Time Series Mining Report: Forecasting United States Alcohol Consumption Per Capita Over Time.

By: Randy Koliha, Isaiah Sarria, Owen Telis, Michael Farrelly

Introduction

In this report we will be demonstrating time series techniques to forecast United States total alcohol consumption per capita. Our goal is to predict what the next 6 years of alcohol consumption would be after 2010, Then we want to compare these predictions to actual values by utilizing error estimators to evaluate which forecasting method(s) is better to use to estimate future United States total Alcohol consumption per capita.

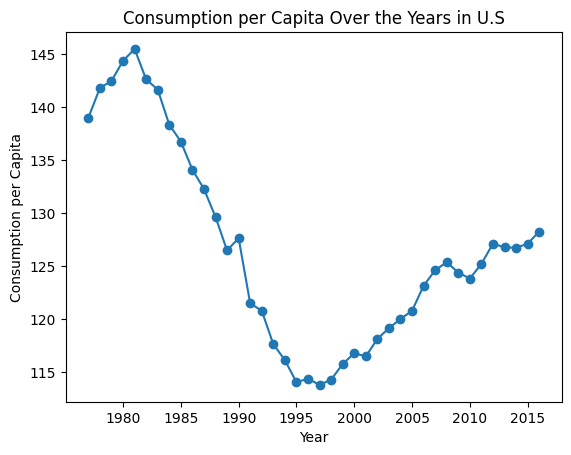
Data and Preprocessing

Our data comes from the National Institute on Alcohol Abuse and Alcoholism and was taken directly from Kaggle as a dataset. In order to prepare the data, we gathered the total alcohol consumption per capita by year for all of the United States. This mainly included merging data from all states as the data was given on a state-by-state basis. Upon completing this and other general preprocessing steps, the data was ready to be analyzed.

Methodology

As for the methods used for this project, we focused on three central forecasting techniques and calculated the corresponding errors for those forecasts. We aim to forecast the last six years of the data with each method and then calculate the corresponding errors to the actual values of the last six years that the data holds. Our first forecasting method was the exponential moving average (EMA). This technique works by weighing the difference between the current period's price and the previous EMA and adding the result to the previous EMA. The second forecasting technique utilized in this project was exponential smoothing. The exponential smoothing technique works by assigning more value on recent observations and decreasing weights to older observations. The last forecasting technique we used was linear regression, which is not inherently a forecasting technique on its own, however, the slope of the line was determined on the old data and then plotted using the same slope to continue the regression line as a forecast. Upon completion of all forecasting methods, errors for the forecasts were calculated by using error techniques like MAD, MSE, and MAPE. We applied each error technique to each forecasting technique to determine which forecast had the most accuracy. We found that forecasting technique 1, or exponential moving average, was the most accurate when error techniques were applied to it, but we found that specifically for exponential moving average, MSE provided the lowest error.

Results and Discussion



Exponential moving average forecast.

Figure 1: Consumption of Alcohol per Capita in the US from 1977-2016

Here in Figure 1 is the graph output when we took every state and aggregated them by year. The line graph shows that total alcohol consumption starts at a high peak, dips off, and then starts an incline trend.

A graph showing the growth of the average and the average of the average of the average of the average of the average of the average of the average of the average of the average of the average of

Description automatically generated

Figure 2: Lines described on chart, Green shows prediction

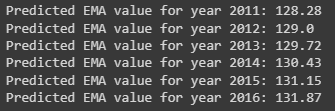


Figure 3: Prediction results for the EMA model.

The graph, Figure 2, of exponential smoothing average displays the predicted values stay close to the actual values. We decided on an alpha value that gives the graph a very close fit but not too close to prevent overfitting. This gives the predictions some wiggle room. The last six values in the predicted values output are what the model predicted.

A graph of the growth of the company

Description automatically generated with medium confidenceExponential Smoothing Forecast

Figure 4: Values for ES forecast

A screenshot of a computer screen

Description automatically generated

Figure 5: Predicted values of ES model

As seen in the graph, Figure 4, the predicted values appear to lag the actual values when looking at earlier years. Due to the nature of exponential smoothing, the technique will put more weight on the more recent values to make a proper prediction. Figure 5 shows the output of exponential smoothing forecasts for the last six years of the data.

A graph with blue dots and red line

Description automatically generatedLinear Regression Forecast

Figure 6: Linear Regression on previous US Total Consumption data



A screen shot of a computer screen

Description automatically generated

Figure 7: Prediction results from Regression Technique & regression equation

Figure 6 of the linear regression model the points form a linear line that the desired years can be plugged into. Once the years that are desired for prediction are plugged in the model outputs what the consumption should be. The corresponding linear equation representing the line shows the slope and the intercept and the years that can be plugged in. The output of 6-year predictions is displayed here as well.

Error Estimators

Figure 8: MAD, MSE, and MAPE Techniques applied to forecasting methods. Technique 1 refers to EMA, Technique 2 refers to ES and technique 3 refers to Regression prediction

A screen shot of a black screen

Description automatically generated

As seen in the error estimators’ output, Figure 8, we have MAD, MSE and MAPE to test our error for each forecast. These were gathered by implementing each estimator with the prediction against the actual values that were taken out for our testing. Here we found that exponential moving average had the lowest errors compared to the other forecasting techniques. Also, MSE had the lowest error for exponential moving average and MAPE was the lowest forecast error for the rest of the forecasts. We concluded that exponential moving average would be the best technique for forecasting total alcohol consumption by year in the United States.

Sources

<https://www.kaggle.com/datasets/linzey/alcohol-consumption-us>

Appendix

Python Code

# Import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sb

from sklearn.linear\_model import LinearRegression

# Mount Google Drive into Google Colab

from google.colab import drive

drive.mount('/content/drive')

# Import Data from Google Colab

df = pd.read\_csv("Alcohol\_Consumption\_US.csv")

# Selecting the Texas data

specific\_state = 'Texas'

filtered\_df = df[df['State'] == specific\_state]

# Plotting the line chart of Texas' bevergae consumption

plt.plot(filtered\_df['Year'], filtered\_df['All beverages (Per capita consumption)'], marker='o', linestyle='-')

# Adding labels and title

plt.xlabel('Year')

plt.ylabel('Consumption per Capita')

plt.title(f'Consumption per Capita Over the Years - {specific\_state}')

# Display the plot

plt.show()

# Selecting the Ohio data

specific\_state = 'Ohio'

filtered\_df = df[df['State'] == specific\_state]

# Plotting the line chart of Ohio's bevergae consumption

plt.plot(filtered\_df['Year'], filtered\_df['All beverages (Per capita consumption)'], marker='o', linestyle='-')

# Adding labels and title

plt.xlabel('Year')

plt.ylabel('Consumption per Capita')

plt.title(f'Consumption per Capita Over the Years - {specific\_state}')

# Display the plot

plt.show()

# Aggregate consumptions of the entire US by year

total\_consumption\_by\_year = df.groupby('Year')['All beverages (Per capita consumption)'].sum().reset\_index()

# Display the result

print(total\_consumption\_by\_year)

plt.plot(total\_consumption\_by\_year['Year'], total\_consumption\_by\_year['All beverages (Per capita consumption)'], marker='o', linestyle='-')

# Adding labels and title

plt.xlabel('Year')

plt.ylabel('Consumption per Capita')

plt.title(f'Consumption per Capita Over the Years in U.S')

# Display the plot

plt.show()

# Setting alpha for Exponential Moving Average

alpha = 0.5

# Calculate the 6-Year Exponential Moving Average

total\_consumption\_by\_year['6-Year EMA'] = total\_consumption\_by\_year['All beverages (Per capita consumption)'].ewm(alpha=alpha, adjust=False).mean()

# Exclude the last six values from both actual values and EMA

total\_consumption\_by\_year\_minus\_last\_six = total\_consumption\_by\_year.iloc[:-6]

# Make a prediction for the next 6 years based on the calculated EMA

last\_year = total\_consumption\_by\_year\_minus\_last\_six['Year'].max()

prediction\_years = range(last\_year + 1, last\_year + 7)

predicted\_values = total\_consumption\_by\_year\_minus\_last\_six['6-Year EMA'].iloc[-1] + (total\_consumption\_by\_year['6-Year EMA'].iloc[-1] - total\_consumption\_by\_year['6-Year EMA'].iloc[-2]) \* range(1, 7)

# Plot the data and prediction

plt.figure(figsize=(10, 6))

plt.plot(total\_consumption\_by\_year['Year'], total\_consumption\_by\_year['All beverages (Per capita consumption)'], label='Actual Values')

plt.plot(total\_consumption\_by\_year\_minus\_last\_six['Year'], total\_consumption\_by\_year\_minus\_last\_six['6-Year EMA'], label=f'6-Year EMA (alpha={alpha})')

plt.plot(prediction\_years, predicted\_values, label='6-Year EMA Prediction', linestyle='--')

plt.title('Exponential Moving Average and 6-Year Prediction')

plt.xlabel('Year')

plt.ylabel('Value')

plt.legend()

plt.show()

# Prints predicted values of EMA

for i, value in enumerate(predicted\_values, start=1):

    rounded\_value = round(value, 2)

    print(f"Predicted EMA value for year {i+2010}: {rounded\_value}")

# Loading and storing aggregated column

total\_consumption\_by\_year\_m = total\_consumption\_by\_year['All beverages (Per capita consumption)']

# Initialize the forecast for period 1 (just for calculation)

forecast = total\_consumption\_by\_year\_m[0]

# Smoothing constant

smoothing\_constant = 0.4

# List to store the forecasts

forecasts = []

# Calculate forecast for periods 2 to 2016 using exponential smoothing

for i in range(1, min(2017, len(total\_consumption\_by\_year\_m) + 1)):

    if i == 1:

        # Forecast for period 2 using naive approach

        forecast = total\_consumption\_by\_year\_m[0]

    else:

        # Exponential smoothing for subsequent periods

        forecast = smoothing\_constant \* total\_consumption\_by\_year\_m[i - 2] + (1 - smoothing\_constant) \* forecast

    forecasts.append(round(forecast, 2))  # Round to two decimals

# Display forecasts for periods 2 (1978) to 40 (2016)

for i, forecast in enumerate(forecasts[1:], start=2):

    print(f"Exponential Smoothing Forecast for Year {i+1976}: {forecast}")

# Defining a variable for the y-axis of years

years = range(1977, 1977 + len(total\_consumption\_by\_year\_m))

# Plotting the forecast

plt.plot(years, total\_consumption\_by\_year\_m, label='Original Data', marker='o')

plt.plot(years, forecasts, label='Forecasted Values', marker='o')

plt.title('Exponential Smoothing Forecast')

plt.xlabel('Year')

plt.ylabel('Total Consumption')

plt.legend()

plt.show()

# Removing last 6 values because that is what we will be predicting

X = total\_consumption\_by\_year.iloc[:-6]['Year'].values.reshape(-1, 1)  # Reshape to make it a 2D array

y = total\_consumption\_by\_year.iloc[:-6]['All beverages (Per capita consumption)'].values

# Create a linear regression model

model = LinearRegression()

# Train the model

model.fit(X, y)

# Make predictions

y\_pred = model.predict(X)

# Plot the data and the linear regression line

plt.scatter(X, y, label='Actual Data')

plt.plot(X, y\_pred, 'r-', label='Linear Regression')

plt.xlabel('Years')

plt.ylabel('Consumption')

plt.title('Linear Regression: Years vs US Total Consumption Per Capita')

plt.legend()

plt.show()

# Generate the next five years

future\_years = np.arange(2011, 2017).reshape(-1, 1)

# Use the trained model to make predictions for the future years

future\_consumption = model.predict(future\_years)

# Display the predicted consumption for the next five years

for year, consumption in zip(future\_years.flatten(), future\_consumption):

    print(f"Year: {year}, Predicted Consumption: {consumption}")

# Get the slope (coefficient) and intercept of the linear regression model

slope = model.coef\_[0]

intercept = model.intercept\_

# Display the regression equation

print(f"Regression Equation: y = {slope:.3f} \* Years + {intercept:.3f}")

# Demand and forecasted values for both techniques

actual = total\_consumption\_by\_year['All beverages (Per capita consumption)'][-6:]

technique1\_forecast = predicted\_values

technique2\_forecast = forecasts[-6:]

technique3\_forecast = future\_consumption

# Calculate absolute errors for both techniques

error\_technique1 = np.abs(actual - technique1\_forecast)

error\_technique2 = np.abs(actual - technique2\_forecast)

error\_technique3 = np.abs(actual - technique3\_forecast)

# Calculate Mean Absolute Deviation (MAD) and round to two decimal places

mad\_technique1 = np.round(np.mean(error\_technique1), 2)

mad\_technique2 = np.round(np.mean(error\_technique2), 2)

mad\_technique3 = np.round(np.mean(error\_technique3), 2)

print("MAD for Technique 1:", mad\_technique1)

print("MAD for Technique 2:", mad\_technique2)

print("MAD for Technique 3:", mad\_technique3)

# Calculate MSE for each forecast by squaring MAD

mse1 = (mad\_technique1\*\*2)

mse2 = (mad\_technique2\*\*2)

mse3 = (mad\_technique3\*\*2)

print("MSE for Technique 1:", mse1)

print("MSE for Technique 2:", mse2)

print("MSE for Technique 3:", mse3)

# Calculate MAPE by dividing the MAD by the sum of the actual values divided by the number of actual values and then multiplying the quotient by 100

mape1 = (mad\_technique1 / (sum(actual)/len(actual)))\*100

mape2 = (mad\_technique2 / (sum(actual)/len(actual)))\*100

mape3 = (mad\_technique3 / (sum(actual)/len(actual)))\*100

print("MAPE for Technique 1:", mape1)

print("MAPE for Technique 2:", mape2)

print("MAPE for Technique 3:", mape3)

print("forecast: Exponential Moving Average")

print("")

print("MAD for Technique 1:", round(mad\_technique1, 2))

print("MSE for Technique 1:", round(mse1, 2))

print("MAPE for Technique 1:", round(mape1, 2))

print("")

print("forecast: Exponential Smoothing")

print("")

print("MAD for Technique 2:", round(mad\_technique2, 2))

print("MSE for Technique 2:", round(mse2, 2))

print("MAPE for Technique 2:", round(mape2, 2))

print("")

print("forecast: Regression")

print("")

print("MAD for Technique 3:", round(mad\_technique3, 2))

print("MSE for Technique 3:", round(mse3, 2))

print("MAPE for Technique 3:", round(mape3, 2))