57º SBPO
SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
Gramado • RS
DE 5 A 9 DE OUTUBRO DE 2025

SOME STATISTICAL LEARNING TECHNIQUES FOR STOCK RETURN PREDICTION

Antoine Campos
PUC Rio
Rio de Janeiro - Brazil
antoine.campos@etu.minesparis.psl.eu

Grégor Roncin
PUC Rio
Rio de Janeiro - Brazil
gregor.roncin@etu.minesparis.psl.eu

ABSTRACT

This paper presents the application of several linear and non-linear Statistical Learning techniques for predicting stock movements (upward or downward) in financial markets. The dataset comes from the quantitative investment fund Qube Research & Technologies and was made available for a Data Challenge featuring a public accuracy leaderboard. The project began with feature engineering based on existing variables in the dataset and normalization. We then applied methods including basic logistic regression, Lasso regression, and PCA-based regression. We also used Support Vector Classifier, Random Forests, and various boosting techniques, all evaluated using cross-validation and GridSearch. The results were excellent, placing us 2nd in France out of 960 participants.

KEYWORDS. Classification. Features Engineering. Quantitative Finance.

1. Introduction

The rise of big data and computational power has profoundly transformed practices in financial markets. At the heart of this evolution lies quantitative finance, a discipline that leverages mathematical and statistical tools to model financial dynamics, assess risks, and design investment strategies. Among the most influential players in the field are systematic investment funds, which make automated decisions based on algorithms extracting quantitative signals from large volumes of market data.

Over the past decade, these funds have increasingly adopted techniques from Statistical Learning such as Gu et al. [2020] shows. Going beyond classical linear models, these methods enable the capture of complex, often non-linear relationships between market variables and future asset behavior. The goal is clear: to accurately anticipate price movements and thus gain a competitive edge in fast-paced, highly competitive markets.

In this context, our project focuses on the practical application of several supervised Statistical Learning methods both linear and non linear to a real world problem of stock movement prediction. The dataset, provided by Qube Research & Technologies, was part of a national Data Challenge aimed at evaluating participants' ability to build robust predictive models. This work illustrates not only the range of approaches that can be employed from logistic regression to ensemble methods such as random forests and boosting but also the importance of critical preprocessing steps such as feature engineering, normalization, and rigorous model evaluation through cross validation and hyperparameter tuning.

2. Theoretical Foundations

This section introduces the main Statistical Learning methods we applied in the project, drawing on concepts from *An Introduction to Statistical Learning* (James et al., 2013). Each of these methods addresses the binary classification task—predicting an upward or downward movement—by exploiting different ways of modeling the relationships between explanatory variables (features) and the target variable. We refer the reader to James et al. [2013] for a comprehensive treatment of these techniques.

2.1. Logistic Regression and Lasso

Logistic regression is a linear classification model commonly used when the response variable is binary. It models the probability of belonging to class $y = 1$ using the logistic function:

$$P(Y = 1|X) = \frac{1}{1 + \exp(-\beta_0 - \beta_1 X_1 - \cdots - \beta_p X_p)}$$

Unlike linear regression, this model doesn't aim to approximate the response directly, but rather its log-odds.

To avoid overfitting and automatically select the most relevant features, we used the Lasso (Least Absolute Shrinkage and Selection Operator) variant, which adds an L_1 penalty on the absolute values of the coefficients:

$$\min_{\beta} L(\beta) + \lambda \sum_{j=1}^{p} |\beta_j|$$

This regularization encourages sparsity by shrinking some coefficients to zero, making the model more parsimonious and interpretable while reducing variance.

2.2. Dimensionality Reduction: Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is an unsupervised method for dimensionality reduction. It transforms a potentially correlated set of variables into a new system of orthogonal components, ranked by explained variance.

By projecting data onto the first few principal components, we retain most of the information while reducing model complexity. We applied PCA ahead of certain linear models to mitigate noise and prevent overfitting, especially in high-dimensional settings.

2.3. Decision Trees, Bagging, and Random Forests

Decision trees are non-parametric models that recursively partition the feature space into regions that are as homogeneous as possible with respect to the target class. While highly interpretable, they are prone to high variance—small changes in the training data can significantly alter the tree structure.

Bagging (Bootstrap Aggregating) addresses this issue by training multiple trees on bootstrap samples and aggregating their predictions (e.g., via majority vote). Random Forests extend this idea by randomly selecting a subset of features at each split, introducing additional diversity.

These ensemble methods significantly reduce variance without increasing bias and proved to be particularly robust in our use case.

2.4. Boosting: Additive Models and Error Weighting

Boosting is another ensemble technique, but unlike bagging, it builds models sequentially. Each new tree is trained to correct the errors made by the previous ones. The final model is a weighted combination of these "weak learners," resulting in a highly effective overall predictor.

Algorithms like Gradient Boosting and XGBoost optimize these weights efficiently and are capable of capturing complex relationships. These models performed especially well in our highly competitive predictive context, such as the QRT Data Challenge.

2.5. K-Nearest Neighbors (KNN)

The K-Nearest Neighbors classifier (KNN) relies on a straightforward idea: to predict the class of a new observation, identify the k closest points in the feature space and assign the most common class among them.

KNN is a non-parametric method that makes no strong assumptions about the decision boundary's shape. While simple and interpretable, it is highly sensitive to the number of features (curse of dimensionality) and requires careful scaling of variables to avoid distance-related biases.

Though KNN was not the top-performing model in our project, it served as a simple and interpretable baseline.

3. Methodology

For problems in quantitative finance, the most critical phase often lies in data preparation: selecting features, engineering new ones, and normalizing inputs. Leveraging our understanding of market dynamics, we were able to preprocess the dataset effectively—removing uninformative features and mathematically constructing new ones that were more predictive.

This process is detailed in Section 4. Once preprocessing was complete, our methodology was straightforward: systematically test a range of models, starting with linear approaches and progressively exploring non-linear ones. For each, we carefully selected and optimized hyperparameters using cross-validation, most often with 5 folds despite the

dataset's large size. Once the optimal hyperparameters were determined, we evaluated final model performance on a held-out test set to obtain the final accuracy and other performance metrics.

4. Data analysis

4.1. Variable structure and typology

This dataset is designed for a binary classification task aiming to predict the direction (sign) of residual stock returns.

Structure of the Dataset

- **Training input dataset** (features): Contains 418,595 rows and 47 columns.
- **Training output dataset** (target): Contains 418,595 rows and 2 columns: `ID` and `RET`.

Each row corresponds to a unique observation, combining features related to a specific stock and a specific (anonymized) date.

Input Features

The input datasets contain 47 columns, which can be grouped as follows:

- **ID**: A unique identifier for each row (observation).
- **DATE**: An index representing the date of observation. Note that dates are anonymized and randomly ordered, hence there is no temporal continuity.
- **STOCK**: An index identifying the stock to which the observation refers.
- **SECTOR**: A categorical index representing the broad economic sector of the stock.
- **INDUSTRY**: A categorical index indicating the industry classification of the stock.
- **INDUSTRY_GROUP**: A higher-level grouping of similar industries.
- **SUB_INDUSTRY**: A more granular classification nested within industries.
- **RET_1 to RET_20**: The residual returns of the stock over the last 20 days. `RET_1` corresponds to the most recent return (lag 1), `RET_2` to the return from two days prior, and so on. These returns have been adjusted to remove market-wide effects (residualized).
- **VOLUME_1 to VOLUME_20**: The relative volume traded over the last 20 days. Similar to the return variables, `VOLUME_1` is the volume from the previous day, and so forth.

These variables are primarily numerical (continuous) except for the categorical indices such as `STOCK`, `DATE`, `SECTOR`, `INDUSTRY`, `INDUSTRY_GROUP`, and `SUB_INDUSTRY`, which are intended for categorical encoding during preprocessing.

Target Variable

The output (label) dataset includes:

- **ID**: Matches the ID from the input dataset.
- **RET**: A binary label indicating the sign of the stock's residual return at time t. It takes values in $\{0, 1\}$:
  - 1 if the return is positive,
  - 0 if the return is zero or negative.

The task consists of learning a mapping from the 46 features to the binary target indicating whether the stock's return will be positive the next day.

Objective

The objective is to build a predictive model that can forecast the sign of the residual return (`RET`) for each test sample, using the 46 explanatory variables provided.

4.2. Features Engineering

A key component of our approach was enhancing the raw data through the creation of derived variables. This feature engineering step aimed to capture relevant market behaviors using empirical financial insights. Our engineered variables were structured around the following themes:

Momentum Indicators

These features reflect recent price dynamics, building on the well-documented momentum effect:

- `MEAN_RET_5`: average daily return over the past 5 days
- `RET_TREND = RET_20 - RET_19`: change in residual return over the last day.
- `RET_SKEW`: skewness of returns over the past 20 days
- `POS_RET_RATIO`: proportion of positive-return days among the last 20

Reversal Indicator

- `REVERSAL_SCORE = -RET_20 * MEAN_RET_5` : this score increases when the daily return is negative despite recent positive momentum, signaling a possible technical rebound.

Volatility Measures

- `VOLATILITY_20`: standard deviation of returns over the past 20 days

Volume Activity (Liquidity and Anomalies)

- `VOLUME_SPIKE = VOLUME_1 / mean(VOLUME_20 to VOLUME_11)`
- `VOLUME_TREND = VOLUME_20 - VOLUME_19`
- `CORR_RET_VOLUME`: 20-day correlation between returns and volume

Return-Volume Interactions

- `RET_VOLUME_INTERACTION = RET_1 * VOLUME_1`

Encoding of Categorical Variables

- One-hot encoding for linear models
- Target encoding (mean encoding) for tree-based models

All encodings were computed only on the training data to avoid data leakage.

Differentiation and Lags

- `DELTA_RET_i = RET_i - RET_(i+1)` for $i = 1$ to 19
- `ROLLING_DIFF_MEAN`: mean of the first five `DELTA_RET`

All features were standardized (zero mean, unit variance) before being fed into the models, ensuring compatibility with scale-sensitive algorithms.

4.3. New data analysis

In order to prepare the dataset for statistical modeling, we performed a comprehensive exploratory data analysis (EDA), distinguishing between continuous and categorical (binary) variables to ensure the robustness and relevance of the methods applied.

4.3.1. Outlier Detection

The presence of extreme values within continuous variables was evaluated using two complementary approaches:

a. Z-Score Method (Standardized Values)

Outliers were identified by applying a threshold of 3 standard deviations from the mean (i.e., $|Z| > 3$):

Variables such as `RET_SKEW` (2.00% of observations), `DELTA_RET_2` (1.93%), and `RET_3` (1.87%) exhibited the highest proportions of outliers.

The majority of features demonstrated minimal outlier presence, for example, `CORR_RET_VOLUME` showed 0% outliers.

These findings suggest that, although certain return-based variables display mild tail risk, the overall dataset remains stable under assumptions of Gaussian distribution.

b. Interquartile Range (IQR) Method

A second detection method utilized the interquartile range (IQR), flagging values outside the interval $[Q_1 - 1.5 \times \text{IQR}, Q_3 + 1.5 \times \text{IQR}]$:

Notably high outlier proportions were observed for engineered features such as `REVERSAL_SCORE` (20.3%), `VOLUME_SPIKE` (16.6%), and `RET_VOLUME_INTERACTION` (16.1%).

These results highlight that certain derived or higher-order features possess heavier tails or exhibit non-linear dispersion patterns.

4.3.2. Correlation analysis

The Pearson correlation matrix (see Figure 1) reveals several notable patterns:

- **Significant correlations among past returns:** Variables such as `RET_8`, `RET_15`, and `RET_18` exhibit moderate positive correlations with `RET_1` (approximately 0.3 to 0.5), which is consistent with short-term return persistence.

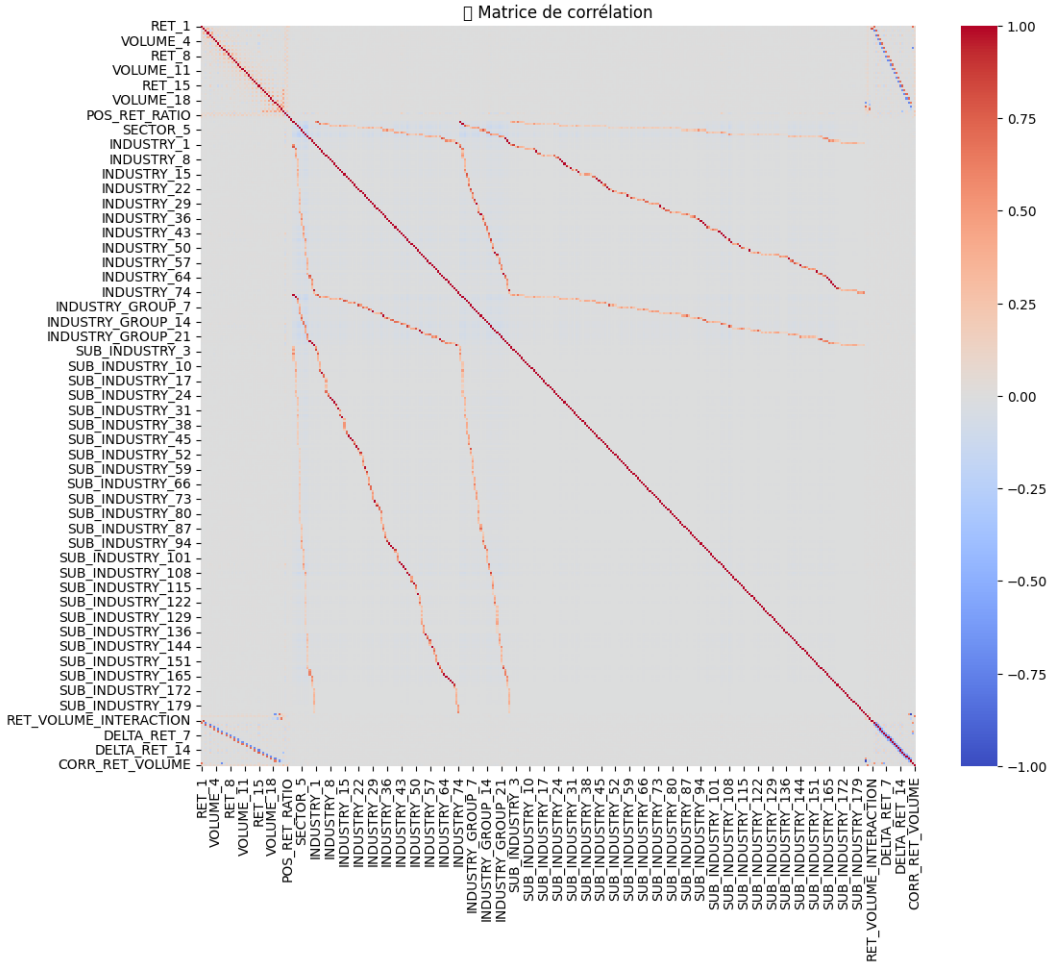

Figure 1: Correlation Matrix of the complete dataset X

- **Weak linear correlations between volumes and returns:** The `VOLUME_*` variables show weak correlation with `RET_1`, suggesting that their influence may be nonlinear or dependent on contextual factors.
- **Strong internal correlations among categorical variables:** One-hot encoded variables representing industry and sector display strong internal collinearity, which is expected due to their mutually exclusive encoding scheme.

4.3.3. Algorithmic implications

The presence of outliers concentrated in certain derived variables (notably `REVERSAL_SCORE`, `VOLUME_SPIKE`, and `RET_VOLUME_INTERACTION`), combined with moderate correlation patterns among lagged return variables, indicates a structural complexity in the data distribution.

Indeed, these variables with a high proportion of outliers appear to represent extreme market phenomena or nonlinear behaviors that are not captured by the simple linear relationships observed in historical returns (e.g., `RET_1`, `RET_8`, `RET_15`). This dissociation between the relatively stable and moderate correlation structure in returns and the localized

presence of outliers in specific features suggests that critical information is carried by these extreme observations, which may disrupt global linear relationships.

Furthermore, the weak linear correlation between volume and returns, coupled with the high incidence of outliers in volume-related features, supports the hypothesis that these variables encapsulate conditional and sporadic effects (spikes, market anomalies) that do not follow the regular dynamics of returns. These phenomena introduce heterogeneity in the data distribution that could bias simple linear models due to violations of homoscedasticity and normality assumptions.

Finally, the strong internal collinearity of categorical variables highlights structural redundancy which, when combined with extreme dispersion in certain continuous variables, poses a challenge in simultaneously capturing robust signals and rare but informative events.

Key implications for the algorithm:

The algorithm must handle a dual dynamic: a moderately stable and correlated structure in lagged returns, alongside sporadic and extreme signals carried by high-variability, outlier-prone features.

The coexistence of these phenomena is likely to generate complex, nonlinear interactions that may affect model stability and generalization if not simultaneously accounted for.

Thus, the model should integrate both the continuity of trends (via correlations) and the rarity of extreme events (via outliers) within a unified modeling approach.

5. Results

5.1. Logistic Regression

A. Naive regression

The first statistical learning algorithm we applied was logistic regression on the unmodified dataset, without the inclusion of newly engineered features. As a standard method for classification tasks, logistic regression served as our starting point, allowing us to benchmark performance with the most basic model setup—a raw logistic regression on the original dataset. See Table 1.

Table 1: Performance Metrics for Naive Logistic Regression

Metric	Score
Accuracy Score	0.5106
Precision Score	0.5105
Recall Score	0.5106
F1 Score	0.5103
ROC AUC Score	0.5105

As expected, the results obtained were relatively modest compared to the more advanced approaches presented later. We then trained the same basic logistic regression model on the dataset augmented with the newly engineered features. The performance significantly improved compared to the baseline, as shown in Table 2.

B. Lasso

The next step involved applying a more sophisticated variant of logistic regression: **Lasso logistic regression**. Given that the dataset now contains a large number of explanatory variables—some of which are derived transformations, and others potentially redundant or weakly informative—the Lasso method acts as an automatic variable selector.

Table 2: Performance Metrics for Logistic Regression with Engineered Features

Metric	Score
Accuracy Score	0.5216
Precision Score	0.5216
Recall Score	0.5216
F1 Score	0.5216
ROC AUC Score	0.5216

By applying L1 regularization, Lasso not only performs feature selection by shrinking some coefficients to zero, but also helps reduce overfitting and improves the interpretability of the model.

The results were conclusive, with **16 coefficients reduced to zero** for a regularization strength of $\lambda = 1$, and **142 coefficients reduced to zero** for $\lambda = 10$ and performance metrics slightly better than those obtained from the basic logistic regression model (see Table 3).

Table 3: Performance Metrics for Lasso Logistic Regression

Model	Accuracy	Precision	Recall	F1 Score	ROC AUC
Lasso (16 zeroed)	0.5223	0.5224	0.5223	0.5222	0.5224
Lasso (142 zeroed)	0.5222	0.5224	0.5222	0.5218	0.5223

C. PCA pre-treatment of the data

We also explored an alternative strategy with a similar goal: applying a **Principal Component Analysis (PCA)** prior to basic logistic regression. PCA can reduce dataset dimensionality and redundancy, as well as filter out noise. The key idea is to make the model more generalizable on unseen test data: by projecting the data onto the top $n = 50$ principal components, we eliminate low-variance directions often associated with noise.

The choice of $n = 50$ components was not arbitrary—it was guided by cross-validation to balance information retention and generalization. Since logistic regression is a linear model, it can benefit from a more structured and compressed feature space.

However, the results did not show any significant improvement over standard logistic regression. This might be due to the large size of the dataset, which may already dilute noise and make dimensionality reduction less impactful.

The table below summarizes the results of this approach: Table 4.

Table 4: Performance Metrics for Logistic Regression with Engineered Features and PCA

Metric	Score
Accuracy Score	0.5153
Precision Score	0.5155
Recall Score	0.5153
F1 Score	0.5143
ROC AUC Score	0.5154

5.1.1. Support Vector Classifier

The next method we decided to try was the **Linear Support Vector Classifier (SVC)**. Unlike logistic regression, which adjusts the decision boundary using all training

points, the linear SVC focuses only on the *support vectors* — the points closest to the boundary between classes.

This property makes the SVC more robust to noise and potentially better suited to noisy data, such as financial market variables. Furthermore, this method is well-suited to high-dimensional spaces, which is the case in our dataset.

We selected a relatively small regularization parameter, $C = 0.2$, to allow for a more flexible margin. This value yielded better cross-validated performance than other tested values (e.g., C = 1, C = 5, C = 0.01). The rationale is to permit some misclassifications in the training set, given the high dimensionality and noisy nature of financial data, which likely prevent linear separability.

However, the results obtained were quite average, as shown by the performance metrics in Table 5

Table 5: Performance Metrics for Linear Support Vector Classifier (SVC)

Metric	**Score**
Accuracy Score	0.5095
Precision Score	0.5129
Recall Score	0.5095
F1 Score	0.4644
ROC AUC Score	0.5085

5.1.2. Trees and Forests

The family of tree-based models proved particularly relevant in our study, especially due to their ability to capture complex non-linear interactions between financial variables. We evaluated several variants: simple decision tree, random forest, gradient boosting, XGBoost, and LightGBM. The performance of each model in terms of accuracy, F1-score, and AUC is presented in **TABLE**.

A. Decision Tree (DecisionTreeClassifier)

A maximum depth of 10 was imposed on the decision tree to limit overfitting while maintaining a degree of expressiveness. Despite this, its performance remained low (see Table 6), confirming that this model, although interpretable, is too simplistic for such complex financial data. It tends to focus on volatility and return-related variables, but lacks the ability to generalize effectively.

Table 6: Performance Metrics for Tree-Based Models

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**ROC AUC**
Decision Tree	0.5226	0.5227	0.5226	0.5220	0.5277
Random Forest	0.5490	0.5491	0.5490	0.5487	0.5749
XGBoost	0.5646	0.5646	0.5646	0.5646	0.5922
LightGBM	0.5599	0.5599	0.5599	0.5598	0.5858

B. Random Forest (RandomForestClassifier)

With 351 trees and a maximum depth of 15, the random forest leverages the variance reduction from ensemble learning. It achieves slightly better performance compared to

the simple decision tree (see Table 6), particularly in F1-score. This model is more effective at identifying key variables, such as lagged returns and volume indicators, by capturing interactions more robustly.

C. Gradient Boosting (GradientBoostingClassifier)

The Gradient Boosting model uses 100 shallow trees (max_depth=3) with a learning rate of 0.1, following a controlled, incremental learning strategy. This helps to progressively correct errors without overfitting. The results in Table 6 indicate moderate improvements over the previous models, although managing the bias-variance tradeoff remains challenging.

D. XGBoost (XGBClassifier)

XGBoost, configured with 600 trees, a depth of 10, and a learning rate of 0.1, leverages regularized boosting to handle complex structures. This model achieves significantly higher performance, especially in terms of AUC (see Table 6), indicating stronger class discrimination. It assigns growing importance to both sectoral and temporal variables, reflecting a more refined capture of underlying signals.

E. LightGBM (LGBMClassifier)

LightGBM stands out for its efficiency and scalability. Configured with 600 trees, a maximum depth of 18, a learning rate of 0.1, and `min_child_samples`=30, it models complex interactions while avoiding overfitting to small subgroups. The results (Table 6) confirm its competitiveness, with scores comparable to XGBoost. It emphasizes recent return variables (RET_x), volume metrics, and certain momentum indicators, highlighting its sensitivity to short-term market dynamics.

5.1.3. KNN (K nearest neighbors)

K-Nearest Neighbors is a simple yet powerful algorithm that assigns a class to an observation based on the classes of the k closest neighbors in the feature space. Its relevance in our context stems from several key aspects:

- **Non-parametric nature:** KNN makes no assumptions about the underlying data distribution or the form of the decision boundary, making it well-suited to capturing non-linear or local relationships—frequently observed in market behavior.
- **Adaptability to local structure:** The model's reliance on neighboring samples allows it to adapt locally to localized effects and market anomalies.
- **Built-in regularization via k:** The hyperparameter k controls model complexity. Smaller values of k can capture finer details but are prone to overfitting, while larger values smooth the classification boundary and reduce variance.

Using cross-validation, we selected $k = 21$ as the optimal number of neighbors. The KNN model performed strongly on our financial classification task, confirming its suitability for the problem. The following table summarizes its performance metrics Table 7

Table 7: Performance Metrics K-Nearest Neighbors (K = 21)

Metric	Score
Accuracy	0.5681
Precision	0.5687
Recall	0.5681
F1 Score	0.5674
ROC AUC Score	0.5682

6. Conclusion

We performed feature engineering by creating new features derived from existing dataset columns, followed by the application of various Statistical Learning algorithms. As expected, random forest boosting methods and k-nearest neighbors emerged as the best-performing models for our problem. Using these nonlinear algorithms, we achieved a test set accuracy exceeding 56%, which would place us second on the data challenge leaderboard out of more than 960 participants. Our results echo recent findings in the literature Gu et al. [2020] showing that machine learning methods can uncover weak yet exploitable signals in asset return data.

Predicting financial market movements is a notoriously difficult task, as markets are influenced by a multitude of often random or unobservable factors. Nevertheless, rigorous data preparation combined with well-chosen Statistical Learning methods enables systematic investment funds to detect weak signals and build potentially profitable strategies. This project demonstrates that even in the face of apparent market unpredictability, a well-structured quantitative approach can provide a real competitive advantage.

References

Gu, S., Kelly, B., and Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer, 1 edition. URL https://www.statlearning.com/.