Machine Learning Engineer Nanodegree

Capstone Project

Classification of Galaxies, Stars and Quasars Based on Photometric Data

Sara A. Alhamed
February 6th, 2019

I. Definition

Project Overview

Figure 1

In this project, I will be writing a code to identify three celestial objects from each other: galaxies, stars and quasars, which are very bright and active star-like objects found in the nucleus of some large galaxies. Base on numerical measurements gathered from observing these similar objects' lights and radiation statistical and mathematical models can classify them more correctly and accurately than by looking at their images alone which can be very hard to differentiate them especially between stars and quasars. You see a star in the middle of the image in Figure 1? This is actually what is known as a Quasar, named Quasar 3C 273 taken by the Hubble Space Telescope and might very much look like a star in the images but they are completely different objects and Astronomy is full of such things that looking at their images alone cannot correctly identify them.

Problem Statement

Data collected by www.SDSS.org is huge and is increasing annually as more and more parts of the sky are surveyed. Analyzing data from space for a certain purpose is something Astronomers have always been doing to classify celestial bodies, understand the nature of stars by analyzing their light waves, for navigation and mapping purposes, ...etc.

Classification using Machine Learning however, is bringing this process of Astronomical data analysis to whole new levels, it is faster dealing with thousands if not millions of records of numerical data or images or maps, picking up patterns and relations better than any time before, applying complex statistical and mathematical operations. The problem I want to work on in this project is the classification of three celestial bodies: Galaxies, Stars and Quasars (a massive and extremely remote celestial object, emitting exceptionally large amounts of energy, and typically having a star-like image in a telescope).

The goal of my project is to achieve high accuracy classification of Galaxies, Stars and Quasars based on their numeric photometric data collected by SDSS. For the classification, I will be implementing Support Vector Machine algorithm, Random Forest algorithm and Multilayer Perceptron Classifier.

Metrics

Four common evaluation metrics for supervised learning algorithms I will be using for evaluating Support Vector Machine, Random Forest and Multilayer Perceptron Classifier:

• Accuracy = $\frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$ Probably not the best metric to be used when having an imbalanced dataset, however, accuracy will be used here for the sake of comparison with the benchmark results.	• Precision = $\frac{tp}{tp + fp}$ Precision is an important metric that refers to the percentage of results of the model that are relevant.
• Recall = $\frac{tp}{tp + fn}$ Recall is also an important metric, it reference to the percentage of relevant results that the model got right.	• F1-Score = $(1 + \beta^2)\frac{\text{precision} \times \text{recall}}{\beta^2 \text{precision} + \text{recall}}$ The F1 score is a measure of a test's accuracy considering both the precision p and the recall r of the test to compute the score which is what we need to evaluate a model built on an imbalanced dataset.

Unsupervised learning evaluation metric I will be using for evaluating K-means Clustering and Gaussian Mixture:

- Silhouette:

  A technique provides a succinct graphical representation of how well each object I cluster:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Where s(i) lies in the range of [-1,1]. For s(i) == 1, indicates that the data set (i) is well matched in the cluster assignment. The closer s(i) to 1 the better the clustering model [1].

II. Analysis

Data Exploration

- **Dataset:**

The Sloan Digital Sky Survey releases their collected data every two to three years and their latest Data Release is DR15 containing Astronomical observations from the beginning of this survey in 2000 through July of 2017 as the survey now is in its fourth phase. SDSS data can be accessed through

SQL queries on any of their databases, downloaded as bulk data and other ways. The dataset I will be working on will has 10,000 entries taken from the table *dbo.SpecPhotoAll* in the SDSS DR 14 database shared on Kaggle (www.kaggle.com/lucidlenn/sloan-digital-sky-survey/home).

First 10 records:

	objid	ra	dec	u	g	r	i	z	run	rerun	camcol	field	specobjid	class	redshift	plate	mjd	fiberid
2	1.24E+18	183.53133	0.089693	19.47406	17.0424	15.94699	15.50342	15.22531	752	301	4	267	3.72E+18	STAR	-8.96E-06	3306	54922	491
3	1.24E+18	183.59837	0.135285	18.6628	17.21449	16.67637	16.48922	16.3915	752	301	4	267	3.64E+17	STAR	-5.49E-05	323	51615	541
4	1.24E+18	183.68021	0.1261851	19.38298	18.19169	17.47428	17.08732	16.80125	752	301	4	268	3.23E+17	GALAXY	0.1231112	287	52023	513
5	1.24E+18	183.87053	0.0499107	17.76536	16.60272	16.16116	15.98233	15.90438	752	301	4	269	3.72E+18	STAR	-0.0001106	3306	54922	510
6	1.24E+18	183.88329	0.1025568	17.55025	16.26342	16.43869	16.55492	16.61326	752	301	4	269	3.72E+18	STAR	0.0005904	3306	54922	512
7	1.24E+18	183.84717	0.1736942	19.43133	18.46779	18.16451	18.01475	18.04155	752	301	4	269	3.65E+17	STAR	0.0003146	324	51666	594
8	1.24E+18	183.86438	0.0192007	19.38322	17.88995	17.10537	16.66393	16.36955	752	301	4	269	3.23E+17	GALAXY	0.1002423	287	52023	559
9	1.24E+18	183.90008	0.1874733	18.97993	17.84496	17.38022	17.20673	17.07071	752	301	4	269	3.72E+18	STAR	0.0003148	3306	54922	515
10	1.24E+18	183.92459	0.0972458	17.90616	16.97172	16.67541	16.53776	16.47596	752	301	4	270	3.64E+17	STAR	8.91E-05	323	51615	595

Figure 2

The dataset is imbalanced most of the entries are galaxies then stars and only 8.5% are quasars:

GALAXY	4998
STAR	4152
QSO	850

Figure 3

- **Outliers, missing values, duplicates:**

The dataset has little outliers to almost no outliers worth removing. It also has no missing or duplicate values.

- **Features:**

The original dataset has 18 columns, one column is the target variables (class), 13 colmns are features, the other four columns are IDs and constants that I have dropped from the dataset. Most of the features are numeric continuous data since they are measurements except Camcol feature which is of a categorical numerical type. The 10,000 objects to be classified are described by basic photometric and spectrometric data about galaxies, quasars and stars which will be used as features in this classification [2]:

- Redshift: the displacement of the spectrum of the object toward longer (red) wavelengths meaning that the greater the redshift manifested by light emanating from such an object, the greater the distance of the object and the larger its recessional velocity making it such an important feature when it comes to identify the celestial objects we have in this project.
- Thuan-Gunn astronomic magnitude system: u, g, r, i, z : representing the response of the 5 bands of the telescope as a way of characterizing the brightness of astronomical sources.
- Run, camcol and field: features which describe a field within the object image taken by the SDSS.
- Right Ascension and Declination: astronomical coordinates specify the direction of a point on the celestial sphere in the equatorial coordinate system.
- Plate: each spectroscopic exposure employs a large, thin, circular metal plate that positions optical fibers via holes drilled at the locations of the images in the telescope focal plane.
- MJD: Modified Julian Date, used to indicate the date that a given piece of SDSS data (image or spectrum) was taken.

This heat map shows the correlation between the 13 features. Most of them have weak correlations to each other's so I will be working on all the 13 features:

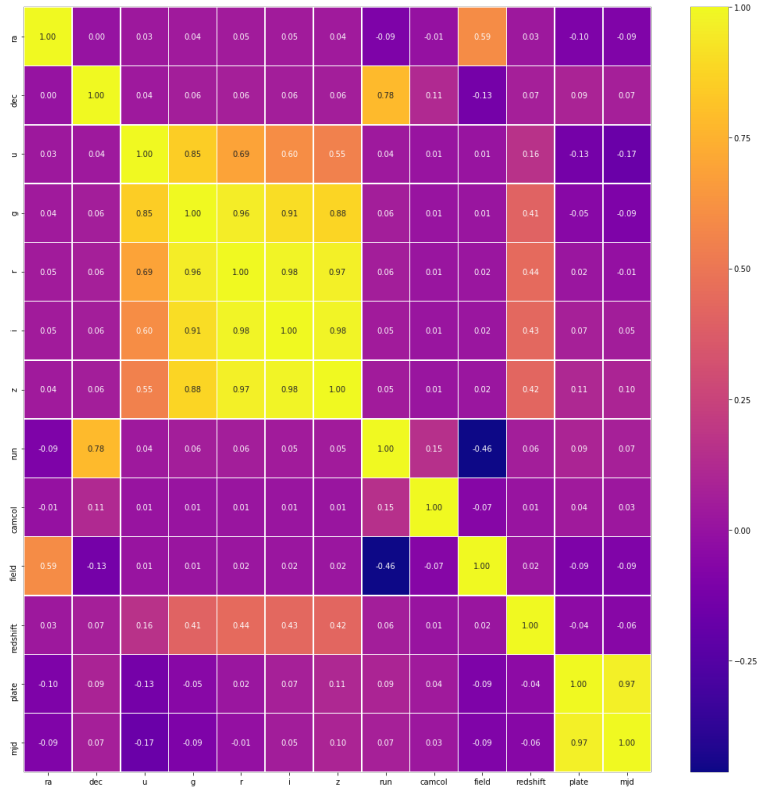

Figure 4

- **Normalization:**

Used MinMax Scaler for normalization to make all features be in a scale between 0 and 1.

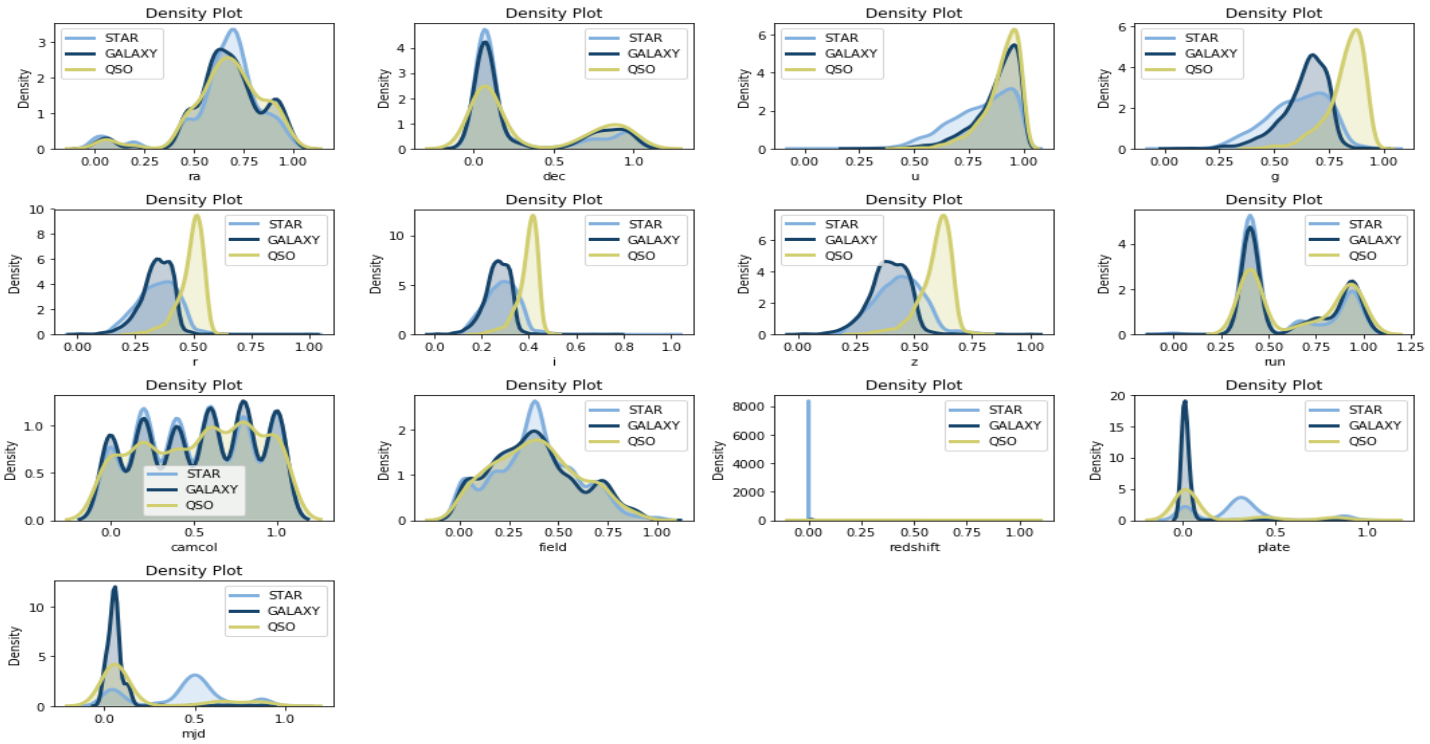

Figure 5

Exploratory Visualization

Redshift is an important feature in this classification and it is a main characteristic of the celestial objects to be classified and it differs greatly between them:

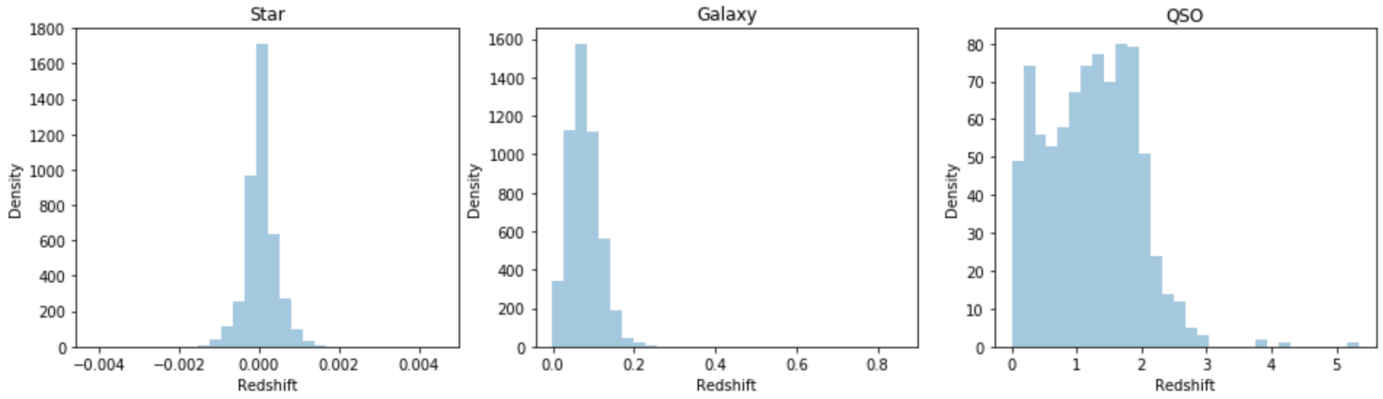

Figure 6

Mean Redshift of Stars: `0.080325` Mean Redshift of Galaxies: `0.000043` Mean Redshift of Quasars: `1.218366`

The visualization below shows all features in different graphs and the correlations between all features in scatter plots, blue is for Stars, green for Quasars and orange for Galaxies:

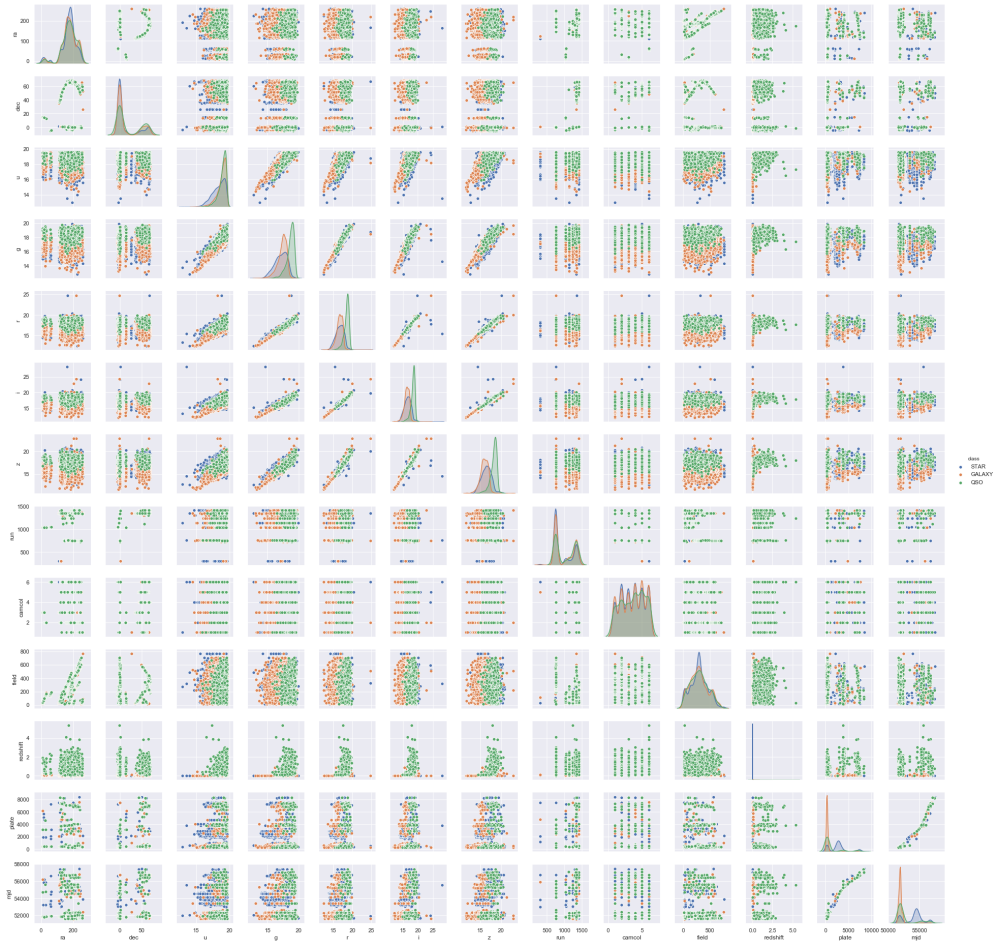

Figure 7

Algorithms and Techniques

- **Grid Search Cross Validation Technique:**

Grid Search works by evaluating all the combinations from a list of desired hyper-parameters specified for a given algorithm and reports which combination has the best accuracy.

Cross Validation used here to solve the problem of an imbalanced dataset by dividing the training set into k folds and then try the different combinations where each of the combinations will use a different fold as the test set and the remaining k-1 folds as the train set. k is the desired number of folds, I have seen it usually works well with k=5 [3].

Figure 8

I have three supervised learning algorithms I am working on in this project to classify the celestial objects into galaxies, stars and quasars and because of the many parameters I have for each one of these algorithms a technique like grid search is vital. Also, because of the imbalanced dataset I have at hand cross validation will solve this problem so that the models do not become biased in favor of the classes that has more observations belonging to them.

Supervised Algorithms:

- **Support Vector Machine:**

SVM is a supervised learning algorithm used mostly for classification, like the problem I have in this project. A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. A good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

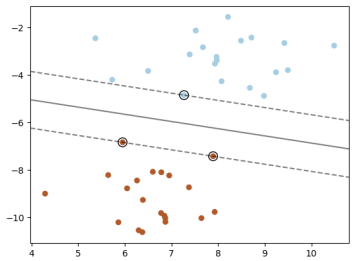

Figure 9

SVM is effective in high dimensional spaces, uses a subset of training points in the decision function (called support vectors) making it memory efficient, different Kernel functions can be specified for the decision function to deal with non-linear and multi-class problems [4].

SVMs are a suitable algorithm for the problem I have in this project for they are a classification algorithm, works fine with high dimensions, works well with non-linear and multi-class problems.

- **Random Forest:**

Random forests are a powerful ensemble model that can be used to build predictive models for both classification and regression problems. The model creates an entire forest of random uncorrelated decision trees to arrive at the best possible answer [5].

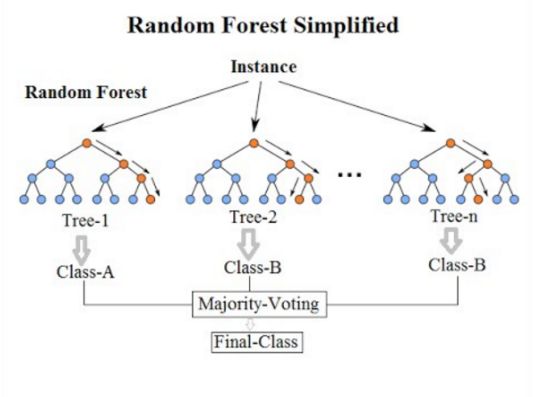

Figure 10

RF is a good algorithm to be used in the problem I have because it is a powerful algorithm and does not overfit easily.

- **Multilayer Perceptron Classifier:**

Multilayer perceptron classifier (MLPC) is a classifier based on the feedforward artificial neural network. MLPC consists of multiple layers of nodes. Each layer is fully connected to the next layer in the network. Nodes in the input layer represent the input data. All other nodes map inputs to outputs by a linear combination of the inputs with the node's weights and bias b and applying an activation function [6].

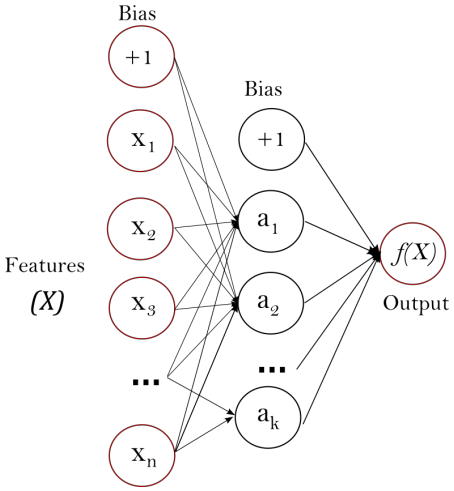

Figure 11

MLPC algorithm is a good candidate when the problem is classification like the one in this project because of the different activation functions that can be used to get the best results.

Unsupervised Algorithms:

Although the main problem I have in this project is classification, I thought it would be interesting to see the results unsupervised algorithms yield.

- **K-Means:**

It is an unsupervised learning algorithm used mostly for clustering. It works by grouping similar data points together and discover underlying patterns. To achieve this objective, K-means looks for a fixed number (k) of clusters in a dataset defining a target number k, which refers to the number of centroids needed in the dataset. A centroid is the imaginary or real location representing the center of the cluster. The picture below shows four clusters and four centroids [7]:

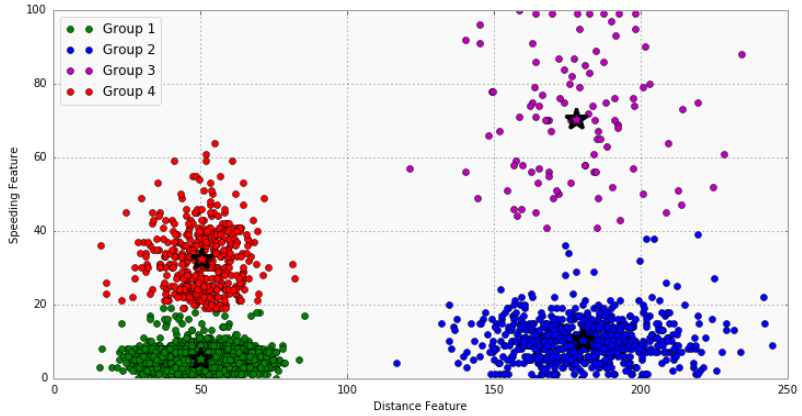

Figure 12

- **Gaussian Mixture:**

An unsupervised learning technique was developed to solve the problem k-means clustering has. A probabilistic approach to clustering works by estimating Gaussian distribution parameters such as mean and Variance for each cluster and weight of a cluster. After learning the parameters for each data point, the probabilities of it belonging to each of the clusters is calculated [8].

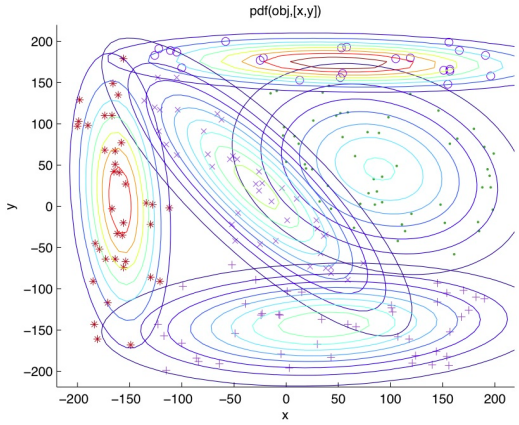

Figure 13

Benchmark

Many similar works have been done to solve this problem or another similar problem given data close to the data I am working with. However, I found most of them were published in papers and journals that require payment for access and read so I referred to open platforms like kaggle.com to choose a benchmark from:

Classifier:	Accuracy:
LightGBM [9]	99.121%
Random Forest [9]	98.969%
Naïve Base [9]	86.393%
Support Vector Machine (RBF Kernel) [10]	97.47%

III. Methodology

Data Preprocessing

- **MinMax Scaler for Normalization Process:**

I have used MinMax Scaler for feature transformation to yield better results. This transformation is often used as an alternative to zero mean, transforming features by scaling each feature to a given range, in my case between 0 and 1. Normalization is important for it makes all the features in the same range making it easier for the algorithms to pick patterns when all the features are equal on the same scale, hence better results. This normalization does not delete outliers or make the features normally distributed as this kind of normalization like Log Normalization is not suitable to be applied on the kind of data I have.

- **Correlation:**

I intuitively dropped columns that the model might cheat from, like ID columns, three of them were in the dataset. I also dropped a useless column ('rerun') that has a constant value. I then used pairwise correlation to see if any of the rest of the columns is highly correlated to any of the others.

- **Outlier detection:**

The common outlier formula in a boxplot to give a good idea if there are many outliers. There was very few outliers in some of the columns dropping them would only cause further imbalance. The following boxplot is of the **Right Ascension** and **Declination** features before Normalization:

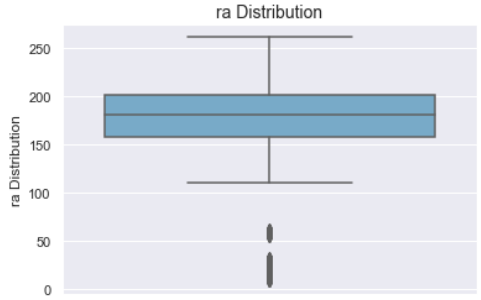

Figure 14

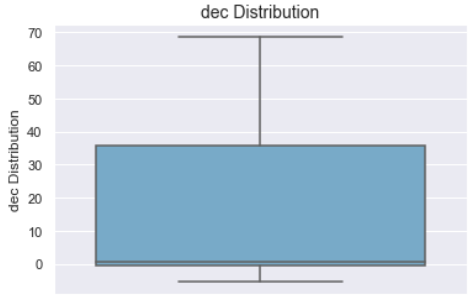

Figure 15

Implementation

After reading the dataset and visualizing to understand the data and after the pre-processing steps specified above, comes the classification and supervised learning implementation that I will be showing here where modules and classes imported will be included:

- **Splitting the dataset:**

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(ndata, cdata['class'], test_size = 0.33)
```

- **First classification model, SVM:**

```
#SVM classifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

clf1 = SVC(random_state = 42)

param1 = {'C': [0.1,1, 10, 100, 1000], 'gamma': [1,0.1,0.01,0.001,0.0001], 'kernel': ['rbf']}

grid1 = GridSearchCV(clf1, param1, cv = 5)

grid_fit1 = grid1.fit(X_train, y_train)

best_clf1 = grid_fit1.best_estimator_

print(grid_fit1.best_estimator_)
```

SVM model was created, then using Grid Search and Cross Validation the best model parameters were estimated to be:

```
SVC(C=1000, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.1, kernel='rbf',
  max_iter=-1, probability=False, random_state=42, shrinking=True,
  tol=0.001, verbose=False)
```

Best estimated SVM model is now trained on the data.

- **Second classification model, Random Forest:**

```
#Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier

clf2 = RandomForestClassifier(random_state = 42)

param2 = {'min_samples_split' : [2,4,8,15], 'bootstrap' : [True, False], 'min_samples_leaf' : [1,2,3,5]}

grid2 = GridSearchCV(clf2, param2, cv = 5)

grid_fit2 = grid2.fit(X_train, y_train)

best_clf2 = grid_fit2.best_estimator_

print(grid_fit2.best_estimator_)
```

Random Forest model was created, then using Grid Search and Cross Validation the best model parameters were estimated to be:

```
RandomForestClassifier(bootstrap=False, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=3, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob score=False, random state=42, verbose=0, warm start=False)
```

Best estimated RF model is now trained on the data.

- **Third classification model, MLPC:**

```
#MLP classifier
from sklearn.neural_network import MLPClassifier

clf3 = MLPClassifier()

param3 = {"activation": ["tanh", "logistic", "relu"]}

grid3 = GridSearchCV(clf3, param3, cv = 5)

grid_fit3 = grid3.fit(X_train, y_train)

best_clf3 = grid_fit3.best_estimator_

print(grid_fit3.best_estimator_)
```

MLPC model was created, then using Grid Search and Cross Validation the best model parameters were estimated to be:

```
MLPClassifier(activation='tanh', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(100,), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=False, warm_start=False)
```

Best estimated MLPC model is now trained on the data.

No major problems were encountered during building and training the models. Grid Search really made it easier and more effective to specify the parameters which are considered one of the most difficult to get just right. One small misunderstanding on my part was that I wanted to partition the dataset into k-folds before building the models or implementing Grid Search, I have tried doing that and it took some of my time trying until I figured that SciKit has it in their Grid Search class where I can specify the number of folds.

- **K-means Clustering:**

Three clusters, where X is the dataset unlabeled. Predicted clusters are stored in Preditions1.

```
from sklearn.cluster import KMeans

clu1 = KMeans(n_clusters = 3)
clu1.fit(X)

Prediction1 = clu1.predict(X)
```

- **Gaussian Mixture Clustering:**

Same as K-means, three clusters, where X is the dataset unlabeled. Predicted clusters are stored in Preditions2.

```
from sklearn.mixture import GaussianMixture

clu2 = GaussianMixture(n_components = 3)
clu2.fit(X)

Prediction2 = clu2.predict(X)
```

Refinement

I have used Grid Search to specify the best model parameters and Cross Validation to deal with the imbalanced dataset and these two techniques have made considerably better results as I have shown the difference they made in the accuracy of the models in the Implementation section of the document. As for the SVM model, the difference in results were from 86% to 97%, however, it made little to no difference to the Random Forest classifier.

IV. Results

Model Evaluation and Validation

- **Supervised Learning Models:**

The three classification models were trained on cross validation folds as a measure to combat the imbalanced dataset. The models also gave excellent precision, recall and F1-scores for each of the three classes:

SVM

	precision	recall	f1-score	support
GALAXY	0.98	0.98	0.98	1668
QSO	0.98	0.94	0.96	284
STAR	0.98	0.99	0.99	1348

RF

	precision	recall	f1-score	support
GALAXY	0.98	0.99	0.99	1677
QSO	0.98	0.91	0.94	277
STAR	0.99	1.00	0.99	1346

MLPC

	precision	recall	f1-score	support
GALAXY	0.97	0.98	0.98	1677
QSO	0.97	0.90	0.93	277
STAR	0.98	0.99	0.98	1346

For each algorithm, two models are tested: the model with best parameters estimated by Grid Search and the model without parameters estimation. Their results are compared here:

SVM	RF	MLPC
SVM Predictions without using GridSearchCV: Accuracy: 0.8718 Classification Report: precision recall f1-score support GALAXY 0.82 0.97 0.89 1668 QSO 1.00 0.75 0.86 284 STAR 0.95 0.77 0.85 1348 avg / total 0.89 0.87 0.87 3300 SVM Predictions using GridSearchCV: Accuracy: 0.9818 Classification Report: precision recall f1-score support GALAXY 0.98 0.98 0.98 1668 QSO 0.98 0.94 0.96 284 STAR 0.98 0.99 0.99 1348 avg / total 0.98 0.98 0.98 3300	RF Predictions without using GridSearchCV: Accuracy: 0.9879 Classification Report: precision recall f1-score support GALAXY 0.98 0.99 0.99 1677 QSO 0.97 0.92 0.94 277 STAR 1.00 1.00 1.00 1346 avg / total 0.99 0.99 0.99 3300 RF Predictions using GridSearchCV: Accuracy: 0.9864 Classification Report: precision recall f1-score support GALAXY 0.98 0.99 0.99 1677 QSO 0.98 0.91 0.94 277 STAR 0.99 1.00 0.99 1346 avg / total 0.99 0.99 0.99 3300	MLP Predictions without using GridSearchCV: Accuracy: 0.9703 Classification Report: precision recall f1-score support GALAXY 0.98 0.96 0.97 1677 QSO 0.97 0.92 0.94 277 STAR 0.96 0.99 0.98 1346 avg / total 0.97 0.97 0.97 3300 MLP Predictions using GridSearchCV: Accuracy: 0.9748 Classification Report: precision recall f1-score support GALAXY 0.97 0.98 0.98 1677 QSO 0.97 0.90 0.93 277 STAR 0.98 0.99 0.98 1346 avg / total 0.97 0.97 0.97 3300

All three models performed very well yielding high accuracies, precisions and f1-scores. The table of results above also shows the accuracies of the models when predicting each of the three classes: Galaxy, Star and Quasar.

- **Unsupervised Learning Models:**

K-means scored a higher Silhouette score than Gaussian Mixture:

K-Means	Gaussian Mixture
Silhouette Score: 0.8174301505857815	Silhouette Score: 0.5762740132214409

Plotting of z and u below shows the clustering results, where 0 for stars, 1 for galaxies and 2 for quasars:

Actual Plot of z and u	K-means	Gaussian Mixture

Figure 16

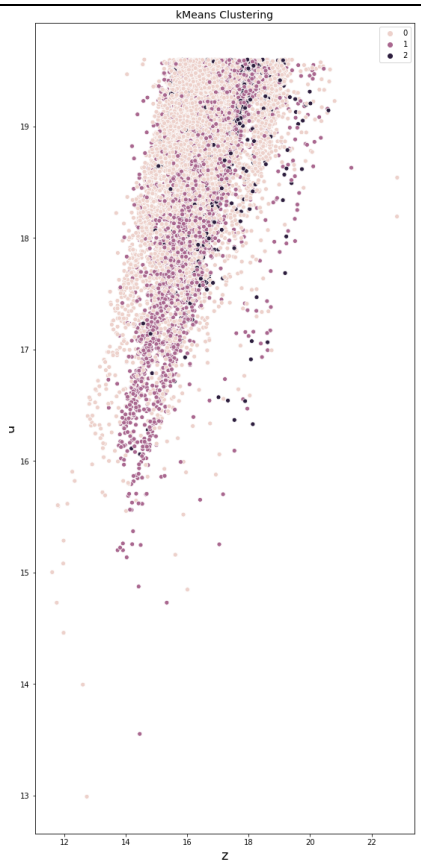

Figure 17

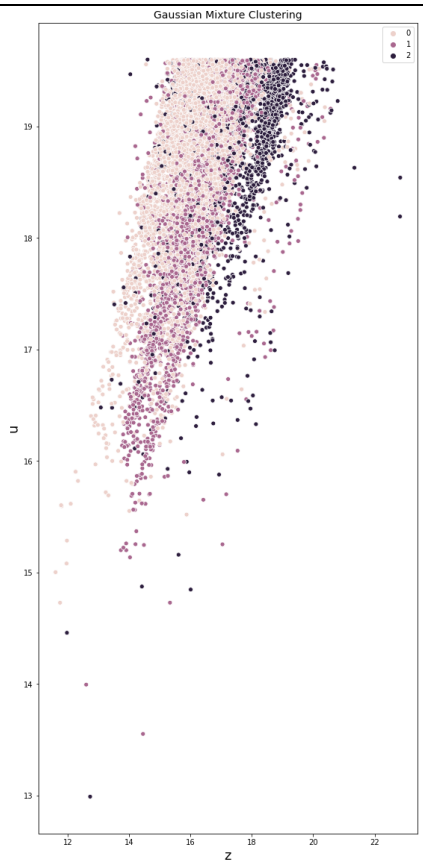

Figure 18

Justification

This Project's models:		Benchmark's models:	
SVM's Accuracy:	98.18%	SVM's Accuracy [10]:	97.47%
RF's Accuracy:	98.64%	RF's Accuracy [9]:	98.96%

The models in this project have achieved results same as or stronger than benchmark models compared above. Accuracies higher than 95% in classification problems are significant solutions and even higher accuracies could be achieved provided larger datasets.

V. Conclusion

Free-Form Visualization

Heat map of each Confusion Matrix of each classifier showing the numbers of actual observations predicted as which class of the three classes:

SVM

Figure 19

SVM has predicted 37 galaxies as stars (out of 4998 record) and 11 quasars (out of 850 record) as galaxies, which are the two highest numbers of mistakes made by the classifier.

RF

Figure 20

While RF has predicted 15 galaxies (out of 4998 record) as stars and 18 quasars as galaxies, which are the two highest numbers of mistakes made by the classifier.

MLPC

MLPC has predicted 37 galaxies (out of 4998 record) as stars and 11 (out of 850 record) quasars as galaxies, the same mistakes as SVM.

Reflection

I have wanted to work on a similar problem, a problem related to astronomy or physics and found this one to be interesting, fun to work with and actually the data was in good quality and available for the public. In the pre-processing step, there was not much work to be done on the dataset other than the MinMax scaling and dropping useless columns or columns that the models could cheat from. Little correlation was found between the features so I started implementing the algorithms. After models were built, I have testing them and then moved to implementing some unsupervised learning techniques to see how that would go.

The solution to the classification of Galaxy-Star-Quasar problem was achieved, the models built yielded high accuracies.

Improvement

I would really like to have worked on a larger dataset, I have tried in the beginning a dataset of 100,000 observations but that was computationally expensive to my machine. Having a large dataset to work with shows the real capabilities of the algorithms and maybe try deep learning to solve problems with such large datasets.