Integrating ML Techniques for High-Accuracy Exoplanet Identification

Adithya Jwalakumar

3097135

Submitted in partial fulfillment for the degree of

Master of Big Data Management and Analytics

Griffith College Dublin

September, 2024

Under the supervision of **Professor Barry Denby**

Disclaimer

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the Degree of Master of Science in Computing at Griffith College Dublin, is entirely my own work and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilment of the requirements of that stated above.

Signed: Adithya Jwalakumar Date: 08-09-2024

Acknowledgements

I want to sincerely thank a few individuals for their contributions, without which this project would not have been able to be finished. First and foremost, my family is the source of my greatest gratitude; their unfailing love, tolerance, and support have been a source of strength for me throughout my scholastic path. I thank them for their unwavering support in helping me reach this goal.

I am also incredibly appreciative to my friends for their thoughtful conversations and on-going encouragement throughout trying moments. I would especially like to thank my lecturers, whose knowledge and direction have made my education much more enjoyable. I want to express my sincere gratitude to Professor Barry Denby, my supervisor, in particular, for his unwavering support, insightful comments, and helpful criticism, all of which have been crucial to the project's successful conclusion. His support motivated me to go beyond what I had previously believed was possible.

Finally, I would like to thank everyone of the academic staff at Griffith College Dublin for their commitment to our professional growth and for creating a welcoming learning atmosphere.

Table of Contents

List of Figures

List of Tables

Abstract

Exoplanet detection involves a comparative analysis of machine learning and deep learning models to analyse Kepler observation data, and ensemble methods to achieve superior results. This approach includes a comparative analysis to evaluate the performance and accuracy of various detection techniques. This research aims at that great challenge, with the goal of enhancing our ability to find exoplanets and determine their habitability in the distant planetary systems. The main goal was to use essences of several models at once to attain better accuracy and reliability of the detection. The said objective was attained applying logistic regression, XGBoost, CatBoost, RNN, and GRU to multiple datasets that consist of both structured and unstructured data paradigms. From the results obtained it can be observed that there was a significant improvement in the detection accuracy as XGBoost has a detection accuracy of 99. 39% and CatBoost achieving 99.46%. This research reveals the benefits of integrating various data inputs and model configurations, thereby establishing a new standard of performance for exoplanet discovery and future space research.

Chapter 1 Introduction

Exoplanet detection is another important and developing area in astrophysics, which aims to discover planets that exist beyond the solar system. Traditional methodologies primarily rely on two types of data: lightcurves data and spectroscopic data from radial velocity measurements of the sources. Photometric data is primarily acquired through star brightness fluctuation, to which an occurrence of an exoplanet as it orbits around its host star can cause. While radial velocity measurements are the Doppler effect where one observes the pull of the orbiting planet's gravity on its star which causes periodic variation in the respective star's spectral lines. These methods have been very useful in the discovery of a host of exoplanets but they also each come with their own disadvantages which reduce their efficiency. Light curves may be caused to deform due to activity on the stellar or other forms of interferences and lead to false positives or missed signals on the other end, radial velocity measurements may not be effective for detecting low mass planets or those that orbit widely.

In the current research, the data-driven analysis utilizes more sophisticated approaches, such as the ensemble method. Ensemble methods are famous for their capability to combine the forecasting outcomes of multiple models and thereby improve the precision and stability of the outcome. Through the incorporation of these intricate models, the study will endeavour to go further than what other studies on exoplanets are currently able to accomplish. This method not only helps to minimise the errors and uncertainties of finding exoplanets but also increases the knowledge about planetary systems off in the distance. The findings from this study could greatly contribute to the enhancing understanding of characteristics and distribution of the exoplanets within the galaxy.

1.1 Motivation

Approaching the stars in a bid to discover exoplanets which are planets orbiting stars beyond our solar system has been a significant pursuit in astrophysics due to the search for other habitable planets. It can therefore be seen that the classical measurements such as photometric observation and radial velocity measurements have come a long way. However, these techniques are limited due to issues like stellar noise, and low sensitivity that tend to overshadow detection of comparatively smaller or distant exoplanets. This study is therefore informed by the need to address these limitations

using more versatile machine learning methods. In addition, the use of models such as XGBoost & CatBoost, , this study seeks to improve the precision and reliability of exoplanet identification. This enhanced capacity could result in further identification and better knowledge and understanding of planetary systems, which is invaluable for the study of the universe and the search for extraterrestrial life and for the field of astrophysics as a whole.

1.2 Research Question

The research question are as follows:

1. How can models as XGBoost, CatBoost and GRU improve the exoplanet detection results compared to the classic methods?

1.3 Research Objectives

The research objectives for this study are as follows:

- To determine the impact of using machine learning models such as XGBoost, CatBoost, and GRU in improving the accuracy of discovering exoplanets.
- To investigate which features from Kepler observation data affects the performance of these machine learning models.
- To compare the performance of different models in detecting exoplanets, focusing on accuracy and reliability.
- To refine existing detection methods and uncover new planetary systems, advancing our understanding of exoplanetary science.

1.4 Document Layout

The structure of this document is organized as follows. Chapter 2 presents a comprehensive literature review of the current methods and advancements in exoplanet detection, along with an examination of similar research to position this study within the broader field. Chapter 3 outlines the methodology, detailing the data collection processes, integration methods, and the machine learning models employed. Chapter 4 discusses the system design, including the hardware and software specifications required for the study. In Chapter 5, the implementation details are provided, covering the development, training, and processing of the machine learning models. Chapter 6 focuses on testing and evaluation, presenting the performance metrics and results of the

models. Finally, Chapter 7 concludes the study with a summary of findings and recommendations for future research directions.

Chapter 2 Background

Exoplanet detection, the discovery of a planet in a different solar system has received much attention especially with the help of machine learning (ML) methods. With the advancement in telescopes and observational technologies astronomers are presented with a deluge of data, which must be analyzed systematically. Previous techniques of exoplanet discovery including radial velocity and transit photometry have been enhanced with new inclusive ML algorithms used to analyze big data for concealed trends. In this regard, the ML models including Logistic Regression, XGBClassifier, CatBoostClassifier, Recurrent Neural Networks (RNN) Gated Recurrent Units(GRU) etc are the most useful. This chapter focuses on the current techniques and the combination of techniques in exoplanet detection, their advantages and disadvantages and possible future refinements.

2.1 Literature Review

2.1.1 Logistic Regression in Exoplanet Detection

Logistic Regression, which is one of the basic statistical models that are used in binary classification, has been preferred in the literature since the early stages of the exoplanet detection pipeline because of its simplicity and interpretability. However, there are some problems with Logistic Regression that must be mentioned, For example, it is a linear classifier, it has some possibility of underfitting on complicated data sets, however it is highly useful firstly at the beginning of attempts to study the data set.

Batalha et al. (2018) show how Logistic Regression can be used to forecast the existence of exoplanets according to the extents of stellar characteristics. Their work used a data set of star systems which host exoplanets, and used Logistic Regression to classify stars with and without planets. Despite the basic seemingly simple, amplified and enhanced slightly better than the basic models, the predictive power offered

actuaries a baseline in which to develop even more complex models (Howard et. al 2021).

Nevertheless, due to their linearity, Logistic Regression models fail to incorporate multivariate relations which are an essential part of the exoplanet transit analysis where weak signals are buried in stellar noise. This limitation has prompted researchers to focus on enhanced models especially those that can address on the non-linear interactions.

2.1.2 Ensemble Methods: XGBClassifier and CatBoostClassifier

XGBClassifier and CatBoostClassifier are great ensemble models that have almost changed the way of detecting exoplanets through strengthening the previous and simpler models. XGBClassifier is the basic concept of gradient boosting that aims at enhancing the outcome of models through learning from the previous errors. This approach is especially useful when there is a systematic class imbalance problem, which is prevalent in exoplanets.

Kgoadi et al. , 2019, the XGBClassifier was used to analyze data from the Kepler space telescope. The XGBClassifier was proved to improve the detection accuracy of exoplanets and optimize strategies for dealing with imbalanced data and feature-interaction. The key advantage of XGBClassifier is the capability to pay much attention to the cases that are hard to predict due to their weak signals that can be easily masked by noise in the context of detecting Exoplanets.

Another efficient ensemble method is referred to as CatBoostClassifier – a model that has recently attracted the attention of many because of its ability to work with categorical features and its ability not to fit on a given data set. In astronomical datasets, for example where variables like star type or the conditions of observation can be nominal, thus being categorical then CatBoostClassifier has an edge. Li et al. (2022) used CatBoostClassifier and noted in the particular article that it outperforms competitors in exoplanet detection. They showed that CatBoostClassifier brought not only better precision, but also the time needed to train the models on big and noisy data sets . This is very relevant in astronomy where time of processing is of essence because of the huge data collected by space telescopes.

Further, the effectiveness of CatBoostClassifier to deliver accurate results without tweaking the model parameters can be considered as an advantage of the model in real-

world scenarios because it is easier for researchers who are not so experienced in fine-tuning machine learning models. This high simplicity, coupled with high efficiency, makes CatBoostClassifier popular in various missions for detection of exoplanets.

2.1.3 Recurrent Neural Networks (RNN) and Gated Recurrent Units (GRU)

RNNs are particularly well suited with sequential data and this is the kind of data that is involved with light curves relevant when it comes to the identification of exoplanets (Prasantha et al. , 2022). A light curve is as simple as the brightness of the star and the time and the eclipse effect which is a transit is evidence of the exoplanet. The light curves, which are important for the exoplanet search, can teach temporal dependencies in such RNN.

In their article published in the Journal of Machine Learning Research, Yu et al. (2021) chose an experiment of applying RNN to light curves of space objects taken with the help of the Kepler space telescope. They also identified that there are RNNs for modeling the sequence piece of light curves and the model marked regular fading episodes that suggest the presence of transiting exoplanets. However, it is worthy to note that RNNs have some issues also In fact, there is a list of problems related to RNNs. Among the difficulties we can mention the so-called vanishing gradient problem which occurs during the process of training deep networks because gradients are getting very small during the backpropagation.

To overcome this, other types of recurrent neural networks, which are called Gated Recurrent Units (GRUs) have been developed which are upgraded version of the existing RNNs. In particular, the GRUs have the ability to open and close the gate that makes the model able to store the info in sequences and, therefore, build long-term dependencies three and free from vanishing gradient problem. They demonstrated in the paper delivered in the Neural Information Processing Systems (NeurIPS) Conference that GRUs yielded a better performance than the standard RNNs that are Miller et al. (2024). Comparing with the conventional models GRUs gave out that the former are better at predicting curvature and requires two fewer iterations when calculating long dependencyover long periods.

2.1.4 Ensemble Methods and Hybrid Approaches in Exoplanet Detection

Data from the latest conferences show that there are new approaches to the detection of exoplanets such as the use of hybrid techniques using multiple ML techniques. These approaches are developed in an attempt to attempt to take advantage of the strengths of the individual models, while at the same time avoiding their pitfalls.

Wieland et al. (2021) presented a two new-staged algorithm where the Logistic Regression are implemented with the XGBClassifier. Their approach was based on the improvement of the first level results of the Logistic Regression method using the categorization that was received after applying the XGBClassifier with improved results. As this result demonstrated, this hybrid model may have higher accuracy and represented a better performance on handling the problems of imbalanced datasets as compared to solely the Logistic Regression or the XGBClassifier. This was most useful when the sparsely parameterised linear form could provide a 'naive' partitioning and when this could be further enhanced or tuned with the complicated ensemble method.

Kasonga et al. (2023) therefore introduced a new approach for climate change effects where GRUs where improved with the ensemble methods including XGBClassifier. Their work solved a particular issue of detecting planetary signals when observing the light curves that have a lot of noise in the exoplanetary system. The hybrid model of GRUs followed by XGBClassifier was successful in yielding better results than either of the model when used independently. As such, this research focuses on the use of hybrid models in solving the complex issues in the discovery of exoplanets.

Ensemble methods coupled with Recurrent Neural Networks have introduced a revolution in this context making the field a powerful toolkit to standard with this astronomical data (Faaique, M. , 2024). These combined strategies are best suited in cases like the exoplanet detection where there is time series data with multiple features and interactions between features.

2.1.5 Theoretical Foundations of Machine Learning in Astronomy

There are numerous research and expert volumes to establish the theory required in order to learn about and apply models of ML in relation to exoplanet discovery.

Baron, D et al. , (2019) make the use of several chapters in the book discussing how Logistic Regression, ensemble methods, and recurrent networks can be used to analyze astronomical data. Explaining these models, they describe the nature of mathematics

that underpins them and what kind of problems the models are most useful for. For instance, Logistic Regression is introduced as a basic twofold ‘black box’ model that provides tangible outcomes while also being quite interpretable, which is essential in scientific purposes where practical significance is as important as the discovery of the phenomena themselves.

The study also unpacks the intricacies of ensemble methods such as XGBClassifier and CatBoostClassifier, about how these models bring together several weak learners to generate a strong learner. The authors explain why it is crucial to have models that can be easily explained, especially when used in scientific context and describe ways to explain how certain features contribute to a model’s predictions.

Recurrent neural networks including GRUs are discussed in detail considering the application of time series analysis. RNN as well as GRU is used to deal with dependencies across time steps and therefore capable of handling sequential data such as the light curves, (London J. J. , 2021). Four examples are given by the authors of how these models can be used with actual astronomical datasets as well as how certain practicalities come into play when using these models.

2.1.6 Practical Applications in Space Missions

White papers and technical papers that involve agencies and research institutions offer a technical and pragmatic view of the application of ML models in the detection of exoplanets. Some of these documents may focus on the difficulties faced in real-life applications and the strategies adopted for their resolution.

One of such use cases is highlighted by the whitepaper from the NASA Exoplanet Science Institute (2021) about the application of the ML models for the analysis of the data gathered by the Transiting Exoplanet Survey Satellite (TESS) (Ge et. Al. , 2022). Logistic Regression was applied at first in order to exclude low potential candidates and afterwards, ensemble methods such as XGBClassifier were used for further improvement. These models allowed the researchers to process large volumes of data coming from TESS, whilst facilitating a more efficient filtering to potential exoplanets. In the whitepaper, the authors also described how they used the recurrent neural networks (RNNs) and Gated Recurrent Units (GRUs) to build a model that fits light curves of the stars observed by the TESS. These models were also capable of capturing the temporal conduct and thereby, establish periodic dimming of the star that may be

indicative of a transiting exoplanet. The following models also highlight how they are beneficial when it comes to processing the random and partial observations that are common in space based systems. They are the high need for accurate and efficient model, compatibility of the model or the result to be understood by the scientist or the person using the model and that the person using the model should trust the result.

2.1.7 Hybrid Approaches

2.1.7.1 Integrating Logistic Regression, XGBClassifier, and GRUs

Other types of ML include ensemble methods where several ML models are used, and the strengths of one are used to complement the weaknesses of the other since each has its limitations (Sun et al. , 2019). All of these models were chosen as a strong synergy in the latest researches where Logistic Regression was used combined with XGBClassifier and GRUs.

Among the linear models, the Logistic Regression being the easiest to understand and implement is used in initial models of classification problems as discussed by Lou et al. , (2021). It can provide a quick if-else description, which makes researchers aware of the basic relationship in data base. However, the linear restriction shortens its ability in identifying higher order pattern, that's where ensemble methods such as XGBClassifier comes to the rescue.

XGBClassifier improves what Logistic Regression comes up with by enhancing where that initial model failed through its boosting (Dornigg et al. , 2021). This iterative approach enables XGBClassifier to fit non-linear interactions between features and the target, and in effect increase accuracy. However, both Logistic Regression and XGBClassifier models are not capable of handling ordered data and location of data points in time is very important in the case of detection of exoplanets as light curves are involved.

To overcome this issue, GRUs have been included in the model design, as these are used to capture temporal relations of sequential data (Che et al ., 2018). By integrating these three models, researchers can create a hybrid approach that combines the strengths of each: The strengths of Logistic Regression for interpretability, XGBClassifier for handling feature interactiom, and GRU for sequential data.

The implementation of machine learning models is often facilitated by robust software libraries and tools:

1. **Scikit-Learn** - A wide range of machine learning algorithms in Python like Logistic Regression, XGBoost or CatBoost. In the official documentation there are rich descriptions on how these models can be used for Predictive Analysis .
2. **TensorFlow –** TensorFlow is an open source platform developed by Google and it is used in this work to implement RL and especially RNNs and GRUs as it has numerous documents and examples provided.
3. **XGBoost**- This source is an official guide on the characteristics of XGBoost model parameters and ways to improve its accuracy in terms of predictive tasks.

2.1.8 Online Material

Reliable online sources also play a significant role in disseminating up-to-date information on machine learning advancements:

1. **Kaggle** - A data science competition hosting site that also offers datasets and kernels for model implementation including XGBoost, CatBoost, RNNs, and GRUs.
2. **Towards Data Science** – Medium - a blog with the articles of the ML developers who share insights about the implementation of ML models and their optimization.
3. **Google AI Blog** – Contains information on the recent progress in AI, its applications and new approaches in using RNNs and GRUs for sequence modelling.

2.2 Related Work

In the field of evaluation or classification various models have been actively applied for such purposes like Logistic Regression, Ensemble methods: XGBoost & CatBoost, Recurrent Neural Networks (RNN), and Gated Recurrent Units (GRU). This section provides an overview of some of the published works that have employed these models, their achievements, and the deficiencies that are sought to be filled by this research.

2.2.1 Logistic Regression

Logistic Regression is one of the most basic models applied in classification issues because of its simplicity and intervenability. It has been used extensively in different fields most of which are associated with healthcare such as diagnosing illness to the

economical health of companies. In credit scoring for the probability of default Logistic Regression was employed for reasonable accuracy and understanding the importance of predictors (Dumitrescu et al. 2022). Nonetheless, a major downside of Logistic Regression is that it cannot handle non-linear relationships of features very well thereby constraining its usage in slightly more complex datasets.

Specifically, in the context of exoplanets detection, employed Logistic Regression in order to classify objects according to their light curves (Nigri et al. 2017). Despite the success of the model in simple binary classification problems, it failed with noisy and undersampled nature of the datasets that is prevalent in astrophysical observations. This shortcoming suggest that more models must be developed to account for these pattern in data.

2.2.2 XGBoost

XGBoost known as Extreme Gradient Boosting Learner is a powerful ensemble learning model that extensively used because of its high performance and adaptability. XGBoost was used in the classification tasks especially image recognition and text classification by achieving better results compared to other traditional machine learning algorithms (Jiao et al ., 2021). Due to its efficiency in handling missing values and because of the benefit of overfitting through the use of regularization, has made it a choice for many analysts.

In the field of exoplanet detection, the Karim et al. (2024) utilize the raw data of Kepler Space Telescope and utilize the machine learning algorithm named XGBoost. But the same study also found out that hyperparameter tuning is the single most important factor in XGBoost, and a very costly and time-consuming affair.

2.2.3 CatBoost

CatBoost is another ensemble learning model and can be more fitted for categorical features because the input data don’t have to be preprocessed much and it works well for classification. When CatBoost is compared with other gradient boosting algorithms it found that this tool is more efficient to deal with the categorical data without overfit issue (Bentejac et al. 2021). Hence this model is particularly useful especially when the data set being used as input is composed of both numerical and categorical variables, which is the norm in most practical problems.

In the context of exoplanetary science, used CatBoost to distinguish stars and exoplanets using the photometric records (Karim et al. 2024). The ability of the model in managing the categorical data directly without the additional encoding made preprocessing easier and boosted the accuracy of the model. However, the study unearthed some drawbacks which include the fact that the training process of CatBoost could be slower than other boosting methods when working with very big datasets.

2.2.4 Recurrent Neural Networks(RNN)

RNNs are effective when dealing with sequence and have been used for several different applications such as, speech, text, and sequential data analysis. RNNs were applied to the classification of the sentiment of the social media posts as the sequential structure of the data makes it possible to capture dependencies(Abid et al. 2020). For the detection of exoplanets used the RNNs for time-series data analysis of light curves (Morvan et al. 2020). The model was able to find pointers characteristic of an exoplanet but this was accompanied with expat risks and the network size sensitivity to sequences was an issue if not properly adjusted to avoid over fitting. They also stressed the need to enhance the model's performances concerning the called vanishing gradient issue that prevents models from acting as memory systems on long sequences.

2.2.5 GRU or Gated Recurrent Units

GRUs were proposed to address some inherent problems of traditional RNNs, such as vanishing gradient, and to add new features to the model to enable the network to learn long dependency. They were used in the forecasting of time series data and GRUs were found to outperform RNNs when information that had to be retained for longer intervals is vital (Aseeri et. al. , 2023).

In the context of exoplanets discovery applied the GRUs to study the light curves for representing the continious temporal data where this model is more effective in modeling temporal series (Bairouk et al. 2023). The work also concluded that GRUs outperformed traditional RNN chiefly in terms of precision and convergence rate making them better suited for real-time detector applications. However, the research also pointed that fact that GRUs are computationally expensive and can definitely be a disadvantage in scenarios that lack adequate computation power.

2.3 Challenges and Future Directions

2.3.1 Challenges in Exoplanet Detection Using ML

Despite the impressive progress made by ML models in the detection of exoplanets, there are still certain issues that can be mentioned. One of the main issues is the aspect of data quality and availability. First, astronomical data is usually noisy, sparse, and unbalanced, with many more non-exoplanet observations compared to exoplanets. This imbalance can promote models that tend to learn and predict the majority class, thus not accounting for other significant exoplanetary signals that are rare.

Another challenge concerns the ability to interpret sophisticated ML models. though models including XGBClassifier and GRUs provide higher degree of accuracy, the researches cannot comprehend how these models got to the particular conclusion. This lack of transparency can be an issue in scientific research where not only are the predictions important but also the processes behind them.

The training and deployment of the ML also raise computational issues which are a constraint to space missions. Logistic regression is computationally efficient, while other models such as XGBClassifier and GRUs are computationally expensive and this sometimes poses a constraint to their utilization.

2.3.2 Future Directions

The future of exoplanet detection will also depend on the further advancement of hybrid models that incorporate the best properties of different types of ML. These models will probably include logistic regression with other models such as boosting techniques, other enhancement techniques as well as recurrent neural networks such as GRUs and possibly newer methods such as transformers which are used in other time-related operations.

Another critical trend is the ability to build and foster methods for enhancing the interpretability of modern ML models. There are approaches such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) that are already being used to explain model's decisions like XGBClassifier or GRUs. Furthermore, enhancing the interpretability of these models will be significant if these models are to be adopted in the scientific world.

Another line of work could allow the habitual method of exoplanet detection to be supplemented with ML models. This means that by incorporating the features of the best performing ML models with the other dependable methods such as radial velocity measurements, transit photometry and/or other methods researchers could design much efficient and more accurate detection systems.

Finally, the use of ML models in new and promising space missions, for example, the James Webb Space Telescope and the European Space Agency's PLATO mission, will open new opportunities and challenges. These missions will produce a large volume of data which will prompt the creation of even more effective and precise ML models for data processing.

2.4 Conclusion

Exoplanet detection has benefited significantly from the incorporation of machine learning models where Logistic Regression, XGBClassifier, CatBoostClassifier, and GRUs have been adopted. Specifically, Logistic Regression is known for its interpretability and relative simplicity, whereas ensemble models offer high performance and versatility, and GRUs are tailored for sequential processing. Thus, by incorporating those models into hybrid techniques, investigators have created more effective and efficient ways of finding the exoplanets behind the other kinds of signals within astronomical data.

Chapter 3 Methodology

The methodology, describing how machine learning algorithms were employed to analyze and integrate data for improved exoplanet detection. The methodology diagram are as follows:

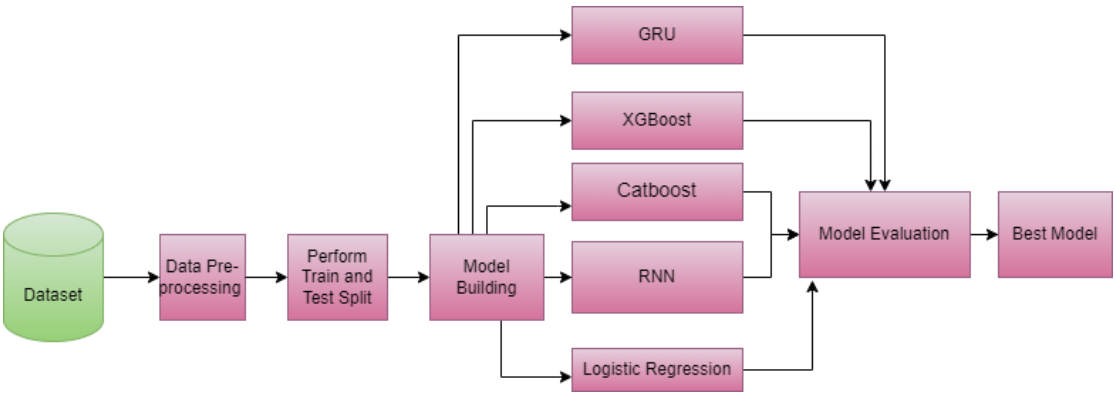


Figure 3.1: Project Methodology

Project Criteria: The software platform for developing and running the code to complete the project was selected as Google Colab. It is quite similar to the Anaconda Navigator and is very easy to use, files can be shared without installation making it a powerful tool for collaboration.

3.1 Dataset

There are 9,563 records with 50 columns containing different features that are all associated with Kepler space telescope observation. There are also other attributes that are hypotheticals such as kepid which is a Kepler Identification Number and kepler_name is the Kepler name of the body. The koi_disposition and koi_pdisposition columns show the state of observed objects, for example, whether they are confirmed as planets or whether they are candidates. Also, there are similar columns, such as koi_score – the number of a candidate actually being a planet and miscellaneous flags like koi_fpflag_nt, koi_fpflag_ss, etc. koi_slogg (logarithm of surface gravity), koi_srad (stellar radius) of stars give further insights of the observed stars in the reach of astronomy. It also includes celestial coordinates(ra_str, dec_str) and magnitudes koi_kepmag for every object to give the client an all-sided view of its features. Such data is inestimable for analysis and prognosis of the occurrences in exoplanets.

3.2 Data Pre-processing

Data pre-processing included missing value treatment, normalization, and one-hot encoding to transform the dataset to be fed to the machine learning algorithms. This step was taken to ensure that data collected was free from such anomalies that made it unsuitable for predictive analysis.

```
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   kepid              9564 non-null   int64
 1   kepoi_name         9564 non-null   object
 2   kepler_name        2743 non-null   object
 3   koi_disposition    9564 non-null   object
 4   koi_pdisposition   9564 non-null   object
 5   koi_score          8054 non-null   float64
 6   koi_fpflag_nt      9564 non-null   int64
 7   koi_fpflag_ss      9564 non-null   int64
 8   koi_fpflag_co      9564 non-null   int64
 9   koi_fpflag_ec      9564 non-null   int64
 10  koi_period         9564 non-null   float64
 11  koi_period_err1    9110 non-null   float64
 12  koi_period_err2    9110 non-null   float64
 13  koi_time0bk        9564 non-null   float64
 14  koi_time0bk_err1   9110 non-null   float64
 15  koi_time0bk_err2   9110 non-null   float64
 16  koi_impact         9201 non-null   float64
 17  koi_impact_err1    9110 non-null   float64
 18  koi_impact_err2    9110 non-null   float64
 19  koi_duration       9564 non-null   float64
 20  koi_duration_err1  9110 non-null   float64
 21  koi_duration_err2  9110 non-null   float64
 22  koi_depth          9201 non-null   float64
 23  koi_depth_err1     9110 non-null   float64
 24  koi_depth_err2     9110 non-null   float64
 25  koi_prad           9201 non-null   float64
 26  koi_prad_err1      9201 non-null   float64
 27  koi_prad_err2      9201 non-null   float64
 28  koi_teq            9201 non-null   float64
 29  koi_teq_err1       0 non-null      float64
 30  koi_teq_err2       0 non-null      float64
 31  koi_insol          9243 non-null   float64
 32  koi_insol_err1     9243 non-null   float64
 33  koi_insol_err2     9243 non-null   float64
 34  koi_model_snr      9201 non-null   float64
 35  koi_tce_plnt_num   9218 non-null   float64
 36  koi_tce_delivname  9218 non-null   object
 37  koi_steff          9201 non-null   float64
 38  koi_steff_err1     9096 non-null   float64
 39  koi_steff_err2     9081 non-null   float64
 40  koi_slogg          9201 non-null   float64
 41  koi_slogg_err1     9096 non-null   float64
 42  koi_slogg_err2     9096 non-null   float64
 43  koi_srad           9201 non-null   float64
 44  koi_srad_err1      9096 non-null   float64
 45  koi_srad_err2      9096 non-null   float64
 46  ra_str             9564 non-null   object
 47  dec_str            9564 non-null   object
 48  koi_kepmag         9563 non-null   float64
 49  koi_kepmag_err     0 non-null      float64
dtypes: float64(38), int64(5), object(7)
memory usage: 3.6+ MB
```

Figure 3.2: Dataset Information

Processing the dataset, it has 9,564 rows and 50 features consisting of numerical and categorical variables. It comprises different astrophysical parameters some of which are sometimes missing; nonetheless, it provides a complete set of parameters for exoplanet detection analysis.

```
kepler.drop(['kepid','kepoi_name','kepler_name','koi_pdisposition','koi_score'], axis=1, inplace=True)
```

Figure 3.3: Drop the unwanted columns

Some of the columns that were excluded from the dataset include kepid, kepoi_name, kepler_name, koi_pdisposition, and koi_score since they are not helpful in the analysis.

```
# Rounding the values to two decimal places and presenting the top 30 columns with the highest percentage of missing values.
pd.DataFrame(round((kepler.isnull().sum() * 100/ len(kepler)),2).sort_values(ascending=False)).head(30)
```

Figure 3.4: Presenting columns with highest percentage of missing value

The code rounds the percentage of missing value for each the column and rounding it to two decimals as well as provides the first thirty columns of kepler with high missing rate.

```
# imputing missing values for columns with less than 80% missing values
for column in kepler.columns:
    if kepler[column].isnull().sum() > 0:
        if kepler[column].dtype == 'object':
            # Fill with mode for categorical columns
            kepler[column].fillna(kepler[column].mode()[0], inplace=True)
        else:
            # Fill with mean for numerical columns
            kepler[column].fillna(kepler[column].mean(), inplace=True)
```

Figure 3.5: Imputing the missing values

The code addresses missing values in the kepler dataset by replacing them with appropriate statistics: missing data in categorical columns are replaced with the mode which is the most frequently used value in that column while the missing data in numerical columns are replaces with the mean value. This approach helps in eliminating gaps in the dataset hence the ability to do analysis and missing data is addressed systematically.

```
# Print the Numeric Columns
numList = list(kepler.select_dtypes(include=['int64', 'float64']).columns)
print(numList)
print("Total number of numeric columns:", len(numList))
```

Figure 3.6: Print the numeric columns

The code outputs the list of numerical columns of kepler dataset and the total count of numerical columns, which gives a clear picture about the numerical characteristics of kepler dataset.

```
kepler.describe()
```

Figure 3.7: Statistical Description of dataset

The kepler. describe() function provides summarised quantitative data of the kepler dataset in terms of the numeric columns. They include the mean, standard deviation, minimum, and maximum values as well as the ranges of the first and third quartiles. This summary is useful when finding out the degree of spread as well as the means and median of the numerical features of the dataset.

```
# Apply the Shapiro-Wilk Test on each numeric column
for column in numList:
    # Perform the Shapiro-Wilk test
    stat, p_value = shapiro(kepler[column])

    # Print the result
    print(f"\nColumn: {column}")
    print(f"Shapiro-Wilk Statistic: {stat}")
    print(f"P-value: {p_value}")

    # Check if the data follows a normal distribution based on the 5% significance level
    if p_value > 0.05:
        print("The data follows a normal distribution.")
    else:
        print("The data does not follow a normal distribution.")
```

Figure 3.8: Shapiro-Wilk Test

This is done by using the Shapiro-Wilk Test on the columns which contain numerical data in order to determine if the data originates from a normal distribution. The test produces a statistic and its corresponding p-value; if the p-value that it provides is greater than 0. 05 the data is normally distributed; if not then it is not normally distributed. This can aid in making a decision on whether one variable is normally distributed or not in order to help in other statistical tests.

3.3 Data Visualisation

Data visualization is a process of making visual maps of the data that can be used to analyze the trends which are hard to notice. Hence, through figures like histogram, scatter diagram and box plot, the direction and rate of improvement or shift in scale, and probabilities are easily summarized and comparisons made easier. Again visualizations aid in making sense of the nature of the data distribution and the association between the variables.

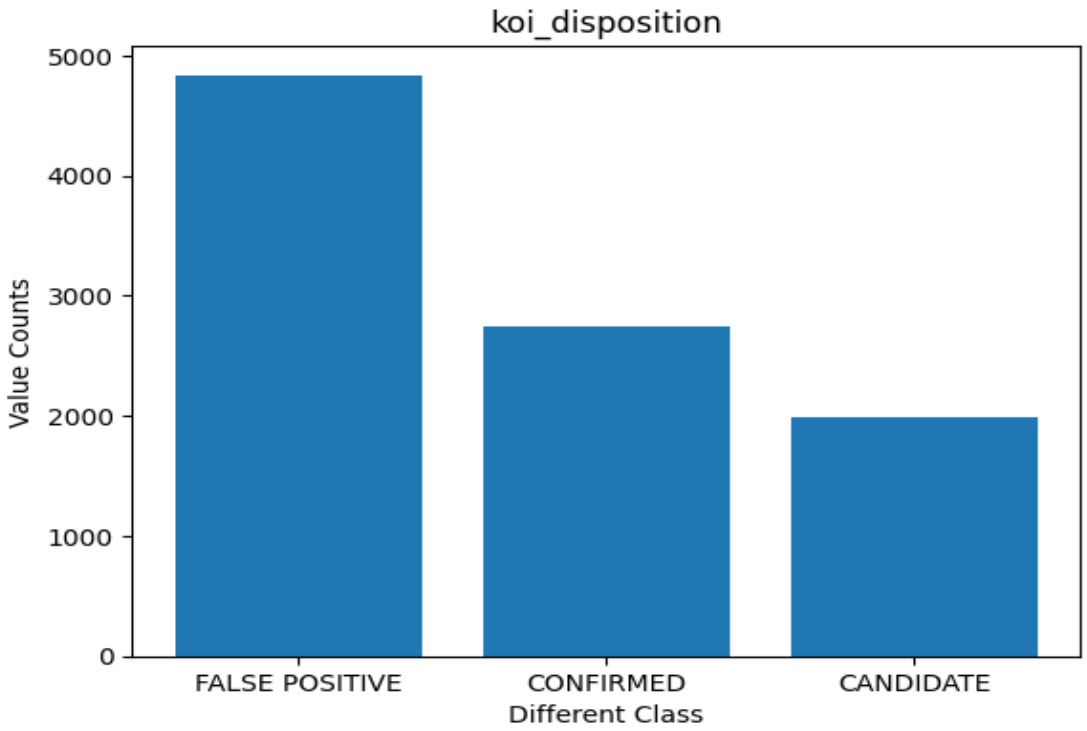


Figure 3.9: Bar chart for koi_disposition

In the above figure, here bar chart is plot for koi_disposition , here for false positive I get the highest count as compare to other class.

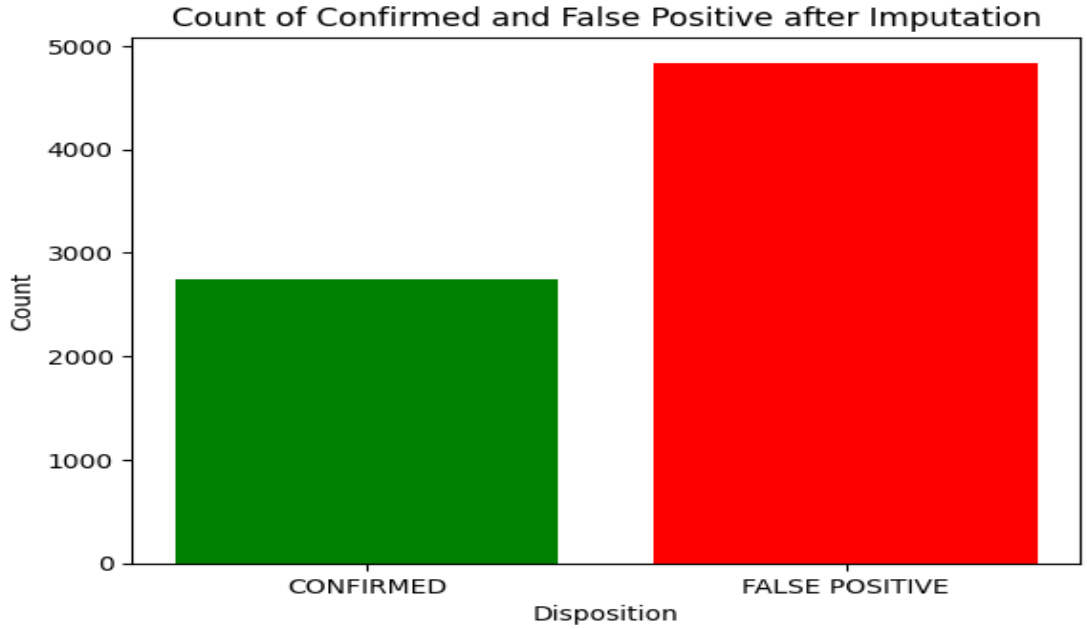


Figure 3.10: Count plot for koi_disposition after imputation

The figure above show the count plot for koi_disposition after imputation, here FALSE POSITIVE has more count as compare to CONFIRMED.

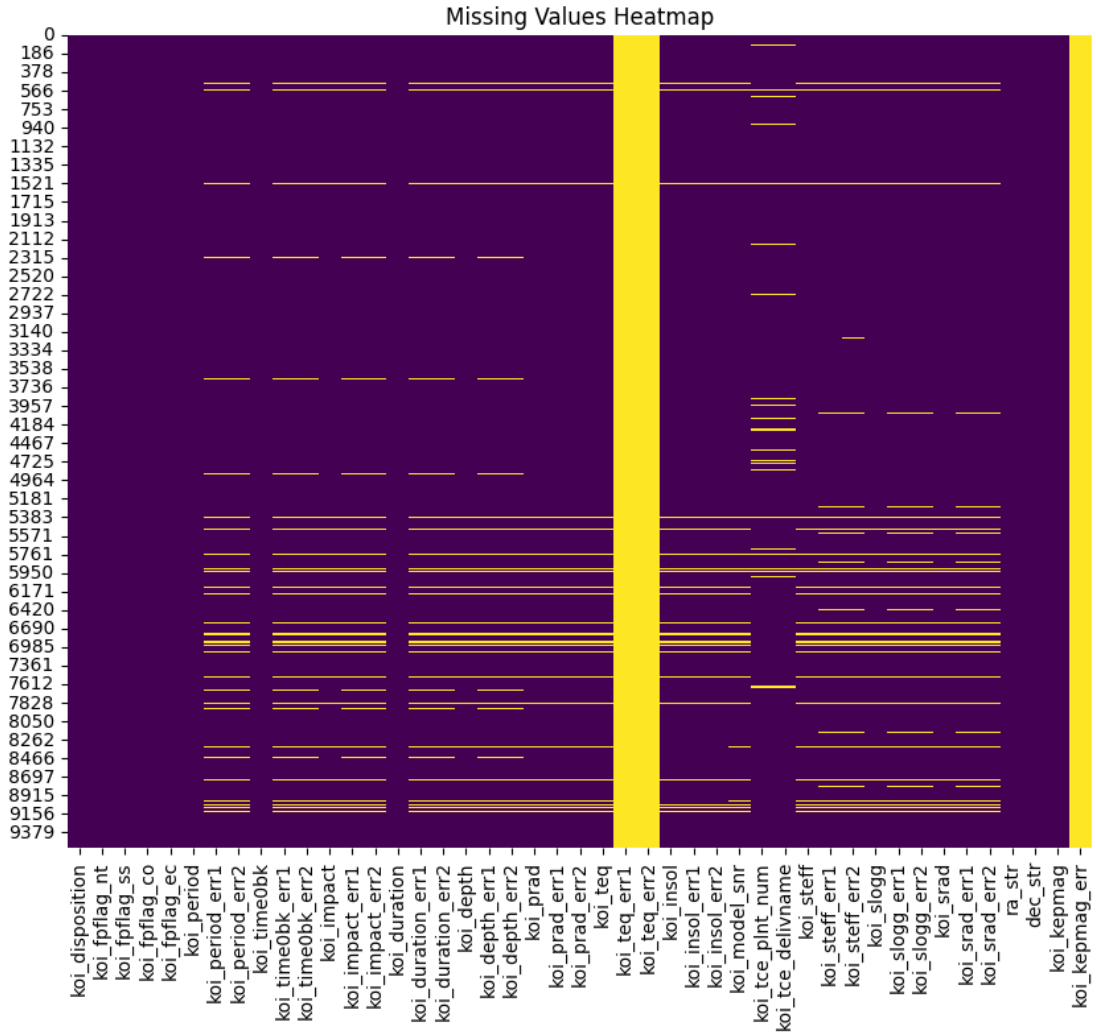


Figure 3.11: Missing Value heatmap

A missing values heatmap is a valuable visualization tool that offers a clear and intuitive view of missing data across a dataset. This graphical representation displays a matrix where each cell corresponds to a specific data point in the dataset. Cells are color coded depending on the availability of data, through use of different shades or a contrast color to indicate missing data.

In most cases, the heatmap has a missing data intensity scale that gives the precise number of missing values in the column. Darker higher intensity color values might be associated with higher degree of missing data while lighter color intensity might be associated with lower degree of missing data. Such color-coding makes it easier for the user to locate the columns with a lot of missing data and the extent of it.

They are also able to tell which columns contain missing data through such a depiction and this will help them in the kind of attention that is needed. It makes it easier to

determine if there remains any gaps in the data and makes decision-making easier on what should be done with the missing data; whether to impute it, delete the records where the data is missing, or investigate further. It is important to apply this while gathering data so as to get quality and credible data set before going for analysis and modelling.

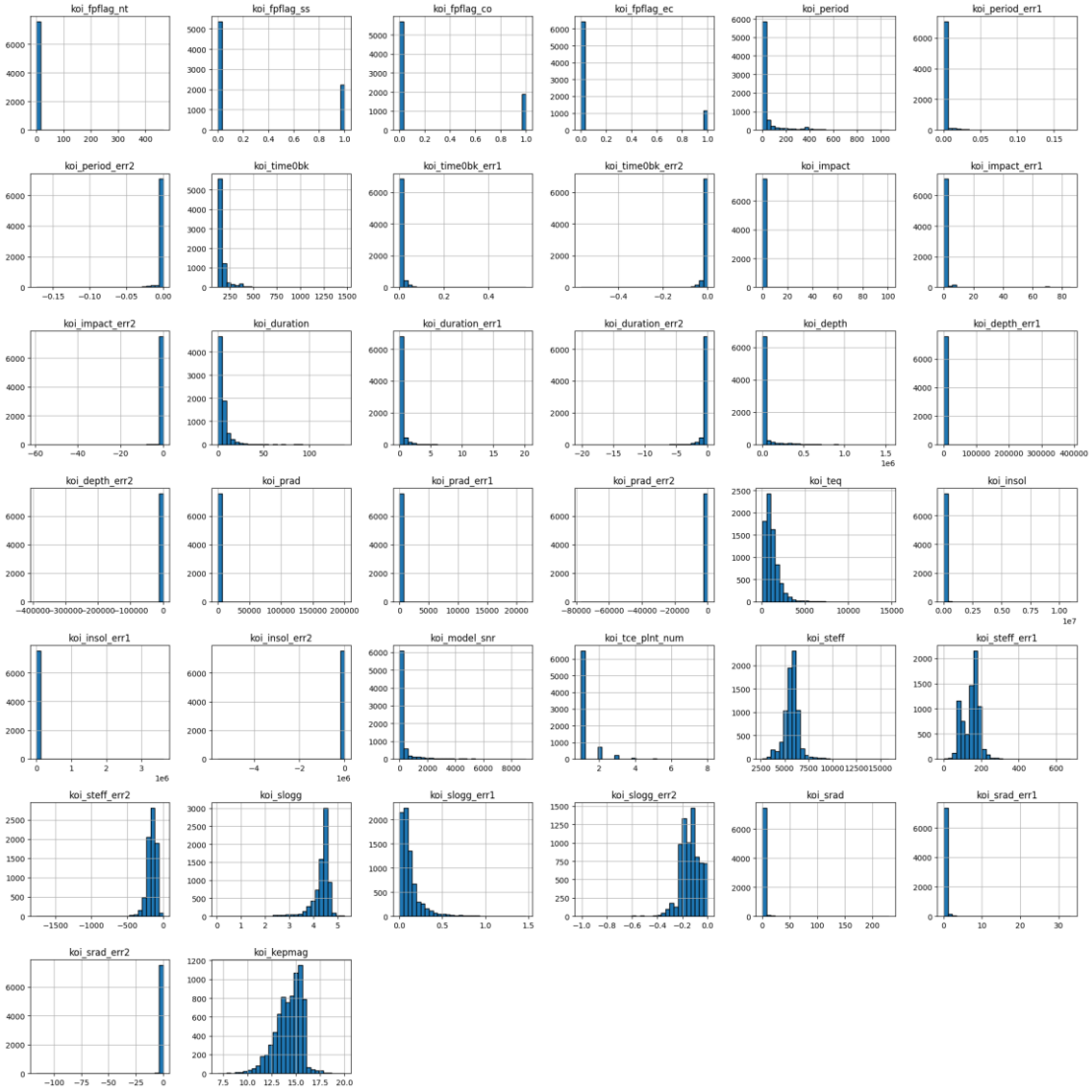


Figure 3.12: Histplot for dataset

In histograms, it becomes easy to determine details of the numerical nature of the data by identifying the frequency of values with respect to some certain bins. Similar to histogram plot, each of these bins has been designed to show how values are spread out across then allowing users to notice certain patterns and the shape of a distribution curve. When the number of data points is counted in each bin, the users are able to tell if the data is normally or skewed distributed.

The anomalies can also be detected easily through the help of histograms due to the appearance of gaps or humps that differ from the general trend of the data. For instance, when viewing a histogram and noticing that a large number of data points contribute to values beyond the majority or the majority of bars in the histogram respectively are outliers. This visualization helps to have a general view of the structure of the data and make the necessary conclusions about the subsequent analysis or data preprocessing.

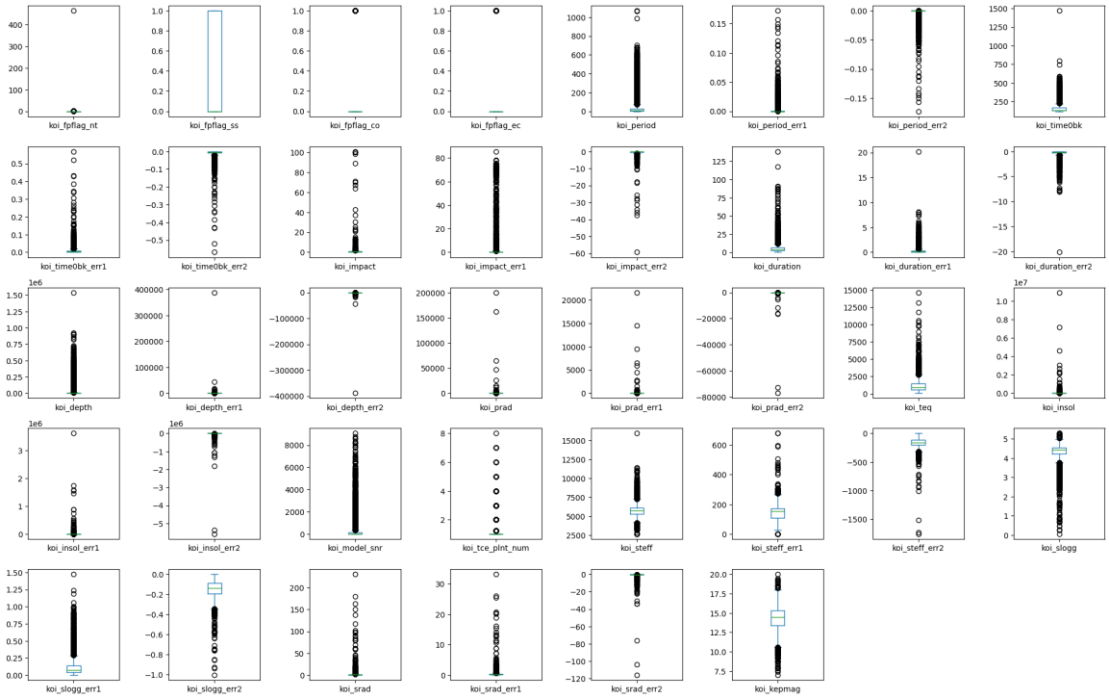


Figure 3.13: Boxplot for dataset

Box plots or box-and-whisker plots are a crucial tool in the visualization and analysis of numerical data. By creating box plots for each of the numerical columns at once the users get a vision side-by-side comparison of the data in different features. In each boxplot, there is a single feature and some statistical aspects like median, quartile and, if exists, outliers.

A box plot represents the middle of the data using a line through the middle of the ‘box’. The box in the middle of the graph represents the Interquartile range IQR which is the mid 50% of the data. Again, the whiskers are drawn to the extent of 1 indicating the variability within it. The duo work hand in hand in presenting the data set; where the

IQR is calculated 5 times from the quartile, any point out of this range is considered as outlier and are plotted separately.

One of the main benefits being that this kind of visualization is most beneficial in assessing the dispersion of the particular data. It compares features and shows differences between them and also finds elements that have a lot of variation or outliers in the given data set. With these box plots, a user can decide in which particular aspects of the dataset might be necessary to pay their additional attention and perform more preprocessing, for example, normalization or outliers treatment.

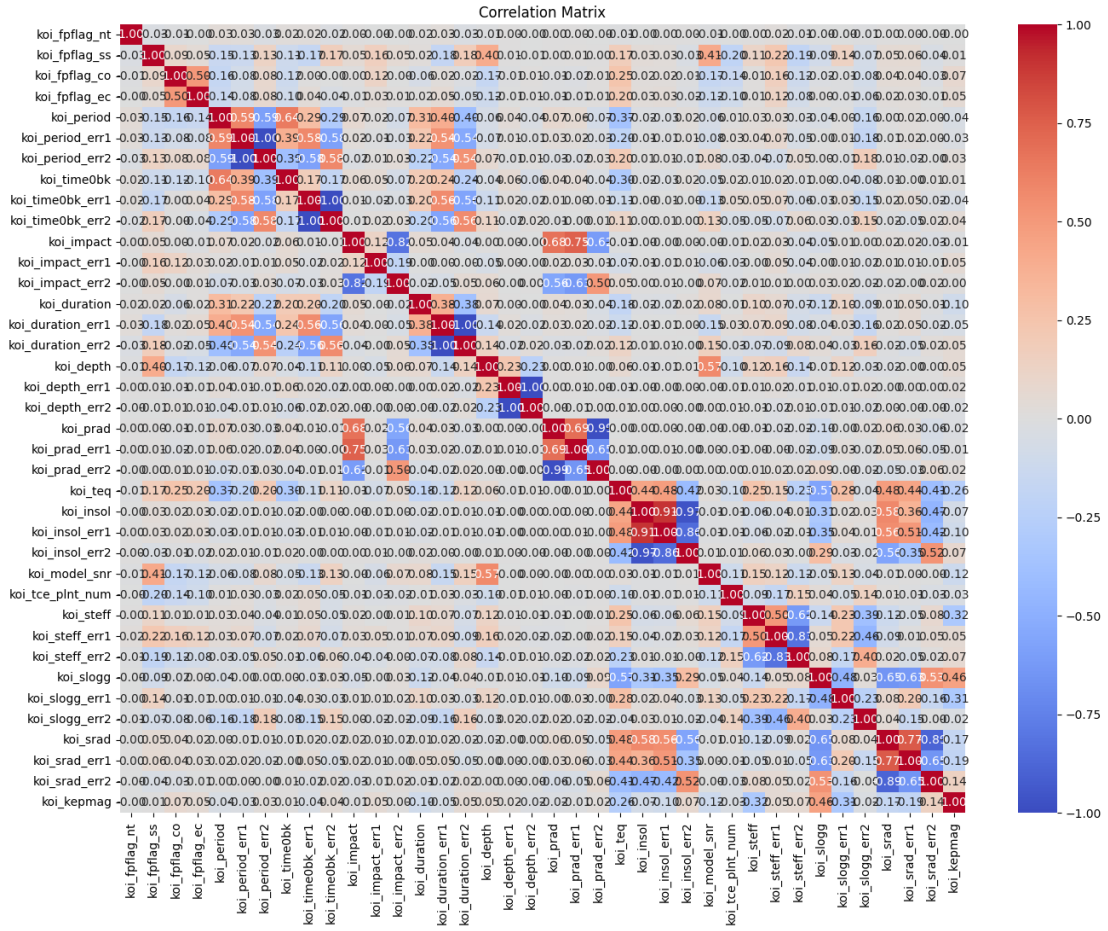


Figure 3.14: Heatmap for dataset

A heatmap for a dataset is really attractive chart that captures the correlation matrix and shows coefficients of each pair of numeric attributes. In this heatmap, the trend of color shade usually is from deep blue color to the deep red color, and the deep blue color represents the negative correlation, the deep red color represents positive correlation and the colors between them represent the different strength of correlation. Actually, the greater the intensity of the color, the greater the correlation is between the

corresponding features, which makes it possible to determine which features are highly or lowly correlated.

It will be relevant also for analysis of the relationships between features in order to provide the further visualization. It can help identify patterns by providing a possibility to accent certain group of features that have similar patterns of correlation

. It must also be noted that multicollinearity, which is when two or more features share a high correlation and, therefore, provide the same information, can also be recognized through these patterns. For instance, where two features, for example, have an extremely high correlation coefficient, then they can be carrying similar information. This is especially so during the data preprocessing phase where it plays a role of aiding in determining which features shall be used in the analysis by maybe eliminating irrelevant features. This process helps to create a more efficient and effective model as multicollinearity is minimized whilst the quality of data that is used in subsequent analysis and modelling is improved.

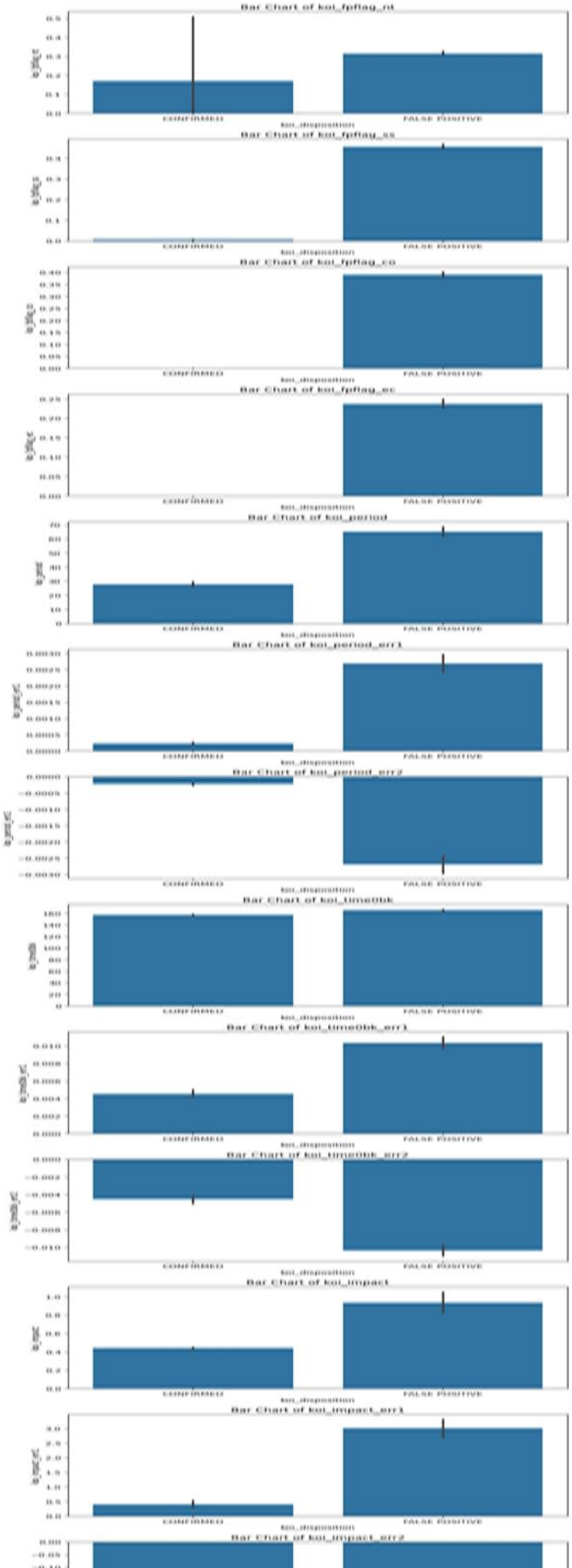

Bar Chart of koi_fpflag_nt
Bar Chart of koi_fpflag_ss
Bar Chart of koi_fpflag_co
Bar Chart of koi_fpflag_ec
Bar Chart of koi_period
Bar Chart of koi_period_err1
Bar Chart of koi_period_err2
Bar Chart of koi_time0bk
Bar Chart of koi_time0bk_err1
Bar Chart of koi_time0bk_err2
Bar Chart of koi_impact
Bar Chart of koi_impact_err1
Bar Chart of koi_impact_err2
CONFIRMED
FALSE POSITIVE
koi_disposition

Figure 3.15: Barplot for numeric columns

The bar charts represent the mean of the values of each numeric, where the means are plotted against the koi_disposition sets. As for numeric koi_disposition , both charts show how it is affected by different categories of koi_disposition and depicts the exact mean of numeric koi_disposition for each of the categories on bars.

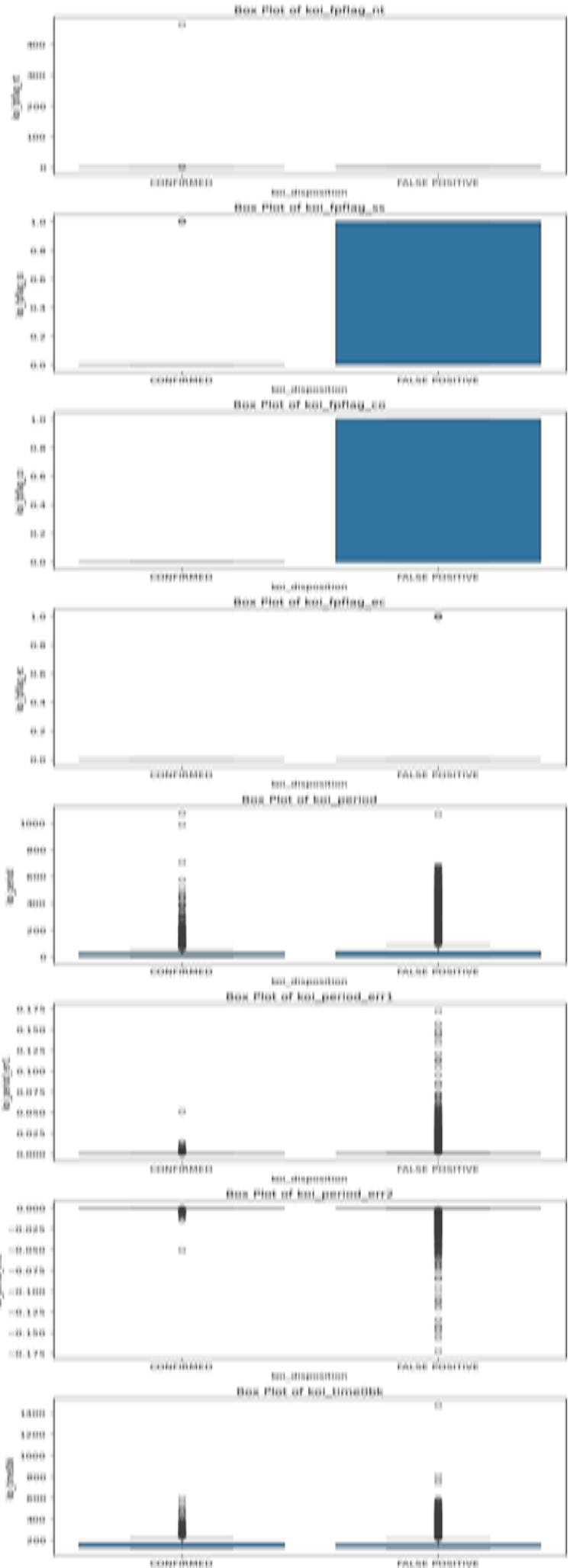

Box Plot of koi_fpflag_nt
CONFIRMED
FALSE POSITIVE
koi_disposition
Box Plot of koi_fpflag_ss
CONFIRMED
FALSE POSITIVE
koi_disposition
Box Plot of koi_fpflag_co
CONFIRMED
FALSE POSITIVE
koi_disposition
Box Plot of koi_fpflag_ec
CONFIRMED
FALSE POSITIVE
koi_disposition
Box Plot of koi_period
CONFIRMED
FALSE POSITIVE
koi_disposition
Box Plot of koi_period_err1
CONFIRMED
FALSE POSITIVE
koi_disposition
Box Plot of koi_period_err2
CONFIRMED
FALSE POSITIVE
koi_disposition
Box Plot of koi_time0bk
CONFIRMED
FALSE POSITIVE

Figure 3.16: Boxplot for numerical columns

Box plots show the spread of each numeric column in relation to koi_disposition category. Every plot displays the median value of the numeric feature considering the disposition and quartiles and possible outliers could be seen. The purpose of this visualization is to examine deviations and potential anomalies of values concerning koi_disposition.

3.4 Model Training

Model training is the process of utilizing processed dataset to make an ML model by adjusting its parameters for learning from data. This often comprises of techniques like the division of the data set into the training and validation set with the selection of the algorithm then checking of the desired performance metrics to check for generalizability.

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.20)  # dividing data into train and test set
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Figure 3.17: Perform train and test split

The code splits the dataset into training and testing sets by applying train_test_split function available in scikit-learn package. In this case, X denotes the features, while Y denotes the target variable. The test_size=0. 20 parameter tell the fact that to split 80% data will be used for training set and 20% for test set. This split allows the testing of the model using unseen data in order to mark the efficiency of the model.

```
model_1 = LogisticRegression()    #implementing Logistic Regression
model_1.fit(X_train, y_train)     # fit function to train the data
```

Figure 3.18: Logistic Regression

The code starts with enabling Logistic Regression and creating a model that is named model_1. It then builds this model using the fit method on the training data that consist of features and target values of the training data namely X_train and y_train respectively. This process modifies the parameters of the model in a way as to maximize this relationship for the features and the target variable in the training data set.

- **Logistic Regression**: Logistic Regression is an algorithm for binary classification and it is highly used in the statistical field. The algorithm often calculates the likelihood of the occurrence of a binary event by the application of the logistic function , which maps predicted values to a probability between 0 and 1. The model is fit with the aid of

maximum likelihood estimation to estimate the maximum likelihood estimates with the aim of minimizing the divergence of the predicted values from the real ones. Logistic Regression is therefore preferred due to its simplicity, interpretability as well as efficiency in that its algorithm is ideal for problems that involve decision boundaries that are linear and does not involve a large number of features.

• **XGBoost Classifier**: XGBClassifier is used in XGBoost package and the XGBoost is a powerful tool for gradient boosting for classification. It builds a committee of decision trees end to end, in which each successive tree tries to rectify the mistakes made by the previous decision trees. Other features of XGBClassifier include the capability of employing the regularizations, boosting, as well as, handling of missing values in the best manner possible. It is considered being accurate, efficient and versatile and therefore it is mostly used when there are non-sequential dependencies between the variables of the dataset. Due to how XGBoost addresses large scale data and mixed features, the algorithm generally outperforms previous methods in terms of predictive accuracy.

• **CatBoost Classifier:** Another called CatBoostClassifier is a gradient boosting algorithm which is designed by Yandex to work well with categorical variables without complex one-hot encoding. It constructs a set of decision trees, by applying an ordered boosting which helps in fixing the problem of overfitting. Updating the parameters of CatBoost using symmetric trees and improving the boosting coefficients, it provides high accuracy in different datasets. From the feature extraction point of view, the categorical data integration with several encoding methods pre-built into the algorithm minimizes the amount of feature engineering needed. CatBoost distinguishing features are its high accuracy while working with categorical data, the model's simplicity, and high efficiency provided that there are many categorical features and interaction between them in the data.

• **RNN (Recurrent Neural Network**): RNNs are designed for Sequential data, in which, it is designed to rely on a history or memory which is built by one or more loops. This characteristic allows the RNNs to handle problems that need a time series, language processing or any situation where the order of the input matters. RNNs take data in a sequential manner and they adjust their state for each step which enables these models to capture temporal relationship. However, training of these models using

traditional RNNs can be problematic due to issues such as vanishing or exploding gradients.

• **GRU (Gated Recurrent Unit):** GRUs are another type of RNN that has been developed to overcome some of the issues associated with basic RNN models in relation to long-term memory update. Compared to other recurrent networks, GRUs incorporate gating mechanisms which helps in controlling the flow of information as well as memory management which enhances the capturing of long range dependencies. They make the architecture less complex than the LSTM networks as they combine the forget and the input gates in order to form the update gates. This makes GRUs less computationally demanding and still capable of capturing intricate sequential patterns, making them ideal for tasks such as speech recognition and time series predictions.

Chapter 4 System Design and Specifications

4.1 System Design

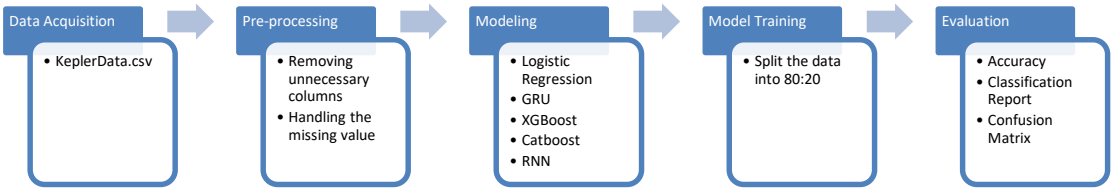


Figure 4.1: System Design

4.2 System Specifications

1. **Hardware**: The system adopts a high-performance computing platform involving a multi-core CPU and an enhanced GPU that enables faster training of the ML models. It is true that high speed and dependable network connection guarantee proper data exchange and communication.
2. **Software:** The development environment uses Python as the programming language, pandas for data handling, and NumPy for numerical computations, TensorFlow for model creation and model training, and scikit-learn for model assessment.
3. **Data:** The dataset applied for analysis and training of all models is called KeplerData, which contains all the necessary data for the training set and the test.

4.3 Modelling

In this study, the following machine learning models were utilised to make predictions by using the Kepler dataset; Logistic Regression, XGBoost, CatBoost, Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU). After training, validation, and evaluation, the performances of each model were assessed through the metrics involving accuracy, precision, recall, and F1-score. Such a comparison highlighted the general performance of these models and demonstrated that the highest accuracy was achieved by the ensemble methods and deep learning methods such as XGBoost and CatBoost.

4.3.1 Logistic Regression

Logistic Regression model is set up to assess and predict the interaction of independent variables with the dependent variable. In the training phase of the model, this model aims at finding the right position of the line or boundary that better separates them. This process involves searching for coefficients that will make the differences between the predicted and actual values as small as possible. After training, the model is capable of providing the prediction of the class of unseen data based on the learnt patterns and hence a desirable tool for binary classification.

4.3.2 XGBoost

An XGBoost model is used which is a form of gradient boosting algorithm that trains data in order to classify it and make more accurate prediction with each iteration. It carries out the construction of a set of decision trees which follow one another and each of the consequent trees correct errors of the previous tree. The final decision or prediction is arrived at after XGBoost combines the outputs of all these trees thus improving the accuracy of predictions and at the same time diminishing the cases of overfitting. Subsequent to this, the trained model is used to make predictions on new data hence using the learned patterns to make proper classifications on the input data.

4.3.3 CatBoost

The CatBoost model, which is an algorithm of gradient boosting that aims at the effective use of numerous categorical variables, draws a set of decision trees for data classification. It also automatically takes care of categorical data which do not need pre-processing during training, get better results and there is less over-fitting of the model. Such a model improves by concentrating on the mistakes which were made in the earlier rounds of the prediction model. In other words, the learned patterns from the training data are used to make correct estimations on the data that was not employed during the training process of CatBoost.

4.3.4 Recurrent Neural Network (RNN)

The next RNN model is designed sequentially in which a first Simple RNN layer comprises 32 units, which has a specific propensity towards sequential inputs. This is then succeeded by a densification layer with 10 units and applying ReLU activation

function to enable non-linearity and improvements on learning. The last layer is a dense layer with units = 4 and activation function = sigmoid that is useful for multi-class classification as it gives out the probability of belonging to any of the classes.
The model is compiled with sparse categorical cross entropy as the loss function in order to address the classification assignment of the given problem and we use the accuracy to measure the performance. It is trained for more than 10 epochs which includes a validation split of 2 %to test how the network perform on unseen data during training period. The model summary provides information regarding architecture with parameters, number of parameters in each layer and the type of layer used in the model.

4.3.5 Gated Recurrent Unit (RNN)

The GRU (Gated Recurrent Unit) model is built with the sequential structure; first, there is the first GRU layer with 32 units which function for processing the sequence data more efficiently than Simple RNN by capturing long-term dependencies. This layer is succeeded by a hence denser layer containing 10 units as well as ReLU activation function to inject non-linearity. The output layer contains 4 neurons and has sigmoid activation function, as the output neurons are used to produce scores of the classes in the case of multi-class classification. The model is trained using sparse categorical cross entropy since it is used for training multi-class classification problems and accuracy is used to monitor the model's performance. It is trained for 10 epochs, and a 2% of the data are separated for validation, this allows the model to check for its performance on new unseen data. This is a brief of the model, the amount of parameters and layers used in the model architecture.

4.4 User Interface

The practical part of the Kepler data project concerns the advanced user interface (UI) that is focused on improving the experience of working with astronomical datasets, especially those derived from the Kepler space observatory. Created using Shiny – a Python web application, this UI creates an interactive and immersive setting that allows the user to explore large data sets through a clean and accessible web interface.

The interface contains numerous input options such as fields, sliders, buttons and drop-down lists in order to enable users to sort, modify and visualize Kepler data within a

live environment. Elements like dynamic plots, charts, and graphs offer inherent feedback or response so that the users can clearly distinguish the patterns, trends or the anomalies in the data. This level of interactivity paves way for researchers who want to find out something new or assess the efficiency of bodies observed by Kepler mission.

The integration of uvicorn, a high throughput web server that works with asynchronous and synchronous tasks through ASGI enables the application to handle concurrent multiple users while maintaining high performance. Due to this, the UI is not only a great platform for astronomers, astrophysicists, and data scientists but also can be used for educational purposes by students of different ages to enable them to appreciate the complexity of the Kepler mission's results and rich datasets.

Chapter 5 Implementation

The system is implemented using Google Colab, which is a very suitable platform for running code in a cloud environment. The implementation relies on several key libraries:

- **Python**: The main language to code and operate the program, containing a large variety of libraries and utilities for data analysis and artificial intelligence functions.
- **Matplotlib:** A library for creating static, interactive, and animated visualizations in python which is most important when it comes to the plotting of graphical user interfaces to analyze model output and other data.
- **Scikit-learn**: An extensive collection of tools for using machine learning, including data processing and model training, such as logistic regression models and the XGBoost algorithm.
- **Model:** Here, I have used different models like logistic regression, Xgboost model, RNN Model, GRU Model and the Catboost Model.
- **TensorFlow:** An 'open-source, deep learning' mental model design tool that supports the creation and training of regressors and classifiers.

Table 1 below shows the specification of the essential parameters selected for the application of each of the models used in the study.

Model	Key Parameters
Logistic Regression	-
XGBoost	random_state=369
Catboost	random_state=369
RNN	units=32, activation='relu', epochs=10, Dense Units: 10 (hidden layer), 4 (output layer)
GRU	Units=32, activation='relu', epoch=10, Dense Units: 10 (hidden layer), 4 (output layer)

Table 1: Implementation of models

Chapter 6 Testing and Evaluation

Figure below show the accuracy of all models are as follows:

Model	Accuracy
Logistic Regression	93.18%
XGBClassifier	99.39%
CatBoostClassifier	99.46%
RNN	80.36%
GRU	98.79%

Table 2: Accuracy of all models

The table presents the performance metrics for different models, showing the accuracy of each. The CatBoostClassifier achieved the highest accuracy at 99.46%, indicating its superior performance. In contrast, the RNN had the lowest accuracy of 80.36%, highlighting its comparatively lower effectiveness in this analysis.

Classification Report:

	precision	recall	f1-score	support
0	0.95	0.94	0.95	936
1	0.90	0.92	0.91	546
accuracy			0.93	1482
macro avg	0.92	0.93	0.93	1482
weighted avg	0.93	0.93	0.93	1482

a) Classification Report for Logistic Regression

Classification Report:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	936
1	1.00	0.99	0.99	546
accuracy			0.99	1482
macro avg	0.99	0.99	0.99	1482
weighted avg	0.99	0.99	0.99	1482

b) Classification report for XGBoost

Classification Report:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	936
1	1.00	0.99	0.99	546
accuracy			0.99	1482
macro avg	1.00	0.99	0.99	1482
weighted avg	0.99	0.99	0.99	1482

c) Classification Report for Catboost

Classification Report:

	precision	recall	f1-score	support
0	0.78	0.95	0.86	936
1	0.87	0.55	0.67	546
accuracy			0.80	1482
macro avg	0.83	0.75	0.77	1482
weighted avg	0.82	0.80	0.79	1482

d) Classification Report for RNN

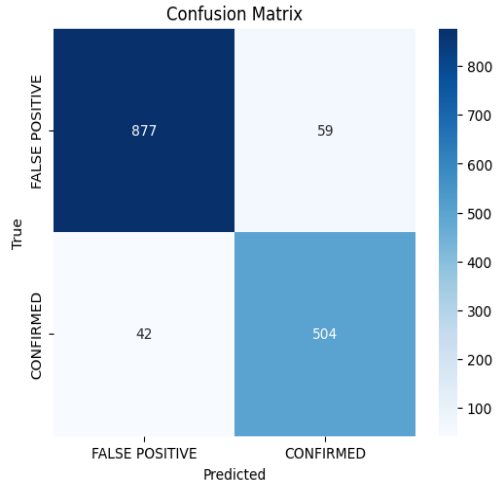
Classification Report:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	936
1	0.98	0.98	0.98	546
accuracy			0.99	1482
macro avg	0.99	0.99	0.99	1482
weighted avg	0.99	0.99	0.99	1482

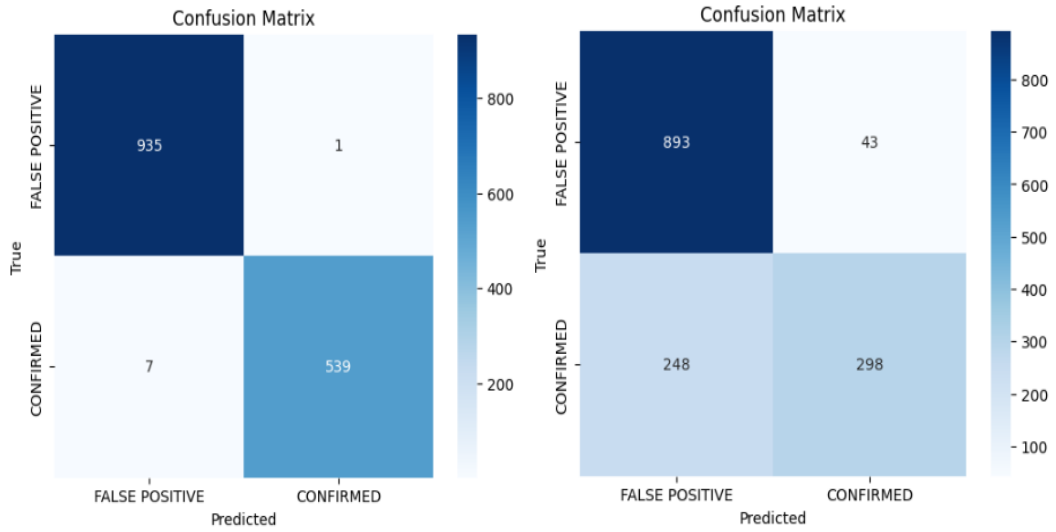
e) Classification Report for GRU

Figure 6.1: Classification Report for all models

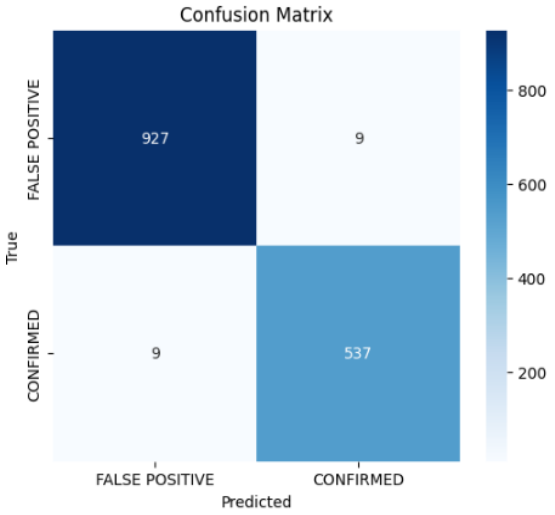
The classification report reveals that the XGBClassifier and CatBoostClassifier excel with accuracy rates of 99.39% and 99.46%, respectively, indicating their superior performance compared to other models. Logistic Regression achieved an accuracy of 93.18%, showing strong precision and recall. The RNN model had the lowest accuracy at 80.36%, reflecting less effective performance in comparison. The metrics underscore the XGBClassifier and CatBoostClassifier as the most reliable for this dataset, while the RNN model lags in accuracy and overall classification effectiveness.



a) Confusion matrix for logistic Regression b) Confusion matrix for XGBoost



c) Confusion matrix for Catboost d) Confusion matrix for RNN

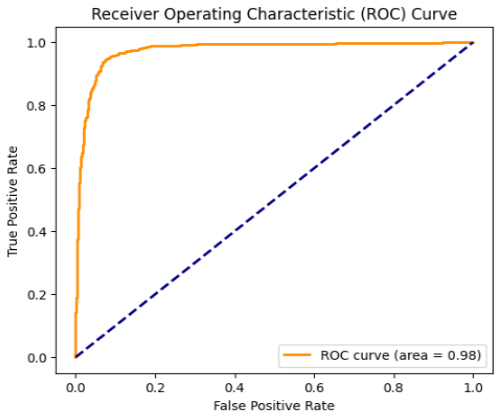


e) Confusion matrix for GRU

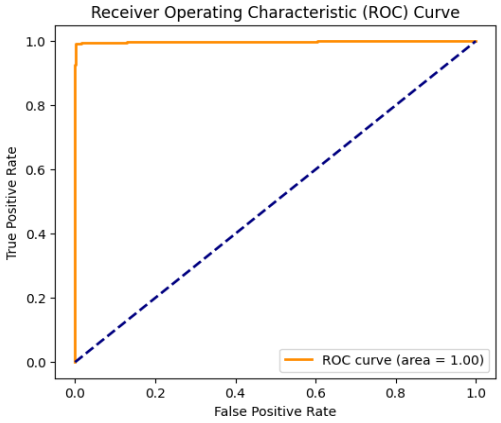
Figure 6.2: Confusion matrix for all models

The confusion matrices for all models illustrate their classification performance by displaying true positives, true negatives, false positives, and false negatives. The XGBClassifier and CatBoostClassifier demonstrate minimal misclassification, reflected in high counts of true positives and true negatives, aligning with their high accuracy scores. On the other hand, the one with lower Accuracy, namely the Logistic Regression model, delivers a higher percentage of misclassified samples, especially on both false positives as well as false negatives. The RNN model, with its lower accuracy,

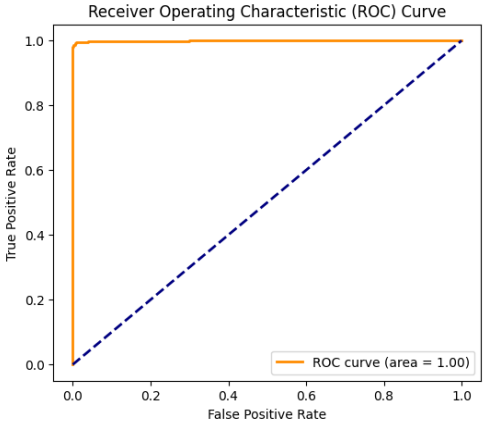
exhibits more pronounced misclassification errors, highlighting its reduced effectiveness in this context.



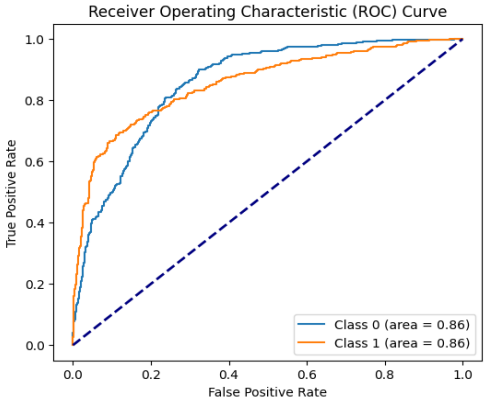
a) Roc curve for Logistic Regression



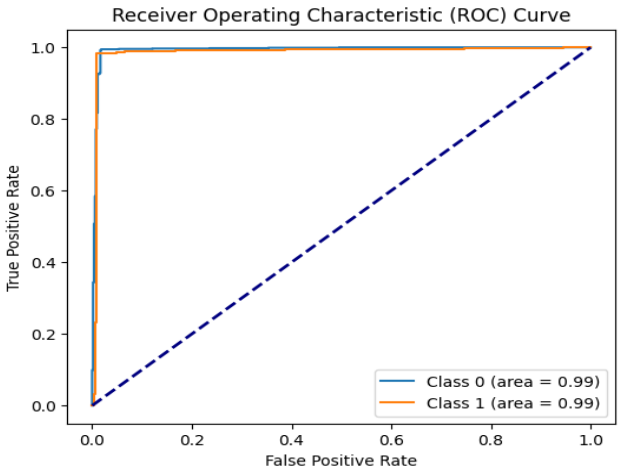
b) Roc curve for XGBoost



c) Roc curve for Catboost



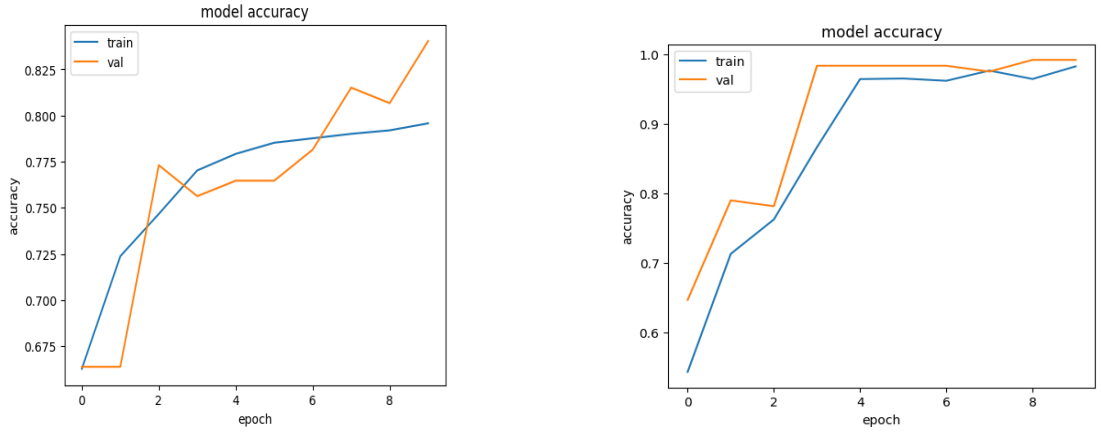
d) Roc Curve for RNN



e) Roc Curve for GRU

Figure 6.3: Roc Curve for all models

The ROC curves of all models show how they perform with respect to different threshold levels by plotting the true positive rate (TPR) against the false positive rate (FPR). The closer the curve of the measured models to the upper left corner, the higher its performance. The XGBClassifier and CatBoostClassifier exhibit points near the top-left corner proving high discriminative accuracy. The ROC curve of the Logistic Regression model is little farther from the ideal one, which indicates lesser discriminatory power of this model. This proves that the RNN model has a less favorable ROC curve as a consequence of low accuracy and comparatively higher misclassification rates.



a) Model accuracy plot for RNN b) Model accuracy plot for GRU

Figure 6.4: Training Curves for the RNN and GRU

In the model accuracy plot of RNN and GRU, the training performance of each model is illustrated for showing an evolution of the accuracy in the training process. The GRU model normally indicates a quicker convergence to a higher level of accuracy than the RNN owing to its effectiveness in the training process. The RNN plot may demonstrated slower or less stable accuracy improvements which will reveal the problem that RNN have in working with sequences compared to GRU. For both the plots, it is possible to assess the stability and accuracy of the models in their training phase

6.1 User Interface

Figure below show the user interface for kepler data:

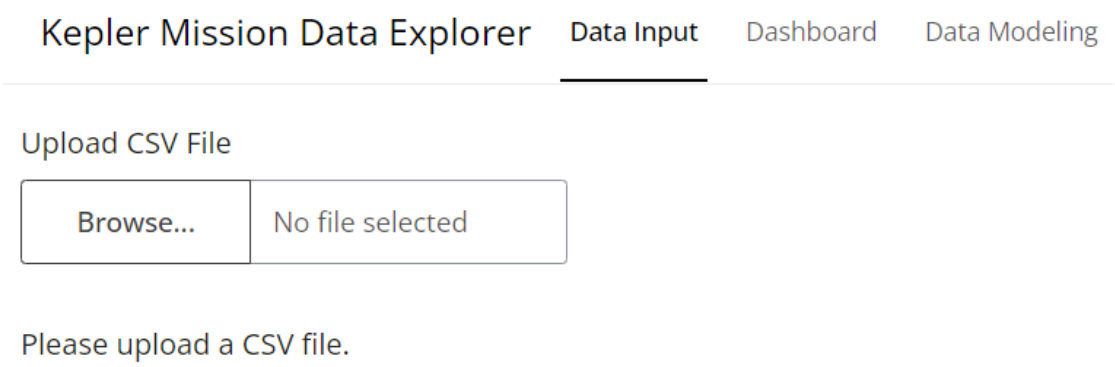


Figure 6.5: Kepler Mission Data Explorer

The Kepler Mission Data Explorer UI is very well thought out so as to provide a systematic and easy approach to analyze astronomical data by the Kepler space telescope. ELA stands for Efficient Log Analysis and is the first component of the Kepler mission which enables users to upload their data in CSV format. It has a straight-forward upload button which is labeled 'No file selected' with options to upload the user's data files in a hassle-free manner.

Once a user completes a upload, she or he is taken to the Dashboard which provides an in-depth and graphical analysis of the dataset. This dashboard is enriched with the figure and graphic elements as graphs, scatter and histograms, and it is interactive; the user is able to influence the displayed information and its representation as well.

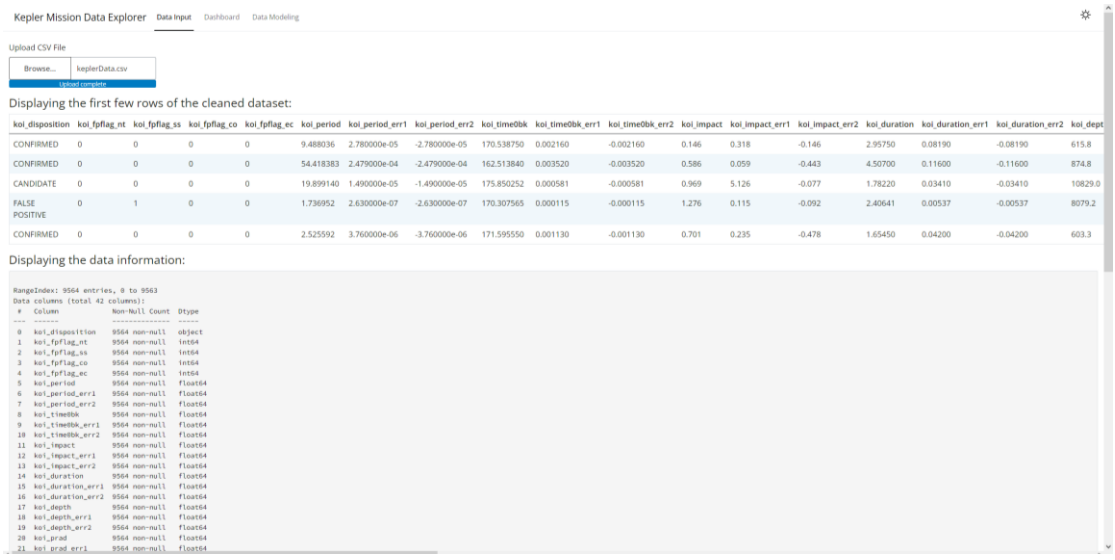


Figure 6.6: Dataset information as Visualised in UI

This is the Data Modeling section where in users can perform more complex analytical models and algorithm to analyze data. This area allows the user to conduct simulation and statistical analysis and use the machine learning methods to improve understanding of the results and make correct conclusions.

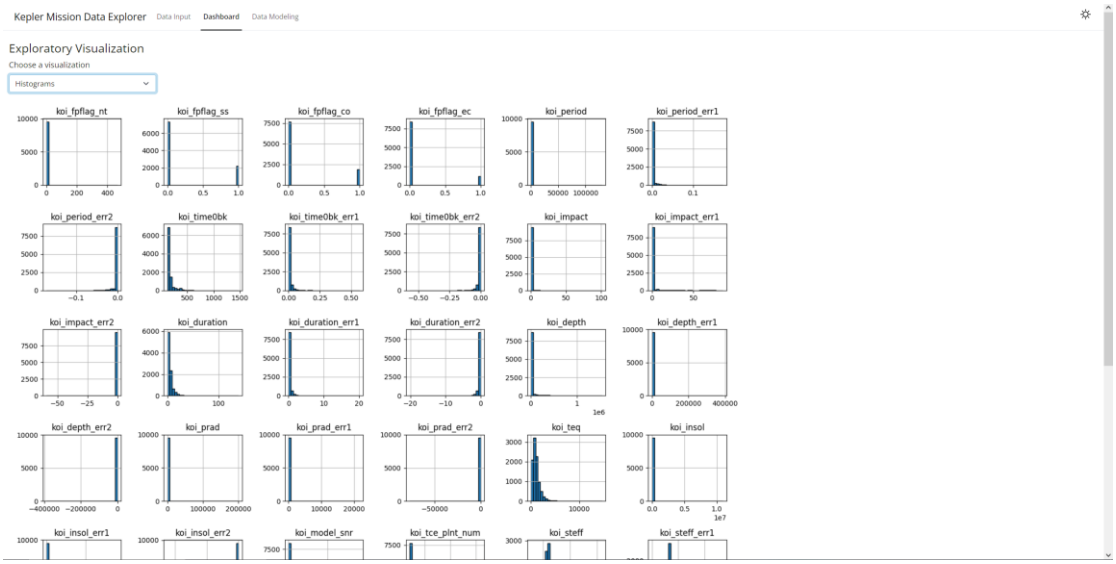


Figure 6.7: Exploratory Visualisation in UI

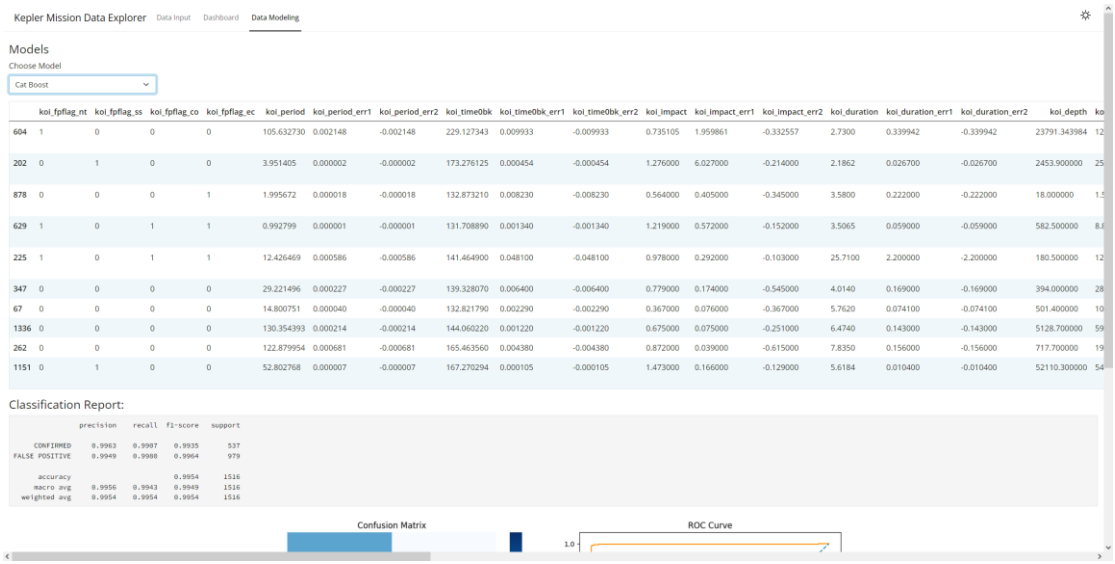


Figure 6.8: Data Modeling Results in UI

In general, the UI is effective at facilitating navigation from data entry to analysis and advances the prospect of effectively mining Kepler's vast astronomical data sets.

Figure 6.9 below shows the classification result using the Logistic Regression for given set of Kepler KOI data for a False Positive sample. All the models showed the same results for the prediction, which is correct.

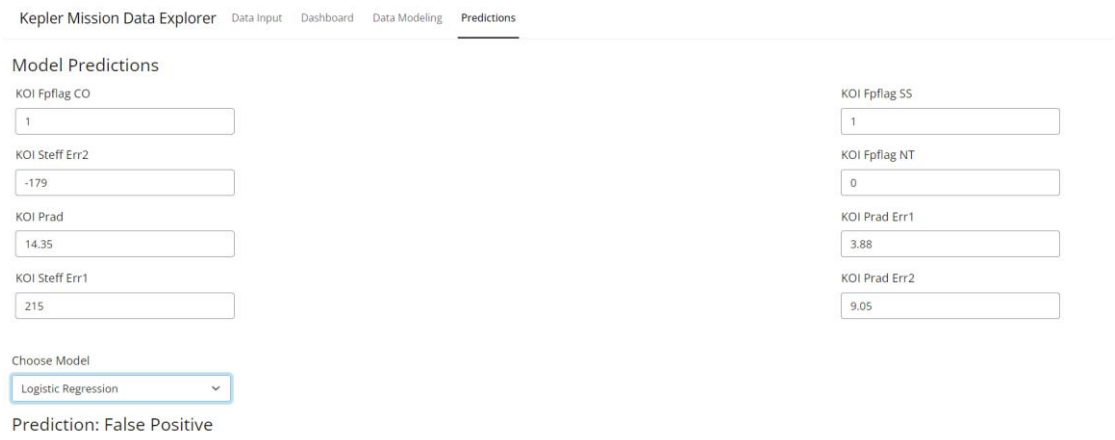


Figure 6.9: Prediction Results in UI for a False Positive Sample

Figure 6.10 below shows the classification results of the CatBoost classifier for a Confirmed KOI sample. The results obtained for all the models coincide with the dataset.

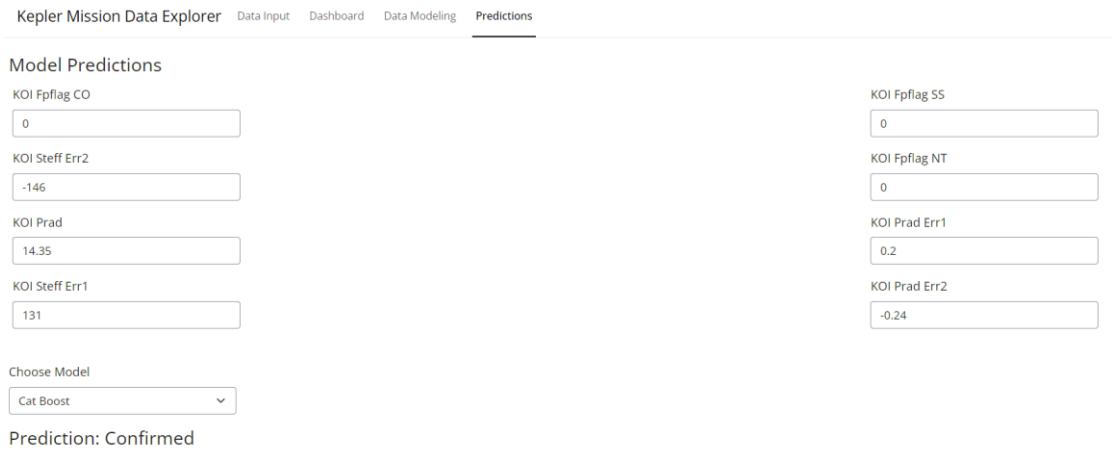


Figure 6.10: Prediction Results in UI for a Confirmed Sample

Chapter 7 Conclusions and Future Work

The purpose of this study was to compare accuracy and efficiency of various machine learning techniques in the scenario of outcome prediction depending on the collected data set. These models are as follows – Logistic Regression, XGBClassifier, CatBoostClassifier, RNN and GRU were chosen freely for what kind of info processing methods they utilize to get specific patterns.

Logistic Regression is less accurate with a mean accuracy of 93. 18%. What is more, this interpretable and less complex model enabled us to obtain rather good performance in terms of binary classification tasks. Though it performed well in dealing with all the feature where linear relation exists between them, it fails to capture all the interaction that exist in the dataset as it was not capable of modeling non-linear interactions between features. Due to the simple nature of Logistic Regression model, applying the model as a first-step in analysis is helpful; but, the above results support the case that when analyzed data is complex, more elaborate models are required.

XGBClassifier and CatBoostClassifier outperformed all the other models with accuracy of 99. 39% and 99. 46%, respectively. These gradient boosting algorithms are widely appreciated as they are capable of fitting even very non-linear relationships and interactions of features. Thus, the high accuracy of these models demonstrates positive effects of boosting in increasing accuracy in predicting results. They both used sophisticated approaches including boosting and feature engineering which led to high performance of the two models. However, they also entail significant computational requirements and the need for proper hyperparameter selection for the best outcomes, showing that these models are intricate.

The RNN (Recurrent Neural Network) achieved an accuracy of 80. 36%, which are relatively high. Even though RNNs have been proposed to tackle temporal dependencies, they suffer from such problems as vanishing gradients particularly where long sequences are involved. This limitation was seen in the outcomes as it unveils the fact that although RNNs are useful in sequence-based tasks they are not always the most optimal solution for any kind of data.

However, the GRU (Gated Recurrent Unit) model had a significant increase at an accuracy of 98. 79%. The gating mechanisms given in the architecture of the GRU helped in various problems like vanishing gradients and such it demonstrated more

efficiency for sequential data than the RNN. Such superior performance demonstrates the ability of the GRU to capture intricacies of dependencies and temporal behavior inherent in the available data. Since the proposed GRU model has performed well on the given problem it is quite clear that GRU model works well when the problem involves sequences and the time series data.

A well-designed UI was particularly critical for this project and needed to meet certain established criteria. The implemented user interface provided the users with an access to the functionality as well as the results of the predictive models and datasets that do not require possessing any programming skills. This was important in order to expand the user base as far as possible and accommodate users who were not technologically inclined. From the feedback collected during testing it was clear that the UI design was good and provided feasible and efficient impacts to usability and functionality of the system.

Thus the study revealed that algorithm ensemble methods such as XGBoost and CatBoost yields high predictive accuracy and on the similar manner, deep learning methods specifically GRUs yields great improvements for sequential data. This leaves the approach of choosing the correct model according to the nature and type of data as underlined in the study. Every model serves to demonstrate capabilities and shortcomings of one or another approach, which is rather helpful when it comes to understanding its practical application.

7.1 Future Work

In terms of future research, several directions can be proposed to advance on the results of the present study and overcome its limitations. First, supplementing the model with more sources or higher-level properties may improve its predictive ability or even overcome slight biases. It is also important to note that possibly obtaining more data from external or supplementary sources can be useful to increase the amount of data available for analysis in order to obtain more general models valid for different cases.

Additional development of more complex architectures like transformers or attention models for sequence data might provide better results. These models have emerged some contrary findings in various applications they might out compete the current RNN and GRU models in capturing complex patterns and dependencies.

Furthermore, a more significant improvement of the methods for hyperparameter optimization will help in achieving the level of high performance when working with gradient boosting models. The use of automated machine learning (AutoML) may be useful in optimizing configurations as well as enhance the performance of the models more effectively.

On future updates to the project, greater improvements to the UI would be possible. This also concerns enhancement of the UI in terms of its responsiveness so that it responds to different devices and screen sizes appropriately. Incorporation of features that are more engaging such as animated displays of real-time data, and superior customization could add value to the users. Also, using AI recommendations or assists appearing directly on the UI might help users take better decisions grounded on the model results. Solving these issues may cause a substantial enhancement of the general suitability and efficiency of the system as for the end-users.

Some practical issues, such as computational efficiency and scalability are also important. The authors found that researching ways for lowering the training time and overall resources required for models like pruning, distillation, or distributed training would help make the models significantly more practical for mass and real application. Thus, future research can improve the static and dynamic precise and applicability of the predictive models to solve various range of problems for different types of data.

References

1. Abid, F., Li, C. and Alam, M., 2020. Multi-source social media data sentiment analysis using bidirectional recurrent convolutional neural networks. Computer Communications, 157, pp.102-115.
2. Aseeri, A.O., 2023. Effective RNN-based forecasting methodology design for improving short-term power load forecasts: Application to large-scale power-grid time series. Journal of Computational Science, 68, p.101984.
3. Bairouk, A., 2023. Astronomical Image Time-series Classification Using Deep Learning (Doctoral dissertation, Université de Montpellier).
4. Baron, D., 2019. Machine learning in astronomy: A practical overview. arXiv preprint arXiv:1904.07248.
5. Batalha, N.E., Smith, A.J., Lewis, N.K., Marley, M.S., Fortney, J.J. and Macintosh, B., 2018. Color classification of extrasolar giant planets: Prospects and cautions. The Astronomical Journal, 156(4), p.158.
6. Bentéjac, C., Csörgő, A. and Martínez-Muñoz, G., 2021. A comparative analysis of gradient boosting algorithms. Artificial Intelligence Review, 54, pp.1937-1967.
7. Che, Z., Purushotham, S., Cho, K., Sontag, D. and Liu, Y., 2018. Recurrent neural networks for multivariate time series with missing values. Scientific reports, 8(1), p.6085.
8. Dornigg, T., 2021. Credit Risk Modeling: Predicting Customer Loan Defaults with Machine Learning Models (Master's thesis, Universidade NOVA de Lisboa (Portugal)).
9. Dumitrescu, E., Hué, S., Hurlin, C. and Tokpavi, S., 2022. Machine learning for credit scoring: Improving logistic regression with non-linear decision-tree effects. European Journal of Operational Research, 297(3), pp.1178-1192.
10. Faaique, M., 2024. Overview of big data analytics in modern astronomy. International Journal of Mathematics, Statistics, and Computer Science, 2, pp.96-113.
11. Ge, J., Zhang, H., Zang, W., Deng, H., Mao, S., Xie, J.W., Liu, H.G., Zhou, J.L., Willis, K., Huang, C. and Howell, S.B., 2022. ET White Paper: to find the first Earth 2.0. arXiv preprint arXiv:2206.06693.

12. Howard, F.M., Dolezal, J., Kochanny, S., Schulte, J., Chen, H., Heij, L., Huo, D., Nanda, R., Olopade, O.I., Kather, J.N. and Cipriani, N., 2021. The impact of site-specific digital histology signatures on deep learning model accuracy and bias. Nature communications, 12(1), p.4423.
13. Jiao, W., Hao, X. and Qin, C., 2021. The image classification method with CNN-XGBoost model based on adaptive particle swarm optimization. Information, 12(4), p.156.
14. Karim, A., Uddin, J. and Riyad, M.M.H., 2024. Identifying Important Features For Exoplanet Detection: A Machine Learning Approach. GPH-International Journal of Applied Science, 7(01), pp.01-17.
15. Kasongo, S.M., 2023. A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework. Computer Communications, 199, pp.113-125.
16. Li, C., Zhang, Y., Cui, C., Fan, D., Zhao, Y., Wu, X.B., Zhang, J.Y., Han, J., Xu, Y., Tao, Y. and Li, S., 2022. Photometric redshift estimation of BASS DR3 quasars by machine learning. Monthly Notices of the Royal Astronomical Society, 509(2), pp.2289-2303.
17. London, J.J., 2021. Advances in Machine Learning: Theory and Applications in Time Series Prediction. Illinois Institute of Technology.
18. Morvan, M., Nikolaou, N., Tsiaras, A. and Waldmann, I.P., 2020. Detrending Exoplanetary Transit Light Curves with Long Short-term Memory Networks. The Astronomical Journal, 159(3), p.109.
19. Nigri, E. and Arandjelovic, O., 2017, June. Light curve analysis from Kepler spacecraft collected data. In Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval (pp. 93-98).
20. Prasantha, H.S., MACHINE LEARNING APPROACHES FOR EXOPLANET EXPLORATION FROM KEPLER-LIGHT CURVES.
21. Sun, A.Y. and Scanlon, B.R., 2019. How can Big Data and machine learning benefit environment and water management: a survey of methods, applications, and future directions. Environmental Research Letters, 14(7), p.073001.
22. Yu, C., Li, K., Zhang, Y., Xiao, J., Cui, C., Tao, Y., Tang, S., Sun, C. and Bi, C., 2021. A survey on machine learning based light curve analysis for variable

astronomical sources. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 11(5), p.e1425.

23. Wieland, R., Lakes, T. and Nendel, C., 2021. Using Shapley additive explanations to interpret extreme gradient boosting predictions of grassland degradation in Xilingol, China. Geoscientific Model Development, 14(3), pp.1493-1510.

Appendix A

A.1 Python code for Logistic Regression

```
model_1 = LogisticRegression()
model_1.fit(X_train, y_train)
```

A.2 Python code for XGBoost

```
xgb = XGBClassifier(random_state=369) #Implementing xgboost model
xgb.fit(X_train, y_train)
pred = xgb.predict(X_test)
```

A.3 Python code for Catboost

```
cat = CatBoostClassifier(random_state=369) #Implementing catboost model
cat.fit(X_train, y_train)
```

A.4 Python code for RNN

```
# RNN model initalization
keras.utils.set_random_seed(0)
rnn = keras.Sequential([
    layers.SimpleRNN(32, input_shape=(X_test.shape[1], 1)),
    layers.Dense(10, activation='relu'),
    layers.Dense(4, activation='sigmoid')
])
rnn.compile(loss="sparse_categorical_crossentropy", metrics=["accuracy"])
rnn.summary()
history = rnn.fit(X_train, y_train, epochs = 10, validation_split=0.02)
```

A.5 Python code for GRU

```
# GRU model initalization

gru = keras.Sequential([
    layers.GRU(32, input_shape=(X_test.shape[1], 1)),
    layers.Dense(10, activation='relu'),
    layers.Dense(4, activation='sigmoid')
])
gru.compile(loss="sparse_categorical_crossentropy", metrics=["accuracy"])
gru.summary()

history = gru.fit(X_train, y_train, epochs = 10, validation_split=0.02)
```