Classification of Stars and Quasars

Machine Learning Project (UE17CS303)

Prakruti Rao
PES1201701126
Department of Computer Science and Engineering
PES University

Ashika Meryl Pinto
PES1201701346
Department of Computer Science and Engineering
PES University

***Abstract*—This project attempts to classify photometric data collected from the Galaxy Evolution Explorer(GALEX) and the Sloan Digital Sky Survey (SDSS) over the North Galactic region and Equatorial region in to spectroscopic classes of Stars and Quasars. The problem is daunting because stars and quasars are still inextricably mixed and no clear linear/non-linear boundary separates the two entities. A K Nearest Neighbour classifier is used to tackle this classification problem. To evaluate the correctness of the classifiers, we report the accuracy and other performance metrics and find a reasonably satisfactory range of 94-98 percent .**

I. Introduction

A quasar is an extremely luminous active galactic nucleus (AGN), in which a supermassive black hole with mass ranging from millions to billions of times the mass of the Sun is surrounded by a gaseous accretion disk.A star is an astronomical object consisting of a luminous spheroid of plasma held together by its own gravity. One of the major challenges of of large scale photometric surveys is the separation of stars and quasars. Both types of sources have a compact optical morphology and are hence difficult to separate .

The number of quasars in our data set is much larger than the number of stars. This imbalance between the two classes results in a poor automation of the task and it demands the need to up-sample the star observations in order to match the number of quasars. To increase the number of stars we have used a random up sampling method. For every instance in the majority class, a minority sample at random is chosen resulting in clones of the minority class. This is done to enhance the classification task which would have otherwise produced results biased towards the majority class. Now the total number of rows in the data set is equal to double the number of instances in the majority class.

Samples of our data set belong to a well-defined category and this is used for learning the classification task. In our case, the samples that belong to a particular category, as derived from the spectroscopic mapping, are used for training the classifier. This method in machine learning is known as a supervised learning approach.

We have used the K Nearest Neighbour supervised learning classification algorithm. K Nearest Neighbour is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure. A 10 fold cross validation was done to validate the results.

II. Data Set and the Problem

The data set is divided into 4 different catalogs. Catalog 1 contains data about the North Galactic Region only. Catalog 2 contains data about the Equatorial Region only.Catalog 3 has combined data from the North Galactic Region and Equatorial Region.

A. Catalog 1

This catalog consists of only samples from the North Galactic region that have the attribute fuv. The entire feature list was populated with pairwise differences.

B. Catalog 2

This catalog consists of only samples from the Equatorial region that have the attribute fuv. The entire feature list was populated with pairwise differences.

C. Catalog 3

This catalog consists of only samples from the North Galactic region and Equatorial Region combined. All the instances have the attribute fuv. The entire feature list was populated with pairwise differences.

D. Catalog 4

This catalog consists samples from both the regions. The fuv attribute and pairwise differences with the fuv attribute have been removed in this catalog.

In our project, we use the spectroscopic labels as the primary class label. In the dataset, there are no photometric labels that tell us if a given instance is a quasar or a star, this information can be retrieved only from the spectroscopic data.Thus, this is the problem statement we want to address i.e. to classify photometric data into spectroscopic classes.Stars and quasars look very similar in their optical images. But the spectral energy distribution for stars and quasars is different and so the optical bands SDSS namely u, g, r, i and z can be used to separate them.

Identify applicable funding agency here. If none, delete this.

III. Detailed Methodology

We now present deep analysis of methods designed to accomplish the reported outcome. As mentioned above, the data set has four catalogs. Since the third catalog includes details of both the North Galactic Region and the Equatorial Region, we chose to fist train our model on the third catalog.Since a major imbalance was found in the target classes , we up sampled the minority class(stars). For every instance of the majority class a random instance from the minority class was chosen. Once the dataset was upsampled, we dropped columns like galexobjid and sdssobjid which would not contribute much to predicting the class since they are only identification numbers.Spectrometric redshift was also dropped. The dataset was split into X and Y where X had the features and Y had the respective classes (star/ quasar). Both X and Y were converted into numpy arrays for ease of use. The next step was to split X and Y into X_train,X_test, Y_train and Y_test.A 70-30 train test split was used.

The next step was to transform all the attributes to have mean 0 and standard deviation 1.This is essential because the larger ranged attributes should not affect/contribute to the result more that the others. To do this , two functions were created to calculate the means and standard deviations of each of the columns of the training data(X_train). Using these means and standard deviations, both the train and test data was standardized. ((x - mean)/standard deviation)

After the upsampling , splitting and standard transformation was done, a function to implement the K Nearest Neighbours Algorithm was written.K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. The measure of proximity we chose was Euclidian distance.

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

Fig. 1. Euclidean Distance

There are other ways of calculating distance, and one way might be preferable depending on the problem we are solving. However, the straight-line distance (also called the Euclidean distance) is a popular and familiar choice.

IV. The KNN Algorithm:

-Load the data set
-Initialize K to your chosen number of neighbors
-For each instance in the data set:
-Calculate the distance between the query example and the current example from the data.
-Add the distance and the index of the example to an ordered collection
-Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
-Pick the first K entries from the sorted collection
-Get the labels of the selected K entries
-Return the mode of the K labels

To select the K that was right for our data, we ran the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encountered while maintaining the algorithm's ability to accurately make predictions when it is given data that it hasn't seen before.The above algorithm was implemented from scratch in python and the classification for every instance in X_train was stored in a list. These predictions were compared with the actual target values to calculate the accuracy and the confusion matrix was derived.The accuracy obtained was 96.68 %. Next, both catalogs 1 and 2 were tested on the same model that was built on catalog 3. Accuracies for the same were measured. Along with this, a 10 fold cross validation was done on catalog three only just to ensure good performance with all splits of train and test data. Catalog 4 does not contain certain FUV and related attributes. So we trained and tested the model separately on catalog 4.

V. Results

Correctness of the classification model is determined by the usage of metrics such as accuracy, precision, recall, and F Score.

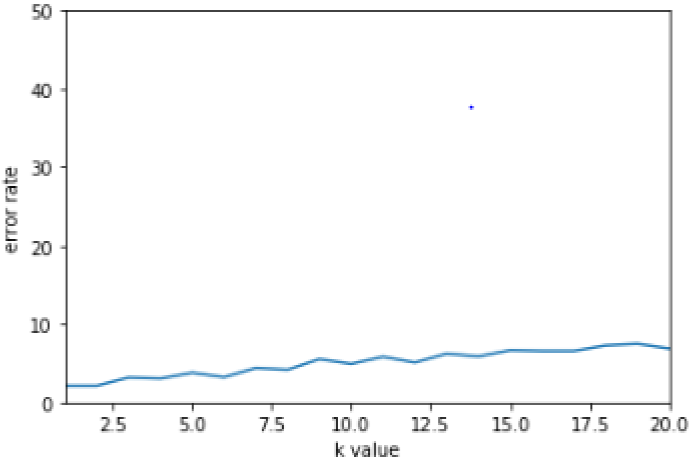
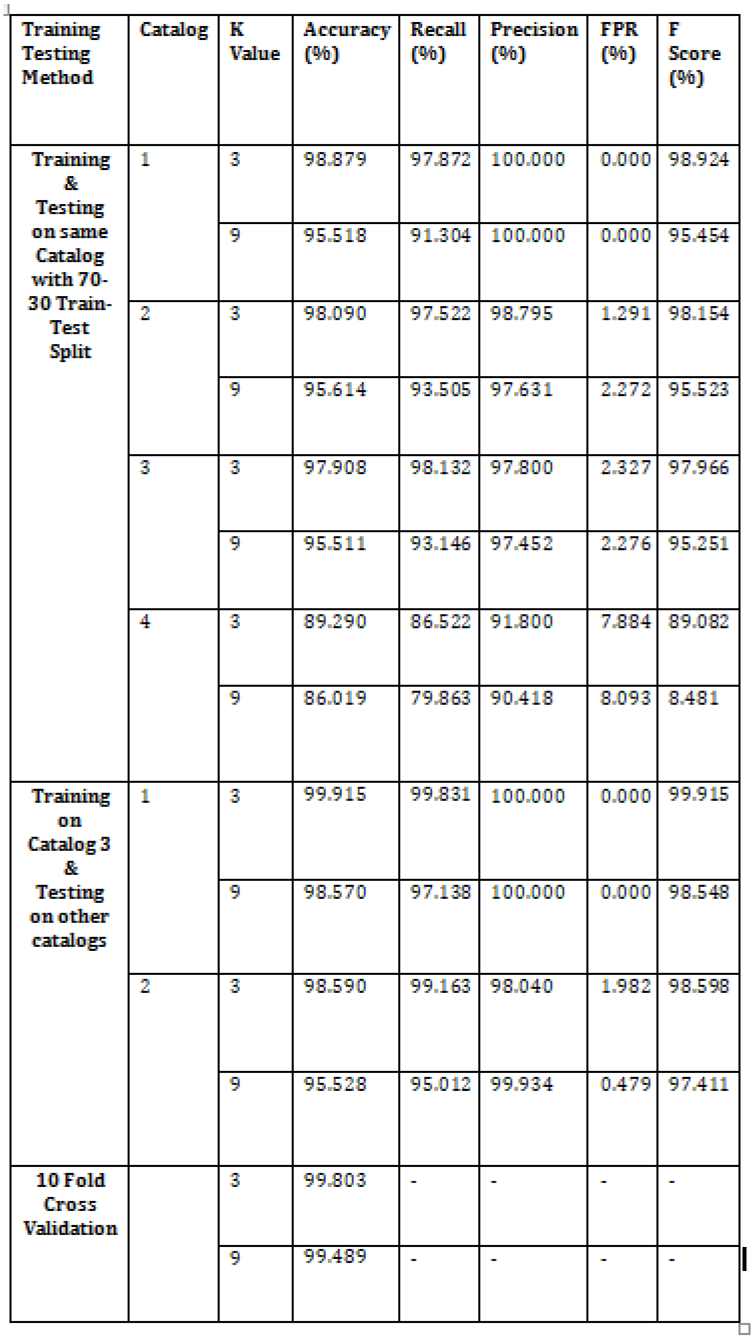


Fig. 2. K Value VS Accuracy

A.

The above figure is a plot of K value VS accuracy. From the figure, we can observe that the error rate does not vary much for k values between 1 and 10 and tends to increase as k values increases above 10. From this we can infer that a k value of 2 is suitable for the KNN model, but any k value less than 10 will also work and not make much of a difference.

Below we present the various results of training and testing :



Training Testing Method	Catalog	K Value	Accuracy (%)	Recall (%)	Precision (%)	FPR (%)	F Score (%)
Training & Testing on same Catalog with 70-30 Train-Test Split	1	3	98.879	97.872	100.000	0.000	98.924
		9	95.518	91.304	100.000	0.000	95.454
	2	3	98.090	97.522	98.795	1.291	98.154
		9	95.614	93.505	97.631	2.272	95.523
	3	3	97.908	98.132	97.800	2.327	97.966
		9	95.511	93.146	97.452	2.276	95.251
	4	3	89.290	86.522	91.800	7.884	89.082
		9	86.019	79.863	90.418	8.093	8.481
Training on Catalog 3 & Testing on other catalogs	1	3	99.915	99.831	100.000	0.000	99.915
		9	98.570	97.138	100.000	0.000	98.548
	2	3	98.590	99.163	98.040	1.982	98.598
		9	95.528	95.012	99.934	0.479	97.411
10 Fold Cross Validation		3	99.803	-	-	-	-
		9	99.489	-	-	-	-

Fig. 3. Detailed Results

VI. Conclusion

The KNN model has successfully been able to classify the stars and quasars with a reliable accuracy. This accuracy was achieved due to up-sampling the data, standard transformation , choosing appropriate K values, applying KNN and validating it with 10 fold cross validation.

References

[1] https://www.researchgate.net/publication/33242689_Separating_Stars_from_Quasars_Machine_Learning_Investigation_Using_Photometric_Data

[2] https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/

[3] https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/