CSCA 5622 - Introduction to Machine Learning

Supervised Learning

Dyego Fernandes de Sousa

University of Colorado Boulder

November 1, 2025

Problem Statement & Dataset Overview

The goal of this project is to analyze the relationship between financial, environmental variables and ESG outcomes for a set of companies.

Specifically, the tasks are:

- **Regression**: To predict the ESG score of a company using its revenue and GHG emissions, aiming to quantify how these factors influence overall sustainability performance.
- **Classification**: To categorize companies into ESG risk levels based on their revenues and GHG emission figures, identifying key indicators associated with higher sustainability risks.

Both tasks will provide insights on how economic and environmental data are associated with ESG scoring and risk classification, supporting better data-driven decisions.

Dataset[1] Overview: CSV file containing 251 rows and 57 non-normalized features. Many rows containing empty values. Inexistense of target variables.

Exploratory Data Analysis (EDA)

Preprocessing: *Calculating the Target variables:*

- **esg_score**: A function of *Scope1plus2Total* and *revenuesghgco*, later normalized to be between 0 and 100.
- **esg_risk**: Category based on esg_score: LOW, MEDIUM-LOW, MEDIUM, MEDIUM-HIGH, and HIGH.

Raw Dataset

ESG Score Distribution

Statistic	Value
Count	251.00
Mean	90.64
Std Dev	15.76
Min	0.00
25%	90.14
Median	97.25
75%	98.10
Max	100.00

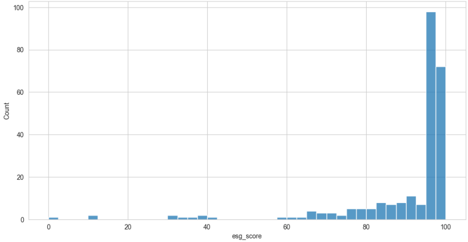


Figure 1: The long tail indicates that there are numbers far apart fom the head. This usually indicates that the target data needs transformation.

Risk Distribution

Risk Level	Count
HIGH	23
MEDIUM-HIGH	16
MEDIUM	35
MEDIUM-LOW	119
LOW	58
Total	**251**

Exploratory Data Analysis (EDA)

After Log Transformation

ESG Score Distribution

Statistic	Value
Count	251.00
Mean	41.56
Std Dev	21.73
Min	0.00
25%	29.08
Median	34.80
75%	56.77
Max	100.00

Risk Distribution

Risk Level	Count
HIGH	23
MEDIUM-HIGH	16
MEDIUM	35
MEDIUM-LOW	119
LOW	58
Total	**251**

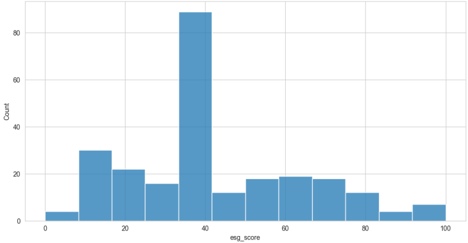


Figure 2: A small tail is still present, but way less concerning.

Feature Engineering

Dropping Columns:

- **1st round** - time-series features (e.g., 2020_food_sales, 2020_fs_adjust).
- **2nd round** - known meaninless features (e.g., id, brandlogos).
- **3rd round** - not important features, after feature importance analysis (e.g., revenuesarea, ghgreportarea).
- **4th round** - strong correlation, after feature importance analysis (e.g., esg_score_raw, esg_score_log).
- **5th round** - all NaN: realzero, yearrevenuesdata

Final set of Features for training:

- **scope2emitundifferentiated**, **brands**, and **scope1+2total** after application of median imputation to handle missing values

Models & Training Approaches

Decision Trees

- Baseline[4] [5].

Random Forest

- For bootstrap aggregation and to mitigate overfitting,.

AdaBoost (Adaptive Boosting)

- For sequential ensembling.

XGBoost[5] (Extreme Gradient Boosting) - *Not taught in the course, but worth trying*

- For advanced gradient boosting and efficient parallel processing.

Support Vector Machine[2](SVM)

- Non-ensemble alternative for comparison.

Models & Training Approaches

- **Train/Test** splitted as 80% and 20% and shuffled;

```python
# Dataset for Regression tasks
self.X_train_reg, self.X_test_reg, self.y_train_reg, self.
    y_test_reg = train_test_split(X, y_regression, test_size=
    test_size, random_state=self.random_state, shuffle=True)
# Dataset for Classification tasks
self.X_train_clf, self.X_test_clf, self.y_train_clf, self.
    y_test_clf = train_test_split(X, y_classification,
    test_size=test_size, random_state=self.random_state,
    stratify=y_classification, shuffle=True)
```

- **All models** were trained to perform **Regression** and **Classification**.

```python
# Method for training models
def train_models(self, models=['dt','ab','rf', 'xgb', 'svm'],
    task='both', hyperparameter_tuning=False, show_viz=False):
    if 'dt' in models:
        self.train_decision_tree(task, hyperparameter_tuning,
            show_viz)
    if 'ab' in models:
        self.train_adaboost(task, hyperparameter_tuning,
            show_viz)
    if 'rf' in models:
        self.train_random_forest(task, hyperparameter_tuning,
            show_viz)
    if 'svm' in models:
        self.train_svm(task, hyperparameter_tuning, show_viz)
    if 'xgb' in models:
        self.train_xgboost(task, hyperparameter_tuning,
            show_viz)
    return self
```

```python
# Regression, without hyperparameter tuning and with
    hyperparameter tuning
trainer.train_models(models=['dt', 'ab', 'rf', 'xgb', 'svm'],
    task='regression', hyperparameter_tuning=False, show_viz=
    True)
trainer.train_models(models=['dt', 'ab', 'rf', 'xgb', 'svm'],
    task='regression', hyperparameter_tuning=True, show_viz=
    True)
# Print model comparison - Regression
trainer.print_model_comparison()
# Classification, without hyperparameter tuning and with
    hyperparameter tuning
trainer.train_models(models=['dt', 'ab', 'rf', 'xgb', 'svm'],
    task='classification', hyperparameter_tuning=False,
    show_viz=True)
trainer.train_models(models=['dt', 'ab', 'rf', 'xgb', 'svm'],
    task='classification', hyperparameter_tuning=True,
    show_viz=True)
# Print model comparison - Classification
trainer.print_model_comparison()
```

Evaluation Metrics

For **Regression**[6]:

- R^2: Model fitness
- RMSE: Root Mean Squared Error
- MAE: Mean Absolute Error

For **Classification**[6]:

- Precision[3]: $\dfrac{TP}{TP + FP}$
- Recall[3]: $\dfrac{TP}{TP + FN}$
- Accuracy: $\dfrac{TP + TN}{TP + TN + FP + FN}$
- F1-Score: $\dfrac{Precision \times Recall}{Precision + Recall}$

Evaluation Metrics & Results

Untuned Models:

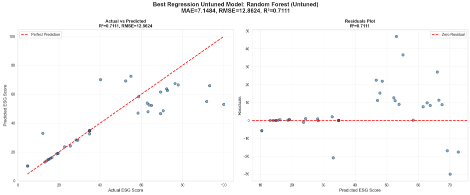
Regression Models

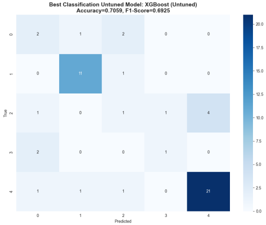
Model	R^2	RMSE	MAE
Random Forest	0.7111	12.8624	7.1484
XGBoost	0.6429	14.3012	7.4126
AdaBoost	0.6059	15.0229	10.5256
Decision Tree	0.5813	15.4847	8.3347
SVM	0.1297	22.3262	16.6253

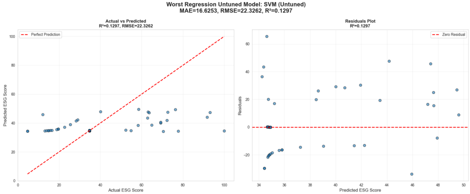
Classification Models

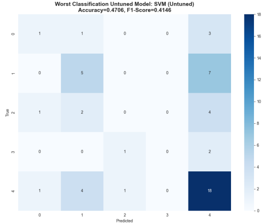
Model	Accuracy	F1-Score
XGBoost	0.7059	0.6925
Random Forest	0.6275	0.6128
Decision Tree	0.6078	0.6214
AdaBoost	0.5882	0.5607
SVM	0.4706	0.4146

Evaluation Metrics & Results









Hyperparameter Tuning - Ranges

Decision Tree

Param	Values
max_depth	[1, 2, 3, 5, 7, 8, 10, 15, None]
min_samples_split	[1, 2, 3, 5, 7, 9, 10, 12, 13, 15, 20, 25]
min_samples_leaf	[1, 2, 4, 8, 10]
max_features	['sqrt', 'log2', None]
class_weight	[None, 'balanced']

Combinations: Reg: **1,620** | Class: **3,240**

AdaBoost

Param	Values
n_estimators	[5, 10, 25, 50, 75, 100, 150, 200, 300]
learning_rate	[0.001, 0.01, 0.1, 0.3, 10]
loss	['linear', 'square', 'exponential']
random_state	[42]

Combinations: Reg: **135** | Class: **45**

Random Forest

Param	Values
n_estimators	[5, 10, 25, 50, 75, 100, 150, 200, 300]
max_depth	[1, 3, 5, 10, 15, 20, 30, None]
min_samples_split	[1, 3, 5, 7, 10, 20, 25]
min_samples_leaf	[1, 2, 3, 4]
bootstrap	[True, False]
max_features	['sqrt', 'log2', None]
class_weight	[None, 'balanced', 'balanced_subsample']

Combinations: Reg: **12,096** | Class: **36,288**

XGBoost

Param	Values
n_estimators	[1, 5, 10, 25, 50, 75, 100, 150, 200]
max_depth	[1, 3, 6, 9, 10]
learning_rate	[0.01, 0.1, 0.3]
subsample	[0.8, 1.0]
colsample_bytree	[0.8, 1.0]

Combinations: Both: **540**

SVM

Param	Values
C	[0.1, 1, 10]
gamma	['scale', 'auto', 0.001, 0.01]
kernel	['rbf', 'linear']
class_weight	[None, 'balanced']

Combinations: Reg: **24** | Class: **48**

Hyperparameter Tuning - Results

GridSearchCV was utilized to systematically evaluate and identify the optimal hyperparameter combinations. The best-performing configurations are summarized below:

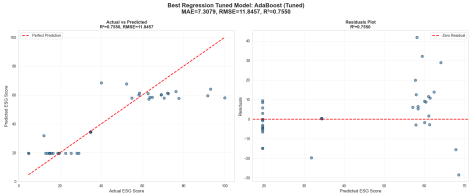
Regression Models

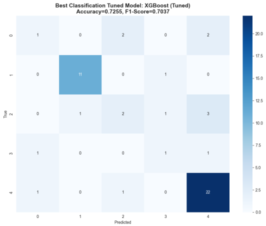
Model	R^2	RMSE	MAE	Best Hyperparameters
AdaBoost	0.7550	11.8457	7.3079	lr=0.001, loss='exponential', n_est=300
Decision Tree	0.7460	12.0611	6.7837	max_depth=5, min_samples_leaf=10, min_samples_split=2
Random Forest	0.7317	12.3955	7.0183	bootstrap=True, max_depth=10, min_samples_leaf=4, min_samples_split=10, n_est=75
XGBoost	0.7050	12.9992	8.0360	colsample_bytree=1.0, lr=0.3, max_depth=1, n_est=50, subsample=1.0
SVM	0.2399	20.8651	14.8728	C=10, gamma='scale', kernel='rbf'

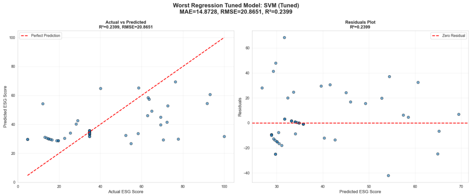
Classification Models

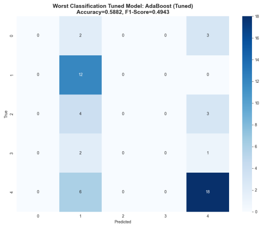
Model	Accuracy	F1-Score	Best Hyperparameters
XGBoost	0.7255	0.7037	colsample_bytree=0.8, lr=0.3, max_depth=1, n_estimators=150, subsample=0.8
Decision Tree	0.6667	0.6645	class_weight=None, max_depth=7, min_samples_leaf=2, min_samples_split=10
Random Forest	0.6667	0.6533	bootstrap=False, class_weight=None, max_depth=10, min_samples_leaf=2, min_samples_split=10, n_estimators=5
SVM	0.5882	0.5645	C=10, class_weight='balanced', gamma='auto', kernel='rbf'
AdaBoost	0.5882	0.4943	lr=0.001, n_estimators=5

Hyperparameter Tuning - Visualization









Optimal Model Selection

Regression

Model	R^2	RMSE	MAE	Best Hyperparameters
AdaBoost (Tuned)	0.7550	11.8457	7.3079	lr=0.001, loss='exponential', n_est=300

Classification

Model	Accuracy	F1-Score	Best Hyperparameters
XGBoost (Tuned)	0.7255	0.7037	colsample_bytree=0.8, lr=0.3, max_depth=1, n_est=150, subsample=0.8

Feature importance for Regression: `scope1+2total` (0.838), `brands` (0.135), `scope2emit...` (0.027). for classification with a better balance: `brands` (0.410), `scope1+2total` (0.330), `scope2emit...` (0.260). The `brands` feature proved valuable for risk categorization.

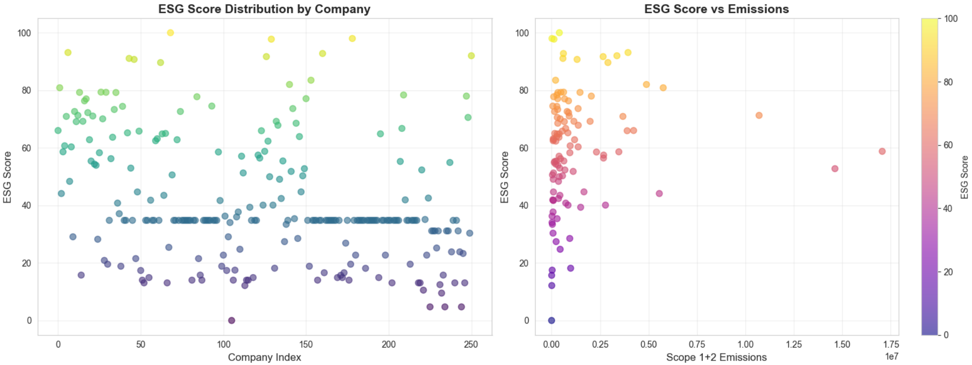
Conclusion

This project demonstrates **Supervised Learning** effectiveness for predicting **ESG scores** and categorizing **ESG Risk**, in special the power of ensemble models.

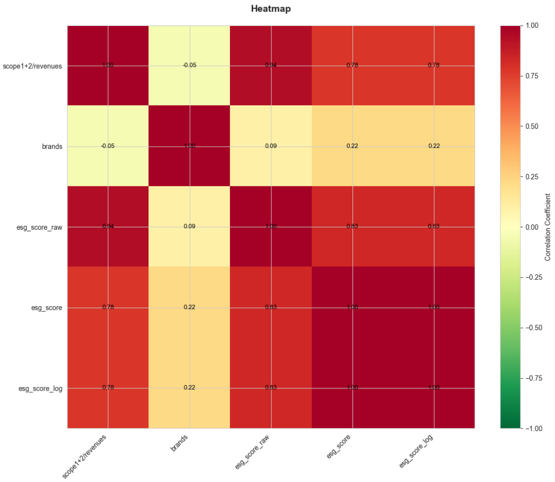
I was also able to meet the **expected deliverables** expectation: One optimal model for ESG Score prediction (regression) one optimal model for ESG Sustainability Risk classification (multi-class classification), a comparative analysis of model performance, hyperparameter optimization results, and feature importance analysis

This investigation highlights machine learning potential in **ESG analytics** and the critical need for diversified features. Future work should incorporate additional environmental, social, and governance variables for robust, interpretable models.

Appendix - Scatterplot



Appendix - Correlation Heatmap



Tools & Resources:

- **Scikit-Learn**: Machine learning library for classification, regression, and model evaluation
- **Matplotlib**: Comprehensive visualization library for creating static, animated, and interactive plots
- **Seaborn**: Statistical data visualization library based on Matplotlib
- **LaTeX Presentation Generator**: A generic Beamer presentation generator that extracts content from Jupyter notebooks. This tool was used to generate the initial structure of this presentation.

---

[1] The dataset used in this project originates from the **GHG Shopper** and **Stakeholder Takeover** initiatives.

[2] SVM as an alternative non-emsamble, non-tree based model.

[3] Used indirectly to calculate F-1 score.

[4] **Kaggle**: Platform for datasets, model sharing, and competitions. The **Introduction to Machine Learning** course influenced exploratory workflow and early modeling choices.

[5] Grigorev, Alexey. *Machine Learning Bookcamp: Build a portfolio of real-life projects*. Manning Publications, 2021. The use of XGBoost in this work was inspired by practical guidance and the dedicated chapter in this book.

[6] James, G., et al (2023). *An Introduction to Statistical Learning: with Applications in Python*. Springer. Source for statistical metrics and model evaluation formulas.

Bonus - Synthetic Dataset

A **synthetic dataset** containing over 50,000 rows has been generated for further experimentation.
This dataset can be utilized by modifying the Jupyter notebook accordingly:

```
trainer = SupervisedLearning(drop_columns=[... list of the columns you want to
    drop ...])

dataset_file = 'synthetic_ghg_data.csv'

trainer.load_data(
    filepath=dataset_file,
    scope_col='scope1+2total',
    revenue_col='revenuesghgco',
    realzero_col='realzero',
    normalize_columns=True
)
```