

# Stellar Classification

## Introduction

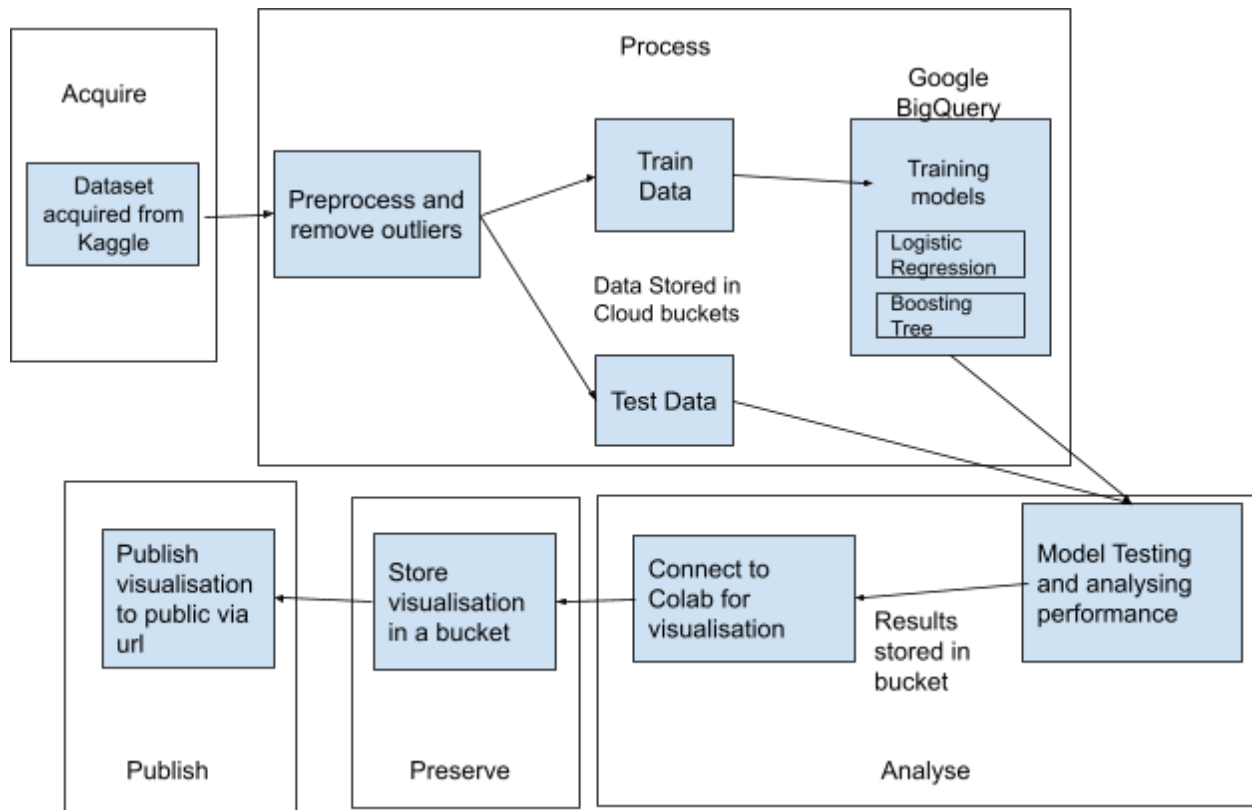
Stellar Classification based on spectral features is one of the foundational concepts of astronomy and in this project we are going to build machine-learning models that will be able to classify objects between galaxies, quasars, and stars. We will be using resources from Cloud(GCP) like computational power, and storage and be able to manage and access them via the internet. During the last few years, cloud computing's reliance has increased a lot and it offers modern businesses efficiency, security, and scalability at a reduced price. There are several use cases today and cloud services are being implemented in almost every industry. Researchers forecast that this trend will keep improving and there will be an enormous growth in expenditure by end users. The stellar dataset we used contains 100000 entries and 17 columns with different spectral attributes. All of them are numerical features and the output variable has three classes: star, galaxy, and quasar.

We have acquired this dataset from Kaggle and we have implemented data preprocessing and the dataset is split to test and train. They will be stored in Google cloud buckets and this data will be acquired in the BigQuery workspace. We implement machine learning models and their performances are analyzed. The results are exported and visualized into a plot and this plot is stored back into a bucket the results are published which can be accessed by the public using an URL.

## Background

Cloud computing is really important for astronomers because of the terabytes or petabytes amounts of data being produced in real-time by various observatories and satellites. It is not easy to process and store this type of data. Building local resources for computation is too expensive, so various space and astronomical organizations are using cloud infrastructure for their various needs. It provides enough computational power to process huge amounts of data quickly and can also be scaled easily. I have always been interested in space organizations and cosmological advances and I believed that this would be a wonderful opportunity to work on astronomical data as there has been huge scope for cloud services on such data.

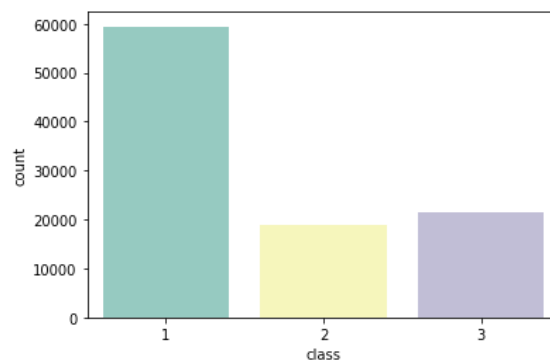
## Methodology



### Flowchart:

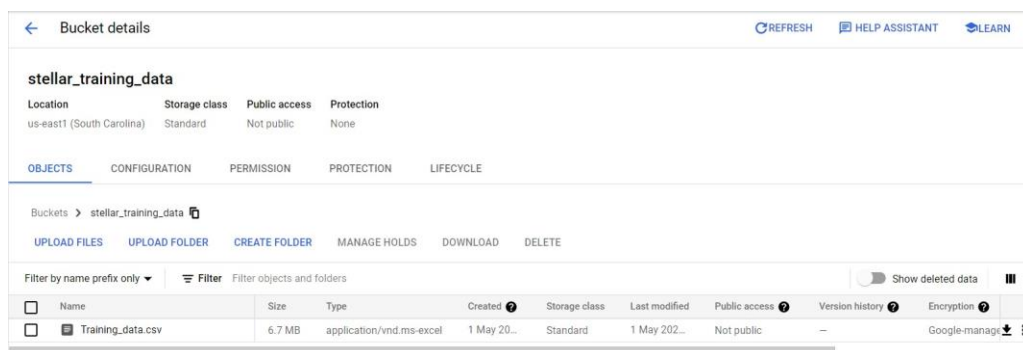
We followed the USGS Data Lifecycle model and below is the detailed workflow of this project.

**Acquire:** We acquired the stellar classification dataset from Kaggle and it is imported to the Jupyter notebook interface. We checked for null values and duplicate values in the dataset. We replaced the class(output) variable with the numerical number 1,2,3 which represents galaxy, quasar, and star respectively.



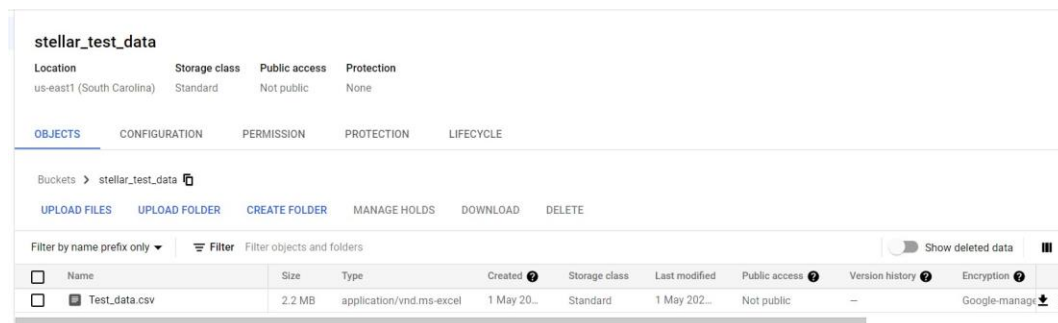
**Process:** We created a function to remove outliers from the data by removing data points that are 1.5 times above IQR(inter-quartile distance) and 1.5 times below IQR from quartile 1 and quartile 3 respectively. We have split the dataset to train and test which is 75 and 25% of the dataset. We have saved the train and test data frames to CSV files.

Later, we opened the google cloud console and created a project with the name 'akveges-stellarclass'. We have created two data buckets in the cloud named stellar-training-data and selected the location as us-east1, selected standard storage class as we are using it for a short time. We enforced public access prevention for security, selected no additional protection required and created the bucket, and uploaded the train data file. Likewise, we have created a bucket with the same options selected for the test data and saved it as stellar-test-data, and uploaded the test data file.



The screenshot shows the 'Bucket details' page for 'stellar\_training\_data' in the Google Cloud console. The bucket is located in 'us-east1 (South Carolina)', uses 'Standard' storage class, has 'Not public' access, and 'None' protection. The 'OBJECTS' tab is selected, showing a list of objects. A single object, 'Training\_data.csv', is listed with a size of 6.7 MB, type 'application/vnd.ms-excel', and created on May 20, 2022. The interface includes navigation links like 'UPLOAD FILES', 'CREATE FOLDER', and 'MANAGE HOLDS'.

stellar_training_data									
Location	Storage class	Public access	Protection						
us-east1 (South Carolina)	Standard	Not public	None						
OBJECTS									
Buckets > stellar_training_data									
[UPLOAD FILES] [UPLOAD FOLDER] [CREATE FOLDER] [MANAGE HOLDS] [DOWNLOAD] [DELETE]									
Filter by name prefix only [Filter] Filter objects and folders [Show deleted data]									
<input type="checkbox"/>	Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption
<input type="checkbox"/>	Training_data.csv	6.7 MB	application/vnd.ms-excel	1 May 20...	Standard	1 May 202...	Not public	—	Google-manage



The screenshot shows the 'Bucket details' page for 'stellar\_test\_data' in the Google Cloud console. The bucket is located in 'us-east1 (South Carolina)', uses 'Standard' storage class, has 'Not public' access, and 'None' protection. The 'OBJECTS' tab is selected, showing a list of objects. A single object, 'Test\_data.csv', is listed with a size of 2.2 MB, type 'application/vnd.ms-excel', and created on May 20, 2022. The interface includes navigation links like 'UPLOAD FILES', 'CREATE FOLDER', and 'MANAGE HOLDS'.

stellar_test_data									
Location	Storage class	Public access	Protection						
us-east1 (South Carolina)	Standard	Not public	None						
OBJECTS									
Buckets > stellar_test_data									
[UPLOAD FILES] [UPLOAD FOLDER] [CREATE FOLDER] [MANAGE HOLDS] [DOWNLOAD] [DELETE]									
Filter by name prefix only [Filter] Filter objects and folders [Show deleted data]									
<input type="checkbox"/>	Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption
<input type="checkbox"/>	Test_data.csv	2.2 MB	application/vnd.ms-excel	1 May 20...	Standard	1 May 202...	Not public	—	Google-manage

Next, we opened BigQuery which is a serverless workspace in GCP which is a data warehouse. It helps in building ML pipelines using SQL. In the BigQuery we have opened our project and created a dataset with a dataset id 'stellar' and set expiry to 5 days. We set Google managed encryption key and created a dataset. In the dataset, we have created tables for testing and training data. We set create tables from location to Google cloud as we have our data in buckets. We set the table name to stellar train table for train data and stellar test table for test data. We selected Auto to detect schema to on and after expanding advanced options, we set header rows to skip as 1.

stellar train table		stellar test table	
<a href="#">QUERY</a> <a href="#">SHARE</a> <a href="#">COPY</a>		<a href="#">QUERY</a> <a href="#">SHARE</a> <a href="#">COPY</a>	
SCHEMA DETAILS PREVIEW		SCHEMA DETAILS PREVIEW	
Table info		Table info	
Table ID	akveges-stellarclass.stellar.stellar train table	Table ID	akveges-stellarclass.stellar.stellar test table
Table size	5.4 MB	Table size	1.8 MB
Long-term storage size	0 B	Long-term storage size	0 B
Number of rows	64,300	Number of rows	21,434
Created	1 May 2022, 18:48:58 UTC-4	Created	1 May 2022, 18:50:02 UTC-4
Last modified	1 May 2022, 18:48:58 UTC-4	Last modified	1 May 2022, 18:50:02 UTC-4
Table expiry	6 May 2022, 18:48:58 UTC-4	Table expiry	6 May 2022, 18:50:02 UTC-4
Data location	us-east1	Data location	us-east1
Default collation	[null]	Default collation	[null]
Description		Description	

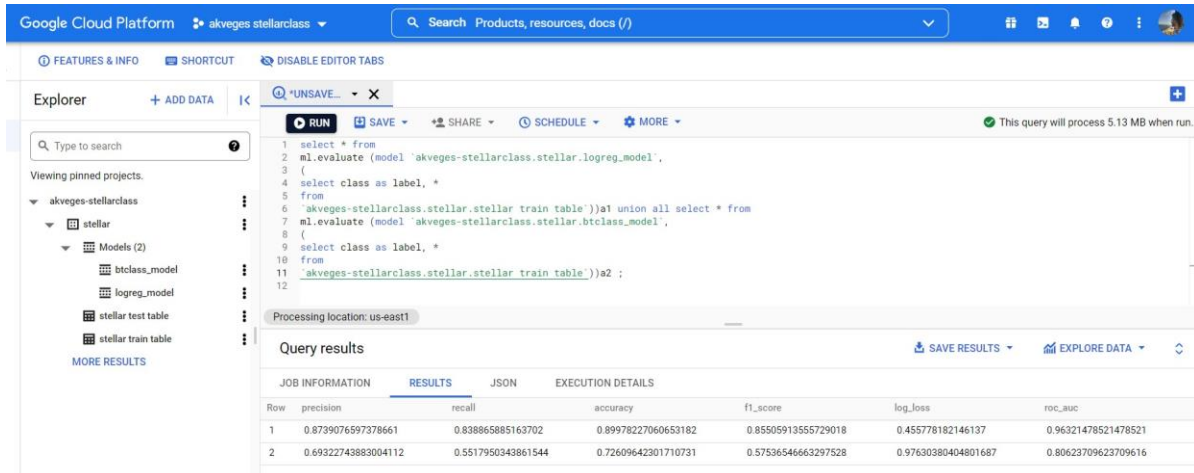
After creating the tables, on the SQL workspace, we built queries to build ML models. We have created two BigQuery ML models, one for logistic regression and one for boosting trees. We are primarily focusing on building pipelines and workflow and not the functioning of algorithms. Our created logistic regression model is named logreg\_model and the class is the output column. The columns we considered for training the model are alpha, delta, u, g, r, i, z, plate, and MJD. After successfully training the model, the training options column shows parameters that are added to the model, when the model is created. In the evaluation dashboard, we can see aggregate metrics and a confusion matrix of true and predicted labels.

The screenshot shows the Google Cloud Platform SQL workspace interface. The top navigation bar includes the Google Cloud Platform logo, the project name 'akveges-stellarclass', a search bar, and various icons. The main workspace is divided into several sections:

- Explorer:** Displays a tree view of the project structure, including 'akveges-stellarclass', 'stellar', 'Models (1)', 'logreg\_model', 'stellar test table', and 'stellar train table'.
- Query Editor:** Contains a SQL query to create or replace a BigQuery ML model named 'logreg\_model' using logistic regression. The query is as follows:
 

```
1 create or replace model `akveges-stellarclass.stellar.logreg_model`
2 options
3 (model_type='logistic_reg')
4 as
5 SELECT
6 class as label,
7 alpha,delta,u,g,r,i,z,plate,MJD,
8 from `akveges-stellarclass.stellar.stellar train table`;
```
- Query results:** Displays the execution details of the query, including:
  - Job Information:** Elapsed time: 1 min 9 sec.
  - Results:** Slot time consumed: 4 min 30 sec.
  - Execution Details:** Stages: Preprocess (0), Train (0), Evaluate (0).
  - Training Iterations:** Completed: 0, Planned: 0.

Our second model has boosted tree classifier and is named btclass\_model and the same class is the output variable. The columns we considered for training the model are alpha, delta, u, g, r, i, z, plate, and MJD. After successfully training the model, the training options column shows parameters that are added to the model, when the model is created. In the evaluation dashboard, we can see aggregate metrics and a confusion matrix of true and predicted labels.



### Analyze:

For evaluating ml models we used ml. evaluate in BigQuery and the outcome is the class column and this gives an output of a table with all aggregate metrics. The metrics that will be in the output are precision, recall, accuracy, f1\_score, log\_loss, and roc\_auc.

We made a SQL query that will take aggregate metrics from both models and give a table of both model metrics as a result. For testing the model we can use the ml. predict function and it will give a predicted class number for the input features.

We created a bucket 'stellar results' to store the results and we store the table of results after creating the bucket. We used the same process to create a bucket as we did before. We imported the results from the SQL query to the bucket.

We opened Google Colaboratory and connected it to the Google BigQuery workspace. After giving authorization, we can import the results as data frame df.

```

+ Text
from google.colab import auth
auth.authenticate_user()
print('Authenticated')

Authenticated

%bigquery --project akveges-stellarclass df
SELECT *
FROM `bigquery-public-data:samples.gsod`

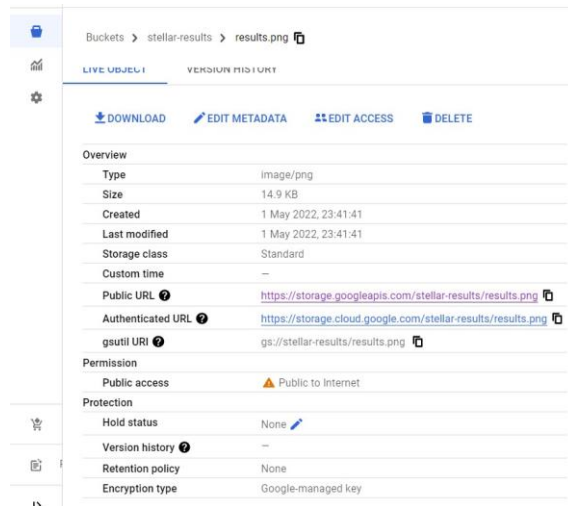
```

We imported necessary libraries like Matplotlib and created a grouped bar plot. The output is a grouped bar plot for all 6 metrics of both models which are shown below.

**Preserve:** We store this visualization in the bucket the same one we stored the results table.

**Publish:** The visualization file in the cloud bucket is opened. When selecting edit access, we can add entries and permit all users. Selecting public for the entity, all users for

Name and reader for access we save and it gives access for public sharing, a public URL is created. This public URL can be accessed by anyone when shared.



## Results

We have created two BigQuery ML models, one for logistic regression and one for boosting trees. For evaluating ml models we used `ml.evaluate` in BigQuery and the outcome is a class column and this gives an output of a table with all aggregate metrics. The metrics that will be in the output are precision, recall, accuracy, `f1_score`, `log_loss`, and `roc_auc`.

We made a SQL query that will take aggregate metrics from both models and give a table of both model metrics as a result. Below is the table that shows the results

The screenshot shows the Google Cloud Platform BigQuery interface. A SQL query is executed, and the results are displayed in a table. The query evaluates two models: `akveges-stellarclass.stellar.logreg_model` and `akveges-stellarclass.stellar.btclass_model`.

```

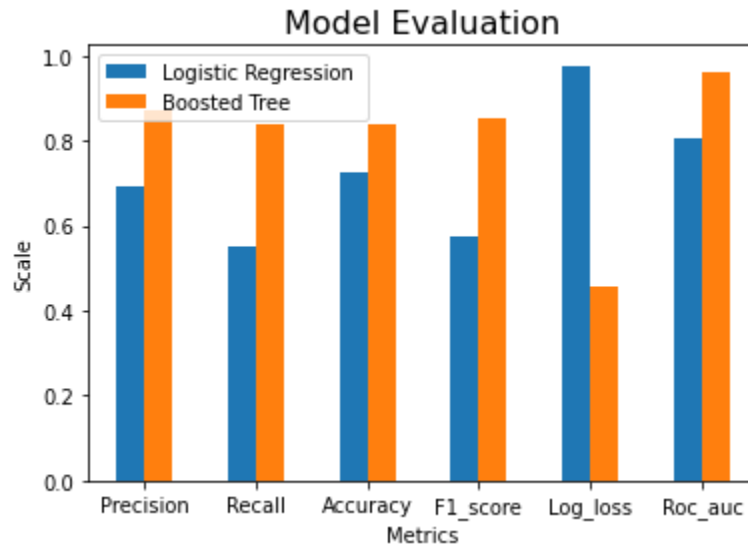
1 select * from
2 ml.evaluate (model `akveges-stellarclass.stellar.logreg_model`,
3 {
4 select class as label, *
5 from
6 `akveges-stellarclass.stellar.stellar train table`))a1 union all select * from
7 ml.evaluate (model `akveges-stellarclass.stellar.btclass_model`,
8 {
9 select class as label, *
10 from
11 `akveges-stellarclass.stellar.stellar train table`))a2 ;
12

```

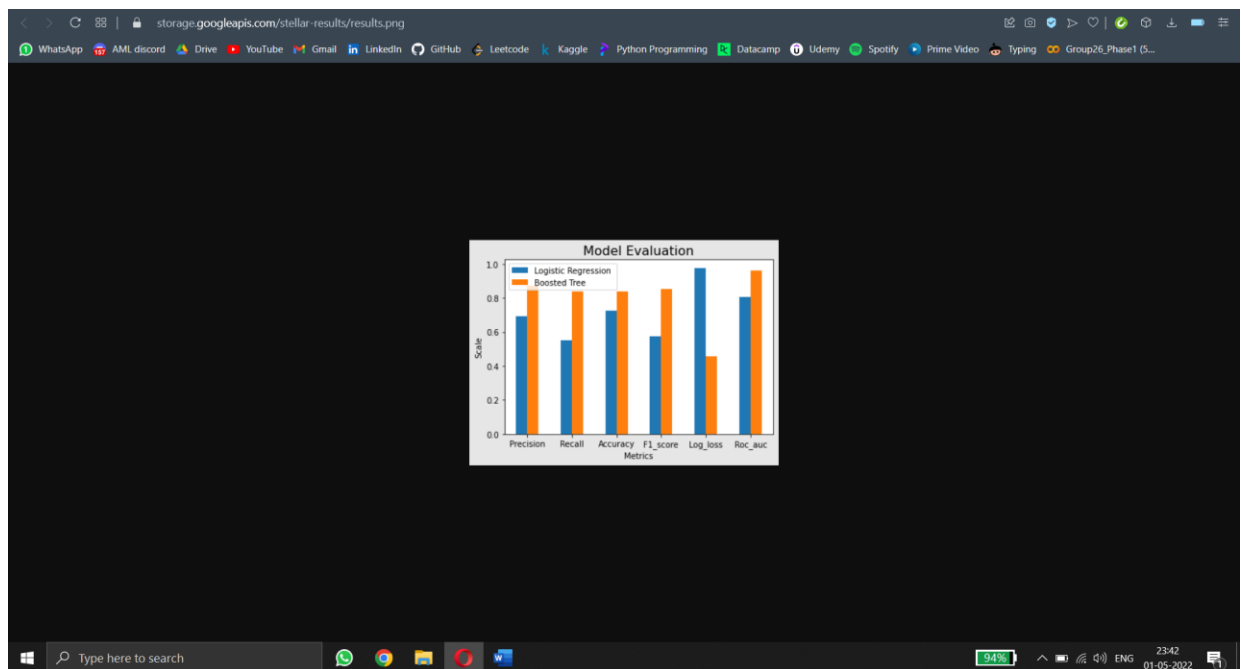
The results table shows the following data:

Row	precision	recall	accuracy	f1_score	log_loss	roc_auc
1	0.8739076597378661	0.838865885163702	0.89978227060653182	0.85505913555729018	0.455778182146137	0.96321478521478521
2	0.69322743883004112	0.5517950343861544	0.72609642301710731	0.57536546663297528	0.97630380404801687	0.80623709623709616

Here is the Matplotlib visualization created in Google Colab on the table of results created by the SQL query.



This is the result of clicking the public URL link of the Google cloud bucket file. The link redirects to a new tab that shows the plot that shows the aggregate metrics of both the trained models which are logistic regression and boosted tree classifier.



## Discussion

While testing the ml models Boosted tree and logistic regression we have measured the aggregate metrics Precision, recall, accuracy, f1 score, log loss, and roc auc.

Accuracy is one of the most basic metrics and it is the ratio of correct predictions to all predictions. We see that boosted tree has got more accuracy than logistic regression.

The next metric is Recall which is the ratio of class samples that are classified precisely. The plot shows that we received more recall for the boosted tree which is 83.8%. If data is not distributed evenly then precision would be a better feature to evaluate the model performance. It's the ratio of true positive values to all positive values. We got 87.3% for the precision of boosted tree. The F1 score combines the importance of both recall and precision metrics. We got an 85.5% F1 score for the boosted tree. For a given model lower log loss value means better predictions and we got a lower log loss of 45.5% for the boosted tree. The more AUC value means the model is performing well in differentiating between positive, and negative classes. This proves that boosted tree classifier is a better-performing model in terms of all the metrics.

We have employed a USGS Data Lifecycle pipeline which ranges from acquire, process, analyze preserve and publish. We have employed data preprocessing and have explored the cloud Big Data cloud platform - Google BigQuery which is a data warehouse for exploring data in SQL workspace and creating models and analyzing them. We deployed a storage model for our dataset in the form of Cloud Bucket. We defined a function to detect and remove outliers from the data during preprocessing. We ingested datasets into BigQuery from buckets. Creating buckets, adding files, changing access to files in the bucket to make them public, data ingestion to BigQuery, and analyzing data using SQL are some of the skills learned from this program that is employed in this project.

I faced a struggle using the data studio which is connected to Bigquery as I am not familiar with the platform. So I had to connect Bigquery and plot the visualization using Colab. And I could not find any other classification models in Bigquery ML other than boosted trees and logistic regression. It would have been better if Bigquery ML supported more models so that I would have been able to run more models and compare the results which gives a better scope of getting the best model for specific data



## Conclusion

We have successfully deployed an end-to-end data lifecycle on Stellar classification data. We were able to deploy two BigQuery ML models on the SQL workspace in the GCP. We preprocessed the data, removed outliers, and created buckets for the storage of this data. We analyzed the performance of these models based on 6 aggregate metrics and visualized them to a plot using Colab connected to BigQuery. We were able to conclude that boosted tree classifier is a better performing model than logistic regression when compared on all the metrics. We preserved this plot in the cloud bucket and we gave public access and publish this file which created a URL that can be accessed by anyone who possesses it. Later in the future, we plan to employ feature engineering and hyperparameter tuning to improve the performance of the models.

## References

- 1 <https://www.kaggle.com/datasets/fedesoriano/stellar-classification-dataset-sdss17>
2. <https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce>
3. [https://www.qwiklabs.com/focuses/1846?catalog\\_rank=%7B%22rank%22%3A1%2C%22num\\_filters%3A0%2C%22has\\_search%3Atrue%7D&parent=catalog&search\\_id=7008005](https://www.qwiklabs.com/focuses/1846?catalog_rank=%7B%22rank%22%3A1%2C%22num_filters%3A0%2C%22has_search%3Atrue%7D&parent=catalog&search_id=7008005)
4. <https://cloud.google.com/storage/docs/access-control/making-data-public>
5. [https://www.qwiklabs.com/focuses/3692?catalog\\_rank=%7B%22rank%22%3A5%2C%22num\\_filters%3A0%2C%22has\\_search%3Atrue%7D&parent=catalog&search\\_id=14163071](https://www.qwiklabs.com/focuses/3692?catalog_rank=%7B%22rank%22%3A5%2C%22num_filters%3A0%2C%22has_search%3Atrue%7D&parent=catalog&search_id=14163071)
6. <https://prog.world/astronomy-big-data-and-clouds-how-technology-helps-to-study-the-universe>