LIFESTYLE FACTORS AND GENERAL HEALTH: A PREDICTIVE ANALYSIS

Pamela Claridy
Georgia Institute of Technology
Atlanta, GA

March 24, 2024

ABSTRACT

For this study, I am analyzing the impact of various lifestyle factors on a person's self-assessed general health. The general health variable is classified into two categories, good or better health and fair or poor health. The lifestyle factors utilized for this study are physical activity, smoking status, alcohol consumption, heart disease, asthma, age, gender, race, education, income, and BMI. The data utilized for this study is the comprehensive 2022 Behavioral Risk Factor Surveillance System (BRFSS) dataset. This dataset contains responses across the United States and its territories. This project seeks to uncover patterns and correlations that can help to support public health initiatives. Throughout this project, I performed rigorous statistical analysis. This study involved the application of data mining techniques, regression analysis, machine learning models, clustering, and pattern recognition. This has provided the ability to develop a better understanding of the relationship between lifestyle behaviors and general health across different demographics.

The findings of the analysis reveal associations between lifestyle factors and the self-assessed general health status which in-turn highlights demographic disparities. Random Forest and Gradient Boosting were used as predictive models and they both provided good accuracy. Clustering analysis, performed as part of my baseline models, displayed patterns that suggest potential targets for tailored public health initiatives.

This study offers a data driven approach for developing targeted health interventions. With the focus on general health rather than a specific disease like heart disease or obesity, allowed for a broader understanding of the lifestyle factors. Future studies could expand this approach by focusing on long-term effects of lifestyle changes on general health status, continuing this study by developing data-informed strategies to promote better health.

1 Introduction

1.1 Problem Description and Motivation

The bigger public health issue is general health status, a global public health concern which encompasses the overall picture of overall health, not limited to certain chronic diseases only. This project is driven by the hypothesis that lifestyle factors, like physical activity, smoking answers, alcohol drinking, diet and others, have a substantial impact on person's perception of general health status. The research looks further into the relationships between these lifestyle behaviours and the differences in how general health is perceived.

1.2 Data Mining Challenges

One of the challenges in this study, which is common to many data mining studies, was dealing with a plethora of variables. The BRFSS (the large national health survey from which I developed my dataset) captures a broad array of data collection from the United States and its territories spanning numerous demographics. It is important to make sure that entered data into the system are both accurate and complete. It is also important to select variables (or features) that

are relevant to the research question of interest. Another challenge is with modeling, in particular, carefully selecting and training the machine learning algorithm in order to balance between how complex the model should be and how accurately it can predict events.

1.3 Problem Solving Strategies

To navigate these challenges, I utilized a wide range of data mining and statistical learning techniques:

- **Regression Analysis** explored the connections between lifestyle behaviors and general health status.
- **Machine Learning Models**, such as decision trees, random forests, and gradient boosting, were tailored to forecast general health outcomes from lifestyle factors.
- **Clustering and Pattern Recognition** methods were used to discover patterns in lifestyle behaviors and their links to general health.

1.4 Accomplished Learning from the Applications

The application of these diverse methods led to some striking findings. The research underscored the important role of lifestyle factors in shaping health perceptions, and also identified important variations by demographics. The development of predictive models also demonstrated how data mining can be used in public health for modeling future health outcomes based on modifiable lifestyle factors.

1.5 Outline of the Report

The report is structured to provide a comprehensive understanding of the research conducted. The outline is as follows:

1. **Introduction** includes a summary of why this is an important study, and gives background on why lifestyle factors are important to general health, and how a public-use dataset mined from the Behavioral Risk Factor Surveillance System (BRFSS) is a specific challenge.
2. **Problem Statement and Data Sources** elaborates on the objectives of the study, describing the BRFSS dataset and the selected variables for analysis.
3. **Methodology** details the step-by-step process of data cleaning, the creation of basic models as a baseline, the advanced machine learning models using Random Forest and Gradient Boosting to improve the performance of the model, and the choice of hyperparameters in those models.
4. **Analysis and Results** reports the results of the model tests, both the predictive accuracies and the statistical significance.
5. **Conclusion** summarizes the key insights derived from the study, emphasizing the effectiveness of the Gradient Boosting model.
6. **Appendix: Python Code** provides detailed Python code used for data cleaning and preprocessing, baseline model development, and the implementation of Random Forest and Gradient Boosting models which include different scenarios with and without hyperparameter tuning.

2 Problem Statement and Data Sources

Understanding how modifiable lifestyle factors can influence general health status is becoming increasingly relevant in the context of public health. The goal of this study is to analyze the relationships between general health and lifestyle factors such as physical activity, smoking status, alcohol consumption, diet, asthma status, race, age, sex, BMI, income, and education. The investigation will be conducted leveraging the 2022 Behavioral Risk Factor Surveillance System (BRFSS) dataset, provided by the Centers for Disease Control and Prevention (CDC). The goal of this research is to understand patterns and correlations that might provide pathways to improve general health through modifiable lifestyle factors.

The BRFSS dataset provides an extensive number of responses collected from a diverse population across the 50 states and territories. The data covers a wide range of health-related behaviors. Detailed information about the dataset, its codebook, summary data quality reports, and the methodologies behind variable collection, is accessible via the CDC's dedicated webpage:

`https://www.cdc.gov/brfss/annual_data/annual_2022.html`

For this analysis, the selected variables include:

- **General Health (`'GenHealth'`)**: A subjective assessment of overall health, categorized good or better health (1) versus fair or poor health (2).
- **Physical Activity (`'PhysAct'`)**: Engagement levels in physical activities, indicative of an active versus sedentary lifestyle.
- **Smoking Status (`'SmokeStatus'`)** and **Alcohol Intake (`'AlcoholIntake'`)**: Behaviors related to smoking and alcohol consumption.
- **Heart Disease Status (`'HeartDisease'`)**: Presence of heart disease diagnosis.
- **Asthma Status (`'AsthmaStatus'`)**: Presence of asthma diagnosis.
- **Demographic Information**: Such as `'AgeGroup'`, `'Gender'`, `'RaceEthnicity'`, `'EduLevel'`, and `'IncomeLevel'`.
- **Body Mass Index (`'BMI'`)**: Reflecting nutritional and health status through categorized BMI values.

This study intends to utilize the BRFSS dataset to determine how lifestyle factors correlate with general health status. The goal is to generate insights to inform public health interventions, ultimately striving to improve the well-being and health outcomes of the broader population.

3 Methodology

The methodology for this study entails several steps, including data cleaning and preprocessing, baseline model development, and advanced model training and evaluation.

3.1 Data Cleaning and Preprocessing

The data cleaning and preprocessing of the dataset ensured the dataset is suitable for analysis. This included reading the dataset, selecting relevant columns, handling missing values, and removing outliers. Renaming the columns to more relevant names and converting data types was also apart of this process. Details of the data cleaning and preprocessing steps can be found in Appendix A.

3.2 Baseline Model Development

After the data cleaning and preprocessing, I developed baseline models using K-Nearest Neighbors (KNN), Logistic Regression, and Decision Trees. These models serve as a starting point for evaluating the predictive performance of the dataset. Details of the baseline model development can be found in Appendix B.

3.3 Random Forest and Gradient Boosting

Next, I applied machine learning techniques, Random Forest and Gradient Boosting, to improve model performance. These models are trained and evaluated using the preprocessed dataset to assess their predictive accuracy. I performed hyperparameter tuning using techniques like Grid Search and Randomized Search to further enhance model performance. This process optimized the model parameters to improve predictive accuracy. Details of the Random Forest and Gradient Boosting models can be found in Appendix C.

3.4 Rationale for Hyperparameter Selection

The selection of hyperparameters for both the Random Forest and Gradient Boosting models entailed a systematic approach aimed at optimizing predictive performance while preventing overfitting. This was perform through cross-validation applied to the training dataset, which allowed for an evaluation of each model's performance across various hyperparameter values.

For the Random Forest model, the choice of `n_estimators`, `max_depth`, `min_samples_split`, and `min_samples_leaf` was informed by a Grid Search CV process. This approach evaluated the model's accuracy across different combinations of the parameters, selecting the parameters that provided the highest accuracy score. The `n_estimators` parameter was employed to determine the optimal number of trees that contribute without excessively increasing computational time. Similarly, `max_depth` was adjusted to find the proper depth that captures the data's underlying patterns without overfitting.

The Gradient Boosting model went through a similar tuning process with parameters such as `n_estimators`, `learning_rate`, `max_depth`, `min_samples_split`, and `min_samples_leaf`. The learning rate's adjustment was particularly critical in this process as it controls the extent to which each tree's corrections influence subsequent trees. This is a key factor in preventing overfitting while improving predictive accuracy.

In both cases, cross-validation served as the cornerstone of the hyperparameter tuning process. This ensured that the selected parameters performed consistently well across different subsets of the training data.

4 Analysis and Results

In my approach to obtain the minimal test error and ultimately more precise predictive accuracy, I resorted to employing different methods for data mining in which I would be applying different models such as K-Nearest Neighbors (KNN), Logistic Regression, Decision Tree, Random Forest, and Gradient Boosting against a cleaned data set to then test against each model's testing accuracy with emphasis on ensemble methods specifically given their high-performance potential.

My initial findings indicated that Logistic Regression led with a testing accuracy of 85.34%, closely followed by KNN at 84.22%. The Decision Tree model presented a lower accuracy of 81.15%. Prior to tuning, the Gradient Boosting model outperformed Random Forest with accuracies of 85.61% and 83.53%, respectively. These results prompted further investigation through hyperparameter tuning.

Upon tuning, improvements were observed in both the Random Forest and Gradient Boosting models. The first tuning phase improved Random Forest's accuracy to 85.64% and Gradient Boosting's to 85.79%. While I considered further model refinement, the subsequent tuning processes were much more time-consuming.

Considering the efficiency and time constraints, I opted to utilize the models obtained from the initial tuning phase. I selected the first tuned Gradient Boosting model which demonstrated a slight advantage in testing accuracy at 85.79%. This decision provides a reasonable balance between high predictive accuracy and computational efficiency and provides an optimal approach for minimizing test error. This aligned with my objectives for accuracy and efficiency in model development.

5 Conclusion

This study of the 2022 Behavioral Risk Factor Surveillance System (BRFSS) dataset provides insights into the significant capability of machine learning models to predict general health outcomes from lifestyle factors. The evaluation of K-Nearest Neighbors (KNN), Logistic Regression, Decision Trees, Random Forest, and Gradient Boosting models revealed notable performance variability. Tuning of the Gradient Boosting model provided the model with the highest predictive accuracy.

The hyperparameter tuning phase was critical in this process as it demonstrated that strategic adjustments to model parameters improves model performance. This process demonstrated the importance of model optimization through balancing computational resources and predictive accuracy.

The decision to choose the initial tuned Gradient Boosting model, with a testing accuracy of 85.79%, was based on the goal to provide the best balance between predictive accuracy and operational efficiency. This accuracy level indicates that the model can correctly predict an individual's self-assessed general health status approximately 85.79% of the time. This prediction considers variables such as sex, age, BMI, income level, education level, smoking status, alcohol consumption, asthma status, and heart disease status. This high degree of accuracy displays the model's effectiveness in utilizing the lifestyle factors to predict the self-assessed general health.

The insights obtained from this study can enhance the understanding of how lifestyle choices affect general health which can provide data-driven, machine learning approaches in the development of public health initiatives. Leveraging these types of insights from the Gradient Boosting model can enable the design of targeted interventions aimed at improving general health outcomes. Additionally, this study lays the groundwork for future research to explore more variables to utilize advanced modeling techniques and fully explore the ways to promote healthier lifestyles.

Appendix: Python Code

A Data Cleaning and Preprocessing

```
import pandas as pd

# Read XPT file
df = pd.read_sas('LLCP2022.XPT␣', format='xport')
df.head()

# Create a list of columns to keep
core_columns = ['_RFHLTH', '_TOTINDA', '_SMOKER3', '_DRNKWK2', '_MICHD', '
    _LTASTH1',
                '_AGEG5YR', '_SEX', '_RACEPR1', '_EDUCAG', '_INCOMG1', '
                    _BMI5CAT']

# Subset the DataFrame
df_subset = df[core_columns]
df_subset.head()

# Change the column names
shortened_columns = {
    '_RFHLTH': 'GenHealth',
    '_TOTINDA': 'PhysAct',
    '_SMOKER3': 'SmokeStatus',
    '_DRNKWK2': 'AlcoholIntake',
    '_MICHD': 'HeartDisease',
    '_LTASTH1': 'AsthmaStatus',
    '_AGEG5YR': 'AgeGroup',
    '_SEX': 'Gender',
    '_RACEPR1': 'RaceEthnicity',
    '_EDUCAG': 'EduLevel',
    '_INCOMG1': 'IncomeLevel',
    '_BMI5CAT': 'BMI'
}

# Apply renaming
data = df_subset.rename(columns=shortened_columns)
print(data.columns)
print(data.head())
print(data.shape)

# Counting NaN values in each column
nan_counts = data.isna().sum()
print(nan_counts)

# Dropping rows where any of the three specific columns have NaN values
data_cleaned = data.dropna(subset=['SmokeStatus', 'HeartDisease', 'BMI'])
print(data_cleaned.shape)

# Counting values out of the specified range in the AlcoholIntake column
out_of_range_count = ((data['AlcoholIntake'] > 98999) | (data['
    AlcoholIntake'] == 99900)).sum()

print(f"Number␣of␣AlcoholIntake␣values␣out␣of␣the␣specified␣range:␣{
    out_of_range_count}")

# Deleting rows where AlcoholIntake values are out of the specified range
```

```
data_filtered = data_cleaned[~((data_cleaned['AlcoholIntake'] > 98999) | (
    data_cleaned['AlcoholIntake'] == 99900))]

print(f"Shape of the original data: {data_cleaned.shape}")
print(f"Shape of the data after removing out-of-range AlcoholIntake values
    : {data_filtered.shape}")

# Converting AlcoholIntake values from float to integer
data_filtered['AlcoholIntake'] = data_filtered['AlcoholIntake'].astype(int
    )

# Display the head of the DataFrame to verify the conversion
data_filtered.describe()

# Calculate z-scores for all numeric columns
z_scores = data_filtered.apply(zscore)

# Identify rows with any column having z-score > 3 or < -3 (outliers)
outliers_mask = (z_scores.abs() > 3).any(axis=1)

# Remove rows that have outliers in any column
data_no_outliers = data_filtered[~outliers_mask]

# Rename the DataFrame
cleaned_data = data_no_outliers

# Save the cleaned data to a new CSV file
cleaned_data.to_csv('cleaned_data.csv', index=False)
```

B Baseline Model Development

```
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

# Load the cleaned data
data = pd.read_csv('cleaned_data.csv')

# Set 'GenHealth' is your target variable
X = data.drop('GenHealth', axis=1)
y = data['GenHealth']

# Ensure 'GenHealth' is encoded as categorical
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded,
    test_size=0.3, random_state=42)

# Initialize baseline models
knn = KNeighborsClassifier()
logreg = LogisticRegression(max_iter=1000)  # Increase max_iter value
decision_tree = DecisionTreeClassifier()
```

```
# Train baseline models
knn.fit(X_train, y_train)
logreg.fit(X_train, y_train)
decision_tree.fit(X_train, y_train)

# Evaluate baseline models using cross-validation
knn_cv_accuracy = cross_val_score(knn, X, y_encoded, cv=5).mean()
logreg_cv_accuracy = cross_val_score(logreg, X, y_encoded, cv=5).mean()
decision_tree_cv_accuracy = cross_val_score(decision_tree, X, y_encoded,
    cv=5).mean()

print(knn_cv_accuracy)
print(logreg_cv_accuracy)
print(decision_tree_cv_accuracy)
```

C Random Forest and Gradient Boosting Models

```
# Scenario 1 (without hyperparameter tuning)
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier,
    GradientBoostingClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
     f1_score
from sklearn.preprocessing import LabelEncoder

# Load the cleaned data
data = pd.read_csv('cleaned_data.csv')

# Set 'GenHealth' is your target variable
X = data.drop('GenHealth', axis=1)
y = data['GenHealth']

# Ensure 'GenHealth' is encoded as categorical
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded,
    test_size=0.3, random_state=42)

# Initialize the models
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
gb_model = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,
     max_depth=1, random_state=42)

# Train the models
rf_model.fit(X_train, y_train)
gb_model.fit(X_train, y_train)

# Make predictions
rf_predictions = rf_model.predict(X_test)
gb_predictions = gb_model.predict(X_test)

# Evaluate performance
print("Random␣Forest␣Performance:")
print(f"Accuracy:␣{accuracy_score(y_test,␣rf_predictions):.4f}")
```

```
print(f"Precision (Weighted): {precision_score(y_test, rf_predictions,
    average='weighted'):.4f}")
print(f"Recall (Weighted): {recall_score(y_test, rf_predictions, average='
    weighted'):.4f}")
print(f"F1 Score (Weighted): {f1_score(y_test, rf_predictions, average='
    weighted'):.4f}")

print("\nGradient Boosting Performance:")
print(f"Accuracy: {accuracy_score(y_test, gb_predictions):.4f}")
print(f"Precision (Weighted): {precision_score(y_test, gb_predictions,
    average='weighted'):.4f}")
print(f"Recall (Weighted): {recall_score(y_test, gb_predictions, average='
    weighted'):.4f}")
print(f"F1 Score (Weighted): {f1_score(y_test, gb_predictions, average='
    weighted'):.4f}")

# Scenario 2 (with hyperparameter tuning)
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier,
    GradientBoostingClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
     f1_score
from sklearn.preprocessing import LabelEncoder

# Load the cleaned data
data = pd.read_csv('cleaned_data.csv')

# Set 'GenHealth' is your target variable
X = data.drop('GenHealth', axis=1)
y = data['GenHealth']

# Ensure 'GenHealth' is encoded as categorical
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded,
    test_size=0.3, random_state=42)

# Hyperparameter tuning for Random Forest
rf_param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}
rf = RandomForestClassifier(random_state=42)
rf_grid_search = GridSearchCV(estimator=rf, param_grid=rf_param_grid, cv
    =3, n_jobs=-1, verbose=2, scoring='accuracy')
rf_grid_search.fit(X_train, y_train)
best_rf_model = rf_grid_search.best_estimator_

# Hyperparameter tuning for Gradient Boosting
gb_param_grid = {
    'n_estimators': [100, 200],
    'learning_rate': [0.01, 0.1],
    'max_depth': [3, 5],
    'min_samples_split': [2, 5],
```

```
    'min_samples_leaf': [1, 2]
}
gb = GradientBoostingClassifier(random_state=42)
gb_grid_search = GridSearchCV(estimator=gb, param_grid=gb_param_grid, cv
    =3, n_jobs=-1, verbose=2, scoring='accuracy')
gb_grid_search.fit(X_train, y_train)
best_gb_model = gb_grid_search.best_estimator_

# Make predictions with the best models
rf_predictions = best_rf_model.predict(X_test)
gb_predictions  = best_gb_model.predict(X_test)

# Evaluate performance
print("Tuned Random Forest Performance:")
print(f"Accuracy: {accuracy_score(y_test, rf_predictions):.4f}")
print(f"Precision (Weighted): {precision_score(y_test, rf_predictions,
    average='weighted'):.4f}")
print(f"Recall (Weighted): {recall_score(y_test, rf_predictions, average='
    weighted'):.4f}")
print(f"F1 Score (Weighted): {f1_score(y_test, rf_predictions, average='
    weighted'):.4f}")

print("\nTuned Gradient Boosting Performance:")
print(f"Accuracy: {accuracy_score(y_test, gb_predictions):.4f}")
print(f"Precision (Weighted): {precision_score(y_test, gb_predictions,
    average='weighted'):.4f}")
print(f"Recall (Weighted): {recall_score(y_test, gb_predictions, average='
    weighted'):.4f}")
print(f"F1 Score (Weighted): {f1_score(y_test, gb_predictions, average='
    weighted'):.4f}")

# Scenario 3 (with further hyperparameter tuning)
import pandas as pd
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier,
    GradientBoostingClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
     f1_score
from sklearn.preprocessing import LabelEncoder
from scipy.stats import randint, uniform

# Load the cleaned data
data = pd.read_csv('cleaned_data.csv')

# Set 'GenHealth' is your target variable
X = data.drop('GenHealth', axis=1)
y = data['GenHealth']

# Ensure 'GenHealth' is encoded as categorical
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded,
    test_size=0.3, random_state=42)

# Expanded Random Forest parameter grid
rf_param_dist = {
    'n_estimators': randint(100, 500),
```

```
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': randint(2, 11),
    'min_samples_leaf': randint(1, 11),
    'max_features': ['sqrt', 'log2', None]
}

# Expanded Gradient Boosting parameter grid
gb_param_dist = {
    'n_estimators': randint(100, 500),
    'learning_rate': uniform(0.01, 0.2),
    'max_depth': randint(3, 10),
    'min_samples_split': randint(2, 11),
    'min_samples_leaf': randint(1, 11),
    'max_features': ['sqrt', 'log2', None]
}

# Random Forest Randomized Search
rf_random_search = RandomizedSearchCV(
    estimator=RandomForestClassifier(random_state=42),
    param_distributions=rf_param_dist,
    n_iter=100, cv=3, verbose=2, random_state=42, n_jobs=-1, scoring='
        accuracy'
)
rf_random_search.fit(X_train, y_train)
best_rf_model = rf_random_search.best_estimator_

# Gradient Boosting Randomized Search
gb_random_search = RandomizedSearchCV(
    estimator=GradientBoostingClassifier(random_state=42),
    param_distributions=gb_param_dist,
    n_iter=100, cv=3, verbose=2, random_state=42, n_jobs=-1, scoring='
        accuracy'
)
gb_random_search.fit(X_train, y_train)
best_gb_model = gb_random_search.best_estimator_

# Evaluate the tuned models
rf_predictions = best_rf_model.predict(X_test)
gb_predictions = best_gb_model.predict(X_test)

print("Tuned␣Random␣Forest␣Performance:")
print(f"Accuracy:␣{accuracy_score(y_test,␣rf_predictions):.4f}")
print(f"Precision␣(Weighted):␣{precision_score(y_test,␣rf_predictions,␣
    average='weighted'):.4f}")
print(f"Recall␣(Weighted):␣{recall_score(y_test,␣rf_predictions,␣average='
    weighted'):.4f}")
print(f"F1␣Score␣(Weighted):␣{f1_score(y_test,␣rf_predictions,␣average='
    weighted'):.4f}")

print("\nTuned␣Gradient␣Boosting␣Performance:")
print(f"Accuracy:␣{accuracy_score(y_test,␣gb_predictions):.4f}")
print(f"Precision␣(Weighted):␣{precision_score(y_test,␣gb_predictions,␣
    average='weighted'):.4f}")
print(f"Recall␣(Weighted):␣{recall_score(y_test,␣gb_predictions,␣average='
    weighted'):.4f}")
print(f"F1␣Score␣(Weighted):␣{f1_score(y_test,␣gb_predictions,␣average='
    weighted'):.4f}")
```