Power Spectrum Generation and Analysis Using Machine Learning Methods

Introduction

The 21-cm line of neutral hydrogen, shifted to longer wavelengths (redshifted) due to the expansion of the universe, is a crucial tool for investigating the different phases of our universe's evolution. By observing this 21-cm line, we can create maps of the early universe at various cosmic times, providing deep insights into the structures created by the first sources of light and the origin and evolution of these sources.

To explore the history of the intergalactic medium, from the formation of the first stars and galaxies (Cosmic Dawn) to the complete ionization of the universe (Epoch of Reionization), the 21-cm line is invaluable. Large interferometric arrays are currently the best means to detect the redshifted 21-cm line, allowing us to study the distribution of neutral hydrogen across different redshifts.

However, detecting this faint 21-cm signal is extremely challenging, as it is overshadowed by much brighter foregrounds from galactic and extragalactic sources. Additionally, instrumental variations and interference from Earth's ionosphere make ground-based observations even more difficult. Various methods have been developed to mitigate foreground contamination, but they often require assumptions about the foreground model.

In recent years, machine learning techniques have found applications in cosmology and astrophysics. Artificial Neural Networks (ANNs) have been used to emulate and predict various aspects of the 21-cm signal, including the power spectrum and global signal. These ANN-based frameworks offer a promising alternative to extract astrophysical information from the observed 21-cm power spectrum directly, without relying on complex foreground models.

In this study, we present a different machine learning techniques to extract the 21-cm power spectrum and reionization parameters from simulated datasets.

Methodology

Generating Power Spectrum of Cores with Adjusted Astrophysical Parameters

In this step, we utilize the py21cmmc library to generate a dataset consisting of power spectra. The power spectra are calculated by adjusting three key astrophysical parameters: 'HII_EFF_FACTOR', 'ION_Tvir_MIN', and 'R_BUBBLE_MAX'. These parameters play a crucial role in shaping the characteristics of the power spectrum.

To generate the dataset, we follow a series of steps outlined below:

1. Importing Required Libraries: We import the necessary libraries, including numpy, py21cmmc, csv, argparse, and random, to facilitate the generation of power spectra.

2. Parsing Command-Line Arguments: We utilize the argparse library to parse command-line arguments. The arguments include the number of generations (no_of_sim) and the file number (f_no) to be used in the file naming convention.

3. Setting Up the Parameters: We initialize variables for the number of simulations (n) and the file number (x) based on the command-line arguments. These variables control the number of iterations in the generation process.

4. Iterative Generation of Power Spectra: We iterate n times, generating power spectra for each iteration. Within each iteration, we randomly select values for the astrophysical parameters 'HII_EFF_FACTOR', 'ION_Tvir_MIN', and 'R_BUBBLE_MAX' within specified ranges. These ranges ensure a diverse set of parameter values.
The astrophysical parameter ranges are mentioned in the below link:
https://21cmmc.readthedocs.io/en/latest/faqs/ParameterRanges.html?highlight=USE_MASS_DEPENDENT_ZETA

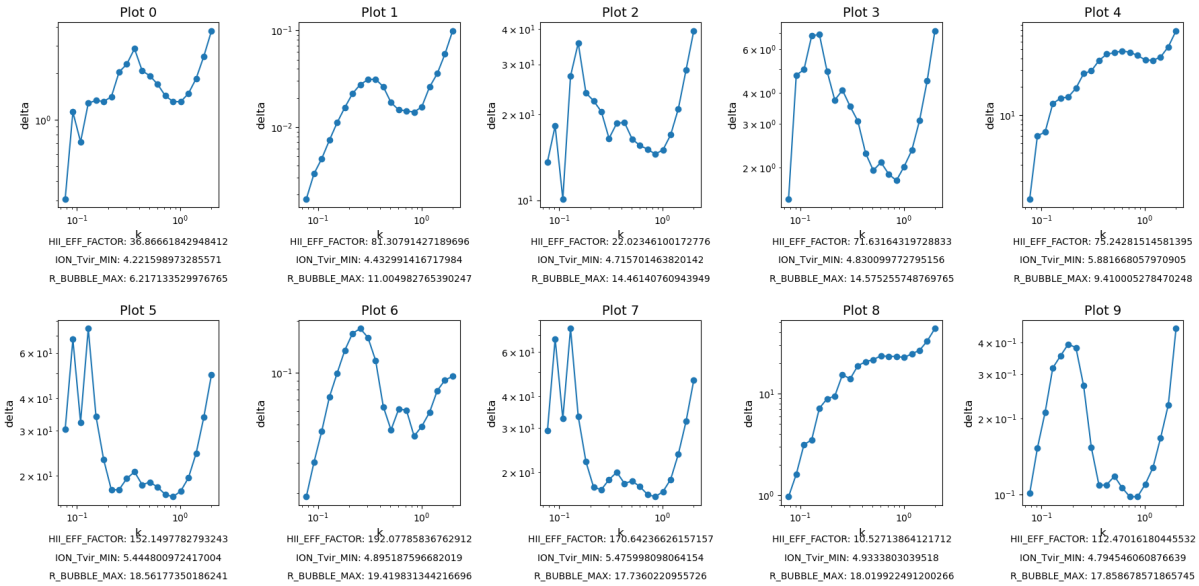
5. CoreCoevalModule: We create a core using the CoreCoevalModule from py21cmmc. This core represents a coeval box at a specific redshift and contains user-defined parameters such as HII_DIM (dimension of the ionized region) and BOX_LEN (physical size of the box). The astrophysical parameters are also set within the core.

6. Generating Data: We generate the power spectrum data by calling the Likelihood1DPowerCoeval function from py21cmmc. This function takes the datafile, noisefile, minimum k-value (min_k), maximum k-value (max_k), and a simulate flag as input parameters. The datafile is used to store the generated power spectrum data.

7. Saving Information: We save the randomly selected astrophysical parameter values for each iteration in a CSV file for future reference and analysis.

8. Cleaning the Cache: To ensure the independence of each iteration, we clear the cache using cache_tools.clear_cache().

By following these steps, we successfully generate a dataset of power spectra, each associated with specific values of 'HII_EFF_FACTOR', 'ION_Tvir_MIN', and 'R_BUBBLE_MAX'. This dataset serves as the foundation for training a machine learning model to predict these astrophysical parameters based on the power spectrum data.



The plots illustrate the relationship between the power spectrum (delta) and spatial scale (k), showcasing how the astrophysical parameters (HII_EFF_FACTOR, ION_Tvir_MIN, R_BUBBLE_MAX) shape the power spectrum characteristics.

Cleansing the Data

After generating the power spectrum data, it is necessary to cleanse and preprocess the data before using it for training a machine learning model. In this step, we perform the following data cleansing tasks:

1. Extracting k and delta Arrays: The power spectrum data is stored in the form of arrays containing the values of k and delta. The array of k values represents the spatial scales at which the power spectrum is measured, while the array of delta values represents the corresponding power spectrum values. Each power spectrum has 22 delta values associated with it.

2. Handling NaN Values: In the delta array, the first two values are NaN for every power spectrum. These NaN values do not provide meaningful information and can interfere with the training process. Hence, we remove these first two values from the delta array for each power spectrum.

By performing these cleansing steps, we ensure that the power spectrum data is in a suitable format for further analysis and training of a machine learning model.

Applying Machine Learning for Parameter Prediction

To predict the astrophysical parameters (HII_EFF_FACTOR, ION_Tvir_MIN, R_BUBBLE_MAX) based on the power spectrum data, a machine learning approach is employed using regression

techniques. During the training phase, regression models are trained using a dataset consisting of power spectra and their corresponding astrophysical parameter values. The power spectrum data, specifically the delta values, serve as input features, while the astrophysical parameters act as target variables.

Regression models learn to identify patterns and relationships between the input power spectrum data and the target variables. By capturing these patterns, the models become capable of making predictions on unseen power spectra, estimating the corresponding astrophysical parameters.

The goal of training the regression models is to create an accurate and reliable predictive model. This model can then be used to predict the astrophysical parameters for new, unseen power spectra, providing insights into the underlying physical processes shaping the observed power spectra in the early universe.

Approach 1: Regression Method using Dense and CNN Layers

In this approach, we employ regression techniques to predict the values of the astrophysical parameters using the power spectrum data. Two types of layers are used: Dense Layers and CNN Layers.

1. Dense Layers

- Data Preprocessing: The power spectrum data and corresponding astrophysical parameters are loaded and preprocessed. The features (input) are extracted from the power spectrum data, excluding the first two elements in each row. The labels (output) are the values of the astrophysical parameters.

- Data Splitting: The dataset is split into training and testing sets using the train_test_split function, with a test size of 20%.

- Neural Network Architecture: A Sequential model is defined with dense layers. The model consists of multiple dense layers with ReLU activation, with the number of neurons gradually decreasing towards the output layer.

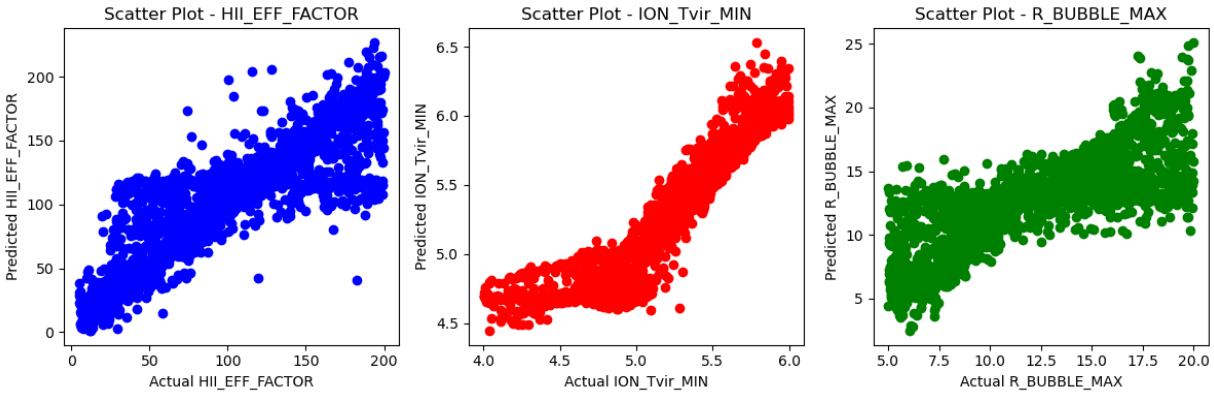
```
model = keras.Sequential([
    keras.layers.Dense(64, activation='relu', input_shape=(20,)),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(3)  # 3 output variables
])
```

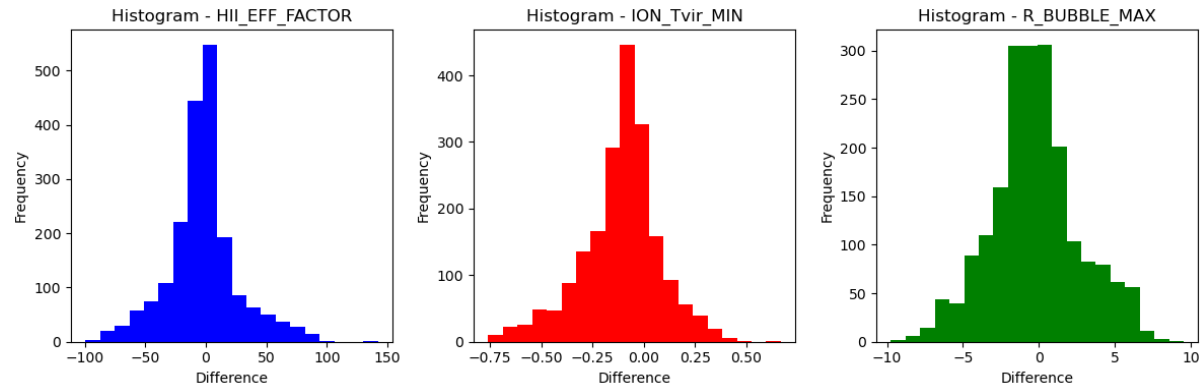
- Model Training: The model is compiled using the Adam optimizer and mean squared error loss function. Early stopping is implemented to prevent overfitting. The model is then trained on the training data for a specified number of epochs, with a batch size of 32.

- Model Evaluation: The trained model is evaluated on the testing data. Predictions are made on the testing data, and metrics such as mean absolute error (MAE) and R-squared score (R^2) are calculated.

Results

Test Loss: 209.64663696289062
Mean Absolute Error (MAE): 6.314094154300093
R^2 Score: 0.7877725154733147





Variable	RMSE	MAE	MSE	R2- Score
HII_EFF_FACTOR	29.1501	20.2648	849.7280	0.7387
ION_Tvir_MIN	0.2224	0.1667	0.0495	0.8084
R_BUBBLE_MAX	2.9252	2.2491	8.5567	0.5506

2. CNN Layers

- Data Preprocessing: Similar to the Dense Layers approach, the power spectrum data and astrophysical parameters are preprocessed.

- Data Splitting: The dataset is split into training and testing sets.

- Reshaping Input Data: The input data is reshaped to match the expected shape for a 1D CNN. An additional dimension is added to the input data using np.expand_dims.

- Neural Network Architecture (1D CNN): A Sequential model is defined with 1D convolutional layers, max pooling layers, and dense layers. The model captures spatial patterns in the power spectrum data using convolutional layers and reduces the dimensionality using max pooling layers.

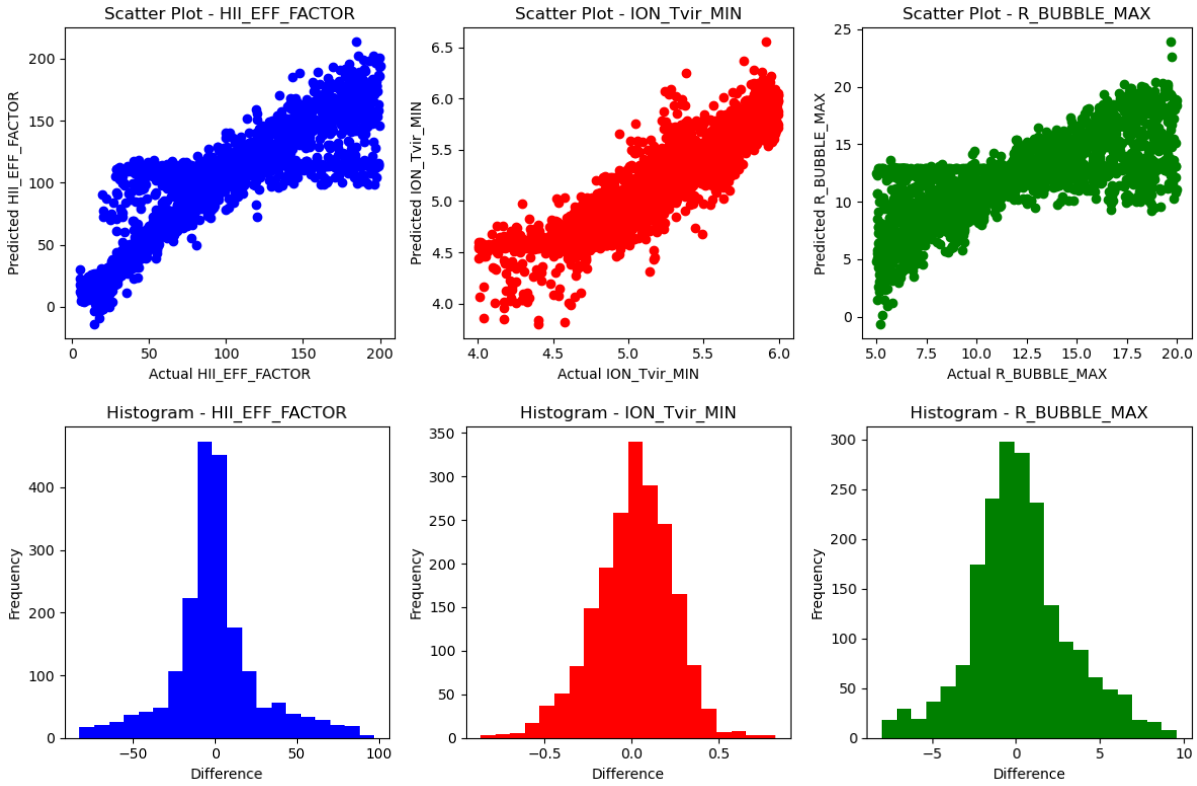
```
# Define the neural network architecture (1D CNN)
model = keras.Sequential([
    keras.layers.Conv1D(64, 3, activation='relu', input_shape=(20, 1)),
    keras.layers.MaxPooling1D(2),
    keras.layers.Conv1D(32, 3, activation='relu'),
    keras.layers.MaxPooling1D(2),
    keras.layers.Flatten(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(3)  # 3 output variables
])
```

- Model Training and Evaluation: Similar to the Dense Layers approach, the model is trained and evaluated.

The models are saved for future use, and the performance of each model is assessed using metrics such as test loss, MAE, and R^2 score. These metrics provide insights into the model's ability to accurately predict the astrophysical parameters based on the power spectrum data.

Results

Test Loss: 245.82383728027344
Mean Absolute Error (MAE): 6.864696938825312
R^2 Score: 0.7053104324806306



Variable	RMSE	MAE	MSE	R2- Score
HII_EFF_FACTOR	26.9926	18.1964	728.5987	0.7760
ION_Tvir_MIN	0.2253	0.1768	0.0508	0.8033
R_BUBBLE_MAX	2.9702	2.2209	8.8220	0.5367

By comparing the results of the Dense Layers and CNN Layers approaches, we can determine the effectiveness of each model in predicting the astrophysical parameters. The evaluation metrics provide valuable information about the model's performance, allowing us to assess its accuracy and reliability in parameter estimation.

Approach 2: Classification Method using Dense and CNN Layers

In this approach, we adopt a classification technique to predict the values of the astrophysical parameters. The parameter ranges for 'HII_EFF_FACTOR', 'ION_Tvir_MIN', and 'R_BUBBLE_MAX' are divided into 25 classes each. Each class represents a specific range of values within the overall parameter range.

Here is a detailed explanation of the approach:

1. Class Definition:
   - For each parameter, such as 'HII_EFF_FACTOR', the range [5, 200] is divided into 25 classes. Each class represents a sub-range of the parameter values.
   - For example, if the 'HII_EFF_FACTOR' value for a given power spectrum falls between 5 and 12, it will be assigned to class 0. If it falls between 12 and 19, it will be assigned to class 1, and so on.
   - Similarly, this class assignment process is performed for 'ION_Tvir_MIN' and 'R_BUBBLE_MAX' parameters, resulting in class assignments for each power spectrum.

2. Data Preprocessing:
   - The combined dataset containing the power spectrum data and astrophysical parameters is loaded.
   - The features are extracted from the power spectrum data, excluding the first two elements in each row.
   - The labels are the assigned classes for each parameter value.

3. Data Splitting:
   - The dataset is split into training and testing sets, preserving the class distribution in both sets.

4. Dense Layers:
   - Neural Network Architecture: A Sequential model is defined with dense layers.
   - Model Training: The model is compiled using appropriate loss and optimizer functions.
   - Early stopping is implemented to prevent overfitting.
   - The model is trained on the training data, where the input features are the power spectrum data and the target is the assigned class for each parameter value.
   - Model Evaluation: The trained model is evaluated on the testing data. Predictions are made for each power spectrum, and the assigned classes are compared with the predicted classes.
   - Evaluation metrics such as accuracy, precision, recall, and F1-score are calculated.

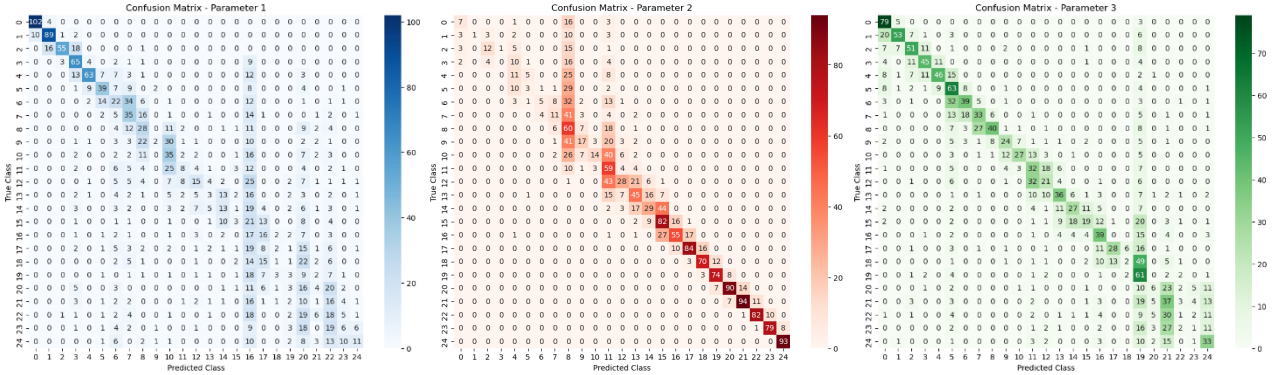
5. CNN Layers:
   - Data Preprocessing and Data Splitting steps remain the same.
   - Reshaping Input Data: The input data is reshaped to match the expected shape for a 1D CNN.
   - Neural Network Architecture (1D CNN): A Sequential model is defined with 1D convolutional layers, max pooling layers, and dense layers.
   - Model Training and Evaluation follow similar steps as the Dense Layers approach.

By employing the classification method, we transform the parameter prediction problem into a multi-class classification task. The models are trained to recognize patterns in the power spectrum data and assign the correct class to each parameter value. The performance of each model is evaluated using accuracy, precision, recall, and F1-score.

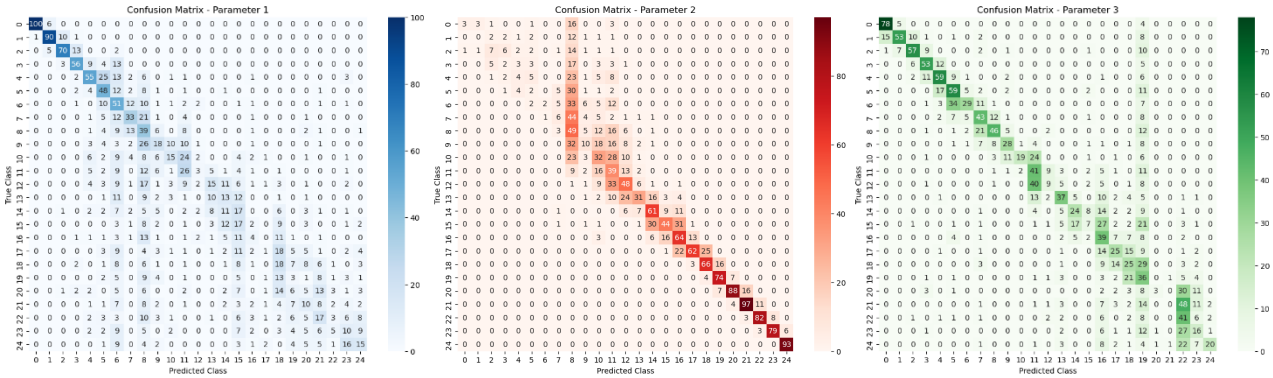
Comparing the results of the Dense Layers and CNN Layers approaches allows us to determine which model performs better in classifying the parameter values. The evaluation metrics provide insights into the model's ability to accurately assign the correct class to each parameter, indicating its effectiveness in predicting the astrophysical parameters based on the power spectrum data.

Results

ANN



CNN



The confusion matrices display the classification model's performance in predicting 'HII_EFF_FACTOR', 'ION_Tvir_MIN', and 'R_BUBBLE_MAX'. Diagonal elements show correct predictions, while off-diagonal elements represent misclassifications. Intensity indicates occurrences, and a strong diagonal line indicates accurate predictions. The matrices provide insights into the model's effectiveness and guide evaluation and refinement.

AstroPowerSpec-ML: Predicting Astrophysical Parameters from Power Spectrum Data using Machine Learning

This research project focuses on predicting astrophysical parameters, namely 'HII_EFF_FACTOR', 'ION_Tvir_MIN', and 'R_BUBBLE_MAX', based on power spectrum data in astrophysics. The power spectrum data captures the statistical properties of the early universe, and by understanding the relationship between the power spectrum and astrophysical parameters, we gain insights into the underlying physical processes.

The research workflow involves three key steps:

1. Data Generation and Cleansing:
   - Generate a large dataset of power spectra by varying the astrophysical parameters within specified ranges.
   - Cleanse the data by removing irrelevant information and handling any inconsistencies or missing values.

2. Machine Learning Model Development:
   - Implement machine learning models to predict the astrophysical parameters based on the power spectrum data.
   - Two approaches are explored:
        - Approach 1: Regression Method:
        - Utilize dense and convolutional neural networks (CNN) to establish a relationship between power spectra and parameters.
        - Train the models using the generated dataset and evaluate their performance.

        - Approach 2: Classification Method:
        - Transform the parameter prediction problem into a multi-class classification task.
        - Assign classes to the parameter values based on their ranges.
        - Train dense and CNN models to classify the parameters using power spectrum data.

3. Model Evaluation and Analysis:
   - Assess the performance of the developed models using appropriate evaluation metrics such as mean absolute error, R-squared score, accuracy, precision, recall, and F1-score.
   - Compare the results of the regression and classification approaches to determine the most effective method for predicting astrophysical parameters.

The project includes the necessary code, data, and documentation to reproduce and understand the research. The code is organized into distinct sections, including data generation, data cleansing, model development, training, and evaluation. The accompanying documentation provides detailed explanations of the methods employed and the results obtained.

The research findings contribute to the field of astrophysics by advancing our understanding of the relationship between power spectra and astrophysical parameters. The predictive models developed through this research can be utilized to estimate these parameters for new power spectra, enabling further exploration and analysis of the early universe.