Week 6: Tree Ensembles Assignment

ARIMA/GARCH + Random Forest & XGBoost
for Portfolio Optimization

MSc Banking and Finance – FinTech Course

Due: we will discuss this in class..

1 Assignment Overview

Objective: Implement tree-based ensemble methods for portfolio optimization using Week 5 engineered features.

Deliverables:

- Jupyter Notebook (.ipynb) with code and analysis
- PDF Report (3-5 pages) summarizing findings
- Results file (.csv) with portfolio weights

2 Assignment Tasks

2.1 Task 1: Model Implementation & Comparison (30%)

Requirements:

1. Implement three models: Random Forest, Gradient Boosting, and XGBoost
2. Train each model on Week 5 features (use at least 15 features)
3. Perform 5-fold cross-validation for each model
4. Report MSE, R^2, and MAE for train and test sets
5. Conduct paired t-tests between models (report p-values)

Expected Output:

Model	Train R^2	Test R^2	MSE	p-value
Random Forest	0.XXX	0.XXX	0.XXXXXX	—
Gradient Boosting	0.XXX	0.XXX	0.XXXXXX	0.XXX
XGBoost	0.XXX	0.XXX	0.XXXXXX	0.XXX

2.2 Task 2: Hyperparameter Tuning (25%)

Requirements:

1. Select your best-performing model from Task 1
2. Define hyperparameter grid with at least 5 parameters
3. Use GridSearchCV with 5-fold cross-validation
4. Create validation curves for top 3 hyperparameters
5. Report optimal hyperparameters with justification

Minimum Grid Size:

- Random Forest: Test at least 36 combinations
- Gradient Boosting: Test at least 36 combinations
- XGBoost: Test at least 48 combinations

Justification: Explain why optimal values make sense (150-200 words).

2.3 Task 3: Feature Importance & SHAP (25%)

Requirements:

1. Extract built-in feature importance from your best model
2. Compute SHAP values for test set
3. Create SHAP summary plot
4. Compare built-in importance vs. SHAP importance (plot both)
5. Identify top 5 features and explain their impact (200-250 words)

Analysis Questions:

- Which features drive predictions most strongly?
- Do you observe any non-linear interactions?
- Are results financially interpretable?

2.4 Task 4: Portfolio Optimization & Backtesting (20%)

Requirements:

1. Use best model to generate portfolio weights for BTC, ETH, DOGE
2. Implement rolling-window backtesting (minimum 30-day window)
3. Compare three strategies:
   - Equal-weighted (1/3, 1/3, 1/3)
   - Week 5 Lasso-based portfolio
   - Week 6 Tree-based portfolio
4. Calculate: Sharpe ratio, max drawdown, win rate, total return
5. Visualize cumulative returns

Performance Table:

Strategy	Sharpe	Max DD	Win Rate	Total Ret.
Equal-Weighted	X.XX	-XX.X%	XX.X%	XX.X%
Lasso (Week 5)	X.XX	-XX.X%	XX.X%	XX.X%
Tree Ensemble	X.XX	-XX.X%	XX.X%	XX.X%

3 Bonus Tasks (Optional, +10%)

Choose **one** of the following:

3.1 Option A: Custom Ensemble Stacking

Combine Ridge (Week 5), Random Forest, and XGBoost predictions using optimal weights. Show improvement over individual models.

3.2 Option B: Regime Detection

Use decision tree splits to identify market regimes. Create separate portfolio strategies for each regime.

3.3 Option C: Early Stopping Analysis

Implement early stopping for Gradient Boosting/XGBoost. Plot training vs. validation error curves. Determine optimal number of iterations.

4 Evaluation Rubric

Criterion	Description	Points
Code Quality	Clean, documented, functional, follows best practices	20
Model Performance	All models trained, tuned, significant improvement shown	25
Analysis Depth	Insightful interpretations, connects theory to results	25
Statistical Rigor	Proper testing, cross-validation, significance tests	15
Visualizations	Professional, clear, informative plots	10
Report Quality	Well-written, organized, concise	5
Total		**100**
Bonus	(Optional)	**+10**

5 Quick Start Code Templates

5.1 Model Training

```
# Random Forest
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=100, max_depth=10,
                           random_state=42, n_jobs=-1)
rf.fit(X_train, y_train)

# XGBoost
import xgboost as xgb
xgb_model = xgb.XGBRegressor(n_estimators=100, learning_rate=0.1,
                             max_depth=6, reg_alpha=0.1, reg_lambda=1.0)
xgb_model.fit(X_train, y_train)
```

5.2 Hyperparameter Tuning

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [5, 10, 15],
    'learning_rate': [0.01, 0.05, 0.1]
}

grid_search = GridSearchCV(model, param_grid, cv=5,
                           scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)
print(f"Best params: {grid_search.best_params_}")
```

5.3 SHAP Analysis

```
import shap

explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_test)

```

```
# Summary plot
shap.summary_plot(shap_values, X_test, feature_names=feature_names)
```

6 Detailed Grading Breakdown

6.1 Excellent (90-100%)

- All models properly implemented with comprehensive analysis
- Rigorous hyperparameter tuning with clear justification
- Deep SHAP interpretation connecting to finance theory
- Professional visualizations with publication quality
- Statistical tests properly conducted and interpreted

6.2 Good (75-89%)

- Models trained with adequate tuning
- SHAP analysis present with reasonable interpretation
- Clear visualizations and proper documentation
- Some statistical testing conducted

6.3 Satisfactory (60-74%)

- Basic implementation functional
- Minimal tuning and analysis
- Adequate visualizations
- Limited statistical validation

7 Common Mistakes to Avoid

1. **Not using cross-validation** – Single train/test split is insufficient
2. **Scaling features for trees** – Trees don't need StandardScaler
3. **Default hyperparameters** – Always tune your models
4. **Ignoring overfitting** – Monitor train-test gap
5. **No SHAP analysis** – Feature importance alone is insufficient
6. **Missing statistical tests** – Must compare models rigorously
7. **Poor documentation** – Explain your code and decisions
8. **Late submission** – Start early, deadline is firm

8 Resources

8.1 Documentation

- scikit-learn: https://scikit-learn.org/stable/
- XGBoost: https://xgboost.readthedocs.io/
- SHAP: https://shap.readthedocs.io/