Stock Outperformance Prediction Report

Kaggle user ID: michaelxie553

1. Collaboration Process

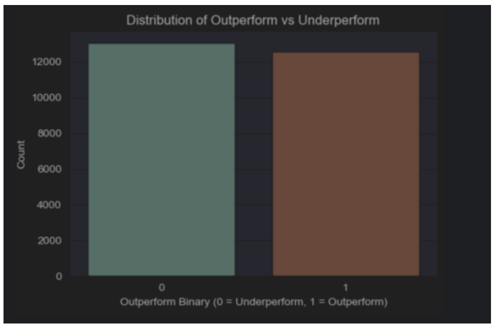
This machine learning project was completed by a group of two students. The feature development and modelling process was highly collaborative, with regular discussions held in person and via Zoom to align on EDA findings, preprocessing strategies, and feature selection decisions. We shared responsibilities by jointly validating data preparation steps, while at the modelling stage each member explored two different classification algorithms. Results were then compared and discussed to identify the most effective approach, ensuring both diversity of experimentation and consensus-driven decision making.

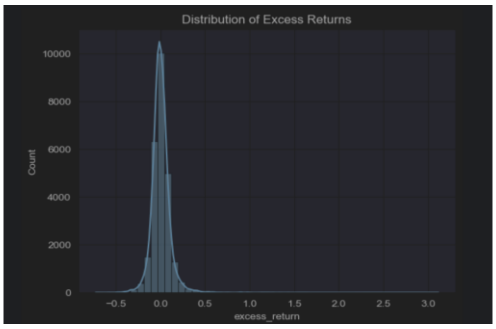
2. Pre-Modelling

2.1. Exploratory Data Analysis

The goal of Exploratory Data Analysis (EDA) in our stock prediction project is to understand the structure, distribution, and relationships of features within the datasets to inform our preprocessing steps and modelling. For example, we check for dataset shape, missing values, feature input format, outliers, categories within categorical features. After going through each datasets, we discovered the followings:

training_targets.csv: The distribution of our target variable 'outperform_binary ' for the classification model is almost perfectly evenly distributed which means we do not need to consider techniques to address class imbalance such as sampling. For 'excess_return' however, its distribution is right skewed which is not abnormal for stock returns.





company_info.csv: There is a huge imbalance across sectors. Technology is the largest sector which is not surprising since US markets are heavily weighted toward tech, while some sectors are relatively small. This may lead to overfitting because if most outperformers come from Tech, the model might rely too heavily on "sector = Tech" as a predictor. Consequently, it might lead to poor generalization for small sectors:

stock_data.csv: distributions of many features such as price_range_ratio, avg_volume_3m and volatility_3m are heavily right-skewed. Transformations may be necessary for normalization (e.g., log or Box-Cox transformations) to stabilize variance, reduce skewness and outlier, and improve model performance.

Other datasets: index, optional data: distributions of many of these features are also heavily right-skewed. But to preserve interpretability and inference for useful macro information such as "when fed_rate changes by x, excess_return changes by x", we shall approach them with caution.

2.2 Data Preprocessing

Missing Values - stock_data.csv

● No missing values were detected in our datasets except for stock_data.csv, where nulls occur in features that require historical lookbacks (e.g. return_6m, volatility_6m).

- Additional exploration shows that these missing values only occur at the beginning of various stocks, as such we are unable to impute based on past month's data.
- Low proportion of missing values at just 0.0052 of total rows, hence we simply remove them

Lagging/Alignment - stock_data.csv, monashindex.csv & optional_data

- In order to avoid look-ahead bias, we shift all temporal features backwards by one month so that features from the previous month are used to predict a stock's performance in the future.

Encoding Categorical Values - company_info.csv

Many classification models require numerical values, as such, it is necessary to encode our categorical values by either one-hot-encoding or ordinal encoding. We explored possible values for each feature and performed the following encodings:

- One-hot-encoding (Nominal features): sector, business_model, geographic_focus, business_maturity, competitive_position and asset_intensity
- Ordinal Encoding (Ordinal features):
- market_cap_category → Small < Mid < Large
- revenue_tier → Tier_3 < Tier_2 < Tier_1
- profitability_profile → Low_Margin < Standard < High_Margin
- financial_strength → Developing < Stable < Strong

2.3 Feature Engineering

After performing point-biserial correlation across existing features with our target variable "outperform_binary" revealed that the significant majority of features contained low univariate predicting power (all below 0.10). Hence, we were led to assume that stock outperformance is not driven by any single factor in isolation but rather by the combined effect of multiple features and their interactions. For example, the following are some of the engineered features (not all) and their rationals:

- **Yield Slope (`10y_treasury − 5y_treasury`)** – Shape of the yield curve; inversions often signal recession risks affecting equities.
- **Real Rate (`fed_rate − inflation_rate`)** – True cost of borrowing; high values can reduce investment and spending, weighing on stocks.
- **Regime High Inflation (`inflation_rate > median`)** – Flags periods of elevated inflation; pressures margins and dampens sentiment.
- **Sector Interactions (`unemployment_rate × sector`, `vix × sector`)** – Allows macro shocks to vary by sector; cyclical industries more sensitive than defensives

2.4 Feature Selection

As all features are relatively weak univariate predictors of our target variable, we applied feature selection by identifying and removing highly collinear predictors. Collinearity can inflate variance, distort feature importance, and reduce model stability, particularly for linear models where coefficients become unreliable. We used a number of methods:

Cramér's V Test: We performed Cramer's V test across categorical features, and it revealed that some of them are highly collinear. For example, market_cap_category and revenue_tier have a perfect association score (V = 1.0), indicating redundancy. Hence, we decided to drop revenue_tier.

Collinearity & Correlation: We performed a collinearity heatmap and flagged all features who have a collinearity of above 0.8 with another feature. Then went through each of these feature combinations and applied point-biserial correlation as a secondary metric to decide which of the two features to remove. To resolve collinearities, we performed a comparison across each feature using univariate predictional power and p-value to decide which feature to keep.

2.5 Feature Transformations

We select a subset of features for transformation to reduce skewness and stabilize variance. The chosen set (prices, returns, volatilities, macro factors) are the ones with:

- Large numeric scale differences (prices vs interest rates vs index points).
- Skew/heavy-tailed behavior (returns, VIX, volatility, volume ratios).
- Continuous distributions where linear model assumptions matter.

The not-chosen set are:
- Already standardized (ratios, binaries).
- Categorical/ordinal encodings.
- Derived interactions where scaling is less impactful than model regularization.

We automated the feature transformation process by having each feature tested with log, square root, Box–Cox, and Yeo–Johnson transformations, and the best option was selected using a composite score of skewness, kurtosis, and normality.

3. Classification Model Training

We explored a range of different classification models and ultimately decided to pick Logistic Regression, RandomForest and SVM. Methodology to optimising our models followed an iterative approach. For validating each one of our models, we used a time-aware validation scheme to avoid look-ahead bias. Data from all months prior to the validation month were used for training, with performance evaluated on the subsequent month.

3.1 Logistic Regression (L1 Regularisation)

Logistic regression is a simple, stable baseline for binary classification that remains transparent—its coefficients reveal each feature's directional effect on outperformance. With L1/L2 regularization, it naturally handles high-dimensional, collinear data by shrinking or zeroing weak predictors, effectively performing feature selection.

A range of techniques were explored to optimise our logistic regression model:

- Hyperparameter optimisation through gridsearch, focusing on regularisation
- Dimensionality Reduction methods such as Truncated SVD and PCA
- A combination of feature transformations
- Polynomial Expansion to help identify non-linear feature interactions

Model	Mean	Mean F1		Mean ROC	Notes
Default L1 Baseline	0.5519	0.1626		0.4594	Underfit. AUC below 0.5 which
Stronger Regularisation.	0.4816	0.3722		0.5000	Better performance, suggests
Stronger Regularisation	0.4816	0.3722		0.5000	No gains, transformations
PCA (n=5) + Stronger	0.5431	0.5741		0.5331	PCA reduced collinearity and
Polynomial Expansion	0.5402	0.2169		0.5003	Introduced high variance, poor
Best HyperParameters	0.4547	0.6175		0.5293	Inflated F1 score, over
Final Model:	**0.5431**	**0.5741**		**0.5331**	**Best Balance**

Overall, L1 regularisation combined with PCA performed the best overall with a more balanced precision-recall trade-off. Stronger regularisation strength was crucial in shrinking coefficients of noisy predictors as our feature space significantly lacked strong univariate predictors. Transformations unfortunately did not help as all of the transformed features were reduced to 0 during regularisation. PCA also drastically improved performance, likely helping to further reduce significant collinearity between multiple features.

3.2 Random Forest

Random Forest was selected as the baseline classification model due to its robustness in handling heterogeneous tabular data, ability to capture non-linear relationships, and built-in mechanisms for reducing overfitting through ensemble averaging.

A small hyperparameter grid search was conducted across tree depth, number of estimators, minimum samples per leaf, and feature sampling strategies, but results were poor with the best one having an F1 score of just 0.25. To further improve performance, permutation importance was applied to identify noisy features.

Model Setup	Mean F1	Std F1	Notes
Full feature set	0.13	0.09	High noise, weak signals, unstable results
Best hyperparameters (no	0.25	0.08	Fully grown trees with leaf regularisation
Pruned features (5 vars)	0.50	0.12	Strong, stable signals from lagged features

While pruning and hyperparameter tuning led to meaningful improvements, the overall performance of Random Forest remains relatively poor in absolute terms. An F1 score of 0.50 indicates only modest predictive capability and suggests the model struggles to capture the inherent noise and structural complexity of financial markets.

3.3 Support Vector Machine (SVM)

SVM was selected for its suitability for high-dimensional data, as well as our feature space's compatibility with dimension reduction using methods such as PCA. By maximising margin between classes, SVM can perform well in settings where signals are weak. Kernels such as RBF can also capture non-linear relationships which is likely to be the case for stock prediction.

Model	Mean Accuracy	Mean F1	Mean ROC AUC	Notes
Linear Kernel	0.5480	0.3209	0.4619	Unstable, underfitting
Linear Kernel + PCA (n=5)	0.5776	0.2296	0.5727	Better fit, but low F1 score
RBF Kernel	0.5093	0.3688	0.4769	RBF can capture non linear relationships
RBF Kernel + PCA (n=5)	0.5246	0.4080	0.5170	Even Better fit, stronger predictive power
Final Model: RBF + Best	**0.4992**	**0.4784**	**0.4929**	**Most stable variant**

Overall, SVM performance was mixed. Linear kernels underfit, while PCA slightly improved stability but hurt F1. The RBF kernel captured some non-linear structure, and with tuning plus PCA, produced the most stable results, though overall AUC remained close to random. This suggests SVMs add robustness but did not consistently outperform simpler regularised logistic models in this setting.

3.4 Model Comparison Table

Model	Mean F1 Score	Mean ROC AUC
Final Logistic	**0.5741**	**0.5331**
Final RandomForest	0.5000	0.5120
Final SVM	0.4784	0.4929

Overall, we decided to use our logistic regression model as our final classification model due to its strong performance with noisy predictors.

4. Regression Model Training

4.1 Model Exploration

Evaluating our regression models followed the same process as our classification models, using different months to validate our predictions and training only on months before the validation month.

Average Metrics per Model

Metric	Lasso	SVM	XGBoost	CatBoost	LightGBM	DecisionTree

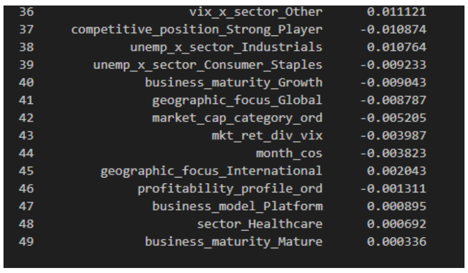
RMSE	**0.0955**	**0.1021**	**0.1074**	**0.1098**	**0.1141**	**0.1659**
MAE	**0.0665**	**0.0706**	**0.0747**	**0.0761**	**0.0796**	**0.1251**
R²	**-0.0387**	**-0.1999**	**-0.3138**	**-0.3976**	**-0.5376**	**-2.5212**

Exploration of a range of models fitted using the same features as our classification task revealed strong results in lasso regression and tree models such as XGBoost and CatBoost.

We used Lasso Regression as a baseline and optimised CatBoost due to its ability to handle nonlinear feature interactions and categorical data, as well as its generalisation ability which helps mitigate overfitting issues.

4.2 Feature Space Exploration

Running pearson correlation tests with our existing features and our target variable “excess_return” revealed a significant number of features had extremely low univariate correlation to our target variable. Coupled with poor prediction results where every stock was predicted the same excess_return, suggests that there is too much noise in the features which is preventing our models from



```
36                 vix_x_sector_Other    0.011121
37   competitive_position_Strong_Player   -0.010874
38          unemp_x_sector_Industrials    0.010764
39     unemp_x_sector_Consumer_Staples   -0.009233
40            business_maturity_Growth   -0.009043
41            geographic_focus_Global   -0.008787
42            market_cap_category_ord   -0.005205
43                    mkt_ret_div_vix   -0.003987
44                          month_cos   -0.003823
45      geographic_focus_International    0.002043
46          profitability_profile_ord   -0.001311
47            business_model_Platform    0.000895
48                  sector_Healthcare    0.000692
49            business_maturity_Mature    0.000336
```

learning meaningful relationships.
We employ gridsearch using a threshold for pearson correlation between (0.01, 0.10) to prune features and compare performance. Cross-validated results showed that removing features with an absolute correlation of below 0.02, reducing the feature space from 55 to 27 performed the best with an RMSE of 0.09567 - marginally better than lasso regression.

4.3 Hyperparameter Optimisation

We then ran grid search over an extensive parameter grid for CatBoost to obtain the optimum parameters which was then combined with our reduced feature selection to achieve a Kaggle score of 0.06445.

---

5. Conclusion

Our experiments highlight the difficulty of predicting stock outperformance given the weak univariate signals and noisy feature space. Among classification models, Logistic Regression with L1 regularisation and PCA emerged as the most balanced and interpretable approach, outperforming both Random Forest and SVM in terms of stability and precision–recall trade-off. For regression, Lasso Regression provided a strong baseline while CatBoost, combined with feature pruning and hyperparameter optimisation, delivered the best overall RMSE on the holdout set. These results suggest that simpler, regularised models paired with dimensionality reduction or feature pruning are more effective than complex non-linear methods when working with noisy financial data.

Key Predictors: Vix, month_end_close_usd, Vix X sector,