms. All models are calibrated over the first 180 time periods, then used to forecast over the last 60 time periods. MSE and MAD is calculated based on the difference between the forecasts and the actual outputs over this validatioin set. Standard deviations for these values are reported in parentheses.

Simulation Study Results	MSE	MAD
OLS	1.647	1.018
	(.008)	(.002)
Fama-Macbeth	1.647	1.018
	(.008)	(.002)
Fama-French	1.506	.976
	(.006)	(.002)
ARIMA	1.071	.826
	(.003)	(.001)
Kernel Regression	.999	.798
	(.002)	(.001)
Random Forest	1.487	.970
	(.005)	(.002)
Gradient Boosted Tree	1.210	.878
	(.003)	(.001)
RNN	1.067	.824
	(.0314)	(.012)
LSTM	1.020	.806
	(.003)	(.002)

Over the simulated data, mean squared error and mean absolute deviation corroborated each other–sorting candidate models by either measure results in the exact same order. Additionally, results are remarkably consistent between datasets, with very little standard deviation in MSE and MAD for all models. The only exception to this is in RNN, whose standard deviation for both values is five to ten times higher than that of other models.

Neither OLS nor Fama-Macbeth have any way to accommodate for firm specific parameters, which is an important element of the simulated datasets. Consequently, they perform very

similarly as the worst performing models. Just ahead of them, Fama-French does allow for firm specific parameters, but uses very limited information to forecast. This leads to better performance than the lower benchmarks, but worse than any other model.

The most surprising result here is random forest, which only slightly outperforms Fama-French. This contrasts against the performance of gradient boosted trees, which is much more accurate. Given the difference between the two styles, overall this suggests that decision trees face consistent errors when fitting over this kind of dataset, and noisy errors is less of a relevant issue.

As expected, kernel regression performs the best over simulated data; this is not likely to be the case when evaluated over a dataset not tailored to it. Of more interest is how the performance of other models compares to it, and if any come close to the same efficiency at forecasting with firm specific coefficients. Indeed, LSTM comes very close to the same performance, without requiring identifying which variables the coefficients depend on in advance. RNN shows similar effectiveness, and narrowly beats out ARIMA–which is also pretty well specified to the problem. This shows that the neural networks perform nearly as well as a regression model tailored to the underlying distribution.

3 Data

I choose to analyze NYSE, AMEX, and Nasdaq stocks over the period from January 2000 to December 2020, spanning 21 years and 252 months. Monthly stock returns data are obtained from CRSP.

I include Fama-French variables $R_m - r_f, SMB, HML$, downloaded from Kenneth R. French's website.

Other variables I use include systematic risk $BETA$, firm size, book-to-market ratio, liquidity measures in turnover and its coefficient of variation, momentum effects, and idiosyncratic

volatility. Each of these need to be calculated or estimated from other variables.

Firm size is set as market value of equity ME, calculated as the product of monthly closing price and outstanding share numbers (found in CRSP data).

Book-to-market BE/ME is yearend book value of common equity divided by yearend market value of equity. Annual book equity is obtained from COMPUSTAT data.

Turnover for a given month is calculated as trading volume divided by number of shares, and the variable $TURN$ used in the forecast is the mean turnover over the prior 36 months. $CVTURN$ is the coefficient of variation of that same set of turnovers.

Momentum effects are captured by geometric average of returns from month $t-7$ to month $t-2$; I added 100 to each returns value in order to average over a multiplier rather than a rate, as geometric average doesn't work for zero or negative values.

Since it's designed to capture systematic risk and ignore idiosyncratic risk, $BETA$ is more precisely estimated over diversified portfolios than individual firms. As such, I use Fama and French (1992)'s method to estimate this variable. For each firm-month I use the previous 60 months to estimate $r_{i,t} = \alpha + \beta(R_m - r_f)_{i,t} + \epsilon$, assigning a pre-ranking β to each stock. Then I assign stocks to 10x10 portfolios based on size and β deciles, with portfolios updating every month. On each portfolio, I estimate the full time-series regression of equally-weighted portfolio return on value-weighted market return and its one-month lagged value: $EVRET_{p,t} = a + b_{0,p}VWRET_t + b_{1,p}VWRET_{t-1} + \epsilon$. Portfolio beta is estimated as $BETA_p = b_{0,p} + b_{1,p}$, and assigned to every stock in the portfolio.

It is not possible to observe idiosyncratic volatility directly, so this variable needs to be estimated. Fundamentally, this is done by isolating the effect of systematic factors and seeing what variance remains in the returns data. This means that any estimator will work within some model framework, and thus the key variable cannot be entirely independent of existing models.

Regardless, the estimator should still be a good proxy for the real value, and any changes to its value should reflect differences in the true idiosyncratic volatility. I will keep in mind the potential for the estimation method to introduce artifacts, but I'm still confident in its results.

With this in mind, I choose to use the estimator AHXZ (2006) uses, which most other authors continued to use; this serves to make the data and results of the study more directly comparable to previous literature. Daily returns data is obtained from CRSP, and daily Fama-French factors are obtained from Kenneth R. French's website. In each month, daily return of individual stocks are regressed over the daily Fama-French factors, obtaining residuals after systematic risk is accounted for. Idiosyncratic risk is calculated as the standard deviation of those residuals. I exclude months with fewer than 15 trading days, to avoid any effects of infrequent trading.

Validation

Table II

Validation of Monte Carlo Results Over Real Data

The table reports the MSE and the MAD of the selected models over monthly returns data. Data is obtained from CRSP, and encompasses all NYSE, AMEX, and Nasdaq stocks, spanning from January 2000 to December 2020. All models are calibrated over the first 189 time periods, then used to forecast over the last 63 time periods. MSE and MAD is calculated based on the difference between the forecasts and the actual outputs over this validation set.

Empirical Data Results	MSE	MAD
OLS	.0210	.0843
Fama-Macbeth	.0248	.0944
Fama-French	.0227	.585
ARIMA	.0306	.0613
Kernel Regression	.0208	.0840
Random Forest	.0203	.0850
Gradient Boosted Tree	.0185	.0808
RNN	.0189	.0675
LSTM	.0189	.0674

When I look at each model over this data, differences in performance show how elements absent in the simulated data can change the outcome. Most saliently, kernel regression performs worse than any of the machine learning models. Unlike the simulated data, the real data has features it cannot capture such as nonlinearities; additionally, coefficients are likely affected by multiple factors, not all of them identifiable. Xiao (2010) uses firm size, and my implementation follows suit.

As expected, despite kernel regression performing the best over the Monte Carlo study, it's not feasible to get the same performance for real data. It maintains a restriction to linear relationships, and one can't perfectly identify the factors affecting coefficients in a real complex system the same way they can in a simulated system. So the performance here validates my decision to disqualify kernel regression from being the final model.

ARIMA's performance is hurt the most: this can likely be credited to the data showing variable timespan for each firm. The performance of time series regression scales heavily to timespan, and few firms are open for the entire period. I did not attempt to simulate missing data or variable timespans in the data simulation method, so the results line up with the expected contrast between datasets.

Looking only at MSE, gradient boosted trees are the best performing model on real data. It also shows significantly worse MAD, so I can't claim it performs strictly better than the neural network models. I choose not to take this as a strong indicator of a discrepancy with the simulated data, though it's certainly a complication.

Based on the results of the simulation study, and the validation against real data, LSTM network work best for forecasting data when firms may respond differently to exposure to the same risks. For this reason, I choose to advance my analysis analysis using LSTM networks.

4 Goodness of Fit Testing

Research Question

Though I've selected a model that should fit the data well, using it for evaluation is not a simple task. In contrast to parametric models such as Fama-Macbeth, neural network models such as LSTM are essentially a black box, and does not offer simple parameters to interpret. So to use the model for evaluation, I start with a modest research question–does idiosyncratic volatility have an impact on stock returns? If I can prove there is an effect in this framework, then I can move on to evaluating what that effect is.

In principle, goodness of fit R^2 can be used to answer this research question. R^2 only requires predicted values and real values of the output variable to be calculated, and does not depend on any elements of the model's structure. Hence by comparing the fit of a model without the principle variable, and the fit of a model with this variable, I can determine whether its addition adds anything to the prediction.

Analyzing goodness of fit will allow me to draw a conclusion from the model without needing to scrutinize the neural network too closely. If I do determine that idiosyncratic volatility is valuable, then the followup would be what kind of effect it has on stock returns–positive or negative. This would certainly require a more in-depth analysis of the fitted model, but is irrelevant if volatility has no effect.

Methodology

The central approach is to fit an LSTM model over the two datasets, and compare their R^2 values. For details of how I construct the LSTM model, refer to Appendix A. However, performing this test is not as simple as removing the principle variable to create the control dataset. When I've done this, the result is a consistent increase in out-of-sample R^2, despite

having less variables and information to work with. This even applies when the principle variable is unambiguously important to prediction, such as any of the Fama-French factors.

For machine learning methods, overfitting is a perpetual concern. While great at detecting patterns in data, these methods have trouble recognizing erroneous patterns and eliminating them from the model. Many factors impact the magnitude of this effect, but one of them is number of variables. When more variables are in the dataset, even more erroneous patterns get introduced and overfitting becomes a stronger concern. More often than not, this leads to seemingly poorer out-of-sample performance, as the results indicate.

The standard way to address this factor is to prune the set of variables, training the model on a more limited but more consistently informative dataset. Indeed, several techniques have been developed for determining which variables to keep and which to drop (for an overview, see Khalid, Khalil, and Nasreen 2014). However, our aim is not prediction, but analysis, and pruning variables would hamper that purpose. For prediction, maximizing performance is the main concern; for analysis, it is more important to keep the models comparable to each other. Using an algorithm to prune variables means dropping several control variables and not capturing their contribution in analysis. More notably, it is not likely that both datasets will end with the same number of variables, so overfitting pressure will still not be the same for both models.

Ultimately to solve this issue for analysis, I reasoned that rather than minimize this pressure, I should aim to keep it equivalent between both models. So I construct two datasets with the same number of variables: a dataset that contains both the control variables and the principle variable, and a control dataset that contains the control variables and a uniformly distributed white noise variable. Overfitting concerns from number of variables will impact both models in exactly the same way, so all differences in performance will be related to how the principle variable and the white noise variable affects forecasting. This reframes the question as "does idiosyncratic volatility add more to the prediction than a white noise variable?"

In addition, I also fit the datasets over two benchmark models in Fama-Macbeth and OLS. This is primarily to replicate what other papers have done over my dataset, ensuring there's no quirks in the dataset or problems introduced by the methodology. It would also help present a contrast with the results through machine learning.

Worth noting is that data is split into two groups: a training dataset covering the first 3/4 of the data, and a testing dataset covering the last quarter. For both the LSTM models and the benchmark models, the training dataset is used to fit the model. Testing data is used to evaluate out-of-sample performance where relevant.

5 Results

Efficiency Gain

Table III

Impact of Idiosyncratic Volatility on Stock Returns in Each Framework

The table reports the performance of each model on a control dataset and the full dataset, using the most representative measure of performance. LSTM uses testing R^2, which evaluates R^2 over the testing dataset rather than the data it's calibrated against. Fama-Macbeth uses mean R^2, which evaluates R^2 for each cross section as the model is being fit before aggregating both this value and the model. OLS uses the more traditional R^2, calculated in-sample. Over the control dataset, the idiosyncratic volatility variable is replaced with a uniformly distributed white noise variable; no modifications are made to the full dataset.

	LSTM	Fama-Macbeth	OLS
	Testing R^2	Mean R^2	Sample R^2
Control	.9066	.0469	.0082
IVOL	.9076	.0523	.0086

While all models use some variant of R^2 to evaluate their performance, different characteristics prompted me to abandon a unified approach. If I choose one of the given measures to compare each of their performances, invariably it will clash with some property of another model,

typically under-representing their accuracy. As such, for each model I choose an approach that best reflects their performance, and evaluate their efficiency against each other in broad strokes.

OLS uses the default method, calculating the R^2 value over the same sample it was fit over. When I evaluate it over the testing set instead, R^2 tends to be a negative number, which definitely does not reflect its accuracy.

Since Fama-Macbeth focuses on fitting to each cross-section–taking the average only to have an aggregate parameter–its performance is best measured by its fit to each cross-section. As such, most of the existing literature uses mean R^2. To calculate this measure, as one fits the cross-section they would evaluate its R^2. Then once all cross-sections have been fit, one would take the mean of the collection of R^2 values. This is the ideal way to evaluate the performance of cross-sectional models, but is completely inapplicable out-of-sample. Additionally, it serves no purpose with any other kind of model.

Uniquely, LSTM should not be evaluated with in-sample measures. Machine learning models in general fit very aggressively to their training sample, and require measures to limit overfitting and avoid catching erroneous patterns. Consequently, in-sample R^2 would over-represent its efficiency. I instead use the model to predict over the testing data, and calculate out-of-sample R^2 using these predictions. This evaluates how well the fitted model performs in a more general context, and best reflects its efficiency.

Comparing the machine learning model to the standard models, the results are striking. The LSTM network explains more than 90% of the variance of returns in the testing data, while Fama-Macbeth explains closer to 5% of each cross-section's variance. This suggests that LSTM captures a lot more of the information contained in the data than Fama-Macbeth does, and its results would be more reflective of the underlying reality.

Fundamentally, Fama-Macbeth is more limited in what kind of information it can capture.

It and many other standard models are linear models, and only capable of capturing linear relationships. This can be cheated through functional forms, either reworking a model into linear terms or adding variables to explicitly capture specific relationships–such as interaction terms. However, this only works for patterns and relationships one can anticipate; unknown or unanticipated patterns will not be captured.

LSTM and most machine learning models make no assumption about the underlying model. The way they are constructed, they can simulate many kinds of relationships between variables in different formats, without needing them explicitly defined. So any difference in their performance would depend on how much of the underlying relationships rest in more complicated patterns, versus linearity.

This finding clarifies and explains why the literature on the idiosyncratic volatility puzzle shows such contradictory results. The difference in performance suggests that most of the variation in returns can only be explained by features that linear models cannot detect, and thus the linear model is a poor fit for the problem. So the best each paper can work with is a linear approximation of a nonlinear relationship. Due to the mispecified model, each study can only see part of the relationship, and minor differences in methodology leads to them seeing different parts of it from each other. It's entirely likely that with their differences, one approximation will find a negative coefficient, while another would find a positive coefficient. Contradictory outputs are expected within a mispecified framework.

Idiosyncratic Volatility

While R^2 increases by a lot for Fama-Macbeth, the same does not apply to LSTM. Under a model that better reflects the data, idiosyncratic volatility provides roughly the same value for prediction as a white noise variable. Contradicting the bulk of the literature around this variable, these results suggest that idiosyncratic volatility is redundant for explaining stock returns.

What is likely occurring here is a story of three factors: the information contained in each variable, the information that cross-sectional models are capable of capturing, and the information that neural network models are capable of capturing. In this tale, idiosyncratic risk carries information about the stock, but this is not information unique to it. Rather, this variable reflects information derived from other variables, moving with and summarizing them in some nonlinear relationship.

In the linear models that most research relies on, it is possible to capture the effect of nonlinear interactions on the dependent variable, but only when it's explicitly specified as a parameter. So when the model excludes idiosyncratic volatility, it can capture the direct linear effect of its control variables, but not the impact of more complex interactions. Adding idiosyncratic volatility expands the model's framework, letting it capture more information about its component variables and improve the performance of its predictions.

Neural networks are designed to seek out any kinds of patterns or relationships in the inputs that can be used to predict the output, without any need for parametrization. This occurs by creating several layers of arbitrarily many nodes. Each node looks at a subset of nodes or variables in the previous layer, and represents one possible interaction or feature of that subset. Through this design, neural networks aggressively capture interaction effects between variables and other types of complexities.

With a large enough web of connections, neural networks can incorporate most possible configurations of the provided variables. One can surmise that if a candidate variable can be derived from other inputs, then at least one of the nodes would be analyzing a nearly identical term. Whether or not it's provided the candidate, the model captures and uses the same information. In both scenarios, the model would perform the same; we can conclude a variable is redundant if it does not offer relevant unique information.

Between these three factors, it is likely that idiosyncratic volatility offers value not through

unique information, but through its exposure to more complicated elements of the other risk factors. For the neural network framework, volatility becomes a redundant factor as its impact is already fully captured by its exposure to these elements. Cross-sectional models don't naturally capture those same elements; adding idiosyncratic volatility shows it information about common risk factors that it would not otherwise have access to. Hence rather than directly providing information on stock returns, idiosyncratic volatility can elucidate the effects of other variables through its nonlinear relationships with them; in nonparametric prediction, this variable can only reveal what it already knew.

To illustrate how this makes it important, let's have a deeper look at the standard linear regression model with n observations and k variables:

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{e}$$

Here, $\mathbf{y}$ is an $n \times 1$ vector representing observations of the dependent variable, $\mathbf{X}$ is an $n \times k$ matrix representing observations of the independent variables, $\mathbf{b}$ is a $k \times 1$ vector of unknown coefficients, and $\mathbf{e}$ is an $n \times 1$ vector of unobserved error terms. Without loss of generality, let's assume that $\mathbf{y}$ and $\mathbf{X}$ have been centered to have a mean of zero. To estimate $\mathbf{b}$, multiplying the equation by $\mathbf{X}'$ and rearranging terms yields,

$$\mathbf{X}'\mathbf{y} - \mathbf{X}'\mathbf{e} = \mathbf{X}'\mathbf{X}\mathbf{b}$$

$\mathbf{X}'\mathbf{e}$ will equal n times the covariance of $\mathbf{e}$ with each column of $\mathbf{X}$–i.e. the covariance of the error term with each independent variable. Since $\mathbf{e}$ is unobserved, this term remains unknown.

Solving for $\mathbf{b}$ yields,

$$\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\mathbf{y} - \mathbf{X}'\mathbf{e})$$

Due to the unobserved term, it is impossible to solve for $\mathbf{b}$ without making certain assumptions. Traditionally, least squares estimators assume that $\mathbf{X}'\mathbf{e} = \mathbf{0}$ in expectation. In other words, these assume that their is no covariance between any of the independent variables and the error term. This permits one to drop the last term, giving

$$\hat{\mathbf{b}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

How well this estimator performs depends on whether or not that assumption holds. To showcase its importance, let's substitute for $\mathbf{y}$ in the estimator, then take the expectation.

$$E[\hat{\mathbf{b}}] = E[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'(\mathbf{X}\mathbf{b} + \mathbf{e})]$$

$$= \mathbf{b} + E[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{e}]$$

$\hat{\mathbf{b}}$ is unbiased if $E[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{e}] = \mathbf{0}$, which only occurs when there is no correlation between $\mathbf{X}$ and $\mathbf{e}$. If the model is well specified, then this is an appropriate assumption and unbiased estimator. After all, the error term represents the variation in $\mathbf{y}$ that the model cannot explain. A well specified model will capture the full effects of its dependent variables on the independent variable, meaning there's no relationship with $\mathbf{e}$. When this is not true, any explanatory power not captured by the model will instead be captured by the error term. In this scenario, covariance can no longer be expected to be zero, and the model misrepresents the impact of its variables on $\mathbf{y}$.

In cases where the model doesn't capture some explainable variation, variables that can capture missing model features become very important–even if that's all it offers. Expanding $\mathbf{b}$ and $\mathbf{X}$ to include these variables ensures the regression does not omit an important part of the relationship. Consequently, it will no longer be represented within the error term. Even without offering unique information, this not only permits the model to explain more of the variation in $\mathbf{y}$, but also reduces error covariance and accomponying bias.

By analogy, what a neural network would do is aggressively extract model features, autonomously expanding the model. As it detects useful features of the variables in the data, it adds more columns to $\mathbf{X}$ and more coefficients to $\mathbf{b}$–explicitly specified features becomes redundant with this process. By its final state these have expanded to encompass all possible ways the original set of variables can explain returns. Such a thorough process ensures it will not be possible to explain any more of $\mathbf{e}$ through information derived from $\mathbf{X}$, and there is no relationship between these two elements. If one wants to explain more of the error term, they would have to find new variables with genuinely new information.

There's a more general case where the estimate for idiosyncratic volatility contains two components: information unique to the firm or industry, and information based on the more common risk factors. Based on how the neural network operates, it will improve its performance if the former component is related to returns, but not the latter component; linear models will respond to either component. If both components were valuable, then it would be reflected in an improvement in both models. If neither component was valuable, then neither model would improve with volatility included. Since the neural networks did not respond but the linear models did, it suggests that only the common risk component has any relationship with returns.

Overall, these findings elucidate how idiosyncratic risk contributes to returns prediction, and it's more complicated than additional information. Common risk factors have a number of features that are important to their relationship with asset returns. However, most of these features affect returns in a nonlinear manner, which can't ordinarily be captured by traditional linear models, so we only see part of the whole story. Idiosyncratic volatility mitigates this by encapsulating some of these important features, presenting it in a way that linear models can forecast with. By doing this, traditional methods can access more of the information in common risk factors, ensuring a better fit than if the variable was excluded. Despite lacking unique information, idiosyncratic volatility is still valuable for reflecting undetectable information. Without

the variable, the model suffers, losing insight on the features it encapsulates.

6 Discussion

Neural network methodologies are notable in how they do not require any model specification to function. As described previously, fitting a model using a neural network prompts it to test a complex network of patterns and features in the inputs, and how/if each of those features impacts the output variables. Hence by its process of training, the neural network will determine the structure of the relationships on its own–or at least a close approximation.

Due to this nature, a neural network's performance depends on two main factors: how much information it has, and the quality of the information. If the model learns from a biased dataset, it will internalize those biases, skewing its conclusions. So for best results, one would have to make sure they don't introduce any biases in the ways they construct their datasets or collect data. More important here is the issue of how much information it has. Neural network models are good at extracting most of the usable information in any variable or set of variables, but it won't be able to conjure up information that isn't present in any of its input variables. If it were asked to classify images, then only given the image size as inputs, it will not give useful results. Ultimately what matters to it is how much information its input variables covers, and its performance will be affected if some form of crucial information fails to be contained within the variable set.

For more traditional forms of modeling, it's vital to make the right assumptions about the structure of the underlying model. The researcher has to specify what ways their input variables can impact the output variables, and which features to pay attention to. If an input has a strictly linear impact on the output, linear regression methods will have no trouble fitting the relationship accurately. But more often, the underlying structure is more complicated than a system of linear relationships–a linear regression cannot properly capture diminishing or escalating returns, or any interaction effects of complimentary inputs. This changes if the researcher already knows

to look out for those features–functional forms such as logarithmic transformation or additional summary variables will frame nonlinear features in a way that linear regression can capture. So if the structure is known in advance, regression methods do a good job at fitting data and predicting how various risk factors affect returns.

The problem is that it's very difficult to know which features and structural elements to look out for in advance. While elements such as diminishing returns can be obvious, there are a lot of risk factors that can interact in a multitude of complicated ways. With a complicated system, several of these interactions will be impactful, but a person wouldn't have a way of telling which of the many possible interactions should be checked for. Hence in a circumstance where most available information can't be used, the model's performance is more impacted by how much information its input variables can communicate–in linear fashion–than how much information they contain. As far as linear regression is concerned, information being obfuscated is just as bad as it being missing.

In short, the neural network's goodness-of-fit corresponds to the total explanatory power of the input variables–with all of their features. The regression's goodness-of-fit corresponds solely to the explanatory power of the input's linear features–along with any explicitly specified elements. Looking back at the empirical results, a linear relationship between risk factors and returns only explains about 5% of the variation in returns, while a nonparametric model lets them explain about 90% of the variation in returns. This vast difference illustrates just how much information is concealed by model misspecification issues. Here the standard risk factors contain an impressive amount of information about asset returns, enough that when all the information is captured and used, only 10% of the variation in returns still needs explaining. But only about 5% of that information is accessible through linear regression–the rest can only be found in other features of the underlying model. Understanding the relationship between risk factors and returns will require understanding which features make up the remaining 95%

of the explanatory power.

Idiosyncratic volatility is important to traditional modeling, precisely because it exposes one of these hidden features. Regardless of whether it contains any unique information, it does represent some interaction between common risk factors. Looking at the Fama-Macbeth results, mean R^2 rises from .0469 to .0523 when this variable is added. This improvement suggests that the feature this represents is an important element of the underlying model, enough to improve prediction by over 10% when it's specified in the linear framework.

That being said, such an element is less valuable to neural network models. By their nature, neural network models test every possible feature of their inputs, in order to extract as much information out of them as possible for prediction. This means that whichever feature idiosyncratic volatility captures, the neural network already tests for. So if volatility can't offer unique information, its contribution is redundant with the efforts that the neural network already makes. My results shows that adding idiosyncratic volatility to the set of inputs has the same effect as adding a white noise variable. So idiosyncratic volatility does not expand the total set of information; anything it offers can be derived from the other risk factors.

To summarize, the neural network results show that idiosyncratic volatility does not contribute any unique information, and can only contribute through unearthing information that other models didn't detect. Fama-Macbeth results show that it does have a big contribution, and therefore reveals an especially important feature of common risk factors. Rather than acting as an independent variable, it becomes a vehicle for making the linear model more closely specified to the underlying structure of the data.

This opens up more questions about precisely which feature it's reflecting, as idiosyncratic volatility isn't intended to relate to common risk factors at all. After all, idiosyncratic volatility is meant to represent firm or industry specific risk factors, isolated from risk factors shared by everyone. Properly speaking, a firm's actual idiosyncratic risk has no relationship to common

risk factors, and presents purely unique information. If we accept this to be true, then the important feature must be a consequence of the popular estimators for idiosyncratic risk. The method popularized by AHXZ (2006) uses Fama-French factors to represent systematic risk and isolates its effect on volatility; this has the potential to leave in elements from the other common risk factors. And regardless of choice of variables, most alternative estimators assume a linear model of returns forecasting when identifying how much risk is systematic. Hence when making these estimates, the resulting variable is only insulated from linear relationships with other risk factors. All more complex elements of the risk factor-return relationship–nonlinearities, interaction effects, etc.–remain embedded in the estimate for volatility. The final variable will be effected by a mixture of these complexities that the base model could not capture, and genuinely unique risk factors. And for traditional models, the final impact on performance from idiosyncratic volatility will represent the combined effects of these two elements.

This applies in the traditional model because it does not intrinsically capture either of these features, but the same does not apply to neural network models. It is already going to factor any complexities within common risk factors into its predictions, so the only contribution it will detect is that of genuinely unique risk factors. And since the neural network did not detect any worthwhile contribution, this element does not appear to hold value. So if unique risk factors don't explain returns, but the estimate for idiosyncratic risk does, then the IVOL variable only contributes through exposure to elements from common risk factors. That it contains these elements at all amounts to a quirk in estimation.

So ultimately the variable improves forecasting by helping the linear model more closely match the underlying structure of returns forecasting. Many important elements of common risk factors will not be reflected in a linear model–estimated idiosyncratic volatility includes them in the model in a similar way to explicitly specifying them. Overall, it's suggestive of how important the element is, and how much weaker the model is for lacking it.

This explanation also can explain the inconsistent results in the greater body of literature. Though Ang's research popularized their way of estimating idiosyncratic risk, many other researchers favored different methods, with variations in the method or broader disagreements in what constitutes systematic risk. As such, each estimator likely captures or emphasizes different features of common risk factors. Some of these features will be more important. Some will be less important. Some will have a positive effect. Some will be negative. The results of the study can vary wildly, depending on which features its estimator or dataset ends up emphasizing; inconsistent results are to be expected.

Studying variables through neural network modeling is valuable for detecting cases such as this. Though its black box nature makes it less ideal for research, it responds differently to new information in ways that can reveal how and why a variable is important. Since it's only sensitive to unique information, it can verify whether the variable of interest adds anything unique, or if its value can be derived from other variables. The latter case would suggest that something significant is missing from the model.

7 Conclusion

In this paper, I analyze the relationship between idiosyncratic risk and stock returns by fitting monthly returns data over a neural network model, as well as a cross sectional model. Through comparing the predictive performance of both models with volatility vs a white noise variable, I quantify how much information each model extracts out of the same dataset, as well as how much information is provided by idiosyncratic volatility. Over the most appropriate goodness of fit estimators, LSTM explains approximately 90% of the variation in returns, while the traditional Fama-Macbeth model explains closer to 5% of the variation. This shows that most of the information useful to returns forecasting cannot be captured through linear methods; more complex features are much more important than linearity to the underlying model.

My results show that idiosyncratic volatility only performs as well as a white noise variable

in LSTM–a finding that contrasts against the results in most of the past literature. Over Fama-Macbeth, my results are more consistent with past literature, with a roughly 10% increase in explanatory power when volatility is in the data. This behavior is less consistent with a new useful variable, and more similar to the results of expanding the model by specifying a feature of the dataset. The impact of idiosyncratic volatility on returns is entirely captured by its nonlinear relationships with common risk factors–something not detected by linear models.

Overall these results show that there are a lot of features of common risk factors that are important to returns forecasting, but linear relationships aren't exposed to most of these features. The value of idiosyncratic volatility lies in its reflection of some of these important features, making them available to the model. While it may be redundant for investors using the latest technology, idiosyncratic volatility remains useful to understanding returns for this kind of insight.

Appendix A

To implement neural network forecasting, I made use of Python's tensorflow package. The model is constructed with 4 LSTM layers of 50 units/cells. Between each LSTM layer is a dropout layer set to randomly ignore 20% of cells during training. This is a measure against overfitting– each time the network updates, the dropout layer conceals a portion of the preceding layer from the following layer. Consequently as the following layer updates its weights, it evaluates itself against a random portion of the previous layer rather than the entire layer, with this portion changing with each iteration. This reduces the chance of the network focusing too much on erroneous patterns during training. Naturally, dropout layers are not used during forecasting.

The model's structure is capped with a one unit dense layer, generating an output from the network of hidden variables. Once the model is constructed, it is then compiled and trained using the "adam" optimizer and the mean squared error loss function. This network is trained over 50 iterations/"epochs".

Table IV

The LSTM Model Summary

Layer (type)	Output Shape	Parameters
lstm (LSTM)	(None, None, 50)	12200
dropout (Dropout)	(None, None, 50)	0
lstm_1 (LSTM)	(None, None, 50)	20200
dropout_1 (Dropout)	(None, None, 50)	0
lstm_2 (LSTM)	(None, None, 50)	20200
dropout_2 (Dropout)	(None, None, 50)	0
lstm_3 (LSTM)	(None, None, 50)	20200
dropout_3 (Dropout)	(None, None, 50)	0
dense (Dense)	(None, None, 1)	51
Total number of parameters: 72851		
Trainable parameters: 72851		
Non-trainable parameters: 0		

Appendix B

The goal of the Monte Carlo simulation was to generate datasets with a set of specific features in order to test how well each candidate model forecasts over these features. Specifically, the dataset should include firm specific coefficients, and variables with varying distributions and time-series properties.

In the first time period, $x_1, x_2 \sim N(0,1)$, $x_3, x_4 \sim U(0,1)$, $x_5, x_6 \sim U(-1,1)$, $x_7, x_8 \sim Binom(.5)$ for each firm. In following time periods, x_2, x_4, x_6, and x_8 follow the same distribution–independent from previous time periods–while x_1, x_3, and x_5 follow a moving average of $x_{j,t} = .9x_{j,t-1} + .1\epsilon_{j,t}$ where $\epsilon_{j,t}$ has same distribution as $x_{j,0}$. Similarly, x_7 is assigned a 90% chance of staying the same in the next time period. This construction captures some of the common variable distributions in the data, and creates a feature where only some variables have moving-average properties.

Each firm is assigned its own constant value $z_i \sim U(-1,1)$, which is used to assign coefficients for each variable for each firm, using the following underlying formulas.

- $\beta_1(z_i) = \ln(2 + z_i)$
- $\beta_2(z_i) = z_i^3$
- $\beta_3(z_i) = -e^{-z_i}$
- $\beta_4(z_i) = \cos(\pi z_i)$
- $\beta_5(z_i) = -z_i$
- $\beta_6(z_i) = \sin(5\pi z_i + .25\pi)$
- $\beta_7(z_i) = .4$
- $\beta_8(z_i) = -.4$

Output is then calculated via $r_{i,t} = \beta(z_i)'\mathbf{x}_{i,t-1} + \epsilon_{i,t}$, where $\epsilon_{i,t}$ follows a standard normal distribution.

This data generating process is used to simulate 10 datasets, each providing their own evaluation of every candidate model. Accuracy of each model is measured with MSE and MAD over each of these datasets, then I take the mean and standard deviations of these measures to judge their overall performance.

References

Ang, A., Hodrick, R.J., Xing, Y., Zhang, X., 2006. The cross-section of volatil-ity and expected returns. Journal of Finance 61, 259-299.

Bali, Turan G., Cakici, Nusret, Whitelaw, Rogert F., 2011. Maxing out: Stocks as lotteries and the cross-section of expected returns. Journal of Financial Economics 99, 427-446.

Barberis, N., Huang, M., 2008. Stocks as Lotteries: The Implications of Probability Weighting for Security Prices. The American Economic Review 98, 2066-2100.

Bergbrant, Mikael, Kassa, Haimanot, 2021. Is idiosyncratic volatility related to returns? Evidence from a subset of firms with quality idiosyncratic volatility estimates. Journal of Banking and Finance 127.

Boehme, Rodney D., Danielsen, Bartley R., Kumar, Praveen, Sorescu, Sorin M., 2009. Idiosyncratic risk and the cross-section of stock returns: Merton (1987) meets Miller (1977). Journal of Financial Markets 12, 438-468.

Box, G., Jenkins, G., 1970. Time Series Analysis: Forecasting and Control. Holden-Day, San Francisco.

Boyer, Brian, Mitton, Todd, Vorkink, Keith, 2010. Expected Idiosyncratic Skewness. The Review of Financial Studies 23, 169-202.

Cao, Jie, Chordia, Tarun, Zhan, Xintong, 2021. The Calendar Effects of the Idiosyncratic-Volatility Puzzle: A Tale of Two Days? Management Science 67.

Chabi-Yo, Fousseni, Yang, Jun, 2009. Default Risk, Idiosyncratic Coskewness and Equity Returns. Unpublished working paper. Ohio State University.

Chen, Honghui, Zheng, Minrong, 2021. IPO underperformance and the idiosyncratic risk puzzle. Journal of Banking and Finance 131.

Chen, Linda H., Jiang, George J., Xu, Danielle D., Yao, Tong, 2020. Dissecting the idiosyncratic volatility anomaly. Journal of Empirical Finance 59, 193-209.

Engle, Robert F., Hansen, Martin Klint, Karagozoglu, Ahmet K., Lunde, Asger, 2021. News and Idiosyncratic Volatility: The Public Information Processing Hypothesis. Journal of Financial Econometrics 19, 1-38.

Fama, Eugene F., French, Kenneth R., 1992. The Cross-Section of Expected Stock Returns. Journal of Finance 47, 427-465.

Fama, Eugene F., French, Kenneth R., 1993. Common risk factors in the returns on stocks and bonds. Journal of Financial Economics 33, 3-56.

Fama, Eugene F., MacBeth, James D., 1973. Risk, Return, and Equilibrium: Empirical Tests. Journal of Political Economy, 81, 607-636.

Fink, Jason D., Fink, Kristin E., He, Hui, 2012. Expected Idiosyncratic Volatility Measures and Expected Returns. Financial Management 41, 519-553.

Friedman, J. H., 2001. Greedy function approximation: a gradient boosting machine. Annals of Statistics, 1189-1232.

Fu, Fangjian, 2009. Idiosyncratic Risk and the Cross-Section of Expected Stock Returns. Journal of Financial Economics 91, 24-37.

Garcia, R., Mantilla-GarcÃa, D., Martellini, L.,2014. A Model-Free Measure of Aggregate Idiosyncratic Volatility and the Prediction of Market Returns. The Journal of Financial and Quantitative Analysis 49, 1133-1165.

Guo, H., Kassa, H., Ferguson, M. F., 2014. On the Relation between EGARCH Idiosyncratic Volatility and Expected Stock Returns. The Journal of Financial and Quantitative Analysis, 49, 271-296.

Han, Yufeng, Lesmond, David, 2011. Liquidity Biases and the Pricing of Cross-sectional Idiosyncratic Volatility. The Review of Financial Studies 24, 1590-1629.

Ho, T. K., 1995. Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition 1, 278-282.

Hochreiter, S., Schmidhuber, J"urgen, 1997. Long short-term memory. Neural Computation 9, 1735-1780.

Hou, Kewei, Loh, Roger K., 2016. Have we solved the idiosyncratic volatility puzzle? Journal of Financial Economics 121, 167-194.

Huang, Wei, Liu, Qianqiu, Rhee, S. Ghon, Zhang, Liang, 2010. Return Reversals, Idiosyncratic Risk, and Expected Returns. The Review of Financial Studies 23, 147-168.

Hur, Jungshik, Singh, Vivek, 2022. The role of investor attention in idiosyncratic volatility puzzle and new results. Review of Quantitative Finance and Accounting 58, 409-434.

Jondeau, Eric, Zhang, Qunzi, Zhu, Xiaoneng, 2019. Average skewness matters. Journal of Financial Economics 134, 29-47.

Lorek, K.S., Willinger, G.L., 2009. New evidence pertaining to the prediction of operating cash flows. Rev Quant Finan Acc 32, 1-15.

Lorek, K.S., Willinger, G.L., 2011. Multi-Step-Ahead Quarterly Cash-Flow Prediction Models. Accounting Horizons 25, 71-86.

Ludvigson, Sydney C., Ng, Serena, 2007. The empirical risk-return relation: a factor analysis approach. Journal of Financial Economics 83, 171-222.

Merton, R. C.,1987. A Simple Model of Capital Market Equilibrium with Incomplete Information. The Journal of Finance 42, 483-510.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. Nature 323, 533-536.

S. Khalid, T. Khalil and S. Nasreen 2014. A survey of feature selection and feature extraction techniques in machine learning. Science and Information Conference, London, UK, 2014, 372-378.

Stambaugh, R. F., Yu, J., Yuan, Y., 2015. Arbitrage Asymmetry and the Idiosyncratic Volatility Puzzle. The Journal of Finance 70, 1903-1948.

Wan, C., Xiao, Z., 2014. Idiosyncratic Volatility, Expected Windfall, and the Cross-Section of Stock Returns. Essays in Honor of Peter C. B. Phillips (Advances in Econometrics, Vol. 33), 713-749.

Zhang, Xindong, Li, Jianying, Wang, Xiaoli, Hu, Xiaoxin, 2021. The Idiosyncratic Volatility Puzzle: A Time-Specific Anomaly. Journal of Mathematical Finance 11, 294-312.