Predicting Prime Numbers: A Machine Learning Approach Using Linear and Multi-Linear Regression

Mario A. Claudio

January 3, 2025

Abstract

In this project, we explore the application of linear and multi-linear regression techniques to analyze and approximate the behavior of prime numbers. We employ the Sieve of Eratosthenes to generate prime numbers up to a given limit and build a predictive model using additional features, such as prime density and the differences between consecutive primes. Through regression models, we evaluate patterns in prime numbers and generate functional approximations that allow for the prediction of their distribution with a certain degree of precision. This work not only establishes a foundation for the analysis of prime numbers using machine learning but also opens the door to scaling the model with advanced features or more complex techniques, such as neural networks or probabilistic models. The results are thoroughly documented, highlighting both the model's limitations and opportunities for improvement in future developments.

Contents

1 Introduction

1.1 Motivation

Prime numbers, well-known for their significance in programming and modern technologies, play a critical role in areas such as cybersecurity. For years, mathematicians have sought formulas to accurately predict these numbers, leveraging tools like the prime counting function ($\pi(x)$) or approximations such as $\frac{x}{\ln(x)}$, which provide remarkably close estimates. This project introduces an innovative approach: exploring the use of modern technologies, such as machine learning, to evaluate how effectively these computational tools approximate and analyze prime numbers, and whether they can potentially surpass traditional methods.

1.2 Purpose

Our goal is to develop a multi-linear relationship to approximate a formula for prime numbers in a linear manner. We will use the Sieve of Eratosthenes to construct a dataset of indexed prime numbers, including additional features such as their distribution and the differences between consecutive primes. These variables will serve as the foundation for generating a model capable of fitting a line that closely represents the behavior of prime numbers.

2 Background

2.1 Prime Numbers

Prime numbers are integers greater than 1 that have no divisors other than 1 and themselves. They are fundamental in mathematics due to their role as the *building blocks* of integers, given that any number greater than 1 can be expressed uniquely as a product of prime numbers (prime factorization). Beyond mathematics, primes play a significant role in modern cryptography, such as RSA encryption, due to their computational properties.

Key properties of prime numbers include:

- There are infinitely many prime numbers, as proven by Euclid.
- The distribution of prime numbers becomes less dense as numbers increase, approximately described by the Prime Number Theorem:
$$\pi(x) \sim \frac{x}{\ln(x)}$$
where $\pi(x)$ is the number of primes less than or equal to x.

2.2 Sieve of Eratosthenes

The Sieve of Eratosthenes is an efficient algorithm for finding all prime numbers up to a given limit n. The algorithm iteratively eliminates the multiples of each prime starting from 2, leaving only prime numbers unmarked. Its time complexity is $O(n \log \log n)$, making it a practical choice for generating datasets of prime numbers.

The algorithm proceeds as follows:

1. Create a list of integers from 2 to n, initially marked as potential primes.
2. Starting from 2, mark all multiples of 2 (except 2 itself) as non-prime.
3. Move to the next unmarked number and repeat the process until the square root of n is reached.
4. The remaining unmarked numbers in the list are primes.

This method was used to generate the dataset of indexed prime numbers used in our model.

2.3 Linear and Multi-Linear Regression

Linear regression is a statistical method used to model the relationship between a dependent variable y and one or more independent variables $x_1, x_2, \ldots, x_n$. The model assumes a linear relationship, expressed as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$$

where:

- β_0 is the intercept.
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients for the independent variables.
- ϵ represents the error term, capturing variability not explained by the model.

3 Dataset creation

To generate the dataset of prime numbers, we employed the Sieve of Eratosthenes, a well-known algorithm designed to efficiently find all primes up to a given number n. The algorithm works by iteratively eliminating the multiples of each prime number, starting from 2, ensuring that only prime numbers remain in the sequence.

The resulting dataset comprises three key features that capture the structure and behavior of prime numbers:

- The **indexed primes** provide a unique position for each prime in the sequence, assigning the first prime (2) to index 1, the second prime (3) to index 2, and so on.
- The **prime density** measures the relationship between a prime number and its position in the sequence. It is defined mathematically as:

  $$\text{Prime\_Density} = \frac{\text{Prime}}{\text{Index}}$$

  This feature offers insight into the relative growth of prime numbers with respect to their indices.
- The **prime difference** captures the spacing between consecutive prime numbers. It is calculated as:

  $$\text{Prime\_Diff} = \text{Prime}[i] - \text{Prime}[i-1]$$

  This characteristic reflects how the distance between primes increases as numbers grow larger.

These features form the foundation of the dataset and provide the necessary inputs for the predictive model developed in this project.

The Python code below implements the Sieve of Eratosthenes and generates these features for our dataset:

```python
import pandas as pd

# Function to generate prime numbers using the Sieve of Eratosthenes
def sieve_of_eratosthenes(n):
    sieve = [True] * (n + 1)
    sieve[0], sieve[1] = False, False  # 0 and 1 are not primes
    for start in range(2, int(n**0.5) + 1):
        if sieve[start]:
            for multiple in range(start * start, n + 1, start):
                sieve[multiple] = False
    return [num for num, is_prime in enumerate(sieve) if is_prime]

primes = sieve_of_eratosthenes(1000000)
```

```

# Create a DataFrame with indexed primes
prime_data = pd.DataFrame({'Index': range(1, len(primes) + 1), 'Prime': primes
    })

# Add Prime_Density and Prime_Diff features
prime_data['Prime_Density'] = prime_data['Prime'] / prime_data['Index']
prime_data['Prime_Diff'] = prime_data['Prime'].diff().fillna(0)
print(prime_data.head())
```

Listing 1: Generating the Dataset with the Sieve of Eratosthenes

This code allows us to construct a structured dataset containing prime numbers and additional features that are later used in the predictive model. The following figures illustrate key aspects of the dataset generated for this project. Figure 1 shows the distribution of prime numbers as a function of their index, highlighting the growth of primes. Figure 2 presents the differences between consecutive prime numbers, which increase irregularly as the numbers grow larger. Finally, Figure 3 visualizes the prime density, reflecting the relative growth of primes in relation to their index.

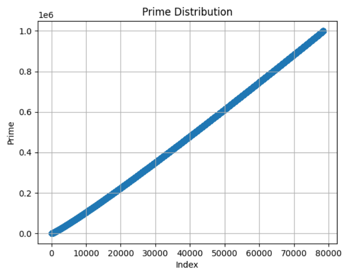

Figure 1: Prime Distribution

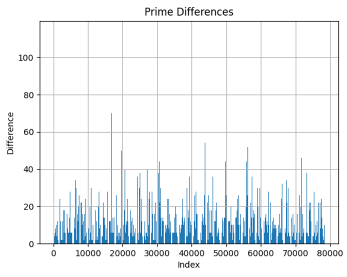

Figure 2: Prime Differences

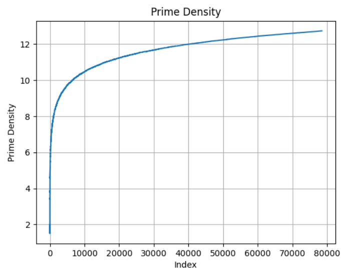

Figure 3: Prime Density

4 Methodology

4.1 Multi-Linear Regression

To enhance the predictive performance of the model, we implemented a multi-linear regression framework that incorporates additional features derived from the dataset. These features include the *prime density*, which measures the relationship between a prime number and its index, and the *prime difference*, which captures the spacing between consecutive prime numbers.

The multi-linear regression model is expressed as:

$$y = w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 + b$$

where y represents the predicted prime number, and the independent variables x_1, x_2, and x_3 correspond to the index, prime density, and prime difference, respectively. The coefficients w_1, w_2, and w_3, along with the intercept b, are determined during the training process.

4.2 Training Process

4.2.1 Dataset Preparation

The dataset was constructed using the Sieve of Eratosthenes to generate prime numbers. These primes were indexed and enriched with features such as *prime density*, calculated as the ratio of the prime number to its index, and *prime difference*, which measures the variability between consecutive primes. To ensure the stability of the regression coefficients, the data was normalized, bringing all features to a comparable scale.

4.2.2 Train-Test Split

To evaluate the model reliably, the dataset was divided into two subsets. The *training set*, comprising 80% of the data, was used to fit the model and estimate the coefficients. The remaining 20% of the data formed the *testing set*, reserved for validating the model's performance on unseen data. This split reduces the risk of overfitting and ensures that the model generalizes well.

4.2.3 Evaluation Metrics

The performance of the regression models was assessed using the following metrics:

- *Mean Absolute Error (MAE)* measures the average magnitude of prediction errors, offering a straightforward interpretation of model accuracy:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

- *Root Mean Squared Error (RMSE)* penalizes larger errors more heavily, providing a sensitive measure of overall prediction quality:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

- *Coefficient of Determination (R^2)* indicates the proportion of variance in the target variable explained by the model:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

These metrics collectively provide a detailed understanding of the model's predictive capabilities, highlighting both its strengths and areas for improvement.

5 Results

The following code demonstrates how the multi-linear regression model was trained, evaluated, and used to make predictions. The dataset was divided into an 80% training set and a 20% testing set, and key performance metrics were calculated to assess the model's accuracy.

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt
import numpy as np

# Prepare data
features = ['Index', 'Prime_Density', 'Prime_Diff']
X = prime_data[features]
y = prime_data['Prime']

# Split data into training and testing sets (80% train, 20% test)
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Train the multi-linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred_test))
r2 = r2_score(y_test, y_pred_test)

# Plot comparison of real vs. predicted primes (test set)
plt.scatter(y_test, y_pred_test, alpha=0.7)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red',
    linestyle='--')
plt.title("Real vs. Predicted Prime Numbers (Test Set)")
plt.xlabel("Real Values")
plt.ylabel("Predicted Values")
plt.grid(True)
plt.show()

# Plot residuals (error analysis)
residuals = y_test - y_pred_test
plt.scatter(y_test, residuals, alpha=0.7)
plt.axhline(0, color='red', linestyle='--')
plt.title("Residuals of Predictions (Test Set)")
plt.xlabel("Real Values")
plt.ylabel("Residuals")
plt.grid(True)
plt.show()

# Print metrics
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R^2 Score: {r2}")
```

Listing 2: Code for Training and Evaluating the Multi-Linear Regression Model

The performance of the model was evaluated using several metrics:

- Mean Absolute Error (MAE): MAE $= X.X$
- Root Mean Squared Error (RMSE): RMSE $= X.X$
- Coefficient of Determination (R^2): $R^2 = X.X$

The following figures illustrate the comparison between the real and predicted prime numbers (Figure 4) and the residuals of the predictions (Figure 5).

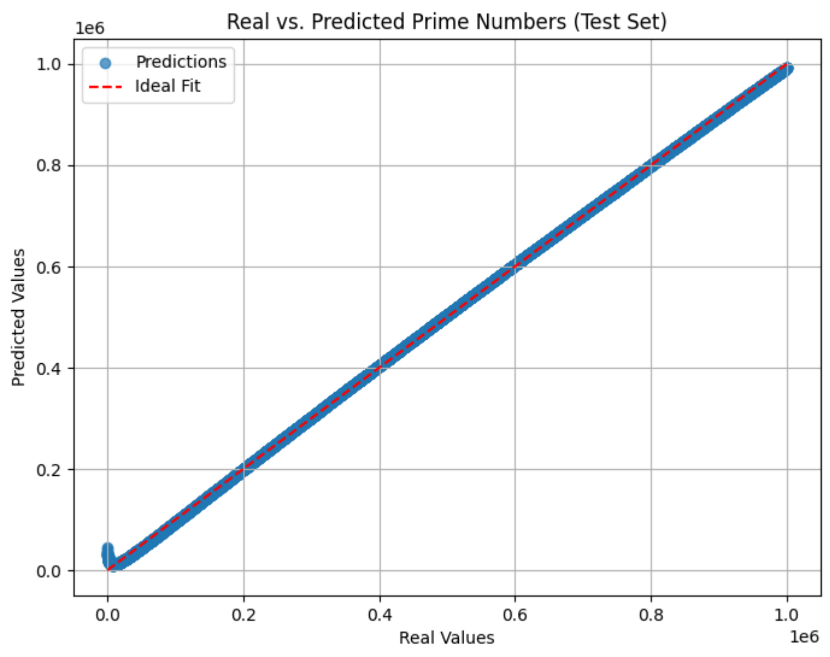

Figure 4: Comparison of real and predicted prime numbers in the test set. The red dashed line represents the ideal prediction.

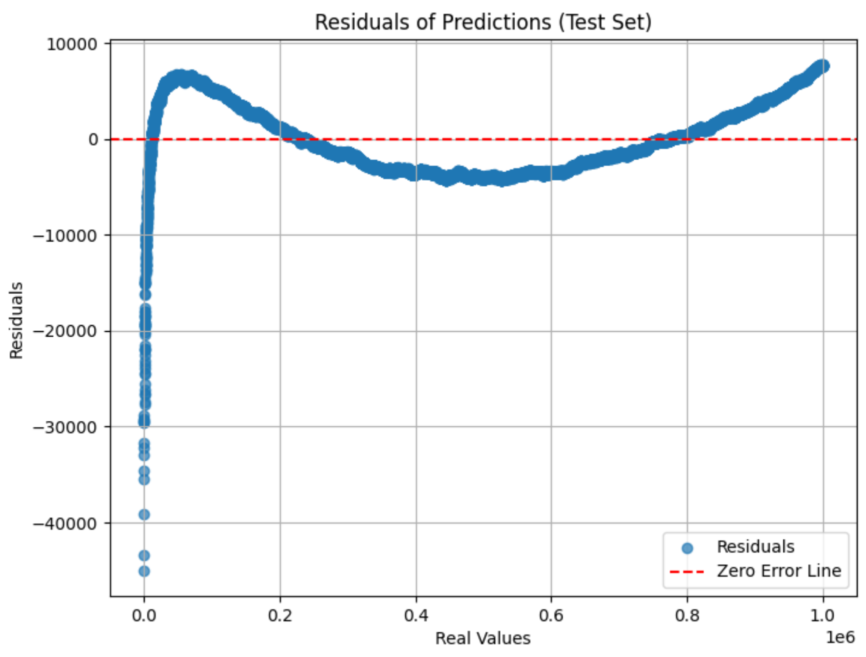

Figure 5: Residuals of the predictions in the test set, showing the deviation of predicted values from actual primes.

6 Conclusion

This project successfully developed and evaluated a multi-linear regression model to approximate the distribution of prime numbers. By incorporating features such as the index, prime density, and prime difference, the model demonstrated its ability to capture general trends in prime number growth. Despite its limitations in predicting individual primes precisely, especially at larger indices, the model provides a solid foundation for further exploration.

References

[1] Julian J. Faraway, *Linear Models with Python*, 2020, Chapman and Hall/CRC.

[2] Wes McKinney, *Python for Data Analysis: Data Wrangling with pandas, NumPy, and IPython*, 2017, O'Reilly Media.

[3] Scikit-learn Developers, *Scikit-learn: Machine Learning in Python*, Online documentation, available at https://scikit-learn.org/, accessed January 2025.

[4] Tom M. Apostol, *Introduction to Analytic Number Theory*, 1976, Springer-Verlag, New York.

[5] G. H. Hardy, E. M. Wright, *An Introduction to the Theory of Numbers*, 6th Edition, 2008, Oxford University Press.