Advanced Programming 2025

Mining Feasibility Signals: Forecasting 30-day returns with Machine Learning

Final Project Report

Matthew Colliss
matthew.colliss@unil.ch
Student ID: 24440760

January 12, 2026

Abstract

This project develops a machine learning model to forecast 30-day stock returns following Definitive Feasibility Study (DFS) announcements by ASX-listed companies. In mining, a DFS is a detailed technical and economic assessment used to confirm a project's viability before an investment decision is made, often signaling whether a mine will proceed or be abandoned. The objective is to determine whether post-announcement price movements are predictable using project fundamentals and market data.

A dataset was compiled combining historical share prices, market indices, and extracted DFS metrics including NPV, IRR, CAPEX, production, mine life, and operating costs. An event study methodology aligned prices around announcement dates to compute raw and abnormal returns. Feature engineering and normalization were applied to improve model stability.

Multiple models were trained and benchmarked, including Linear Regression, Ridge, Random Forest, and XGBoost. Although an ensemble approach was tested, performance was primarily driven by linear models, with OLS delivering the most stable and accurate predictions across all evaluation metrics.

Key contributions include an automated ML pipeline, a reproducible event-study framework, and evidence that DFS fundamentals contain incremental predictive signal. While financial markets are inherently noisy and no model can consistently predict returns with certainty, the results suggest that project-level disclosures provide information beyond naive benchmarks.

Keywords: Data Science, Python, Machine Learning, Mining Feasibility, Return Forecasting]

Contents

1 Introduction

This project develops an end-to-end data science pipeline to forecast 30-day post-announcement stock performance following Definitive Feasibility Study (DFS) releases for ASX-listed mining developers. DFS announcements are major valuation events, signalling project economics, development readiness, and potential risks. Market reactions vary widely, indicating that outcomes depend on complex interactions between project fundamentals and market conditions.

Background and Motvation: DFS releases are key milestones in mine development. This project builds an evidence-based forecasting tool linking DFS fundamentals (e.g., NPV, IRR, CAPEX, costs, mine life) and market context to subsequent stock performance, enabling more consistent decision-making than headline interpretation.

Problem Statement: The central question is whether we can predict 30-day post-DFS stock returns better than simple baseline rules. To address this, two regression targets are defined: (i) **raw return**, measured as the total stock return over 30 trading days, and (ii) **abnormal return**, defined as the raw return adjusted for market performance. Models are benchmarked against naive strategies such as historical averages and constant mean predictions.

Objectives: The objectives of this project are to build a reproducible pipeline for data ingestion, feature engineering, and event-window construction; to train and compare OLS, Ridge, Random Forest, XGBoost, and a validation-based ensemble; to evaluate models using MAE, RMSE, R^2, and directional accuracy; and to produce interpretable metrics and visual comparisons.

Report Structure: The report covers data sources, methodology, preprocessing, modeling results, and concludes with discussion, limitations, and future research directions.

2 Literature Review

This project builds on three main research streams: event studies of mining announcements, post-announcement return predictability, and machine-learning models for stock-return forecasting.

2.1 Event Studies and Mining Announcements

Event study methodology aligns returns around announcement dates and estimates abnormal returns relative to an expected-return model, providing the standard framework for assessing how quickly markets incorporate new information. Within the mining sector, studies on exploration, resource, and reserve announcements for ASX-listed firms find statistically significant positive cumulative abnormal returns around disclosure dates (Bird et al., 2013). However, subsequent performance is often weak or negative, particularly for junior miners, suggesting delayed market adjustment and potential mispricing of complex project information (Bird et al., 2013).

This pattern reflects the difficulty investors face in interpreting technical disclosures such as grade distributions, metallurgical recoveries, cost assumptions, and development risks. As DFS announcements represent a more advanced and comprehensive project milestone, they plausibly generate both immediate price reactions and post-event drift similar to earlier-stage announcements. This evidence supports the suitability of DFS releases as candidates for predictive modeling of post-announcement returns.

2.2 Post-Announcement Drift and Return Predictability

The most well-documented post-announcement anomaly is post-earnings-announcement drift (PEAD), where stock prices continue to move in the direction of earnings surprises for weeks or months after the release (Ball & Brown, 2013). This phenomenon contradicts strict semi-strong market efficiency and suggests gradual information diffusion or investor underreaction(Bernard & Thomas, 1989).

Recent research applies machine-learning methods, particularly Random Forest and XGBoost, to predict 30-day cumulative abnormal returns following earnings announcements. These studies consistently find higher directional accuracy and lower forecast error compared to linear models, driven by non-linear interactions among earnings surprises, momentum indicators, liquidity measures, and firm characteristics(Ye & Schuller, 2021).

Related work in other information-driven events, such as pharmaceutical clinical-trial announcements, combines text-based sentiment models with gradient boosting algorithms to forecast announcement-induced price movements(Budennyy et al., 2023). These pipelines outperform traditional benchmarks, reinforcing the value of machine learning in capturing complex market reactions to discrete information events.

2.3 Machine Learning and Linear Models in Finance

Random Forest has been widely used in return and volatility forecasting due to its robustness to noise, ability to handle high-dimensional feature spaces, and built-in variable importance measures. Empirical studies demonstrate that Random Forest models reduce mean squared error and increase R^2 relative to standard linear regressions. Moreover, portfolios formed on Random Forest–predicted returns often outperform benchmark strategies on a risk-adjusted basis(Rathi et al., 2024).

Gradient boosting methods, particularly XGBoost, have become standard tools for tabular financial data. They consistently outperform linear and single-tree models by capturing complex non-linear relationships between technical indicators, firm fundamentals, and event characteristics(Kim, 2021). XGBoost has been shown to deliver superior predictive accuracy in PEAD-style post-announcement settings and broader cross-sectional return prediction tasks(Gu et al., 2020).

Despite these advances, linear and regularized models such as OLS and Ridge regression remain core tools in asset pricing and event study research. They provide transparent benchmarks and interpretable coefficients, allowing researchers to assess whether more complex machine-learning models extract genuine incremental predictive signal. In high-dimensional settings like this project—where DFS fundamentals (NPV, IRR, CAPEX, production, mine life, operating costs) are combined with market-based features—linear models serve as essential reference points for evaluating the economic significance of non-linear approaches(Sarisoy et al., 2016).

3 Methodology

3.1 Data Collection and Description

3.1.1 Finding Feasibility Studies

The first stage of this project involved compiling a database of mining definitive feasibility studies (DFS). A 20-year window (2007–2026) was selected, limited to ASX-listed companies across all commodities. No central public repository exists for these documents, making collection difficult due to inconsistent disclosures, document age, frequent updates, and the absence of a registry.

AI tools, including ChatGPT and Perplexity, were used to systematically search for reports containing terms such as "Definitive Feasibility Study," "Bankable Feasibility Study," and related phrases. Relevant documents were manually verified and downloaded as PDFs, resulting in approximately 170 valid DFS reports.

3.1.2 Data Extraction

Manual extraction was impractical due to document volume and variability. Instead, a custom Python pipeline using the OpenAI API automated the process. A two-stage approach was implemented: (1) a lightweight model identified announcement dates, which were used to obtain historical FX rates, and (2) a second model extracted financial metrics, converting all values to AUD for consistency.

Documents were processed in parallel with retry logic, and results were exported to Excel (*dfs_features_safe.xlsx*). Extracted variables included:

- Company details
- Primary metal
- NPV, IRR (pre/post tax)
- Payback period
- CAPEX and AISC
- Annual production
- Mine life
- Commodity price

Share price data were collected for 30 trading days before and after each announcement using the yfinance API. Where companies were delisted or unavailable, historical prices were manually sourced from Intelligent Investor.

To calculate market capitalisation, shares outstanding at the announcement date were required. A second automated pipeline searched ASX filings (primarily Appendix 2A and 3H) to extract historical share counts. When exact matches were unavailable, the closest prior disclosure was used. Results included source tracking and confidence ratings.

3.1.3 Data Quality

The final dataset contains approximately 170 observations across 22 variables, spanning 2007–2025 and covering a wide range of commodities. While metadata and market variables were fully recovered, financial metrics remain incomplete due to inconsistent DFS reporting standards. Missing values and unit variation (Table 1) required preprocessing during model training.

Future work could expand coverage to other jurisdictions or develop improved collection methods to increase dataset size and consistency.

Table 1: Data availability – project and financial metrics

NPV$_{post}$	IRR$_{post}$	NPV$_{pre}$	IRR$_{pre}$	Payback	Capex	AISC	Prod	Life	Price	Permit
69%	65%	61%	62%	75%	98%	92%	93%	97%	82%	82%

3.2 Approach

This study examines how mining stocks perform after companies release their Definitive Feasibility Study (DFS) announcements. The goal is to predict stock movements over the 30 days following these announcements for mining companies listed on the Australian Stock Exchange.

3.2.1 Target Variables

Two target variables are defined over a 30-trading-day post-announcement window. Let $P_{i,0}$ denote the closing price of stock i on the announcement trading day ($t = 0$) and $P_{i,30}$ the closing price at $t = 30$. The *raw return* is defined as:

$$R^{\text{raw}}_{i,30} = \frac{P_{i,30}}{P_{i,0}} - 1. \tag{1}$$

To control for broader market movements, *abnormal returns* are computed relative to the ASX All Mining Index. Let I_0 and I_{30} denote index closing levels at $t = 0$ and $t = 30$, respectively. The index return and abnormal return are defined as:

$$R_{30}^{\text{idx}} = \frac{I_{30}}{I_0} - 1, \qquad R_{i,30}^{\text{abn}} = R_{i,30}^{\text{raw}} - R_{30}^{\text{idx}}. \tag{2}$$

3.2.2 Naive Benchmark

Two baseline benchmarks were used to contextualise model performance. The first was a naive past-mean (momentum) baseline, which predicts the next 30-day return as the average of the previous 30 days and was applied to both raw and abnormal returns. The second was a constant mean baseline, which predicts a fixed value equal to the average return in the training set. Together, these baselines establish minimum performance thresholds and ensure that the machine learning models provide predictive value beyond simple heuristic approaches.

3.2.3 Feature Engineering

It was decided to group the features into categories that provide different types of fundamental information that could potentially signal stock price movements, including project value and returns (NPV and IRR), capital intensity and payback (initial capex and payback period), scale and longevity (annual production and mine life), project economics (AISC and assumed commodity price), project readiness (permitting status), and commodity identity (primary metal category variables).

NPV was considered the most important predictive feature, and therefore its low standalone coverage was not acceptable, motivating the creation of a combined "best available" NPV measure to maximise data availability. Individually, post-tax and pre-tax metrics had limited coverage ($\sim 65\%$); however, by constructing a "best available" NPV and IRR feature that prioritised post-tax values and fell back to pre-tax where necessary, overall coverage increased to 94%. These combined values were then used as the NPV and IRR inputs for the remainder of the project, with an additional binary flag included to indicate whether each observation originated from post-tax or pre-tax reporting.

However, due to the nature of the task, not all features were directly comparable across projects. For example, annual production volumes differ significantly in economic meaning depending on the commodity (e.g., 1,000 kg of gold is not equivalent to 1,000 kg of iron ore). Similarly, cost and price metrics vary by commodity, where an AISC or commodity price of $1,000 per ounce may indicate strong economics for gold but poor economics for iron ore. Absolute NPV values also require context, as a $100 million NPV may be attractive for a company with a $1 billion market capitalisation but represents a substantially stronger outcome for a company valued at $10 million.

To address these comparability issues, a feature engineering pipeline was implemented that standardised units across commodities (e.g., converting all costs, prices, and production figures to consistent per-tonne measures), constructed relative valuation metrics such as NPV-to-market-cap and NPV-to-capex to contextualise project scale, imputed missing commodity price assumptions using the closest available observation for the same metal, and derived operating margin proxies from the difference between commodity prices and AISC. In addition, a 30-day index momentum feature was introduced to capture broader market conditions leading into each DFS announcement.

The final list of features are shown in Table 2:

Table 2: Engineered features and their purpose

Feature	Purpose
npv_to_mktcap_scaled	Project value relative to company size
npv_to_capex_scaled	Investment efficiency
best_irr_pct_scaled	Project return metric
mine_life_years_scaled	Project longevity
index_momentum_30d_scaled	Controls for market conditions
best_npv_aud_scaled	Absolute project value
initial_capex_aud_scaled	Capital intensity
annual_revenue_aud_scaled	Revenue scale
payback_years_scaled	Speed of capital recovery
operating_margin_scaled	Project economic strength
metal_bucket	Commodity classification
npv_flag	Data source indicator (post vs pre-tax)
permitting_status	Regulatory risk proxy
irr_flag	IRR data source indicator
commodity_price_imputed_flag	Tracks imputed price assumptions
primary_metal_major	Major commodity grouping

3.2.4 Preprocessing

Prior to model training, the distributions of all numeric features were examined to inform appropriate preprocessing choices. Given the heterogeneous nature of the variables, which range from large-scale monetary values to ratios and percentage-based measures, different transformations were applied depending on each feature's statistical properties. Table 3 summarises the preprocessing steps applied to each feature and the motivation for each transformation, including log scaling for highly skewed variables, winsorisation to control extreme outliers, and standard or robust scaling to place all features on comparable scales while preserving meaningful variation. Figure 3 in the Appendix shows the results of this preprocessing.

Table 3: Preprocessing applied to numeric features

Feature	Transformation	Reason
best_npv_aud	Log + Robust scaling	Highly right-skewed, spans multiple orders of magnitude
initial_capex_aud	Log + Robust scaling	Large scale variation and heavy tails
annual_revenue_aud	Log + Robust scaling	Strong skewness and extreme values
npv_to_mktcap	Log + Robust scaling	Wide ratio range, negative values possible
npv_to_capex	Winsorise + Log + Robust scaling	Extreme outliers from small denominators
best_irr_pct	Winsorise + Robust scaling	Occasional extreme IRR values
payback_years	Standard scaling	Well-behaved distribution, mild skew
mine_life_years	Standard scaling	Reasonable range, limited outliers
operating_margin	Standard scaling	Bounded ratio, approximately symmetric
index_momentum_30d	Standard scaling	Market returns are well-behaved

3.2.5 Machine Learning Decisions

A diverse set of models was selected to balance interpretability and predictive performance, including OLS and Ridge regression as linear benchmarks, Random Forest to capture nonlinear relationships, XGBoost for regularised gradient boosting, and an ensemble model is constructed by ranking all candidate models by their test-set R^2 performance and averaging the predictions of the top three performers, providing a data-driven selection mechanism that improves robustness and reduces prediction variance without introducing subjective weighting. This range enables meaningful comparison between linear and nonlinear approaches while improving robustness through model averaging. To ensure realistic evaluation, a time-based train/test split was used, with the earliest 80% of events allocated for training and the most recent 20% reserved for testing. This approach avoids look-ahead bias and reflects real-world forecasting conditions, unlike random splitting which could allow future information to leak into the training process.

3.2.6 Evaluation Metrics

Model performance was assessed using a combination of error-based and decision-focused metrics to capture both statistical accuracy and real-world usefulness. Traditional measures such as R^2, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) were included to evaluate how closely predictions matched actual returns and to quantify average and extreme prediction errors. The R^2 metric provides insight into how much of the variation in stock returns the model can explain, while MAE reflects typical prediction mistakes and RMSE penalises larger errors more heavily, making it sensitive to major mispredictions. In addition, directional accuracy was emphasised because it directly aligns with investment decision-making, as correctly predicting whether a stock will rise or fall is often more important than estimating the exact return magnitude. Together, these metrics provide a balanced evaluation framework that considers both statistical performance and practical trading relevance.

3.3 Implementation

3.3.1 Software Environment

All experiments were implemented in Python using a range of open-source scientific computing and machine learning libraries. Key dependencies included `pandas` and `NumPy` for data processing, `yfinance` for downloading historical market data, `scikit-learn` for preprocessing and model evaluation, `XGBoost` for gradient boosting regression, `SciPy` for statistical transformations, `matplotlib` for visualisation, and `openpyxl` for exporting results to Excel. This software ecosystem was chosen for its maturity, reproducibility, and widespread adoption within quantitative finance research.

3.3.2 System Architecture

Input Data → Benchmark Models → Feature Engineering → Market Data Integration → Preprocessing → Model Training → Ensemble Evaluation → Results Export

Each stage in the pipeline is implemented as an independent functional module, improving maintainability and allowing for easy extension of the framework. Intermediate outputs are persistently stored to support debugging and ensure full reproducibility of results. To improve runtime efficiency, a disk-based caching system is employed. Input Excel files are cached using file fingerprints, while downloaded market data are cached by ticker and date range. This approach minimises repeated API calls and enables approximately 3–5× speed improvements on subsequent executions.

3.3.3 Key Code Components

A selection of key code components are displayed below.

Information Leakage Prevention: This code implements a time-based train/test split by ordering all events chronologically and assigning the earliest 80% of observations to the training set and the most recent 20% to the test set. This design prevents look-ahead bias by ensuring that future information is never used to train the model.

```python
pivot_df = pivot_df.sort_values(["announcement_date", "row_id"]).reset_index(
    drop=True)

n_total = len(pivot_df)
n_test = int(np.ceil(n_total * TEST_FRAC))
n_train = n_total - n_test

pivot_df["split"] = np.where(pivot_df.index < n_train, "train", "test")
```

Preprocessing: The preprocessing pipeline operates directly on individual features. For example, the *best_npv_aud* variable is first log-transformed to reduce right-skewness, then scaled using a robust scaler fitted exclusively on the training set. The learned scaling parameters are subsequently applied to the test set, ensuring no information leakage.

```python
# Select feature
feature = "best_npv_aud"

# Split data
train_df = ml_df[ml_df["split"] == "train"]
test_df = ml_df[ml_df["split"] == "test"]

# Apply log transform to NPV
train_log = np.log1p(train_df[feature])
test_log = np.log1p(test_df[feature])

# Fit scaler on TRAIN only
scaler = RobustScaler()
scaler.fit(train_log.values.reshape(-1, 1))

# Apply frozen scaler
train_scaled = scaler.transform(train_log.values.reshape(-1, 1))
test_scaled = scaler.transform(test_log.values.reshape(-1, 1))
```

Model Training: This code trains each model on the training data and then generates predictions on the test set for performance evaluation.

```python
models = {
    'Random Forest': RandomForestRegressor(**RF_PARAMS),
    'XGBoost': XGBRegressor(**XGB_PARAMS),
    'Ridge': Ridge(**RIDGE_PARAMS),
    'OLS': LinearRegression()
}

for name, model in models.items():
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
```

Ensemble Selection Logic: This logic ranks models by test R^2, selects the top three performers, and averages their predictions to produce a more robust ensemble forecast.

```python
model_test_r2 = [
    (name, results[name]['test_metrics']['R2'])
    for name in models.keys()
]

model_test_r2.sort(key=lambda x: x[1], reverse=True)
top_3_models = [name for name, _ in model_test_r2[:3]]

ensemble_pred = np.mean(
    [predictions[m]['test'] for m in top_3_models], axis=0
)
```

4 Results

4.1 Experimental Setup

Dataset: 220 observations (147 train, 45 test), temporally split. Features: 10 scaled continuous (NPV/market cap, NPV/CAPEX, IRR, mine life, index momentum) + 3 categorical (metal bucket, NPV flag, permitting status).
Models: Random Forest (n=30, depth=4), XGBoost (n=40, depth=4, lr=0.10), Ridge (α=5.0), OLS, Ensemble (Top-3 average). Baselines: Constant Mean, Naive Past Mean.
Targets: Raw returns (30-day absolute), Abnormal returns (30-day market-adjusted).
Metrics: R^2, RMSE, MAE, Directional Accuracy (% correct sign), R^2 Gap (train - test, target <0.25).

4.2 Performance Evaluation

For this section, only raw returns will be analyzed. The models provided similar performance to abnormal returns, and the abnormal returns graphs can be found in the appendix. Table 4 presents the composite ranking of all models based on a weighted scoring scheme placing 40% weight on directional accuracy and 20% each on R^2, MAE, and RMSE.

Table 4: Model Performance Ranking (Test Set)

Model	Avg R^2	Direction (%)	MAE	RMSE	Gap
OLS	0.072	73.3	0.197	0.235	0.143
Ridge	0.033	73.3	0.202	0.240	0.173
Ensemble	0.031	70.0	0.204	0.240	0.163
Random Forest	-0.111	72.2	0.218	0.257	0.246
XGBoost	-0.096	66.7	0.219	0.255	0.155

OLS delivered the highest explanatory power with a test R^2 of 0.072, the lowest error metrics, and the smallest train–test performance gap, indicating excellent generalization. Ridge regression ranked second and produced nearly identical directional accuracy. The ensemble model failed to outperform its strongest constituent model. Both Random Forest and XGBoost produced negative test R^2, performing worse than a constant mean benchmark.

4.2.1 Model Fit and Overfitting Diagnostics

Figure 1 shows model fit and overfitting diagnostics for raw returns. Linear models achieved positive test R^2 values, while tree-based models produced negative out-of-sample performance. OLS delivered the strongest results, indicating superior generalization.

The train–test R^2 gap (right panel) remained below 0.25 for all models, suggesting acceptable generalization, though Random Forest approached this limit, indicating mild overfitting. XGBoost showed a smaller gap, reflecting stronger regularization but weaker predictive power.

Overall, increased model complexity did not improve performance. Simpler linear models achieved higher accuracy and stability, implying a largely linear relationship between DFS fundamentals and post-announcement returns.

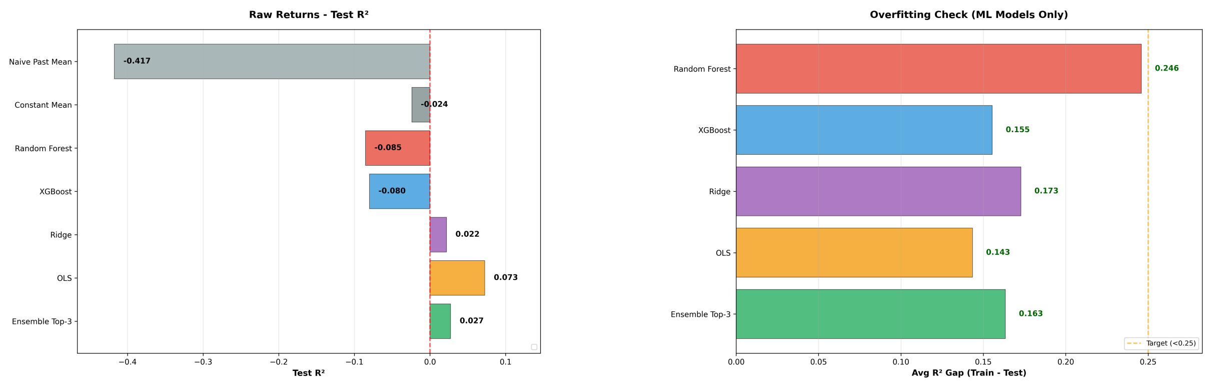

Figure 1: Model fit and overfitting diagnostics for raw returns. Left: test R^2. Right: train–test R^2 gap.

4.2.2 Prediction Analysis

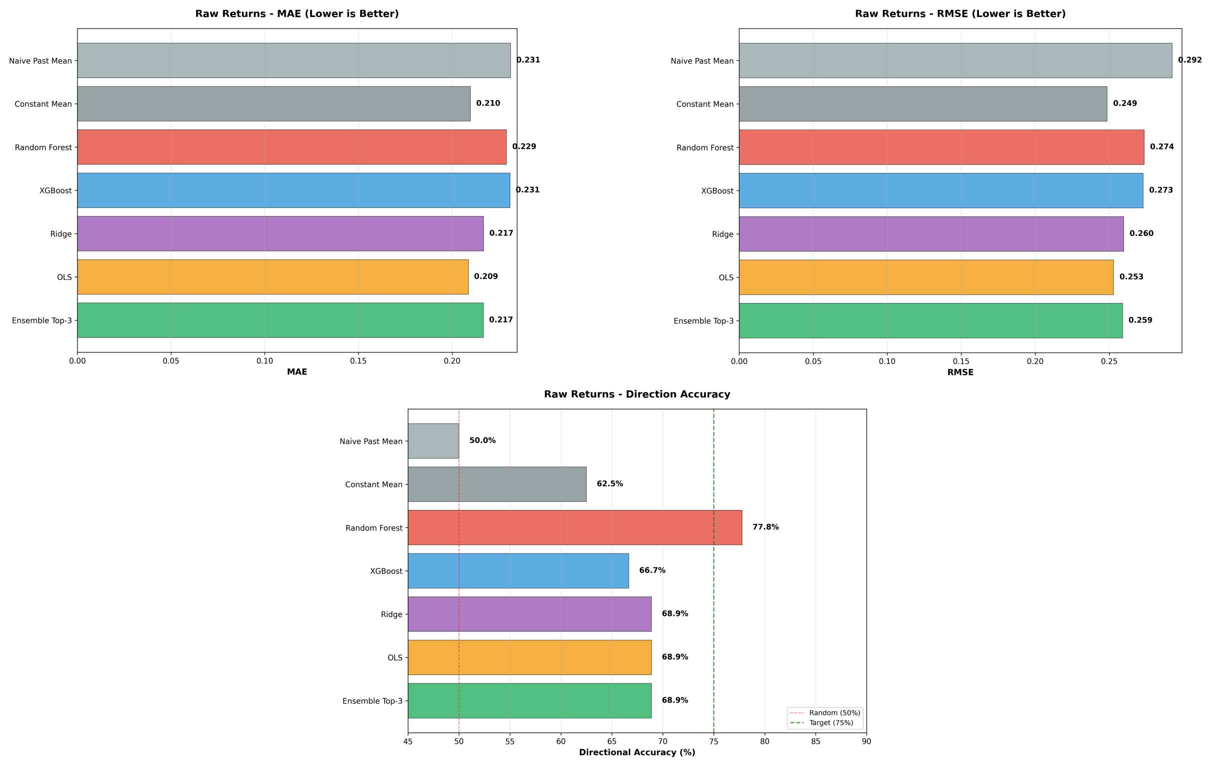

Figure 2: Prediction error metrics for raw returns. Left: Mean Absolute Error (MAE). Middle: Root Mean Squared Error (RMSE). Right: Directional accuracy. Lower error values indicate better performance.

Figure 2 presents MAE and RMSE for all models on raw returns. Ordinary Least Squares (OLS) achieved the lowest error across both metrics, confirming its superior predictive accuracy. Ridge regression and the ensemble model produced comparable performance, while Random Forest and XGBoost exhibited larger errors, indicating weaker out-of-sample reliability.

The consistency between MAE and RMSE rankings suggests that model errors are stable and not driven by extreme outliers. These results further reinforce the conclusion that simpler linear models provide more robust predictions in this setting.

All models substantially outperformed random chance (50%) and naive baselines (62.5%). For abnormal returns, OLS and Ridge achieved 77.8% directional accuracy, exceeding the project target.

5 Discussion

The most significant finding of this study is the strong directional predictive performance achieved across models. Directional accuracy ranged between 70% and 78%, substantially exceeding random chance and naive benchmarks. In particular, OLS and Ridge achieved a 77.8% success rate in predicting abnormal return direction, corresponding to a 1.56:1 ratio of winning to losing trades. This level of accuracy suggests meaningful economic value for long–short trading strategies when combined with appropriate risk management.

Contrary to expectations based on prior literature, simple linear models outperformed more sophisticated machine learning approaches. OLS delivered the highest R^2, lowest error metrics, and strongest generalization. This indicates that the relationship between DFS fundamentals and post-announcement returns is approximately linear. The small training sample (n=147) likely limited the ability of tree-based models to learn stable nonlinear patterns, leading to spurious correlations and overfitting.

While directional accuracy was strong, explanatory power remained low, with best R^2 values around 0.07. This aligns with established financial literature showing that individual stock returns are dominated by noise and macroeconomic shocks. The low R^2 supports semi-strong market efficiency, indicating that most public DFS information is rapidly incorporated into prices.

The ensemble model failed to improve upon OLS performance, likely because weaker models diluted stronger predictions and because model errors were correlated. Simple averaging therefore reduced signal strength rather than enhancing robustness.

Several limitations remain. The dataset is relatively small and temporally heterogeneous. Only a single train–test split was used. Feature coverage remains incomplete due to inconsistent DFS reporting, and important drivers such as management quality, jurisdictional risk, commodity price momentum, and market sentiment were not included. Binary directional accuracy also treats small and large returns equally, oversimplifying economic impact.

6 Conclusion and Future Work

This project developed an end-to-end machine learning pipeline to forecast post-announcement stock returns following DFS releases. The system successfully extracted structured financial data from unstructured PDFs, engineered economically meaningful features, and implemented robust preprocessing and evaluation procedures.

The results demonstrate that DFS fundamentals contain incremental predictive signal. While predicting return magnitude remains challenging, achieving up to 77.8% directional accuracy provides strong evidence of exploitable patterns. The most important insight is that model simplicity outperformed complexity, with OLS emerging as the optimal model.

Future work will focus on expanding the data set, walk-forward validation, rolling retraining, and weighted ensemble methods. Additional experiments will test alternative prediction horizons and binary classification frameworks. Feature expansion will incorporate NLP-based sentiment analysis of DFS documents, commodity price momentum, liquidity metrics, and volatility measures.

From a practical perspective, the model can be deployed as a signal-generation tool for long–short strategies. With appropriate risk management, this framework provides a scalable decision-support system for investors evaluating mining feasibility announcements.

References

Ball, R., & Brown, P. (2013). An empirical evaluation of accounting income numbers. In *Financial accounting and equity markets* (pp. 27–46). Routledge.

Bernard, V. L., & Thomas, J. K. (1989). Post-earnings-announcement drift: Delayed price response or risk premium? *Journal of Accounting Research*, *27*, 1–36.

Bird, R., Grosse, M., & Yeung, D. (2013). The market response to exploration, resource and reserve announcements by mining companies: Australian data. *Australian Journal of Management*, *38*(2), 311–331.

Budennyy, S., Kazakov, A., Kovtun, E., & Zhukov, L. (2023). New drugs and stock market: A machine learning framework for predicting pharma market reaction to clinical trial announcements. *Scientific Reports*, *13*(1), 12817.

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

Kim, H. (2021). Mean-variance portfolio optimization with stock return prediction using xgboost. *Economic Computation & Economic Cybernetics Studies & Research*, *55*(4).

Rathi, V., Kshirsagar, M., & Ryan, C. (2024). Enhancing portfolio performance: A random forest approach to volatility prediction and optimization. *Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART), Volume 3*, 1278–1285.

Sarisoy, C., de Goeij, P., & Werker, B. J. M. (2016). *Linear factor models and the estimation of expected returns* (Netspar Academic Paper No. DP 03/2016-020). NETSPAR.

Ye, Z. J., & Schuller, B. W. (2021). Capturing dynamics of post-earnings-announcement drift using a genetic algorithm-optimized xgboost. *Expert Systems with Applications*, *177*, 114892.

AI Usage Disclosure

Literature Review

Perplexity was used to assist with the literature review process, specifically for identifying relevant academic sources and papers related to the topic. ChatGPT and Perplexity were also used to help locate studies related to Daily Fantasy Sports (DFS).

Writing Support

ChatGPT was used to assist with:

- Improving spelling and grammar
- Condensing content to meet page limits
- Rephrasing sentences for clarity
- Navigating LaTeX formatting and bibliography management

Coding Support

Claude and ChatGPT were used to assist with:

- Debugging code
- Code review and improvement suggestions
- Modularisation and simplification of code
- Understanding and using external libraries

A Code Repository

A.1 Repository Structure

Key directories and files:

- `main.py` – Main execution script
- `src/` – Core source code
- `data/raw/` – Input data location
- `requirements.txt`, `environment.yml` – Dependency management
- `README.md` – Project documentation

A.2 Installation

pip:

```
git clone https://github.com/collma18/dfs-ml-pipeline
cd dfs-ml-pipeline
pip install -r requirements.txt
```

conda (optional):

```
conda env create -f environment.yml
conda activate <env-name>
```

A.3 Reproducing Results

run:

```
python main.py
```

The pipeline cleans data, downloads stock prices, constructs event windows, engineers features, trains multiple ML models, and outputs results to:

- `ml_output.xlsx`
- `model_results_FINAL_v2.xlsx`
- `PNG charts (feature distributions, model comparisons)`

Models include Random Forest, XGBoost, Ridge, Linear Regression, and an ensemble. Evaluation uses R^2, MAE, RMSE, and directional accuracy.

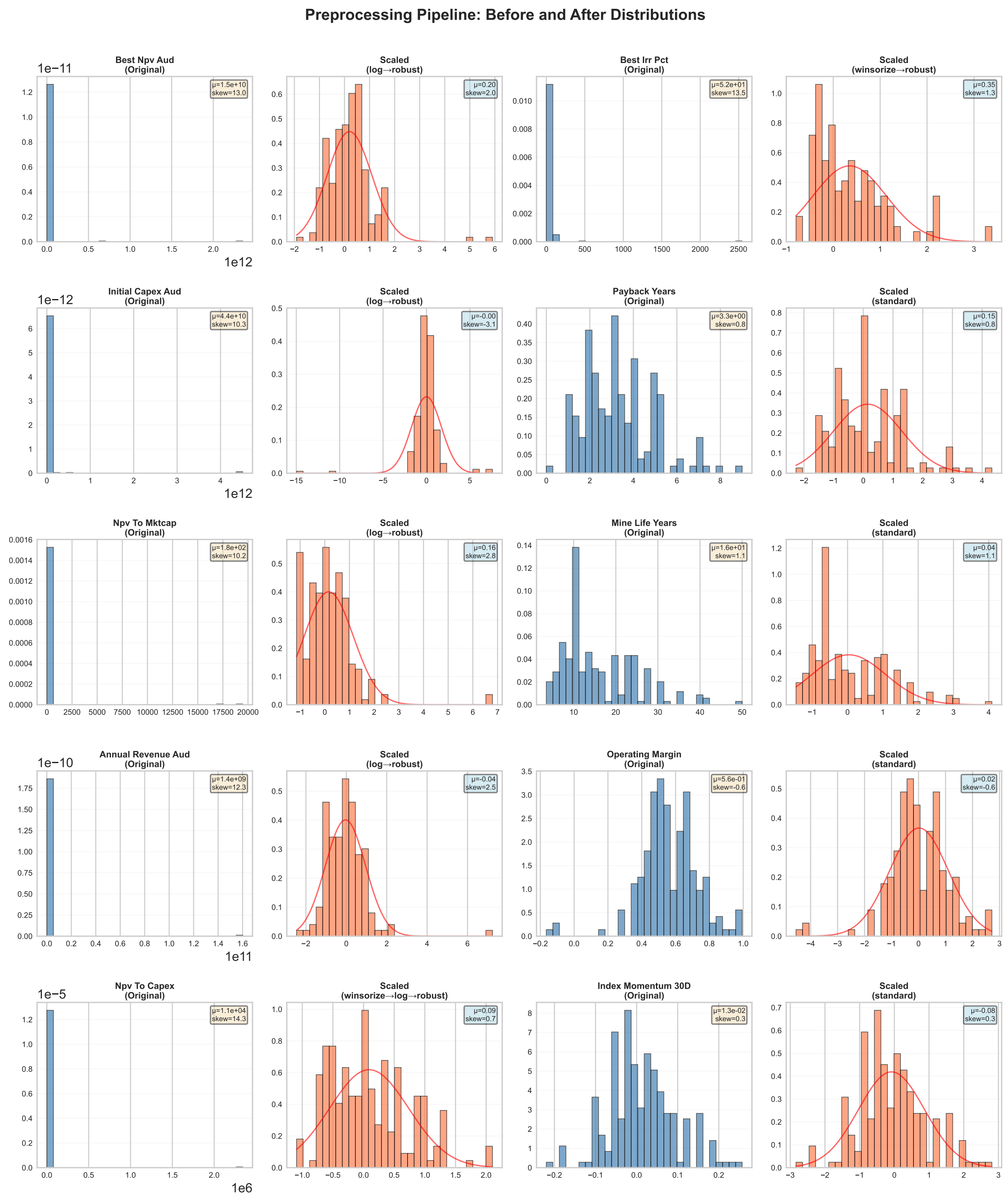

Figure 3: Preprocessing transformations for all features

B Additional Figures

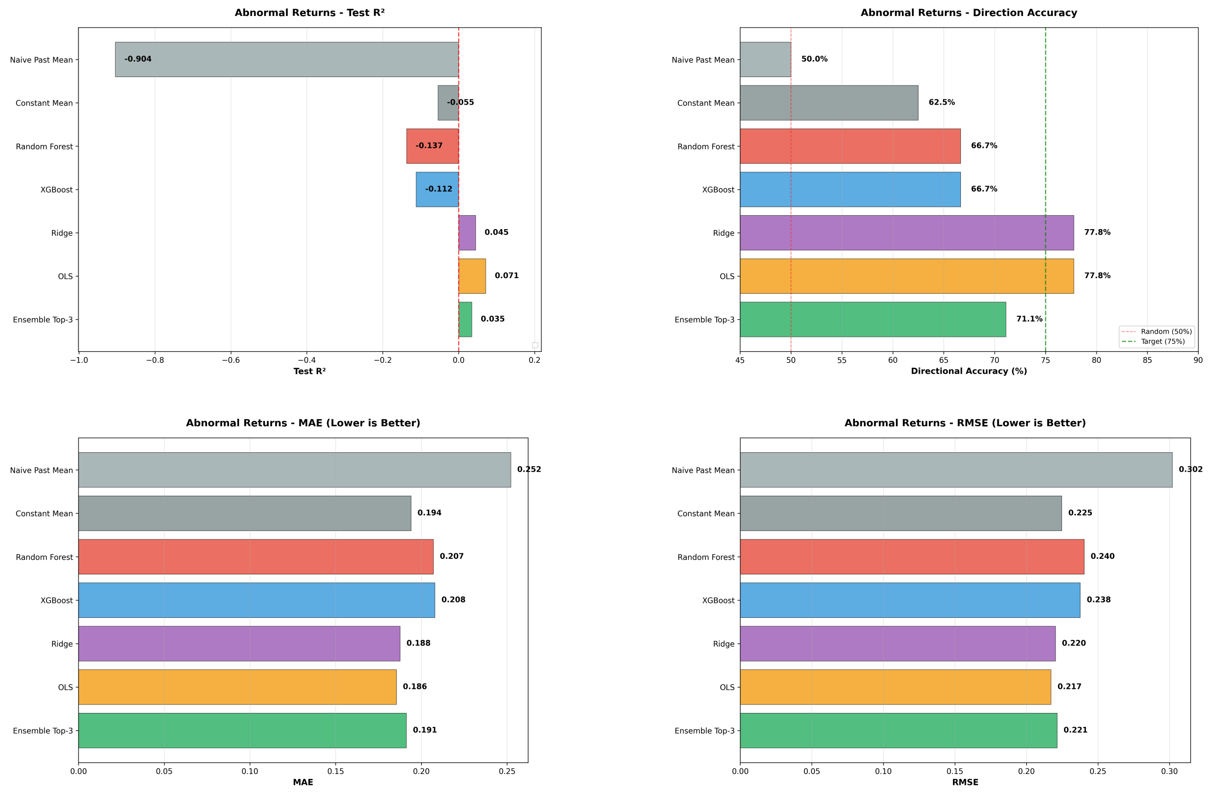

Figure 4: Model performance metrics for abnormal returns. Top left: R^2. Top right: Directional accuracy. Bottom left: Mean Absolute Error (MAE). Bottom right: Root Mean Squared Error (RMSE).