Modelling Star Types to Predict Black Hole Formation: Insights for Astronomical Detection

Janesh Hasija
University of Maryland, College Park
College Park, USA
jhasija@umd.edu

***Abstract*—This research focuses on estimating the possibility for black hole generation in stars by categorising them as 'Dwarf' or 'Giant'. The dataset includes important star characteristics such as visual apparent magnitude (Vmag), distance from Earth (Plx), the B-V colour index, and stellar classifications (SpType). Methodologically, the study preprocessed data substantially, dealt with missing values, eliminated outliers, and included a new 'Temperature' feature based on the B-V colour index. Several classification models were used and assessed, including Logistic Regression, Decision Trees, K-Nearest Neighbours (KNN), Gaussian Naive Bayes, and Linear Discriminant Analysis. The results highlighted the robustness of the Logistic Regression, K- Nearest Neighbor and Linear Discriminant Analysis model in predicting star kinds, with significant accuracies and AUC ratings. However, Gaussian Naive Bayes also performed well, warranting more investigation. The relevance of this discovery rests in its ability to assist astronomers by giving predictive models for identifying stars with possible black hole development tendencies. Early identification might have a tremendous influence on astronomical research, providing more information about star development and black hole generation.**

***Index Terms*—Decision Tree, Logistic Regression, area under curve, Star Dataset, K-Nearest Neighbors, Naive Bayes Classifier, Linear discriminant analysis**

I. Introduction

The application of machine learning algorithms has become essential in today's decision-making processes, offering insightful analysis and forecasts for challenging issues. This code uses various classification models to predict whether the star is a dwarf or giant. The model choice has to be made based on whether the star after collision will create the black hole or not.

The study of star classifications and their propensity to grow into black holes is an important astronomical question. This study tackles the choice issue of properly classifying stars as 'Dwarf' or 'Giant,' with the goal of predicting their likelihood of black hole creation. The prediction models created here are intended to be a trailblazing tool for early detection in the astronomical realm.

The judgement has significant significance for astronomers and astrophysicists. Early detection of stars with the potential to create black holes allows for more concentrated studies and in-depth research, providing a window into the lifespan of stars and the mechanics behind black hole development. This essential decision-making process substantially advances astronomy study and expands our understanding of celestial bodies.

A. *Data set*

Stellar Classification divides stars into distinct groups based on their spectral data. The Morgan-Keenan (MK) classification system is the contemporary stellar classification system. It use the traditional HR classification method to classify stars based on their chromaticity and Roman numerals to classify stellar size. The datset contains 99999 rows and 6 columns but after pre-processing of data it went down to 32018 rows and 7 columns. The information used in this research includes essential star observational parameters such as visual apparent magnitude (Vmag), distance from Earth (Plx), B-V colour index, and stellar classifications (SpType). The target class has 0 or 1 values which represent whether the star is dwarf or not. Amag represents Absolute magnitude of the star and B-V(B_V) represents the color index. The temperature column consists of the temperature of the individual stars which is calculated on the given formula. Vmag represents the visual magnitude of the star whereas Plx represents the distance between the star and the earth and e_plx represents the error of plx respectively. The SpType column gives information about spectral classification where Roman Numerals ¿= III are giants and ¡III are dwarfs. This dataset serves as the foundation for training and evaluating prediction algorithms designed to detect patterns and traits indicative of possible black hole generation.[1]

B. *Literature Review*

This finding is supported by a thorough literature assessment, which acknowledges past studies in star categorization and black hole development prediction. It builds on prominent publications that have investigated comparable approaches or issue fields. This research tries to improve on known methodology or offer fresh ways for more accurate forecasts by synthesising and analysing previous studies, therefore contributing to breakthroughs in astronomical observations and predictions.[2]

This article describes the data pre treatment processes, model selection, and assessment criteria for Logistic Regression, Decision Trees, K-Nearest Neighbours (KNN), Gaussian Naive Bayes, and Linear Discriminant Analysis in the following parts. Following that, the study gives the data, discussion,

and conclusions resulting from these models' predictions of star types indicative of black hole creation potential.

II. Methodology

The star dataset is a huge dataset and requires cleaning and preprocessing of dataset. For the better prediction of the model, the various pandas inbuilt function are used to clean the data. The dataset is trained on the classification models like Decision Trees, Logistic Regression, K-Nearest Neighbor, Gaussian Naive Bayes Linear Discriminant Analysis and then ROC curve is plotted and the area under the curve is calculated for all the models. The ROC Curve is a plot between true positive rate (TPR) vs the false positive rate (FPR) at different classification thresholds. The efficiency of the model is shown by the ROC AUC score. The model's ability to discriminate between the positive and negative classes is greater when the AUC is larger. A classifier that achieves an AUC value of 1 is said to be entirely capable of differentiating between all Positive and Negative class points. All Negatives are predicted by the classifier to be Positives, and vice versa, when the AUC value is 0.

A. Versions Used

The methodology employed in this study is grounded in Python, utilizing version 3.11.2, and relies on crucial libraries such as NumPy (version 1.25.2), sklearn (version 1.3.0), and Pandas (version 2.1.0) for data manipulation and analysis.

B. Pre-Processing Data

The function begins by using Pandas to load the Star99999_raw.csv dataset. It checks the structure of the dataset and looks for missing values with isnull().sum(). To guarantee a clean dataset for analysis, rows with missing values are eliminated using dropna() and the new dataset is stored in **raw** variable.

```
raw.loc[:,'Vmag']=pd.to_numeric(raw['Vmag'],
downcast='float',errors='coerce')
raw.loc[:,'Plx']=pd.to_numeric(raw['Plx'],
downcast='float',errors='coerce')
raw.loc[:,'B-V']=pd.to_numeric(raw['B-V'],
downcast='float',errors='coerce')
```

The above code appears to try to convert columns in a DataFrame called raw to numeric data types while managing problems using the pd.to_numeric() method. Let's dissect it:

The line changes the DataFrame raw's 'Vmag','Plx','e_plx' column to numeric data. If feasible, the downcast='float' argument tries to downcast the data to a smaller numeric type, whilst errors='coerce' converts problematic entries to NaN (Not a Number) values if they can't be converted to numeric. These procedures appear to be part of data preparation, where certain columns are converted to an appropriate numeric data format, maybe for subsequent analysis or modelling. The errors='coerce' argument will assist manage any problematic or non-convertible elements by converting them to NaN values, maintaining the DataFrame's overall integrity.

To avoid naming conflicts, columns are renamed ('BV' to 'B_V'). pd.to_numeric() is used to transform the data types of the columns, notably those relating to features such as Visual Apparent Magnitude (Vmag), Distance between the Star and the Earth (Plx), and the B-V colour index (B_V).

```
raw.query("Plx==0")
raw = raw.query('Plx!=0')
raw.shape
raw.query('Plx==0')
```

The first line queries the DataFrame raw to find rows with the 'Plx' column equal to zero. It returns rows in which the parallax ('Plx') value is equal to zero.

The second line reassigns the DataFrame raw by removing rows when the 'Plx' column is greater than zero. Essentially, it is modifying raw to include only rows with a 'Plx' greater than zero.

The third line function returns the shape of the DataFrame raw following the filtering procedures. The number of rows and columns in the DataFrame is returned and using dropna() function to remove value with zero if present and the new dataset is stored in new_data variable.

Now, to classify the data some more features are required to classify the star. So, Amag column is added by calculating the absolute magnitude of the star by using the formula:

```
M= m+5(log P+ 1)
```

where M is the Absolute Magnitude m is the visual apparent magnitude of the star P is the Plx value that represents the distance between the star and the earth

```
new_data['Amag'] = new_data.Vmag +
5 *(np.log10(((new_data.Plx)))+1)
```

This above code adds the new column Amag with the help of the above formula.

Temperature is an important element in defining a star's type, brightness, and development. When a black hole consumes material from a nearby star, an accretion disc is formed. This accretion disc might get incredibly hot and generate light.

The method may be used to calculate the effective temperature of a star based on its B-V colour index. The B-V colour index measures the brightness difference between a star's blue and visible light. It is calculated by subtracting the star's visual magnitude from its blue magnitude.

```
df['Temperature'] = 7090/(df.B_V + 0.72)
```

In order to classify the star there is a need to categorize star first on the basis of values provided in SpType. Using the below function to categorize star into three categories Dwarf, Giant and other using the code below:

```
def label_gen_stars(star):
    dwarf = ['D','VI', 'VII', 'V']
    giant = ['IV', 'III', 'II',
    'Ib', 'Ia', 'Ia-O']
    for i in dwarf :
```

```
        if i in star:
            return 'Dwarf'
    for i in giant:
        if i in star:
            return 'Giant'
    return 'Other'
new_data['Star_Type'] = new_data.SpType.
apply(label_gen_stars)
```

The function accepts a parameter star, which probably denotes a star's categorization type ('IV', 'III', etc.). Based on the specified classification, the function returns a label indicating whether the star is categorised as a "Dwarf," "Giant," or "Other."

This function appears to be part of a process in which stars are classified into wider groups such as dwarfs, giants, or others depending on their categorization types. It's most usually used to build or update a Data Frame column that summarises star kinds into these bigger categories for analysis or modelling.

```
new_data['Star_Type'] = new_data.SpType.
apply(label_gen_stars)
new_data.Target.value_counts()
```

The first line categorises stars based on their classification types ('SpType') into larger categories such as 'Dwarf', 'Giant', or 'Other' and stores this categorization in the 'Star_Type' column. The second line then counts the values in the 'Target' column, indicating how many stars fit into each category following the preceding 'SpType' classification procedure.

```
new_data['Star_Type'] = new_data.SpType.
apply(label_gen_stars)
new_data.Star_Type.value_counts()
new_data = new_data.query
('Star_Type■!=■"Other"')

new_data.query('Star_Type■==■"Other"')

new_data.Star_Type.value_counts()
```

This series of actions categorise stars based on categorization kinds, filter out any stars classified as 'Other,' and offer a count of 'Dwarf' and 'Giant' categories for prediction of black hole, focusing solely on these two categories.

```
new_data['Target'] = np.where(new_data.Star_
== 'Giant', 0,1)
new_data.Target.value_counts()
```

The code effectively constructs a binary classification label ('Target') in which 'Giant' stars are classed as 0 and all other kinds as 1. This type of modification is frequently done to prepare data for classification tasks when the model is attempting to predict a binary outcome.

Outliers are found by filtering data points that fall outside of the estimated whiskers (lw and rw) with the Interquartile Range (IQR) technique. Using boolean indexing (outliers.index), these outliers are then deleted from the dataset. The completed dataset is saved in variables such as df and final.

```
summary = new_data.describe().T
summary['IQR']=summary['75%']-summary['25%']
summary.head()

summary['IQR']=summary['75%']-summary['25%']


summary['cutoff']=round(summary.IQR*1.6, 3)
summary.head()

summary['lw']=round(summary['25%']-
summary.cutoff, 3)
summary['rw']=round(summary['75%']
+summary.cutoff, 3)
summary.head()
```

The first line of above code computes descriptive statistics for the numeric columns in the DataFrame 'new_data' and transposes it to produce a DataFrame with summary statistics.

The Second line of above code calculates the interquartile range (IQR) for each numeric column is calculated by subtracting the 25th percentile (Q1) from the 75th percentile (Q3). This step aids in determining the range of values around the median.

A cutoff value is calculated here. The IQR is multiplied by 1.6 to determine the outlier detection threshold. This multiplication factor (1.6) is frequently used to establish an outlier detection threshold based on the IQR.

The lower (lw) and upper (rw) whiskers are calculated using these lines for outlier detection. The lower whisker (lw) is determined by subtracting the cutoff value from the 25th percentile, whereas the upper whisker (rw) is determined by adding the cutoff value to the 75th percentile. This stage establishes the limits beyond which data points are considered outliers using the IQR approach.

The overall objective of these computations is to determine the range in which the majority of data points reside and to detect any outliers that fall outside of these bounds. This procedure aids in identifying and filtering outlier data items that may skew analysis or modelling outcomes.

```
# create a df with outliers
outliers=pd.DataFrame
(columns=new_data.columns)

#loop to detect outliers in each column
for col in summary.index:
    lower=summary.at[col,'lw']
    upper=summary.at[col,'rw']
    results=new_data[(new_data[col]<lower)|
```

```
        (new_data[col]>upper)].copy()
    results['Outlier']=col
    outliers=outliers.append(results)

outliers.shape
```

The method appears to be aimed at locating and collecting outlier data points across several columns of 'new_data' depending on previously determined whisker boundaries, and then eliminating these outliers from the dataset. The structure of 'new_data' after outliers are eliminated is presented to demonstrate how many rows were deleted as a result of this outlier identification method.

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
sns.heatmap(new_data.isnull(),
cbar=False, cmap='viridis')
plt.title('NaN Values in DataFrame')
plt.show()
```

Seaborn is a fantastic visualisation toolkit for statistical graphics charting in Python. It has excellent default styles and colour palettes to make statistics charts more appealing. It is designed on top of the matplotlib software and is tightly connected with pandas data structures [4].The resultant heatmap shows the locations of missing values in the 'new_data' DataFrame. The colour of each cell denotes whether or not the associated value in the DataFrame is missing (NaN or null). This visualisation aids in the identification of missing data patterns or concentrations across distinct columns or rows in the dataset.

```
new_data.isnull().sum()
final=new_data.dropna()
```

The isnull().sum() function returns the number of missing values in each column, whereas dropna() eliminates rows with missing values and puts the resultant DataFrame in final. This step may result in a reduction in the total number of rows in the dataset by deleting any rows with missing values.

C. Processing Data

Parameters

```
X_train, X_test, y_train,
y_test = train_test_split
(final.drop('Target', axis = 1),
final.Target, test_size = 0.25,
random_state = 42,
stratify = final.Target)
```

final.drop('Target', axis = 1: The 'Target' column is the only one that is not chosen as the feature set (X) in this section.

final.Target: This designates the target variable (y) as the 'Target' column.

train_test_split: The dataset is divided into training and testing sets using this function.

test_size = 0.25: It states that testing will utilise 25% of the data, while training will use the remaining 75%.

random state = Reproducibility is ensured by setting a seed for the creation of random numbers. Every time the code is executed, the same seed (42 in this case) will partition the data in the same manner.

stratify = final.Target: By using this option, you may be guaranteed that the training and testing sets include the same percentage of target classes as the original dataset did during the splitting process. To guarantee that all classes are represented in both sets, it is especially helpful for datasets that are unbalanced.

```
scaler = StandardScaler()
scaler.fit(X_train)
StandardScaler()
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

By guaranteeing that the test set is not utilised to affect the scaling parameters, the procedure of fitting the scaler on the training data and then applying the same transformation to both training and testing data helps preserve consistency and eliminates data leaking.

D. Algorithm Used

On this data set the following algorithms were implemented and the working of these algorithms are being referenced below:

1. Logistic Regression It Makes use of sklearn.linear_model's LogisticRegression() method and Predicts using the scaled test data (X_test_scaled) and fits the logistic regression model to the scaled training data (X_train_scaled)[5]. It Produces a classification report and computes the confusion matrix, accuracy, precision, recall, and F1-score.The goal of using logistic regression is to classify binary data using astronomical characteristics. The brief detail about confusion matrix , accuracy, precision, recall and F1- score can be found in the referenced paper [7].

2. Decision Trees: It Makes use of sklearn.tree's DecisionTreeClassifier() and Predicts using scaled test data and fits the decision tree model to scaled training data. Decision trees are used with the purpose of segmenting data according to feature thresholds for categorization for which the further detail can be found in the referenced paper [6].

3. KNN, or K-Nearest Neighbours: It Uses KNeighborsClassifier() from sklearn.neighbors in its implementation and To find the optimal k value, cross-validation is performed. Then, the KNN model is trained using the scaled training data and makes predictions using the scaled test data [8].

```
for k in k_values:
    knn = KNeighborsClassifier
    (n_neighbors=k)
    scores = cross_val_score
    (knn, X_train_scaled, y_train, cv=5)
    mean_scores.append(np.mean(scores))
```

```
    accuracy = np.mean(scores)

    # Check if this k gives better accuracy
    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_k = k
```

The above is used to determine the ideal the value of k that optimises the k-NN model's accuracy given the dataset. The results display the most K value discovered and accuracy related to it. For instance, when the value of k was 2 then the accuracy was 85% but for k=3 there was a sudden spike which gave the accuracy of 88%. Furthermore, a line plot is produced to show the average accuracy for various k values, helping to choose the best K graphically. The graph shows that the accuracy didn't change much everytime the value of k was increased from k=7 to k=12 but from k=12 to k=21 the accuracy of approximately 90% was observed. This procedure aids in choosing an acceptable k value which was 15 for the k-NN method on the star dataset that strikes a compromise between model complexity and performance.

4. Naive Bayes Gaussian:

It Makes use of sklearn.naive_bayes' GaussianNB() function and Predicts using scaled test data and fits the Gaussian Naive Bayes model to the scaled training data. The goal of Gaussian Naive Bayes is to perform probabilistic classification based on feature distributions, assuming that features are conditionally independent. The working of naive bayes can be found in the referenced paper.[9]

5. Discriminant analysis that is linear (LDA): It Uses sklearn.discriminant_analysis's LinearDiscriminantAnalysis() in its implementation and Predicts using scaled test data and fits the LDA model to the scaled training data. The goal of LDA is to divide classes according to feature distributions by producing linear decision boundaries for which the further detail can be found in the referenced paper [9].

Each algorithm is applied to classify stars into distinct types using various machine learning techniques, and their performances are evaluated based on multiple classification metrics like confusion matrix, accuracy, precision, recall, F1-score which help to identify the best-performing model for the given astronomical data set [7].

E. Libraries, Tools and Modules:

Libraries: NumPy: A basic Python module for scientific computing that may be used for mathematical functions, arrays, and numerical operations.

Pandas: Offers tools and data structures for data analysis and manipulation; especially helpful for working with DataFrame objects and managing structured data.

Scikit-learn (sklearn): A machine learning package called Scikit-learn (sklearn) provides a number of tools for preprocessing, model selection, regression, clustering, and classification.

Matplotlib: is a comprehensive Python visualisation toolkit that includes plots, charts, histograms, and other visualisations that may be made static, interactive, or animated.

Seaborn: Seaborn offers high-level interfaces for creating visually appealing and educational statistical visualisations. It is built on top of Matplotlib.

Tools:

Jupyter Notebooks are interactive settings for experimenting with programming, data analysis, and visualisation. You may create and share documents with live code, equations, visualisations, and narrative prose using the open-source online application Jupyter Notebook. Because of its interactive and adaptable character, it is frequently utilised in data analysis, scientific research, machine learning, and teaching [10].

F. Result Validation

To evaluate and contrast the models' performance , the measurements are used to plot ROC curves which is calculated using roc_curve().

```
Metrics=metrics.classification_report
(y_pred1,y_test)
lr_fpr, lr_tpr, threshold1 = metrics.roc_curve
(y_test, y_pred1)
lr_roc_auc = metrics.auc
(lr_fpr, lr_tpr)
print(Metrics)
```

By comparing the anticipated (y_pred1) and actual (y_test) values, the first line of the report produces a comparison of accuracy, recall, F1-score, and other metrics. By Using actual target labels (y_test) and expected probabilities (y_pred1), metrics.roc_curve(y_test, y_pred1) calculates the Receiver Operating Characteristic (ROC) curve for the Logistic Regression model. The model's performance is measured by computing the AUC score from the ROC curve using metrics.auc(lr_fpr, lr_tpr). A plot of the ROC curve for the Logistic Regression model is produced using the above code.

The above thing is done for all the models as The Receiver Operating Characteristic (ROC) curve is used in classification problems as a graphical representation of the model's performance across various thresholds. It makes it possible to compare the performance of several models visually. In general, models with curves towards the upper-left corner—that is, with greater true positive rates and smaller false positive rates—are superior. Looking at the curve all the models had a curve above 50% which tells us that all the models performed well and can be used to classify whether the star is a dwarf or giant.

III. Results

The results were computed by testing it on testing data, comprising 25% of the dataset. The results comprises of 25% testing data and 75% training data.

Computational Results:

Logistic Regression : Looking at the results, it was found that logistic regression had the accuracy of 89.36%. When looking at other factors, the logistic regression had a precision of 89.43%. Looking at the confusion matrix the model correctly predicted 3343 giants and 3810 Correctly predicted

dwarfs. However, the model incorrectly predicted 395 dwarfs as giants and 457 giants as dwarfs. This model performed exceptionally well on the star dataset as compared to other models.

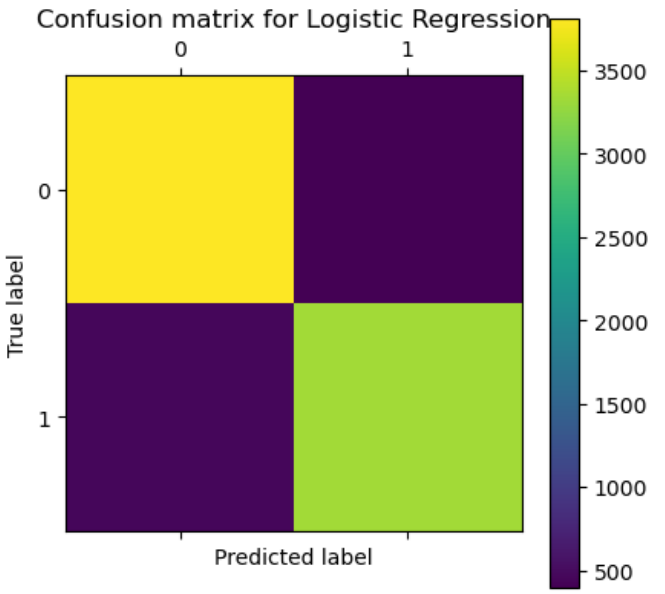

Fig. 1.

Linear Discriminant Analysis:

The model had an Accuracy of 88.13% which means it correctly classified dwarf and giant stars.The precision of the model is 89.45%. This means that when the model predicts a star to be a giant for black hole formation, it's correct around 89.45% of the time. The recall of the model is 87.42%. This indicates that out of all the actual giant stars for black hole formation, the model correctly identifies around 87.42% of them.

Looking at the confusion matrices the model Correctly predicted 3322 giants and 3813 Correctly predicted dwarfs. However, the model incorrectly predicted 392 dwarfs as giants and 478 giants as dwarfs. This model performed exceptionally well on the star dataset as compared to other models.

K-Nearest Neighbors (KNN): The accuracy of K- Neareast Neighbors for k=15 was 89%. This accuracy was maximum among all the values of k. The below graph gives the details about other values.Looking at the confusion matrix the model Correctly predicted 3307 giants and 3844 Correctly predicted dwarfs. But the model incorrectly predicted 361 dwarfs as giants and 493 giants as dwarfs.

The precision of the model is 89.36%. This means that when the model predicts a star to be a giant for black hole formation, it's correct around 89.36% of the time. The recall of the model is 88.36%. This indicates that out of all the actual giant stars for black hole formation, the model correctly identifies around 88.36% of them.

Gaussian Naive Bayes:

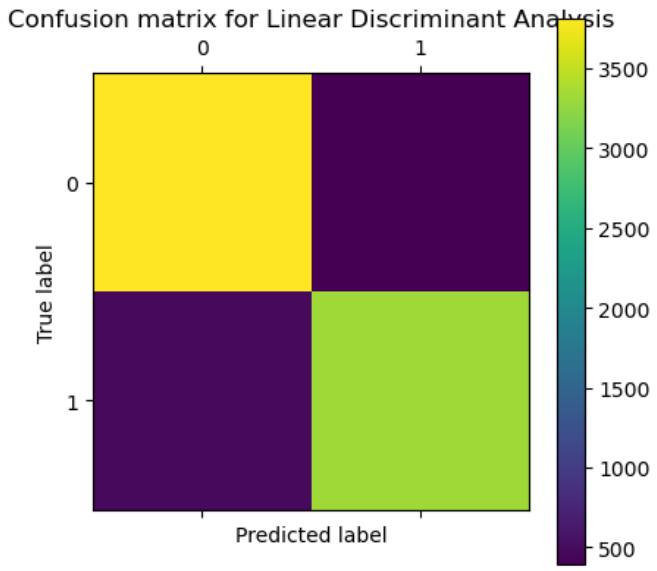

Fig. 2.

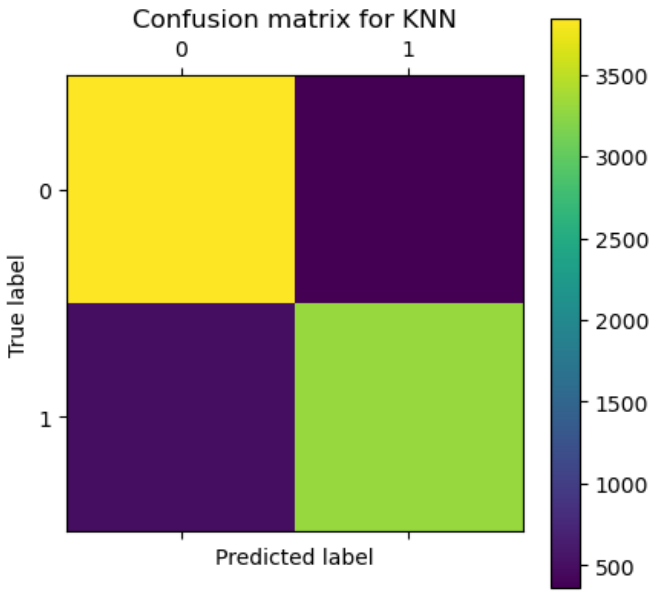

Fig. 3.

The model had an Accuracy of 88.54% which means it correctly classified dwarf and giant stars.The precision of the model is 89.05%. This means that when the model predicts a star to be a giant for black hole formation, it's correct around 89.05% of the time. The recall of the model is 86.5%. This indicates that out of all the actual giant stars for black hole formation, the model correctly identifies around 86.5% of them. Looking at the confusion matrix in Fig 4, the model Correctly predicted 3322 giants and 3813 Correctly predicted dwarfs. But the model incorrectly predicted 392 dwarfs as giants and 478 giants as dwarfs.

Decision Tree: Out of all the models decision tree model

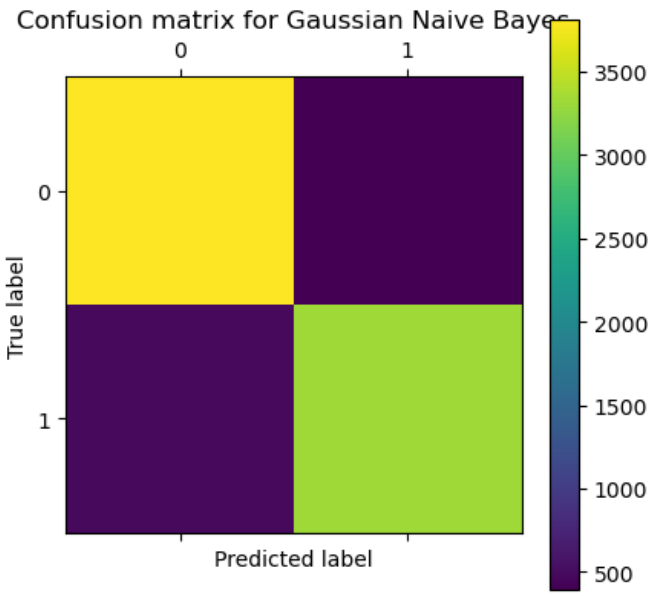

Fig. 4.

had the lowest accuracy of 83.40%. The precision of the model is 81.92%. This means that when the model predicts a star to be a giant for black hole formation, it's correct around 81.92% of the time. The recall of the model is 83.45%. This indicates that out of all the actual giant stars for black hole formation, the model correctly identifies around 83.45% of them.Looking at the confusion matrix in Fig 5, the model Correctly predicted 3168 giants and 3487 Correctly predicted dwarfs.

Fig. 5.

A. *Results Discussion*

Based on the test set data, the logistic regression model, Linear Discriminant Analysis and KNN appears to function rather well, as indicated by its accuracy of 89.36%, 89.13% and 90.3% respectively. But the selection of the model can be best selected by looking at the ROC curve in Fig 6 which predicts the performance of the model.

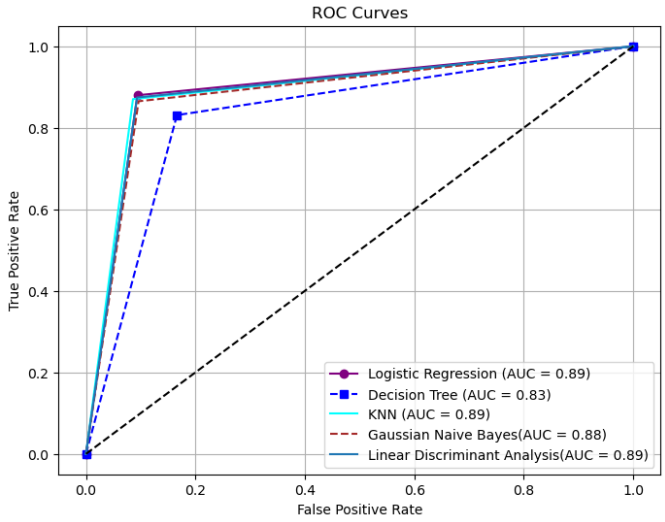

Fig. 6.

B. *Process of Making Decisions:*

Looking at Fig 6 it is evident that all the models performed extremely well as all have a precision of above 50% as seen in the graph. But among all the models decision tree model performed the least while Logistic Regression, KNN and Linear Discriminant performed outstanding well. The Naive Bayes classifier had a precision of 88% as evident from the graph.

IV. Discussion

Good performance of Logistic Regression was anticipated because of its applicability to binary classification problems, which produced high F1 score, accuracy, precision, and recall. Compared to Logistic Regression, Decision Tree performance may be slightly poorer as a result of possible overfitting to the training set. Because the KNN is proximity-based and can handle non-linear correlations in the data, it perform well. Performance of Naive Bayes and LDA were meet as these techniques rely on the independence of features and may work effectively since their underlying presumptions align with the distribution of the data.With an AUC of 0.89 for logistic regression, KNN and Linear Discriminant Model, the models has an 89% chance of accurately classifying or ranking two data points that belong to different classes; in other words, the positive point has a greater prediction probability than the negative class. Moreover, with an AUC of 0.88 for Naive Bayes, the model has an 88% chance of accurately classifying or ranking two data points that belong to different classes;

in other words, the positive point has a greater prediction probability than the negative class. But seeing the graph above, logistic regression, KNN and Linear Discriminant model were above the basline indicating that all the models can be used to predict the accuracy of the star dataset.

A. *Importance of Work*

While selecting the best among model is a tough task but Logistic regression , KNN and LDA can be used to predict whether the star is dwarf or giant. Based on the size of the star, astronomers will get to know whether the star will form black hole or not because black holes are formed when extremely massive stars collide during their last moments of life. When a dying star loses internal pressure, gravity takes hold and the star collapses in on itself. If the star is big enough to sustain such a collapse, ultimately the star's centre will become a black hole. So in order for astronomers to study black hole, these classification models can help them get information about star size. The astronomers must now decide whether to prioritise lowering false positives or false negatives while taking into account the formation of black hole. Because there are a lot of stakes, choosing a model that fits the astronomer's risk tolerance and strategic goals requires considerable thought. But astronomers should take both into account as the formation of black hole depends on the giant star and even if dwarf is classified as giant or giant classified as dwarf will effect the accuracy of the models because in both the cases the accuracy and other factors will be affected. So I think the balance between both the values will give us the model. But all the three models which have high accuracy can be used to predict the type of star and which will help astronomers in their study.

This finding has deep implications for star categorization that are felt in the fields of astronomy, astrophysics, and space exploration. Fundamentally, comprehending and classifying stars provides the groundwork for figuring out the makeup, dynamics, and evolution of the cosmos. This categorization is not only an academic endeavour; it serves as the foundation for innumerable astronomical theories, hypotheses, and real-world applications.

Determining the precise categorization of stars is crucial to solving the enigmas surrounding celestial entities. It facilitates the comprehension of the basic processes underlying the cosmos, the life cycle of stars, and the complexity of galaxies. Furthermore, it is essential to interstellar exploration, satellite navigation systems, and astrophysics research.

B. *Future Work*

In future research, Exploring additional features related to stars like their spectral characteristics or radiance could enhance classification accuracy and Fine-tuning hyperparameters of models like Decision Trees might improve their performance.Moreover, Implementing ensemble techniques like Random Forests or Gradient Boosting could potentially boost accuracy by aggregating multiple models' predictions.

REFERENCES

[1] https://www.kaggle.com/datasets/vinesmsuic/star-categorization-giants-and-dwarfs/data
[2] https://www.kaggle.com/code/habiburrahamanfahim/potential-black-hole-categorization/notebook
[3] https://pyastronomy.readthedocs.io/en/latest/pyaslDoc/aslDoc/aslExt
[4] https://www.geeksforgeeks.org/introduction-to-seaborn-python/
[5] J. Pamnani, A. Ketkar, J. Hasija, and D. Lnu, Classification using Logistic Regression
[6] J. Pamnani, A. Ketkar, J. Hasija, and D. Lnu, Decision Tree Analysis on Iris Dataset
[7] J. Hasija, A Comprehensive Study of Classification Algorithms on Iris Data Set
[8] J. Pamnani, A. Ketkar, J. Hasija, and D. Lnu, K-Nearest Neighbor: A Comprehensive Study on 'K' and Decision Metrics
[9] J. Hasija: Classification of Iris dataset using Naive Bayes and Linear Discriminant analysis classifiers
[10] https://jupyter-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html