GROUP WORK PROJECT #3
GROUP NUMBER: 6127

MScFE 632: Machine Learning in Finance

FULL LEGAL NAME	LOCATION (COUNTRY)	EMAIL ADDRESS	MARK X FOR ANY NON-CONTRIBUTING MEMBER
Yang Xue	China	KierenXue@gmail.com	
Carlos García Guillamón	Spain	carlosgguil@outlook.es	
Ebenezer Yeboah	Ghana	ebenezeryeboah46@gmail.com	

Statement of integrity: By typing the names of all group members in the text boxes below, you confirm that the assignment submitted is original work produced by the group (excluding any non-contributing members identified with an "X" above).

Team member 1	**Yang Xue**
Team member 2	**Carlos García Guillamón**
Team member 3	**Ebenezer Yeboah**

Use the box below to explain any attempts to reach out to a non-contributing member. Type (N/A) if all members contributed.
Note: You may be required to provide proof of your outreach to non-contributing members upon request.

Step 2: Outline

The outline for this deliverable is the following one:

- Issue 1: Optimizing Hyperparameters
    - Technical
        - Basics
        - Importance of hyperparameters in model performance
        - Representation of hyperparameters in different models
        - Optimization techniques
        - Evaluation of Hyperparameter Optimization
        - Metrics for Hyperparameter Optimization
        - Visualization Techniques
    - Non-technical
        - Introduction to the Concept
        - Impact on Model Performance
        - Optimization Methods
        - Real-World Examples
        - Case Study
- Issue 2: Optimizing the Bias-Variance Tradeoff
    - Technical
        - Basics
        - Illustration
        - Mathematical interpretation
        - What's best? Good practices
    - Non-technical
        - Discussion
        - Conclusions
- Issue 3: Applying Ensemble Learning - Bagging, Boosting, or Stacking
    - Technical
        - Process
        - Illustration
        - Metrics and Evaluation
        - Journal and Reference
    - Non-technical
        - Description
        - Basics
        - Advantages
        - Disadvantages
        - Discussion

Step 3: Issues

Issue 1: Optimizing hyperparameters

Author: Ebenezer Yeboah

Reviewer: Carlos Garcia Guillamon

Technical

- **Basics:**

  Hyperparameters are crucial components in the realm of machine learning models, distinguished from internal parameters as they are set prior to the training process and are not learned from the data. These parameters significantly influence the model's performance, generalization capabilities, and computational efficiency. For instance, in a neural network, hyperparameters include the number of layers, the number of neurons per layer, and the learning rate. Choosing the right hyperparameters can greatly enhance a model's accuracy and robustness while also optimizing training time and resource usage.

- **Importance of Hyperparameters in Model Performance**

  Proper tuning of hyperparameters can significantly improve the accuracy, efficiency, and generalization ability of a model. Also, poorly chosen hyperparameters can lead to suboptimal models that either underfit or overfit the data. The process of selecting the best hyperparameters is known as hyperparameter optimization or tuning, and it involves finding the set of hyperparameters that results in the best performance according to a predefined metric, such as accuracy, precision, or F1-score. Effective hyperparameter tuning is essential for building robust and reliable machine learning models.

- **Representation of Hyperparameters in Different Models**

  Support Vector Machines (SVM):

  ➢ C (Regularization parameter): This hyperparameter balances the trade-off between minimizing training error and avoiding overfitting. A lower value of C results in a softer margin that permits some misclassifications, while a higher value of C seeks a stricter margin with fewer misclassifications.
  ➢ Kernel type: It determines the function used to map the data into a higher-dimensional space, with common options being linear, polynomial, and radial basis function (RBF) kernels.
  ➢ Gamma (for RBF kernel): This defines how far the influence of a single training example reaches, with low values indicating a far-reaching influence and high values indicating a closer influence. A low value of gamma means that the influence is far, making the decision boundary smoother, while a high value of

gamma means that the influence is close, resulting in a more complex decision boundary.

Linear Discriminant Analysis (LDA):

➢ Number of components: This specifies the number of linear discriminants to be used for dimensionality reduction, which can enhance the performance of subsequent classifiers.

Neural Networks:

➢ Learning rate: This step size used by the optimization algorithm to update the model weights during training.
➢ Number of layers: It determines the depth of the neural network, influencing its capacity to learn complex patterns.
➢ Number of neurons per layer: This defines the width of each layer, affecting the model's ability to represent the input data.
➢ Batch size: The number of training examples used to calculate the gradient in each iteration, impacting the stability and speed of the training process.

- **Optimization Techniques:**
  - ➢ Grid Search: Grid Search is a brute-force method for hyperparameter optimization that systematically works through multiple combinations of parameter values. It builds a model for every combination of hyperparameters specified and evaluates each model's performance to determine the best parameter set.
  - ➢ Random Search: Random Search, unlike Grid Search, selects random combinations of hyperparameters to train and evaluate models. This method is less exhaustive but can be more efficient as it does not evaluate all possible combinations.
  - ➢ Bayesian Optimization: Bayesian Optimization uses probabilistic models to predict the performance of hyperparameter combinations and choose the next set of parameters to evaluate. It focuses on exploring the parameter space intelligently rather than exhaustively.

- **Evaluation of Hyperparameter Optimization:**
  - ➢ Cross Validation: Cross-validation is a technique for assessing how well a model generalizes to an independent dataset. It involves dividing the data into multiple parts, training the model on some of these parts, and evaluating it on the others. This procedure is repeated several times, and the results are averaged to give an overall measure of the model's performance.
  - ➢ Validation and Test Sets: Data is divided into three segments: training, validation, and test sets. The training set is used to build the model, the validation set is

utilized to fine-tune hyperparameters, and the test set is employed to assess the model's final performance.

- **Metrics for Hyperparameter Optimization:** That of classification and regression will be described.
    - ➢ **Classification Metrics:**
    - ❖ Accuracy: The proportion of correctly classified instances among the total instances.
      $Accuracy\ = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ predictions}$.
    - ❖ Precision: The proportion of true positive instances among the predicted positives.
      $Precision\ = \frac{True\ Positives}{True\ Positives + false\ positives}$.
    - ❖ Recall: The proportion of true positive instances among the actual positives.
      $Recall\ = \frac{True\ positives}{True\ positives + false\ Negatives}$.
    - ❖ F1 -score: This can be described as the mean of the precision and recall.
      $F1score\ =\ 2 \times \frac{Precision \times Recall}{Precision + Recall}$.
    - ❖ ROC-AUC

    - ➢ **Regression Metrics:**
    - ❖ Mean Absolute Error: The average absolute difference between predicted and actual values.
    - ❖ Mean Squared Error: The average squared difference between predicted and actual values.
    - ❖ R-Squared: The proportion of variance in the target variable explained by the model.

❖ **Visualization Techniques**
- ➢ Learning Curves: Learning curves show the model's performance on the training and validation sets as a function of the number of training samples. They help identify whether the model is overfitting or underfitting.
- ➢ Confusion Matrix: The confusion matrix provides a summary of the prediction results on the classification problem. It shows the number of correct and incorrect predictions broken down by each class.
- ➢ ROC Curve - AUC: The ROC curve is a graphical representation of the true positive rate versus the false positive rate. The AUC provides a single metric that summarizes the model’s overall ability to distinguish between classes across all possible thresholds.

Non-technical

- **Introduction to the Concept:**
  - ➢ Simple Explanation of What Hyperparameters Are:

    Hyperparameters are settings in machine learning algorithms that you decide before training the model. Think of them as the dials and switches on a washing machine. Just like you set the washing machine to the right mode depending on the type of clothes you are washing, hyperparameters need to be set correctly for a machine learning model to perform well. They control how the model learns and processes the data.

  - ➢ Importance of choosing the Right Hyperparameters

    Choosing the right hyperparameters is important for building an effective and efficient machine learning model. Correctly tuned hyperparameters can lead to a model that makes accurate predictions. On the other hand, poorly chosen hyperparameters can cause the model to perform poorly, either by not learning enough from the data which can be described as underfitting or by learning too much, including noise and irrelevant details which can be described as overfitting. Therefore, optimizing hyperparameters is essential to ensure that the model generalizes well and makes reliable predictions.

- **Impact on Model Performance:**
  - ➢ How Hyperparameters Affect the Outcome of the Model:
    They can affect different aspects of learning, such as the speed of learning, the complexity of the model and its ability to generalize to new data. For example, in a neural network, the learning rate determines how quickly or slowly the model adjusts its parameters.
  - ➢ Hyperparameter Choices:
    Learning Rate in Neural Networks: A well-chosen learning rate helps the model converge to a good solution efficiently without overshooting the optimal values.
    Regularization Parameter in SVM: A balanced regularization parameter avoids overfitting by penalizing overly complex models while still capturing the important patterns in the data.
    Number of Neurons in a Neural Network: An appropriate number of neurons ensures that the model has enough capacity to learn the patterns without becoming too complex and prone to overfitting.

- **Optimization Methods**
  Grid Search
  Random Search
  Bayesian Optimization

Importance of Experimentation and Iteration

Experimenting with different hyperparameters is essential because there isn't a one-size-fits-all solution in machine learning. Every dataset is unique, and what works well for one dataset might not work for another.

Iteration is the process of repeatedly trying different combinations, learning from each attempt, and gradually getting closer to the best settings. By experimenting and iterating, you can discover the optimal hyperparameters that help the model perform its best, ensuring it makes accurate predictions and provides valuable insights.

- **Real-World Examples**

Netflix Recommendation System:
Netflix uses machine learning to recommend movies and TV shows to its users. By optimizing the hyperparameters of their recommendation algorithms, Netflix has significantly improved the accuracy of its recommendations. This optimization has led to higher user satisfaction and increased viewer engagement. Through techniques like grid search and Bayesian optimization, Netflix fine-tuned its models to predict user preferences more accurately.

Google Search Algorithm:
Google's search engine relies heavily on machine learning to rank pages and provide the most relevant search results. By optimizing hyperparameters, Google has enhanced its algorithms' performance, ensuring faster and more accurate search results. This has been achieved through continuous experimentation and iteration, improving the user experience and maintaining Google's position as the leading search engine.

- **Case Study**
    - ➢ Healthcare Diagnosis:

A hospital implemented a machine learning model to assist doctors in diagnosing diseases from medical images. By optimizing the hyperparameters, the accuracy of the model improved significantly. This led to quicker and more accurate diagnoses, enhancing patient care and treatment outcomes. The process involved experimenting with different hyperparameter settings to find the optimal configuration that provided the best diagnostic accuracy.

- ➢ Financial Fraud Detection:

A bank used machine learning to detect fraudulent transactions. Initially, their model had a high rate of false positives and false negatives. By optimizing the hyperparameters, they reduced these errors and improved the model's reliability. This optimization process involved techniques like random search and grid search, which helped the bank protect its customers' accounts more effectively and reduce financial losses due to fraud.

Issue 2: Optimizing the Bias-Variance Tradeoff

Author: Carlos Garcia Guillamon

Reviewer: Yang Xue

Technical

- **Basics**

In machine learning, the bias-variance tradeoff is a characteristic of models which relates the accuracy of the models and their complexity. Both bias and variance are sources of error in a model:

- Bias is related to systematic errors in the model which lead to wrong predictions. It can arise either from insufficient training data, or from a lack of complexity in the model that can lead to wrong predictions.
- Variance is related to the variability of the dataset. This source of error arises due to data fluctuations, in particular when data contains too much noise and the model used tries to capture this noise. This occurs when the models used are too complex; for example, when polynomials of a very high order are used in regression, or when hyperparameters are tuned to produce an estimator which adapts too much to the training data.

The tradeoff is reflected in the fact that, in a model, when bias increases then variance decreases, and vice-versa. A model which is too simple will present a high bias but a low variance, leading to underfitting. In this case, this model will perform poorly on the training and testing sets, yielding large errors when applied to both. In the opposite case, a too complex model will have a high variance but a low bias, leading to overfitting. When overfitting occurs, the model will fit the training data almost perfectly, leading to very small errors. However, this model will perform very poorly on the test dataset, thus making it not suitable for practical uses. The objective is, therefore, to find the proper combination of hyperparameters that yields the lowest total error, without producing neither underfitting nor overfitting: this is the tradeoff.

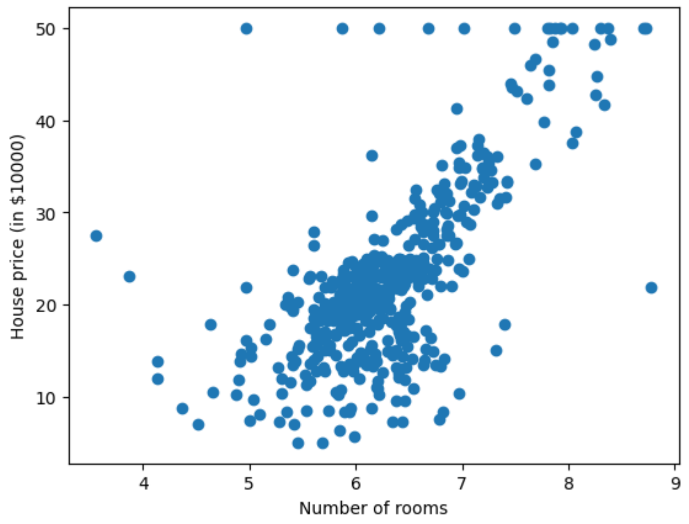
- **Illustration**

The bias-variance tradeoff can be visualized with a regression example. The dataset considered is the Boston housing dataset available at this link, originally from the Statlib library developed at Carnegie Mellon University. The dataset presents the values for properties in the suburbs of Boston. Several parameters are used to characterize each housing. More information on the dataset can be found here.

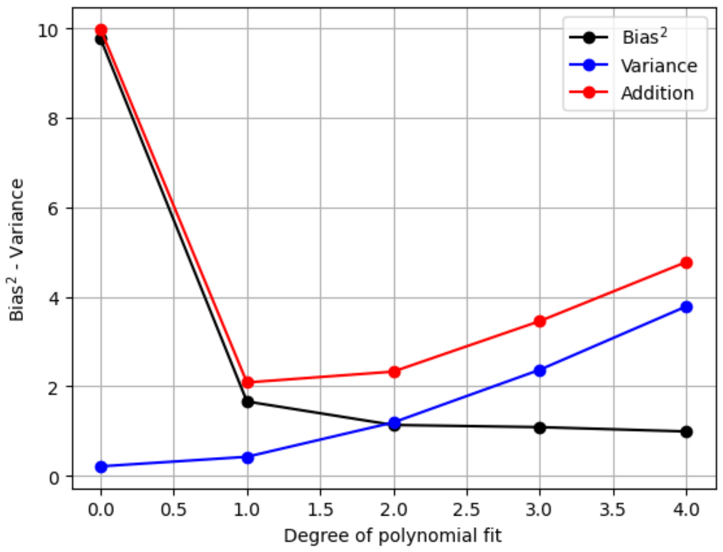
Polynomial regression will be applied to this dataset. In polynomial regression, a dependent variable *y* can be estimated as a non-linear function from the independent variable *x* as follows:

$$y = y_0 + x + x^2 + \ldots + x^n$$

Where y_0 is the intercept and n is the degree of the polynomial considered. This example is suitable to explain the bias-variance tradeoff with a simple model that can be easily coded in python. For application to this dataset, the independent variable considered will be the average number of rooms, and the dependent variable will be the house value. The relation between these two variables in the dataset can be illustrated in the following scatterplot:



As observed in the previous figure, the data is relatively linear, so a non-linear estimator like polynomial regression might not perform better than a linear one. Applying the code available in the notebook, the following relation between bias and variance as a function of the degree of polynomial fit can be obtained:



As shown in the graph, a low degree of polynomial results in a very high bias and low variance, indicating underfitting. By increasing the degree, the bias decreases and the variance increases. The addition of both produces then a the classical U-shaped curve reflecting the bias-variance tradeoff. The minimum of this curve is located at a degree of 1, which corresponds to a linear fit: therefore, in this case, the data can be properly represented with a linear polynomial. Extrapolating to polynomial degree to higher values increases the error monotonically, reflecting overfitting.

- **Mathematical interpretation**

Mathematically speaking, it is considered that a dependent variable *y* is related to a set of independent variables *X* through a relation of the following type:

$$y = f(X) + \varepsilon$$

Where *f(X)* is a function of the independent parameters and ε is a term accounting for noise. The idea behind a model is to produce an estimator for function *f(X)*, which will be denoted as *f*(X)*. Such estimator can be whichever function, from the regressor previously introduced to more complex models such as neural networks. Therefore, the expected squared error for a given set of parameters *X* will be:

$$Error = E[(y - f*(X))^2]$$

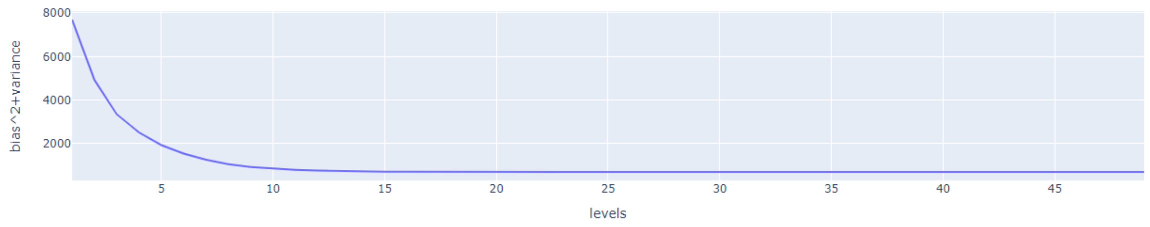
By introducing the real relation between *y* and *f(X)*, this error can be further decomposed as:

$$Error = (E[f*(X)] - f(X))^2 + E[(f*(X) - E[f*(X)])^2] + \sigma^2$$

In this expression, the first term corresponds to squared bias, the second one is the variance, and the last one is what is known as irreductible error, which comes from noise in the dataset. This is the reason why, in order to compare bias and variance, the bias must be squared: to be consistent with the units.

- **What's best? Good practices**

To find the best trade-off between bias and variance, and hence avoid both underfitting and overfitting, it is convenient to evaluate the model performance by using graphical tools reflecting both variables. The plot displayed above, which shows the U-shaped curve, is the most useful and common tool to visualize the most optimal hyperparameter for a model (in this case, the degree of polynomial fit). Another example for this type of plot is the one corresponding to Figure 7 of the Lesson Notes from Module 7, Lesson 1 from the actual course. This one displays the following shape:



Where the addition of bias^2 and variance are represented as a function of the corresponding hyperparameter, which in this case is the depth (level) of a decision tree regressor. In this case, increasing levels does not increase the total addition of both, showing that this model could work well even with a high complexity. Nevertheless, bias and variance should be also evaluated separately, to understand better their dependence on the tree depth.

Non-technical

- **Discussion**

In this issue, the question to answer is *'How well can the models be expected to work for predicting future cases?'*. In the previous graphs, it has been shown how the bias and variance can be evaluated with simple python code (which can be found in the associated notebook). By the proper selection of graphical tools and plots, a few charts can yield a deep insight into the evolution of these magnitudes with the model hyperparameters. Therefore, models that are constructed after these analyses will be less prone to underfitting and overfitting, and are expected to perform better when applied to future cases which are not included in the training datasets.

- **Conclusions**

Bias and variance are intrinsic characteristics of a model that cannot be avoided, and which can cause the models to produce either underfitting or overfitting, therefore making them unreliable for predicting real cases. Nevertheless, a proper characterization of their behavior, specially with graphical tools which are simple to code and easy to interpret, can help in model optimization. These type of analyses should always be considered when working with machine learning models, in order to properly select the correct level of hyperparameters and hence improve the performance of estimators.

Issue 3: Applying Ensemble Learning- Bagging, Boosting or Stacking

Author: Yang Xue

Reviewer: Ebenezer Yeboah

Technical

Process:

- Bagging

  The first step is creating multiple subsets of the training data using bootstrapping. Then, train a separate model on each subset. Finally, combine the predictions of each model by averaging (for regression) or majority voting (for classification).

- Boosting

  Train an initial model on the training data as the first step. The next step is evaluating the model and assigning higher weights to the misclassified instances. Train a new model on the weighted data, focusing more on the hard-to-classify instances.

- Stacking

  The first step is training multiple base models based on the training data. Hence, generate predictions from each base model. Use these predictions as input features to train a meta-learner model. The final step is using the meta-learner model to make the final predictions.

Illustration

In this case, a complicated case is used as an example of ensemble learning. The Bitcoin price is used as the dependent variable, and independent variables are the S&P 500, Gold index, Silver index, and EURUSD.

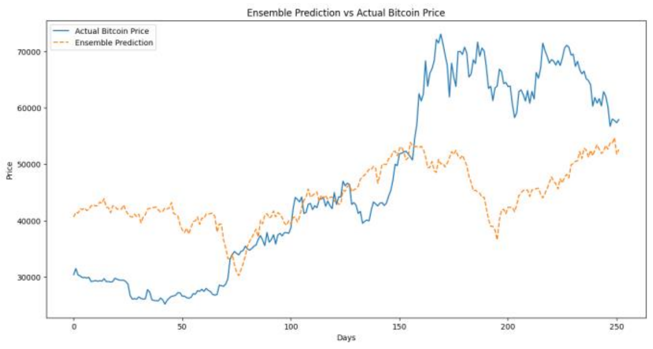


Figure: the Actual Bitcoin Price vs Prediction with Ensemble Prediction

Due to the complexity of the Bitcoin price, it can be very hard to be predicted by basic machine learning techniques. However, with the combination of a simple machine learning algorithm and a neural network algorithm, it can predict some trends in the middle. The overall MAE (Mean Absoluate Error) is 195892 which indicates the model has a lot to improve. It averages the MAE of two basic algorithms, and make the performance generally stable.

Metrics and Evaluation

The performance evaluation of ensemble learning is similar to the evaluation of machine learning. These metrics include accuracy, precision, recall, and F1-Score. Advanced, ROC, AUC, and confusion matrix can also be used for evaluation.

- Accuracy: The proportion of correctly classified instances among the total instances.
$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ predictions}.$$
- Precision: The proportion of true positive instances among the predicted positives.
$$Precision = \frac{True\ Positives}{True\ Positives + false\ positives}.$$
- Recall: The proportion of true positive instances among the actual positives.
$$Recall = \frac{True\ positives}{True\ positives + false\ Negatives}.$$
- F1 -score: This can be described as the mean of the precision and recall.
$$F1score = 2 \times \frac{Precision \times Recall}{Precision + Recall}.$$
- ROC and AUC: The Receiver Operating Characteristic (ROC) curve plots the true positive rate against the false positive rate. The Area Under the Curve (AUC) quantifies the overall ability of the model to discriminate between positive and negative classes.
- Confusion Matrix: A table that shows the number of true positives, true negatives, false positives, and false negatives, and this matrix provides a detailed performance.

Journal and Reference

- Polikar, Robi. "Ensemble learning." Ensemble machine learning: Methods and applications (2012): 1-34.
- Webb, Geoffrey I., and Zijian Zheng. "Multistrategy ensemble learning: Reducing error by combining ensemble learning techniques." IEEE Transactions on Knowledge and Data Engineering 16.8 (2004): 980-991.

Non-technical

Description

In the previous portfolios, we have discussed a range of machine-learning models and algorithms, but different algorithms have their advantages and disadvantages which limit their performance in different scenarios. With the model combination or ensemble learning, disadvantages can be made up.

Basics

Ensemble learning is a machine learning paradigm where multiple models are trained to solve the same problem and then combined to get better performance. In this case, bagging, boosting, and stacking are combined to solve problems. With the combination of multiple machine learning models, the accuracy, robustness, and generalizability of results are supposed to be improved.

- Bagging

  Bagging is an ensemble technique that aims to reduce the variance of a model. The basic process is training multiple versions of a machine-learning model, and then averaging them. The Random Forest algorithm is the most common example of a bagging machine learning algorithm.

- Boosting

  Boosting is a machine-learning technique that aims to reduce both bias and variance by sequentially training models. Each new model attempts to correct the errors made by the previous models. Boosting focuses on training models that perform well on instances that the previous models struggled with, often by assigning higher weights to these instances. Typical example algorithms are AdaBoost and XGBoost.

- Stacking

  Generally, stacking is a technique that aims to improve the prediction performance of machine-learning algorithms by adopting the combination of multiple different types of models. Decision trees, support vector machines, and neural networks are typical example algorithms of stacking techniques in machine learning.

Advantages

By adopting the combination of various machine learning techniques (bagging, boosting, and stacking), the prediction can be improved significantly.

- **Improved Accuracy**: the main reason to combine some machine learning models is to improve accuracy. Specifically, stacking is used for accuracy improvement.

- **Reduced Overfitting**: As a common issue in most machine learning algorithms, overfitting occurs a lot. Techniques like bagging help to reduce overfitting.
- **Increased Robustness**: Ensemble methods are often more robust to the peculiarities of the training data, as they can mitigate the impact of outliers and noisy data points.
- **Reduction of Bias and Variance**: Boosting can reduce both bias and variance, while bagging primarily reduces variance, leading to models that are both accurate and stable.

Disadvantages

Machine learning algorithms have some disadvantages which may not be able to be reduced. For example, the model complexity and longer training time, as they are robust. In addition, compared with simple regression and mathematical models, machine learning models can be very hard to implement and interpret.

- **Increased Complexity**: machine learning models and neural networks are well-known for their complexity. With applying ensemble, machine learning becomes more complex.
- **Long-time training and significant computation resource consumption**: as machine learning algorithms, they require big data and lots of computation power. Applying ensemble learning makes the situation worse.
- **Difficult to implement and interpret**: with a great amount of parameters, the machine learning models and results are hard to interpret as well as implement.

Discussion

Ensemble learning is a way to combine various machine learning models to perform better and minimize their disadvantages. Ensemble learning brings many improvements including accuracy improvement, overfitting reduction, robustness increase, bias reduction, and variance reduction. With these improvements, ensemble learning should be able to produce stable and good predictions. However, the improvement requires a great amount of training data and computation resources to support it. The training time is also much longer to obtain better predictions.

Step 5: Marketing material - Bagging, Stacking and Boosting

In ensemble learning techniques, there are three methods that are particularly popular: bagging, stacking and boosting. The basics of these methods and their main advantages and drawbacks have been discussed previously in this report. Furthermore, they have been applied to a real case for predicting future bitcoin performance.

Even though the application to the Bitcoin case did not produce satisfactory results, this was attributed to the complexity of the Bitcoin (since it is a highly volatile asset) and to the simplifications made in the models. As discussed in the notes, and as seen in the course content, ensemble learning is a powerful tool for making the most of machine learning models. Bagging is a powerful tool to be applied in a dataset when the data is limited, in order to reduce the variance of the model (which, as we say previously when discussing the bias-variance tradeoff, is an important source of error). Similarly, boosting can be applied with specific models, such as XGBoost or Adaboost, in order to reduce simultaneously both bias and variance. If more complex models are desired, then stacking can be applied in order to combine different models and hence improve the overall performance. Therefore, it is observed how these three strategies can be applied for different purposes, but with the same goal: improve the performance of machine learning models.