Predicting Stock Market Reactions to Earnings Surprises

1. Abstract

This project explores the prediction of stock market reactions to earnings surprises by analysing the relationship between actual earnings and analysts' forecasts. The study employs OLS quantiles analysis and Random Forest models.

The results show that including earnings surprises (surpriseP) in models improves prediction accuracy. The "EA with SurpriseP with Variables" model provides the best performance, emphasizing the importance of earnings surprises in forecasting returns. Despite some limitations and size of the sample, the findings offer insights for investors, while needing improvement of the models.

2. Introduction

When companies announce their earnings, stock prices can rise or fall, especially if the earnings differ from what was expected. This project examines the possibility of predicting stock price movements following these announcements. By analysing the relationship between actual earnings and analysts' forecasts, we aim to develop a predictive model using historical earnings and stock price data. My goal is to create a model that helps investors make informed decisions and better understand the impact of earnings surprises on stock returns.

To achieve this, we will use OLS, quantiles analysis and Random Forest model, to identify patterns that can predict future stock price movements after earnings announcements. This project focuses on simple yet effective techniques. By the end of this research, we hope to provide a clear understanding of how earnings surprises affect stock returns and deliver a practical model for predicting these effects. This work could help investors get a better idea of how stock prices might move, which is useful for making smart investment choices.

3. Description of the research

The central inquiry of this research revolves around the predictive relationship between earnings surprises and stock returns. We define an earnings surprise as the difference between the actual earnings a company reports and the earnings that analysts expected. Our research seeks to answer the following questions: Can the magnitude of earnings surprises serve as a reliable indicator for predicting stock returns in the days following the announcement? If so, by how much do these surprises influence stock returns?

In pursuing these questions, we will consider several financial variables that could interact with earnings surprises to affect stock returns. These include market value, leverage, and other commonly analysed factors in financial studies, providing additional context to the earnings surprise event.

While the detailed data handling and processing will be elaborated in the methodology section, it is important to note here that we will be examining historical data for a range of

companies over several years. This data encompasses not only earnings surprises and stock returns but also other financial indicators. By comprehensively analysing this historical data, we aim to identify patterns and develop a robust predictive model.

This research is an idea that emanates from the work that I do as an assistant for Professor Markarian at the University of Lausanne.

4. Description of the data set

The most difficult part of this project was handling the data set. Reading large SAS files, dealing with computer freezing issues, calculating lead returns, and merging the data sets were all very challenging. But I think this is common when preparing data; knowing what we are looking for is the hardest part. Once that is clear, setting up a model and running it becomes easier. The data comes from the Center for Research in Security Prices (CRSP).

CRSP File

At first there is one initial SAS file with more than 20GB of data, compiling companies' prices and other variables of the SNP500 per date. On the side there is a dividend file.

With those, I calculated returns with dividends and without, among others and compiled the results into a SAS file *"daynight_returns"*. I selected only a few variables for this file, such as day and night ret including permno (company identifier), date, open price, close price, and returns and others.

With these results, I calculated then the lag and lead returns for both after-hours (night) and market-hours (day). The returns included dividends because I thought it would be more interesting to include them. I calculated up to 5 lag and lead returns.

To calculate "**day_return**" including dividends, I used the formula as follow:

$$\frac{price_{divdiend} - open}{open}$$

For the **"night_return"** including dividends, we shift the price dividend by one day. This way, we can measure the return from the closing price of yesterday to the open price of today. The formula for after-hours returns is:

$$\frac{open - lag\_price_{divdiend}}{lag\_price_{divdiend}}$$

In the following figure we can the timing of the interesting variables.

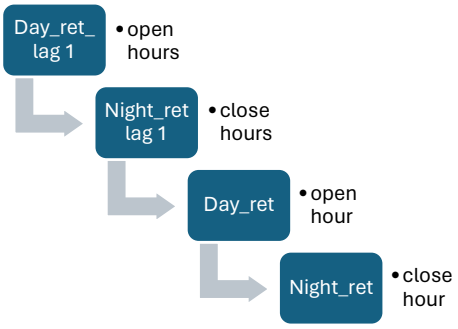


Figure 1

We don't need to recalculate the night_ret_lag2, for example, because we can obtain it by calculating the night return and shifting by one period. The same applies to the lead returns, as the return of tomorrow becomes the lead return of today, allowing us to shift backwards to obtain today's lead return.

We do all this for each permno (company) because the returns of one company are logically different from those of another.

For reading a SAS file, it is straightforward because pandas has a built-in function called pd. read_sas. However, since the file is very large, I had to read it in chunks, with each chunk containing 50,000 rows. For better memory efficiency, I converted the variables from float64 to float32, as the file is already quite big.

After processing, the shape of the file was (48'767'214 rows, 28 columns). So millions of rows. To make it more manageable for the computer, I decided to simply drop all rows

with NaN values, reducing the shape to (48,518,316 rows, 28 columns). This reduction is less than one percent, so replacing 1% of NaN values is not significant and that is why I proceed so.

Earning announcement file

The second file is also a SAS file containing the dates of earnings announcements, which occur quarterly. This file includes several important variables:

-EADate: The date of the earnings announcement.

-surpriseP: the quarterly earnings announcement minus the median analyst forecast, normalized by the stock price

-mkvval: The market value of the firm, which provides insight into the size of the company and its stock's liquidity.

-mb: The market-to-book ratio, which can indicate whether a stock is undervalued or overvalued.

-leverage: Total debt over total assets, reflecting the company's financial risk.

-cal_qtr & **cal_yr**: are the quarter and the year.

- ffindustry: sectors based on FAMA factors

Merge

I merged the two databases as follows:

```
merged_data = pd.merge(daynight_clean, ea_data, how='left', left_on=['date', 'permno'], right_on=['EAdate', 'permno'])
```

This merge used the columns date and permno from the first file, and EAdate and permno from the second file. After trying different methods, I found that a left join worked best. To manage zero values, I removed all rows containing zeros to maintain data consistency. Although zeros could represent non-trading days and might be useful for calculations, I chose this approach for simplicity and to handle the large file size. This reduced the final dataset to (13,146,884 rows, 28 columns), cutting the initial size by 75%.

It's important to note that I could have used forward or backward filling or other imputation methods. But as the files were very big and I but faced memory issues and computational power. I decided to proceed like this. For better-quality research, it would be better to keep these values and consider other ways to process very big amounts of data like Prophet or others for handling missing data.

To further reduce the dataset size, I took only the first 100 rows per permno. This made the data more manageable and easier to merge with the other dataset. Additionally, I reset the index to facilitate column comparison in Python, which made the process simpler.

I considered removing outliers, especially in the surpriseP column, where 99% of the values are around 0.00x, but some outliers have a surpriseP of 25. However, I decided to keep these outliers in the model to ensure it predicts a more realistic view. Including outliers helps the model handle both regular performance and extreme events, or black swans.

5. The Methodology applied

OLS model

First, I ran the simplest model to establish a basic framework. I initially performed some OLS regressions to calculate lead_returns1 and night returns. Given that announcements typically occur just before the closing hours (around 4 PM) or during closing hours, I wanted to ensure that the impact caused by the most recent values was captured, as these are likely the most influential ones. The initial models did not include the earning surpriseP values, aiming

to explain lead_return1 using all the lag variables as independent variables along with day and night returns (day 0).

Without surpriseP

In the first model (without surpirseP), we can see that the regression (night_return) has a higher R-squared (0.017) compared to the first regression (lead_1_day_ret) with an R-squared of 0.002. This indicates that the second one explains a larger portion of the variability in the dependent variable compared to the first model. The coefficients in both models are statistically significant. In the first model, the coefficient for night_return is -0.0465, indicating a stronger negative relationship compared to other variables. In the second, day_return has a coefficient of -0.0698, which is also quite substantial. We can see that the period before is always more impactful.

The standard errors being very small (almost zero) might indicate an issue in the model. This can occur due collinearity issues or other.

Including surpriseP

Next, I included the earnings data meaning, the surpriseP varaible for both regressions. The impact of surpriseP on night return was much more significant than on lag_1_day_ret, confirming that the effect of announcements made before or during closing hours is more pronounced. However, this does not explain why there is practically no effect on lead 1.

The inclusion of surpriseP slightly improved the R-squared values in both models, suggesting that surpriseP adds some explanatory power, though the increase is modest. In the lead_1_day_ret model, surpriseP appeared to be significant, improving the model's fit. In the night_return model, surpriseP was highly significant with a positive coefficient (0.0288), indicating that surprises in analyst predictions are associated with higher night returns.

Please note that the number of observations is drastically reduced. It goes from 14506854 to 102284 so we reduced the data base by 99%. The smaller sample size in these models compared to the initial dataset might have led to more precise standard error estimates. Including surpriseP may have stabilized these estimates. But it is a big issue for the research, and I didn't realize until very late of the project.

After including surpriseP, both models showed improved R-squared values, but the night_return model still had relatively higher explanatory power. The inclusion of surpriseP significantly improved the fit of both models, making it a valuable predictor of returns. Its effect was particularly notable in the night_return model, although the model still explained a small portion of the variance.

Including variables on top of surpriseP

Then I added some variables to the night_return model. However, the R-squared remained the same, and these variables did not add much explanatory power. Only cal_qtr and leverage were significant at the 0.05 level.

Only negative and positive surpriseP

Before changing the model, I wanted to confirm if negative or positive announcements were more impactful on night returns. I split the data into positive and negative surpriseP. The results showed that positive surpriseP affects night_return positively and vice versa. This is logical, but the R-squared remained low, suggesting that we should change the model.

Interestingly, surpriseP with only positive impacts was more significant than when analyzing only for negative surpriseP. This suggests that the magnitude of negative surprises might not be a strong predictor of night returns. Maybe this confirm the economic theory stating that we tend to be more impacted by negative news than by positive news.

Quartiles analysis

Then I analyse the quartiles of positive and negative surpriseP (see in annexes). The variance of the quantiles is higher when we are dealing only with negative surpriseP (figure 2).

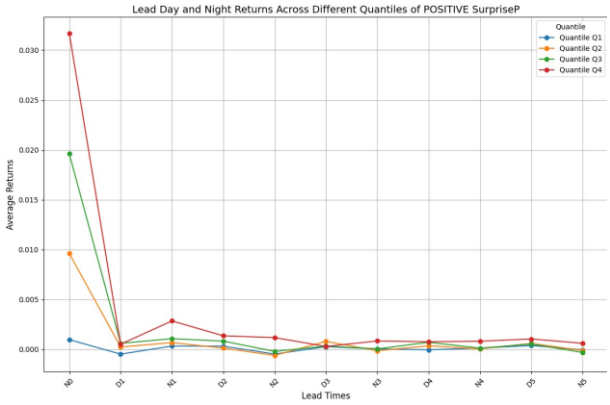


Figure 2

We can see that the quartiles tend to stabilize them after Night 0 (N0). I chose then to predict only night_return and lead_1_day_ret and, because the time to run the random forest model for the non-earnings was already too long to include more variables. I realized until very late that earnings data was drastically reduce. I could have included more lead day and night and have a more rich and precise research.

Random Forest model

Given the complexity and non-linearity of financial market data, I choose to use Random fores model. It is well-suited for such data because it can capture non-linear interactions without requiring data transformation.

Random Forest operates by constructing multiple decision trees during training and outputting the mean prediction of the individual trees. This method handles non-linearity effectively. Unlike OLS, which poorly explains variability in linear models, Random Forest is robust to outliers. By averaging multiple trees, it is less sensitive to outliers compared to single decision trees.

Model setup

I first compared the effects of non-earnings and earnings periods to isolate the effect of surpriseP.

NO Earnings model**:** This model excludes earnings announcements to capture regular market dynamics without the noise from SurpriseP. Initially, the data was too large for my computer, taking 10 hours without success.

To manage this, I used Principal Component Analysis (PCA), a dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional form while retaining as much original variance as possible. Although some information was lost, PCA reduced the matrix size, allowing the model to run in 200 minutes. The MSE with PCA was 0.0009441927216915945.

Next, I compared these results with a model using arbitrary parameters to fine-tune later. Here are the parameters for my random forest model:

- n_estimators=100: Sets the number of trees in the forest. I chose 100 to balance accuracy and training time.

- max_depth=10: Limits the maximum depth of each tree to prevent overfitting and reduce complexity.

- max_features='sqrt': Uses the square root of the total number of features at each split to make the model faster and less prone to overfitting.

- min_samples_split=5: Ensures nodes are split only if they have at least 5 samples, preventing the model from learning noise in small splits.

- min_samples_leaf=4: Sets the minimum number of samples at a leaf node to 4, promoting better generalization.

- random_state=42: Ensures reproducibility of the results.

- test_sample=0.2: Sets aside 20% of the data for testing.

The MSE with the arbitrary parameters was 0.00082773832043384, lower than with PCA, indicating better performance. Therefore, I proceeded with further analysis and performed a Grid Search for optimal parameters and to optimize the model parameters.

It is called hyperparameter tuning, and it helps reduce overfitting and underfitting by selecting the best parameters for the model.

Overfitting occurs when a model learns the training data too well, capturing noise and fluctuations, leading to high accuracy on training data but poor generalization to new data.

Underfitting happens when a model is too simple to capture the underlying patterns in the data, resulting in poor performance on both training and test data.

By using GridSearchCV, I aimed to find a balance between overfitting and underfitting, ensuring the model generalizes well to unseen data. While I could have used RandomizedSearchCV, I chose GridSearchCV for simplicity and consistency for the models.

After optimizing the parameters for the earnings-announcement model without surpriseP, I compared the MSE of both models (with and without earnings). This allows me to better isolate the effect of surpriseP.

I then calculated different models to isolate the effect of surpriseP further. For these models, I used the same hyperparameters as the earnings model without surpriseP for consistency and computational efficiency.

The other models are: “EA with SurpriseP” with Variables to better isolate the effect of surpriseP.

Finally, I compared the results of models with only positive surprises and negative surprises to observe the differences between them “EA POSITVE and NEGATIVE”.

I hyperthuned the “NO EARNINGS” model, then the “EA without surpirseP”. And I kept the same parameters of the “EA without surpriseP”. I finally properly hyperthuned the parameters for the positive earnings model to better predict positive announcement.

The models were assessed using Mean Squared Error (MSE) and R-squared (R^2) metrics, both for the test set and through cross-validation. Additionally, feature importance analysis provided insights into which variables most significantly influenced the predictions.

6. Implementation of the code

The main code for all the research is “Main..” that extracts the results from the other Annexes file and calculate the lead and lag returns. There is the secondary file to calculate the different returns and compile them into the results file.

I used Jupyter Notebook for its interactive and user-friendly environment when dealing with data as opposite of the Advanced Programming Project where I simply use python files. The libraries used included pandas and numpy for data manipulation, scikit-learn for machine learning models, and matplotlib for plotting.

To manage the extensive data processing and model training tasks, the code was designed to handle large datasets efficiently, using techniques like chunk processing and dimensionality reduction using PCA. And a Random Forest Model

Version control was maintained using GitHub, which facilitated tracking changes in the codebase. The code was committed and pushed to a private repository using GitHub Desktop.

7. Results

Cross-validation

The **NO EARNINGS** model, serving as the baseline, shows a reasonable cross validated MSE of 0. 0.002164914 and R^2 of 0.033530923, so it explains 3% of the model which is not that great This model does not include any earnings-related variables.

The **EA without SurpriseP**, don't improve anything at all, but this is also kind of a baseline too, the cross-validated MSE increases a bit and it it indicates that the model's predictions are less accurate. The cross-validated R^2 decreases significantly. The model explains almost none of the variance in returns, suggesting that earnings-related variables without surpriseP do not add predictive power.

The **EA with SurpriseP** model, drastically improves performance. It lowers the cross-validated MSE compared to previous model. This shows that predictions are more accurate when surpriseP is included. Also, it shows a significantly higher cross validated R^2: The model explains about 6.4% of the variance in returns, a substantial improvement over the other models. This indicates that surpriseP is a good predictor for returns. But be aware that the data is very small compared to the baseline.

Then the best model is the **EA with SurpriseP with Variables**. The cross-validated MSE is 0.002272445 very similar as the one without the variables but it has the best R2 (7%). It is better than one without variables but not that much. The results seem not that great but for financial research I would suggest that they can be interesting. Because it is relatively hard to predict returns overall.

Between the **EA Negative and EA positive with SurpriseP with Variables**, they have a similar cross-validated R2 but the positive performs better in terms of cross-validated MSE. Suggesting that positive earnings surprises are generally easier to predict due to more consistent market reactions. Investors tend to respond predictably to good news, leading to a clearer pattern in the data.

The **Hypertuned EA Positive** performed almost identical as the positive EA meaning that hypertuning this model was not that impactful.

The figure 4 (see annexes) shows the comparison across the models, including the MSE and R2 and after the cross validation. "EA with SurpriseP with Variables" seems the best.

Feature importance analysis

Random Forest, feature importance is often calculated based on the mean decrease in impurity (Gini impurity or entropy) brought by each feature across all trees in the forest. It allows to classify the pertinence and importance of the features.

EA with SurpriseP: The surpriseP feature is highly significant, along with other variables like lag_1_day_ret. And adding features such as leverage, ffindustry, and cal_qtr plays substantial roles but doesn't change the lead importance of surpirseP.

The **interesting part** is when comparing the features importance the positive and negative surpirseP from the features of all the earning with variables model. The surpriseP represents 0.6 initially.

In the **EA Positive surpriseP**, even if the surpirseP remains crucial, the importance is lowered by almost half. All the others feature variables stay relatively the same.

But in **EA Negative** the surpirseP feature importance is drastically lowered, by around 85%. The most important features become daynight with 0,25 point of importance, and surpriseP comes after mkval and lag1 day ret with 0,10 points of importance. Meaning that

the variables have a different impact when there are negative announcements indicating more variability and less predictable patterns.

Finally, I wanted to calculate the effect of the effect of surpriseP on returs with a simplified analysis, as it doesn't account for different interactions, calculated as follow:

$$CR(MSE)_{with\ SurpriseP} - CR(MSE)_{without\ surpriseP}$$

$$= 0.002292251 - 0.002501511$$

$$= -0.00020926$$

By comparing the Cross-Validated Mean Squared Errors (CR(MSE)), we observe that including surpriseP reduces the CR(MSE) by approximately 0.00020926. This reduction quantifies the positive impact of surpriseP on the model's predictive accuracy. The difference represents 9.1% of the CR(MSE) with surpriseP, showing a impact. However, this may not be the best method to measure the impact but offers a simple way.

Critiques of the research

I tried to separated night return and lead return to analyse their effects separately, but this didn't significantly change the model. The R2 value remained low, indicating that the separation wasn't the issue.

Then I compare the actual to predicted values (see annexes). For all models look almost the same and don't explain much, likely due to the low R2 values or the dispersion of the data.

Despite the (MSE) and CR(MSE) being appealing, the low R2 across all models suggests that the OLS and the random forest model might not capture all relationships, indicating that the target variables are challenging to predict. The main issues in the research are, I think, multicollinearity, as explaining returns over returns is difficult, especially in short time periods like this case.

Additionally, removing outliers, might have had a significant impact even if Random Forest is robust to outliers. As for the smaller sample size after including only earnings data. It reduced the number of observations drastically, which likely affected the model's performance.

But overall, even though the R2 is low, it doesn't mean the model is bad. The inclusion of surpriseP significantly enhances the model's predictive power. The results demonstrate that surpriseP is an important variable for predicting night_return and lead_1_day_ret. This indicates that earnings surprises can provide good information that captures market reactions.

8. Conclusion

This project demonstrates that the magnitude of earnings surprises (surpriseP) can serve as a reliable indicator for predicting stock returns in the days following an announcement. The inclusion of earnings surprises in predictive models significantly enhances their accuracy, as evidenced by the "EA with SurpriseP with Variables" model.

The findings suggest that positive earnings surprises are generally easier to predict due to more consistent market reactions, leading to clearer patterns in the data. Investors tend to respond predictably to good news, which helps in making more accurate predictions.

However, the models still have room for improvement to capture the all the complexity. Future work could involve removing outliers, using predictive models like LSTM or Gradient Boosting, and addressing multicollinearity.

Overall, the insights gained from this model can aid in making informed trading decisions. While further refinement and testing are necessary, the incorporation of earnings surprises provides a valuable tool for predicting stock market behaviour.

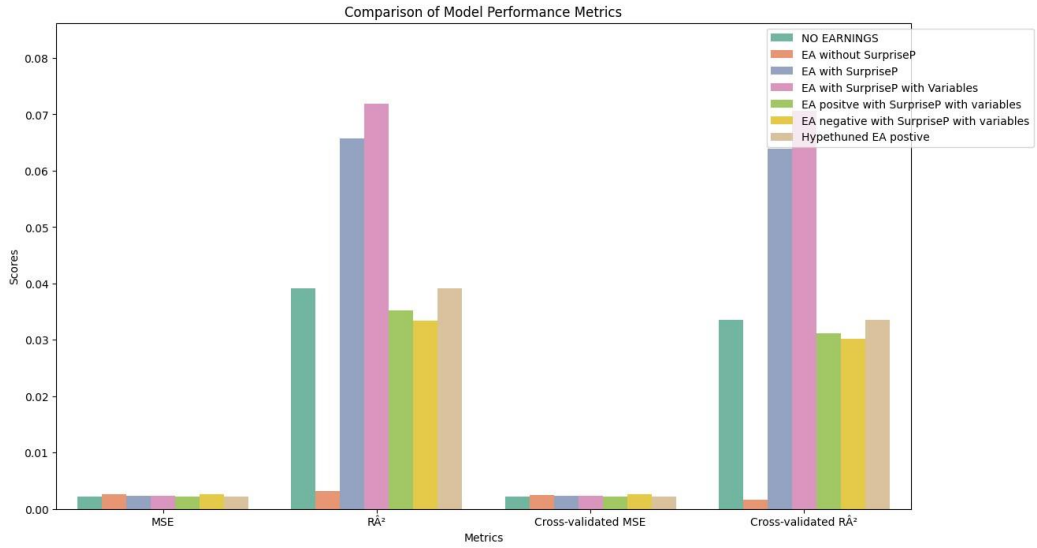
Resources and Annexes

Chat GPT, Copilot, Wikipedia

Results of all the models:

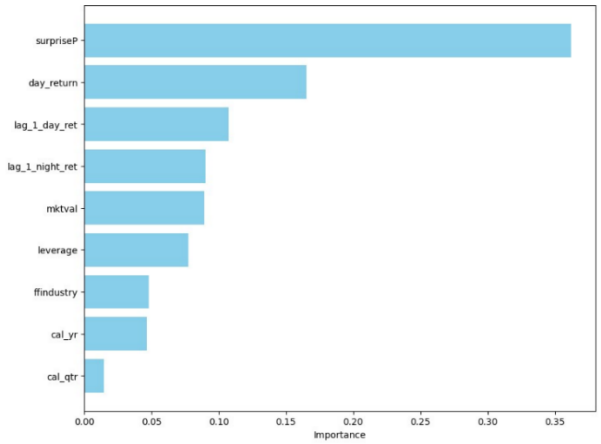
Model	MSE	RÂ²	Cross-validated MSE	Cross-validated RÂ²
NO EARNINGS	0.002147417	0.039136158	0.002164914	0.033530923
EA without SurpriseP	0.002553246	0.003171928	0.002501511	0.00165695
EA with SurpriseP	0.002332199	0.065770522	0.002292251	0.063874825
EA with SurpriseP with Variables	0.002311663	0.071832594	0.002272445	0.070662811
EA positve with SurpriseP with variables	0.002156423	0.035162266	0.002170487	0.031125407
EA negative with SurpriseP with variables	0.002606242	0.033378774	0.002551091	0.030156128
Hypethuned EA postive	0.002147417	0.039136158	0.002164914	0.033530923

Comparison of all the models

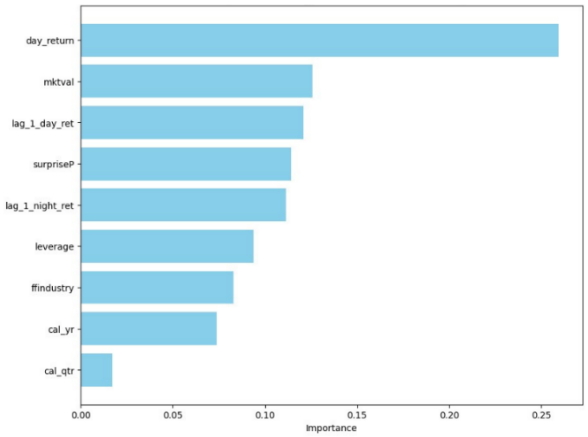


Feature importance:

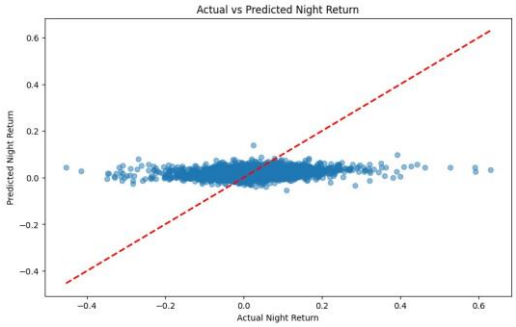
Only positive earnings

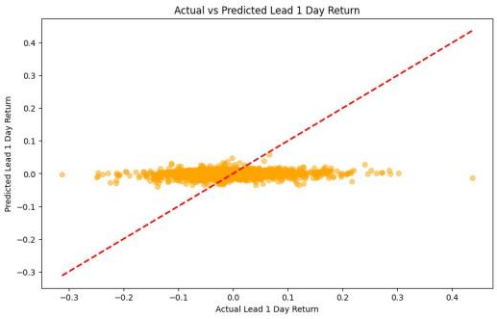


Only negative earnings



Actual vs Predicted values:





OLS results:

OLS Regression Results

Dep. Variable:	lead_1_day_ret	R-squared:	0.002
Model:	OLS	Adj. R-squared:	0.002
Method:	Least Squares	F-statistic:	2524.
Date:	Wed, 15 May 2024	Prob (F-statistic):	0.00
Time:	00:08:32	Log-Likelihood:	2.8306e+07
No. Observations:	14506854	AIC:	-5.661e+07
Df Residuals:	14506841	BIC:	-5.661e+07
Df Model:	12		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0002	9.05e-06	18.325	0.000	0.000	0.000
day_return	0.0019	0.000	6.837	0.000	0.001	0.002
night_return	-0.0465	0.000	-107.858	0.000	-0.047	-0.046
lag_1_day_ret	-0.0023	0.000	-8.991	0.000	-0.003	-0.002
lag_1_night_ret	-0.0172	0.000	-42.331	0.000	-0.018	-0.016
lag_2_day_ret	0.0035	0.000	18.475	0.000	0.003	0.004
lag_2_night_ret	-0.0242	0.000	-60.395	0.000	-0.025	-0.023
lag_3_day_ret	0.0028	0.000	14.588	0.000	0.002	0.003
lag_3_night_ret	-0.0093	0.000	-23.154	0.000	-0.010	-0.008
lag_4_day_ret	-4.479e-05	0.000	-0.234	0.815	-0.000	0.000
lag_4_night_ret	-0.0319	0.000	-79.969	0.000	-0.033	-0.031
lag_5_day_ret	0.0013	0.000	6.878	0.000	0.001	0.002
lag_5_night_ret	-0.0154	0.000	-38.647	0.000	-0.016	-0.015

Omnibus:	43603145.354	Durbin-Watson:	2.003
Prob(Omnibus):	0.000	Jarque-Bera (JB):	268876710504131.625

OLS for lead return 1 without surpirseP

OLS Regression Results

Dep. Variable:	night_return	R-squared:	0.017
Model:	OLS	Adj. R-squared:	0.017
Method:	Least Squares	F-statistic:	2.214e+04
Date:	Wed, 15 May 2024	Prob (F-statistic):	0.00
Time:	00:08:46	Log-Likelihood:	3.5503e+07
No. Observations:	14506854	AIC:	-7.101e+07
Df Residuals:	14506842	BIC:	-7.101e+07
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0006	5.51e-06	102.878	0.000	0.001	0.001
day_return	-0.0698	0.000	-414.243	0.000	-0.070	-0.070
lag_1_day_ret	-0.0156	0.000	-101.921	0.000	-0.016	-0.015
lag_1_night_ret	0.0131	0.000	53.075	0.000	0.013	0.014
lag_2_day_ret	-0.0045	0.000	-38.637	0.000	-0.005	-0.004
lag_2_night_ret	0.0323	0.000	132.140	0.000	0.032	0.033
lag_3_day_ret	-0.0055	0.000	-47.278	0.000	-0.006	-0.005
lag_3_night_ret	0.0120	0.000	49.424	0.000	0.012	0.013
lag_4_day_ret	-0.0062	0.000	-53.066	0.000	-0.006	-0.006
lag_4_night_ret	0.0113	0.000	46.639	0.000	0.011	0.012
lag_5_day_ret	-0.0027	0.000	-23.382	0.000	-0.003	-0.002
lag_5_night_ret	0.0258	0.000	106.775	0.000	0.025	0.026

Omnibus:	29598812.985	Durbin-Watson:	1.999
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2840958130169.120

OLS for night return without surpriseP

OLS Regression Results

Dep. Variable:	night_return	R-squared:	0.003
Model:	OLS	Adj. R-squared:	0.003
Method:	Least Squares	F-statistic:	27.53
Date:	Wed, 15 May 2024	Prob (F-statistic):	2.52e-63
Time:	00:08:48	Log-Likelihood:	1.4438e+05
No. Observations:	102284	AIC:	-2.887e+05
Df Residuals:	102271	BIC:	-2.886e+05
Df Model:	12		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0019	0.000	10.229	0.000	0.002	0.002
day_return	0.0008	0.003	0.266	0.791	-0.005	0.007
lag_1_day_ret	-0.0306	0.005	-5.590	0.000	-0.041	-0.020
lag_1_night_ret	-0.0245	0.009	-2.682	0.007	-0.042	-0.007
lag_2_day_ret	-0.0201	0.006	-3.501	0.000	-0.031	-0.009
lag_2_night_ret	0.0072	0.011	0.683	0.494	-0.013	0.028
lag_3_day_ret	-0.0153	0.006	-2.563	0.010	-0.027	-0.004
lag_3_night_ret	0.0041	0.009	0.454	0.650	-0.014	0.022
lag_4_day_ret	-0.0231	0.006	-3.999	0.000	-0.034	-0.012
lag_4_night_ret	0.0185	0.010	1.781	0.075	-0.002	0.039
lag_5_day_ret	0.0076	0.006	1.346	0.178	-0.003	0.019
lag_5_night_ret	0.0547	0.010	5.340	0.000	0.035	0.075
surpriseP	0.0288	0.002	15.140	0.000	0.025	0.033

Omnibus:	19398.322	Durbin-Watson:	1.960
Prob(Omnibus):	0.000	Jarque-Bera (JB):	572739.956

OLS for night return with surpriseP

OLS Regression Results

Dep. Variable:	night_return	R-squared:	0.003
Model:	OLS	Adj. R-squared:	0.003
Method:	Least Squares	F-statistic:	27.30
Date:	Wed, 15 May 2024	Prob (F-statistic):	9.13e-58
Time:	00:08:48	Log-Likelihood:	1.4436e+05
No. Observations:	102284	AIC:	-2.887e+05
Df Residuals:	102272	BIC:	-2.886e+05
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0700	0.061	1.141	0.254	-0.050	0.190
day_return	0.0002	0.003	0.069	0.945	-0.006	0.006
lag_2_day_ret	-0.0201	0.006	-3.488	0.000	-0.031	-0.009
lag_2_night_ret	0.0086	0.011	0.819	0.413	-0.012	0.029
lag_1_day_ret	-0.0308	0.005	-5.635	0.000	-0.042	-0.020
lag_1_night_ret	-0.0212	0.009	-2.330	0.020	-0.039	-0.003
surpriseP	0.0289	0.002	15.162	0.000	0.025	0.033
ffindustry	2.829e-05	1.49e-05	1.894	0.058	-9.89e-07	5.76e-05
leverage	0.0020	0.001	3.065	0.002	0.001	0.003
mktval	-1.428e-09	5.04e-09	-0.283	0.777	-1.13e-08	8.45e-09
cal_qtr	-0.0006	0.000	-3.408	0.001	-0.001	-0.000
cal_yr	-3.412e-05	3.05e-05	-1.117	0.264	-9.4e-05	2.57e-05

Omnibus:	19385.659	Durbin-Watson:	1.960
Prob(Omnibus):	0.000	Jarque-Bera (JB):	575401.986
Skew:	-0.085	Prob(JB):	0.00

OLS with new variables for night ret

Please refers to the code for the only positive and negative OLS models, for the random forest analysis, and to the excel for the quartiles.

Repository: https://github.com/Mariutoto/Earninigs-annoucements-impact-on-returns-ADA-project-2024