Predicting National Alcohol Consumption Using Socioeconomic and Health Indicators

October 8, 2024

1 Introduction

Alcohol has long played a significant role in social and cultural contexts, both positively and negatively. While moderate consumption can contribute to social engagement, excessive consumption is a known risk factor for various health issues and crime. Understanding the patterns of alcohol consumption is crucial for public health, policy development, and education.

In this project, I aim to predict alcohol consumption based on a variety of socio-economic and health-related factors using machine learning techniques. The project focuses on evaluating the performance of different machine learning models to effectively predict alcohol consumption, which could inform future interventions and policies.

This report is structured as follows: Section 1 (Introduction) provides background and context of the project. Section 2 (Problem Formulation) describes how this task is translated into a machine learning problem and introduces the dataset. Section 3 (Methods) covers the dataset, preprocessing, model selection, and the reasoning behind the choices made. Section 4 (Results) presents the findings from the model evaluations and comparisons. Finally, Section 5 (Conclusion) summarizes the key insights.

2 Problem Formulation

This project aims to predict alcohol consumption per capita across different countries based on features such as GDP per capita, adult mortality and BMI. The goal is to create a regression model that can estimate alcohol consumption as a continuous variable, making this a supervised machine learning task.

The dataset used for this project is titled "Life Expectancy (WHO) Fixed", created by Lasha Gochiashvili, and obtained on Kaggle [2]. Each data point in the dataset represents a specific country in a given year, with data spanning from 2000 to 2015. It includes 21 variables and 2864 data points covering 179 countries. The dataset provides information on demographic, health, and economic factors that influence life expectancy and other health metrics. The dataset contains categorical data (e.g., region, country), continuous data (e.g., infant deaths, life expectancy), and binary data (e.g., whether the country is developing or developed). There are no missing values in the dataset, as they have already been handled by the author using strategies such as closest three-year average and average of the region.

For this project, I have selected the following features to predict alcohol consumption: region (categorical, e.g., Africa, South America), GDP per capita (measured in USD), schooling (measured as average years that people aged 25+ spent in formal education), adult mortality (representing deaths of adults per 1000 population), and BMI (expressed in kg/m^2, representing the average body mass index of the population). The label value is alcohol consumption, recorded as liters of pure alcohol per person aged 15+ years.

3 Methods

3.1 Dataset

The dataset consists of 2,864 records from 179 countries, spanning the years 2000 to 2015, as described in the Problem Formulation section. Before applying machine learning models, some preprocessing steps were necessary to ensure the data was in optimal format.

Initially, data integrity was verified to confirm that there were no missing values, as stated. Next, categorical variables, specifically region, were transformed into numerical formats suitable for machine learning algorithms. This process, known as one-hot encoding, creates binary columns for each category. For instance, if a country is in the "European Union" region, the "Region European Union" column will have a value of 1, while other region columns will have a value of 0 for that record. One-hot encoding allows the model to process categorical data effectively. [1]

For the regression modeling, feature selection was guided by both data analysis and domain knowledge. The chosen features — region, GDP per capita, schooling, adult mortality, and BMI — were selected for their relevance and potential impact on alcohol consumption. These features were deemed important based on their correlations with the target variable and their significance in the literature on alcohol consumption [4].

For evaluating the models' performance, I selected k-fold cross-validation. This method is especially advantageous for datasets of moderate size, such as mine, which consists of 2,864 data points. K-fold cross-validation divides the dataset into k equally sized subsets (folds). The model is trained on $k-1$ of these folds and validated on the remaining fold. This process is repeated k times, each time using a different fold for validation, and the results are averaged to provide a robust estimate of model performance. In this case, I used $k = 5$, a commonly selected value that balances computational efficiency and validation robustness. [3]

3.2 Linear Regression Model

I selected Linear Regression as the first machine learning model to predict alcohol consumption. Linear Regression was chosen because it is a simple yet powerful model for predicting a continuous target variable, such as alcohol consumption. The features selected for this task, including GDP per capita, schooling, adult mortality, BMI, and regional categories, have the potential to exhibit linear relationships with the target variable. Linear Regression also provides coefficients that can be directly related to the importance of each feature in predicting alcohol consumption. This is valuable for understanding how different socio-economic and health-related factors contribute to alcohol consumption. [3]

I chose Mean Squared Error (MSE) as the loss function because it is a standard metric for regression problems that effectively measures the average squared difference between predicted and actual values. MSE penalizes larger errors more severely, which is useful for ensuring that significant deviations from the true values are minimized. Its widespread use and computational simplicity make it an appropriate choice for evaluating the accuracy of the Linear Regression model. [3]

3.3 Random Forest Regression Model

The second machine learning model I decided to use was Random Forest Regression. Random Forests were chosen due to their ability to capture non-linear relationships and complex interactions between the features, which is important in modeling alcohol consumption, as the relationship between the socio-economic and health-related features and alcohol consumption may not be strictly linear. The Random Forest model is an ensemble learning method that builds a collection of decision trees, each trained on random subsets of the data and features. The final prediction is based on the average of the predictions from these trees, which helps reduce overfitting and improves generalization compared to using a single decision tree. This makes it well-suited for datasets like mine, where complex relationships may exist between features such as GDP per capita, schooling, and adult mortality. [5]

An additional advantage of Random Forests is their ability to provide feature importance rankings, offering insights into which features are most influential in predicting alcohol consumption. This helps in understanding which factors, such as GDP or regional classifications, have the greatest impact on alcohol consumption. [5]

As with the Linear Regression model, I used Mean Squared Error (MSE) as the loss function. This decision was made to maintain consistency across models, allowing for a direct comparison of performance. As mentioned in the Linear Regression Model section, MSE is a standard metric in regression tasks that measures the average squared difference between the actual and predicted values.

4 Results

In this project, both Linear Regression and Random Forest Regression were used to predict alcohol consumption based on socio-economic and health-related indicators. The performance of each model was evaluated using k-fold cross-validation, resulting in a robust estimate of model performance through the use of multiple train-validation splits. Following this, a separate test set was used to assess the final model's generalization ability.

4.1 Linear Regression Model Results

For the Linear Regression model, the average training MSE was calculated to be 5.01, while the average validation MSE was 5.09. The close values indicate that the model is generalizing well and is not significantly overfitting. The root of the validation MSE is 2.26, which is in the same units as alcohol consumption. Given that the values of alcohol consumption in the data set vary between 0 and 18, the root of MSE represents approximately 13% of the maximum value in the dataset. Considering these factors, the Linear Regression model performs reasonably well.

4.2 Random Forest Regression Model Results

In contrast, the Random Forest Regression model achieved an average training MSE of 0.16 and a validation MSE of 1.07. Although the validation MSE is significantly higher than the training MSE, the model is generalizing reasonably well, as the validation MSE remains quite low. The root of the validation MSE is 1.03, which represents approximately 6% of the maximum value in the dataset. Overall, the Random Forest Regression model performs exceptionally well.

4.3 Comparing the Models

Based on the comparative analysis of validation errors, the Random Forest Regression model was chosen as the final method for predicting alcohol consumption. While both models performed reasonably well, Random Forest's ability to capture complex, non-linear relationships in the data provided a significant performance boost, as reflected in its much lower validation MSE. Specifically, the Random Forest model achieved a validation MSE of 1.07, which is substantially lower than the 5.09 validation MSE observed for the Linear Regression model.

The Random Forest model's higher complexity allowed it to better model the non-linear relationships between socio-economic and health-related indicators and alcohol consumption. On the other hand, the Linear Regression model's simplicity makes it easier to interpret, but it struggled to capture these complexities, leading to higher prediction errors.

4.4 Test Error of the Chosen Model

After conducting k-fold cross-validation, the Random Forest model was fitted on the entire training set (80% of the entire dataset) and evaluated on a separate test set (20% of the entire dataset). The test set performance yielded a test MSE of 0.71, and a corresponding root MSE of 0.84, which represents approximately 5% of the maximum value in the dataset.

These results indicate that the model generalizes well on unseen data, as evidenced by the low test MSE and root MSE values. The model's performance is consistent, as the validation MSE values range from 0.96 to 1.29. This consistency suggests that the model is robust in predicting alcohol consumption.

5 Conclusion

This project aimed to predict national alcohol consumption using socio-economic and health indicators through Linear Regression and Random Forest Regression models. Both models were evaluated using k-fold cross-validation, with the Random Forest model validated on a separate test set.

The Random Forest model outperformed Linear Regression, demonstrating superior predictive power and effectively capturing the complex relationships within the data. Its performance was further confirmed on the test set, validating its ability to generalize well to unseen data.

Despite these positive results, there is still room for improvement. Expanding the feature set to include cultural or policy-related factors could enhance the model's predictive capabilities. Additionally, utilizing advanced hyperparameter tuning techniques or exploring alternative machine learning methods, such as gradient boosting or neural networks, may lead to further enhancements in prediction accuracy.

References

[1] Ana Rojo Echeburúa. What is one hot encoding and how to implement it in python, 2024. URL https://www.datacamp.com/tutorial/one-hot-encoding-python-tutorial.

[2] Lasha Gochiashvili. Life Expectancy (WHO) Fixed, 2023. URL https://www.kaggle.com/ds/3065197.

[3] A. Jung. *Machine Learning: The Basics*. Springer, Singapore, 2022.

[4] Hannah Ritchie and Max Roser. Alcohol consumption. *Our World in Data*, 2022. URL https://ourworldindata.org/alcohol-consumption.

[5] Sumbatilinda. Random forests regression by example. *Medium*, 2024. URL https://medium.com/@sumbatilinda/random-forests-regression-by-example-1baa062506f5.

alcoholConsumption

October 8, 2024

[1]:
```
import pandas as pd
import numpy as np

# Load the data set
raw_df = pd.read_csv('Life-Expectancy-Data-Updated.csv')

# Display basic info (to verify no values are missing)
raw_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2864 entries, 0 to 2863
Data columns (total 21 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Country                       2864 non-null   object
 1   Region                        2864 non-null   object
 2   Year                          2864 non-null   int64
 3   Infant_deaths                 2864 non-null   float64
 4   Under_five_deaths             2864 non-null   float64
 5   Adult_mortality               2864 non-null   float64
 6   Alcohol_consumption           2864 non-null   float64
 7   Hepatitis_B                   2864 non-null   int64
 8   Measles                       2864 non-null   int64
 9   BMI                           2864 non-null   float64
 10  Polio                         2864 non-null   int64
 11  Diphtheria                    2864 non-null   int64
 12  Incidents_HIV                 2864 non-null   float64
 13  GDP_per_capita                2864 non-null   int64
 14  Population_mln                2864 non-null   float64
 15  Thinness_ten_nineteen_years   2864 non-null   float64
 16  Thinness_five_nine_years      2864 non-null   float64
 17  Schooling                     2864 non-null   float64
 18  Economy_status_Developed      2864 non-null   int64
 19  Economy_status_Developing     2864 non-null   int64
 20  Life_expectancy               2864 non-null   float64
dtypes: float64(11), int64(8), object(2)
memory usage: 470.0+ KB
```

[2]:
```
raw_df.describe()
```

[2]:

	Year	Infant_deaths	Under_five_deaths	Adult_mortality
count	2864.000000	2864.000000	2864.000000	2864.000000
mean	2007.500000	30.363792	42.938268	192.251775
std	4.610577	27.538117	44.569974	114.910281
min	2000.000000	1.800000	2.300000	49.384000
25%	2003.750000	8.100000	9.675000	106.910250
50%	2007.500000	19.600000	23.100000	163.841500
75%	2011.250000	47.350000	66.000000	246.791375
max	2015.000000	138.100000	224.900000	719.360500

	Alcohol_consumption	Hepatitis_B	Measles	BMI
count	2864.000000	2864.000000	2864.000000	2864.000000
mean	4.820882	84.292598	77.344972	25.032926
std	3.981949	15.995511	18.659693	2.193905
min	0.000000	12.000000	10.000000	19.800000
25%	1.200000	78.000000	64.000000	23.200000
50%	4.020000	89.000000	83.000000	25.500000
75%	7.777500	96.000000	93.000000	26.400000
max	17.870000	99.000000	99.000000	32.100000

	Polio	Diphtheria	Incidents_HIV	GDP_per_capita
count	2864.000000	2864.000000	2864.000000	2864.000000
mean	86.499651	86.271648	0.894288	11540.924930
std	15.080365	15.534225	2.381389	16934.788931
min	8.000000	16.000000	0.010000	148.000000
25%	81.000000	81.000000	0.080000	1415.750000
50%	93.000000	93.000000	0.150000	4217.000000
75%	97.000000	97.000000	0.460000	12557.000000
max	99.000000	99.000000	21.680000	112418.000000

	Population_mln	Thinness_ten_nineteen_years	Thinness_five_nine_years
count	2864.000000	2864.000000	2864.000000
mean	36.675915	4.865852	4.899825
std	136.485867	4.438234	4.525217
min	0.080000	0.100000	0.100000
25%	2.097500	1.600000	1.600000
50%	7.850000	3.300000	3.400000
75%	23.687500	7.200000	7.300000
max	1379.860000	27.700000	28.600000

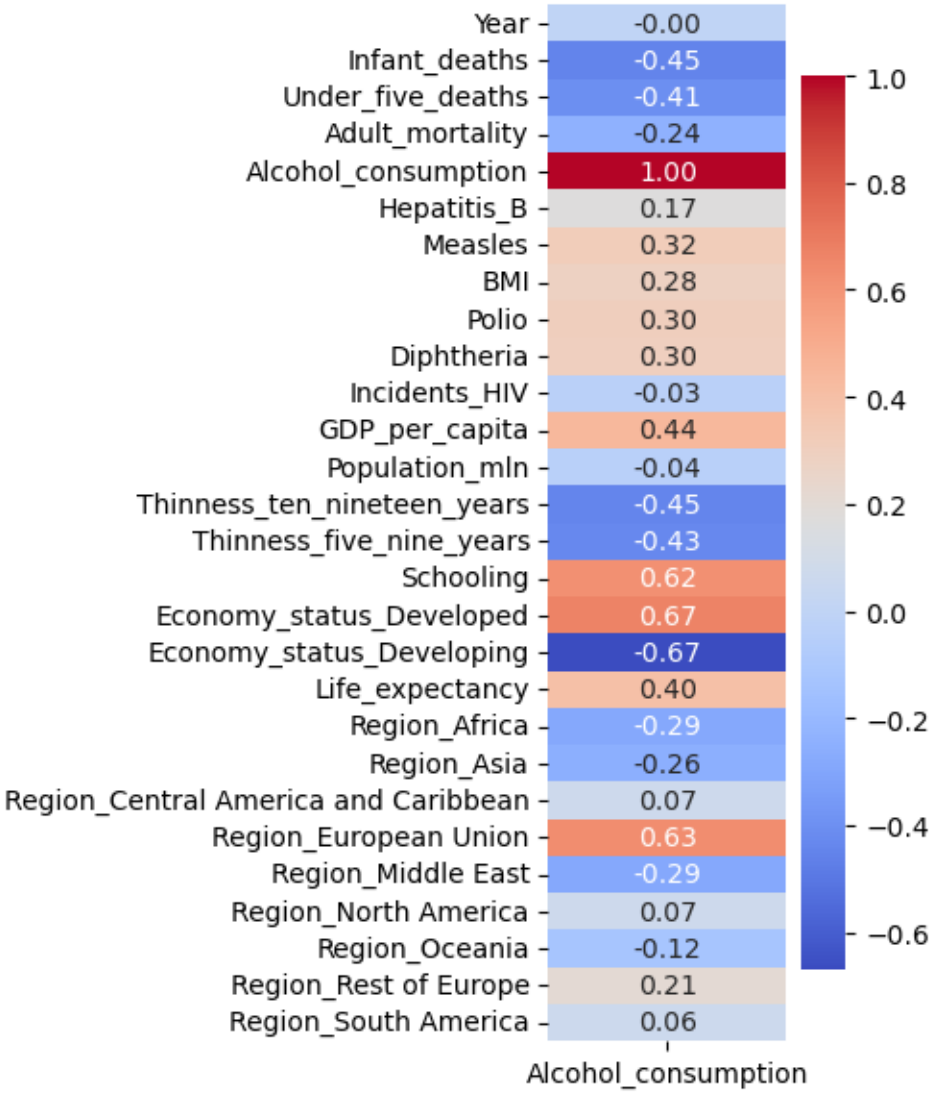
	Schooling	Economy_status_Developed	Economy_status_Developing
count	2864.000000	2864.000000	2864.000000
mean	7.632123	0.206704	0.793296
std	3.171556	0.405012	0.405012
min	1.100000	0.000000	0.000000

```
25%       5.100000                0.000000                1.000000
50%       7.800000                0.000000                1.000000
75%      10.300000                0.000000                1.000000
max      14.100000                1.000000                1.000000

       Life_expectancy
count      2864.000000
mean         68.856075
std           9.405608
min          39.400000
25%          62.700000
50%          71.400000
75%          75.400000
max          83.800000
```

[3]:
```
# One-hot encode categorical variables
data_encoded = pd.get_dummies(raw_df, columns=['Region'])

# Select region columns
region_columns = data_encoded[[col for col in data_encoded.columns if col.
 ↪startswith('Region')]]

# Check the number of datapoints in each region
true_counts = region_columns.sum()
print(true_counts)
```

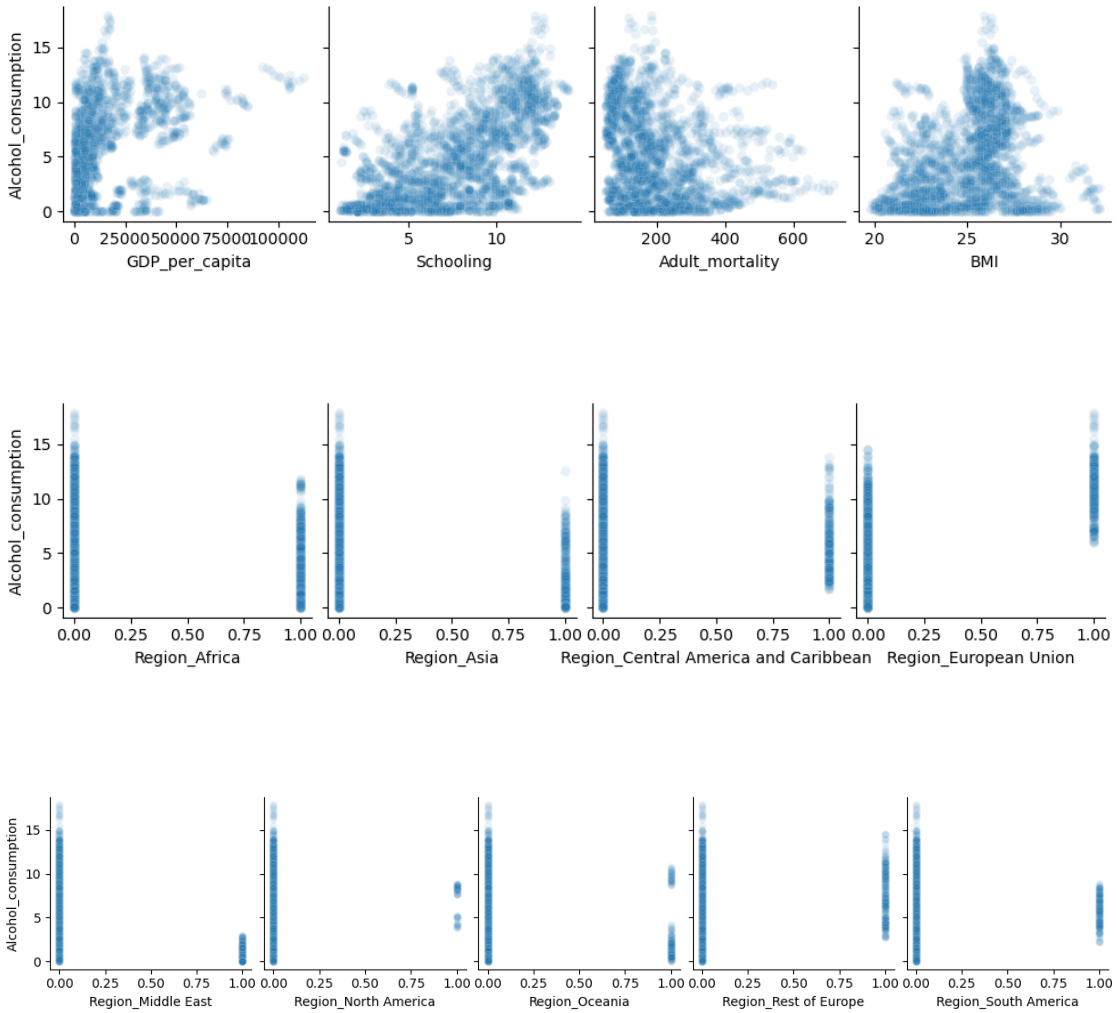
```
Region_Africa                           816
Region_Asia                             432
Region_Central America and Caribbean    304
Region_European Union                   432
Region_Middle East                      224
Region_North America                     48
Region_Oceania                          176
Region_Rest of Europe                   240
Region_South America                    192
dtype: int64
```

[4]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Display correlations with alcohol consumption
correlation_matrix = data_encoded.drop(['Country'], axis=1).corr()
plt.figure(figsize=(2,7))
sns.heatmap(correlation_matrix['Alcohol_consumption'].to_frame(), annot=True,␣
 ↪cmap='coolwarm', fmt=".2f")
plt.show()
```



[5]:
```
# Display pairwise relationships between alcohol consumption and the selected␣
↪features

sns.pairplot(data_encoded,
             y_vars=['Alcohol_consumption'],
             x_vars=['GDP_per_capita', 'Schooling', 'Adult_mortality', 'BMI'],
             plot_kws={'alpha': 0.1})

sns.pairplot(data_encoded,
             y_vars=['Alcohol_consumption'],
             x_vars=['Region_Africa', 'Region_Asia', 'Region_Central America␣
↪and Caribbean',
                     'Region_European Union'],
```

```
            plot_kws={'alpha': 0.1})

sns.pairplot(data_encoded,
             y_vars=['Alcohol_consumption'],
             x_vars=['Region_Middle East', 'Region_North America',
                     'Region_Oceania', 'Region_Rest of Europe', 'Region_South␣
 ↪America'],
             plot_kws={'alpha': 0.1})

plt.show()
```



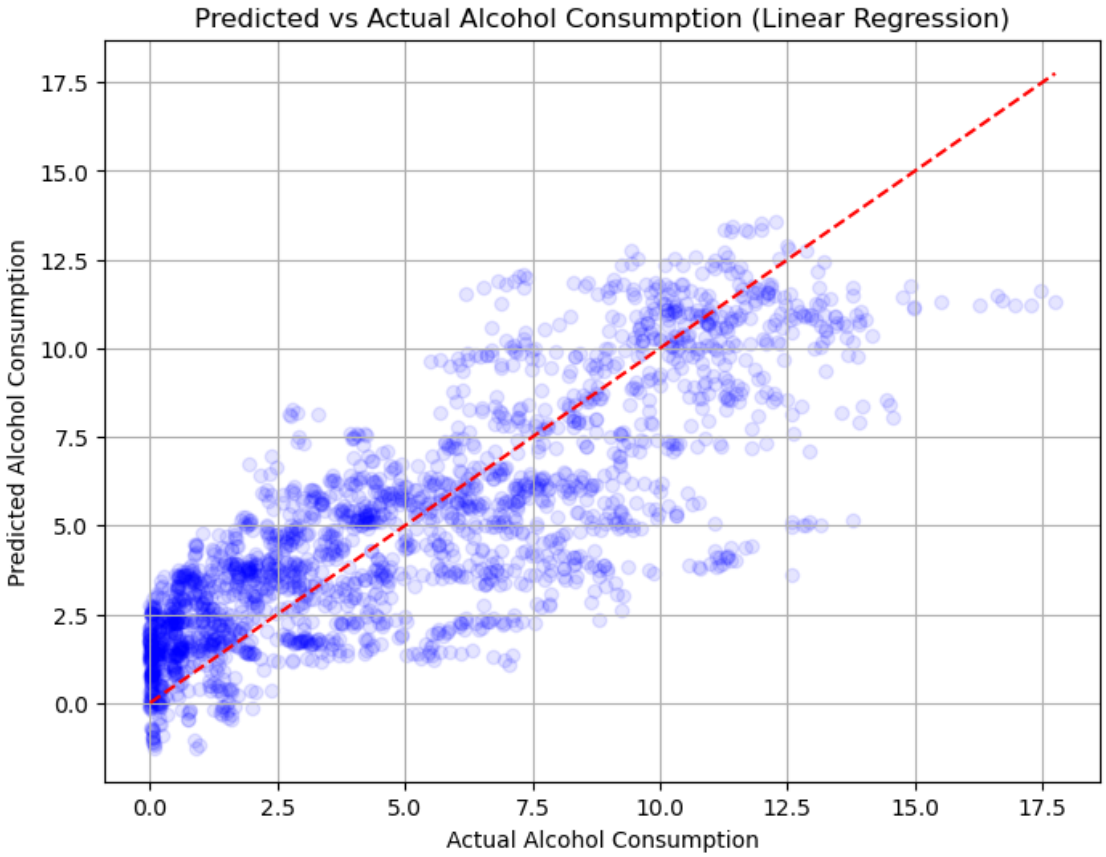
[6]:
```
# Linear Regression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

```
from sklearn.model_selection import KFold, cross_val_score, cross_val_predict,␣
 ↪train_test_split
import math

# Specify the features and target variable
features = ['GDP_per_capita', 'Schooling', 'Adult_mortality', 'BMI'] + \
           [col for col in data_encoded.columns if col.startswith('Region')]
target = 'Alcohol_consumption'

# Separate the features and target
X = data_encoded[features]
y = data_encoded[target]

# Split data into training and test sets
X, X_test_final, y, y_test_final = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=1)

# Initialize the linear regression model
lr_model = LinearRegression()

# Set up K-fold cross-validation with k=5
kf = KFold(n_splits=5, shuffle=True, random_state=1)

# Lists to store training and validation MSE for each fold
training_errors = []
validation_errors = []

# Perform k-fold cross-validation
for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    # Fit the model on the training data
    lr_model.fit(X_train, y_train)

    # Make predictions on the training set
    y_train_pred = lr_model.predict(X_train)
    training_mse = mean_squared_error(y_train, y_train_pred)
    training_errors.append(training_mse)

    # Make predictions on the validation set
    y_test_pred = lr_model.predict(X_test)
    validation_mse = mean_squared_error(y_test, y_test_pred)
    validation_errors.append(validation_mse)

# Print training and validation errors for each fold
print("Linear Regression")
```

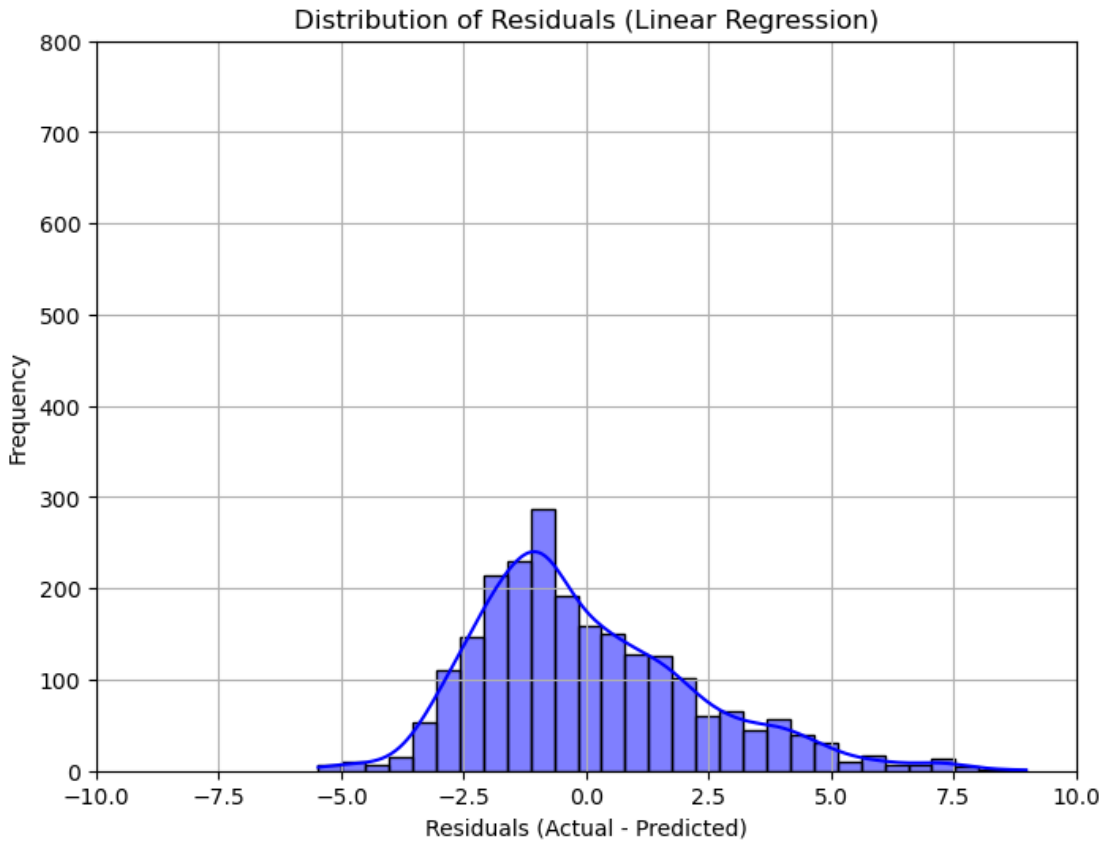
```
print("Training Mean Squared Errors for each fold:", training_errors)
print("Validation Mean Squared Errors for each fold:", validation_errors)

# Calculate and print the mean of training and validation MSE
training_mse_mean = sum(training_errors) / len(training_errors)
validation_mse_mean = sum(validation_errors) / len(validation_errors)

print(f'Mean Training MSE: {training_mse_mean}')
print(f'Mean Validation MSE: {validation_mse_mean}')

# Print RMSE for validation
print(f'Root of Validation MSE: {math.sqrt(validation_mse_mean)}')
```

```
Linear Regression
Training Mean Squared Errors for each fold: [4.9882800204310795,
4.984827128302022, 5.1579530171028996, 4.960234478862073, 4.970171486815257]
Validation Mean Squared Errors for each fold: [5.162044347700043,
5.185627799024728, 4.520550258269881, 5.284150157393688, 5.274791906838051]
Mean Training MSE: 5.012293226302666
Mean Validation MSE: 5.085432893845279
Root of Validation MSE: 2.255090440280673
```

[7]:
```
# Cross-validation predictions (using all validation folds combined)
y_pred = cross_val_predict(lr_model, X, y, cv=kf)

# Plot Actual vs. Predicted
plt.figure(figsize=(8, 6))
plt.scatter(y, y_pred, color='blue', alpha=0.1)
plt.plot([min(y), max(y)], [min(y), max(y)], color='red', linestyle='--')  #␣
 ↪Perfect prediction line
plt.title('Predicted vs Actual Alcohol Consumption (Linear Regression)')
plt.xlabel('Actual Alcohol Consumption')
plt.ylabel('Predicted Alcohol Consumption')
plt.grid(True)
plt.show()
```



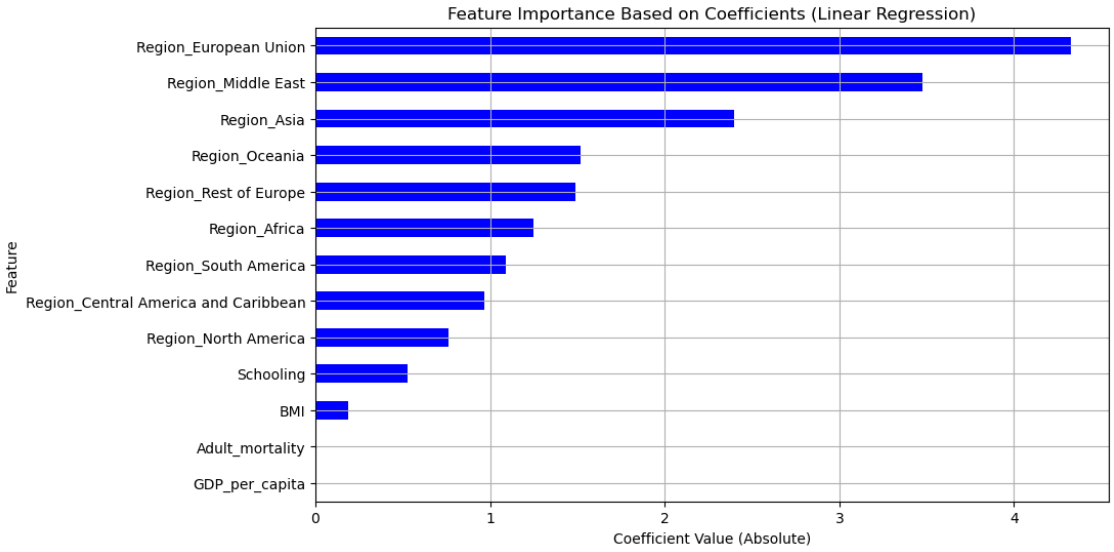
[8]:
```
# Plot the distribution of residuals
residuals = y - y_pred
plt.figure(figsize=(8, 6))
sns.histplot(residuals, kde=True, color='blue', bins=30)
plt.xlim(-10,10)
plt.ylim(0,800)
plt.title('Distribution of Residuals (Linear Regression)')
plt.xlabel('Residuals (Actual - Predicted)')
plt.ylabel('Frequency')
plt.grid(True)

plt.show()
```



[9]:
```
# Fit the model on entire training set to visualize coefficients
lr_model.fit(X, y)

# Get the coefficients from the model
coefficients = lr_model.coef_

# Pair and print coefficients with feature names
feature_importance = pd.Series(coefficients, index=features)
print(feature_importance)

# Sort the coefficients by absolute value
feature_importance_sorted = feature_importance.abs().sort_values(ascending=True)

# Plot the coefficients
plt.figure(figsize=(10, 6))
feature_importance_sorted.plot(kind='barh', color='blue')
plt.title('Feature Importance Based on Coefficients (Linear Regression)')
plt.xlabel('Coefficient Value (Absolute)')
plt.ylabel('Feature')
plt.grid(True)
```

```
plt.show()
```

```
GDP_per_capita                          0.000029
Schooling                               0.527378
Adult_mortality                         0.006747
BMI                                    -0.186956
Region_Africa                          -1.249087
Region_Asia                            -2.396466
Region_Central America and Caribbean    0.966591
Region_European Union                   4.326048
Region_Middle East                     -3.472559
Region_North America                    0.764036
Region_Oceania                         -1.519368
Region_Rest of Europe                   1.489621
Region_South America                    1.091185
dtype: float64
```



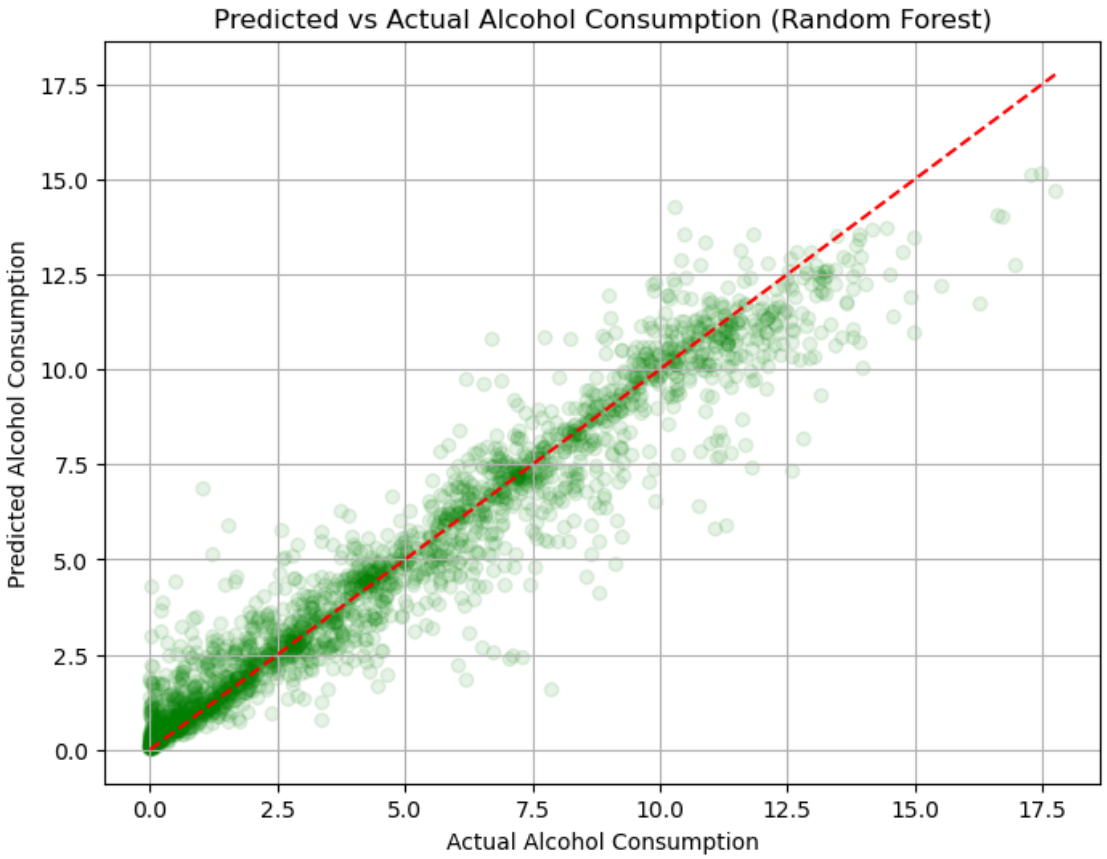
[10]:
```
# Random Forest Regression
from sklearn.ensemble import RandomForestRegressor

# Initialize the random forest regression model
rf_model = RandomForestRegressor(n_estimators=100, random_state=1)

# Set up K-fold cross-validation with k=5
kf = KFold(n_splits=5, shuffle=True, random_state=1)

# Lists to store training and validation MSE for each fold
training_errors = []
validation_errors = []
```

```
# Perform k-fold cross-validation
for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    # Fit the model on the training data
    rf_model.fit(X_train, y_train)

    # Make predictions on the training set
    y_train_pred = rf_model.predict(X_train)
    training_mse = mean_squared_error(y_train, y_train_pred)
    training_errors.append(training_mse)

    # Make predictions on the validation set
    y_test_pred = rf_model.predict(X_test)
    validation_mse = mean_squared_error(y_test, y_test_pred)
    validation_errors.append(validation_mse)

# Print training and validation errors for each fold
print("Random Forest Regression")
print("Training Mean Squared Errors for each fold:", training_errors)
print("Validation Mean Squared Errors for each fold:", validation_errors)

# Calculate and print the mean of training and validation MSE
training_mse_mean = sum(training_errors) / len(training_errors)
validation_mse_mean = sum(validation_errors) / len(validation_errors)

print(f'Mean Training MSE: {training_mse_mean}')
print(f'Mean Validation MSE: {validation_mse_mean}')

# Print RMSE for validation
print(f'Root of Validation MSE: {math.sqrt(validation_mse_mean)}')
```

```
Random Forest Regression
Training Mean Squared Errors for each fold: [0.15681878991694237,
0.15665146771134675, 0.1542750234964432, 0.1619725434513205,
0.15390520196512292]
Validation Mean Squared Errors for each fold: [1.0441104493082232,
1.0663652023201577, 0.9956083388891354, 0.9604140993239192, 1.2891023467939213]
Mean Training MSE: 0.15672460530823515
Mean Validation MSE: 1.0711200873270712
Root of Validation MSE: 1.0349493163083259
```

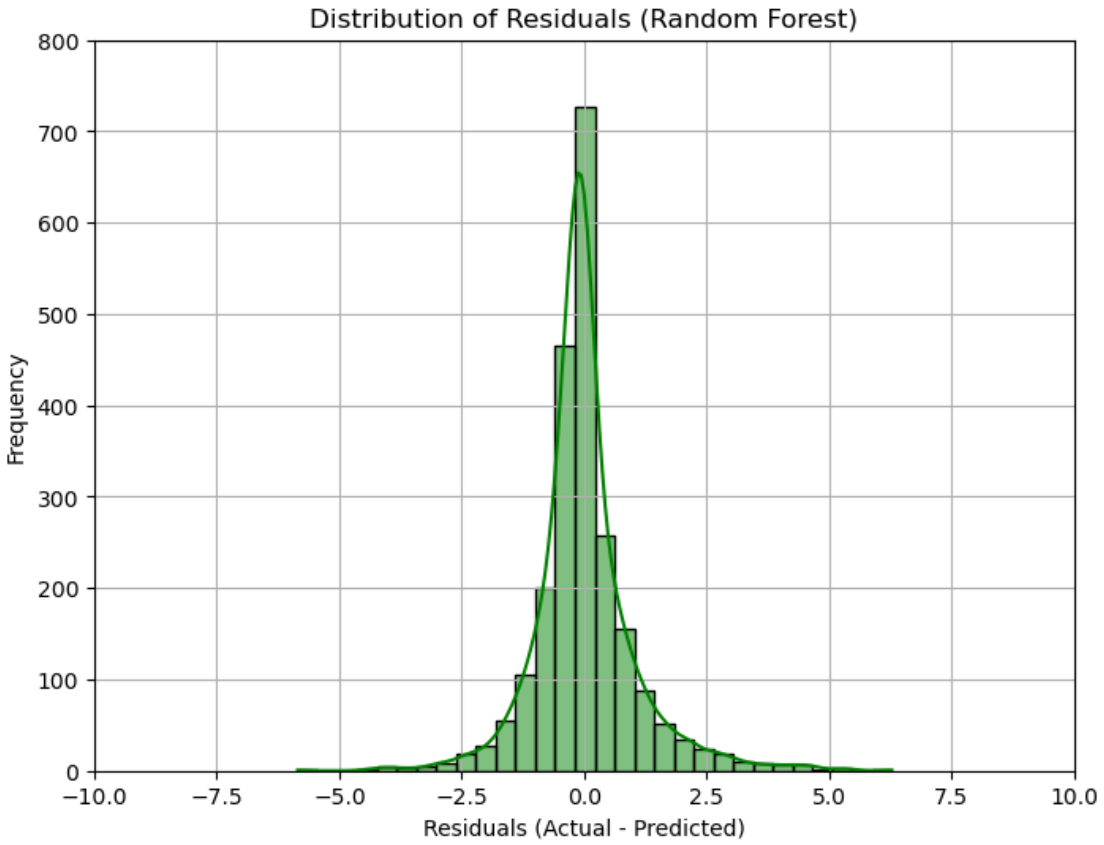
[11]:
```
# Cross-validation predictions
y_pred_rf = cross_val_predict(rf_model, X, y, cv=kf)
```

```
# Plot Actual vs. Predicted for Random Forest
plt.figure(figsize=(8, 6))
plt.scatter(y, y_pred_rf, color='green', alpha=0.1)
plt.plot([min(y), max(y)], [min(y), max(y)], color='red', linestyle='--')  #␣
 ↪Perfect prediction line
plt.title('Predicted vs Actual Alcohol Consumption (Random Forest)')
plt.xlabel('Actual Alcohol Consumption')
plt.ylabel('Predicted Alcohol Consumption')
plt.grid(True)
plt.show()
```

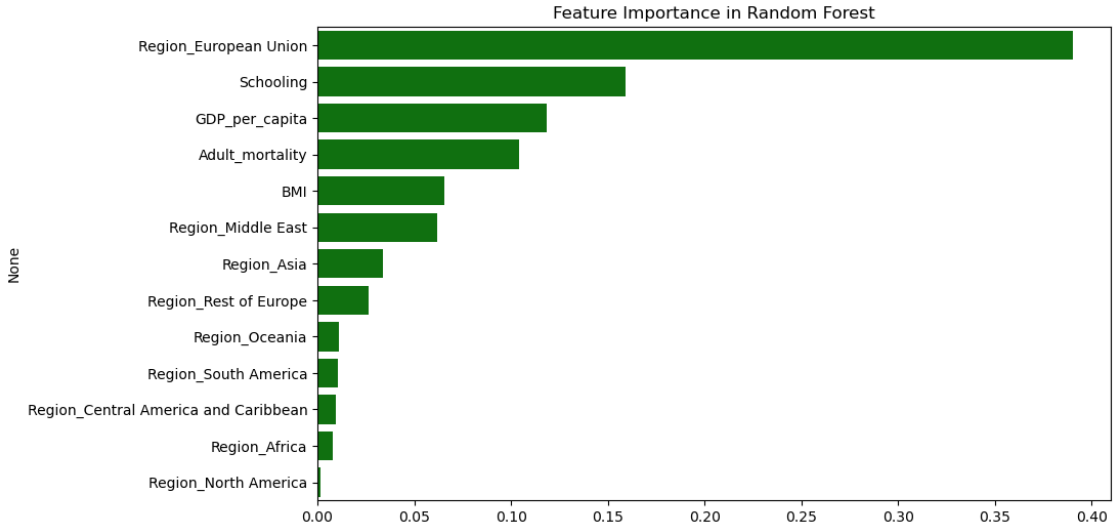


[12]:
```
# Calculate residuals across all folds
residuals = y - y_pred_rf

# Plot the distribution of residuals
plt.figure(figsize=(8, 6))
sns.histplot(residuals, kde=True, color='green', bins=30)
plt.xlim(-10,10)
plt.ylim(0,800)
plt.title('Distribution of Residuals (Random Forest)')
```

```
plt.xlabel('Residuals (Actual - Predicted)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



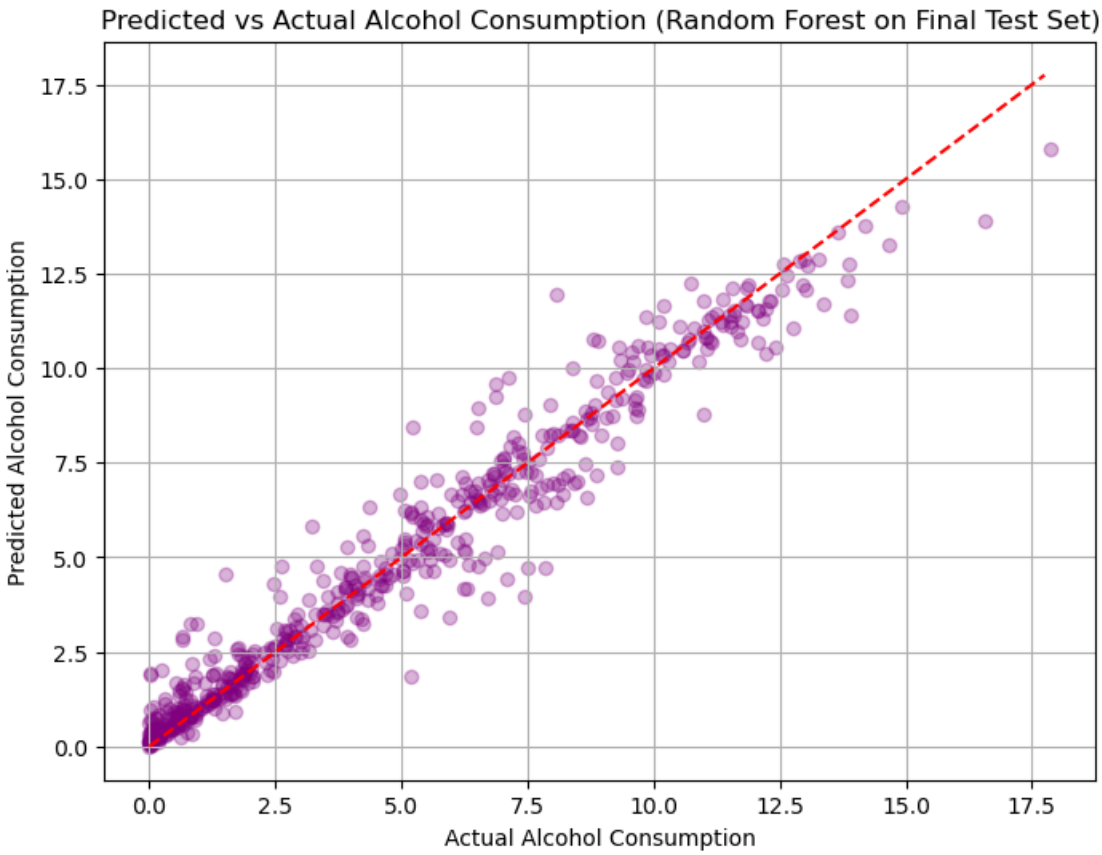
[13]:
```
# Fit the model to the entire training set
rf_model.fit(X, y)

# Get feature importance
importances = rf_model.feature_importances_
indices = np.argsort(importances)[::-1]

# Plot feature importance
plt.figure(figsize=(10, 6))
sns.barplot(x=importances[indices], y=X.columns[indices], color='green')
plt.title('Feature Importance in Random Forest')
plt.show()
```



[14]:
```
# Evaluate the Random Forest model on the test set
y_test_pred = rf_model.predict(X_test_final)
test_mse = mean_squared_error(y_test_final, y_test_pred)

# Print MSE and RMSE
print(f'Test MSE: {test_mse}')
print(f'Root of Test MSE: {math.sqrt(test_mse)}')
```

```
Test MSE: 0.7128384720580747
Root of Test MSE: 0.8442976205450745
```

[15]:
```
# Scatter plot of predicted vs actual values (random forest on final test set)
plt.figure(figsize=(8, 6))
plt.scatter(y_test_final, y_test_pred, color='purple', alpha=0.3)
plt.plot([min(y), max(y)], [min(y), max(y)], color='red', linestyle='--')  #␣
 ↪Perfect prediction line
plt.title('Predicted vs Actual Alcohol Consumption (Random Forest on Final Test␣
 ↪Set)')
plt.xlabel('Actual Alcohol Consumption')
plt.ylabel('Predicted Alcohol Consumption')
plt.grid(True)
plt.show()
```


Predicted vs Actual Alcohol Consumption (Random Forest on Final Test Set)
Predicted Alcohol Consumption
Actual Alcohol Consumption
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

[ ]: