📊 ESG Normalization Functions

Environmental, Social, and Governance (ESG) ratings are reported by different agencies using **different scales and philosophies**. To aggregate them into a unified score, we must **normalize** them to a common scale, in this case **0–10**. This is done using **expert-defined buckets** or rules based on each agency's methodology.

1. `normalize_sustainalytics(score)`

What is Sustainalytics?

- A **risk-based** ESG scoring agency.
- Scores range from **0 (lowest risk)** to **100+ (highest risk)**.
- Lower scores are **better**.

Logic:

python
CopyEdit
```
if score <= 10: return 10.0      # Negligible risk
elif score <= 20: return 8.5     # Low risk
elif score <= 30: return 7.0     # Medium risk
elif score <= 40: return 5.0     # High risk
else: return 2.5                 # Severe risk
```

Explanation:

- The scores are **inversely mapped** (i.e., low risk = high score).
- The bucketization reflects **qualitative labels** used by Sustainalytics (e.g., Negligible, Low, Medium, etc.).
- Returns a normalized value between **2.5 and 10**, with **10** being best.

2. `normalize_lseg(score)`

What is LSEG?

- LSEG (London Stock Exchange Group) provides **performance-based** ESG scores.
- Higher is **better**.

Logic:

python
CopyEdit
```
if score >= 85: return 10.0      # Excellent
elif score >= 70: return 8.0     # Good
```

```
elif score >= 55: return 6.5    # Fair
elif score >= 40: return 4.5    # Weak
else: return 2.5                # Very Weak
```

Explanation:

- Direct mapping since LSEG scores are already **positively correlated** with ESG performance.
- Bucket thresholds are based on industry standards and empirical distributions.

3. `normalize_msci(rating)`

What is MSCI?

- MSCI provides ESG **letter ratings**: AAA (best) to CCC (worst).
- Qualitative rather than numeric, so mapping is needed.

Logic:

python
CopyEdit

```
rating_map = {
    'AAA': 10.0, 'AA': 8.5, 'A': 7.0, 'BBB': 6.0,
    'BB': 5.0, 'B': 3.5, 'CCC': 2.0
}
```

Explanation:

- Converts ordinal categories into **quantitative scale**.
- Values are assigned to reflect **relative ESG strength** in consistent intervals.

⚖️ Weighted ESG Score Calculation

python
CopyEdit

```
def compute_weighted_esg(sus, lseg, msci,
weights={'SUSTAINALYTICS': 0.3, 'LSEG': 0.4, 'MSCI': 0.3}):
```

This function computes a **consolidated ESG score (out of 10)** using a **weighted average**.

Key Steps:

1. **Score Dictionary**:
   python
   CopyEdit

```
scores = {'SUSTAINALYTICS': sus, 'LSEG': lseg, 'MSCI': msci}
```

2.

3. **Filter Valid Scores**:
   python
   CopyEdit

```
valid = {k: v for k, v in scores.items() if v is not None}
```

4.

   - Excludes missing/null scores.
   - Ensures we don't bias the result if one agency didn't report a rating.

5. **Dynamic Rebalancing of Weights**:
   python
   CopyEdit

```
total_weight = sum(weights[k] for k in valid)
```

6.

   - Normalizes weights so they still sum to 1, even when one or more sources are missing.

7. **Weighted Average Computation**:
   python
   CopyEdit

```
return round(sum(valid[k] * (weights[k] / total_weight) for k in valid), 2)
```

8.

   - Multiplies each score by its adjusted weight.
   - Ensures a fair combination of available data.

Example:

Source	Raw Score	Normalized	Weight
Sustainalytics	25	7.0	30%
LSEG	77	8.0	40%
MSCI	BBB	6.0	30%

All present → Normalized ESG Score =
`(0.3 * 7.0 + 0.4 * 8.0 + 0.3 * 6.0) = 7.1`

If MSCI is missing:
`(0.3 / 0.7 * 7.0 + 0.4 / 0.7 * 8.0) = 7.571`

`Other way which requires, Historical ESG data:`

We use a **supervised machine learning model** — **Random Forest Regression** — to **learn patterns** from historical ESG ratings and map them to a **single normalized ESG score**.

🏗️ Data Inputs

We use ESG data from the KnowESG platform. Each company has:

- **Sustainalytics**: Risk-based numeric score (lower is better).
- **LSEG (Refinitiv)**: Performance-based numeric score (higher is better).
- **MSCI**: Letter grade (AAA to CCC), indicating ESG risk category.
- **Target_ESG**: A manually curated or heuristically derived normalized score between **1–10**, used to train the model.

Sample record:

```
Company         | Sustainalytics | LSEG | MSCI | Target_ESG
------------------------------------------------------------
UBS             | 22.1           | 75   | AA   | 7.5
JPMorgan        | 16.5           | 85   | A    | 8.0
```

🔁 MSCI Encoding

MSCI's categorical grades are converted into numerical values using:

```
AAA → 10.0, AA → 8.5, A → 7.0, BBB → 6.0,
BB → 5.0, B → 3.5, CCC → 2.0
```

⚙️ Modeling Approach

- **Algorithm**: `RandomForestRegressor` from Scikit-learn.
- **Reasoning**:
  - Handles **non-linear relationships** well.
  - **Robust to outliers** and irrelevant features.
  - Can capture **interactions** between agency scores.
- **Training Pipeline**:
  - Inputs: Sustainalytics, LSEG, and encoded MSCI scores.
  - Output: Target_ESG (ground-truth normalized ESG score).
  - Splitting: 80% training, 20% test.
  - Model Evaluation: RMSE and R^2 score.

✅ Benefits of ML-Based Normalization

- **Learned weighting**: Model learns which agency contributes more to the ESG profile rather than fixed manual weights.
- **Non-linearity**: Unlike simple weighted averages, Random Forests can capture hidden thresholds and rules.
- **Flexibility**: Can generalize to new companies and adapt over time with retraining.
- **Imputation-friendly**: Can handle missing agency ratings by training with synthetic or real-world partially missing data.

📈 Evaluation Metrics

- **R^2 Score**: Measures how well the model explains variance in the normalized ESG score.
- **RMSE**: Reflects how far predicted scores deviate from actual (lower is better).

Example:

```
R² Score: 0.89
RMSE: 0.42
```

🧪 Inference (Prediction)

Once trained, the model can predict the normalized ESG score for new companies using just:

- `Sustainalytics score`
- `LSEG score`
- `MSCI letter rating`

Example:

```
python
CopyEdit
predict_esg(model, sustainalytics=18.5, lseg=80,
msci_letter='AA') → 8.12
```

`How to calculate the sentiment score from ESG reports:`

Here's how such a sentiment score is calculated:

🔷 Step-by-Step Process:

1. **Input Text Collection**:
   - Source data: ESG reports
2. **Text Preprocessing**:
   - Lowercasing, punctuation removal
   - Stopword removal
   - Lemmatization or stemming
3. **Sentiment Model Application**:
   Typically, one of the following is used:
   - **ML/DL-based model** (e.g., BERT, RoBERTa fine-tuned for sentiment):

        - Text is passed through a transformer model.
        - Output is a sentiment class (positive/neutral/negative) with a **confidence score**.
        - A mapping like `Positive → 1`, `Neutral → 0`, `Negative → -1` is used.
        - Average score across documents gives the "**Average Sentiment Score**".

4. **Score Aggregation**:
    - Sentiment scores from multiple documents for the same company are **averaged**.
    - Final result is a float between `-1.0` and `1.0` (or sometimes `0` to `1`, depending on model).

Based on your `average_sentiment.txt`, a sample line might look like:

```
Average Sentiment Score: 0.16
```

This suggests:

- The score was already **pre-aggregated**.
- It's likely between `-1` to `1`, where:
    - **Positive values** → Positive sentiment (more favorable ESG narrative)
    - **Negative values** → Negative sentiment (greenwashing or controversy concerns)
    - **0** → Neutral tone

We can then **scale** this score in your greenwashing anomaly detection logic:

```
discrepancy = avg_esg - (sentiment_score * 10)
```

```
How this Sentiment score and ESG data is used for Anomaly
detection:
```

4. Sentiment Score Ingestion

The `average_sentiment.txt` file contains a sentiment score extracted from company ESG reports using NLP techniques (e.g., FinBERT or VADER). It typically ranges between **-1 to +1**.

We scale it to align with ESG (e.g., `sentiment_score * 15`) and use it as a soft signal of **narrative tone** around ESG performance.

5. Anomaly Detection Using Isolation Forest

What is an Anomaly?

An anomaly, in this context, occurs when a company's reported ESG scores **contradict the public sentiment** derived from text (e.g., inflated scores but negative media coverage = possible greenwashing).

Algorithm:

We use **Isolation Forest**, a tree-based unsupervised ML algorithm that isolates outliers:

- 500 synthetic data points are generated for normal ESG & sentiment ranges.
- The actual company's ESG and sentiment score are tested against this trained model.
- If flagged as `-1`, it's an anomaly.

```
IsolationForest(n_estimators=150, contamination=0.05)
```

Why Isolation Forest?

- Effective on small datasets.
- Works well with multidimensional data.
- Robust to outliers and doesn't require labels.