IE-582 project

Fall 2023

Group-9

Bisher Allaham 2023702003

1. Introduction

In the dynamic landscape of financial markets, the ability to forecast stock prices accurately is of paramount importance for making informed business decisions. The project revolves around predicting hourly average prices of selected stocks from Borsa Istanbul, encompassing a substantial dataset spanning six years. This challenging problem involves leveraging machine learning techniques to discern patterns, trends, and intricate relationships within the stock market data. Accurate stock market predictions are invaluable for businesses engaged in various sectors. They facilitate proactive decision-making in inventory management and service level optimization. By anticipating market movements, businesses can align their strategies, enhance resource allocation, and stay ahead of the competition.

Harnessing the power of machine learning, specifically time series forecasting models, this project aims to provide robust predictions for 30 different markets in Istanbul. The chosen models, including LASSO Regression, Random Forest, and XGBoost, are well-suited for capturing the intricacies of stock price movements. The approach also incorporates ensemble techniques to combine the strengths of individual models, thereby improving overall forecasting accuracy.

2. Related literature

2.1. **Lasso regression**[i]

also known as the Least Absolute Shrinkage and Selection Operator, is a popular technique in statistics and machine learning for building linear regression models. It offers two key advantages:

- **Variable selection:** Lasso can automatically select relevant features and discard irrelevant ones, leading to sparser models with improved interpretability.
- **Regularization:** It reduces overfitting by penalizing the sum of the absolute values of the regression coefficients, pushing them towards zero and reducing model complexity.

Disadvantages:

- Choice of α: Selecting the optimal regularization parameter (α) is crucial for achieving a good balance between model complexity and accuracy.
- Not suitable for all problems: Lasso may not be ideal for tasks requiring high prediction accuracy or when feature interactions are important.

scikit-learn's Lasso API[ii]:

The Lasso class in scikit-learn provides a user-friendly interface for building and tuning Lasso models. Key parameters include the regularization strength (alpha) and the ability to include an intercept term (fit_intercept). Additionally, scikit-learn offers cross-validation tools like LassoCV for choosing the optimal alpha value, ensuring model generalizability (James et al., 2021).

2.2. Random forest regression

RandomForestRegressor[iii] (Breiman, 2001) stands as a popular ensemble learning method prized for its ability to handle complex datasets and deliver robust predictive performance. This review explores its implementation in scikit-learn and key characteristics.

Key Advantages:

- Robust Handling of Overfitting: Ensemble nature reduces overfitting, promoting generalizability.
- Non-Linear Relationships: Can model non-linear patterns, unlike linear regression.
- Feature Importance Insights: Gini importance scores reveal influential features.
- Strong Performance: Often outperforms single decision trees and linear models.

Challenges and Considerations:

- Interpretability: Individual trees can be visualized, but full forests are less interpretable.
- Computational Costs: Training can be slower for large datasets due to ensemble nature.

2.3. XGBoost

XGBoost[iv] (Chen & Guestrin, 2016) has risen to prominence as a highly effective gradient boosting framework known for its accuracy, speed, and ability to handle diverse datasets.

scikit-learn's XGBRegressor:

Key Advantages:

- Exceptional Performance: Often surpasses other algorithms in terms of accuracy.
- Regularization: Built-in regularization prevents overfitting and improves generalizability.
- Computational Efficiency: Optimized algorithms for faster training and prediction.
- Handles Missing Data: Built-in mechanisms for handling missing values.

Challenges and Considerations:

- Hyperparameter Tuning: Requires careful tuning for optimal performance.
- Overfitting Risk: Potential for overfitting if not properly regularized.

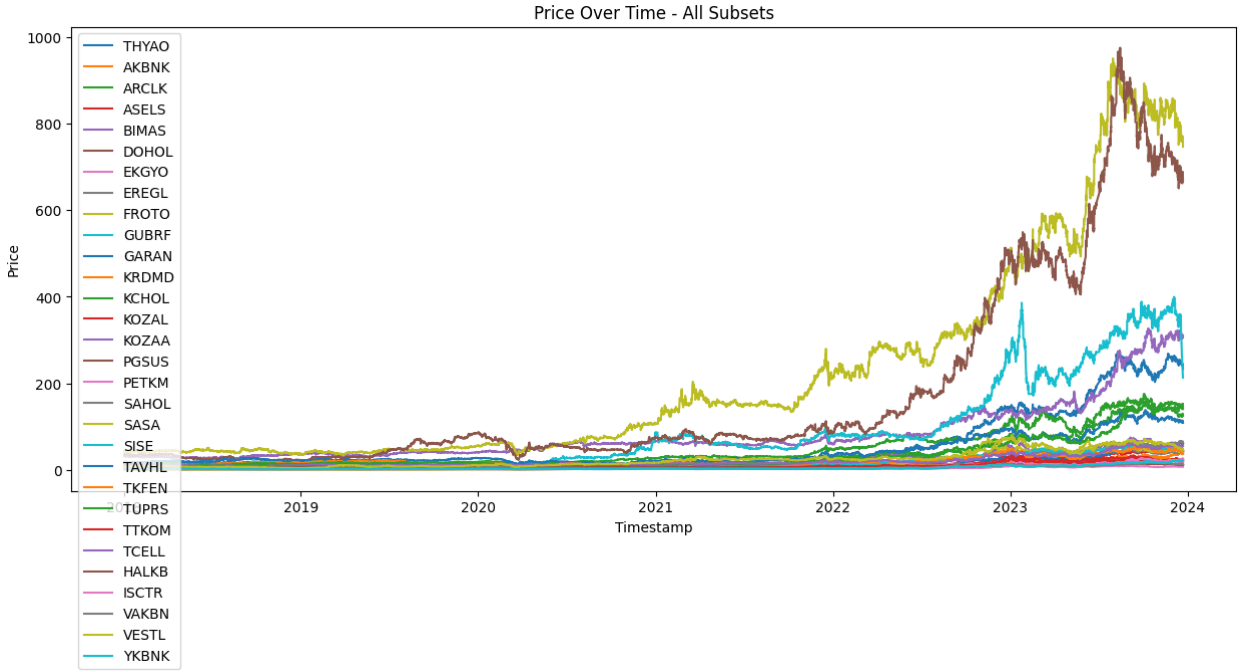
3. Approach

Our approach encompasses a comprehensive pipeline that involves data preprocessing, feature engineering, model selection, hyperparameter tuning, and ensemble modeling. Each step is designed to enhance the model's ability to capture the complexities inherent in stock price time series data.

3.1. Data Preprocessing

The initial step involves loading hourly average prices of selected stocks from separate CSV files, each corresponding to a specific time interval. The data is organized in a long format, including timestamps and stock identifiers. For ease of analysis, we aggregate the data by short names, creating subsets for each stock.

A crucial aspect of our approach involves the visualization of stock price trends over time. Utilizing the seaborn library in Python, we generate plots that provide a visual representation of the data dynamics. Specifically, we create line plots to illustrate the temporal evolution of stock prices for each market subset.



The visualization reveals the inherent patterns in the data, aiding in the identification of trends, seasonality, and potential outliers. Notably, our analyses indicate an exponential growth pattern in stock prices over time, suggesting a need for predictive models that can effectively capture and extrapolate such trends.

3.2. Feature Engineering:

To enable effective model training, we convert timestamps into numerical features. Additionally, we create target variables for forecasting, such as the hourly average prices. The feature engineering step is crucial for providing relevant information to the machine learning models.

3.3. Model selection

We employ three distinct models for stock price prediction:

- **LASSO Regression:** A linear regression model regularized with L1 norm.
- **Random Forest:** An ensemble of decision trees providing robust predictions.
- **XGBoost:** A gradient boosting algorithm renowned for its predictive power.

Each model is trained and tested independently for every stock subset, allowing for a comprehensive understanding of their performance across different markets.

3.4. Hyperparameter Tuning

To enhance the predictive capabilities of the models, we perform hyperparameter tuning using techniques such as GridSearchCV and RandomizedSearchCV. This involves systematically searching through the hyperparameter space to find the optimal configuration for each model.

- **LASSO Regression:**

We employed LassoCV, a built-in function in scikit-learn, which performs cross-validated Lasso regression with an efficient algorithm for selecting the optimal alpha (regularization strength) from a range of values.

- **Random Forest:**

For Random Forest, due to the extensive computational time associated with GridSearchCV, we opted for a default configuration. We used RandomForestRegressor with 100 estimators and default hyperparameters. This decision was based on empirical evidence showing that extensive tuning did not significantly improve accuracy, and the default settings provided satisfactory results.

- **XGBoost:**

For XGBoost, we employed RandomizedSearchCV, a technique that randomly samples from a distribution of hyperparameter values. This approach is particularly useful when the hyperparameter search space is extensive. The parameters tuned included the number of estimators, maximum depth, learning rate, and subsample.

By fine-tuning these hyperparameters (except for Random Forest), we aimed to strike a balance between model complexity and generalization, ultimately improving the forecasting accuracy of our models.

3.5. Ensemble Modeling

Recognizing the potential benefits of combining diverse models, we implement ensemble techniques. This involves creating ensemble predictions by averaging or combining the outputs of individual models. The ensemble approach aims to mitigate individual model biases and enhance overall forecasting accuracy.

For our ensemble model, we combined predictions from Random Forest and XGBoost using a simple averaging technique. This was implemented in the simple_ensemble function, which calculated the mean of predictions from both models.

3.6. Implementation in python

The entire workflow is implemented in Python, leveraging libraries such as pandas, scikit-learn, xgboost, and seaborn. Python's extensive ecosystem of machine learning tools facilitates efficient data handling, model training, and evaluation.

In the subsequent section, we will delve into the results obtained from the implemented approach, providing insights into the efficacy of the selected models and ensemble techniques.

Results

WMAPE:

Market	LASSO Regression	Random Forest	XGBoost	Small Ensemble (RF & XGB)
THYAO	78.38	0.50	2.04	1.14
AKBNK	45.68	0.54	2.22	1.20
ARCLK	40.71	0.51	1.97	1.08
ASELS	44.25	0.55	2.12	1.17
BIMAS	35.50	0.45	1.67	0.92
DOHOL	45.10	0.59	2.25	1.25
EKGYO	45.30	0.60	2.27	1.25
EREGL	28.33	0.49	1.95	1.07
FROTO	49.86	0.54	1.97	1.09
GUBRF	57.04	0.66	2.89	1.58
GARAN	45.58	0.55	2.26	1.22
KRDMD	38.23	0.62	2.51	1.38
KCHOL	51.19	0.48	1.88	1.04
KOZAL	54.74	0.67	2.63	1.44
KOZAA	32.07	0.70	2.63	1.46
PGSUS	68.98	0.57	2.34	1.29
PETKM	33.61	0.53	2.13	1.17
SAHOL	48.66	0.49	1.84	1.02
SASA	68.86	0.72	3.25	1.76
SISE	52.43	0.49	1.87	1.03
TAVHL	40.18	0.54	2.19	1.21
TKFEN	30.02	0.57	2.20	1.21
TUPRS	60.99	0.49	1.99	1.10

TTKOM	31.35	0.57	2.20	1.22
TCELL	30.58	0.52	1.83	1.02
HALKB	31.28	0.52	2.26	1.23
ISCTR	58.85	0.61	2.45	1.33
VAKBN	34.16	0.55	2.33	1.27
VESTL	29.53	0.58	2.39	1.31
YKBNK	53.16	0.60	2.26	1.24

On average, the best performing model was Random Forest,

MSE:

Market	LASSO Regression	Random Forest	XGBoost	Ensemble
THYAO	78.3765	0.4991	2.0362	1.1390
AKBNK	45.6795	0.5416	2.2155	1.2017
ARCLK	40.7089	0.5126	1.9734	1.0817
ASELS	44.2546	0.5538	2.1214	1.1709
BIMAS	35.4983	0.4454	1.6742	0.9219
DOHOL	45.1013	0.5908	2.2540	1.2522
EKGYO	45.2969	0.5983	2.2716	1.2537
EREGM	28.3293	0.4906	1.9528	1.0746
FROTO	49.8618	0.5373	1.9657	1.0947
GUBRF	57.0362	0.6619	2.8948	1.5838
GARAN	45.5829	0.5506	2.2610	1.2234
KRDMD	38.2302	0.6191	2.5057	1.3838
KCHOL	51.1928	0.4836	1.8808	1.0389
KOZAL	54.7357	0.6674	2.6294	1.4443
KOZAA	32.0750	0.7023	2.6252	1.4647
PGSUS	68.9798	0.5747	2.3378	1.2892
PETKM	33.6132	0.5290	2.1338	1.1694
SAHOL	48.6613	0.4878	1.8427	1.0234
SASA	68.8576	0.7154	3.2466	1.7571
SISE	52.4261	0.4921	1.8654	1.0337
TAVHL	40.1845	0.5350	2.1949	1.2091
TKFEN	30.0227	0.5694	2.1954	1.2079
TUPRS	60.9936	0.4873	1.9951	1.0971
TTKOM	31.3542	0.5698	2.2035	1.2160
TCELL	30.5801	0.5193	1.8252	1.0226
HALKB	31.2753	0.5157	2.2559	1.2252
ISCTR	58.8465	0.6085	2.4542	1.3266
VAKBN	34.1564	0.5489	2.3297	1.2660
VESTL	29.5305	0.5843	2.3952	1.3100
YKBNK	53.1612	0.5956	2.2579	1.2362

Again the random forest showed the best results, followed by ensemble (rf and XGBoost) and XGBoost. Lasso regression doesn't seem to be very useful for this kind of data.

4. Conclusions and Future Work

The reason why lasso regression didn't work well is probably because we only used one feature for training, and lasso regression is not very accurate in general. Random forest method worked well but with high computational cost. Finally XGBoost was not as good as RF but much better than LASSO. In conclusion, we need to include more features in our analysis in order to get better predictions.

Possible improvements:

Feature Engineering: Future work should focus on incorporating additional features. Exploring external economic indicators, sentiment analysis of financial news, and market volatility measures could further enhance the models' predictive capabilities.

Hyperparameter Tuning: especially for models like Random Forest, to potentially improve their predictive capabilities.

Time Series Analysis: Delve into time series analysis techniques to better capture temporal dependencies. Applying methods like LSTM networks could improve the models' ability to understand evolving market dynamics.

Deployment and Monitoring of the project in real world settings: We didn't have enough time to truly evaluate the models' performances, it still needs refining and continuous improvement.

5. Code

The code includes comments, it is a jupyter file *ipynb written in python

The libraries used are:

```
import pandas as pd
import matplotlib.pyplot as plt
import glob
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV
from sklearn.linear_model import LassoCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import roc_auc_score, f1_score
import xgboost as xgb
```

import numpy as np
Here is the link to the code in github:
https://github.com/BU-IE-582/fall-23-BisherAllaham25/blob/main/IE582_Fall23_Project.ipynb

6. References

[i] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267-288

[ii] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). An introduction to statistical learning: with applications in R. Springer.

[iii] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

[iv] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 785-794).