Using Machine Learning to Estimate the Photometric Redshift of Galaxies

---

Shayaan Salim

Supervisor(s):
Nukri Komin, Hairong Bau

A research report submitted in partial fulfillment of the requirements for the degree of Master of Science in Artificial Intelligence

in the

School of Computer Science and Applied Mathematics
University of the Witwatersrand, Johannesburg

1 August 2023

Declaration

I, Shayaan Salim, declare that this research report is my own, unaided work. It is being submitted for the degree of Master of Science in Artificial Intelligence at the University of the Witwatersrand, Johannesburg. It has not been submitted for any degree or examination at any other university.

Shayaan Salim

1 August 2023

Abstract

Machine learning has emerged as a crucial tool in the field of cosmology and astrophysics, leading to extensive research in this area. This research study aims to utilize machine learning models to estimate the redshift of galaxies, with a primary focus on utilizing photometric data to obtain accurate results. Five machine learning algorithms, including XGBoost, Random Forests, K-nearest neighbors, Artificial Neural Networks, and Polynomial Regression, are employed to estimate the redshifts, trained on photometric data derived from the Sloan Digital Sky Survey (SDSS) Data Release 17 database. Furthermore, various input parameters from the SDSS database are explored to achieve the most accurate redshift values. The research incorporates a comparative analysis, utilizing different evaluation metrics and statistical tests to determine the best-performing algorithm. The results indicate that the XGBoost algorithm achieves the highest accuracy, with an R^2 value of 0.94, a Root Mean Square Error (RMSE) of 0.03, and a Mean Absolute Average Percentage (MAPE) of 12.04% when trained on the optimal feature subset. In comparison, the base model achieved an R^2 of 0.84, a RMSE of 0.05, and a MAPE of 20.89%. The study contributes to the existing literature by utilizing photometric data during model training and comparing different high-performing algorithms from the literature.

Acknowledgements

The author gratefully acknowledges the efforts of Dr. Helen Robertson, that provided detailed guidance through the creation of lecture material and consultations. The author would also like to express gratitude to Prof. Nukri Komin and Dr. Hairong Bau for supervising the project and Prof. Terence Van Zyl for assisting with the template used for this report.

Contents

Declaration i

Abstract ii

Acknowledgements iii

List of Figures vi

List of Tables vii

1 Introduction 1
- 1.1 Literature review . . . 3
- 1.2 Problem Statement . . . 6
- 1.3 Research Aims and Objectives . . . 7
  - 1.3.1 Research Aims . . . 7
  - 1.3.2 Objectives . . . 7
- 1.4 Limitations . . . 8
- 1.5 Overview . . . 8

2 Research Methodology 10
- 2.1 Available Methodologies . . . 10
- 2.2 Adopted Approach . . . 12
  - 2.2.1 Machine Learning Models . . . 12
    - Random Forests . . . 12
    - XGBoost . . . 14
    - ANN . . . 16
    - KNN . . . 18
    - Polynomial Regression . . . 20
    - Genetic Algorithms . . . 22

2.2.2 Data Collection and Data Pre-Processing 23
2.2.3 Evaluation Metric 27
2.2.4 Summary of Packages Used 29
2.2.5 Ethical Considerations 29
2.2.6 Overview of Implementation Process 30

3 Results and Analysis **31**
3.1 Results and Analysis 31
3.1.1 Data Analysis 31
3.1.2 Hyperparameter Tuning 36
3.1.3 Base model 38
3.1.4 Genetic Algorithms 40
3.1.5 Analysis of Features 43
3.1.6 Best Models 45
3.1.7 PCA Analysis 50
3.1.8 Statistical Tests 52
3.1.9 Future Recommendations 54
3.2 Summary 55

4 Conclusion **56**

A Appendix Title **57**
A.1 Query for Data Retrieval 57

Bibliography **60**

List of Figures

2.1 Support Vector Regression Hyperplane Visualization 11
2.2 Random Forest Representation: Figure presents the workflow to obtain an output from random forest algorithm 13
2.3 Visualization of ANN model 17
2.4 Visualization of KNN model 19
2.5 Diagram of Implementation: Figure displays a block diagram of the overall implementation of the project. It describes the workflow process which would be required to reproduce the research 30

3.1 Histogram of Spectroscopic Redshift Data Obtained from SDSS Database 32
3.2 Density Plot of Redshfit Data 33
3.3 Correlation Plot of u-g-r-i-z Features Obtained from SDSS Dataset . . 35
3.4 Obervsational vs Predicted Redshift Values for the XGBoost Algorithm Trained on Base Subset of Features 40
3.5 MAPE Values Obtained Over Various Iterations for Genetic Algorithms Optimization 40
3.6 Correlation Plot of the Best Features Obtained From Genetric Algorithms 44
3.7 Observational vs Predicted Redshift Values for XGBoost algorithm trained on Optimal Features 46
3.8 Histogram of Errors Between Observational and Predicted Redshift Values 47
3.9 2D Histogram of Observational vs Predicted Redshift Values 48
3.10 Variance Ratio vs Principal Component: Figure represents the number of principal components it would require to explain the variance in the data 51

List of Tables

2.1 Subset of Features from the SpecPhoto Table 25

3.1 Summary Statistics of Redshift Data Obtained from SDSS Database . 34
3.2 Best Parameters Found from Hyperparameter Tuning 36
3.3 Performance metrics of various machine learning models 38
3.4 Best Features Obtained Through Genetic Algorithms 42
3.5 Performance metrics of various machine learning models (including variances) 46
3.6 XGBoost trained on data transformed by PCA 52

Chapter 1

Introduction

In the early 1900's many scientists proposed contrasting theories on the origin of the universe (Ellis, 1984). From the various theories, there is a strong consensus towards the Big Bang theory. The Big Bang theory is the prevailing scientific explanation for the origin of the universe. According to the theory, the universe began as a single point consisting of infinite density and temperature, which rapidly inflated in a massive explosion around 13.8 billion years ago (Terr, 2013). This explosion created all matter and energy in the universe, including the stars, galaxies, and all the elements in the observable universe (Thompson, Harrub, and May, 2003). The theory rapidly gained popularity after Edwin Hubble's critical discovery that the universe is expanding. In 1929, Edwin Hubble measured the spectroscopic redshifts and the relative distance of a number of distant galaxies. When plotting the redshift against the relative distance of galaxies from earth Edwin Hubble found that the redshift values of distant galaxies increase as a linear function of their distance (Thompson, Harrub, and May, 2003). The only viable explanation for this phenomenon is that the universe is expanding (Enrera, 2021). The expansion of the universe suggested that the further back in time the universe is observed the smaller it would be (Bahcall, 2015). This observation supports the Big Bang theory in that the origin of the universe began from an extremely tiny ball of infinite density and immense heat. This is scientifically known as a singularity (Fox, 2002).

The relationship between the redshift and relative distance of galaxies became scientifically known as Hubble's law. In recent times, Hubble's law is used to measure the approximate distances of galaxies. This technique, however, requires the redshift of galaxies to be calculated. Redshift is best measured using spectroscopic analysis, which is based on comparing the spectrum of galaxies against a known

spectrum. The difference in emission or absorption lines is then used to calculate the precise redshift value (Lehnert et al., 2010). Redshift can be described as an example of the Doppler effect. As the object moves further away from the point of reference, light and sound waves are stretched apart. This causes the light waves to have a longer wavelength and thus they are moved toward the red end of the electromagnetic spectrum. This is known as redshift, whereas the inverse of this occurrence is known as blueshift ("Redshift" 2020).

The spectroscopic measurement of redshift provides accurate values. However, spectroscopic analysis is highly inefficient and infeasible for large catalogs of galaxies. Additionally, Hubble measured the relative distances of these galaxies using a variety of techniques, including Cepheid variable stars, which are stars that pulsate in a regular and predictable way. By measuring the time it takes for a Cepheid star to pulsate, astronomers can determine its intrinsic brightness, and then use this information to calculate its distance. However, for most distant galaxies the distance calculated in this way is not entirely accurate (Baryshev et al., 2012). An alternative method to obtaining redshift is through the use of photometric data. This method involves using the measured brightness of galaxies in different color bands to obtain an approximation for redshift. This method is known as photometric redshift estimation and will be investigated in this research using machine learning techniques (Brammer, Dokkum, and Coppi, 2008).

Advances in computing technology have revolutionized the field of scientific research, enabling scientists and researchers to simulate and model complex real-world problems that were previously impossible to solve. Machine Learning is a subfield of Artificial Intelligence and has developed as a powerful tool in this endeavor, allowing for the development of sophisticated models that can learn and improve from large datasets. The application of machine learning in the field of astrophysics and cosmology has opened up new avenues of research, providing researchers with the ability to simulate universes in a fraction of the time, leading to important breakthroughs in our understanding of the cosmos (Mathuriya et al., 2018). As such, this research aims to analyze various machine learning models to estimate the redshift of galaxies. Specifically, the models are designed to use photometric data, which is obtained from the measurements of the brightness of galaxies

in different bands of light. Additionally, a machine learning technique known as feature selection will also be implemented, which is the process of identifying the most relevant features in the dataset and is used to improve the accuracy of the models. The proposed method has the potential to improve the accuracy and efficiency of redshift estimation, which could have important implications for our understanding of the cosmos.

This research project details the design, technologies and limitations of the implemented methodology. The remainder of Chapter 1 provides information on existing literature and formally describes the research aims, goals and limitations. Chapter 2 provides an in-depth review of the technologies and methodologies that will be used to achieve the desired goals. Chapter 3 presents the findings and analyses of the study. Lastly, Chapter 4 concludes the research report by summarizing the key findings and offering final remarks.

1.1 Literature review

The utilization of machine learning within the domain of astrophysics has the potential to facilitate a more expeditious and nuanced understanding of the universe, thereby representing a significant advancement in the ability to comprehend the cosmos. Redshift values are a key characteristic for extra-galactic studies (Pasquet et al., 2019). Redshift is a measure of the degree to which light from distant galaxies has been shifted towards longer, redder wavelengths due to the expansion of the universe. Thus, estimating the redshift of galaxies is critical for understanding the large-scale structure of the universe and the processes that have driven its evolution over time. As a result, many researchers have conducted work in an attempt to better estimate redshift.

Wang, Zhang, and Zhao (2008) states that estimating the redshift of a galaxy is a complex non-linear problem. The paper looks into existing research and uses Artificial Neural Networks (ANNs) and Support Vector Regression (SVR) models for

predicting photometric redshift values. These types of models can be used for predicting trends, however, can not easily be extrapolated for values near the spectroscopic limit. The spectroscopic limit represents the boundary beyond which obtaining precise spectroscopic measurements of a celestial object becomes challenging or unfeasible due to technical limitations and the nature of the observed signals. As estimating redshift can be classified as a regression problem, the authors looks into using Linear Regression and Polynomial Regression. Polynomial Regression is a special case of multiple linear regression where the relationship between input and output is defined by an nth degree polynomial. In the paper, training data of 330000 samples obtained from the Sloan Digital Sky Survey (SDSS) is used. The models are designed, implemented and compared using the Root Mean Square Error (RMSE) to gauge their effectiveness. The Polynomial model outperforms the linear model and it is suggested that outliers be removed from the data to obtain better results. The authors state the resulting model is simpler to use than other empirical methods for calculating redshift and provides an equivalent accuracy. Similar to the work by Wang, Zhang, and Zhao (2008), Momtaz, Salimi, and Shakeri (2022) also use data from the SDSS collection. However, instead of using regression, Momtaz, Salimi, and Shakeri (2022) make use of decision trees and random forests. The paper describes the usage of the five optical magnitudes, u-g-r-i-z, as the input to the models. The results obtained are evaluated based on RMSE, accuracy and normalized standard deviation. The authors conclude by stating that the random forest model is able to provide more accurate predictions and achieves an overall better performance than the decision tree model.

De Wei and Yang (2019) use supervised learning approaches to estimate redshift. The noteworthy algorithms used by De Wei and Yang (2019) are Extreme Gradient Boosting (XGBoost) and ANNs. Similar to Wang, Zhang, and Zhao (2008) and Momtaz, Salimi, and Shakeri (2022), De Wei and Yang (2019) use RMSE to measure the effectiveness of the model. The authors concludes by stating that the XGBoost algorithm outperforms the ANN and is able to achieve an RMSE of 0.07. Furthermore, Zhang et al. (2020) similar to De Wei and Yang (2019), uses the XGBoost algorithm and thereafter compares it to the K Nearest Neighbour (KNN) algorithm. The KNN algorithm is a non-parametric method used for classification and regression where for regression, the output is the average of the k closest training examples.

The authors use the Mean Absolute Error (MAE) and Mean Square Error (MSE) metrics to compare the models. From the obtained results, it is seen that the KNN model achieves better results than XGBoost. The authors conclude by stating the KNN algorithm is highly suggested for large amounts of data.

Schuldt et al. (2021) discusses in-depth the need for estimating redshift and its relation to Hubble's law. The paper focuses on an algorithm named NetZ, which is a new method based on Convolutional Neural Networks (CNNs). The model performs well for redshift between the range of 0-4 and is only limited by available training data. Hoyle et al. (2015) uses data augmentation techniques to enhance the accuracy obtained using decision trees. The author states that the model performs well for lower values of redshift. D'Isanto and Polsterer (2018) uses a deep convolutional network that is combined with a mixture density network. The end result estimation is expressed as a Gaussian mixture model representing the Probability Density Functions (PDFs) in the redshift space.

Lastly, the work conducted by the Henghes et al. (2021) encompasses a thorough analysis of various machine learning models for photoredshift estimation. The study incorporates SDSS Data Release 12 and explores multiple machine learning models, such as Random Forest, Gradient Boosting, and KNN. The dataset used consists of 1,639,348 samples, with the u-g-r-i-z variables used in model training. It is noteworthy that only galaxies with a redshift under 1 were considered in this research, and additional variables that could enhance model training were not explored. The study concludes by identifying Random Forest as the most effective model, achieving an MSE of 0.0042, with other models, such as Gradient Boosting and KNN, yielding similar results.

From the literature reviewed, it was found that several machine learning models are particularly well-suited for estimating redshift values. These models include XGBoost, random forests, KNNs, ANNs, and polynomial regression. All these models have demonstrated high performance in different regression problems and have been used extensively in astronomical applications. Given the importance of redshift estimation in astronomy, this research uses a select number of machine learning models to estimate the redshift of galaxies. The models are designed to

use photometric data to estimate the redshift, which is a widely accepted approach in astronomy. Photometric data includes measurements of the flux or brightness of an object in different wavelengths, which can provide crucial information about the object's redshift. Furthermore, the models are compared to identify the most suitable and best-performing model for the estimation of the redshift of galaxies. The comparison is based on various performance metrics such as accuracy, RMSE and MAE. The aim of the research report is to identify a model that provides the most accurate and reliable estimates of redshift values for galaxies.

1.2 Problem Statement

The field of astrophysics and cosmology has seen a surge in the popularity of machine learning. As such, astrophysicists and cosmologists would be highly interested in developing models that aid in studying the universe. Redshift value provides crucial information on the distance and velocity of celestial objects, such as galaxies. Therefore, the development of a model that can provide accurate redshift estimation is of great importance in the field of astronomy. Ideally, a model that can provide accurate estimation for redshifts between galaxies is desired. There is a lack of literature that analyzes and compares multiple machine learning algorithms to identify the most efficient algorithm and input features for redshift estimation. This research report proposes to develop multiple machine learning models for the estimation of redshifts between galaxies. In addition, feature selection techniques will be used to identify the most prevalent features for photometric redshift estimation. This approach bridges the gap between the current literature and the ideal situation by utilizing photometric data for redshift estimation and analyzing, comparing, and identifying the most suitable algorithm and features for the task. By doing so, this research project aims to contribute to the field by providing valuable insights into the accurate estimation of redshift values, which can aid in advancing the understanding of the universe.

1.3 Research Aims and Objectives

The research aims and objectives assist in scoping the project.

1.3.1 Research Aims

The aim of this research report is not only to develop and compare multiple machine learning models for the estimation of redshifts between galaxies but also to identify the most suitable algorithm and features for the task. This involves the use of feature selection techniques to identify the most relevant features for photometric redshift estimation.

1.3.2 Objectives

In order to achieve the aim of developing and comparing multiple machine learning models for the estimation of redshifts between galaxies, the following objectives have been identified:

1. Obtain a suitable dataset of photometric data that can be used to train the machine learning models. This involves identifying a relevant dataset that includes the appropriate photometric data.

2. Implement appropriate data pre-processing techniques to ensure that the models can achieve the best possible results. This may involve feature scaling, data normalization, or other techniques that can help to improve the performance of the models.

3. Train the machine learning models using the selected dataset to generate a baseline model.

4. Use optimization algorithms to experiment with different subsets of features to obtain more accurate models.

5. Use a range of evaluation metrics to assess the effectiveness of the models, including metrics such as accuracy, RMSE, and MAE. This helps to ensure that the models are performing as expected and provides a benchmark for comparing different models.

6. Compare the performance and accuracy of the different machine learning models to identify the most effective approach for estimating redshifts between galaxies. This involves analyzing the results of the evaluation metrics and drawing conclusions about the relative strengths and weaknesses of each model.

1.4 Limitations

In this section, the scope of the project is outlined, and its limitations are discussed. Several limitations that could affect the project's success have been identified, and are provided below:

1. The model will be trained on astronomical data, which is large. Therefore, it is possible that the dataset may need to be truncated, as processing too large of a dataset would require an excessive amount of time to train the models. This could impact the accuracy of the results obtained and limit the scope of the project.

2. The research may need to utilize multi-threading if the models take too long to train. This can significantly improve the speed of the training process, but may also require additional hardware resources or programming expertise.

3. The research project has a fixed time frame for completion, as the due date for the final report is on the 23rd of March 2023. This limits the amount of time available for data collection, analysis, and modeling. Consequently, the project may not be able to explore all possible avenues for improving model accuracy or evaluating the effectiveness of different machine learning algorithms in estimating redshifts between galaxies.

1.5 Overview

The primary objective of this research project is to determine the most effective machine learning model for estimating redshifts of galaxies. This report assesses the performance of several machine learning models, namely XGBoost, random forests, KNNs, ANNs and Polynomial Regression. The models are trained on photometric

data with available spectroscopic results, and feature selection techniques are applied to identify the most crucial parameters for redshift estimation. Additionally, the pre-processing of photometric data is carried out to enhance the accuracy of the models. The efficacy of the models is measured using various metrics, including MAE, MSE and R-squared. A thorough comparison between the different models is performed to determine the best-performing model. The outcome of this study provides critical insights into the most effective methods for estimating photometric redshifts of galaxies using machine learning.

Chapter 2

Research Methodology

This section provides an overview of both the chosen and available methodologies used in order to complete the research project.

2.1 Available Methodologies

As demonstrated in the existing literature, there are several different methodologies that can be employed in the estimation of redshift values for galaxies. The primary objective of this research is to create machine learning models capable of accurately estimating the photometric redshift of galaxies and to conduct a comparative analysis of their performance. While some of the most effective models have already been discussed in the literature, a number of other models could potentially be used as well, some of which are discussed below.

A SVR model is a modern machine learning algorithm that is used in classification and regression type problems. In the context of the research, the SVR would group galaxies with similar redshifts into categories and use these categories to estimate the redshift of new galaxies. The SVR model accomplishes this by constructing a hyperplane in a multi-dimensional space that best separates the galaxies based on their redshifts (Smola and Schölkopf, 2004). Figure 2.1 provides an illustration of this process. This hyperplane could then be used with new data to determine if the galaxy belongs in a certain range of redshift values. However, as the nature of estimating redshift is very complex and curve-fitting approaches can not easily be extrapolated to values near the spectroscopic limit. SVRs would not be able to achieve good accuracy. (Wang, Zhang, and Zhao, 2008).

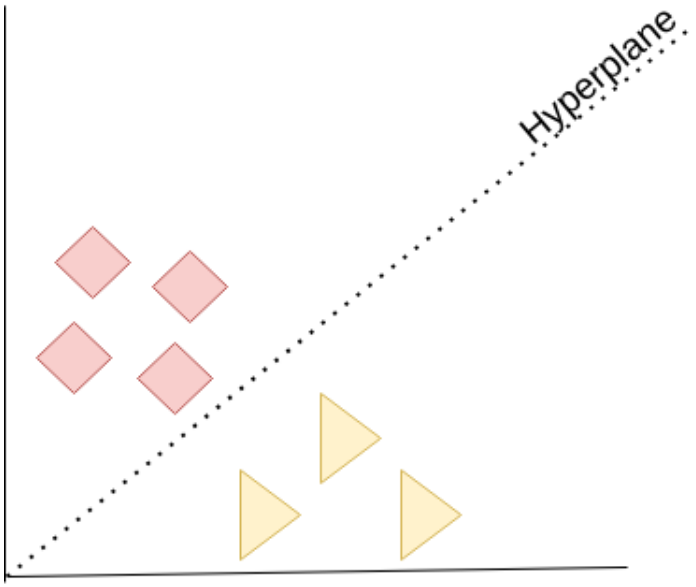

FIGURE 2.1: Support Vector Regression Hyperplane Visualization

Decision trees are tree-like structures that are created from datasets. These trees are based on conditional control statements and are made up of nodes and edges which represent decisions and outcomes. They are used to solve classification and regression problems (Kingsford and Salzberg, 2008). The first node of the tree is referred to as the root node, while the last nodes are known as leaf nodes. A decision tree is constructed by initially calculating the information gain which is based on the entropy and subsequently selecting the best feature for the given branch at a point. The entropy of a set is referred to as the measure of disorder in a system. The entropy is calculated using Equation 2.1, in which S is the dataset and p_i is the probability of class i. The information gain is a measure of how well a feature classifies the target column and is calculated using Equation 2.2, in which, the left-hand side of the equation, S is the dataset and A is any feature column. On the right-hand side of the equation, S_v is the select set of rows in S for which the feature column A has value v and $|S|$ is the number of rows in S (Magee, 1964).

$$Entropy(S) = -\sum_{v=1}^{n} p_i \times log_2(p_i) \tag{2.1}$$

$$IG(S, A) = Entropy(S) - \sum_{i=1}^{n} \frac{|S_v|}{|S|} \times Entropy(S_v) \tag{2.2}$$

Decision trees may fall short in capturing all the intricate details of complex regression problems that involve multiple variables with non-linear relationships. This is primarily due to the tendency of the algorithm to over-fit. However, an alternative approach that overcomes this issue is random forest - an ensemble method that employs multiple decision trees and combines their predictions to produce a final result. Random forest significantly reduces the over-fitting problem and generates more precise results, especially in cases where the problem involves many variables. Additionally, the random forest algorithm offers feature importance analysis, which helps in identifying the crucial variables in the model. As such, utilizing random forest for complex regression problems can deliver more precise and reliable outcomes compared to a single decision tree.

2.2 Adopted Approach

The solution to the research problem can be formulated by using various machine learning models to estimate the redshift of galaxies. Once the models are trained, they are then evaluated using various machine learning evaluation metrics to gauge the effectiveness of each model. Additionally, a comparative analysis is conducted using statistical techniques to ascertain the best model.

2.2.1 Machine Learning Models

Random Forests

The random forest model is a widely used supervised machine learning algorithm that is made up of multiple individual decision trees. As discussed above, a decision tree is a tree-like structure in which each internal node represents a trial on an attribute, each branch represents the trial's outcome, and each leaf node represents

a decision made (Biau and Scornet, 2016). The random forest model combines decision trees together in order to obtain a more accurate prediction of results.

The difference between decision trees and random forests is that in decision trees, a set of rules are created to make predictions, whereas random forests randomly select features to build several decision trees and then average the results. Random forests thus obtain the best feature in a subset of features. Decision trees may also suffer from over-fitting, while random forests try to prevent the problem of over-fitting through the use of sub-trees (Yiu, 2019).

As the trees are individual in a random forest model, there is a low correlation between the trees, which allows for fewer errors to occur, as each tree is mutually exclusive. Each tree produces a prediction, and once all tree predictions have been completed, the most widely produced prediction is seen as the best result, and that result is chosen. The prediction process is formally known as majority voting. This process is visualized below in Figure 2.2.

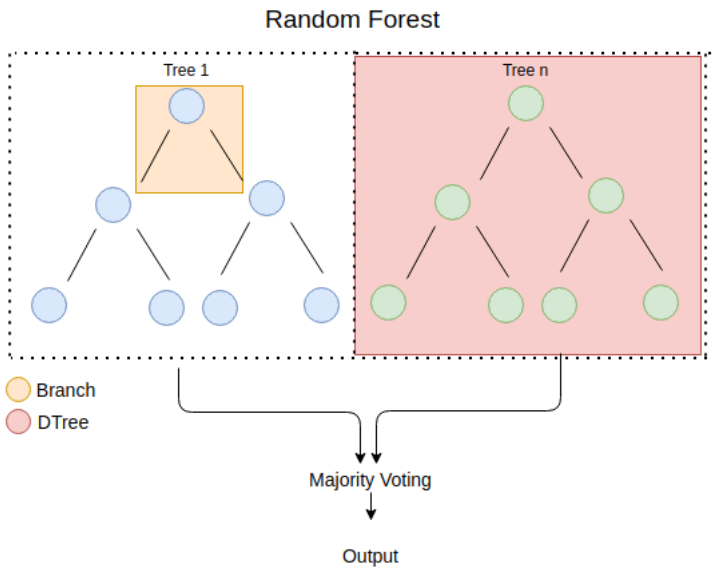

FIGURE 2.2: Random Forest Representation: Figure presents the workflow to obtain an output from random forest algorithm

Random forests are used for both regression and classification problems, which

make them versatile to use (Biau and Scornet, 2016). This research employs the random forest model which is based on the following procedure.

The Random Forest algorithm takes as input a dataset with N examples and M features and a parameter *n_estimators* that determines the number of trees to include in the forest. For each tree, a random subset of the input data is selected, with replacement, so that each subset has N examples. Additionally, a random subset of M features is selected as the input for each tree, so that each tree uses a different subset of features.

Each decision tree is trained on its subset of the data using a standard decision tree algorithm. At each node in the tree, the algorithm selects the feature that best splits the data based on a criterion such as information gain as described by Equation 2.2. The tree continues to split until a stopping criterion is reached, such as a minimum number of examples at a leaf node.

To make a prediction with the random forest model, the algorithm first runs the input data through each tree in the forest, producing n predictions. For a regression problem, the final prediction is typically the mean of the n predictions as shown in Figure 2.2 above.

XGBoost

XGBoost is a state-of-the-art machine learning algorithm that is widely used for solving regression, classification, and ranking problems. As a decision-tree-based algorithm, XGBoost is an ensemble learning method that combines multiple decision trees to make highly accurate predictions. The algorithm uses a gradient-boosting framework that iteratively adds new decision trees to the ensemble, with each tree aiming to correct the errors of the previous ones. The algorithm's efficacy is further enhanced by a range of advanced features, such as early stopping, tree pruning, and parallel computing. Early stopping helps prevent over-fitting, while tree pruning simplifies the decision trees and makes the data more interpretable. The algorithm's success is attributed to its ability to handle noisy or sparse data, which is achieved through the use of a regularized objective function that balances the training loss and a penalty term to prevent over-fitting. This approach

allows XGBoost to learn highly complex, nonlinear relationships between input features and output targets, which is especially useful in many real-world applications (Chen and Guestrin, 2016).

In addition to its exceptional performance, XGBoost is a popular choice for many machine learning tasks due to its strong support for parallel computing, which enables it to efficiently train models on large datasets with many features.

The equations of the XGBoost algorithm can be written as follows:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i),\ f_k \in \mathcal{F} \tag{2.3}$$

$$L(\Theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{2.4}$$

In Equations 2.3 and 2.4 the $\hat{y}_i$ is the predicted value for the i-th instance, $f_k(x_i)$ is the prediction of the k-th tree on the i-th instance, $\mathcal{F}$ is the space of all possible regression trees, K is the number of trees in the ensemble and $l(y_i, \hat{y}_i)$ is the loss function that measures the difference between the true label y_i and the predicted label. Lastly, $\Omega(f_k)$ is the regularization term that penalizes complex models and Θ is the set of all model parameters (Chen et al., 2015).

The XGBoost algorithm works by iteratively adding weak models to the ensemble, with each new model focusing on the residuals of the previous model. The algorithm begins by initializing the prediction to a constant value, which is typically the mean of the target variable. Then, at each iteration, a new tree is added to the ensemble that tries to correct the errors of the previous tree. The tree is trained to predict the negative gradient of the loss function, which is the amount by which the current prediction differs from the true label. The prediction of the ensemble is then updated by adding the prediction of the new tree, weighted by a learning rate parameter that controls the contribution of each tree to the final prediction (Chen et al., 2015).

The regularization term $\Omega(f_k)$ encourages the algorithm to create simpler trees by

penalizing the number of leaves and the magnitude of the leaf weights. This helps to prevent over-fitting and improves the generalization performance of the model.

ANN

ANNs were invented in 1958 by psychologist Frank Rosenblatt (Jain, Mao, and Mohiuddin, 1996). They were intended to model how the human brain processed visual data, learned to recognize objects, and were used to discover relationships between input and output features. ANNs are computational models that are built from multi-layer perceptrons. The most common architecture in ANNs consists of three layers, namely, input hidden and output layers. The input layer receives the input data, and each subsequent layer performs a series of nonlinear transformations on the input data. Each neuron receives input from the previous layer and applies a mathematical function to the input to produce an output, which is passed to the next layer. During training, the weights and biases of the neurons are updated to minimize the difference between the predicted output and the actual output. This process is called back-propagation, which involves computing the error between the predicted and actual output and using this error to adjust the weights and biases of the neurons in the network. As a result of this iterative process, ANNs can learn complex patterns and relationships in the data and can make accurate predictions on new, unseen data. Figure 2.3 presents a visual representation of an ANN.

The following procedure is used by the ANN's algorithm to develop a model (Yegnanarayana, 2009):

1. Take in the input features

$$\mathbf{x} : \mathbf{x} = [x_1, x_2, \ldots, x_n]^T \tag{2.5}$$

2. Compute the weighted sum of the inputs to each neuron in the first hidden layer, using weights $\mathbf{W}^{(1)}$ and biases $\mathbf{b}^{(1)}$:

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \tag{2.6}$$

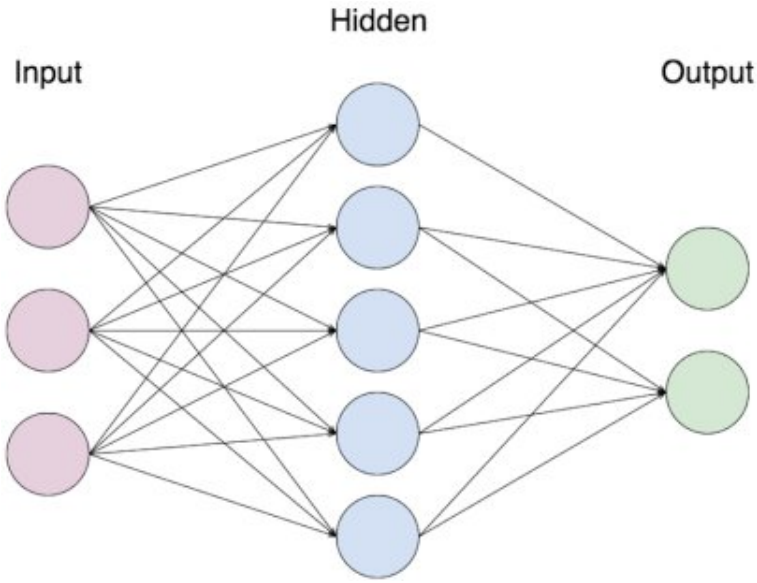

FIGURE 2.3: Visualization of ANN model

3. Apply the sigmoid activation function σ to the weighted sums to obtain the outputs of the first hidden layer:

$$\mathbf{a}^{(1)} = \sigma(\mathbf{z}^{(1)}) \tag{2.7}$$

4. Compute the weighted sum of the outputs from the first hidden layer to the output layer, using weights $\mathbf{W}^{(2)}$ and biases $\mathbf{b}^{(2)}$:

$$\mathbf{z}^{(2)} = \mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)} \tag{2.8}$$

5. Apply the sigmoid activation function to the weighted sums to obtain the final output $\hat{y}$:

$$\hat{y} = \sigma(\mathbf{z}^{(2)}) \tag{2.9}$$

6. Compute the loss between the predicted output $\hat{y}$ and the true output y using a loss function $L(\hat{y}, y)$.

7. Use back-propagation to compute the gradients of the loss with respect to the weights and biases, starting from the output layer and working backward

through the network. The gradient of the loss with respect to the final output is given by:

$$\frac{\partial L}{\partial \mathbf{z}^{(2)}} = \frac{\partial L}{\partial \hat{y}} \cdot \sigma'(\mathbf{z}^{(2)}) \tag{2.10}$$

8. Update the weights and biases using an optimization algorithm, such as gradient descent:

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \alpha \cdot \frac{\partial L}{\partial \mathbf{W}^{(l)}} \tag{2.11}$$

$$\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \alpha \cdot \frac{\partial L}{\partial \mathbf{b}^{(l)}} \tag{2.12}$$

where α is the learning rate and l is the layer.

9. Repeat steps 2-8 for a specified number of iterations or until a set level of performance is achieved.

KNN

KNN is a simple yet powerful supervised machine learning algorithm that can be used to solve different types of problems. The KNN algorithm assumes that similar data exist in close proximity, and aims to capture the idea of similarity. In astronomy, for example, galaxies close together have similar redshifts. KNN can be used to predict the redshift of a new galaxy based on its proximity to other known galaxies. The algorithm finds the K nearest data points in the training dataset and determines their class or value. The classification or regression of the new data point is then determined by the majority vote or average value of the K nearest neighbors. The similarity between data points is measured by calculating the Euclidean distance between them, which is the straight-line distance between two points in a multi-dimensional space. The value of K is a hyperparameter that can be optimized by analyzing the error curves to find the best value for the model. The final value for the new data point is given by taking the average of the K most similar data points. While KNN is easy to implement and interpret, its main drawback is its sensitivity to the choice of distance metric and value of K, which can affect the accuracy of the model. Figure 2.4 displays an example of the KNN for a K value of 3. For regression type problems KNN predicts the continuous outcome by averaging

the observations in the same category (Peterson, 2009).

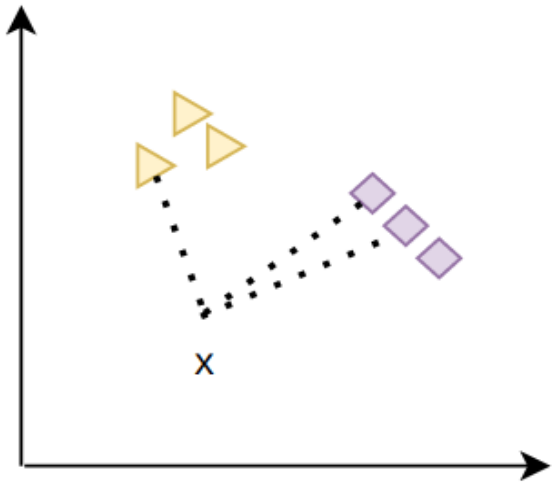

FIGURE 2.4: Visualization of KNN model

For the KNN algorithm, given a training dataset $D = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$, where $\mathbf{x}_i$ is the i-th input feature vector and y_i is the corresponding continuous output value, the KNN algorithm predicts the output for a new observation $\mathbf{x}$ by finding the K closest neighbors in the training set and taking the average of their output values (Peterson, 2009).

To calculate the distance between $\mathbf{x}$ and each training example, a distance metric such as Euclidean distance can be used:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.13}$$

where $d(x, y)$ is the Euclidean distance between two data points x and y in n-dimensional space, and the elements $(x_1, x_2, ..., x_n)$ and $(y_1, y_2, ..., y_n)$ are the respective coordinates in each dimension.

After computing the distances between **x** and all training examples, the K nearest neighbors of **x** with the smallest distances are selected. The predicted output value for **x** is the average of the output values of its K nearest neighbors:

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y_i \tag{2.14}$$

During the training phase, the only step is to store the training data. KNN is a memory-based algorithm, thus it uses the entire training dataset to make predictions for new data points.

The performance of KNN can be affected by the choice of the hyperparameter K, which determines the number of neighbors to consider. A small value of K may result in over-fitting, while a large value of K may result in under-fitting. The optimal value of K can be selected by evaluating the performance of the model on a validation set. Additionally, the distance metric can be customized based on the specific problem, and feature scaling may be necessary to ensure that all features contribute equally to the distance calculation.

Polynomial Regression

Polynomial regression is a type of regression analysis where the relationship between the independent variable (x) and dependent variable (y) is modeled as an n-th degree polynomial function. The algorithm can be used to model nonlinear relationships between the variables. By including higher-order terms in the model, the curve that fits the data can be more flexible and can capture more complex patterns in the data. However, over-fitting can be a problem with higher-order polynomials, which can lead to poor generalization to new data. To avoid over-fitting, regularization techniques such as ridge regression or lasso regression can be used (Ostertagová, 2012).

Polynomial regression is a form of linear regression in which the relationship between the input feature and output variable is modeled as an nth degree polynomial function. The polynomial function can be defined as follows:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \epsilon \tag{2.15}$$

where y is the output variable, x is the input feature, $\beta_0, \beta_1, \ldots, \beta_n$ are the coefficients of the polynomial function, and ϵ is the error term.

The task of polynomial regression is to estimate the values of the coefficients $\beta_0, \beta_1, \ldots, \beta_n$ from the training data, in order to find the best fitting polynomial function that approximates the relationship between the input feature and output variable. To estimate the coefficients, the method of least squares can be used, which involves minimizing the sum of squared residuals between the predicted values and actual values of the output variable. The sum of squared residuals is defined as follows:

$$SS_{res} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.16}$$

where y_i is the observed value and $\hat{y}_i$ is the predicted value of the output variable for the i-th training example.

The predicted value of the output variable can be obtained by substituting the input feature x_i into the polynomial function:

$$\hat{y}_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \cdots + \beta_n x_i^n$$

Additionally, to minimize the sum of squared residuals, the derivative of SS_{res} can be taken with respect to each coefficient β_j and set equal to zero:

$$\frac{\partial}{\partial \beta_j} SS_{res} = -2 \sum_{i=1}^{n} (y_i - \hat{y}_i) x_i^j + 2\lambda j \beta_j = 0$$

for $j = 0, 1, \ldots, n$, where λ is the regularization parameter that controls the degree of penalty on the coefficients to prevent over-fitting.

The above equations form a system of linear equations that can be solved for the values of the coefficients $\beta_0, \beta_1, \ldots, \beta_n$. This can be achieved through using matrix algebra, or by using specialized libraries or software packages.

Once the values of the coefficients are estimated, the polynomial function can be used to make predictions for new data points by substituting the input feature into the function. The degree of the polynomial function can be chosen based on the complexity of the relationship between the input feature and output variable, and the performance of the model can be evaluated using metrics such as the MSE or R-squared.

Genetic Algorithms

Genetic algorithms are machine learning algorithms inspired by the works of Charles Darwin's theory. They are a class of evolutionary algorithms that use a natural selection process to evolve towards an optimal solution. The algorithm works by generating a population of potential solutions and then selecting the fittest individuals from that population for reproduction. The reproduction process is done using genetic operators such as crossover and mutation, which mimic the biological processes of reproduction and mutation in living organisms (Kramer and Kramer, 2017).

The fitness of each individual in the population is evaluated using a fitness function, which quantifies how well each solution performs on the problem being solved. The fittest individuals are then selected for reproduction, where their genetic material is combined to produce offspring for the next generation. This process is repeated for several generations, with the hope that the population will evolve towards a more optimal solution over time. This research uses genetic algorithms for identifying the best subset of features for estimating the redshift of galaxies where features are chosen based on the highest accuracy obtained (Leardi, 1996). The following steps are taken by genetic algorithms to optimize solutions:

1. Generate an initial population with randomized individuals (where individuals mean a group of features)
2. Check the fitness of each individual
3. Select individuals with the highest fitness for reproduction
4. Cross-over the genes from the chosen individuals

5. Mutate some of the genes to offer randomness

6. Repeat for a set number of generations.

The genetic algorithm is coded using the procedure mentioned above. Three functions are created, named *selection() mutation()* and *crossover()*, these functions represent the biological operators that occur in natural reproduction. The selection function randomly selects two individuals and returns the one that has the highest fitness score. A new population is created by repeatedly running the selection function. The crossover function takes two parents and returns two children that have been crossed-over at random points. The probability that the crossover function will allow parent gene's to be crossed is set at 90%. Thereafter, the population is mutated by flipping random bits with a mutation probability of 10%. The population size is chosen as 20 individuals and the algorithm is run for 50 generations. This configuration allows for enough exploration and also allows for generations to converge towards a fitness.

2.2.2 Data Collection and Data Pre-Processing

The dataset is obtained from SDSS where SDSS has organised the most detailed maps of the universe ("Sloan Digital Sky Survey" 2022). SDSS is a massive astronomical project that has created one of the most comprehensive databases of astronomical information in the world. The project has provided an unprecedented view of the universe and has revolutionized our understanding of its structure, composition, and evolution. The SDSS database contains a vast amount of data, including images, spectra, and astrometry for millions of celestial objects, such as stars, galaxies, and quasars. This data has been used by researchers around the world to study a wide range of topics, including the large-scale structure of the universe, the properties of dark matter, the evolution of galaxies, and the behavior of black holes. As a result, the SDSS database has become an essential resource for astronomers and astrophysicists, and it has significantly advanced our knowledge of the universe. This research analyzes the data contained within the SDSS database to explore new insights and discoveries in the field of photometric redshift estimation.

The specific dataset that is used is the SDSS Data Release 17 (DR17). DR17 is the latest release of the fourth phase of the SDSS. DR17 contains observations through

January 2021 and includes all the sky coverage of prior releases. The redshift values were restricted to a maximum value of 2 as extremely high redshift sources may be considered as outliers. These outliers in the data are disregarded as they may negatively impact the accuracy of models. Furthermore, the SDSS dataset did not contain many entries for galaxies with high redshifts. Higher redshift values correspond to further away galaxies, with the furthest galaxy GN-z11 is known to have a redshift value of 11.09.

SDSS includes various tables and views to easily obtain specific data from data surveys. These tables are analyzed to identify an appropriate dataset for the research. It was found that the SpecPhoto table contains suitable attributes for the research. The SpecPhoto table includes a wide range of information on astronomical bodies, such as position in the sky, brightness in different wavelength bands, redshift values, and various other properties related to photometry. Table 2.1 summarises the key features included in the SpecPhoto table which are used for training the models.

TABLE 2.1: Subset of Features from the SpecPhoto Table

Feature	Description
ra	The SDSS database includes the Right Ascension of celestial objects, which is a measure of their angular distance eastward along the celestial equator from the vernal equinox (measured in degrees).
dec	The database also includes the Declination of celestial objects, which is a measure of their angular distance north or south of the celestial equator (measured in degrees).
psfMag	The Point Spread Function magnitude, also known as the "instrumental magnitude", is a measure of an object's brightness as measured by the SDSS instrument in a single exposure, before correcting for atmospheric effects and other factors. This magnitude assumes that the object is a point source.
fiberMag	The Fiber magnitude is a measure of an object's brightness as measured through a small circular fiber during the SDSS observation. This magnitude is a measure of the light that enters the fiber and is therefore not affected by atmospheric effects that can affect the PSF magnitude.
petroMag	The Petrosian magnitude attempts to account for the fact that the brightness profile of many galaxies is not well described by a simple PSF or Gaussian function. The magnitude is derived from the Petrosian radius, which is the radius at which the ratio of surface brightness to average surface brightness within that radius is 0.2. The Petrosian magnitude is then calculated as the total flux within twice the Petrosian radius, divided by the area within twice the Petrosian radius.
modelMag	The Model magnitude is a measure of an object's brightness predicted by a model of its surface brightness profile. The SDSS uses a de Vaucouleurs profile, a type of light distribution often used to describe the surface brightness of elliptical galaxies, to model the object's brightness profile.
cmodelMag	The Composite Model magnitude is a hybrid magnitude that combines the advantages of both the PSF and model magnitudes. It is a weighted average of the PSF and model magnitudes, with the weights determined by the goodness of fit of the model to the data.
extinction	Extinction is a measure of the amount of light absorbed or scattered by the Earth's atmosphere as it passes through to reach the SDSS telescope. The SDSS reports extinction coefficients in each band (u, g, r, i, z) in magnitudes.
dered	The Dereddened magnitude is a magnitude corrected for the effects of extinction due to dust in the milky way galaxy. The correction is made using the extinction coefficients provided by SDSS and assuming a standard extinction law.
mRrCc-r	The variable represents the standard deviation or uncertainty associated with the photometric properties
score	The variable represents the quality score or metric associated with the object's spectroscopic or photometric properties.

Data from the SDSS is retrieved using SQL. SQL is a programming language that is used to manage relational databases where it is used to perform various operations on the data in these databases (w3schools, 2022). This is achieved through the use of queries. The following code snippet is an example of the SQL query used to retrieve data with the full code snippet given in Appendix A.

```
SELECT z AS 'redshift', modelMag_u AS 'u', modelMag_g AS 'g',
modelMag_r AS 'r', modelMag_i AS 'i', modelMag_z AS 'z'
FROM SpecPhoto
WHERE zWarning = 0 AND class = 'GALAXY' AND z BETWEEN {} AND {}
```

SQL is used to retrieve data from the SDSS database. This includes spectroscopic redshift values and photometric data which include the brightness from the five optical bands, u-g-r-i-z. It is important to note that for each query the SDSS database returns a maximum of 500 000 rows of data. As a result, the query requests redshifts between small ranges to ensure that the limit of 500 000 rows is not reached. This can be seen in the SQL query above and where the BETWEEN keyword is used to obtain data within small ranges. The small ranges are then appended to the main data table. Once the data is obtained, the focus is shifted towards removing any null values and outliers from the dataset. Moreover, faint values of redshift were included in the dataset to ensure comprehensive coverage of astronomical objects. Additionally, during the data preparation phase, the warning flags associated with certain data points were not considered to prevent complications in the subsequent analysis.

The data is then pre-processed to improve the performance of the machine learning models. Feature scaling is a common technique used to ensure that all the features of the data are on the same scale. This is done by transforming the values of each feature to a similar scale such as between 0 and 1. Another common technique used in data pre-processing is feature standardization. This involves transforming the data to have a mean of zero and a standard deviation of one. Feature standardization can help to reduce the impact of outliers in the data and it is also a necessary step for principal component analysis.

After the data has been pre-processed, it is split into a training set, a cross-validation set, and a test set using a 70:20:10 ratio respectively. The training set is used to train the machine learning model, while the cross-validation set is used to identify the best subset of features that result in the highest accuracy. Finally, the test set is used to evaluate the performance of the model on unseen data. By using a test set that the model has not seen before, a more accurate measure of how well the model is likely to perform in the real world is obtained.

After rigorous data curation, the final dataset comprised 2,616,730 galaxies within a redshift range spanning from 0 to 2. The inclusion of this extensive sample of galaxies enhances the dataset's representativeness and enables a more comprehensive exploration of astronomical phenomena across different redshift intervals.

2.2.3 Evaluation Metric

The error of any system in general can be obtained by taking the difference between the true value and measured value (Willmott and Matsuura, 2005). However, as there are multiple points of error the error is calculated using the RMSE method described by Equation 2.17. This method allows for the errors of each individual component to be measured.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{2.17}$$

where y_i is the actual value, $\hat{y}_i$ is the estimated value and n is the number of data points. Furtheremore, the Mean Absolute Percentage Error (MAPE) is used to obtain a true representation of the accuracy of the models. MAPE is a commonly used performance metric in machine learning for evaluating the accuracy of a model's predictions. It is a measure of the average absolute percentage difference between the actual and predicted values. MAPE is often used in the context of forecasting and regression problems. A low MAPE value indicates that the model's predictions are close to the actual values, while a high MAPE value suggests that the model's predictions are not accurate. Additionally, an advantage of using MAPE is that it is a percentage-based metric, which allows for easy comparison of the performance

of different models (De Myttenaere et al., 2016).

MAPE is calculated using Equation 2.18:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \tag{2.18}$$

where y_i is the actual value of the i-th observation, $\hat{y}_i$ is the predicted value of the i-th observation, and n is the total number of observations. The absolute difference between the actual and predicted values is divided by the actual value, and then the average of these values is taken and multiplied by 100% to express the result as a percentage.

In addition to other performance metrics, the research will utilize the MSE and MAE to appropriately compare models with works from different literature, as these metrics are commonly used in research papers as outlined in the literature review section. The MSE and MAE are both measures of error or difference between predicted and actual values.

The MSE, defined as the average of the squared differences between predicted and actual values, is calculated as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.19}$$

Where n represents the number of samples, y_i represents the actual value, and $\hat{y}_i$ represents the predicted value for the ith sample. The squared differences are summed and then divided by the number of samples to give the average.

The MAE, defined as the average of the absolute differences between predicted and actual values, is calculated as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.20}$$

Where n represents the number of samples, y_i represents the actual value, and $\hat{y}_i$ represents the predicted value for the ith sample. The absolute differences are

summed and then divided by the number of samples to give the average.

Both the MSE and MAE are valuable metrics for evaluating regression models but possess distinct strengths and weaknesses. The MSE places greater emphasis on large errors because it squares the differences between predicted and actual values, which can lead to outliers having a greater impact on the metric. Conversely, MAE treats all errors uniformly, making it more robust to outliers. However, the MAE may be less sensitive to small changes in predicted values as it does not square the differences.

2.2.4 Summary of Packages Used

The research required several different software packages to be completed. These are given as follows:

1. Python 3.0 - Python will be used as data processing is simpler in python than compared to other languages. Additionally, many implementations of the mentioned machine learning models exist for python.
2. LazyRegressor - LazyRegressor is a Python package for quickly comparing the performance of various machine learning models on a given dataset.
3. PyCharm IDE - This will be used to develop the code on and will help to find errors.
4. Python packages - Various python packages will be used to facilitate the completion of the project. This will include but is not limited to multi-processing, scikit-learn and astroML python packages.
5. SQL - The SQL database programming language will be used to obtain data from the SDSS database

2.2.5 Ethical Considerations

The data used in the research is publicly available and does not involve human participants. Therefore, no ethical clearance is required for this research. The data sources used in the study are obtained from SDSS, which is widely accessible and has been previously used in academic research.

2.2.6 Overview of Implementation Process

The proposed methodology can be succinctly summarized as follows: Initially, the dataset will be obtained from the SDSS database, followed by pre-processing and training to establish a baseline performance. The optimal subset of features will be obtained using an optimization algorithm, and all the models will be retrained with this subset to obtain the best-performing model. The resulting models will undergo a comprehensive analysis, including hyperparameter tuning and statistical tests. An overview of the entire process is presented in Figure 2.5.

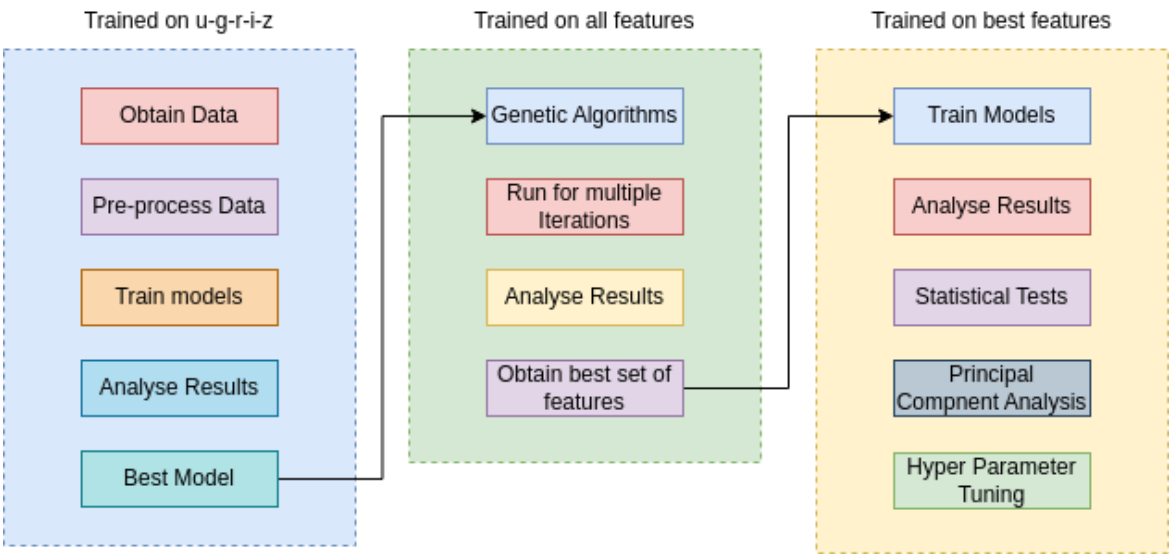

FIGURE 2.5: Diagram of Implementation: Figure displays a block diagram of the overall implementation of the project. It describes the workflow process which would be required to reproduce the research

Chapter 3

Results and Analysis

This chapter presents the findings and results of the proposed methodology detailed in Section 2.2. Furthermore, the chapter provides a thorough analysis of the model's performances, including their strengths and limitations, and discusses the implications of the findings for the research area. Lastly, the chapter suggests possible directions for future research based on the limitations of the chosen approach.

3.1 Results and Analysis

3.1.1 Data Analysis

A histogram is a visual representation of the distribution of a dataset, where data is divided into intervals or "bins" and the number of data points falling into each bin is displayed as a bar. It is a useful tool to gain insights into the shape of the distribution, including the presence of peaks and skewness. For instance, in Figure 3.1, the histogram shows that the distribution is not entirely Gaussian and is skewed. When a histogram shows that the distribution is not entirely Gaussian (bell-shaped) and is skewed, it suggests that the data does not follow a symmetric pattern and has a tail that extends more in one direction than the other. Skewness indicates that the data is not evenly distributed around the mean. Linear models assume that the relationship between the independent variables and the dependent variable is linear. Furthermore, These models assume that the residuals are normally distributed, meaning they follow a Gaussian distribution. As a result, when the data is skewed, the assumptions of linearity and normality may not hold, and linear models may not be able to accurately capture the underlying patterns in the data. To overcome this, more sophisticated machine learning techniques, such as ensemble methods

and neural networks may be more appropriate.

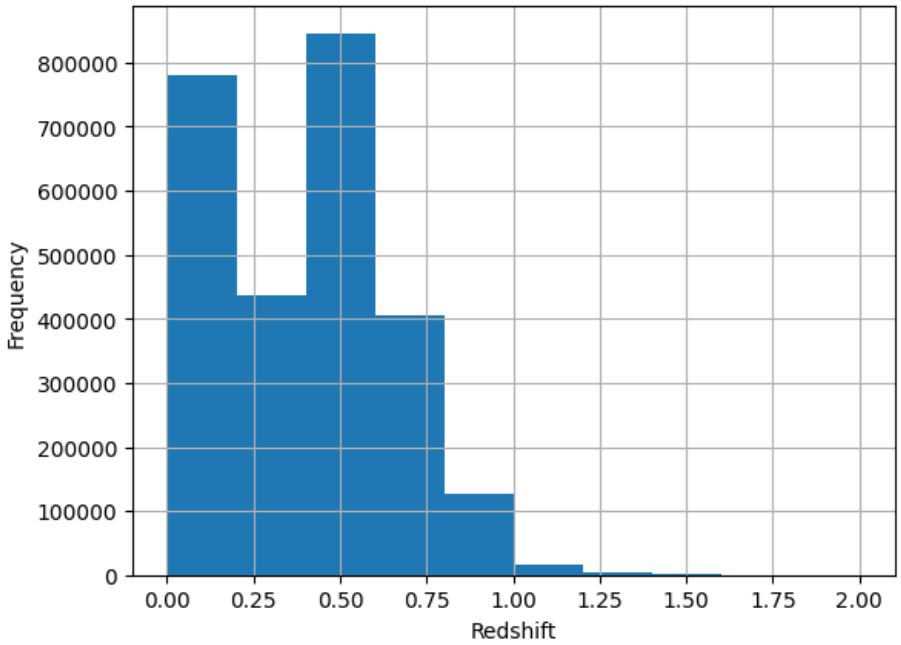

FIGURE 3.1: Histogram of Spectroscopic Redshift Data Obtained from SDSS Database

In addition to histograms, density plots are also a popular tool to visualize the distribution of a dataset. A density plot is a type of plot that uses a smooth curve to depict the distribution of data. Specifically, a density plot represents the probability density function (PDF) of the data, which shows the likelihood of the data being in a particular range of values. Figure 3.2 provides a density plot, which represents the distribution of the data as a smooth curve with 2 major peaks around the values 0.1 and 0.6. The plot indicates that the majority of the data lies within the 0-1 interval. Density plots are particularly useful for comparing distributions between different groups or subsets of a dataset and can provide more accurate comparisons of the shapes of the distributions while accommodating for differences in the scale or range of the data.

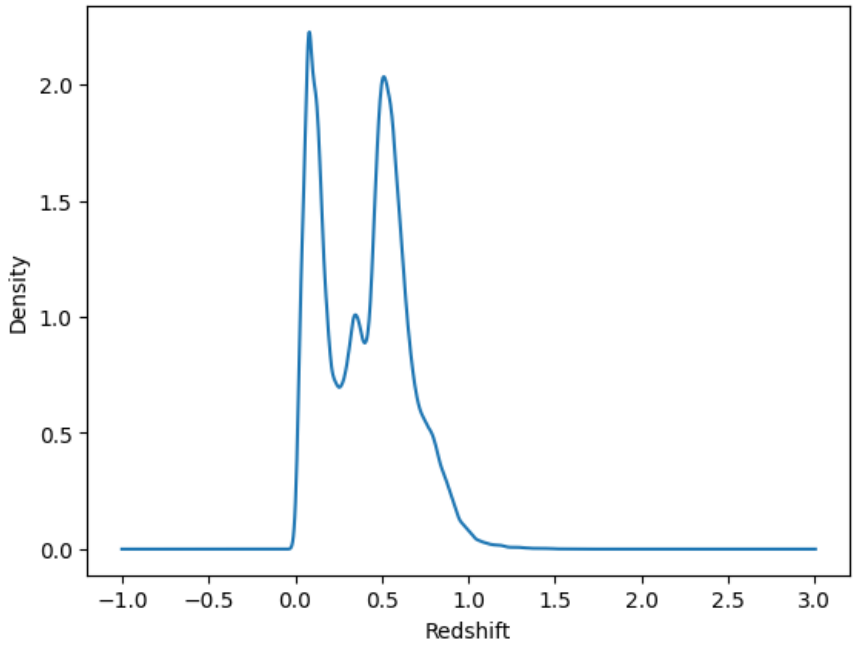

FIGURE 3.2: Density Plot of Redshfit Data

Moreover, Table 3.1 presents a statistical summary of observational redshift data retrieved from the SDSS Database, revealing that the range of values is relatively small. This characteristic of the data can sometimes make it difficult to obtain an accurate model, as small variations may be interpreted as noise. However, it is important to note that the complexity of the model should be commensurate with the complexity of the data. In this case, a simple model, such as linear regression, may be appropriate if the relationships between the variables are mostly linear. Alternatively, if the data is highly non-linear, more complex models such as decision trees or neural networks may be more suitable. Therefore, careful consideration of the data and the modeling approach is necessary to achieve accurate and reliable predictions.

Statistic	Description	Value
Count	represents the number of galaxies in the dataset	2,616,730
Mean	the average value of the redshift data for all the galaxies in the dataset	0.40
Standard Deviation	the amount of variation or dispersion in the redshift data	0.25
Minimum	represents the smallest value observed in the redshift data	0.00
25th Percentile	represents the value below which 25% of the data falls	0.16
50th Percentile	represents the middle value of the dataset	0.44
75th Percentile	represents the value below which 75% of the data falls	0.58
Maximum	represents the largest value observed in the redshift data	2.00

TABLE 3.1: Summary Statistics of Redshift Data Obtained from SDSS Database

Lastly, a correlation plot is provided in Figure 3.3. A correlation plot provides useful information on the relationships between the variables in the dataset. The plot displays the pairwise covariances between six variables, namely redshift, u, g, r, i, and z. The diagonal elements correspond to the correlation of each variable with itself, which have been standardized to equal one. On the other hand, the off-diagonal elements represent the correlation between other variables.

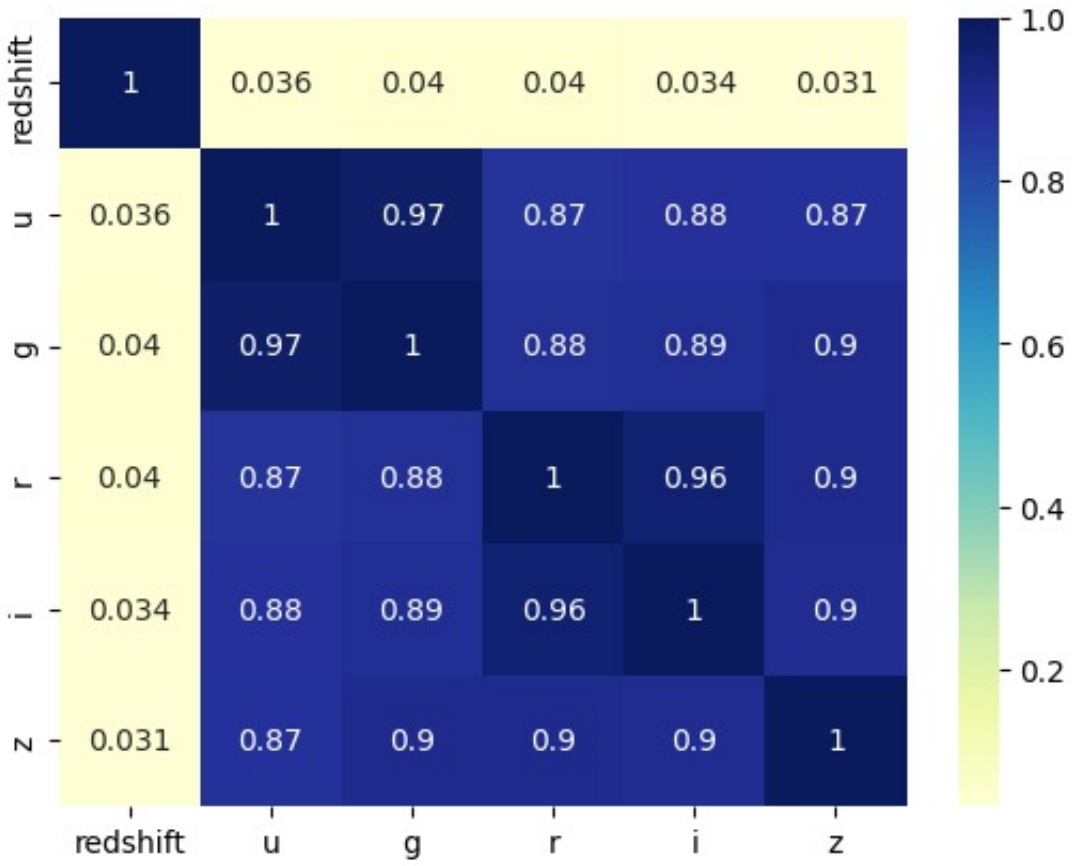

FIGURE 3.3: Correlation Plot of u-g-r-i-z Features Obtained from SDSS Dataset

Analysis of the correlation between redshift and the other variables reveals that they are relatively small, ranging from 0.03 to 0.04. This observation suggests that a weak linear relationship exists between redshift and the other variables. Such a result is consistent with the fact that redshift measures the expansion of the universe and is not directly related to the brightness or color of an astronomical object.

Conversely, the correlations between u, g, r, i, and z are relatively high, ranging from 0.87 to 0.97. This outcome indicates that a strong linear relationship exists between these variables. This observation is not unexpected since u, g, r, i, and z correspond to measures of the brightness of an astronomical object at different wavelengths, and their values are anticipated to be highly correlated.

The analysis suggests that redshift is weakly directly associated with the brightness or color of an astronomical object, whereas the measures of brightness at different wavelengths display a high level of correlation with each other. This information can be leveraged to select appropriate statistical methods for analyzing the data and to gain further insight into results.

3.1.2 Hyperparameter Tuning

Hyperparameter tuning is an essential process in machine learning that involves selecting the optimal set of hyperparameters for an algorithm to achieve better performance. Hyperparameters are parameters that are not learned during training but are set manually before the training process. The goal of hyperparameter tuning is to find the best combination of hyperparameters that will produce the most accurate model possible (Feurer and Hutter, 2019).

After conducting an extensive hyperparameter tuning process, the optimal hyperparameters for different machine learning algorithms have been determined based on their performance metrics. Table 3.2 presents the tuned hyperparameters for the algorithms: XGBoost, Random Forest, ANN, and KNN.

Parameters	XGBoost	Random Forest	ANN	KNN
max_depth	8	197	-	-
learning_rate	0.017	-	0.06	-
n_estimators	248	139	-	-
max_features	-	sqrt	-	-
n_neighbours	-	-	-	10
weights	-	-	-	uniform
algorithm	-	-	-	auto
first_layer_neurons	-	-	60	-
second_layer_neurons	-	-	80	-

TABLE 3.2: Best Parameters Found from Hyperparameter Tuning

In the optimization process for each algorithm, the three most crucial hyperparameters were carefully tuned to achieve the best performance. For XGBoost, the best max_depth hyperparameter was found to be 8, allowing the decision trees in the ensemble model to capture intricate patterns without over-fitting. The learning_rate

was fine-tuned to 0.017 to facilitate efficient convergence to the optimal solution.

Regarding the random forest algorithm, the max_features hyperparameter was assigned the value "sqrt," ensuring a diverse selection of features at each split and bolstering the model's overall robustness. The optimal number of n_estimators was determined to be 139, providing the best performance for the ensemble model.

In the case of ANN's, a learning_rate of 0.06 was found to be optimal. This provides a balance between convergence speed and avoiding overshooting the optimal weight values. The architecture was optimized by tuning the first_layer_neurons and second_layer_neurons hyperparameters, leading to the highest performance with 60 neurons in the first hidden layer and 80 neurons in the second hidden layer, allowing the network to capture complex patterns in the data effectively. Furthermore, it should be noted more complex network architectures possess a greater number of free weights, enabling them to achieve a closer fit to the data. However, in any real dataset, there exists a fundamental limit to the root mean square fit due to random noise present in the input measurements. Consequently, further increases in architecture complexity do not yield significant improvements beyond this limit. For the purpose of research, network architectures were investigated, resulting in the selection of a 4-layer architecture. Moreover, a comprehensive review of the existing literature informed this decision. Particularly, an observed instance of a four-layer architecture performing well in a study by Firth, Lahav, and Somerville (2003) served as a compelling reason for its inclusion in the investigation.

Lastly, for the KNN algorithm, the optimal value for n_neighbours was found to be 10, meaning the model considered the ten nearest data points for classification. This provides a suitable balance between under-fitting and over-fitting, ensuring reliable results. The optimal weights parameter was found to be "uniform," granting equal influence to all data points within the neighborhood for the classification decision.

3.1.3 Base model

Based on the methodology outlined in Section 2.2, the initial stage of the study involved utilizing the base features of u-g-r-i-z. This approach was then utilized to train the models, with the results obtained displayed in Table 3.3. By employing these features, the models were able to effectively generate meaningful data and yield useful insights, providing a foundation for further exploration and analysis. The hardware used for training the models consists of a Mid 2012 MacBook Pro with a 2.5 GHz Dual-Core Intel Core-i5 that runs on MAC OS Catalina version 10.15.3.

Model	R^2	RMSE	MAPE	MAE	MSE	Time (s)
XGBoost	0.84	0.05	20.89%	0.03	0.002	167
Random Forest	0.84	0.05	22.03%	0.03	0.002	221
KNN	0.81	0.05	22.79%	0.03	0.002	784
ANN	0.84	0.05	22.45%	0.03	0.002	238
Polynomial Regression	0.006	0.12	52.02%	0.1	0.016	443

TABLE 3.3: Performance metrics of various machine learning models

The table presents the performance metrics of the five machine learning models: XGBoost, random forest, KNN, ANN, and polynomial regression. The models are evaluated based on several performance metrics, including R-squared, Root Mean Square Error, Mean Absolute Percentage Error, Mean Absolute Error, Mean Square Error, and Time (in seconds) required for each model to generate its predictions.

The R-squared metric measures how much of the variation in the dependent variable is explained by the models. XGBoost, random forest, and ANN models achieved similar R-squared values of 0.84, while KNN had a slightly lower value of 0.81. Polynomial regression had the lowest R-squared value of only 0.006, indicating a poor fit to the data.

The RMSE metric measures the average difference between the predicted and actual values of the dependent variable. XGBoost, random forest, ANN and KNN models achieved similar RMSE values of 0.05, indicating good performance in terms of accuracy. Polynomial regression had the highest RMSE value of 0.12, indicating poor

performance.

Additional metrics included the MAPE, MAE and MSE scores. XGBoost model achieved the best performance with a MAPE of 20.89%. Random forest and ANN models had slightly higher values, while KNN had a value of 22.79%. Polynomial regression had a very high MAPE of 52.02%. The metrics all suggest poor performance from polynomial regression and good performance for XGBoost and Random Forest algorithms.

The Time metric measures the time required for each model to generate its predictions. XGBoost was the fastest model, requiring only 167 seconds, while the other models required significantly longer times. Random forest required 221 seconds, KNN required the longest time of 784 seconds, ANN required 238 seconds, and polynomial regression required 443 seconds. The observed extended training time for the KNN model can be attributed to its unique characteristic of using all available values during the calculation process. Consequently, this direct reliance on data points for computation leads to a more time-consuming training phase compared to other models that learn and generalize patterns from the data.

XGBoost, random forest and ANN models are the best performers in terms of the metrics evaluated. The KNN model has similar performance but takes significantly longer to train. The polynomial regression model has poor performance. From analyzing Table 3.3, it was observed that the algorithm which yielded the fastest and most optimal results for the initial feature set was XGboost. Based on this outcome, it was decided that the XGboost algorithm, in conjunction with genetic algorithms, would be utilized in subsequent stages of the implementation.

Lastly, Figure 3.4 exhibits a comparative representation of the redshift values obtained from the XGBoost algorithm against the observational values. The plot demonstrates a close proximity between the predicted and observed values.

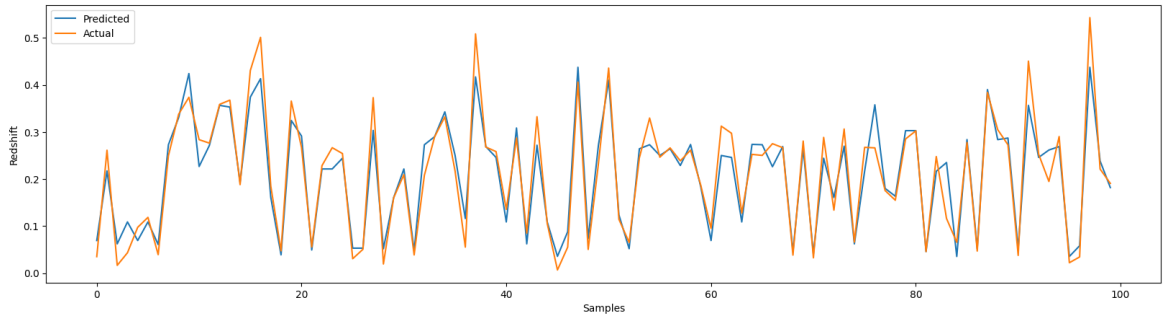

FIGURE 3.4: Obervsational vs Predicted Redshift Values for the XGBoost Algorithm Trained on Base Subset of Features

3.1.4 Genetic Algorithms

In this research, Genetic Algorithms were utilized to optimize the performance of the XGBoost algorithm by identifying the most suitable subset of features - features that provide the most accurate redshift values. The fitness evaluation of the models in each generation and population was based on the MAPE score. Figure 3.5 presents the results obtained from the GA iterations for the fitness function, illustrating the algorithm's progress in optimizing the feature subset.

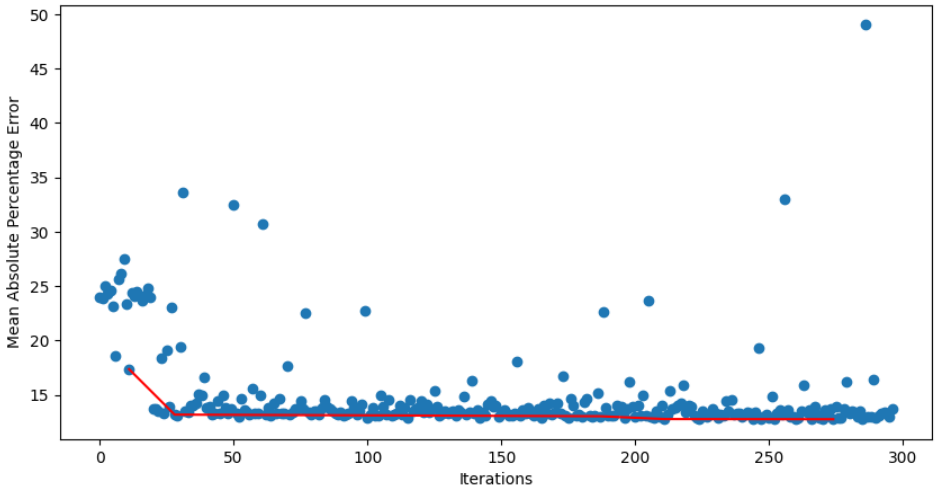

FIGURE 3.5: MAPE Values Obtained Over Various Iterations for Genetic Algorithms Optimization

The provided visualization showcases the results of the performed analysis, where the x-axis represents the number of iterations while the y-axis illustrates the associated error values. Each point on the plot corresponds to a specific iteration and its corresponding error value. The red line depicted in the graph indicates the lowest errors that have been achieved throughout the optimization process.

Upon careful examination of the plot, it is apparent that the algorithm reaches convergence around the 12-13% threshold, with no further significant improvement observed beyond this point. This behavior is indicative of the algorithm having identified an optimal solution within the search space. Further iterations may result in negligible improvements or even potential over-fitting of the model.

Additionally, to obtain a more comprehensive understanding of the optimized model, the features corresponding to the lowest errors obtained have been listed in Table 3.4. Specifically, the rows of Table 3.4 contain feature subsets that emerged as the most optimal during different iterations of the optimization process. These subsets were obtained by subjecting the validation set to the genetic algorithm optimization. Notably, the variables enlisted in Table 3.4 are derived from the u-g-r-i-z components. The results reveal that the algorithm recognizes diverse combinations of these components as the best-performing features.

Features	MAPE	Count
[dec psfMag-g psfMag-z psfMagErr-u psfMagErr-g psfMagErr-i psfMagErr-z fiberMag-u fiberMag-g fiberMag-i fiberMagErr-u fiberMagErr-g fiberMagErr-i petroMag-u petroMagErr-g petroMagErr-r petroMagErr-z modelMag-u modelMag-i modelMag-z modelMagErr-r modelMagErr-i cModelMag-u cModelMag-g cModelMag-z cModelMagErr-u cModelMagErr-g cModelMagErr-z score extinction-r extinction-i extinction-z dered-g dered-i dered-z]	17.37	35
[psfMag-r psfMagErr-u psfMagErr-z fiberMag-u fiberMag-g fiberMag-r fiberMag-z fiberMagErr-r fiberMagErr-i petroMag-u petroMag-i petroMagErr-g petroMagErr-i petroMagErr-z modelMag-u modelMag-r modelMag-i modelMag-z modelMagErr-g modelMagErr-z cModelMag-u cModelMag-g cModelMag-i cModelMag-z cModelMagErr-u cModelMagErr-r extinction-g extinction-r extinction-i dered-g dered-i dered-z]	13.19	32
[ra psfMag-z psfMagErr-u psfMagErr-r psfMagErr-z fiberMag-u fiberMag-g fiberMag-r fiberMag-i fiberMag-z fiberMagErr-g fiberMagErr-i petroMag-g petroMag-z petroMagErr-u petroMagErr-g modelMag-u modelMag-i modelMagErr-g modelMagErr-r modelMagErr-z cModelMag-r cModelMag-i cModelMag-z cModelMagErr-u cModelMagErr-g cModelMagErr-i cModelMagErr-z mRrCc-r extinction-u extinction-z dered-u dered-g dered-z]	13.04	34
[ra psfMag-r psfMag-i psfMagErr-u psfMagErr-r psfMagErr-z fiberMag-u fiberMag-g fiberMag-r fiberMag-i fiberMag-z fiberMagErr-g fiberMagErr-i petroMag-g petroMag-z petroMagErr-u modelMag-u modelMag-i modelMagErr-g modelMagErr-r modelMagErr-z cModelMag-i cModelMag-z cModelMagErr-u cModelMagErr-g cModelMagErr-i cModelMagErr-z mRrCc-r extinction-z dered-u dered-g dered-z]	12.88	32
[ra psfMag-r psfMag-i psfMagErr-u psfMagErr-r psfMagErr-z fiberMag-u fiberMag-g fiberMag-r fiberMag-i fiberMag-z fiberMagErr-g petroMag-g petroMag-z petroMagErr-u modelMag-u modelMag-i modelMagErr-g modelMagErr-r cModelMag-i cModelMag-z cModelMagErr-u cModelMagErr-g cModelMagErr-i cModelMagErr-z mRrCc-r extinction-z dered-u dered-g dered-z]	12.78	30
[ra psfMag-r psfMag-i psfMagErr-u psfMagErr-r psfMagErr-z fiberMag-g fiberMag-r fiberMag-i fiberMag-z fiberMagErr-g fiberMagErr-i petroMag-g petroMag-z petroMagErr-u modelMag-g modelMag-i modelMagErr-g modelMagErr-r modelMagErr-z cModelMag-i cModelMagErr-u cModelMagErr-g cModelMagErr-i cModelMagErr-z mRrCc-r dered-u dered-g dered-r]	12.77	29

TABLE 3.4: Best Features Obtained Through Genetic Algorithms

Moreover, it is crucial to emphasize the rationale behind selecting the initial features. The approach involved incorporating all available information from the SpecPhoto table and allowing the optimization algorithm to discern the most pertinent features. While certain features align with those observed in the literature, such as the dered, extinction, and Petrosian values (found in works by Li et al. (2007)), others do not. Nevertheless, it was deemed more fitting to let the optimization algorithm undertake the feature selection process, thus handling the task more comprehensively.

Analysis of Table 3.4 reveals that despite the incorporation of a greater number of features in some iterations, there is no direct improvement in the resulting score. To comprehensively evaluate the effectiveness of the employed Genetic Algorithms method, a principal component analysis (PCA) has been conducted to compare the performance of the best set of features.

The PCA analysis allows for the identification of the most important features that contribute the most to the observed variance in the dataset. By comparing the best feature sets with the aid of PCA, it is possible to determine the extent to which the genetic algorithm approach has optimized the feature selection process. The findings from this analysis may aid in providing insights into the performance and limitations of the optimization methodology.

3.1.5 Analysis of Features

According to Table 3.4, the Genetic Algorithm optimization selected 29 features that resulted in the best score of 12.77%. This feature set is a combination of various attributes from the SpecPhoto table, including the variation of u-g-r-i-z for different attributes. A correlation plot (Figure 3.6) of all features was generated to further investigate this matter.

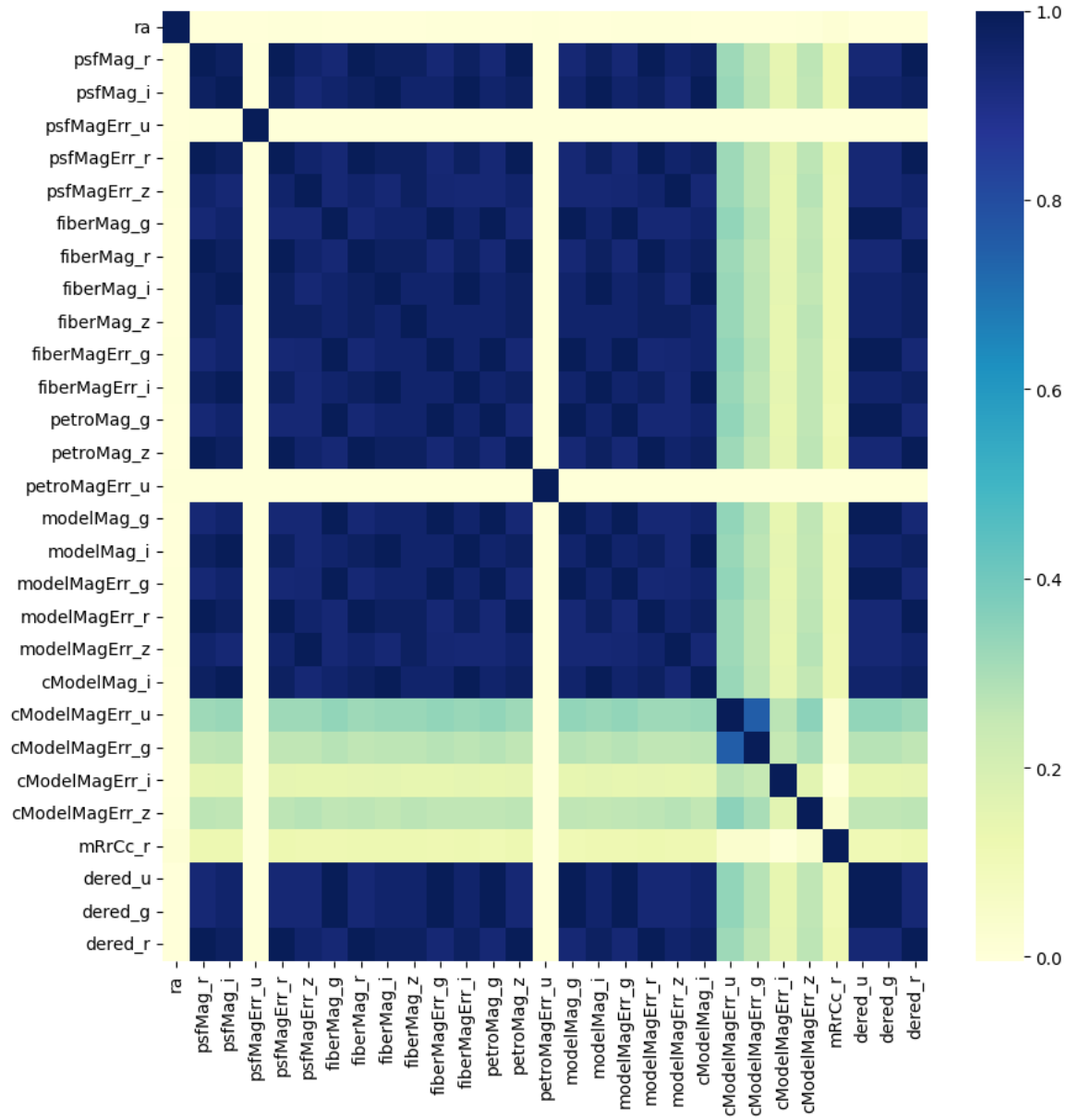

FIGURE 3.6: Correlation Plot of the Best Features Obtained From Genetric Algorithms

The correlation plot reveals that most variables exhibit a strong positive correlation with each other. For instance, psfMag-r and fiberMag-r exhibit a perfect positive linear relationship with a correlation coefficient of 1.00, while psfMag-r and psfMag-i display a strong positive linear relationship with a correlation coefficient of 0.98. The omission of all bands (u-g-r-i-z) for each attribute is due to the high correlation

between them, which does not contribute significantly to the models. Highly correlated data can result in over-fitting and reduce the model's ability to generalize to new data.

Including two highly correlated variables in a model may lead to over-fitting, where the model becomes excessively complex and fits the training data too closely, resulting in poor performance on new data. The effectiveness of the genetic algorithm optimization is demonstrated by its ability to remove sets of features with high correlation, as observed from the correlation plot.

Moreover, the literature supports the significance of these variables within the specific domain of predicting photometric redshifts. This outcome is expected within the astronomy. The methodology employed in this report suggests promising avenues for further investigation. Additionally, it was initially considered impractical to include the variables RA and DEC in future redshift estimations. The rationale behind this lies in the clustering of galaxies, as algorithms identify similarities in redshift values among galaxies within clusters, leading to the inclusion of RA and DEC in the optimization process.

3.1.6 Best Models

Table 3.5 presents the results of the best models obtained. After training the models on the optimal subset of features, the results demonstrate that XGBoost outperforms the other models with the highest R^2 value of 0.94, and the lowest RMSE and MAE values of 0.03 and 0.02, respectively. Additionally, XGBoost requires the least amount of time to train. Therefore, it can be concluded that XGBoost is the best model for this dataset.

Model	R^2	RMSE	MAPE	MAE	MSE	Time (s)
XGBoost	0.94 ± 0.01	0.03 ± 0.01	$12.04\% \pm 2\%$	0.02 ± 0.01	0.001 ± 0.0005	367 ± 28
Random Forest	0.93 ± 0.01	0.03 ± 0.01	$14.93\% \pm 2.5\%$	0.02 ± 0.01	0.002 ± 0.001	483 ± 180
KNN	0.90 ± 0.02	0.04 ± 0.02	$17.91\% \pm 2.5\%$	0.02 ± 0.01	0.002 ± 0.001	1660 ± 435
ANN	0.89 ± 0.02	0.06 ± 0.03	$18.12\% \pm 2.5\%$	0.03 ± 0.02	0.004 ± 0.002	665 ± 137
Polynomial Regression	0.80 ± 0.05	0.1 ± 0.05	$34.43\% \pm 7\%$	0.03 ± 0.02	0.01 ± 0.005	443 ± 172

TABLE 3.5: Performance metrics of various machine learning models (including variances)

However, it is important to note that the random forest algorithm also performs well with RMSE and MAE values of 0.03 and 0.002, respectively, which are quite close to the values of XGBoost. Additionally, the RMSE of XGBoost only beats the Random Forest MAPE score by 2.89%. Thus, it is necessary to conduct statistical tests to confirm whether XGBoost is statistically superior to Random Forest.

Figure 3.7 exhibits a comparative representation of the redshift values obtained from the XGBoost algorithm against the observational values. The plot demonstrates a close proximity between the predicted and observed values.

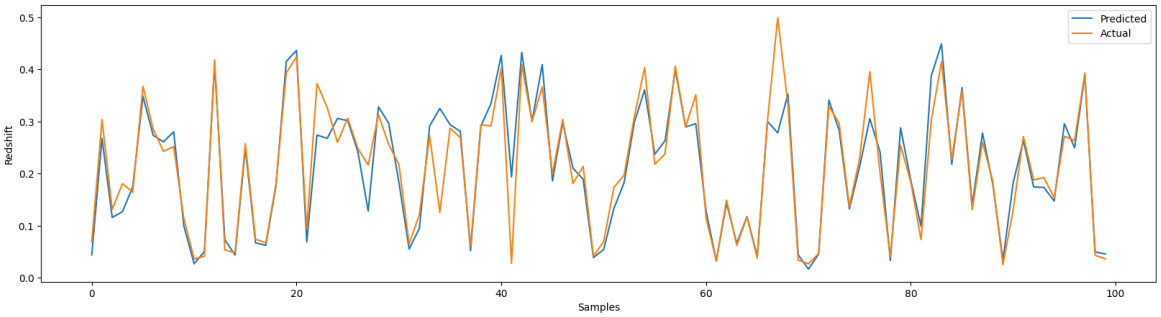

FIGURE 3.7: Observational vs Predicted Redshift Values for XGBoost algorithm trained on Optimal Features

Figure 3.8 presents a histogram of errors, with an overlaid Gaussian distribution fit. Gaussian errors, also known as normally distributed errors or errors with a normal distribution, are a prevalent type of error distribution observed in machine learning models. The majority of errors in such models are small and near to zero, while larger errors occur less frequently and become increasingly improbable as they deviate farther from zero.

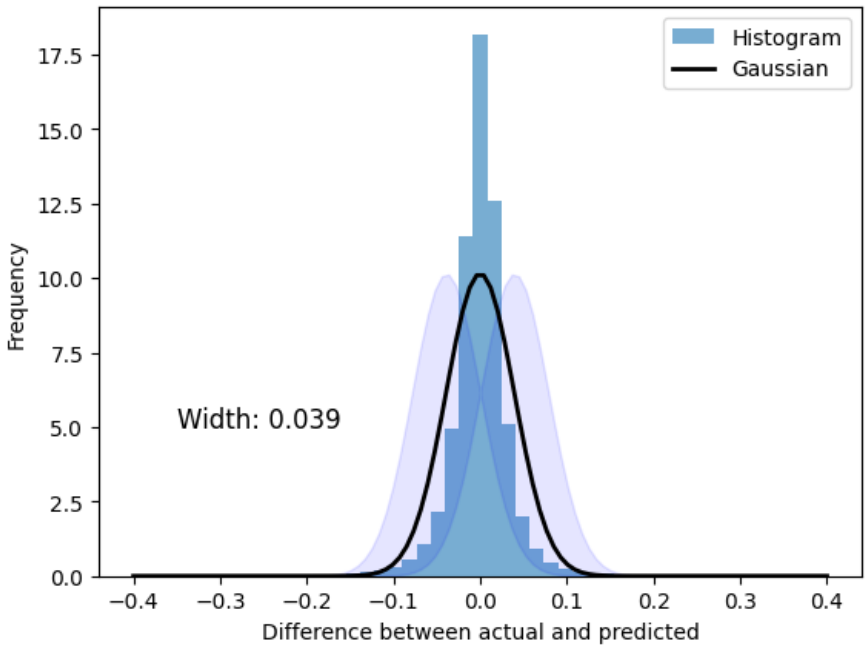

FIGURE 3.8: Histogram of Errors Between Observational and Predicted Redshift Values

In the realm of machine learning, Gaussian errors hold several vital implications. First and foremost, the Gaussian nature of the errors implies that the model is making consistent errors, centered around zero. This is desirable because it indicates that the model is not biased systematically in any particular direction.

Moreover, the Gaussian errors assist in comprehending the model's overall performance. If the errors are significantly skewed toward one direction or another, it may suggest that the model is over-fitting or under-fitting the data, respectively. Conversely, if the errors are symmetrical and bell-shaped, it indicates that the model is

fitting the data well and making accurate predictions. It is seen from the plot, that the width of the Gaussian is 0.03. The width of the Gaussian distribution can be interpreted as a measure of the variability of the prediction errors and is formally known as standard deviation. A standard deviation of 0.03 means that the majority of the differences between the actual and predicted values fall within a range of +/- 0.03 units around the mean difference. This indicates that the model is generally making predictions that are close to the actual values.

The results of the actual vs predicted values are presented in Figure 3.9, which displays a two-dimensional histogram. The almost straight line in the plot signifies a strong correlation between predicted and actual values, suggesting that the model can accurately capture the underlying data patterns. This correlation is a desirable outcome in many machine learning applications, as it indicates good predictive power of the model. However, it is essential to note that a high correlation alone does not necessarily imply that the model is the best fit for the data. Other performance metrics, such as mean squared error and R^2, should also be taken into account while assessing the model's performance.

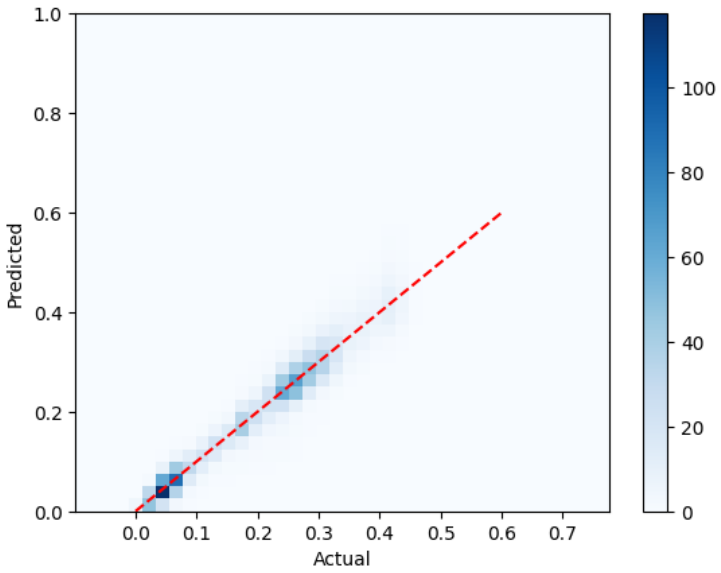

FIGURE 3.9: 2D Histogram of Observational vs Predicted Redshift Values

Additionally, the plot reveals that the predicted and observational values become broader at higher redshifts, which suggests that the error is greater at higher redshifts. This could be attributed to the fact that the majority of the data is concentrated at lower redshifts. It is also apparent that there is no systematic shift at some redshifts. This is an important observation as it indicates that the model is not biased towards specific redshift values and is capable of producing reasonably accurate predictions across the entire range of redshift values.

Comparing the results obtained in Table 3.5 with those from the existing literature revealed that the XGBoost model employed in this study exhibited superior performance over the models reported in the Henghes et al. (2021). Additionally, the other models utilized in this project also outperformed their counterparts from the literature. Specifically, the random forest model in the literature had a MSE of 0.004 and MAE of 0.03, while the random forest model in this project achieved an MSE of 0.002 and MAE of 0.02. This substantial improvement in performance underscores the efficacy of the work conducted in this research.

It is essential to consider certain factors that might contribute to these differences. Firstly, the number of galaxies used in the literature's experiments was notably lower, with only 1,639,348 galaxies, while this research utilized a more extensive dataset of 2,616,730 galaxies. This larger dataset could potentially enable more accurate model training and lead to better predictions. Moreover, the literature's comparison was based solely on the base features u-g-r-i-z, whereas this research delved into exploring additional features. The incorporation of these additional features might have played a crucial role in enhancing the model's predictive capabilities and contributing to the observed improvements. These findings underscore the importance of dataset size and feature selection when striving to improve the performance of machine learning models in astrophysical research.

3.1.7 PCA Analysis

Principal Component Analysis (PCA) is a powerful technique in data analysis that aims to reduce the dimensionality of a large dataset while retaining most of the important information. It is a linear transformation technique that identifies the underlying structure in the data and produces a new set of variables, called principal components, that are uncorrelated with each other. PCA can be used for dimensionality reduction by selecting a subset of the principal components that capture the most variance in the data. The number of principal components to keep is a hyperparameter that can be tuned based on the desired level of variance retention (Maćkiewicz and Ratajczak, 1993).

Initially, the PCA algorithm begins by computing the covariance matrix of the data. The covariance matrix measures the linear dependence between pairs of variables in the data. The covariance between two variables x and y can be computed using the following formula:

$$cov(x, y) = E[(x - E[x])(y - E[y])] \tag{3.1}$$

where $E[x]$ and $E[y]$ are the means of x and y, respectively. The covariance matrix is a square matrix where each element (i, j) is the covariance between the i-th and j-th variables in the data.

Thereafter, PCA computes the eigenvectors and eigenvalues of the covariance matrix. An eigenvector is a non-zero vector that, when multiplied by the covariance matrix, gives a scalar multiple of itself. The scalar multiple is called the eigenvalue. The eigenvectors and eigenvalues are computed using the following equation:

$$Cv = \lambda v \tag{3.2}$$

where C is the covariance matrix, v is the eigenvector, and λ is the eigenvalue. The eigenvectors and eigenvalues are ordered by the magnitude of the eigenvalues, with the largest eigenvalue corresponding to the first principal component.

The eigenvectors obtained form a new coordinate system, where each axis represents a principal component. The first principal component is the direction of

maximum variance in the data, the second principal component is the direction of maximum variance orthogonal to the first principal component, and so on. The number of principal components is equal to the number of variables in the data (Karamizadeh et al., 2013).

PCA then projects the data onto the new coordinate system. The projection of a data point x onto the k-th principal component is given by:

$$z_k = v_k^T(x - \mu) \tag{3.3}$$

where v_k is the k-th eigenvector, μ is the mean of the data, and z_k is the k-th principal component. The projection of the data onto the new coordinate system results in a new representation of the data that captures the most important patterns and structures. Figure 3.10 provides a plot of the PCA analysis of the dataset.

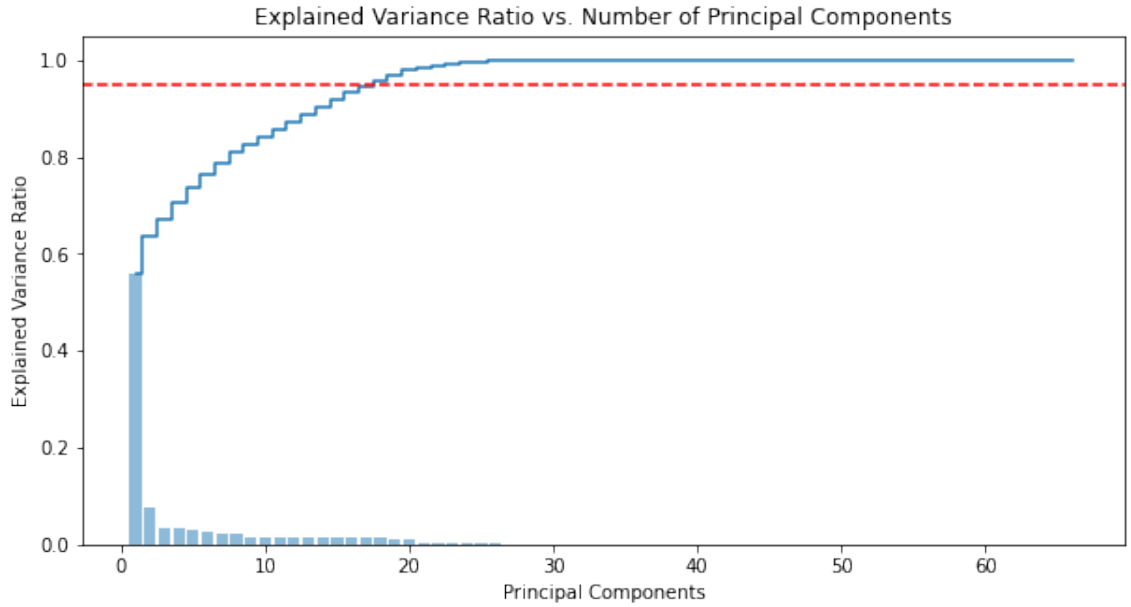

FIGURE 3.10: Variance Ratio vs Principal Component: Figure represents the number of principal components it would require to explain the variance in the data

From the plot, it can be observed that the first principal component (PC1) explains up to 60% of the variance, which is the highest among all the principal components.

Additionally, it can be seen that it takes 26 principal components to explain all the variance and 18 principal components to explain up to 95% of the variance. This information is useful in determining the number of principal components needed to retain a certain percentage of the variance and to perform dimensionality reduction.

A XGBoost Regressor was trained on the data transformed by using the PCA, and the subsequent results of the regression analysis are presented below in Table 3.6:

Model	R^2	RMSE	MAPE	MAE	MSE	Time (s)
XGBoost	0.89	0.04	18.03%	0.02	0.001	154

TABLE 3.6: XGBoost trained on data transformed by PCA

Table 3.6 highlights that the highest MAPE score achieved is 18.03%, which is noticeably higher than the 12% attained by the XGBoost model trained on the best subset of features. This finding provides compelling evidence that the genetic algorithm optimization utilized in the research design is effective in identifying the relevant features for model training.

3.1.8 Statistical Tests

The Wilcoxon signed-rank test is a statistical test used to compare two related samples that cannot be assumed to follow a normal distribution. It is a non-parametric test, meaning that it does not require assumptions about the shape of the underlying distribution of the data. This makes it a useful alternative to the paired t-test, which is used when the data follows a normal distribution (Woolson, 2007).

The Wilcoxon signed-rank test works by ranking the absolute differences between the two related samples and then calculating the sum of the ranks of the positive differences and the sum of the ranks of the negative differences. The null hypothesis of the test is that there is no difference between the two samples, while the alternative hypothesis is that there is a significant difference between the two samples.

The test statistic for the Wilcoxon signed-rank test is calculated using the following formula:

$$W = \sum_{i=1}^{n} R_i \times sign(D_i) \tag{3.4}$$

The test statistic W is defined by the following variables: n, the sample size; R_i, the rank of the i^{th} absolute difference; D_i, the i^{th} absolute difference between the two related samples; and $sign(D_i)$, which represents the sign of the i^{th} absolute difference. The calculation of W measures the magnitude of the difference between the two related samples, while the p-value measures the likelihood of observing the test statistic if the null hypothesis is true. A p-value less than the chosen significance level indicates that the null hypothesis should be rejected in favor of the alternative hypothesis(Rey and Neuhäuser, 2011).

The p-value for the test statistic is calculated using the appropriate distribution for the test. If the sample size is less than 50, the p-value can be found using a table of critical values. If the sample size is larger than 50, a normal approximation can be used to calculate the p-value.

For the purpose of the research the two best performing algorithms are compared, namely, random forest and XGBoost. To ensure that the analysis is reliable and accurate, each algorithm is run 10 times with a new train/test split generated each time, which constitutes the sample size. The results of these repetitions were then used to conduct a Wilcoxon signed rank test at a significance level of 0.05, which is a statistical test that compares two sets of paired data.

Upon computation, it was discovered that the sum of positive ranks was 4, while the sum of negative ranks was 51. These values were used to compute the W value, which was found to be 4. To determine if this result was significant, the critical value for an iteration count of N=10 was obtained from a table of critical values, and it was found to be 8. As the W value was less than the critical value, it was concluded that the null hypothesis can be rejected.

The statistical analysis conducted indicates a significant difference between the performance of the XGBoost algorithm and the random forest algorithm. This finding

is crucial in determining the most efficient algorithm for the task of estimating redshift values. The results suggest that the XGBoost algorithm is superior to the random forest algorithm in terms of predictive accuracy, as measured by the evaluation metric used in this study.

The conclusion drawn from this study is that the XGBoost algorithm is more effective in producing accurate predictions than the Random Forest algorithm. It is important to note that the difference in performance may vary based on the specific dataset and evaluation metric used. Therefore, caution should be exercised when applying these findings to other datasets or tasks.

3.1.9 Future Recommendations

The inclusion of future recommendations in this report is essential as it enables the identification of areas where further research can be conducted to advance the current knowledge base. These recommendations can provide a pathway for researchers and practitioners to address the gaps in the existing literature and expand upon the findings of this study.

The work in this report was confined to the SpecPhoto table from the SDSS database. Future work should consider investigating additional tables that could provide valuable insights. For instance, the PhotoObjAll table contains multiple features that were not explored in this study. One of these features is the airmass attribute, which measures the amount of atmosphere an object is observed through. Airmass is a crucial factor in astronomy, as objects observed at higher air masses are generally of lower quality due to increased light scattering. The inclusion of airmass information in machine learning models can improve their predictive accuracy (Graham et al., 2018).

Moreover, the hyperparameter tuning in this research was limited as it was not the primary research objective. Further studies should conduct comprehensive hyperparameter tuning to optimize the model's performance. Additionally, to extend the sample size for various redshift values, other datasets besides the SDSS database

can be utilized.

Lastly, this study focused solely on estimating redshifts of galaxies. Future works should consider estimating the redshifts of other celestial objects to broaden the scope of the research.

3.2 Summary

The presented research showcases the performance of the XGBoost model, which achieved a low MAPE of 12.04%. To further evaluate the model, an analysis of its features and optimization algorithm was conducted, and basic hyperparameter tuning was performed. The study also included correlation plots and statistical tests to confirm that XGBoost was the best-performing model. These findings provide valuable insights for future research in this field, highlighting the effectiveness of XGBoost in predictive modeling.

Chapter 4

Conclusion

In conclusion, this research builds upon existing literature by using photometric data to estimate the redshift of galaxies, which has not been extensively explored in previous studies. The study utilized a variety of ML-based models, obtained data from the Sloan Digital Sky Survey database via SQL queries, and performed feature selection to identify the most relevant features for redshift estimation. The developed models were trained, optimized, and compared using various performance metrics such as RMSE, MAE, MSE, and MAPE. The results indicate that the XGBoost algorithm is the most effective model for estimating redshift, achieving a remarkable MAPE score of 12.04%. The findings presented in this report hold considerable promise for advancing redshift studies and facilitating more accurate simulations of the universe. By successfully identifying the most suitable algorithm and features for redshift estimation, this research offers valuable insights that can significantly benefit future investigations in this field.

Appendix A

Appendix Title

A.1 Query for Data Retrieval

The following functions have been used to query the SDSS database.

```
def GetTable(z_min = None, z_max = None, dataRelease=SkyServer_DataRelease) :

    #select
    query = "select z,zErr,ra,dec,psfMag_u,psfMag_g,psfMag_r,psfMag_i,
    psfMag_z,psfMagErr_u,psfMagErr_g,psfMagErr_r,psfMagErr_i,
    psfMagErr_z,fiberMag_u,fiberMag_g,fiberMag_r,fiberMag_i,fiberMag_z,
    fiberMagErr_u,fiberMagErr_g,fiberMagErr_r,fiberMagErr_i,fiberMagErr_z,
    petroMag_u,petroMag_g,petroMag_r,petroMag_i,petroMag_z,petroMagErr_u
    ,petroMagErr_g,petroMagErr_r,petroMagErr_i,petroMagErr_z,modelMag_u,
    modelMag_g,modelMag_r,modelMag_i,modelMag_z,modelMagErr_u,modelMagErr_g,
    modelMagErr_r,modelMagErr_i,modelMagErr_z,cModelMag_u,cModelMag_g,cModelMag_r,
    cModelMag_i,cModelMag_z,cModelMagErr_u,cModelMagErr_g,cModelMagErr_r,
    cModelMagErr_i,cModelMagErr_z,mRrCc_r,mRrCcErr_r,score,extinction_u,
    extinction_g,extinction_r,extinction_i,extinction_z,dered_u,dered_g,dered_r,
    dered_i,dered_z"

    #from
    query += " from SpecPhoto"

    #where
    query += " where zWarning = 0 and class = 'GALAXY'"
```

```
    query += " and z between {}".format(z_min)


    query += " and {}".format(z_max)

    print(query)

    return SkyServer.sqlSearch(sql= query, dataRelease=SkyServer_DataRelease)

z_min = 0
z_step = 0.1

table = None

while z_min<2.1 :
    t = GetTable(z_min, z_min + z_step)

    if t.shape[0] == 0 :

        print ('No galaxies at z >= {}'.format(z_min))


    if t.shape[0] == 500000 :

        print ('Limit on rows exceed for z in [{}, {})'.format(z_min, z_min+z_step))

    if table is not None :
        table = table.append(t)
    else :
        table = t
```

```
z_min += z_step
```

Bibliography

Ellis, George FR (1984). "Alternatives to the big bang". In: *Annual review of astronomy and astrophysics* 22, pp. 157–184.

Terr, David (2013). "The Big Bang Theory". In: *Citeseer*, pp. 1–5.

Thompson, Bert, Brad Harrub, and Branyon May (2003). "The Big Bang Theory—A Scientific Critique [Part 1]". In: *Reason & Revelation* 23.5, pp. 32–34.

Enrera, Rochelle (2021). "Evidence of the Big Bang Theory". In: URL: https://study.com/academy/lesson/evidence-for-the-big-bang-theory-background-radiation-red-shift-and-expansion.html.

Bahcall, Neta A (2015). "Hubble's Law and the expanding universe". In: *Proceedings of the National Academy of Sciences* 112.11, pp. 3173–3175.

Fox, Karen C (2002). *The big bang theory: What it is, where it came from, and why it works*. John Wiley & Sons.

Lehnert, Matthew D et al. (2010). "Spectroscopic confirmation of a galaxy at redshift z= 8.6". In: *Nature* 467.7318, pp. 940–942.

"Redshift" (2020). In: URL: https://lco.global/spacebook/light/redshift/.

Baryshev, Yurij et al. (2012). "Cosmological Redshift and the Distance Scale". In: *Fundamental Questions of Practical Cosmology: Exploring the Realm of Galaxies*, pp. 69–89.

Brammer, Gabriel B, Pieter G van Dokkum, and Paolo Coppi (2008). "EAZY: a fast, public photometric redshift code". In: *The Astrophysical Journal* 686.2, p. 1503.

Mathuriya, Amrita et al. (2018). "CosmoFlow: Using deep learning to learn the universe at scale". In: *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, pp. 819–829.

Pasquet, Johanna et al. (2019). "Photometric redshifts from SDSS images using a convolutional neural network". In: *Astronomy & Astrophysics* 621, A26.

Wang, Dan, Yan-xia Zhang, and Yong-heng Zhao (2008). "Predicting photometric redshifts with polynomial regression". In: *Advanced Software and Control for Astronomy II*. Vol. 7019. SPIE, pp. 1105–1114.

Momtaz, Aidin, Mohammad Hossein Salimi, and Soroush Shakeri (2022). "Estimating the Photometric Redshifts of Galaxies and QSOs Using Regression Techniques in Machine Learning". In: *arXiv preprint arXiv:2201.04391*.

De Wei, Kenny Chong and Abel Yang (2019). "Photometric redshift analysis using supervised learning algorithms and deep learning". In: *EPJ Web of Conferences*. Vol. 206. EDP Sciences, p. 09006.

Zhang, Yanxia et al. (2020). "Machine Learning for Photometric Redshift Estimation of Quasars with Different Samples". In: *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*. IEEE, pp. 294–297.

Schuldt, S et al. (2021). "Photometric redshift estimation with a convolutional neural network: NetZ". In: *Astronomy & Astrophysics* 651, A55.

Hoyle, Ben et al. (2015). "Data augmentation for machine learning redshifts applied to Sloan Digital Sky Survey galaxies". In: *Monthly Notices of the Royal Astronomical Society* 450.1, pp. 305–316.

D'Isanto, Antonio and Kai Lars Polsterer (2018). "Photometric redshift estimation via deep learning-generalized and pre-classification-less, image based, fully probabilistic redshifts". In: *Astronomy & Astrophysics* 609, A111.

Henghes, Ben et al. (2021). "Benchmarking and scalability of machine-learning methods for photometric redshift estimation". In: *Monthly Notices of the Royal Astronomical Society* 505.4, pp. 4847–4856. DOI: 10.1093/mnras/stab1513. URL: https://doi.org/10.1093%2Fmnras%2Fstab1513.

Smola, Alex J and Bernhard Schölkopf (2004). "A tutorial on support vector regression". In: *Statistics and computing* 14.3, pp. 199–222.

Kingsford, Carl and Steven L Salzberg (2008). "What are decision trees?" In: *Nature biotechnology* 26.9, pp. 1011–1013.

Magee, John F (1964). *Decision trees for decision making*. Harvard Business Review Brighton, MA, USA.

Biau, Gérard and Erwan Scornet (2016). "A random forest guided tour". In: *Test* 25.2, pp. 197–227.

Yiu, Tony (2019). "Understanding Random Forest". In: URL: https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

Chen, Tianqi and Carlos Guestrin (2016). "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794.

Chen, Tianqi et al. (2015). "Xgboost: extreme gradient boosting". In: *R package version 0.4-2* 1.4, pp. 1–4.

Jain, Anil K, Jianchang Mao, and K Moidin Mohiuddin (1996). "Artificial neural networks: A tutorial". In: *Computer* 29.3, pp. 31–44.

Yegnanarayana, Bayya (2009). *Artificial neural networks*. PHI Learning Pvt. Ltd.

Peterson, Leif E (2009). "K-nearest neighbor". In: *Scholarpedia* 4.2, p. 1883.

Ostertagová, Eva (2012). "Modelling using polynomial regression". In: *Procedia Engineering* 48, pp. 500–506.

Kramer, Oliver and Oliver Kramer (2017). *Genetic algorithms*. Springer.

Leardi, Riccardo (1996). "Genetic algorithms in feature selection". In: *Genetic algorithms in molecular modeling*. Elsevier, pp. 67–86.

"Sloan Digital Sky Survey" (2022). In: *Yahoo Finance*. URL: https://www.sdss.org/.

w3schools (2022). "SQL Tutorial". In: URL: https://www.w3schools.com/sql/.

Willmott, Cort J and Kenji Matsuura (2005). "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance". In: *Climate research* 30.1, pp. 79–82.

De Myttenaere, Arnaud et al. (2016). "Mean absolute percentage error for regression models". In: *Neurocomputing* 192, pp. 38–48.

Feurer, Matthias and Frank Hutter (2019). "Hyperparameter optimization". In: *Automated machine learning: Methods, systems, challenges*, pp. 3–33.

Firth, Andrew E., Ofer Lahav, and Rachel S. Somerville (Mar. 2003). "Estimating photometric redshifts with artificial neural networks". In: *Monthly Notices of the Royal Astronomical Society* 339.4, pp. 1195–1202. ISSN: 0035-8711. DOI: 10.1046/j.1365-8711.2003.06271.x. eprint: https://academic.oup.com/mnras/article-pdf/339/4/1195/18192282/339-4-1195.pdf. URL: https://doi.org/10.1046/j.1365-8711.2003.06271.x.

Li, Li-Li et al. (2007). "Estimating photometric redshifts with artificial neural networks and multi-parameters". In: *Chinese Journal of Astronomy and Astrophysics* 7.3, p. 448.

Maćkiewicz, Andrzej and Waldemar Ratajczak (1993). "Principal components analysis (PCA)". In: *Computers & Geosciences* 19.3, pp. 303–342.

Karamizadeh, Sasan et al. (2013). "An overview of principal component analysis". In: *Journal of Signal and Information Processing* 4.3B, p. 173.

Woolson, Robert F (2007). "Wilcoxon signed-rank test". In: *Wiley encyclopedia of clinical trials*, pp. 1–3.

Rey, Denise and Markus Neuhäuser (2011). "Wilcoxon-signed-rank test". In: *International encyclopedia of statistical science*. Springer, pp. 1658–1659.

Graham, Melissa L. et al. (2018). "Photometric Redshifts with the LSST: Evaluating Survey Observing Strategies". In: *The Astronomical Journal* 155.1, p. 1. DOI: 10.3847/1538-3881/aa99d4. URL: https://doi.org/10.3847%2F1538-3881%2Faa99d4.