Martin Ramberg, Jonas Indreland Skjerpe, William Vestergaard Soma

Performance of XGBoost compared to conventional technical stock analysis for predicting returns

Comparing modern data science tools and machine learning to conventional stock analysis techniques

Termpaper in Decision Theory and Corporate Finance
April 2023

Norwegian University of Science and Technology
Economics and Administration
NTNU Business School

NTNU

ABSTRACT

In this study, we evaluate the effectiveness of an XGBoost (XGB) classification model for predicting 5 percent stock returns over a 14-day period and compare its performance to technical analysis using Price-to-Earnings (P/E) ratios and the naive forecast method. Our findings indicate that the XGB model, when carefully optimized for this task, surpasses the P/E ratio-based multiple analysis in terms of overall performance and model a more balanced outcome. The XGB model's accuracy is however inferior to the naive forecast method, even though it excels in precision, recall, and F1-score. Due to the limited test size of only 12 samples, the conclusions drawn from our study remain inconclusive. Future research with larger test sizes, more data, and increased computing power, may lead to more reliable outcomes.

PREFACE

We have preconceived beliefs that both technical analysis and the machine learning model will not perform well and will be beaten by a simple prediction. We do not predict which model will perform better, but believes they will perform similarly. These beliefs may affect the conclusions of the paper even if we try to remain unbiased. We also believe that analyzing stock price and volume in isolation is not useful and that a random walk model is better without additional financial data. There may be biases in favor of one method over another due to our expertise, confirmation bias, cherry-picking stocks, and availability of data.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

- **EDA** Exploratory Data Analysis
- **NTNU** Norwegian University of Science and Technology
- **ML** Machine learning
- **XGB** eXtreme Gradient Boost (a kind of machine learning model)
- **P/E** Price to Earnings Ratio
- **TA** Technical analysis

CHAPTER

ONE

INTRODUCTION

The world of finance has always been a place of innovation and development. Investors and traders constantly seek ways to gain an edge in the highly competitive and rapidly evolving markets. The use of machine learning algorithms and technical analysis in predicting stock market trends and movements has gained significant traction in recent years. With the lowered entry-level, both in skills and hardware needed, it is possible for the majority of people with a computer to perform this analysis.

This thesis aims to provide a somewhat comprehensive and comparative study of the effectiveness of machine learning models, specifically the XGBoost classification model, and traditional technical analysis in forecasting stock market returns. The primary goal of this research is to assess the accuracy and reliability of these two approaches in predicting the direction of stock prices, taking into consideration various factors such as the choice of companies and prediction thresholds.

1.1 Motivation

With the increasing accessibility of machine learning, anyone can train their own models without specialized degrees. This raises the question of whether everyone should use these techniques for their analysis work. The goal is not to beat the market, but to compare the performance of modern data science tools, which have become more accessible due to easy-to-use libraries and free access to financial data. The authors find this interesting because machine learning models can be deployed with only a few lines of code, and inter-day prices are readily available. The authors aim to determine whether these techniques are effective in predicting the stock market.

1.2 Project description

In this study, we aim to train a machine learning classification model using stock data from the Oslo Stock Exchange. The model will be designed to take financial data as input and predict whether the stock price will increase, decrease, or remain unchanged 14 days into the future compared to the current day.

Furthermore, we will create our own prediction using technical analysis on the same stock, starting from the same point in time. The performance of the machine learning model will be compared to our technical analysis prediction, as well as the naive forecast for the stock. The naive forecast, which employs a random walk model, will serve as a benchmark for the other methods. We will compare the predictions of the machine learning model and technical analysis with a 14-day threshold, aiming to predict whether the return will be positive, negative, or neutral. A return will be classified as either long or short if it reaches at least a 5% threshold, which we have chosen, and neutral otherwise.

We will conduct 12 predictions for each model, divided among three companies at three moments in time. One prediction will be made for each quarter in 2020. The results will be bench-marked against the stock price 14 days after the prediction was made. Should the stock price rise following the 14-day span, it won't be taken into account; similarly, if the stock price climbs within the 14-day window but declines prior to the 14th day, it will also be disregarded. Only data from the prediction date and earlier will be available for making the prediction, which will be discussed in further detail when determining the training and test data split.

The companies we have chosen for the test are:

- Equinor
- SpareBank 1 SR-Bank
- Salmar

These companies were selected because they are mature, likely to have complete data, and have comparable peers. They also represent different sectors, which helps generalize the experiment to encompass a broader range of industries.

CHAPTER

TWO

THEORY

2.1 Tree based models

This section is borrowed from Bjørgum and Lindtveit's paper [1].

This is a family of both regression and classification methods. This subsection is largely recited from Hastie, Tibshirani, and Friedman [2] and James et al. [3]. The models *stratify* or *segment* the predictior space into a number of simple regions. To make a prediction for a given observation, we typically use the mean or the mode response value for the training observations in the region which it belongs. The set of splitting rules can be summarized in a tree, also known as a *Decision tree*. This is great for interpretability, but cannot usually compete with more sophisticated supervised learning methods. For this reason we also introduce *Random forest* and *XGBoost*, which often has better predictive power, at the expense of some loss in interpretation.

2.1.1 Decision trees

As mentioned, this kind of trees work by creating splitting rules. We firstly describe how to grow a regression tree, before we convert it to a classification tree.

2.1.1.0.1 Fitting a regression tree. We have a dataset consisting of p inputs and a response, for each of N observations. The algorithm needs to automatically decide on the splitting variables and split points, as well as what typology (shape) the tree should have. We have a partition into M regions $R_1, R_2, \ldots, R_M$, and we model the response as a constant c_m in each region:

$$f(x) = \sum_{m=1}^{M} c_m I\left(x \in R_m\right). \tag{2.1}$$

In the regression tree, we minimize the sum of squares $\sum(y_i - f(x_i))^2$. We see that the best $\hat{c}_m$ is the average of y_i in region R_m:

$$\hat{c}_m = \text{ave}(y_i | x_i \in R_m). \tag{2.2}$$

Finding the best binary partition in terms of minimum sum of squares is usually computationally infeasible, and we use a greedy algorithm. Starting with all of the data, we use a splitting variable j and split point s, and define the pair of half-planes

$$R_1(j,s) = \{X|X_j \leq s\} \text{ and } R_2(j,s) = \{X|X_j \ > s\}. \tag{2.3}$$

Then we seek the splitting variable j and s that solve

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i - c_1)^2 + \min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i - c_2)^2\right]. \tag{2.4}$$

For all choices of j and s, the inner minimization is solved by

$$\hat{c}_1 = \text{ave}\left(y_i|x_i \in R_1(j,s)\right) \text{ and } \hat{c}_2 = \text{ave}\left(y_i|x_i \in R_2(j,s)\right). \tag{2.5}$$

For each splitting variable, the determination of the split point s can be done quickly and hence by scanning through all of the inputs, determination of the best pair j, s is feasible. Having found the best split, we repeat this splitting in all of the resulting regions.

To avoid overfitting, we also need to set a tree size as a tuning parameter. The optimal tree size should be adaptively chosen from the data. The preferred strategy is to grow a large tree T_0, stopping the splitting process when some minimum node size is reached. We then prine this tree using *cost-complexity pruning.* This pruning works by defining a subtree $T \subset T_0$ to be any tree that can be obtained by pruning T_0, that is, collapsing any number of its internal nodes. We index terminal nodes by m, with node m representing region R_m. Let $|T|$ denote the number of terminal nodes in T. Letting

$$\begin{aligned} N_m &= \#\{x_i \in R_m\}, \\ \hat{c}_m &= \frac{1}{N_m}\sum_{x_i\in R_m} y_i, \\ Q_m(T) &= \frac{1}{N_m}\sum_{x_i\in R_m}(y_i - \hat{c}_m)^2, \end{aligned} \tag{2.6}$$

we define the cost complexity criterion

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|. \tag{2.7}$$

The goal is to find, for each α, the sub-tree $T_\alpha \subseteq T_0$ to minimize $C_\alpha(T)$. The tuning parameter $\alpha \geq 0$ governs the trade-off between tree size and its goodness of fit to the data. α can be adaptively chosen.

For each α there is a unique smallest sub-tree T_α that minimizes $C_\alpha(T)$. To find this sub-tree, we can use *weakest link pruning.* This works by successively collapsing the internal node that produces the smallest per-node increase in $\sum_m N_m Q_m(T)$, and continue until we produce the single-node tree. This gives a finite sequence of subtrees, and this sequence must contain T_α. We then estimate α by cross-validation,and we choose the value $\hat{\alpha}$ that minimizes the sum of squares, resulting in $T_{\hat{\alpha}}$.

2.1.1.0.2 Converting the regression tree to a classification tree. Where we in regression trees used the impurity measure $Q_m(T)$, we need to use another measure. In a node m, representing a region R_m with N_m observations, let

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k), \tag{2.8}$$

the proportion of class k observations in node m. The observations in node m gets classified to class $k(m) = \arg\max_k \hat{p}_{mk}$, the majority class in node m. We then have different measures for the node impurity $Q_m(T)$:

$$\text{Misclassifaction error: } \frac{1}{N_m} \sum_{i \in R_m} I\left(y_i \neq k(m)\right) = 1 - \hat{p}_{mk(m)} \tag{2.9}$$

$$\text{Gini index: } \sum_{k \neq k'} \hat{p}_{mk}\hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}\left(1 - \hat{p}_{mk}\right) \tag{2.10}$$

$$\text{Cross-entropy or deciance: } -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}. \tag{2.11}$$

For two classes, if p is the proportion in the second class, these three measures are $1 - \max(p, 1-p)$, $2p(1-p)$ and $-p\log p - (1-p)\log(1-p)$, respectively. All of the measures are similar, but cross-entropy and Gini index are differentiable, and hence is more amendable to numerical optimization.

2.1.2 Random forests

Random forests build on the decision tree. The forest is constructed by creating a selection of decision trees, and combining the prediction of these trees to make a prediction. A central idea for random forests is bagging.

2.1.2.0.1 Bagging (bootstrap aggregation). This builds on the concept of bootstrapping. A given set of n independent observations $Z_1, \cdots, Z_n$, each with variance σ^2, the variance of the mean $\overline{Z}$ of the observations is given by σ/n. This shows that averaging a set of observations reduces variance. A natural way to reduce the variance and increase the test set accuracy is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions. We can then obtain a low-variance learning model given by

$$\hat{f}_{\text{avg}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(x). \tag{2.12}$$

This method is not practical, as we usually don't have multiple training sets. A solution to this is to bootstrap, meaning that we take repeated samples from the training data set. With this approach we generate B different bootstrapped training data sets. We then train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$, and finally average all the prediction to obtain

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x). \tag{2.13}$$

2.1.2.0.2 Fitting the random forests. The random forest provide an improvement over bagged trees by way of a small tweak that de-correlates the trees. As we do with bagging, we build a number of decision trees on bootstrapped training samples. The difference is that each time a split is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors. At each split, a fresh sample of m predictors is taken, and usually we choose $m \approx \sqrt{p}$. This means that at each split in the trees of a random forest, the model is not allowed to consider a majority of the available predictors. The main difference between a random forest and a bagged tree is choice of predictor subset m.

2.1.3 XGBoost

eXtreme Gradient Boosting (XGBoost), introduced by Chen and Guestrin [4], is another method that builds on the foundation of decision trees. The model uses clever penalization and regularization methods. It is based on a *gradient boosting* algorithm.

2.1.3.0.1 Boosting decision trees. To understand the gradient boosting, we must first describe boosting in the context of decision trees. Boosting works in much the same way as bagging, except that the trees are grown sequentially. Each tree is grown using information from previously grown trees. Boosting does not involve bootstrapping the data set, instead each tree is fit on a modified version of the original data set.

The algorithm for boosting a decision tree can be described as firstly: setting $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.

Then, for all $b = 1, 2, \cdots, B$ we first fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r). After that, we update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{2.14}$$

Then we update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{2.15}$$

Finally, after we have repeated through $b = 1, 2, \cdots, B$, we output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{2.16}$$

Boosting is tuned by three hyper parameters:

1. The number of trees B. Unlike bagging and random forests, boosting can over-fit if B is too large.
2. The shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns.
3. The number d of splits in each tree, which controls the complexity of the boosted ensemble.

Gradient boosting then refers to boosting the tree using its gradients. This requires a differentiable loss function. For classification trees this can be the deviance.

2.1.4 Feature importance metrics in XGBoost

When deciding which variables to use in a model, one should consider multiple factors beyond feature importances such as interpretability, data quality, relevance to the problem, and domain knowledge. That being said, feature importances can provide valuable insights into the predictive power of individual variables. The three types of feature importances you mentioned (weight, gain, and coverage) each have their own strengths and weaknesses.

Weight is the number of times a feature appears in the trees of the model. This measure can be biased towards variables with high cardinality or numerical variables with many unique values. Weight is useful for identifying the most commonly used features by the model, but it doesn't take into account how much the feature contributes to the final predictions.

Gain is the average gain of splits that use the feature to reduce impurity. Gain measures how much the feature contributes to reducing the model's loss function. This measure is more informative than weight, as it takes into account the contribution of each feature to the model's predictive power.

Coverage is the average coverage of splits that use the feature. Coverage measures the relative quantity of observations concerned by a feature. Coverage is useful when you have imbalanced classes and want to know if the model is focusing on a particular subset of the data.

You can use a combination of these measures to rank features and decide which ones to include in your final model. For example, you can start by ranking features by gain, then examine the top features by weight and coverage to see if they provide additional insights or improve the model's interpretability.

It's also important to keep in mind that feature importance measures can be noisy and highly dependent on the specific data set and modeling approach. Therefore, it's always a good practice to cross-validate the results and experiment with different subsets of variables to ensure that the final model is robust and performs well on new, unseen data.

2.2 Regularization

Regularization is a technique used in machine learning to prevent over-fitting of a model. Over-fitting occurs when a model is too complex and fits the training data too well, leading to poor generalization performance on new, unseen data. Regularization methods add constraints to the model to prevent it from becoming too complex and over-fitting the training data. L1 and L2 regularization are two common techniques used for regularization.

L1 regularization, also known as Lasso regularization, adds a penalty term to the loss function of the model that is proportional to the absolute value of the model's coefficients. This penalty term encourages the model to have sparse coefficients, where many of the coefficients are zero. This can be useful in feature selection, as it can identify and remove irrelevant features from the model. On the other hand, L2 regularization, also known as Ridge regularization, adds a penalty term to the loss function of the model that is proportional to the square of the model's coefficients. This penalty term encourages the model to have small, non-zero coefficients. This can help prevent over-fitting by making the model less sensitive to changes in the input data.

Both L1 and L2 regularization can be controlled by a regularization parameter, often denoted as lambda (λ). A higher value of λ will increase the strength of the regularization, resulting in a simpler model with smaller coefficients. However, if λ is too high, the model may become under-fit and have poor performance on both the training and test data.

Regularization techniques like L1 and L2 regularization are useful for feature selection in machine learning models. When we have a large number of features in our data set, these techniques can create less complex (parsimonious) models that address over-fitting and feature selection. Lasso Regression is a regression model that uses L1 regularization, while Ridge Regression uses L2 regularization. The key difference between these two is the penalty term. Ridge regression adds a "squared magnitude" of the coefficient as a penalty term to the loss function, while Lasso regression adds the "absolute value of magnitude" of the coefficient as a penalty term to the loss function.

Lasso regression shrinks the less important feature's coefficient to zero, removing some features altogether, while Ridge regression avoids over-fitting. Traditional methods like cross-validation and step wise regression work well for small sets of features, but regularization techniques like Lasso and Ridge Regression are useful when dealing with a large set of features. Overall, regularization is an effective technique for improving the generalization performance of machine learning models by preventing over-fitting.

2.3 Principal component analysis

Principal Component Analysis (PCA) is a widely-used dimensionality reduction technique in machine learning and data analysis. PCA is a linear transformation technique that aims to reduce the dimensionality of a dataset while preserving as much of the variation in the data as possible. The resulting reduced-dimensional data can then be more easily analyzed and visualized.

The PCA algorithm works by finding the principal components of the data, which are the directions in the data that explain the most variance. These principal components are computed by finding the eigenvectors of the covariance matrix of the data. The eigenvectors with the largest eigenvalues correspond to the directions with the most variance, and these are used to create a new coordinate system for the data.

In the new coordinate system, the data is projected onto the principal components. The first principal component corresponds to the direction in the data that explains the most variance, the second principal component corresponds to the direction that explains the second most variance, and so on. The number of principal components used to represent the data can be chosen based on the amount of variance that needs to be preserved.

PCA has several important applications in machine learning and data analysis. One of the primary applications is dimensionality reduction, where PCA can be used to reduce the dimensionality of a high-dimensional dataset to a lower-dimensional one that still captures most of the variation in the data. This can be useful for visualization, clustering, and other downstream analysis tasks.

PCA can also be used for feature extraction, where it can be used to identify the most important features in a dataset. By projecting the data onto the principal components, we can see which features have the largest influence on the variation in the data.

In addition, PCA can be used for data compression, where it can be used to compress high-dimensional data into a lower-dimensional representation. This can be useful for reducing the storage requirements of large datasets.

In summary, Principal Component Analysis is a powerful and widely-used dimensionality reduction technique that is used in a variety of machine learning and data analysis tasks. By finding the principal components of the data, PCA can reduce the dimensionality of a dataset while preserving as much of the variation as possible. This can be useful for visualization, feature extraction, and data compression.[5]

2.4 The naive forecast

The model assumes that the future value of the variable (denoted as P) at time t+1 is equal to its current value (P_t) plus a random error term (ϵ). The error term is assumed to be normally distributed with a mean of zero and a standard deviation of σ, which represents the random "shocks" to the system. Unlike the random walk with drift model, this model assumes that there is no long-term trend or drift in the data, and that any changes in the variable are due solely to random fluctuations in the system.

This model is often used in finance and economics to model asset prices or exchange rates, as it assumes that prices follow a random walk pattern where current prices are the best predictor of future prices, but are subject to random shocks. It is important to note that this model has its limitations, as it assumes that the underlying data is stationary and that the error terms are independent and identically distributed (IID). In practice, this may not always be the case, especially during periods of high volatility or significant events that can impact the price of the asset. More complex models may be needed to capture these dynamics and improve the accuracy of predictions.

$$P_{t+1} = P_t + \epsilon, \quad \epsilon \sim N(0, \sigma) \tag{2.17}$$

From this model we can get the naive forecast by taking the expected values of P_{t+1} as such:

$$E[P_{t+1}] = E[P_t + \epsilon] \tag{2.18}$$

We assume these these variables to be independent, such that:

$$E[P_{t+1}] = E[P_t] + E[\epsilon] \tag{2.19}$$

Since we already know today's stock price the expected value is itself.

$$E[P_{t+1}] = P_t + E[\epsilon] \tag{2.20}$$

We assumed in the specification of the model that ϵ is normally distributed with mean 0 and standard deviation σ. Therefore the expected values of the shock variable has to be 0.

$$E[P_{t+1}] = P_t + 0 \tag{2.21}$$

$$E[P_{t+1}] = P_t \tag{2.22}$$

We find that the expected value of tomorrow's stock price is the same as the today's stock price when modelling it as Markov model. This is known as the naive forecast. From here we can easily extrapolate the result, and we get that:

$$E[P_{t+14}] = P_t \tag{2.23}$$

Although it is a simple model more sophisticated models often struggle to beat the naive forecast. This is the reason for including it as a benchmark.

We have shown that naive forecast is simply the last observed value of P. How confident should we be in this forecast? What is the standard deviation of our forecast?

$$Var(P_{t+1}) = Var(P_t + \epsilon) \tag{2.24}$$

$$Std(P_{t+1}) = \sigma \tag{2.25}$$

$$Std(P_{t+14}) = 14\sigma \tag{2.26}$$

So in our case the standard deviation will be. This is quite large. Hopefully this aids in illustrating the how the uncertainty of stock predictions increases as the forecast horizon does.

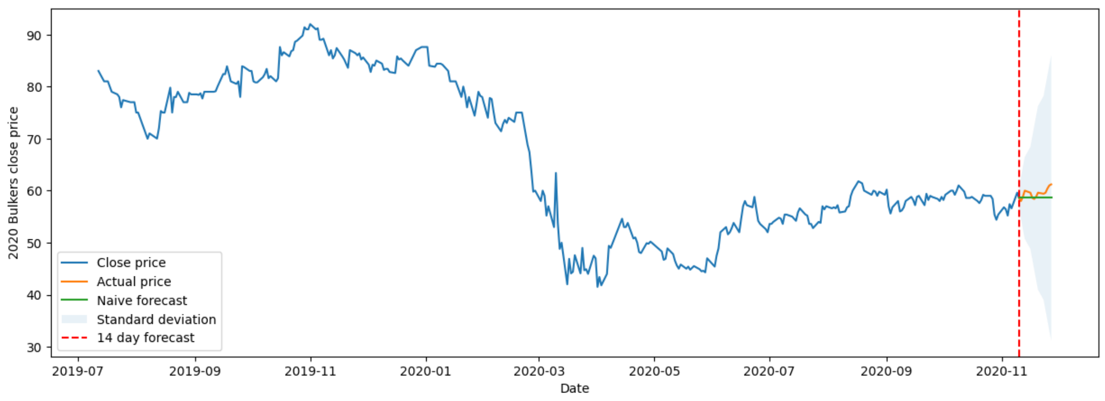


Figure 2.4.1: 14-day naive forecast on 2020 Bulkers close price.

2.5 Multiple evaluation theory

In stock valuation there are two main types of methods used. The first one is based on the intrinsic value and is usually computed by using a discounted cash flow model. The second method is multiple analysis and is regarded as finding the relative value of a company compared to its peers. This relative evaluation is because similar companies should be priced similarly. This makes it best suited for our case since we are trying to predict a movement signal by looking at possible arbitrage opportunities in prices. This implies that we are searching for either overvalued or undervalued assets to predict the trend they should be moving in compared to their peers.

2.5.0.1 What is a multiple

A multiple is a measure of a company's fundamental metric. This enables us to compare companies and evaluate their performance. Different multiples represent different aspects of the business, giving us performance metrics for example growth, productivity, and efficiency. Investors can then quickly get an understanding of the evaluation and further opportunities for growth in a company. In this paper, we focused on the price divided by the earnings multiple to evaluate companies.[6]

2.5.0.2 Price to Earnings multiple (P/E)

This is the most common multiple for the evaluation of a stock. This looks at the share price of a company and divides it by the earnings of the company. This results in a multiple which implies what the company is valued at compared to what they are making. Different levels of P/E may be more present in certain sectors. The value of the P/E multiple may also indicate the trust in a company's future growth and ability to earn more. A P/E of 20 represents the company's stock is valued at 20x times its earnings. A relatively high P/E ratio generally represents an overvalued stock and vice versa with a relatively low P/E value. Thus, startups with very low earnings have a high P/E ratio, and the investors are betting on the company to further grow and earn more in the future.[7]

CHAPTER

THREE

METHODS

3.1 The dataset

We obtained our data for the Oslo Stock Exchange using Titlon, a tool developed by UiT [8]. This provided us with a comprehensive data set containing detailed time series for all stocks ever listed on the Oslo Stock Exchange. The data set includes numerous indicators beyond basic stock information, such as closing price, opening price, and peak price per day. A list of the contents and an explanation of the data from Titlon can be found in the appendix.

For our multiple analysis, we utilized the same Titlon dataset to compare Price-to-Earnings (P/E) ratios. To supplement this, we also referenced charts from TradingView INC, which allowed us to obtain accurate P/E information for stocks not listed on the Oslo Stock Exchange.

3.1.1 Data balance

We have under Parameters of the experiment determined how we will create our signal variable. In essence the stock price will have to move by more that 5 percent in either direction in order to classified as long or short. We find it important, however, to enlighten how this threshold is set affects the balance of the data set as this will affect how model performance is measured.

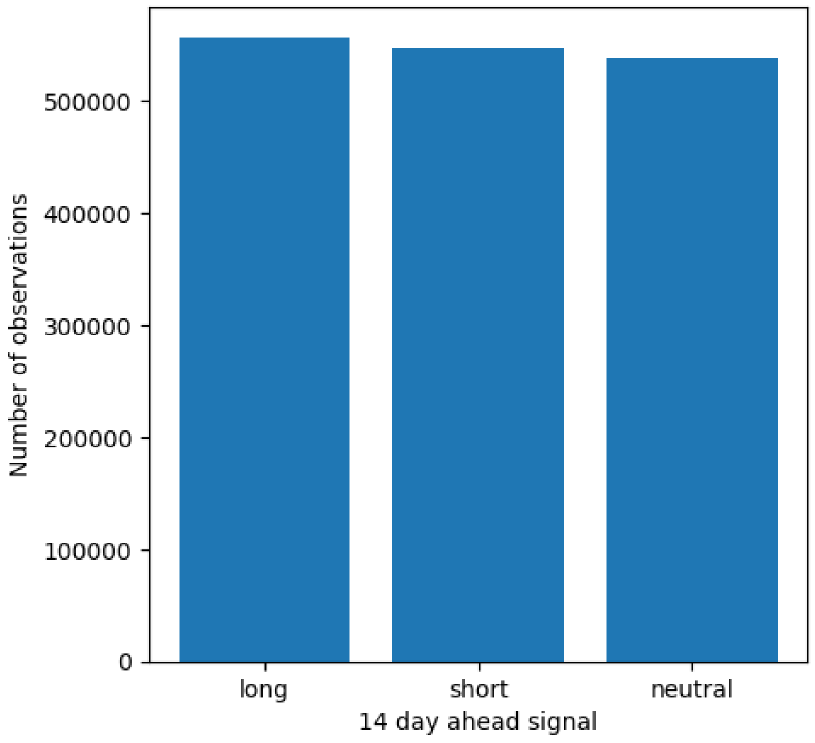


Figure 3.1.1: Data set balance with threshold set to 3 percent.

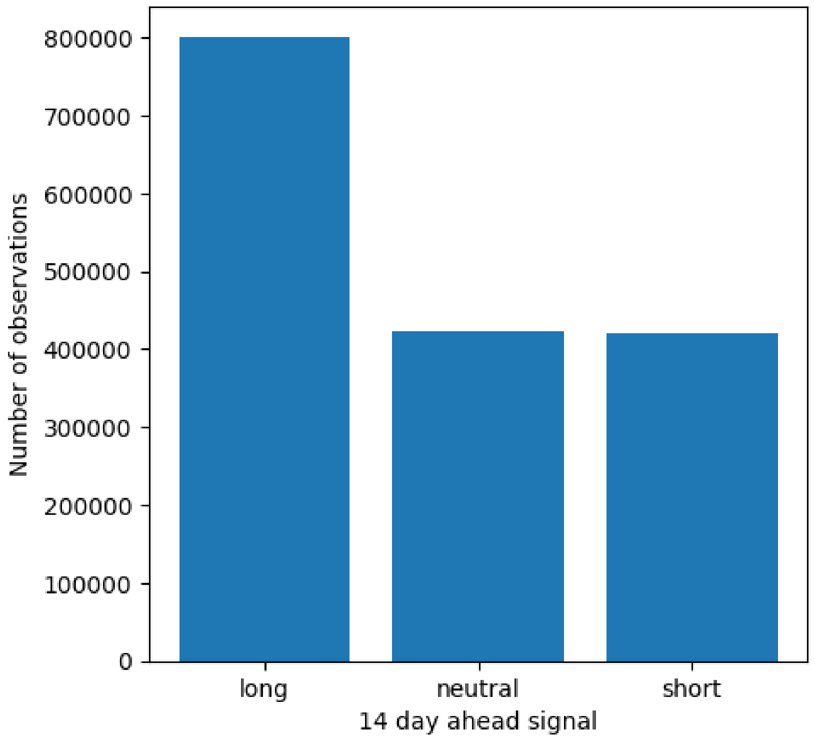


Figure 3.1.2: Data set balance with threshold set to 5 percent.

Although the choice of threshold is largely arbitrary, the consequences for the data are pretty large as you can see in the figures. On one hand we want the threshold to be high enough so that we can be sure it indicates a trend and might make our predictions better, on the other hand, setting the threshold to high will leave returns on the table. Our model could for instance routinely ignore returns of say 4.5 percent.

3.2 The XGBoost model

3.2.1 Data preparation

Our test- train split will be at the beginning of year 2020 in our data. This is to unsure that our results are unbiased. We used the data from 2018 to 2020 as our validation data, and the rest for training. This is done to prevent our model to over-fit.

As we are using XGBoost classification model we did not have to create one hot encoded vectors for all categories as XGBoost will do this on its own using sparse matrices, thus saving memory. We forward-filled some of the numerical data like close and open price, but did not do anything else about missing data as XGBoost handles these gracefully by setting them to zero. In a linear regression model this might be a cause for concern, but in a tree based model like XGBoost it will not be a problem.

From the data set we engineered 50 lags for all numerical columns as well as an assortment of time variables designed to catch on to cyclical features of the data.

3.2.2 Model specification

We scaled all the numerical variables such that they have a mean of zero and a standard deviation of one. Afterwards we performed a PCA transformation of the 1 300 variables and ended up with 20 principal components, which in theory will contain all of the variance except the noise. This along with our categorical variables We ended up with 30 input variables.

3.2.3 Hyper parameter tuning

We first addressed potential class imbalances in the data set by computing the inverse class frequencies as sample weights. This approach assigns higher weights to less frequent classes, which can improve the model's performance when handling imbalanced data sets. The F1-score, a weighted average of precision and recall, was chosen as the performance metric for evaluating the model. The weighted F1-score accounts for class frequencies, making it suitable for imbalanced classification problems.

To identify the optimal set of hyper parameters for the XGBoost classifier, we employed a grid search approach with cross-validation (cv=2). We performed this search iteratively, refining the parameter grid based on the best parameters found in the previous round. In the first round, the initial parameter grid included a range of values for learning rate, max depth of trees, γ , and λ. In the second round, based on the results of the first round, a refined parameter grid was defined with adjusted values for learning rate, max depth, and λ. In the final round, an even more refined parameter grid was used, with further adjustments to the learning rate, max depth, and λ values.

3.2.4 The final model

In order to make the final model more comprehensible, we employed early stopping at 10 trees, and chose the Multi soft probability objective function. We also implemented the histogram-based tree construction method. For multi-class classification, we utilized the log loss function. To address the issue of an imbalanced data set, we assigned weights to samples to ensure equal representation from each category during training. The model trained a total of 6 020 trees, with the optimal number for prediction being 5 921 trees. The minimum loss achieved during training was 1.01198, which is generally considered quite poor.

The final model is an XGBoost classifier designed to perform multi-class classification using a soft probability objective. After feature selection and hyper parameter tuning, the model was constructed with specific parameters and settings. The objective was set to 'Multi:softprob', indicating that the model is designed for multi-class classification problems and outputs class probabilities. Categorical features were enabled, and the tree method parameter was set to 'hist', speeding up the training process by using a histogram-based algorithm to grow the trees.

The model was specified to have a maximum of 100,000 boosting rounds. However, early stopping was likely to halt the process before reaching this limit. To prevent over-fitting, 80 percent of the training samples were randomly selected for each boosting round using a 'subsample' parameter of 0.8, and 80 percent of the features were randomly selected for each tree using the 'colsample_bytree' parameter set to 0.8. The early stopping parameter was set to 100, stopping the training process if the validation performance did not improve for 100 consecutive rounds. The multi-class logarithmic loss ('mlogloss') was used as the evaluation metric during training.

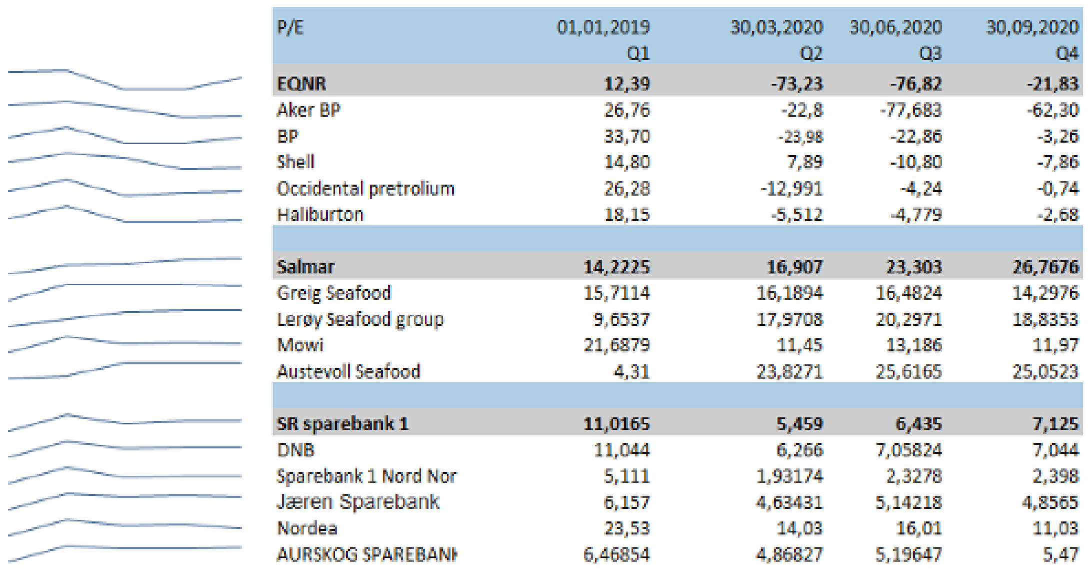
After hyperparameter tuning, the following values were selected: a learning rate of 0.01, a maximum tree depth of 8 and a regularization parameter (reg_lambda) of 50. The final model was then trained on the training data concerning the stocks in question set using the optimized hyper parameters and sample weights computed based on the inverse class frequencies, helping to account for potential class imbalances in the data set. The model was evaluated on both the training and validation sets throughout the training process, with the results being output every 100 rounds for monitoring. After training, the final model was ready for testing.

3.3 Technical analysis

We decided to test the technical analysis evaluation method of comparing multiples in an effort of obtaining the fair market-based evaluation of a company. This kind of analysis is frequently used in academic research to investigate the connections between various factors and to forecast how the stock market will act.

The first step is to choose the group of stocks that will be relevant for our comparisons. This is to ensure that we make the analysis as fair as possible by choosing peers and other competitors that resemble the company in question. Depending on the size and scale of the company we may have to look further then just on the Oslo Stock Exchange for comparable peers. An example here is Equinor, where we decided to look at Shell, Haliburton, Occidental Petroleum inn order to resemble Equinor's global size and output volume. For SR Sparebank 1 and Salmar it was easier to find comparable peers based in Norway, this made the data collection process faster. We decided to go with four peers instead of 5 for Salmar since it was difficult to find comparable peers.

Further after finding peers, we plot their Price to Earnings ratio in a table for each of the four periods which we are trying to predict. In order to combat interference from knowing the future price movements we objectively chose the peers to companies before looking at their performance history. This was to take into account that there exist outliers that may affect out results.

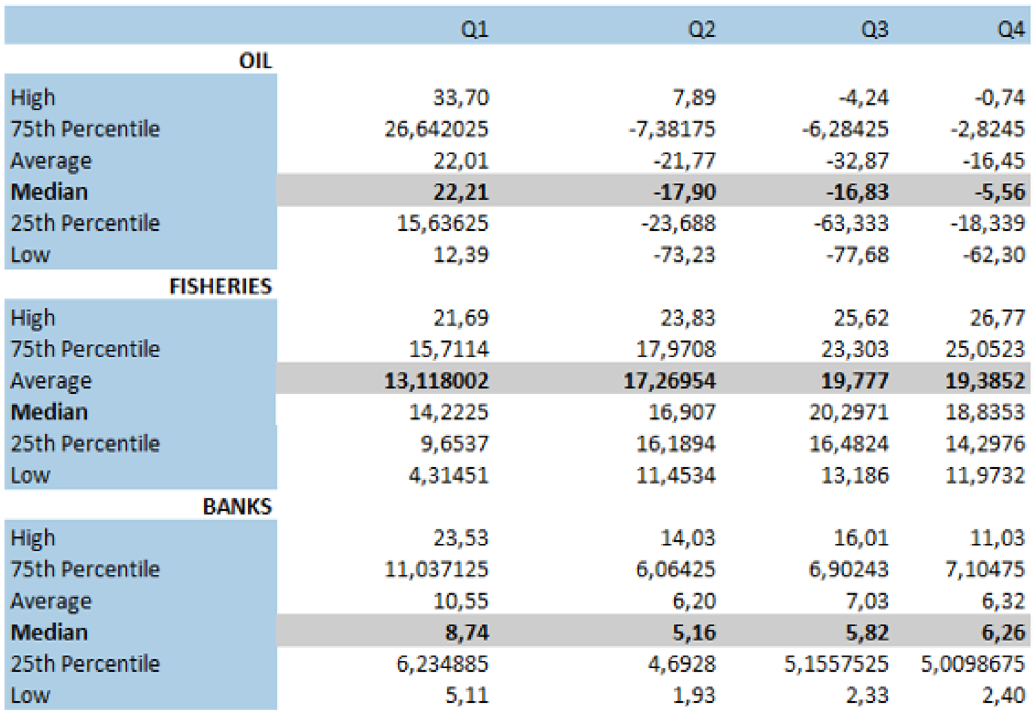


P/E	01,01,2019 Q1	30,03,2020 Q2	30,06,2020 Q3	30,09,2020 Q4
EQNR	**12,39**	**-73,23**	**-76,82**	**-21,83**
Aker BP	26,76	-22,8	-77,683	-62,30
BP	33,70	-23,98	-22,86	-3,26
Shell	14,80	7,89	-10,80	-7,86
Occidental pretrolium	26,28	-12,991	-4,24	-0,74
Halliburton	18,15	-5,512	-4,779	-2,68
Salmar	**14,2225**	**16,907**	**23,303**	**26,7676**
Greig Seafood	15,7114	16,1894	16,4824	14,2976
Lerøy Seafood group	9,6537	17,9708	20,2971	18,8353
Mowi	21,6879	11,45	13,186	11,97
Austevoll Seafood	4,31	23,8271	25,6165	25,0523
SR sparebank 1	**11,0165**	**5,459**	**6,435**	**7,125**
DNB	11,044	6,266	7,05824	7,044
Sparebank 1 Nord Nor	5,111	1,93174	2,3278	2,398
Jæren Sparebank	6,157	4,63431	5,14218	4,8565
Nordea	23,53	14,03	16,01	11,03
AURSKOG SPAREBANI	6,46854	4,86827	5,19647	5,47

Figure 3.3.1: *P/E values for all companies in the analysis*

After the data is collected, we use summary statistics such as median, average, high, low, 25th percentile and 75th percentile. This can give an overall idea of the data and help identify any outliers or unusual patterns in the data. We decided to use the Median statistic as the main multiple for our evaluation method. We can look at the median statistic as the sector's average multiple for what a companies should be valued at. It is important to consider that there is both systemic and non-systemic risk when buying a stock. A drawback of this evaluation method is that it does not consider the non-systemic risk of the stock, which could be the reason for its current P/E ratio.

Now that we have the median P/E ratio for the sector we multiply this ratio with the net income for the last four quarters to the implied market value for the stock. When this is obtained, we divide the implied market value by the number of shares outstanding resulting in a stock price. The resulting stock price may be over or under the price at the, which will result in either a buy or sell recommendation. There must be at least a 1,5 percent difference in the P/E ratio in order to classify as either a buy or short signal. Within the 10 percent area we consider this as a margin of error and conclude with a Hold signal in this case.



	Q1	Q2	Q3	Q4
OIL				
High	33,70	7,89	-4,24	-0,74
75th Percentile	26,642025	-7,38175	-6,28425	-2,8245
Average	22,01	-21,77	-32,87	-16,45
Median	**22,21**	**-17,90**	**-16,83**	**-5,56**
25th Percentile	15,63625	-23,688	-63,333	-18,339
Low	12,39	-73,23	-77,68	-62,30
FISHERIES				
High	21,69	23,83	25,62	26,77
75th Percentile	15,7114	17,9708	23,303	25,0523
Average	**13,118002**	**17,26954**	**19,777**	**19,3852**
Median	14,2225	16,907	20,2971	18,8353
25th Percentile	9,6537	16,1894	16,4824	14,2976
Low	4,31451	11,4534	13,186	11,9732
BANKS				
High	23,53	14,03	16,01	11,03
75th Percentile	11,037125	6,06425	6,90243	7,10475
Average	10,55	6,20	7,03	6,32
Median	**8,74**	**5,16**	**5,82**	**6,26**
25th Percentile	6,234885	4,6928	5,1557525	5,0098675
Low	5,11	1,93	2,33	2,40

Figure 3.3.2: *Statistical info on the P/E values in the different sectors*

We settled on a 5 percent gain in the next 14 days after the quarter as the threshold in order to classify as a successful upward trending stock. If the multiple analysis result in a stock price which is higher than the price at the time, and the stock proceeds to gain 5 percent after the quarter we classify this a successful buy signal. For a successful short signal, we request a 5 percent loss during the next 14 days.

3.4 The naive forecast

The naive forecast will simply predict no change in stock price for all of the tests.

Quarter	**Equinor**	**SpareBank 1 SR-Bank**	**Salmar**
Q1	Neutral	Neutral	Neutral
Q2	Neutral	Neutral	Neutral
Q3	Neutral	Neutral	Neutral
Q4	Neutral	Neutral	Neutral

Table 3.4.1: The naive forecast will always forecast the last observed value. Therefore it will always predict that there will be no change in the stock price.

CHAPTER

FOUR

RESULTS

4.1 The final model

Our final XGB model trained 6 020 trees and used 5 921 for predictions as this was the point where the validation loss started rising again. Even thought the hyper parameter tuning suggested these results, the model has a lot lower loss on the training set than the validation set suggesting poor generalizability and over-fitting.

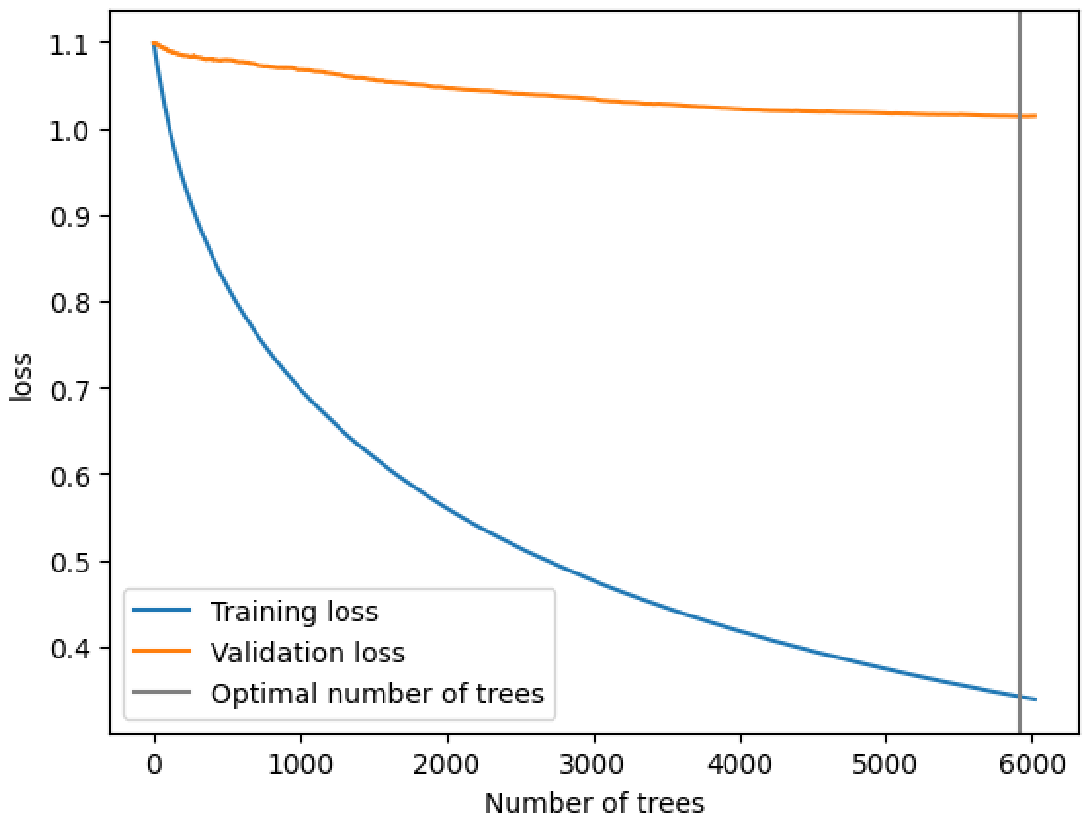


Figure 4.1.1: The loss plotted for the training and validation data set through training.

The model is evaluated based on its accuracy and various other metrics such as precision, recall, and f1-score. The model's overall accuracy is 46.40 percent, which means it can correctly predict the class for 46.40 percent of the test samples. however a better metric to measure the model is the F1-score, of which the weighted average is 43 percent. this is not that great. Another important metric is precision. This measures the probability of a prediction being true. It is for instance more important to to have that a given prediction i likely to be true, rather than missing a prediction, which would be the recall. however a model which routinely misses out on stock changes is bad too. So as a metric for comparison we will look at the F1-score.

	Precision	**Recall**	**F1-score**	**Support**
long	0.34	0.31	0.33	141.00
neutral	0.51	0.59	0.55	296.00
short	0.37	0.28	0.32	160.00
accuracy	0.44			
macro avg	0.41	0.39	0.40	597.00
weighted avg	0.43	0.44	0.43	597.00

Table 4.1.1: Classification rapport final model.

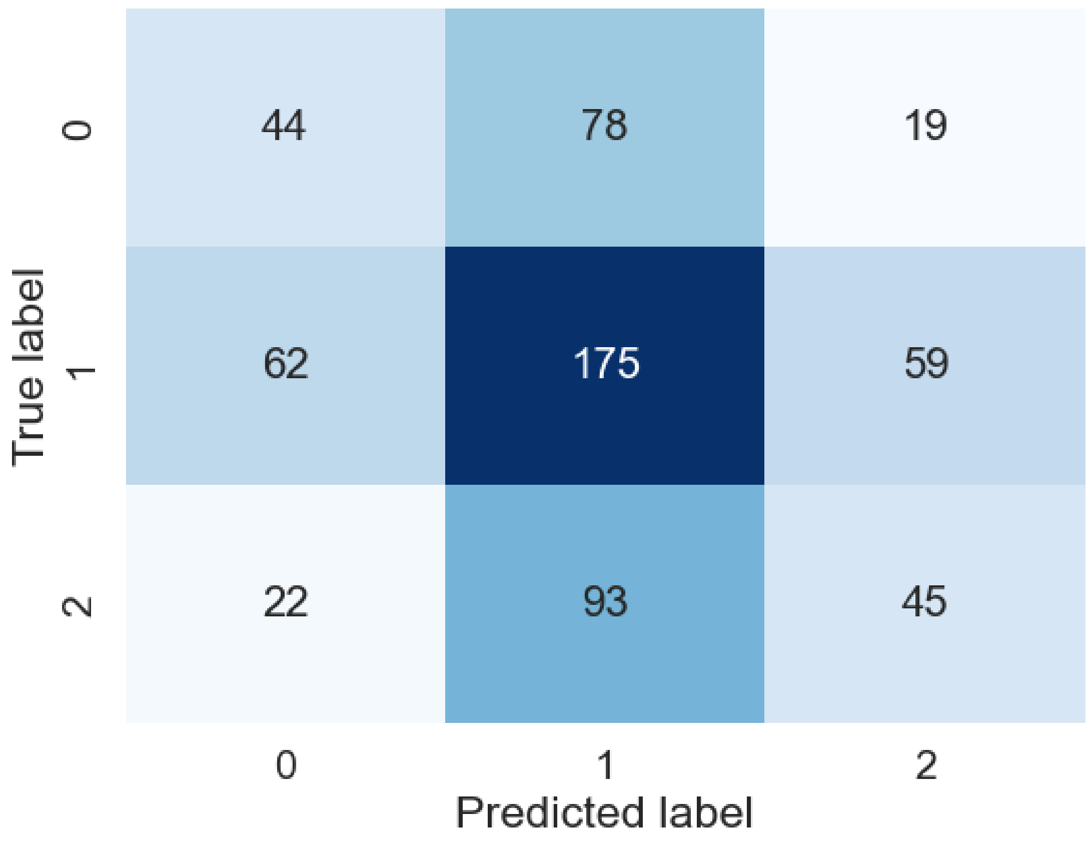


Figure 4.1.2: The confusion matrix from the final model

The classification report presents more detailed information about the model's performance for each class:

4.1.1 Long category

Precision is 0.38, which means that out of all the samples predicted to be long, 38 percent were actually in the long category. Recall is 0.26, indicating that the model identified 26 percent of the actual samples to be long. The F1-score is 0.31, which is a harmonic mean of precision and recall. This single value helps evaluate the model's performance for this specific class.

4.1.2 Neutral category

Precision is 0.51, meaning that out of all the samples predicted as class 1, 51 percent were actually class 1. Recall is 0.68, indicating that the model identified 68% of the actual class 1 samples. The F1-score is 0.58, which is a harmonic mean of precision and recall, providing a single value to evaluate the model's performance for this specific class.

4.1.3 Short category

Precision is 0.36, meaning that out of all the samples predicted as class 2, 36 percent were actually class 2. Recall is 0.25, indicating that the model identified 25 percent of the actual class 2 samples. The F1-score is 0.30, which is a harmonic mean of precision and recall, providing a single value to evaluate the model's performance for this specific class.

Reflecting on the results, the model's performance is suboptimal, with an overall accuracy of only 46.40 percent. The model performs best on class 1 and struggles the most with class 2. The low precision, recall, and f1-scores for each class indicate that the model has difficulty in distinguishing between the classes. This is despite our efforts to reduce bias towards the majority category.

4.1.4 Feature importance

The tree most important features ranked by average gain increase is week of year, month of year and the Symbol. The next most important features are the 10 principal components which together stands for 99 percent of the variance in our numerical variables. After that we get a jumble of various othe principal components and Cyclical variables with gradually lower scores.

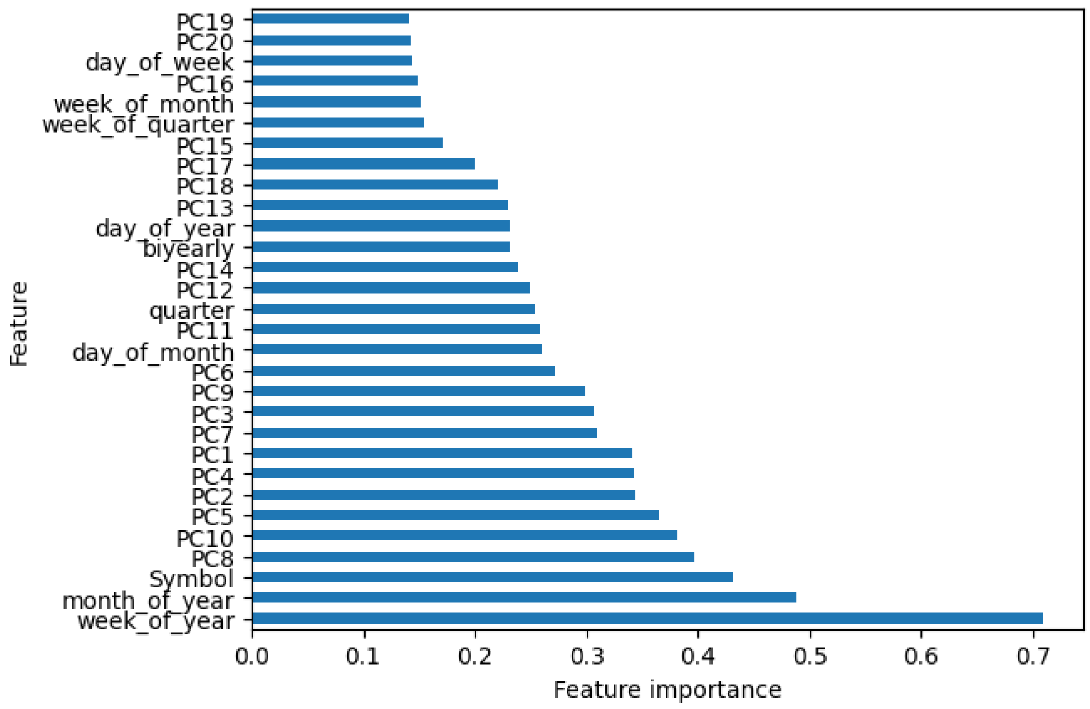


Figure 4.1.3: The gain of each input to the final model.

4.1.5 Data subset

The XGB model fared significantly worse on the sub set of data for comparing with the technical analysis than the entire test set. The specific prediction results are reported below. The overall accuracy has dropped to 41.67 percent, which is lower than the previous result of 46.40 percent. This means the model correctly predicts the class for 41.67 percent of the samples in this new test set.

Comparing these results with the previous ones, the model's performance has worsened. Its overall accuracy has decreased, and it's failing to predict the long and short categories correctly. This could be due to the smaller size of the test set, which has only 12 samples compared to the previous 597 samples. A smaller test set might not be representative of the model's true performance. Additionally, the model's performance on class 1 remains relatively consistent between both test sets.

	precision	**recall**	**f1-score**	**support**
long	0.00	0.00	0.00	2.00
neutral	0.62	0.56	0.59	9.00
short	0.00	0.00	0.00	1.00
accuracy	0.42			
macro avg	0.21	0.19	0.20	12.00
weighted avg	0.47	0.42	0.44	12.00

Table 4.1.2: Classification rapport final model on subset of data.

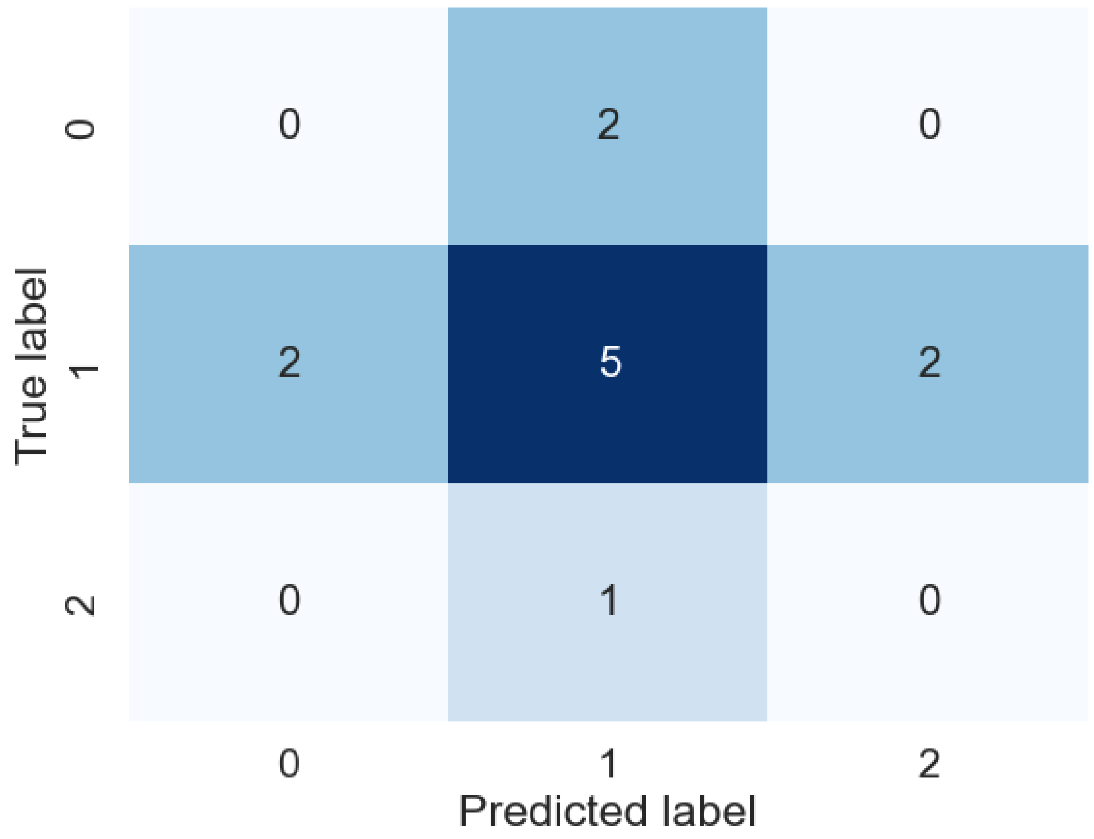
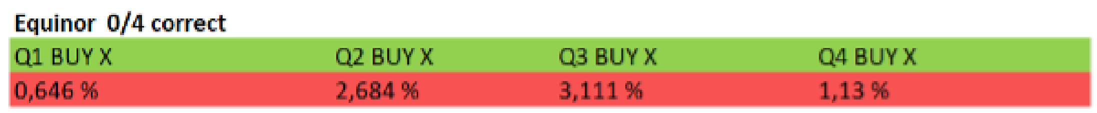


Figure 4.1.4: The confution matrix of XGB model predictions on subset of data.

4.2 Technical analysis

4.2.1 Equinor

For the first quarter analysis results we achieve a stock price of 320,61 in the multiple analysis. This is substantially higher than what the stock price in Q1 2020 is for Equinor, which closed at 177.95 on January 2nd, 2020. From the analysis we can safely state that this is a buy signal. However, if this signal is correct, we must look at the closing prices to determine if this resulted in a gain for the stock. The closing price on January 15 2020 is 179.10 which is a 1.15Kr increase and a 0.646 percent increase in the sin the stock price. This increase is however not high enough to meet our 5 percent threshold for concluding a correct buy signal. later in the year we receive negative P/E values for all the peer stocks and Equinor we cannot apply the P/E evaluation method. The P/E ratios are negative because of Covid-19 and how it affected the global demand for oil and gas products. This is because it results in negative stock prices. However, we can still compare the P/E ratios without calculating the estimated stock price. This will result in a buy/short/hold signal, and we report the corresponding signal results with the real-world price movements in a table (This uses information which is in references under 6.1)

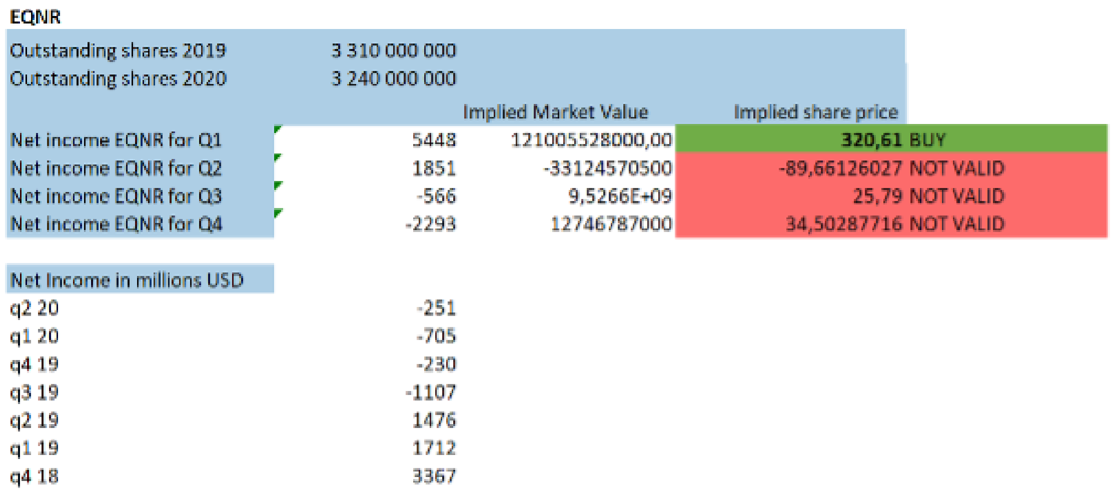


Equinor 0/4 correct

Q1 BUY X	Q2 BUY X	Q3 BUY X	Q4 BUY X
0,646 %	2,684 %	3,111 %	1,13 %

Figure 4.2.1: Buy or Short signals for Equinor

In the case of Equinor we see that the multiple analysis did not correctly predict any of the four periods.



EQNR

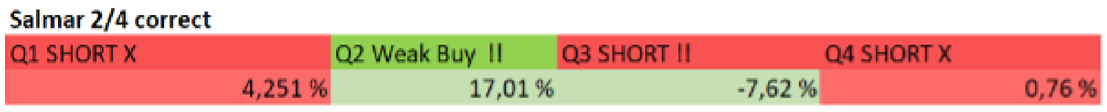
Outstanding shares 2019	3 310 000 000		
Outstanding shares 2020	3 240 000 000		
		Implied Market Value	Implied share price
Net income EQNR for Q1	5448	121005528000,00	320,61 BUY
Net income EQNR for Q2	1851	-33124570500	-89,66126027 NOT VALID
Net income EQNR for Q3	-566	9,5266E+09	25,79 NOT VALID
Net income EQNR for Q4	-2293	12746787000	34,50287716 NOT VALID

Net Income in millions USD	
q2 20	-251
q1 20	-705
q4 19	-230
q3 19	-1107
q2 19	1476
q1 19	1712
q4 18	3367

Figure 4.2.2: Equinor price prediction.

4.2.2 Salmar

Decided to use AVG P/E ratio instead of the median in this case since we only have 4 other peers. In contrast to the oil and gas industry, the fishing industry was less affected in a business sense by Covid 19. There is a special case wherein Q2 we see a Higher P/E for the sector than Salmar is priced at. However, this is only a small marginal difference (2.1%), resulting in a weak buy signal. This is Salmar's best quarter and there is a substantial gain in the next 14 days. One of the signals also predicts a short opportunity in Q3 resulting in a loss of 7,62 percent to the stock price and this qualifies as a correct prediction. (This uses information that is in references under 6.2)

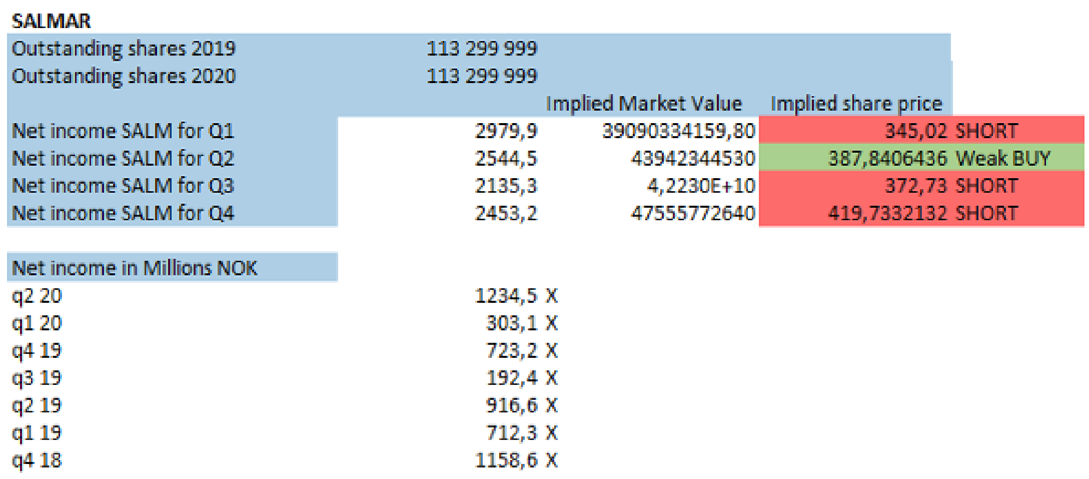


Salmar 2/4 correct

Q1 SHORT X	Q2 Weak Buy !!	Q3 SHORT !!	Q4 SHORT X
4,251 %	17,01 %	-7,62 %	0,76 %

Figure 4.2.3: Buy or Short signals for Salmar.

In the case of Salmar we see that the multiple analysis correctly predicted two of the four periods. (highlighted with green background)



SALMAR

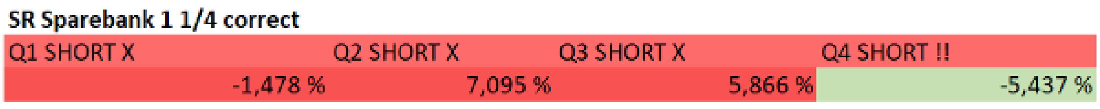
Outstanding shares 2019	113 299 999		
Outstanding shares 2020	113 299 999		
		Implied Market Value	Implied share price
Net income SALM for Q1	2979,9	39090334159,80	345,02 SHORT
Net income SALM for Q2	2544,5	43942344530	387,8406436 Weak BUY
Net income SALM for Q3	2135,3	4,2230E+10	372,73 SHORT
Net income SALM for Q4	2453,2	47555772640	419,7332132 SHORT

Net income in Millions NOK	
q2 20	1234,5 X
q1 20	303,1 X
q4 19	723,2 X
q3 19	192,4 X
q2 19	916,6 X
q1 19	712,3 X
q4 18	1158,6 X

Figure 4.2.4: Salmar price prediction

4.2.3 SR Sparebank 1

In SR Sparbank 1's case, we cannot rely on the P/E Ratio multiple to produce accurate implied share price estimates. The main reason for this is that this stock pays out large dividends, thus resulting in people buying the stock for this reason. Paying out dividends removes money from the company resulting in less ability to grow and expand. In this case we see that by not paying out dividends would result in a higher stock price. We will therefore be resorting to comparing P/E ratios and giving signals based on the positive or negative difference to the median. (This uses information that is in references under 6.3)



SR Sparebank 1 1/4 correct

Q1 SHORT X	Q2 SHORT X	Q3 SHORT X	Q4 SHORT !!
-1,478 %	7,095 %	5,866 %	-5,437 %

Figure 4.2.5: Buy or Short signals for SR Sparebank 1

In the case of SR Sparebank 1 we see that the multiple analysis only correctly predicted one of the four periods. (Highlighted with green background)

	Precision	**Recall**	**F1-score**	**Support**
long	0.20	0.30	0.24	2
neutral	0	0	0	9
short	0.2857	1	0.44	1
accuracy	0.25			
macro avg	0.162	0.43	0.226	12
weighted avg	0.057	0.133	0.0766	12

Table 4.2.1: Classification rapport final model.

Banks share prices typically move slower in comparison to other sectors. This may skew the result of us trying to record a 5 percent move in either direction. Since banks often have a low P/E ratio it may take a larger shift in earnings to move the share price than in other sectors.

4.3 The naive forecast

Comparing the two methods, the naive forecast method has a higher accuracy (50% vs. 46.40%) on this test set. However, it only predicts the neutral category and completely ignores the moving categories, which results in a recall of 100 percent for the neutral category but 0 percent for the moving categories.

While the XGB classification model has a slightly lower accuracy, it does make predictions for all three categories. The model's performance is better across the board for the individual categories compared to the naive forecast method, as evidenced by the higher precision, recall, and F1-scores for each class, except for the recall of the neutral category.

Based on these comparisons, the naive forecast method might not be a good choice if it's essential to predict all three classes. The previous classification model, while having a lower overall accuracy, provides a more balanced performance across the classes.

	precision	**recall**	**f1-score**	**support**
long	0.00	0.00	0.00	141.0
neutral	0.50	1.00	0.66	296.0
short	0.00	0.00	0.00	160.0
accuracy	0.50			
macro avg	0.17	0.33	0.22	597.0
weighted avg	0.25	0.50	0.33	597.0

Table 4.3.1: Classification rapport naive forecast.

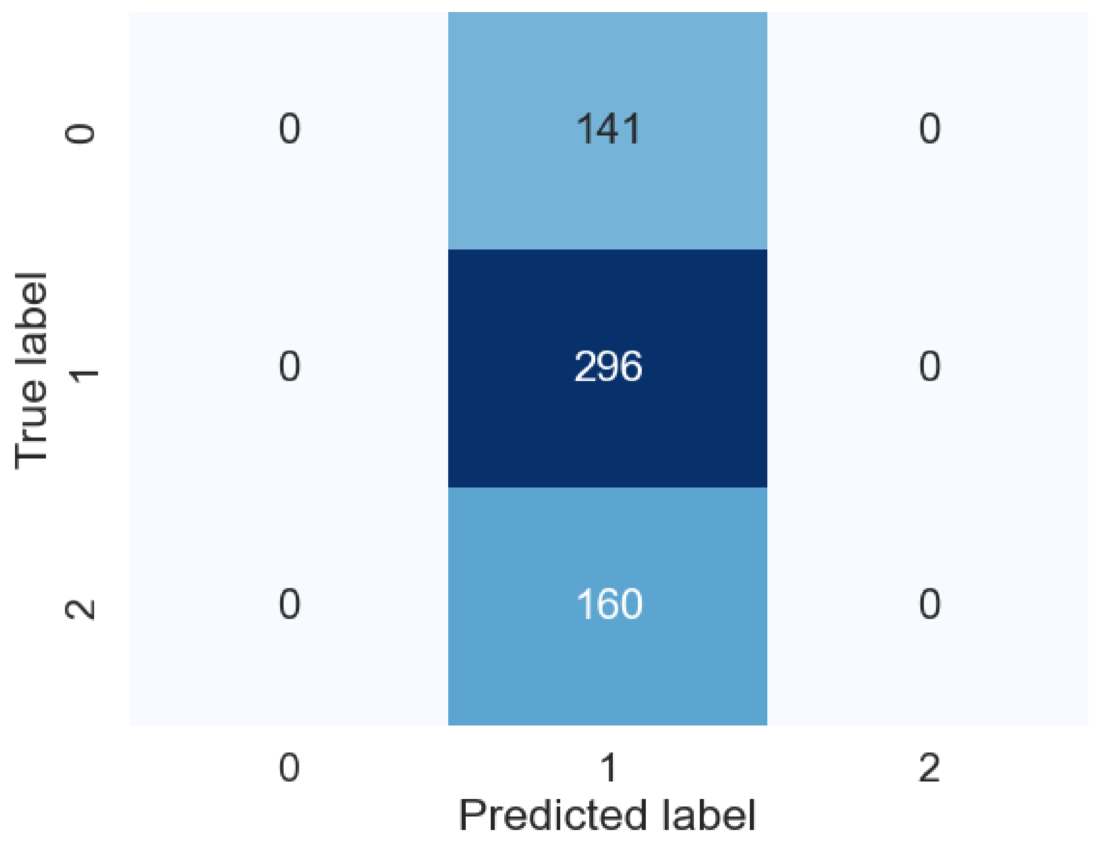


Figure 4.3.1: The confusion matrix of the naive forecast on all test data

4.3.1 Data subset

The naive forecast method has a much higher accuracy (75% vs. 41.67%) on the smaller test set compared to the XGB model. However, similar to the previous comparison, the naive forecast method only predicts the neutral category and completely ignores the moving categories, resulting in a recall of 100 percent for the neutral category but none for for the moving categories.

The previous classification model has a lower overall accuracy on this test set, but it attempts to predict all three categories, albeit with poor performance for the moving categories.

Given these results, if the primary goal is to maximize overall accuracy, the naive forecast method might be a better choice for this specific smaller test set. However, it's important to consider that it only predicts the neutral category and fails to predict the moving categories. If predicting all classes is essential, the previous classification model would be more appropriate, but improvements to the model are necessary for better performance.

	precision	**recall**	**f1-score**	**support**
long	0.00	0.00	0.00	2.00
neutral	0.75	1.00	0.86	9.00
short	0.00	0.00	0.00	1.00
accuracy	0.75			
macro avg	0.25	0.33	0.29	12.00
weighted avg	0.56	0.75	0.64	12.00

Table 4.3.2: Classification rapport naive forecast on subset.

CHAPTER

FIVE

DISCUSSION

5.1 Technical analysis results

The results of the analyses are inconclusive since we only had 3 correctly predicted periods. However, if we reduced the threshold from correctly proving that a signal is correct from a 5 percent change to 1 percent change, we would have correctly predicted 6/12 signals which corresponds to a 50%-win rate. Since we were looking for credible results, we decided on 5 percent movement and therefore 1 percent is too little to even consider as a correct signal. This would fall inside the margin of error and not classify as a positive or negative trend. A threshold of 2.5 percent would correctly predict 5/12 periods.

A $3/12 = 25\%$ success rate is not something to be very proud of. But there are things that could skew our results. This is for example that we assume that the quarterly net income will stay steady and not vary greatly in the future. At the point of the first analysis, we do not know about covid-19, so we would assume that net income will remain steady into the future. This is especially relevant in the case of the oil companies' evaluation due to the fact the oil prices were negative at one point in 2020. This will impact the results we get since the underlying stock is not stable due to the macroeconomic situation at the time.

This evaluation method only looks at the company's net income and determines a fair market price for a company in that sector. It does not consider the location of the company, where it does business, political climate location and what goods this company is supplying and who is buying. This is especially relevant in today's world since we have never been so divided in terms of sanctions, trade bans, and war.

5.2 Comparing the results

The technical analysis predicted change in the stock price to a much higher degree compared to the XGB model, which had a majority bias. Thus making it perform worse than the XGB model in quantitative measures. However, on could argue that predicting change and not getting any is preferred to a model which mostly predicts no change, even when the stock price changes.

None of the methods could however beat the naive forecast on accuracy, although our XGB model did perform better on precision, recall and f1-score, which is unsurprising as the model was optimized for F1-score. This suggests that although it is less accurate it is better at identifying the individual categories, compared to the naive forecast.

The XGB model is more accurate than the technical analysis. it beats it by 68 percent, or 17 percentage points on the accuracy metric. If this is due to the small test size hard to say, but we know that the XGB performed worse on the smaller subset than the entire testing set. This suggests that the sub set is not representative of ether method's true performance.

5.3 Interpretability

When comparing an XGB classification model to multiple analyses, an essential factor to consider is interpretability. P/E ratios are simple to understand and explain, allowing us to grasp why a stock's price might rise or fall in the future. In contrast, the machine learning model is more challenging to comprehend, as it's not clear why the model predicts a particular stock movement.

Although it's possible to visualize individual trees in the XGB model, thousands of them exist, and only 80% are active at any given time. This leaves us with feature importance scores based on principal components from a PCA, which itself is a linear transformation of numerous input variables that have been individually scaled. As a result, the XGB model functions as a black box, requiring users to trust the algorithm blindly when making investment decisions.

Considering these factors, it's not unreasonable to think that the easy-to-understand P/E method might be the preferred choice for investment decision-making.

5.4 Future work

All models presented here have been trained on a laptop. Access to more computing power would enable us to do better hyperparameter tuning. This might just lead to incremental improvements in the models, however, this is currently not known.

As with all financial mining and big data ventures. More data, more kinds of data, and higher quality data would improve the quality of our models.

12 samples is quite a small set of tests, and more tests would make the results bear greater weight.

Because the validation data is quite similar and among the training data there is a risk of overfitting the model even when using early stopping and limiting the number of trees used in prediction. If there is a better way to split the data this might change the conclusions in this paper.

In regards to prediction with P/E ratios we may have greater results when looking at a larger time horizon. This will give the stock more time to adjust to the sector median P/E. This implies that stock market prices may not act rationally from day to day, but rather when observed in a larger time horizon.

CHAPTER
SIX

CONCLUSIONS

Our results suggest that a XGB model is a better predictor of 5 percent stock returns over 14 days than multiple analysis using P/E ratios. This is however, unsurprising as the XGB model was carefully optimized for just this problem on the stocks in question rendering it useless in predicting other stocks, return thresholds or time intervals. The multiple analysis is a more generalized method which can be used on most stocks for most time intervals. Given a different set test parameters one would have to create an entirely new XGB model, but in our testing this would take around 15 minutes given that all data is on hand.

Although neither method could beat the naive forecast in accuracy. The XGB model did outperform it in other metrics such as precision, recall and F1-score.

Our results, however, are not conclusive because of the small test size of only 12 samples. A more rigorous test size, including both more test- parameters and samples, would be necessary in order to conclusively say that one method is better than the other.

REFERENCES

[1] Mathias Grotøy Bjørgum and Aasmund Groven Lindtveit. *Cleaning Up Intraday Noise: An NLP-Based Approach to Predicting Directional Movements on Intraday Stock Data.* 2023.

[2] Trevor J Hastie, Robert J Tibshirani, and Jerome Friedman. *The elements of statistical learning : data mining, inference, and prediction.* 2nd ed. Springer series in statistics. New York: Springer Science + Business Media, 2009. ISBN: 978-0-387-84857-0.

[3] Gareth James et al. *An introduction to statistical learning : with applications in R.* Second edition. Springer texts in statistics. New York: Springer, 2021. ISBN: 978-1-07-161417-4.

[4] Tianqi Chen and Carlos Guestrin. "XGBoost". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, Aug. 2016. DOI: 10.1145/2939672.2939785.

[5] K. H. Esbensen, B. Swarbrick, and F. Westad. *Multivariate Data Analysis.* CAMO Software AS, 2018, pp. 69–99.

[6] J. Chen. "*What Is a Multiple? With Examples, Such as P/E Multiple*". In: *Investopedia* (Dec. 2020). URL: https://www.investopedia.com/terms/m/multiple.asp.

[7] T Smith. " *Multiples Approach*". In: *Investopedia* (July 2022). URL: https://www.investopedia.com/terms/m/multiplesapproach.asp.

[8] *Oslo stock exchange.* 2023. URL: https://titlon.uit.no/.

6.1 Equinor refrences

Equinor. (2019). Fourth quarter 2018 – Financial statements and review (Q4 2018). Equinor.https://cdn.equinor.com/files/h61q9gi9/global/b90c60686fbc965ff4d77a60a69b8
2018-financial-statements-and-review-equinor.pdf

[1]Equinor. (2019). First quarter 2019 – Financial statements and review (Q1 2019). Equinor.https://cdn.equinor.com/files/h61q9gi9/global/ed5d48e13245d06ebe23270e36ba
2019-financial-statements-and-review-equinor.pdf

Equinor. (2019). Second quarter 2019 – Financial statements and review (Q2 2019). Equinor.https://cdn.equinor.com/files/h61q9gi9/global/916e56a59d06186c730b06e85bfd
2019-financial-statements-and-review-equinor.pdf

Equinor. (2019). Third quarter 2019 – Financial statements and review (Q3 2019). Equinor.https://cdn.equinor.com/files/h61q9gi9/global/f97aa2ee33bcf2f8c65cdbb1e60b414013c9
2019-financial-statements-and-review-equinor.pdf

Equinor. (2020). Fourth quarter 2019 – Financial statements and review (Q4 2019). Equinor.https://cdn.equinor.com/files/h61q9gi9/global/6bca14b946c068c519da9132535d4d5db1
2019-financial-statements-and-review-equinor.pdf

Equinor. (2020). First quarter 2020 – Financial statements and review (Q1 2020). Equinor.https://cdn.equinor.com/files/h61q9gi9/global/16009775622bef9c5e328120481eb3f09dd
2020-financial-statements-and-review-equinor.pdf

Equinor. (2020). Second quarter 2020 – Financial statements and review (Q2 2020). Equinor.https://cdn.equinor.com/files/h61q9gi9/global/e58f748a7c0fdf402fb898f038e8622b1a8fa
statements-and-review-q2-2020-equinor.pdf

6.2 Salmar refrences

Salmar. (2019). QUARTERLY REPORT-Fourth quarter 2018 – Financial statements and review (Fourth Quarter 2018). Salmar. https://hugin.info/138695/R/2234948/879657.pdf

Salmar. (2019). QUARTERLY REPORT-First quarter 2019 – Financial statements and review (First Quarter 2019). Salmar. https://hugin.info/138695/R/2244922/886598.pdf

Salmar. (2019). QUARTERLY REPORT-Second quarter 2019 – Financial statements and review (Second Quarter 2019). https://ml-eu.globenewswire.com/Resource/Download/9
fff4-42e4-a83c-62680447452c

Salmar. (2019). QUARTERLY REPORT-Third quarter 2019 – Financial statements and review (Third Quarter 2019). https://ml-eu.globenewswire.com/Resource/Download/ca
2ecb-4606-90c4-bac5078b1848

Salmar. (2020). QUARTERLY REPORT-Fourth quarter 2019 – Financial statements and review (Fourth Quarter 2019). Salmar. https://ml-eu.globenewswire.com/Resource/Dov
656b-4be8-a261-da462e3d0b0a

Salmar. (2020). QUARTERLY REPORT-First quarter 2020 – Financial statements and review (First Quarter 2020). Salmar. https://ml-eu.globenewswire.com/Resource/Download
2d61-4fe4-bda2-b2a2c8b30f35

Salmar. (2020). QUARTERLY REPORT-Second quarter 2020 – Financial statements and review (Second Quarter 2020). Salmar. https://ml-eu.globenewswire.com/Resource/Dov
4f06-45ee-87f6-94674d4c90a2

6.3 SR sparebank 1 refrences

SR Sparebank 1. (2019). Fourth Quarter 2018 (Q4 2018). SR Sparebank 1. `https://www.sparebank1.no/content/dam/SB1/bank/sr-bank/english/about-us/investor/reports/Quarterly_report_SRBANK_Q4_2018.pdf`

SR Sparebank 1. (2019). First Quarter 2019 (Q1 2019). SR Sparebank 1. `https://www.sparebank1.no/content/dam/SB1/bank/sr-bank/om-oss/Investor/Rapporter/2019/q1/Quarterly_report_SRBANK_Q1_2019.pdf`

SR Sparebank 1. (2019). Second Quarter 2019 (Q2 2019). SR Sparebank 1. https://www.sparebank1.no/content/dam/SB1/bank/sr-bank/om-oss/Investor/Rapporter/2019/q2/Q

SR Sparebank 1. (2019). Third Quarter 2019 (Q3 2019). SR Sparebank 1. `https://www.sparebank1.no/content/dam/SB1/bank/sr-bank/om-oss/Investor/`

Rapporter/2019/q3/Quarterly_report_Q3_2019.pdf

SR Sparebank 1. (2020). Fourth Quarter 2019 (Q4 2019). SR Sparebank 1. https://www.sparebank1.no/content/dam/SB1/bank/sr-bank/om-oss/Investor/Rapporter/2019/Q4/Quarterly_report_Q4_2019.pdf

SR Sparebank 1. (2020). First Quarter 2019 (Q1 2020). SR Sparebank 1. https://www.sparebank1.no/content/dam/SB1/bank/sr-bank/om-oss/Investor/Rapporter/2020/Q12020/QuarterlyreportSRBANKQ12020.pdf

SR Sparebank 1. (2020). Second Quarter 2019 (Q2 2020). SR Sparebank 1. https://www.sparebank1.no/content/dam/SB1/bank/sr-bank/om-oss/Investor/Rapporter/2020/Q2_2020/Quarterl_report_SRBANK_Q2_2020.pdf

APPENDICES

A - GITHUB REPOSITORY

All code used for the models of this document are included in the Github repository linked below.

Github repository link

- `https://github.com/WildCry/Semesteroppgave-IF420-v23.git`

B - THE TITLON DATA SET

Below is an overview of the variables included in the Titlon data set along with the explanation given on their website.

- **Date** Date
- **SecurityId** Security id
- **CompanyId** Company id
- **Symbol** Symbol
- **ISIN** ISIN
- **Name** of associated equity
- **BestBidPrice** Best bid price
- **BestAskPrice** Best ask price
- **Open** Open price
- **High** Higest observed price during trading day
- **Low** Lowest observed price during trading day
- **Close** Closing price
- **OfficialNumberOfTrades** Official number of trades
- **OfficialVolume** Official volume
- **UnofficialNumberOfTrades** Unofficial number of trades
- **UnofficialVolume** Unofficial volume
- **VolumeWeightedAveragePrice** Volume weighted average price
- **Price** Price
- **AdjustedPrice** Adjusted price
- **Dividends** Last payed dividend
- **LDividends** Lagged dividend
- **CorpAdj** Corp adjustments (splits etc.)
- **DividendAdj** Dividend adjustment factor
- **Currency** Currency. Currently allways NOK
- **Description** Description
- **CountryCode** Country code
- **SumAnnualDividends** Sum annual dividends
- **NumberOfShares** Number of shares issued (not free float adjusted)
- **CompanyOwnedShares** Company owned shares
- **OutstandingShares** Outstanding shares
- **Exchange** Exchange
- **NOKPerForex** Exchange rate. Currently allways 1
- **mktcap** Market capitalization

- **OSEBXmktshare prevmnth** Prevoius month market share in OSEBX index
- **OSEBXAlpha prevmnth** Prevoius month alpha relative to OSEBX index
- **OSEBXBeta prevmnth** Prevoius month beta relative to OSEBX index
- **SMB** SMB Fama-French factor
- **HML** HML Fama french factor
- **LIQ** LIQ Pastor-Stambaugh factor
- **MOM** MOM Fama-French factor
- **DividendPriceRatio** Dividend price ratio
- **lnDeltaP** log return of the adjusted price (CorpAdj and DividendAdj)
- **lnDeltaOSEBX** log return of the OSEBX index
- **lnDeltaOBX** log return of the OBX index
- **NOWA DayLnrate** Log difference of Norwegian Overnight Weighted Average rate from the norwegian central bank after 2013. NIBOR before that.
- **bills 3month Lnrate** 3 months Norwegian Goverment Bills, from the bondindex table, CloseYield field Sector Sector
- **IN OSEBX** True if the stock is in the OSEBX index at the moment
- **Equity** Amount of Equity
- **Debt** Amount of Debt
- **Earnings** Total earnings
- **debt ratio** Debt ratio
- **PE** Price/Earnings

C - XGB DETAILED RESULTS

- EQNR, 2020-01-02
  - true: neutral, predicted: short
- EQNR, 2020-04-01
  - true: neutral, predicted: neutral
- EQNR, 2020-07-01
  - true: neutral, predicted: neutral
- EQNR, 2020-10-01
  - true: neutral, predicted: neutral
- SALM, 2020-01-02
  - true: neutral, predicted: long
- SALM, 2020-04-01
  - true: neutral, predicted: long
- SALM, 2020-07-01
  - true: long, predicted: neutral
- SALM, 2020-10-01
  - true: long, predicted: neutral
- SRBANK, 2020-01-02
  - true: neutral, predicted: neutral
- SRBANK, 2020-04-01
  - true: neutral, predicted: neutral
- SRBANK, 2020-07-01
  - true: neutral, predicted: short
- SRBANK, 2020-10-01
  - true: short, predicted: neutral