Recovering Predictability from the Implied Volatility Surface: A Vision Transformer Approach[*]

September 21, 2025

Abstract

Placeholder

Keywords: key1, key2, key3

JEL Codes: key1, key2, key3

[*] abc

1 Introduction

2 Literature Review

3 Data

We obtain the equity and index option data from the IvyDB OptionMetrics database, which provides Volatility Surface files (vsurfdYYYY) that contain the interpolated Black-Scholes implied volatilities, starting from January 1996. For each security on each day, a volatility surface with standard maturities of τ days to expiration and moneyness levels measured by option δ is provided. [1]

We select out-of-the-money (OTM) and near at-the-money (ATM) put options with delta levels in [-0.5, -0.1], as well as OTM and near ATM call options with delta levels in [0.1, 0.5], since the deep-in-the-money call options are often illiquid, following Martin (2017), among others. The options are rearranged by their moneyness (implied strike), so the delta levels start from -0.1 to -0.5, then from 0.5 to 0.1 [2]. We also drop the options with maturities equal to 10 days, since the large fraction of missing values. The final volatility surface data contains 10 maturities and 18 delta levels, and a example of the volatility surface is shown in Figure 2. The input features, in result, can be represented as $IV_{i,t} = \{IV_{i,t}(\tau, \delta)\}_{\tau \in T, \delta \in \Delta}$. the volatility surface of security i at time t.

We obtain the daily return data from CRSP for all firms listed on NYSE, AMEX, and NASDAQ, as well as the S&P 500 index. For each security i at time t, we calculate the

[1] OptionMetrics use a methodology based on a kernel smoothing algorithm to interpolate the volatility surface. The maturities are 10, 30, 60, 91, 122, 152, 182, 365, 547, 730 days to expiration, and the option delta levels are from 0.1 to 0.9 with a step of 0.05, positive for call options and negative for put options.

[2] The implied strike order between put option with delta -0.5 and call option with delta 0.5 is uncertain, but their spread is proved to have predictive power for future returns (Yan, 2011), so we keep both of them.

cumulative return on security i from day t to day $t + H$ as:

$$r_{i,t}^{H} = \prod_{j=0}^{H-1} (1 + r_{i,t+j}) - 1 \tag{1}$$

We link the volatility surface data from OptionMetrics with the return data from CRSP using a linking table provided by WRDS. The final dataset spans from January 1996 to August 2023, and Figure 1 shows the coverage of the stocks with available volatility surface data over time. The number of stocks with available volatility surface data increases significantly over time, from below 1000 to over 5000 in recent years.

4 The Vision Transformer Model

In this section, we briefly introduce the vision transformer (ViT) model architecture and our experimental design.

4.1 Brief Introduction of the Vision Transformer Model

First introduced by Vaswani et al. (2017), "Attention Is All You Need," the Transformer model has become a foundational architecture in deep learning, particularly for natural language processing (NLP). Unlike its predecessors, such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, the Transformer does not rely on sequential data processing. Instead, it processes the entire input sequence at once, using a sophisticated mechanism known as self-attention to weigh the importance of different words in the sequence.

The core of the Transformer is its encoder-decoder structure. The encoder's role is to map an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $z = (z_1, ..., z_n)$. The decoder then takes z and generates an output sequence $(y_1, ..., y_m)$, one symbol at a time.

Building on the success of the Transformer in NLP, the Vision Transformer (ViT) was proposed by Dosovitskiy et al. (2020) to apply the same architecture to computer vision tasks, offering a compelling alternative to the widely used Convolutional Neural Networks (CNNs). The fundamental idea behind ViT is to treat an image as a sequence of patches, analogous to how a sentence is treated as a sequence of words.

By converting images into a sequential format, ViT demonstrates that the reliance on convolutions is not a necessity for vision tasks and that a general-purpose attention-based architecture can achieve state-of-the-art (SOTA) performance, especially when pre-trained on large datasets.

The architecture of the ViT, shown in Figure 4, adapts the original Transformer in the following way:

1. **Image Patching and Embedding**: The ViT model first reshapes the input image from a 2D grid of pixels into a sequence of flattened 2D patches. For a image $x \in \mathbb{R}^{H \times W \times C}$, it is divided into N patches $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where (H, W) is the height and width of the original image, C is the number of channels (e.g., RGB), and (P, P) is the resolution of each patch, and $N = (H \times W)/(P^2)$ is the number of patches. These patches are then flattened and mapped to a latent D-dimensional embedding space through a trainable linear projection. Each patch is taken as a token, similar to words in NLP.

2. **Learnable Class Token**: Inspired by the [CLS] token used in BERT (Bidirectional Encoder Representations from Transformers) model (Devlin et al., 2019), a learnable embedding is prepended to the sequence of patch embeddings. The state of this token at the output of the Transformer encoder serves as the aggregate image representation for classification tasks.

3. **Positional Embeddings**: Similar to the positional encodings in the original Transformer, the ViT adds learnable 1D positional embeddings to the patch embeddings to

retain spatial information. These embeddings allow the model to learn the relative positions of the image patches.

The flattened patches, along with the class token and positional embeddings, are served as input to the Transformer encoder:

$$\mathbf{z}_0 = [\mathbf{x}_{class}; \mathbf{x}_p^1 E; \mathbf{x}_p^2 E; ...; \mathbf{x}_p^N E] + E_{pos} \tag{2}$$

where $E \in \mathbb{R}^{(P^2 \cdot C) \times D}$ is the patch embedding projection matrix, $E_{pos} \in \mathbb{R}^{(N+1) \times D}$ is the positional embedding matrix, and $\mathbf{z}_0 \in \mathbb{R}^{(N+1) \times D}$ is the resulting sequence of embedded patches, with length $N + 1$ (including the class token) and embedding dimension D.

4. **Transformer Encoder**: The resulting sequence of embedded patches (including the class token) is then fed directly into a standard Transformer encoder (Vaswani et al., 2017). This encoder is composed of alternating layers of multi-head self-attention (MSA) and multi-layer perceptron (MLP) blocks, similar to the original Transformer encoder, but with some slight modifications:

   - **Multi-Head Self-Attention (MSA)**: This mechanism allows the model to focus on different parts of the input sequence simultaneously, capturing various aspects of the image. Each attention head computes a weighted sum of the input embeddings, where the weights are determined by the similarity between the query and key vectors. These similarity scores are calculated using Query (Q), Key (K), and Value (V) matrices, which are linear projections of the input embeddings. The attention scores are computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{3}$$

   where d_k is the dimension of the key vectors, used for scaling. The multi-head attention mechanism allows the model to jointly attend to information from different representation subspaces at different positions. For layer $l = 1, ..., L$, the MSA block can be expressed as:

$$\mathbf{z}'_l = \mathrm{MSA}(LN(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1} \tag{4}$$

where $LN(\cdot)$ denotes layer normalization, and $\mathbf{z}'_l \in \mathbb{R}^{(N+1)\times D}$ is the output of the MSA block at layer l.

- **Multi-Layer Perceptron (MLP)**: Following the MSA, a position-wise feed-forward network (FFN) is applied to each position independently and identically. This FFN typically consists of two linear layers with a Gaussian Error Linear Unit (GELU) activation in between, unlike the Rectified Linear Unit (ReLU) used in the original Transformer:

  $$\mathrm{MLP}(x) = \mathrm{GELU}(xW_1 + b_1)W_2 + b_2 \tag{5}$$

  where W_1, W_2 are weight matrices and b_1, b_2 are bias vectors. Besides, dropout is applied after each fully connected layer to prevent overfitting. For layer $l = 1, ..., L$, the MLP block can be expressed as:

  $$\mathbf{z}_l = \mathrm{MLP}(LN(\mathbf{z}'_l)) + \mathbf{z}'_l \tag{6}$$

  where $\mathbf{z}_l \in \mathbb{R}^{(N+1)\times D}$ is the output of the MLP block at layer l.

- **Layer Normalization**: In contrast to the post-normalization used in the original Transformer, the encoder blocks in ViT implement pre-normalization, where layer normalization is applied right before the MSA and MLP blocks, which has been shown to lead to mre effective training for deeper networks (Xiong et al., 2020) without the need for learning rate warm-up.

5. **Classification Head**: For image classification, only the output vector corresponding

to the prepended class token in the final layer ($\mathbf{z}_L^0$) is used, which serves as the aggregate representation of the entire image:

$$\mathbf{v} = LN(\mathbf{z}_L^0) \tag{7}$$

This vector $\mathbf{v} \in \mathbb{R}^D$, which serves as an encoded vector representation of the input image, is then passed through a final classification head, typically a single linear layer followed by a softmax function, to produce the class probabilities:

$$\mathbf{y} = \text{softmax}(\mathbf{v}W_c + b_c) \tag{8}$$

where $W_c \in \mathbb{R}^{D \times K}$ and $b_c \in \mathbb{R}^K$ are the weights and bias of the classification head, and K is the number of classes.

4.2 Image Representation of Option-Implied Volatility Surface

In the computer vision (CV) field, a image is typically represented as a three-dimensional matrix with (C, H, W) dimensions, where C is the number of channels (e.g., $C = 3$ for RGB images, and $C = 1$ for grayscale images), H is the height (number of pixels in vertical direction), and W is the width (number of pixels in horizontal direction). Each pixel in the image has a value ranging from 0 to 255, representing the intensity of the color channel(s) at that pixel.

An option-implied volatility surface is a three-dimensional plot that displays the implied volatility of a security options across different strike prices and expirations (Figure 2). As shown in Figure 3, the volatility surface can also be represented as a two-dimensional matrix, which is anlogous to a single-channel grayscale image. The height H of the matrix corresponds to the number of maturities N_τ, and the width W corresponds to the number of moneyness levels N_δ. In our dataset (described in Section 3), the volatility surface contains 10 maturities and 18 moneyness levels, so we can reshape the volatility surface into a "image-like" matrix with dimensions $(C, H, W) = (1, 10, 18)$. The pixel values in the matrix are the corresponding implied volatility levels, which are standardized to have zero mean and unit variance before being fed into the ViT model, just like standard image preprocessing in the computer vision field.

Besides the implied volatility surface of individual stocks, we also consider the implied volatility surface of the S&P 500 index options as an additional input. The index options contain essential information of market-wide risk and risk premia, in time series as well as in cross-section, providing incremental predictive power for the stock returns (Andersen et al., 2015; Ang et al., 2006). The index volatility surface, in hence, can serve as an additional channel for the input volatility surface "image". However, unlike standard RGB images, where the three channels (Red, Green, Blue) contain correlated and complementary information about the same object, the two implied volatility surfaces in our case contain semantically distinct and independent information. This poses a challenge for the standard ViT model, which is designed to process images with correlated channels. Inspired by Bao et al. (2024), we use the so-called Channel ViT model for the embedding of the multiple volatility surfaces.

Specifically, the Channel ViT model, shown in Figure 5, processes each input channel (volatility surface) independently in the embedding stage. Each channel (volatility surface) is split into patches, linearly embedded, and employs a set of learnable channel embeddings to encode the channel-specific information besides the positional embeddings. The resulting sequence of vectors from all channels is then concatenated and fed into a standard Transformer encoder, similar to the original ViT model. Since the two volatility surfaces have the exactly the same structure (same maturities and moneyness levels), they share a common positional encoding matrix.

The ViT model, except for the final classification head, can be viewed as a non-linear mapping $ViT(\cdot|\theta)$ from the input image (volatility surface) to a dense vector representation of the image:

$$\mathbf{v}_{i,t} = \mathbf{ViT}(IV_t|\theta) \tag{9}$$

Where $IV_t = IV_{i,t}$ or $IV_t = (IV_{i,t}, IV_{spx,t})$ depending on whether we use single or multiple volatility surfaces as input. Here, $IV_{i,t}$ is the volatility surface of stock i at time t, and $IV_{spx,t}$ is the volatility surface of the S&P 500 index at time t. $\mathbf{v}_{i,t} \in \mathbb{R}^D$ is vector representation of the information contained in the volatility surfaces, and θ is the set of parameters of the ViT model, excluding the final classification head.

4.3 Discretization of Return

Utilizing the dense vector representation $\mathbf{v}_{i,t}$ obtained from the ViT model, a classification head is attached to perform classification tasks. In our study, such a classification head is a single linear layer followed by a softmax function [3], which maps the vector representation $\mathbf{v}_{i,t}$ to a probability distribution over the classes:

$$\mathbf{p}_{i,t} = \text{softmax}(W_c \cdot \mathbf{v}_{i,t} + b_c) \tag{10}$$

$$\text{softmax}(\mathbf{x})_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, \quad j = 1, ..., K \tag{11}$$

where $W_c \in \mathbb{R}^{K \times D}$ and $b_c \in \mathbb{R}^K$ are the weights and bias of the classification head, and K is the number of classes. The output $\mathbf{p}_{i,t} \in \mathbb{R}^K$ is a probability vector, where each element $\mathbf{p}_{i,t}(k)$ represents the predicted probability of stock i's return falling into class k at time t.

The simplest classification is the binary classification, where the number of classes $K = 2$. For example, we can classify the stock return into "up" and "down" classes based on whether the future return is positive or negative, as in Jiang et al. (2023). Alternatively, the stock

[3] In the original ViT paper, Dosovitskiy et al. (2020) used a multi-layer perceptron (MLP) with one hidden layer at pre-training time and used a single linear layer at fine-tuning time. A recent update, Beyer et al. (2022) from some of the same authors of the original paper suggests that a simple linear layer at the end is not significantly worse than an MLP. We use a single linear layer for simplicity reason.

return can be classified into "safe" and "crash" classes based on whether the future return is above a certain negative value (e.g., -10%). In general, we denote $y_{i,t}$ as the binary variable indicating whether stock i's return falls into the two classes defined by a specific threshold q at time t:

$$y_{i,t} = \mathbf{I}(r_{i,t}^{H} > q) = \begin{cases} 1, & \text{if } r_{i,t}^{H} > q \\ 0, & \text{if } r_{i,t}^{H} \leq q \end{cases} \tag{12}$$

$$\hat{y}_{i,t} = \mathbf{p}_{i,t}(1) \tag{13}$$

where $r_{i,t}^{H}$ is the cumulative return of stock i from day t to day $t + H$, and the output $\hat{y}_{i,t} = \mathbf{p}_{i,t}(1)$ represents the predicted probability of stock i's return being in class 1 at time t. The threshold q can be set to 0 for the "up/down" classification. [4]

For multi-class classification ($K > 2$), the stock returns can be discretized into multiple classes based on quantiles or specific thresholds. Assume $\{q_0, q_1, ..., q_K\}$ are the thresholds that define the K classes, where $q_0 = -\infty$ and $q_K = +\infty$. In general, we denote $y_{i,t}$ as the categorical variable indicating which class stock i's return falls into at time t:

$$y_{i,t} = k, \quad \text{if } r_{i,t}^{H} \in (q_{k-1}, q_k], \quad k = 1, 2, ..., K \tag{14}$$

$$\hat{y}_{i,t}^{k} = \mathbf{p}_{i,t}(k) \tag{15}$$

[4] In the special case of binary classification ($K = 2$), the softmax function reduces to the logistic (sigmoid) function.

4.4 Training Process

We use standard cross-entropy loss function for the classification task. For binary classification, where $y \in \{0, 1\}$, the loss function can be expressed as:

$$\mathcal{L}(y, \hat{y}) = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right] \tag{16}$$

$$= \begin{cases} -\log(\hat{y}), & \text{if } y = 1 \\ -\log(1 - \hat{y}), & \text{if } y = 0 \end{cases} \tag{17}$$

For multi-class classification, where $y \in \{1, 2, ..., K\}$, the loss function can be expressed as:

$$\mathcal{L}(y, \hat{y}) = -\sum_{k=1}^{K} \mathbf{I}(y = k) \log(\hat{y}^k) \tag{18}$$

$$= -\log(\hat{y}^k), \quad \text{if } y = k \tag{19}$$

In the case of imbalanced classes, we can use a weighted cross-entropy loss function to give more importance to the minority classes. The weighted loss function can be expressed as:

$$\mathcal{L}(y, \hat{y}) = -\sum_{k=1}^{K} w_k \cdot \mathbf{I}(y = k) \log(\hat{y}^k) \tag{20}$$

$$= -w_k \cdot \log(\hat{y}^k), \quad \text{if } y = k \tag{21}$$

$$w_k = \frac{N}{K \cdot N_k} \tag{22}$$

where w_k is the weight for class k, N is the total number of samples, and N_k is the number of samples in class k. This weighting scheme helps to balance the influence of each class during training, especially when some classes are underrepresented.

We adopt similar regularization techniques as in Gu et al. (2020), to prevent overfitting and improve the comuputational efficiency. We use a Adaptive Moment Estimation With Weight Decay (AdamW) optimizer (Loshchilov and Hutter, 2019), which is standard in the transformer training and a refined version of the Adam optimizer (Kingma and Ba, 2017). AdamW decouples the weight decay (L2 regularization) from the gradient update, which has been shown to lead to better generalization performance. In this study, we set the learning rate to 1×10^{-5}, the weight decay to 1×10^{-2} and the batch size to 512. We use a dropout rate of 0.2 to prevent overfitting. Dropout is a regularization technique that randomly sets a fraction of the input units to zero during training, which helps prevent overfitting by reducing the model's reliance on specific features. We apply multiple methods for parameter initialization: for weights in the patch embedding layers and classfication head, we use the Xavier initialization (Glorot and Bengio, 2010), which is the best practice for deep neural networks such as CNNs; for parameters in the CLS token and positional embeddings, as well as the weights in the Transformer encoder, we use a truncated normal distribution with a standard deviation of 0.02, following the BERT model Devlin et al. (2019); for layer normalization parameters, we initialize the weights to 1; the biases in all layers are initialized to 0. We use a standard "warm-up" and "cosine decay" learning rate scheduler, which gradually increases the learning rate from zero to the initial value over a specified number of warm-up steps, and then decays it using a cosine function. This approach helps stabilize training in the early stages and allows for better convergence. Early stopping is also employed, where the training process is halted if the validation loss does not improve for a two consecutive epochs, to prevent overfitting and save computational resources.

We train each model using annually updated rolling window, given the time-varying universe of stocks in 1. Specifically, we use a 8-year in-sample period, with 6 years for training and 2 years for validation, to predict the out-of-sample returns for the subsequent year. The first training period starts from January 1996 to December 2003, and the trained model is then used to predict out-of-sample returns for the subsequent year 2004. The model is then updated with the data from 2004, and used to predict out-of-sample returns for 2005.

This process continues until August 2023, the last month of our dataset. Overall, the out-of-sample prediction period is from January 2004 to August 2023, which contains around 20 years of results. The daily dataset provides a large number of training samples, which is crucial for training such a large deep learning model, while we only collect the out-of-sample predictions at the end of each month for empirical analysis.

5 Empirical Results

We begin with the simplest binary classification problem ($K = 2$). Following Jiang et al. (2023), the stock returns are simply classified into ”up” and ”down” classes based on whether the future return is positive or negative. Besides the simplicity and interpretability, one remarkable benefit of such binary classification is that the two classes are naturally more balanced in our dataset, which is to say, there are approximately 50% ”up” labels and 50% ”down” labels. This is crucial for the training of such deep learning or machine learning models, since imbalanced classes can lead to suboptimal performance, as the model may become biased towards the majority class (He and Garcia, 2009).

We focus on the prediction of 1-month ahead stock returns ($H = 21$ trading days), which is most common in the literature of cross-section of stock returns studies. Table 1 reports the portfolio performance of equal-weight and value-weight decile portfolios formed on out-of-sample predicted ”up” probability $\hat{P}(r_{i,t}^{21} > 0)$.

6 Conclusion

References

Andersen, T. G., N. Fusari, and V. Todorov, 2015, "The Risk Premia Embedded in Index Options," *Journal of Financial Economics*, 117 (3), 558–584, September.

Ang, A., R. J. Hodrick, Y. Xing, and X. Zhang, 2006, "The Cross-Section of Volatility and Expected Returns," *The Journal of Finance*, 61 (1), 259–299.

Bao, Y., S. Sivanandan, and T. Karaletsos, 2024, "Channel Vision Transformers: An Image Is Worth 1 x 16 x 16 Words," April.

Beyer, L., X. Zhai, and A. Kolesnikov, 2022, "Better Plain ViT Baselines for ImageNet-1k," May.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova, 2019, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in J. Burstein, C. Doran, and T. Solorio eds. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics, June.

Dosovitskiy, A., L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, and S. Gelly, 2020, "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv preprint arXiv:2010.11929*.

Glorot, X. and Y. Bengio, 2010, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256. JMLR Workshop and Conference Proceedings, March.

Gu, S., B. Kelly, and D. Xiu, 2020, "Empirical Asset Pricing via Machine Learning.," *Review of Financial Studies*, 33 (5), 2223–2273.

He, H. and E. A. Garcia, 2009, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, 21 (9), 1263–1284, September.

Jiang, J., B. Kelly, and D. Xiu, 2023, "(Re-)Imag(in)Ing Price Trends," *The Journal of Finance*, 78 (6), 3193–3249.

Kingma, D. P. and J. Ba, 2017, "Adam: A Method for Stochastic Optimization," January.

Loshchilov, I. and F. Hutter, 2019, "Decoupled Weight Decay Regularization," January.

Martin, I., 2017, "What Is the Expected Return on the Market?*," *The Quarterly Journal of Economics*, 132 (1), 367–433, February.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, 2017, "Attention Is All You Need," *Advances in neural information processing systems*, 30.

Xiong, R., Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and

T.-Y. Liu, 2020, “On Layer Normalization in the Transformer Architecture,” June.

Yan, S., 2011, “Jump Risk, Stock Returns, and Slope of Implied Volatility Smile,” *Journal of Financial Economics*, 99 (1), 216–233, January.

Tables

Table 1: Portfolio Analysis, Binary Classification Model, All Stocks

	Portfolio Sorted on $\hat{P}(r^{21}_{i,t} > 0)$										
	Low	2	3	4	5	6	7	8	9	High	H-L
Panel A: Equal-Weighted Portfolios											
Exc Return	−0.43	−0.02	0.38	0.57	0.81	0.86	0.92	0.91	0.70	0.59	1.02
	(−0.90)	(−0.04)	(0.95)	(1.46)	(2.17)	(2.35)	(2.57)	(2.63)	(2.44)	(2.22)	(2.80)
Sharpe Ratio	−0.20	−0.01	0.21	0.31	0.45	0.49	0.56	0.57	0.50	0.42	0.66
CAPM α	−1.38	−0.96	−0.54	−0.35	−0.09	−0.01	0.10	0.13	−0.01	−0.07	1.31
	(−4.09)	(−3.58)	(−2.32)	(−1.53)	(−0.42)	(−0.06)	(0.47)	(0.66)	(−0.04)	(−0.48)	(3.64)
FF3 α	−1.22	−0.83	−0.42	−0.24	0.02	0.09	0.20	0.20	0.05	−0.03	1.20
	(−4.99)	(−4.13)	(−2.24)	(−1.20)	(0.08)	(0.48)	(0.98)	(1.05)	(0.39)	(−0.17)	(4.10)
FFC α	−1.20	−0.81	−0.38	−0.19	0.09	0.16	0.28	0.27	0.07	0.00	1.20
	(−4.93)	(−4.06)	(−2.14)	(−1.01)	(0.48)	(0.99)	(1.53)	(1.52)	(0.56)	(−0.03)	(3.99)
FF5 α	−0.92	−0.62	−0.28	−0.15	0.05	0.11	0.19	0.18	0.04	−0.01	0.91
	(−4.36)	(−3.51)	(−1.60)	(−0.79)	(0.27)	(0.55)	(0.96)	(0.93)	(0.30)	(−0.04)	(3.68)
FF6 α	−0.90	−0.60	−0.26	−0.11	0.11	0.16	0.25	0.24	0.05	0.01	0.91
	(−4.29)	(−3.47)	(−1.50)	(−0.62)	(0.61)	(0.97)	(1.43)	(1.34)	(0.42)	(0.06)	(3.56)
Q5 α	−0.86	−0.50	−0.14	0.02	0.25	0.28	0.38	0.35	0.15	0.12	0.98
	(−3.06)	(−2.16)	(−0.70)	(0.09)	(1.15)	(1.43)	(1.84)	(1.66)	(1.14)	(0.77)	(2.95)
Panel B: Value-Weighted Portfolios											
Exc Return	−0.39	0.02	0.70	0.81	0.96	0.69	0.85	0.71	0.67	0.64	1.03
	(−0.79)	(0.05)	(1.63)	(2.03)	(2.49)	(1.92)	(2.57)	(2.34)	(2.54)	(2.41)	(2.61)
Sharpe Ratio	−0.17	0.01	0.35	0.42	0.51	0.39	0.51	0.44	0.48	0.46	0.55
CAPM α	−1.36	−0.91	−0.24	−0.14	0.02	−0.19	0.01	−0.08	−0.03	0.00	1.36
	(−4.35)	(−3.19)	(−0.92)	(−0.52)	(0.09)	(−0.88)	(0.03)	(−0.50)	(−0.21)	(−0.01)	(3.63)
FF3 α	−1.24	−0.82	−0.19	−0.08	0.05	−0.14	0.04	−0.07	−0.03	0.01	1.25
	(−4.57)	(−3.30)	(−0.86)	(−0.35)	(0.21)	(−0.68)	(0.21)	(−0.39)	(−0.21)	(0.07)	(3.74)
FFC α	−1.24	−0.82	−0.18	−0.06	0.10	−0.08	0.11	−0.02	−0.01	0.02	1.26
	(−4.44)	(−3.20)	(−0.85)	(−0.27)	(0.40)	(−0.40)	(0.57)	(−0.09)	(−0.07)	(0.15)	(3.63)
FF5 α	−0.92	−0.57	−0.07	0.06	0.19	−0.15	0.08	−0.05	−0.07	0.00	0.92
	(−3.79)	(−2.59)	(−0.34)	(0.29)	(0.77)	(−0.79)	(0.42)	(−0.26)	(−0.54)	(0.02)	(2.96)
FF6 α	−0.91	−0.56	−0.07	0.08	0.22	−0.10	0.13	0.00	−0.06	0.01	0.92
	(−3.67)	(−2.51)	(−0.33)	(0.35)	(0.93)	(−0.56)	(0.73)	(−0.02)	(−0.46)	(0.07)	(2.88)
Q5 α	−0.89	−0.54	0.05	0.11	0.23	−0.06	0.20	−0.03	−0.01	0.10	1.00
	(−2.60)	(−1.75)	(0.22)	(0.44)	(0.89)	(−0.28)	(0.94)	(−0.14)	(−0.08)	(0.62)	(2.31)

Table 2: Correlation Analysis, Binary Classification Model, All Stocks

	$\hat{P}(r_{i,t}^{21} > 0)$	CIV	PIV	ΔCIV	ΔPIV	log(O/S)	IVS_{ATM}	IVS_{OTM}	beta	log(ME)	$ret_{12,2}$	$ret_{1,0}$
$\hat{P}(r_{i,t}^{21} > 0)$	1.00	-0.36	-0.37	-0.02	-0.02	0.02	-0.04	-0.09	-0.15	0.34	0.03	0.05
CIV	-0.36	1.00	0.84	0.20	0.12	-0.22	-0.09	0.08	0.12	-0.43	-0.18	-0.11
PIV	-0.37	0.84	1.00	0.12	0.20	-0.19	0.17	0.11	0.12	-0.43	-0.17	-0.10
ΔCIV	-0.02	0.20	0.12	1.00	0.47	-0.01	-0.32	-0.05	-0.01	-0.01	0.00	-0.13
ΔPIV	-0.02	0.12	0.20	0.47	1.00	-0.01	0.32	0.02	-0.00	-0.00	-0.00	-0.08
log(O/S)	0.02	-0.22	-0.19	-0.01	-0.01	1.00	0.09	-0.23	0.10	0.37	0.13	0.04
IVS_{ATM}	-0.04	-0.09	0.17	-0.32	0.32	0.09	1.00	0.21	0.01	-0.02	-0.00	0.04
IVS_{OTM}	-0.09	0.08	0.11	-0.05	0.02	-0.23	0.21	1.00	-0.01	-0.24	-0.03	0.03
beta	-0.15	0.12	0.12	-0.01	-0.00	0.10	0.01	-0.01	1.00	-0.01	-0.03	-0.00
log(ME)	0.34	-0.43	-0.43	-0.01	-0.00	0.37	-0.02	-0.24	-0.01	1.00	0.18	0.08
$ret_{12,2}$	0.03	-0.18	-0.17	0.00	-0.00	0.13	-0.00	-0.03	-0.03	0.18	1.00	0.02
$ret_{1,0}$	0.05	-0.11	-0.10	-0.13	-0.08	0.04	0.04	0.03	-0.00	0.08	0.02	1.00

Table 3: Portfolio Characteristics, Binary Classification Model, All Stocks

	Low	2	3	4	5	6	7	8	9	High
$\hat{P}(r_{i,t}^{21} > 0)$	0.437	0.503	0.542	0.563	0.577	0.589	0.599	0.608	0.618	0.644
CIV	1.262	0.819	0.690	0.609	0.521	0.450	0.407	0.370	0.381	0.431
PIV	1.160	0.763	0.650	0.579	0.499	0.430	0.388	0.354	0.359	0.413
ΔCIV	0.013	0.002	-0.001	-0.001	-0.001	-0.003	-0.003	-0.002	-0.001	-0.001
ΔPIV	0.011	0.003	-0.001	0.000	-0.000	-0.002	-0.003	-0.003	-0.002	-0.001
log(O/S)	5.383	5.531	5.689	5.726	5.711	5.638	5.635	5.662	5.628	5.379
IVS_{ATM}	0.023	0.007	0.004	0.004	0.005	0.001	-0.001	-0.000	0.001	0.002
IVS_{OTM}	0.112	0.087	0.075	0.072	0.071	0.069	0.067	0.066	0.068	0.090
beta	1.186	1.248	1.262	1.226	1.170	1.110	1.042	0.967	0.894	0.854
log(ME)	12.545	13.134	13.507	13.806	14.111	14.388	14.659	14.862	14.871	14.343
$ret_{12,2}$	0.128	0.193	0.199	0.166	0.149	0.132	0.130	0.125	0.113	0.107
$ret_{1,0}$	-0.460	0.124	0.581	0.837	1.029	1.105	1.191	1.162	1.036	0.857

Table 4: Time Series Regression, Binary Classification Model, All Stocks

	CAPM	FF3	FFC	FF5	FF6	FF6 + Controls
	(1)	(2)	(3)	(4)	(5)	(6)
Alpha	0.013***	0.012***	0.012***	0.009***	0.009***	0.012***
	(0.004)	(0.003)	(0.003)	(0.002)	(0.003)	(0.004)
MKT	-0.396***	-0.213**	-0.213**	-0.233***	-0.223***	-0.177**
	(0.118)	(0.104)	(0.090)	(0.087)	(0.082)	(0.077)
SMB		-0.979***	-0.979***	-0.733***	-0.731***	-0.778***
		(0.177)	(0.179)	(0.130)	(0.130)	(0.145)
HML		0.064	0.064	0.355***	0.375***	0.288**
		(0.097)	(0.114)	(0.126)	(0.128)	(0.114)
RMW				0.829***	0.829***	0.865***
				(0.150)	(0.151)	(0.138)
CMA				-0.522*	-0.540**	-0.566**
				(0.275)	(0.255)	(0.247)
UMD			0.001		0.037	0.079
			(0.128)		(0.119)	(0.105)
ΔCIV						-0.002
						(0.002)
ΔPIV						-0.003
						(0.002)
log(O/S)						0.004**
						(0.002)
IVS_{ATM}						-0.004**
						(0.002)
IVS_{OTM}						0.002
						(0.001)
Observations	236	236	236	236	236	228
R^2	0.108	0.282	0.282	0.362	0.363	0.392
Adjusted R^2	0.104	0.273	0.269	0.348	0.346	0.361
Residual Std. Error	0.051	0.046	0.046	0.043	0.043	0.042
F Statistic	11.238***	12.450***	9.335***	13.899***	11.599***	12.037***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 5: Fama-MacBeth Regression, Binary Classification Model, All Stocks

	(1)	(2)	(3)
$\hat{P}(r_{i,t}^{21} > 0)$	0.060***	0.051**	0.053**
	(0.020)	(0.026)	(0.024)
CIV		-0.017**	-0.016*
		(0.008)	(0.008)
PIV		0.014	0.013
		(0.010)	(0.010)
ΔCIV		-0.017***	-0.015***
		(0.006)	(0.005)
ΔPIV		0.013*	0.013**
		(0.007)	(0.006)
log(O/S)		-0.000	-0.000
		(0.000)	(0.000)
IVS_{ATM}		-0.034***	-0.034***
		(0.010)	(0.009)
IVS_{OTM}		-0.008	-0.005
		(0.005)	(0.005)
beta			0.001
			(0.003)
$ret_{12,2}$			0.001
			(0.003)
$ret_{1,0}$			0.000**
			(0.000)
Observations	904497	354994	346324

Note: *p<0.1; **p<0.05; ***p<0.01

Table 6: Logistic Regression, Future Stock Return

	Dependent variable: $\mathbf{I}(r_{i,t}^{21} > 0)$		
	(1)	(2)	(3)
$\hat{P}(r_{i,t}^{21} > 0)$	0.654***		0.462***
	(0.015)		(0.023)
CIV		-0.160***	-0.174***
		(0.052)	(0.052)
PIV		-0.122**	-0.171***
		(0.056)	(0.057)
ΔCIV		0.207***	0.229***
		(0.047)	(0.047)
ΔPIV		-0.009	-0.001
		(0.046)	(0.046)
log(O/S)		-0.033***	-0.031***
		(0.003)	(0.003)
IVS_{ATM}		-0.080	-0.045
		(0.066)	(0.066)
IVS_{OTM}		-0.034	-0.033
		(0.042)	(0.043)
beta		-0.017**	-0.005
		(0.008)	(0.008)
log(ME)		0.037***	0.028***
		(0.003)	(0.003)
$ret_{12,2}$		-0.025***	-0.017***
		(0.005)	(0.005)
$ret_{1,0}$		-0.001**	-0.001**
		(0.000)	(0.000)
Observations	904497	346324	346324
Pseudo R^2	0.002	0.002	0.003

Note: *p<0.1; **p<0.05; ***p<0.01

Table 7: Logistic Regression, Future Stock Crash

	$\mathbf{I}(r^{H}_{i,t} < -0.1)$			$\mathbf{I}(r^{H}_{i,t} < -0.2)$			$\mathbf{I}(r^{H}_{i,t} < -0.3)$		
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
$\hat{P}(r^{21}_{i,t} > 0)$	-1.158***		-0.748***	-1.566***		-1.239***	-1.596***		-1.426***
	(0.028)		(0.036)	(0.051)		(0.066)	(0.079)		(0.113)
CIV		0.460***	0.478***		0.503***	0.548***		0.707***	0.767***
		(0.111)	(0.115)		(0.125)	(0.133)		(0.109)	(0.113)
PIV		1.436***	1.510***		1.859***	1.943***		1.498***	1.558***
		(0.122)	(0.127)		(0.146)	(0.155)		(0.136)	(0.139)
ΔCIV		-0.096	-0.138**		0.125	0.055		0.321**	0.243
		(0.061)	(0.061)		(0.096)	(0.097)		(0.152)	(0.154)
ΔPIV		0.108*	0.104*		0.287***	0.284***		0.446***	0.441***
		(0.062)	(0.062)		(0.096)	(0.097)		(0.148)	(0.151)
log(O/S)		0.110***	0.105***		0.173***	0.162***		0.223***	0.211***
		(0.004)	(0.004)		(0.008)	(0.008)		(0.014)	(0.014)
IVS_{ATM}		-0.099	-0.163		0.080	0.007		0.289*	0.239
		(0.106)	(0.108)		(0.134)	(0.139)		(0.173)	(0.178)
IVS_{OTM}		-0.310***	-0.315***		-0.286***	-0.306***		0.163	0.140
		(0.061)	(0.062)		(0.101)	(0.103)		(0.150)	(0.154)
beta		0.230***	0.213***		0.315***	0.293***		0.354***	0.339***
		(0.012)	(0.012)		(0.020)	(0.020)		(0.033)	(0.034)
log(ME)		-0.135***	-0.120***		-0.208***	-0.180***		-0.246***	-0.212***
		(0.005)	(0.005)		(0.008)	(0.009)		(0.014)	(0.015)
$ret_{12,2}$		-0.004	-0.014***		-0.012	-0.026**		-0.052***	-0.068***
		(0.005)	(0.005)		(0.009)	(0.011)		(0.014)	(0.016)
$ret_{1,0}$		-0.004***	-0.004***		-0.007***	-0.007***		-0.011***	-0.011***
		(0.000)	(0.000)		(0.001)	(0.001)		(0.001)	(0.001)
Observations	904497	346324	346324	904497	346324	346324	904497	346324	346324
Pseudo R^2	0.004	0.060	0.061	0.005	0.115	0.118	0.005	0.140	0.144

Note: *p<0.1; **p<0.05; ***p<0.01

Figures

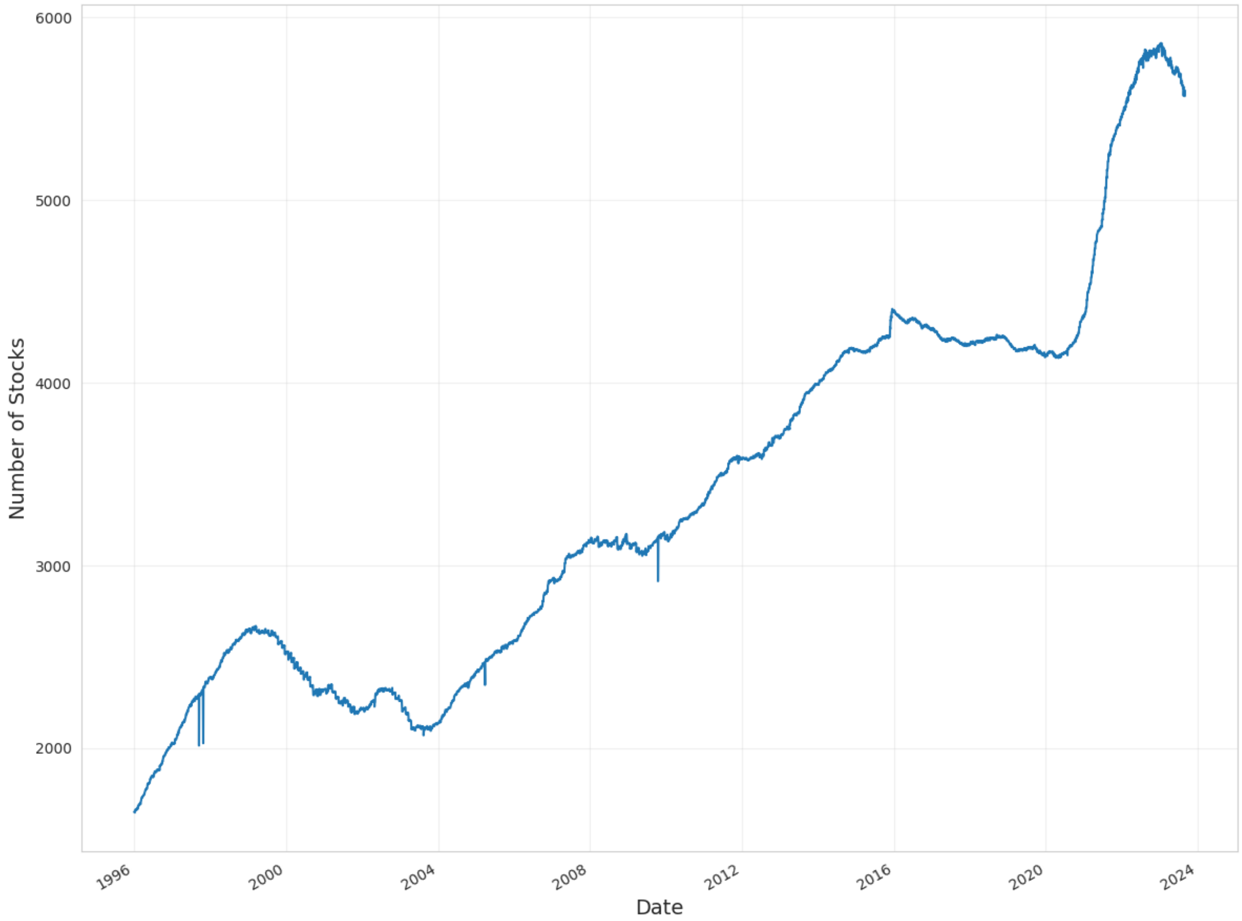

Figure 1: Number of Stocks with Option-Implied Volatility Surface Data (1996-2023). The figure shows the number of stocks with available option-implied volatility surface data from IvDB OptionMetrics, spanning from January 1996 to August 2023.

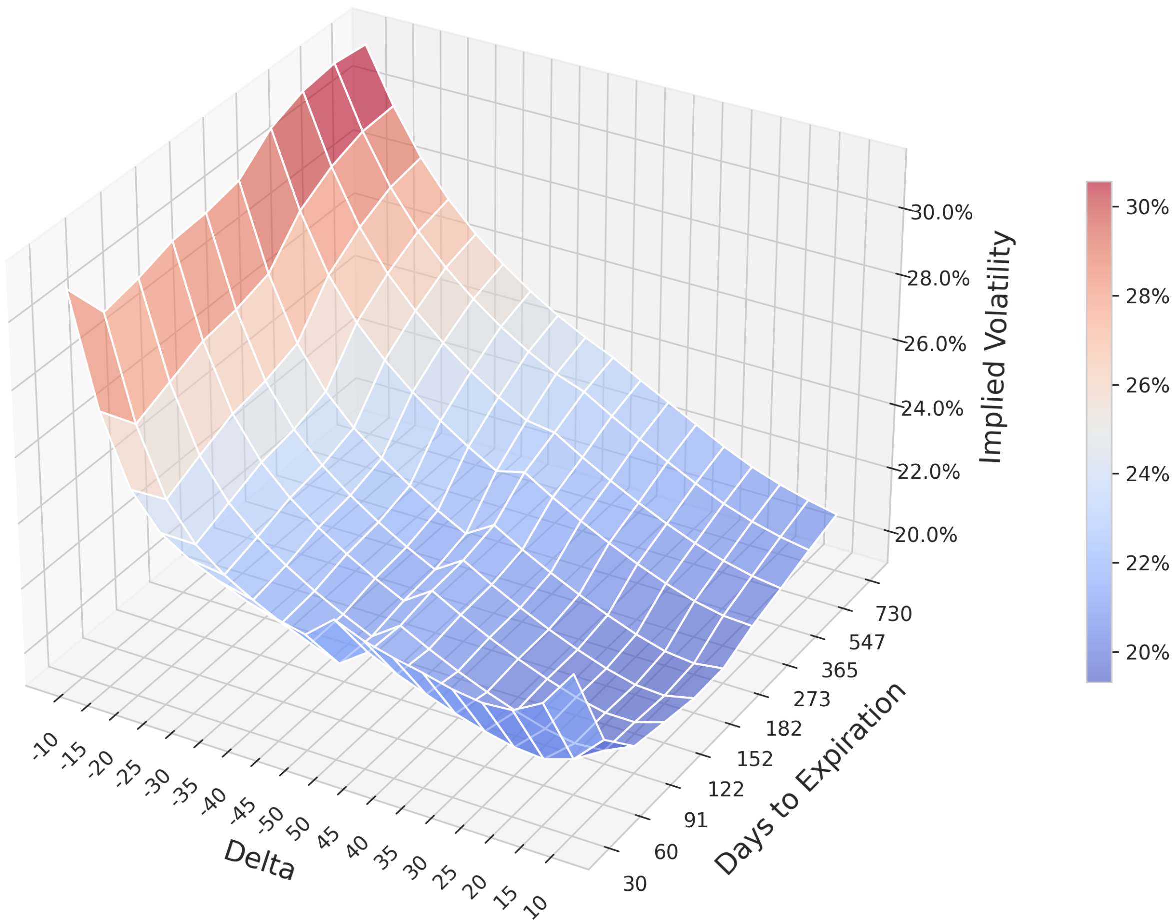

Figure 2: Example of Interpolated Option-Implied Volatility Surface (AAPL, 01/08/2023). The plot shows the volatility surface for Apple Inc. (AAPL) on Aug 1st, 2023, with moneyness (option delta) on the x-axis, days to expiration on the y-axis, and implied volatility levels represented by the color gradient.

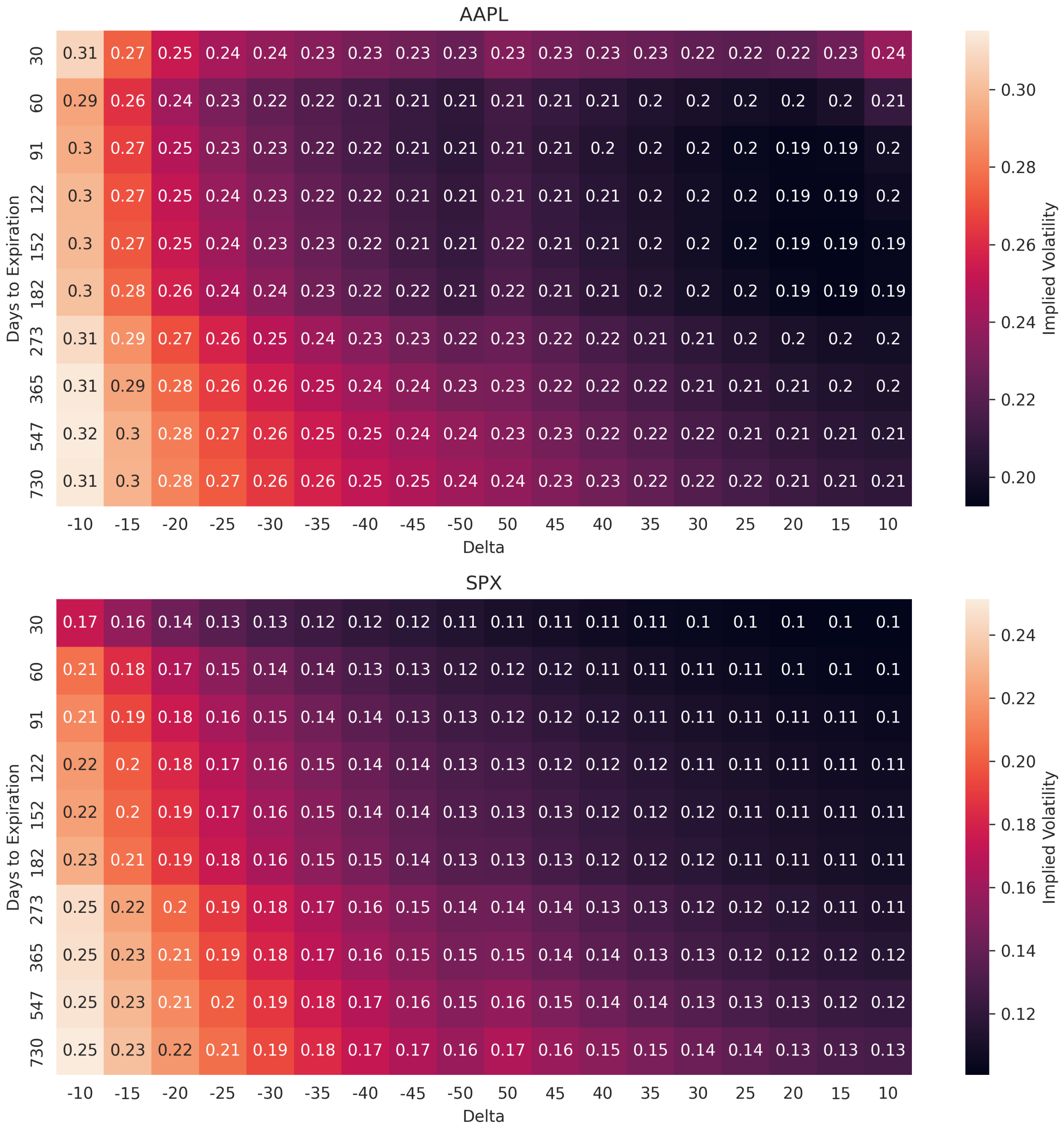

Figure 3: ”Image” Representation of Volatility Surface (01/08/2023). The upper panel shows the heatmap of the volatility surface for Apple Inc. (AAPL), while the lower panel shows the volatility surface for the S&P 500 index (SPX) on the same date. The x-axis represents moneyness (option delta), and the y-axis represents days to expiration, with color intensity indicating implied volatility levels.

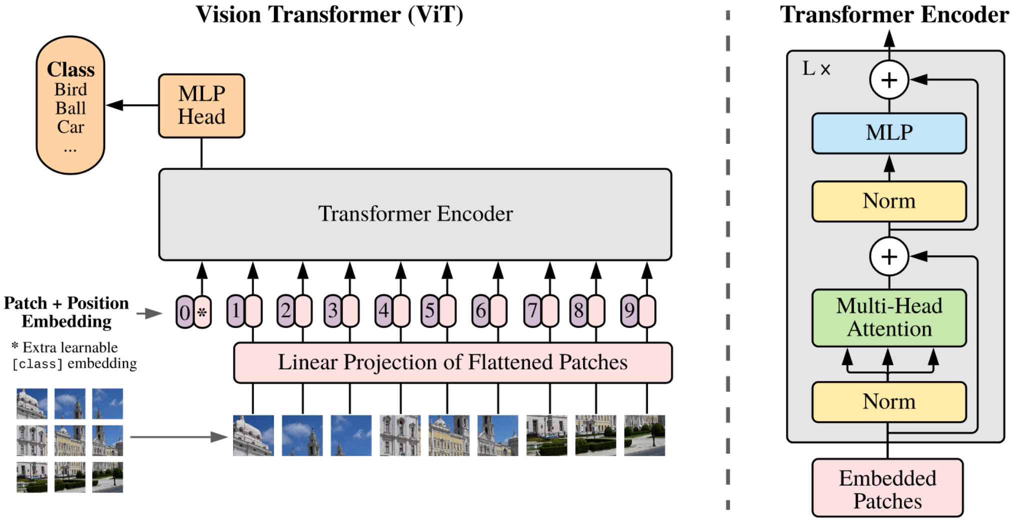

Figure 4: Vision Transformer (ViT) Architecture from Dosovitskiy et al. (2020). The image is split into fixed-size patches, linearly embedded, and then position embeddings are added. The resulting sequence of vectors is fed to a standard Transformer encoder. Similar to BERT, a classification token is prepended to the sequence, and the final hidden state corresponding to this token is used for classification tasks.

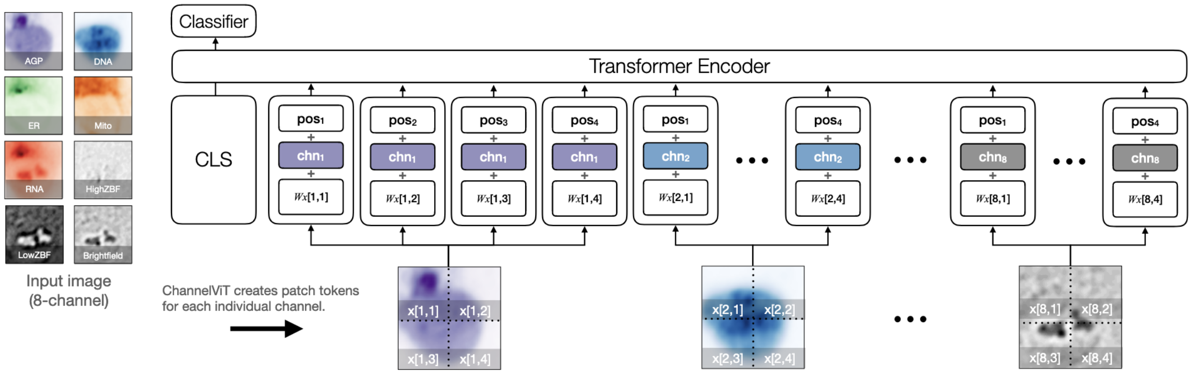

Figure 5: Channel Vision Transformer (ChannelViT) Architecture from Bao et al. (2024). The input image comprises multiple channels potentially carrying semantically distinct and independent information. ChannelViT constructs patch tokens for each individual channel, utilizing a learnable channel embedding **chn** to encode channel-specific information. The linear embedding **W** and positional embeddings **pos** are shared across all channels. The resulting sequence of vectors from all channels is concatenated and fed into a standard Transformer encoder. Similar to BERT, a classification token is prepended to the sequence, and the final hidden state corresponding to this token is used for classification tasks.

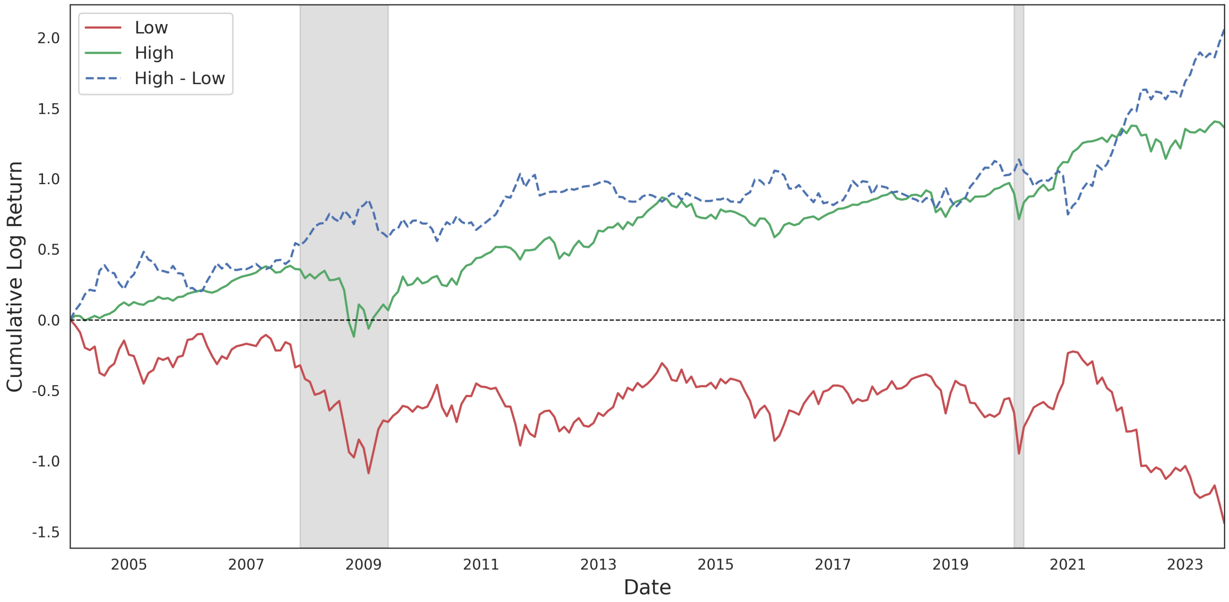

Figure 6: Cumulative Log Return, All Stocks. This figure shows the cumulative log return of equal-weight portfolios formed on out-of-sample predicted ”up” probability $\hat{P}(r_{i,t}^{21} > 0)$ from January 2004 to August 2023. The sample includes all stocks with available option-implied volatility surface data. The ”Low” series represents the decile portfolio of stocks with the lowest predicted ”up” probabilities, while the ”High” series represents the decile portfolio with the highest predicted ”up” probabilities. The ”High - Low” series represents a long position in the ”High” portfolio and a short position in the ”Low” portfolio.

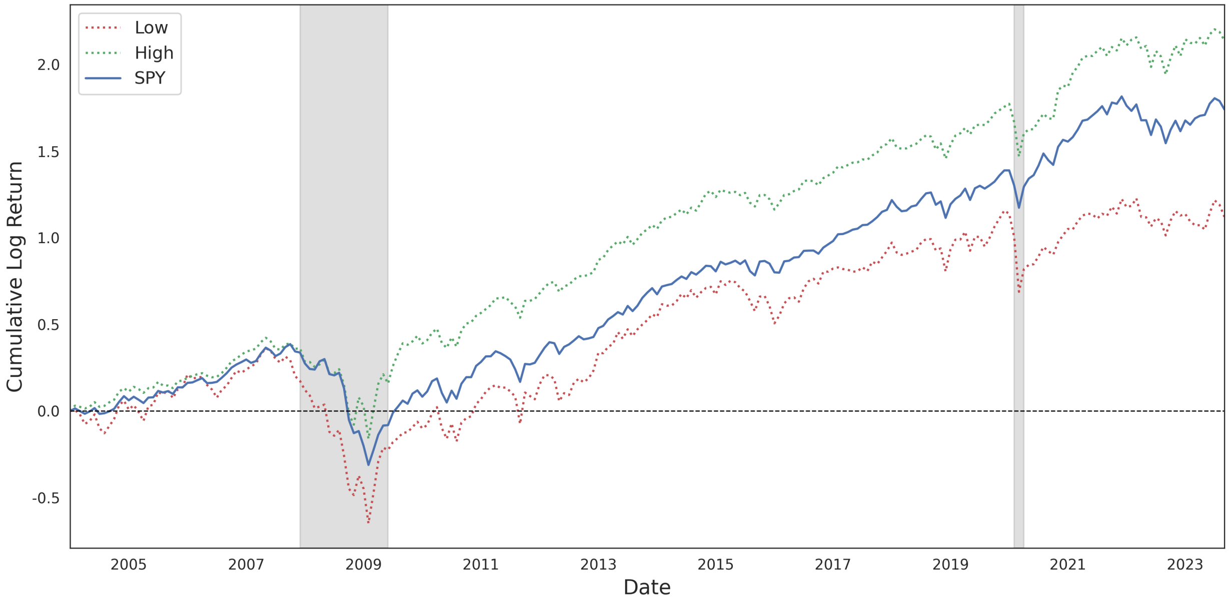

Figure 7: Cumulative Log Return, S&P 500 Stocks. This figure shows the cumulative log return of equal-weight decile portfolios formed on out-of-sample predicted ”up” probability $\hat{P}(r_{i,t}^{21} > 0)$ from January 2004 to August 2023. The sample includes only stocks that are part of the S&P 500 index. The ”Low” series represents the 1/3 stocks with the lowest predicted ”up” probabilities, while the ”High” series represents the 2/3 stocks with the highest predicted ”up” probabilities. The ”SPY” series represents the cumulative log return of the S&P 500 index ETF (SPY) over the same period.

Appendix A. Placeholder

Appendix B. Additional Results

Table 8: Portfolio Analysis, Binary Classification Model, Non-Micro Stocks

	Portfolio Sorted on $\hat{P}(r^{21}_{i,t} > 0)$										
	Low	2	3	4	5	6	7	8	9	High	H-L
Panel A: Equal-Weighted Portfolios											
Exc Return	0.04	0.47	0.61	0.78	0.82	0.82	0.91	0.97	0.78	0.70	0.65
	(0.10)	(1.24)	(1.81)	(2.39)	(2.43)	(2.52)	(2.72)	(3.46)	(2.96)	(2.80)	(2.07)
Sharpe Ratio	0.02	0.26	0.35	0.47	0.50	0.50	0.57	0.67	0.56	0.52	0.43
CAPM α	−0.89	−0.45	−0.27	−0.08	−0.02	−0.01	0.12	0.25	0.09	0.06	0.95
	(−3.11)	(−2.03)	(−1.32)	(−0.42)	(−0.08)	(−0.06)	(0.59)	(1.38)	(0.61)	(0.45)	(2.99)
FF3 α	−0.78	−0.36	−0.18	0.01	0.07	0.08	0.20	0.30	0.13	0.09	0.87
	(−3.36)	(−1.91)	(−1.01)	(0.06)	(0.38)	(0.45)	(1.05)	(1.79)	(0.95)	(0.75)	(3.29)
FFC α	−0.77	−0.36	−0.18	0.03	0.11	0.14	0.26	0.33	0.14	0.11	0.88
	(−3.26)	(−1.85)	(−0.97)	(0.16)	(0.63)	(0.82)	(1.55)	(2.07)	(1.04)	(0.85)	(3.19)
FF5 α	−0.53	−0.23	−0.16	−0.02	0.04	0.06	0.18	0.26	0.06	0.05	0.58
	(−2.59)	(−1.39)	(−0.94)	(−0.14)	(0.21)	(0.33)	(0.98)	(1.53)	(0.47)	(0.38)	(2.40)
FF6 α	−0.52	−0.23	−0.16	−0.01	0.07	0.10	0.24	0.28	0.07	0.06	0.58
	(−2.50)	(−1.34)	(−0.90)	(−0.06)	(0.40)	(0.63)	(1.40)	(1.79)	(0.55)	(0.47)	(2.33)
Q5 α	−0.49	−0.18	−0.06	0.12	0.20	0.23	0.32	0.39	0.22	0.20	0.68
	(−1.62)	(−0.83)	(−0.32)	(0.65)	(1.02)	(1.27)	(1.68)	(2.20)	(1.50)	(1.33)	(1.91)
Panel B: Value-Weighted Portfolios											
Exc Return	−0.09	0.81	0.88	0.73	0.89	0.74	0.74	0.90	0.58	0.66	0.74
	(−0.19)	(1.99)	(2.29)	(2.15)	(2.42)	(2.32)	(2.27)	(3.38)	(2.34)	(2.45)	(2.09)
Sharpe Ratio	−0.04	0.42	0.46	0.43	0.52	0.45	0.46	0.61	0.42	0.49	0.42
CAPM α	−1.03	−0.10	−0.06	−0.13	0.03	−0.09	−0.06	0.17	−0.11	0.04	1.07
	(−3.80)	(−0.42)	(−0.28)	(−0.57)	(0.15)	(−0.46)	(−0.30)	(0.88)	(−0.84)	(0.26)	(3.11)
FF3 α	−0.95	−0.07	−0.04	−0.09	0.07	−0.06	−0.03	0.16	−0.11	0.04	0.98
	(−3.87)	(−0.32)	(−0.19)	(−0.41)	(0.31)	(−0.30)	(−0.15)	(0.80)	(−0.85)	(0.28)	(3.21)
FFC α	−0.95	−0.07	−0.04	−0.06	0.12	0.00	0.04	0.19	−0.10	0.05	1.00
	(−3.79)	(−0.34)	(−0.21)	(−0.31)	(0.59)	(−0.02)	(0.21)	(0.99)	(−0.79)	(0.35)	(3.15)
FF5 α	−0.69	0.06	0.15	−0.03	0.11	−0.04	0.01	0.14	−0.16	−0.02	0.67
	(−3.12)	(0.30)	(0.70)	(−0.16)	(0.49)	(−0.20)	(0.06)	(0.73)	(−1.32)	(−0.11)	(2.35)
FF6 α	−0.69	0.05	0.14	−0.01	0.15	0.01	0.07	0.16	−0.16	−0.01	0.68
	(−3.05)	(0.25)	(0.65)	(−0.08)	(0.74)	(0.03)	(0.37)	(0.89)	(−1.29)	(−0.05)	(2.32)
Q5 α	−0.64	0.10	0.15	0.03	0.21	0.07	0.11	0.15	−0.06	0.11	0.74
	(−2.15)	(0.40)	(0.62)	(0.15)	(0.95)	(0.32)	(0.48)	(0.66)	(−0.41)	(0.62)	(1.89)

Table 9: Portfolio Analysis, Binary Classification Model, S&P 500 Stocks

	Portfolio Sorted on $\hat{P}(r^{21}_{i,t} > 0)$										
	Low	2	3	4	5	6	7	8	9	High	H-L
Panel A: Equal-Weighted Portfolios											
Exc Return	0.28	0.70	0.64	0.78	0.89	1.01	1.01	0.95	0.90	0.84	0.56
	(0.63)	(1.92)	(1.90)	(2.34)	(2.42)	(3.06)	(3.87)	(3.55)	(3.39)	(2.73)	(1.67)
Sharpe Ratio	0.13	0.39	0.39	0.50	0.53	0.62	0.73	0.68	0.63	0.58	0.32
CAPM α	−0.70	−0.18	−0.19	−0.02	0.06	0.20	0.29	0.25	0.25	0.22	0.92
	(−2.37)	(−0.83)	(−0.94)	(−0.10)	(0.33)	(1.17)	(1.74)	(1.66)	(1.55)	(1.25)	(2.75)
FF3 α	−0.59	−0.12	−0.13	0.03	0.11	0.26	0.32	0.27	0.27	0.25	0.84
	(−2.17)	(−0.59)	(−0.68)	(0.16)	(0.56)	(1.51)	(1.99)	(1.82)	(1.73)	(1.45)	(2.55)
FFC α	−0.56	−0.09	−0.10	0.07	0.19	0.33	0.34	0.27	0.28	0.26	0.83
	(−2.09)	(−0.46)	(−0.50)	(0.48)	(1.03)	(2.06)	(2.14)	(1.81)	(1.75)	(1.50)	(2.44)
FF5 α	−0.51	−0.11	−0.15	0.00	0.08	0.18	0.22	0.15	0.16	0.18	0.68
	(−2.03)	(−0.55)	(−0.83)	(0.02)	(0.39)	(1.20)	(1.49)	(1.01)	(1.01)	(0.98)	(2.04)
FF6 α	−0.48	−0.09	−0.13	0.04	0.14	0.24	0.24	0.15	0.17	0.19	0.67
	(−1.92)	(−0.43)	(−0.66)	(0.27)	(0.78)	(1.76)	(1.65)	(1.03)	(1.09)	(1.06)	(1.94)
Q5 α	−0.46	−0.05	−0.09	0.06	0.16	0.32	0.36	0.29	0.34	0.31	0.77
	(−1.40)	(−0.19)	(−0.43)	(0.35)	(0.87)	(2.03)	(2.19)	(1.73)	(1.78)	(1.49)	(1.68)
Panel B: Value-Weighted Portfolios											
Exc Return	0.74	0.80	0.61	0.66	0.80	0.95	1.08	0.66	0.62	0.59	−0.16
	(1.52)	(2.56)	(1.81)	(2.02)	(2.08)	(3.04)	(4.27)	(2.76)	(2.61)	(1.89)	(−0.42)
Sharpe Ratio	0.35	0.47	0.38	0.43	0.49	0.61	0.77	0.48	0.43	0.42	−0.08
CAPM α	−0.23	−0.02	−0.20	−0.12	0.00	0.19	0.36	−0.02	−0.04	−0.03	0.19
	(−0.80)	(−0.08)	(−1.02)	(−0.84)	(0.02)	(1.03)	(1.96)	(−0.12)	(−0.21)	(−0.19)	(0.52)
FF3 α	−0.20	0.00	−0.18	−0.11	0.01	0.20	0.36	−0.03	−0.07	−0.03	0.17
	(−0.70)	(−0.01)	(−0.98)	(−0.77)	(0.04)	(1.07)	(1.95)	(−0.20)	(−0.38)	(−0.19)	(0.46)
FFC α	−0.20	0.01	−0.16	−0.09	0.08	0.24	0.37	−0.03	−0.06	−0.03	0.17
	(−0.71)	(0.06)	(−0.85)	(−0.60)	(0.42)	(1.34)	(2.03)	(−0.17)	(−0.32)	(−0.19)	(0.46)
FF5 α	−0.01	0.07	−0.17	−0.05	0.07	0.15	0.29	−0.08	−0.18	−0.10	−0.08
	(−0.06)	(0.28)	(−0.87)	(−0.32)	(0.31)	(0.87)	(1.74)	(−0.49)	(−1.05)	(−0.56)	(−0.24)
FF6 α	−0.02	0.08	−0.15	−0.03	0.13	0.19	0.30	−0.08	−0.18	−0.10	−0.08
	(−0.06)	(0.33)	(−0.78)	(−0.18)	(0.63)	(1.13)	(1.82)	(−0.48)	(−0.99)	(−0.54)	(−0.23)
Q5 α	−0.14	0.02	−0.18	−0.04	0.07	0.16	0.34	−0.07	−0.07	−0.03	0.11
	(−0.45)	(0.07)	(−0.86)	(−0.27)	(0.35)	(0.93)	(2.02)	(−0.39)	(−0.36)	(−0.16)	(0.24)

Table 10: Time Series Regression, Binary Classification Model, Non-Micro Stocks

	CAPM	FF3	FFC	FF5	FF6	FF6 + Controls
	(1)	(2)	(3)	(4)	(5)	(6)
Alpha	0.010***	0.009***	0.009***	0.006**	0.006**	0.009***
	(0.003)	(0.003)	(0.003)	(0.002)	(0.002)	(0.003)
MKT	-0.410***	-0.255**	-0.264***	-0.259***	-0.262***	-0.215***
	(0.132)	(0.112)	(0.101)	(0.091)	(0.086)	(0.076)
SMB		-0.915***	-0.918***	-0.675***	-0.676***	-0.740***
		(0.167)	(0.166)	(0.138)	(0.137)	(0.134)
HML		0.204**	0.193*	0.415***	0.410***	0.317**
		(0.101)	(0.106)	(0.140)	(0.140)	(0.145)
RMW				0.798***	0.798***	0.851***
				(0.155)	(0.155)	(0.161)
CMA				-0.330	-0.325	-0.382*
				(0.255)	(0.243)	(0.227)
UMD			-0.030		-0.010	0.024
			(0.121)		(0.117)	(0.116)
ΔCIV						-0.001
						(0.002)
ΔPIV						-0.003
						(0.002)
log(O/S)						0.003**
						(0.001)
IVS_{ATM}						-0.003*
						(0.002)
IVS_{OTM}						0.001
						(0.002)
Observations	236	236	236	236	236	228
R^2	0.118	0.286	0.287	0.353	0.353	0.379
Adjusted R^2	0.114	0.277	0.274	0.339	0.336	0.347
Residual Std. Error	0.050	0.045	0.045	0.043	0.043	0.043
F Statistic	9.683***	10.882***	8.508***	12.127***	10.313***	8.758***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 11: Time Series Regression, Binary Classification Model, S&P 500 Stocks

	CAPM	FF3	FFC	FF5	FF6	FF6 + Controls
	(1)	(2)	(3)	(4)	(5)	(6)
Alpha	0.009***	0.008**	0.008**	0.007**	0.007*	0.009**
	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.004)
MKT	-0.494***	-0.386***	-0.373***	-0.388***	-0.374***	-0.321***
	(0.156)	(0.132)	(0.123)	(0.122)	(0.119)	(0.087)
SMB		-0.494***	-0.490***	-0.343*	-0.339*	-0.349*
		(0.188)	(0.186)	(0.201)	(0.199)	(0.195)
HML		-0.119	-0.103	-0.034	-0.006	0.051
		(0.157)	(0.164)	(0.212)	(0.223)	(0.246)
RMW				0.416*	0.416*	0.587***
				(0.220)	(0.219)	(0.225)
CMA				-0.109	-0.134	-0.259
				(0.328)	(0.326)	(0.328)
UMD			0.043		0.052	0.056
			(0.155)		(0.155)	(0.151)
ΔCIV						-0.003
						(0.002)
ΔPIV						-0.001
						(0.002)
log(O/S)						0.002
						(0.002)
IVS_{ATM}						-0.003
						(0.002)
IVS_{OTM}						0.003
						(0.002)
Observations	236	236	236	236	236	228
R^2	0.128	0.166	0.167	0.176	0.177	0.207
Adjusted R^2	0.125	0.156	0.153	0.158	0.155	0.167
Residual Std. Error	0.057	0.056	0.056	0.056	0.056	0.056
F Statistic	10.086***	4.295***	3.564***	3.044**	2.703**	3.961***

Note: *p<0.1; **p<0.05; ***p<0.01