GROUP WORK PROJECT # 03 MScFE 632: Machine Learning in Finance

GROUP NUMBER: ANONYMIZED FOR PRIVACY

FULL LEGAL NAME	LOCATION (COUNTRY)	EMAIL ADDRESS	MARK X FOR ANY NON-CONTRIBUTING MEMBER
ANONYMIZED FOR PRIVACY ETIENNE LANDRY	ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY ANONYMIZED FOR PRIVACY	
WINSTONE ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY ANONYMIZED FOR PRIVACY	
Kristof ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY ANONYMIZED FOR PRIVACY	

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	ANONYMIZED FOR PRIVACY **ETIENNE LANDRY**
Team member 2	**Winstone** ANONYMIZED FOR PRIVACY
Team member 3	**Kristof** ANONYMIZED FOR PRIVACY

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 1-3

GROUP MEMBER 2: ISSUE 1 : OPTIMIZING HYPERPARAMETERS

TECHNICAL SECTION

1. **Introduction**

Hyperparameters are external parameters whose value is set before the machine learning model. These parameters usually influence how the model learns and generalizes the data and are not learned from the data like other model parameters.

Common Hyperparameters in Financial Models

- **Learning_rate:** The hyperparameter determines how the model adjusts its parameters during the training
- **Max_depth**: Used in setting the depth of the tree during training.
- **Max_leaf_nodes**: Limits the number of terminal nodes(leaves) in tree-based models.
- **N_estimator**s: Used to control the number of trees in a model
- **Criterion**: Determines the quality of a split in a decision tree
- **C**: Controls the penalty for misclassification.
- **Gamma:** Sets the level of influence SVM have on the decision boundary
- **Max_features**: Limits the number of features to be evaluated at each split

2. **Optimization Methods**

A. **Manual Search**

The technique involves manually selecting and testing combinations of hyperparameters based on one's prior knowledge, domain knowledge and observed model behaviour. Mainly used in HistGradient Boosting Classifier.

How optimization can be achieved:

- Firstly, select the hyperparameters e.g *max_depth: [3,5,10]*,
- This is followed by training and evaluating each combination manually based on their accuracy and F1 scores
- The model with the highest accuracy and F1 score is then chosen, and the model retrained on the full training set using the selected hyperparameter and later deployed.

B. **Grid Search (GridSearchCV)**

This is a technique that systematically tests combinations of model parameters to find the best ones on model performance.

In python it is implemented in *scikit-learn* using *GridSearchCV* that also performs cross-validation to evaluate each combination.

To apply, you need to define a model to use e.g RandomForestClassifier and then specify a grid of hyperparameters to test e.g.,

```
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [3, 5, None],
    'criterion': ['gini', 'entropy']
}
```

GridSearchCV is then run and each combination is evaluated using cross-validation. The best performing configuration is selected using *grid_search.best_params_*

```
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

C. Random Search

In this method, a combination of hyperparameters is randomly sampled from a defined space.

- Using the technique, the model randomly selects combinations of hyperparameters from a defined space.
- Each sampled configuration is trained and evaluated using cross-validation and the best-performing combination is chosen based on metrics like accuracy or F1 score. E.g.,

```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier

param_dist = {
    'n_estimators': [50, 100, 150],
    'max_depth': [3, 5, 10, None],
    'criterion': ['gini', 'entropy']
}

search = RandomizedSearchCV(RandomForestClassifier(), param_distributions=param_dist, n_iter=10, cv=5)
search.fit(X_train, y_train)
print(search.best_params_)
```

In the example above, the model randomly samples 10 combinations from the set parameters and distributions, then selects the best one based on cross-validation performance.

D. Bayesian Optimization

The technique builds a probabilistic model of the objective function and uses it to select the best hyperparameter combinations using intelligence. In Python it is implemented using *BayesSearchCV* from the *scikit-optimize* library.

- The technique works by defining the search space using ranges or distributions for each hyperparameter.
- BayesSearchCV n then builds a probabilistic model of the objective function and uses it to select the most promising hyperparameter combinations. Combinations likely to improve the model are selected based on past evaluations.
- The process continues iteratively, refining the search intelligently E.g.,

```python
from skopt import BayesSearchCV
from skopt.space import Real, Integer
from sklearn.ensemble import RandomForestClassifier

search_space = {
    'n_estimators': Integer(10, 200),
    'max_depth': Integer(1, 10),
    'max_features': Real(0.1, 1.0)
}

bayes_search = BayesSearchCV(RandomForestClassifier(), search_space, n_iter=20, cv=5, random_state=42)
bayes_search.fit(X_train, y_train)
print(bayes_search.best_params_)
```

The model above runs 20 iterations of cross-validation, and prints the best combination found.

3. Evaluation Metrics used during Tuning

A. Accuracy

Measures the ratio of correct predictions from the total predictions made. The model with the highest accuracy score is selected based on this metric

For a binary classification problem, accuracy can be defined using the components of confusion matrix

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Where:

- **TP**: True Positives
- **TN:** True Negatives
- **FP**: False Positives
- **FN**: False Negatives

B. Precision

The metric measures how many of the predicted positives are actually correct. It is prioritized when false positives are costly.

C. Recall

Measures how many actual positives were correctly identified. Prioritized when false negatives are more costly.

D. F1 Score

This is a harmonic mean of precision and recall. Mostly used when looking for a balance between precision and recall. The preferred model is the one with the highest F1 score.

E. Cross-Validation Score

This measures how well the model will adapt to the unseen data given the configuration set. It helps in detecting overfitting or underfitting in a model.

NON-TECHNICAL SECTION

A. introduction

Hyperparameters are configurations applied to a machine learning model to control how the model learns from the data. The configurations are usually set before the training begins and they significantly influence the model's ability to make accurate and reliable predictions.

In a model to predict loan defaults, configurations can be made on how deep a decision tree can grow or how many trees are to be used in the forest model. These settings will in turn influence the model's effectiveness. To avoid simplistic models (ignores key patterns) and complex models (memorizes data and performs poorly on new data) it is always prudent to select the right hyperparameters

B. Optimization Techniques

Optimization techniques such as manual search, grid search, random search, and Bayesian optimization are used in determining the best model. The choice of a particular technique depends on the model complexity, computational resources available among others

Technique	**How the Technique works**	**When applicable**
Manual Search	Heavily rely on user experience and intuition to choose the configurations	Small spaces
Grid Search	Tests every combination systematically (exhaustive search)	Small and well-defined spaces. The model guarantees thoroughness.
Random Search	Selects random combinations from a distribution.	Large spaces. Is faster than Grid Search.
Bayesian Optimization	Learn from past results using intelligent to choose the next best configuration	In Expensive models where computational efficiency is critical.

The optimization techniques helps us find the best configurations that give the best results of a model to avoid poor and inconsistent models

C. Evaluation Metric to determine the best model

To find the optimal model configuration, we have to systematically test different hyperparameter combinations using the mentioned optimization techniques i.e. grid search, random search, and Bayesian Optimization. The performance metrics mentioned i.e., accuracy, precision, recall, and F1 score are then used to determine the best configured model.

The choice of a metric to use will depend on the objective of the model. For example, if focus is to catch all possible events then recall metric will be prioritized, precision to reduce false alarms. For a balanced dataset then accuracy or F1 score would be prioritized. Cross-validation is usually used to measure how well the model will adapt to unseen or future data.

The best model following the hyperparameter tuning is one that performs consistently across multiple data splits during training. Such a model gives confidence that the chosen parameters are optimal for the model's intended application.

D. Determining the model that is best for future cases

To determine whether the model will work well on future cases, we use cross-validation's performance on unknown data. The model that performs well across different data splits is more likely to generalize well to new data and different scenarios like customer behavior or shifts in markets. Models that perform well both within-sample and out of sample have the ability to perform on new and unknown data hence able to support future decision making.

E. Putting the models together

Sometimes, relying on just a single model isn't enough to capture the full complexity and volatility of financial data. That's why we have to apply ensemble methods like bagging, boosting, and stacking which allow us to combine the predictions of multiple individual models, leading to a much more powerful and stable overall performance.

Applying the ensemble methods makes hyperparameter tuning even more essential because each method has its own settings controlling how the models are combined.

- In bagging, by tuning *n_estimators* and the sample size will help reduce model variance
- In boosting, adjusting the learning rate and the depth of each weak learner will boost accuracy

Optimizing the parameters ensures that the ensemble captures diverse patterns and delivers reliable predictions.

F. Conclusion

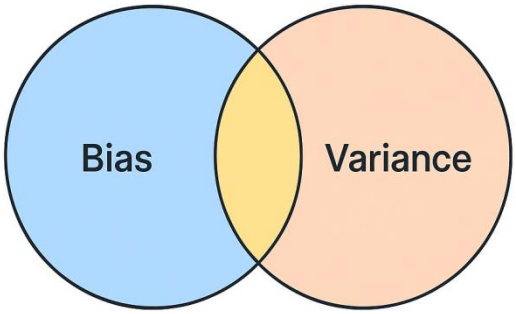
Optimizing hyperparameters is key to building accurate and reliable models as it ensures that the correct configurations are set which guarantees model performance on future and unseen data.

GROUP MEMBER 1 (SECTIONS 1-3)
ISSUE 2: OPTIMIZING THE BIAS-VARIANCE TRADEOFF

I. TECHNICAL SECTION

A- INTRODUCTION TO BIAS-VARIANCE TRADEOFF

When learning algorithms are based upon incorrect assumptions, this results in bias. A model that oversimplifies the data results to higher bias, failing to identify significant patterns. In contrast, variance is positively correlated to sensitivity to small changes in the testing data. So, a well crafted model will try to make a good accurate assumption, while staying less sensitive to small changes: this is the bias-variance tradeoff.



Bias Variance Tradeoff

In financial applications like predicting loan defaults, this tradeoff is critical.

B- FORMULATION

The overall error can be decomposed into three components:

$$\text{Total Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

Each of the above term has a specific purpose in terms of model performance:

Bias²: it's this kind of metric that quantifies the distance between the model's predictions and the true values given its assumptions. A high bias indicates that the model is too rigid and does not capture patterns - underfitting.

Variance: this statistic shows how much the model's predictions change when trained on different datasets. A high variance means that the model is too sensitive to small changes inside the data, leading to overfitting.

Irreducible Error: This represents things like noise or randomness inside the data that no model can retrieve. It sets a lower bound on the achievable error.

For instance, let's assume a supervised learning setup where we want to predict a target variable Y from input features x, using a model $\hat{f}(X)$. The expected prediction error at a point x can be decomposed as:

$$E\left[(Y - \hat{f}(x))^2\right] = \left(\text{Bias}[\hat{f}(x)]\right)^2 + \text{Variance}[\hat{f}(x)] + \sigma^2$$

Where:

- $E\left[(Y - \hat{f}(x))^2\right]$ is the **expected squared error** at point x.
- $\left(\text{Bias}[\hat{f}(x)]\right)^2$ is the **squared bias**, defined as:

$$\text{Bias}[\hat{f}(x)] = E[\hat{f}(x)] - f(x)$$

It measures how far the average prediction is from the true function $f(x)$.

- $\text{Variance}[\hat{f}(x)]$ is the **variance**, defined as:

$$\text{Variance}[\hat{f}(x)] = E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right]$$

It captures how much the predictions vary across different training sets.

- σ^2 is the **irreducible error**, representing noise in the data:

$$\sigma^2 = E\left[(Y - f(x))^2\right]$$

This is the part of the error we cannot reduce, even with a perfect model.

C- MODEL BEHAVIOUR EXAMPLES

ML models would show different levels of bias and variance characteristics, depending on how the assumptions are made or how each model is less or more sensitive to changes in the data:

1- Linear Regression model:

In the Linear regression models, the relationship between input features X = (x_1, x_2, ..., x_n) and the target variable Y is modeled using a linear function:

$$\hat{Y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

- $\hat{Y}$: predicted output (e.g., default risk score)
- β_0: intercept term
- β_i: coefficients for each feature x_i
- The model is trained by minimizing the **Mean Squared Error (MSE)**:

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^{m} (Y_i - \hat{Y}_i)^2$$

As Linear regression assumes a straight line relationship between inputs (x) and outputs (Y), this simplicity may likely lead to high bias - the model may miss complex patterns because the real word relationship may be different from a straight relation it assumes. However, this structure makes it stable across datasets, leading to low variance.

In finance, a linear approach might consistently predict average risk levels but would likely fail to capture subtle signals of default risk, especially when dealing with non-linear or noisy data.

2- Random Forest model

Here, each tree T_j makes a prediction $\hat{Y}_j$, and the final output is the average (for regression) or majority vote (for classification):

- **Regression**:

$$\hat{Y} = \frac{1}{K} \sum_{j=1}^{K} \hat{Y}_j$$

- **Classification**:

$$\hat{Y} = \text{mode}(\hat{Y}_1, \hat{Y}_2, ..., \hat{Y}_K)$$

Where: K: number of trees; each tree is trained on a bootstrap sample of the data and random subsets of features are used to split nodes, increasing diversity.

D- OPTIMIZING THE TRADEOFF

1). Regularization

Lasso and Ridge Regressions are often used for model regularization:

- **Lasso Regression (L1 penalty)**:

$$\text{Loss} = \text{MSE} + \lambda \sum_{i=1}^{n} |\beta_i|$$

Encourages sparsity by shrinking some coefficients to zero.

- **Ridge Regression (L2 penalty)**:

$$\text{Loss} = \text{MSE} + \lambda \sum_{i=1}^{n} \beta_i^2$$

Shrinks coefficients smoothly, reducing model sensitivity.

2). Model complexity control

Controlling model complexity helps manage bias and variance directly (Max Depth and Number of Estimators).

3). Cross-validation

Data is split into K parts.

The model is trained on K-1 parts and tested on the remaining part (this process repeats KK times, and the average score is used).

4). Ensemble Methods

Combining multiple models improves performance and stability (bagging, boosting)

E- PERFORMANCE EVALUATION

To assess how well a model balances bias and variance, we use specific evaluation metrics.

- **Precision**:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Measures how many predicted positives are actually correct.

- **Recall**:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Measures how many actual positives were correctly identified.

- **F1 Score**:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

A high F1 score may be indicating a good balance between precision and recall. It can be useful when both false positives and false negatives matter.

- **Cross Validation Score**
- **Learning Curves**

II. NON-TECHNICAL SECTION

1. Bias vs variance

The goal of training a ML model is to learn insightful patterns from the data. As said above, bias increases with the distance between the models assumptions and the real world values; meanwhile the variance is correlated to the model sensitivity to small changes inside the data.

2. In the financial context.

In financial tasks like predicting loan defaults, the balance between bias and variance levels balance is critical. A high bias model might overlook risky borrowers because it doesn't capture enough detail, while a high variance model might give unstable predictions, reacting too much to noise or rare cases. Both scenarios can lead to poor decisions - either approving risky loans or rejecting safe ones.

3. Real-World Impact

As Underfitting (too much bias) leads to missed risks and weak predictions, overfitting on the other hand (too much variance) causes false alarms and unreliable forecasts. A well-balanced model once put into production, often helps banks and other financial institutions make data-driven decisions.

4. How We Optimize It

To find the right balance between both, we test different model settings, combine models through methods such as bagging and boosting to get better both the stability and the accuracy of the models. Then, we can assess the performance of the model with a new dataset in order to confirm that the model is ready for production.

5. Final Takeaway

The goal is to construct a model that is neither overly basic (risk of higher bias), nor excessively complicated one. It should work well on both past and future data - giving consistent, trustworthy predictions. This often helps financial institutions make informed decisions and adapt confidently to changing conditions.

ISSUE 3: Applying Ensemble Learning—Bagging, Boosting, or Stacking

1. **Technical part**

1. **Theoretical foundation**

For base Learner $f_m(x)$ we can write:

$$\hat{f}(x) = \sum_{m=1}^{M} w_m f_m(x)$$

Where w_m weight depends on the type of ensemble:

- Bagging: equal weights (1/M)
- Boosting: adaptive weights - Models that correct previous errors get more weight
- Stacking: $f_m(x)$ can come from different type of algorithms and the meta-model learns the optimal combining weight

2. **Features of the methods**

- **Bagging**
    - Average prediction of base models
    - E.g random Forest
    - Decreases variance, provides stable predictions
    - Can be used for risk forecasting and volatility modelling

- **Boosting**
    - Sequentially fitted weak learns with adapted weights
    - E.g XGBoost
    - Decreases Bias, prone to overfit
    - Can be used for Credit scoring
- **Stacking**
    - Heterogeneous models combined by meta-learner
    - E.g Neural Network
    - Decreases bias and variance
    - Useful in multi factor predictions

3. **Mathematical Intuition for the methods**

- **Bagging**

$$\mathrm{Var}\left[\bar{f}(x)\right] = \rho\sigma^2 + (1-\rho)\frac{\sigma^2}{M}$$

    - σ^2 is variance
    - ρ is pairwise correlation
    - Variance reduction is proportional to 1/M when models are weakly correlated
- **Boosting**

$$F_m(x) = F_{m-1}(x) - \eta \sum_i \nabla_{F_{m-1}} \mathcal{L}(y_i, F_{m-1}(x_i))$$

    - $\mathcal{L}(y_i, F_{m-1}(x_i))$ is the loss function (squared error, typically)
    - η is learning rate

- $\nabla_{F_{m-1}}\mathcal{L}(\cdot)$ is the gradient of loss with respect to the model prediction
- Minimizes the objective with gradient descent

- **Stacking**

$$\hat{y}_i = g(z_i;\beta) = \beta_0 + \sum_{m=1}^{M} \beta_m f_m(x_i),$$

- where $z_i = (f_1(x_i), \ldots, f_M(x_i))$
- And β learned on validation folds

4. **Diagnostic and interpretability**

- Learning curves help to track overfitting (training vs. validation error)
- Diversity heatmap shows correlation map of base model prediction that helps removing models that actually estimates the same underlying relationship in data (correlation over 0.8)
- Sensitivity analysis is a set of tests conducted to show the change in model outputs to some reasonable change in hyperparameters / parameters

2. **Non-Technical part**

1. **Ensemble methods in General**

Instead of relying on a single forecasting model, we combine several models to create a combined decision maker model. Each member sees the data from a different angle; the ensemble merges their views to make a more balanced decision.

2. **Use-case of different ensemble methods**

Data can be noisy. A model that works today might fail next quarter. Ensembles mitigate that risk by diversifying model opinions

Main approaches:

Bagging

We average many versions of the same model trained on different samples to achieve smoother and more stable forecasts.

Boosting

We teach models sequentially, each one learning from the mistakes of the previous to be more accurate on difficult patterns.

Stacking

We combine different types of models (linear, tree, neural) and let a “meta-model” decide how to weight them, which often results in the best overall balance.

3. **Interpretation from performance perspective**

    - More consistent returns across market regimes.
    - Reduced drawdowns from model failure.
    - Slightly higher computation and tuning cost, but better risk-adjusted stability.3

4. **Indicators of successful models**

    We track out-of-sample accuracy and economic metrics.The ensemble should outperform individual models in stability even if its headline return is only slightly higher.

5. **Final takeaways**

    Use ensemble learning as a risk-control tool for model uncertainty. We prefer the simplest ensemble that maintains strong and stable out-of-sample results. Ongoing monitoring and periodic re-training ensure adaptation to new market conditions.

STEP 4

	Wrote	**Reviewed**
Student A	Issue 2	Issue 1
Student B	Issue 1	Issue 3
Student C	Issue 3	Issue 2

STEP 5

MARKETING ALPHA

A. Introduction

Having clear insights is very critical in today's data-driven financial world. At Team Alpha, we consistently refine and strengthen our machine learning (ML) model capabilities to meet this dynamic world. In addition to building models, we have also grown our capabilities to include advanced hyperparameter tuning and ensemble learning techniques. This ensures that we not only deliver models that are just accurate, but also adaptive, transparent, and dependable.

B. We get it right at Parameter tuning

Great models have right set-ups and that's why we have built capabilities in hyperparameter tuning. Using optimization methods like grid search, random search, and Bayesian optimization, we test different model settings to get better both the stability and the accuracy of the models. Through this, we avoid basic and overly complex models making our predictions consistent and reliable.

C. We adapt the Models to Changing Financial landscape

For the complex and changing financial world, we build adaptive models using the ensemble methods. These methods make our models resilient to market distortions and the changing trends.

- **Bagging**: We average many versions of the same model trained on different samples to achieve smoother and more stable forecasts.
- **Boosting:** We teach models sequentially, each one learning from the mistakes of the previous to be more accurate on difficult patterns.

D. **Stacking:** We combine different types of models (linear, tree, neural) and let a "meta-model" decide how to weight them, which often results in the best overall balance.

E. We build for Trust and for Future

At Team Alpha, we understand the importance of trust, and as a result we build transparent and explainable models. We make sure the logic behind each model is clear regardless of the model being a decision tree or a complex stacked ensemble.

F. Conclusion

We don't just crunch numbers, we deliver insights that drive action. By believing the future belongs to models that are intelligent and smart, flexible and understandable, accurate and powerful, we create

tools that are both technically sound and practically valuable to predict, explain, learn, and guide strategic decisions.

REFERENCES

1. Pedregosa, Fabian, et al. "GridSearchCV." *Scikit-learn 1.7.1 Documentation*, scikit-learn.org, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. Accessed 21 Oct. 2025.
2. "Hyperparameter Tuning in Scikitlearn: GridSearchCV vs RandomizedSearchCV." *PythonTutorials.net*, https://www.pythontutorials.net/scikit-learn/hyperparameter-tuning-in-scikitlearn-gridsearchcv-vs-randomizedsearchcv/. Accessed 17 Oct. 2025.
3. Rede, Archit, and Theodore LaGrow. "Tutorial on Hyperparameter Tuning Using Scikit-learn." *OMSCS 7641: Machine Learning*, 16 Feb. 2024, https://sites.gatech.edu/omscs7641/2024/02/16/tutorial-on-hyperparameter-tuning-using-scikit-learn/. Accessed 21 Oct. 2025.
4. WorldQuant University. *MScFE 632: Machine Learning in Finance – Module 7 Lessons 1–4*. Course materials, WorldQuant University, 2025.