SDSS Classification

Diego McDonald

April 20th 2023

1 Brief Description

Modern astronomy is quickly becoming a data science-oriented field. With the staggering amounts of data being constantly generated by the world's telescopes, Astronomers are forced to learn data science techniques to manage and learn from data. Recent astronomy programs have moved from specialized telescopes designed to stare at small patches of sky, to survey telescopes designed to capture data from vast swaths of the sky. This change in operation is the reason data science has been added to the astronomer's toolbox. I will be adding to this toolbox by performing classification on a subset of the objects observed by the Sloan Digital Sky Survey (SDSS) in order to organize better the immense amount of data generated by astronomical surveys.

2 Problem Statement

`media/HST-SM4.jpeg`

Figure 1: Hubble Space Telescope

`media/sdss.jpg`

Figure 2: Sloan Digital Sky Survey

`media/LSST.jpg`

Figure 3: Large Synoptic Survey Telescope

Astronomy is changing from seeing very dim objects to see very wide, that is, many objects in a single frame. For example, the Hubble Space Telescope (HST) (Figure 1), arguably the most famous telescope due to its ability to see very dim and create stunning images of our Universe, generates on average 18GB of data per week. Meanwhile, the Large Synoptic Survey Telescope (LSST) (Figure 3) is planned to produce over 20TB worth of data per night. However, since the LSST is currently not in production (still being developed), our most recent survey telescope is the Sloan Digital Sky Survey (SDSS) (Figure 2), which produces 175GB of data per night. This study will use only a subset of the data from the SDSS.

3 Objective

The objective of this study is to prove the efficacy of machine learning and artificial intelligence algorithms in the classification of astronomical objects.

The objective of this study is exercise two strategies for organizing data coming from survey telescopes in an effort provide tools for organizing the vast amount of data available to astronomers. I will be using one data set, consisting of 100,000 objects, split between three different classifications: `STAR`, `GALAXY`, and `QSO` (Quasi-Stellar Object). The first strategy will be applying traditional machine learning algorithms to the processed photometric data and comparing the relative performance.

The second strategy will use a subset of images from the $100,000$ objects and apply a fully connected neural network (FCNN) and a convolutional neural network (CNN) to classify each object from their respective image "chips".

In summary, I am performing classification using two strategies:

1. Photometric Classification
2. Image Classification

4 Data Sets

The dataset that inspired this study can be found at this kaggle link. This data set contains $100,000$ records, corresponding to $100,000$ unique objects. Each record contains photometric intensity in 5 different bands: u, g, r, i, and z, all corresponding to different wavelengths, as well as $redshift$, a metric correlated to the distance of the object from the observer, us.

4.1 Photometric Classification

4.1.1 EDA

For the photometric classification strategy, I used to all of the previously mentioned features, (e.g. photometric features u, g, r, i, z and distance feature $redshift$) and first investigated their relative correlations (Figure 4). Noting the high correlation between all of the photometric features, I was only left to conclude that objects that were exceptionally bright in a specific wavelength were also considerably brighter in the other wavelengths. Potentially due to the relative distance between us, the observer, and the objects, but this would require further investigation. At the time of writing, I am sure if these values are range normalized.

Variable	Description
obj_ID	Object Identifier, the unique value that identifies the object in the image catalog used by the CAS
alpha	Right Ascension angle (at J2000 epoch)
delta	Declination angle (at J2000 epoch)
u[*]	Ultraviolet filter in the photometric system
g[*]	Green filter in the photometric system
r[*]	Red filter in the photometric system
i[*]	Near Infrared filter in the photometric system
z[*]	Infrared filter in the photometric system
run_ID	Run Number used to identify the specific scan
rerun_ID	Rerun Number to specify how the image was processed
cam_col	Camera column to identify the scanline within the run
field_ID	Field number to identify each field
spec_obj_ID	Unique ID used for optical spectroscopic objects (this means that 2 different observations with the same spec_obj_ID must share the output class)
class[**]	object class (galaxy, star or quasar object)
redshift[*]	redshift value based on the increase in wavelength
plate	plate ID, identifies each plate in SDSS
MJD	Modified Julian Date, used to indicate when a given piece of SDSS data was taken
fiber_ID	fiber ID that identifies the fiber that pointed the light at the focal plane in each observation

Table 1: Dataframe variables and descriptions. **Features**[*] and **Labels**[**] for training, validation, and testing.

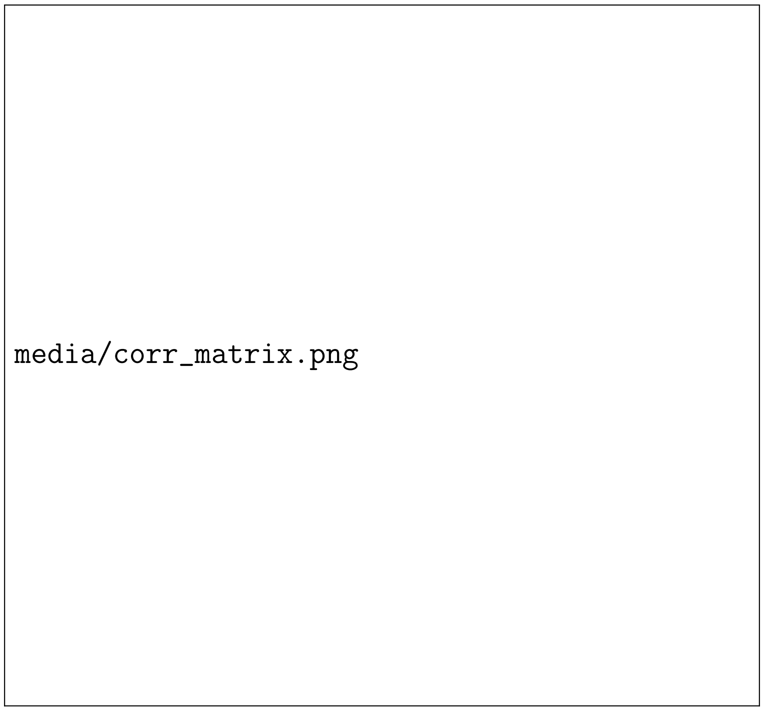


Figure 4: Correlation Matrix amongst features.

4.1.2 Preprocessing

Figure 5 shows the default feature distributions, prior to any scaling or transformations. One important feature to note is the multi-modal nature of each of the distributions, indicating that clustering algorithms may in fact be fruitful here towards achieving classification. I have only performed a cursory investigation into clustering and have not found any useful results, so this study will not consider clustering as a meaningful tool.

To normalize the data distributions, I apply a `StandardScaler` to each feature, excluding $redshift$, which requires more attention. These scaled distributions can be viewed in Figure 6.

Another important feature to mention would be the high skewness of the $redshift$ feature. $Redshift$ measures the change in observed wavelength from source to detection. This is similar to the Doppler effect experienced when a siren approaches and then retreats from the observer. In summary, a negative $redshift$ indicates an object is moving towards the observer, while a positive $redshift$ indicates an object moving away from the observer. The possibility of negative values alongside the high skewness of the $redshift$ feature posed a significant challenge for normalization. Looking at the equation for $redshift$ (Eq. 1), applying a bias of 1 still preserves the underlying meaning of the metric, removing the possibility of negative values and allowing for a `PowerTransform` to help normalize the data.

$$z = \frac{\lambda_{obs}}{\lambda_{rest}} - 1 \tag{1}$$

Again, figure 6 shows the final distributions after applying the `StandardScaler` to the u, g, r, i, z features and biasing and transforming the $redshift$ feature.

media/default_data_distribution.png

Figure 5: Default Feature Distributions

media/standardscale_redshift_bias_transform.png

Figure 6: Standard Scaler + Redshift Bias and Transformation

media/default_class_distributions.png

Figure 7: Default Class Distributions

media/smote_class_distributions.png

Figure 8: Balanced Class Distributions

Post scaling and normalization, I investigate the class distributions in Figure 7. Due to the significant imbalance between the three classes, I applied a SMOTE algorithm to generate new examples and even out the label distributions. Figure 8 shows the class balance after applying the SMOTE algorithm.

Finally, I convert all the labels to one-hot encodings and feed the transformed distributions into the subsequent models in Section 9

4.2 Image Classification

With the Photometric Classification data set processed, I began working on ways to pull the associated images from each of the examples in the photometric dataset. In order to do so, I took advantage of the Python packages, AstroPy and AstroQuery:

4.2.1 Pulling Data

Each of the examples in the photometric dataset also include unique identifiers for specific images taken from the SDSS. I am able to take the `alpha` (right ascension) and `delta` (declination) for each example and to pinpoint the precise location of each object in the night sky. The object position, coupled with the unique identifiers mentioned in Table 1, provide enough information to pass into AstroQuery to download the respective image for each of the 100,000 objects in each of the 5 bands observed (u, g, r, i, z), approximately 500,000 images. Figure 9 provides an example of a normalized image from the SDSS.

4.2.2 Preprocessing

After each image was pulled, I created image "chips," small cutouts of images containing only the object of interest, using a centroid defined as a 2-dimensional Gaussian limited to pixels within 3σ. An example "chip" extraction can be viewed in Figure 10. These chips were then scaled and augmented as shown in Figure 11

Finally, in order to address the label imbalance present in the original data set (see Figure 7.), both the `STAR` and `GALAXY` classes were randomly downsampled to contain only the same number of examples as the `QSO` class.

With the aforementioned scaling, preprocessing, and augmentation finished, the next step was model evaluation.

5 Methodology

5.1 Photometric Classification

The methodology for training traditional machine learning models on the tabular photometric data was painless. The initial dataset was split via 60/20/20 for training/validation/testing, and each subsequent model was trained and then evaluated while also returning helpful metrics along the way to evaluate each model's performance. Each model had hyper-parameters listed in Table 2 media/normalized_example_image_34086.png

Figure 9: Example Image (Normalized) from SDSS.

media/example_image_chip.png

Figure 10: Example Image (Normalized) from SDSS with extracted chip in red.

media/augmented_chip.png

Figure 11: Example Augmentation to an Image Chip

5.2 Image Classification

The methodology to achieve Image Classification was slightly more convoluted. I developed four different neural networks, two fully connected neural networks and two convolutional neural networks, all using slightly different hyper-parameters. For each type of neural network, I first trained and validated an initial model in order to get a benchmark for model performance. Once this initial model was created, I performed the HyperBand hyper-parameter tuning algorithm using `keras-tuner` to maximize accuracy. Table 4 shows the results of hyper-parameter tuning.

6 Block Diagram

media/sdss_classification block diagram_nn.png

Figure 12: Block Diagram for the traditional machine learning models on tabular photometry data.

7 Success/Failure

The measure of success or failure for this project fairly forgiving. Since the training data is processed to become evenly distributed amongst 3 classes (i.e. each class makes up $33.3\bar{3}\%$ of the data), if we were to guess every example as a single class, say `GALAXY`, we would have a $33.3\bar{3}\%$ accuracy. Therefore, anything above a $33.3\bar{3}\%$ accuracy is considered a success.

8 Evaluation Parameters

8.1 Photometric Classification

All of the traditional machine learning models used for photometric classification measured accuracy, recall, precision, false negative rate, and f1 score. For a more holistic evaluation of the models, the true positive rate (TPR) vs false positive rate (FPR) was calculated, resulting in a ROC (receiving operating characteristic) curve. Additionally, the area under the curve (ROC AUC) is calculated using both "one-vs-one" and "one-vs-rest" options to ascertain each individual models performance better. Reference figures 13 and 14 to see how this is applied.

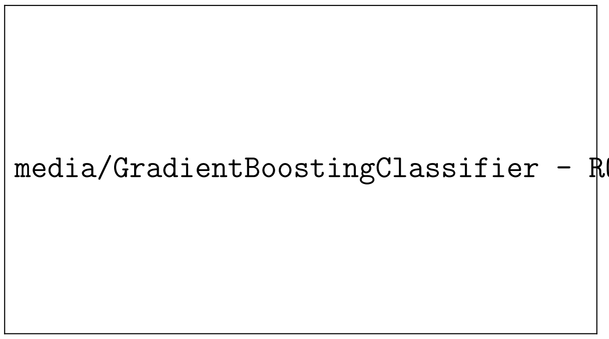


Figure 13: ROC Curve for GradientBoostingClassifier

Figure 14: ROC Curve for RandomForestClassifier

8.2 Image Classification

Each of the neural networks used to classify object image chips used validation accuracy and validation loss for evaluation while training. Once suitable hyper-parameters were determined, each model was then tested against the testing set in order to measure a final testing accuracy. In doing so, we are still able to compare the relative performance of each model to be subsequently applied to an example SDSS image, showing a real world example of how these models can be used. Reference Figure 16 for such an example.

9 Experiments

9.1 Photometric Classification

The above traditional machine learning models were trained on a training set and then validated separate from the testing set in order to preserve the generalizability of the models. This is especially meaningful considering the `VotingClassifier` in Table 2. It is important to use only the validation data set during model analysis in order to determine the best performing models to put into the `VotingClassifier`. The results of experimenting with different models can be viewed in Table 3. Note the higher accuracies of the `RandomForestClassifier` and `GradientBoostingClassifier` as well as accuracy and ROC

AUC score of the `VotingClassifier`. The improved ROC AUC score indicates improved model performance by combining `RandomForestClassifier` and `GradientBoostingClassifier`.

Table 2: Models and their respective hyper-parameters. ***Note:*** Any hyper-parameters not listed are assumed to be default values.

Model	Hyper-parameters
`LogisticRegression`	multiclass = "multinomial"
`DecisionTree`	*Defaults*

`GradientBoosting`	max_depth = 10
	n_estimators = 10
	criterion = "squared_error"
	min_samples_leaf = 1000

`RandomForest`	n_trees = 10
`SupportVectorMachine`	*Defaults*
`KNearestNeighbors`	n_neighbors = 5
`VotingClassifier`	`RandomForest` + `GradientBoosting`

Table 3: Model results evaluating on validation set.

Model	Accuracy	Recall	Precision	$ROCAUC_{OVR}$
`LogisticRegression`	0.953	0.953	0.954	0.989
`DecisionTree`	0.960	0.960	0.962	0.970
`GradientBoosting`*	0.968	0.968	0.969	0.994
`RandomForest`*	0.978	0.978	0.978	0.993
`SVC`	0.969	0.969	0.970	0.993
`KNeighbors`	0.961	0.961	0.962	0.985
`VotingClassifier`*	0.975	0.975	0.975	0.996

9.2 Image Classification

Model	Trainable Parameters	Epochs
Fully Connected NN	1,327,619	5
Fully Connected NN - HyperModel	9,981,443	3
Convolutional NN	16,836,611	5
Convolutional NN - HyperModel	501,075	6

Table 4: Models and their trainable parameters.

The epoch numbers for the Fully Connected Neural Network (FCNN) and the Convolutional Neural Network (CNN) in Table 4 were hand picked at 5 epochs, providing a good balance between validation accuracy and overfitting. Looking at the trainable parameters further, it is worth noting the FCNN-HyperModel had almost 10 million trainable parameters, but only required 3 epochs to achieve sufficient accuracy without over-fitting. Similarly, the inverse can be see with the CNN-HyperModel's 500,000 trainable parameters, requiring one extra epoch to achieve similar accuracy.

10 Results

Table 5: Model accuracies evaluating on training, validation, and testing sets.

Model	Training	Validation	Testing
`VotingClassifier`	0.969	0.969	0.970
`Fully Connected NN`	0.787	0.8146	0.805
`Fully Connected NN - HyperModel`	0.541	0.791	0.784
`Convolutional NN`	0.822	0.808	0.797
`Convolutional NN - HyperModel`	0.808	0.817	0.806

With the results in Table 5, we can see that the `VotingClassifier` model performed exceptionally well, testing at 97% accuracy trained on the tabular photometry data. The deep learning algorithms trained on associated image chips had a significantly lower accuracy, 80%. It is important to note that each approach has been trained on vastly different data sets. While the deep learning models are trained on processed images, I would expect a significant improvement in accuracy should we decide to increase the number of training samples.

11 Discussion

Looking at the results in Table 5, we can easily conclude the traditional machine learning algorithms outperformed the deep learning algorithms. This discrepancy in accuracy is due to the different data sets used for each solution.

The traditional machine learning algorithms were trained and tested on tabular photometric data, proving to be relatively simple to learn from. However, deep learning models were trained on images requiring significant preprocessing in order to be useful. The generation of image chips, while mostly successful, still proved challenging. Figure 17 shows three example image chips containing multiple closely spaced objects. These closely spaced objects make it difficult to discern exactly which is the object of interest, adding to the complexity of the image processing pipeline. There are strategies for "deblending" such images that would assuredly result in increased deep learning model performance.

Additionally, the deep learning models were not trained on all 100,000 objects. Each deep learning model was trained on approximately 75,000 objects, or roughly 375,000 images. This large dataset should still give us quite a bit of information on how to distinguish between a `STAR`, `GALAXY`, and `QSO`, considering the 80% accuracy. Improving the quality of the training images along with providing more images to learn from would result in a significant performance in deep learning models to be hopefully on par with traditional machine learning models trained on photometry.

12 Constraints

Time was the most significant constraint in this analysis. While downloading the initial tabular data of photometry was swift and straightforward, the association of each object in the table with an image from the SDSS, along with the accompanied image processing, took significantly longer. In order to create image chips for each of the 100,000 objects the entire 1489x2048 image must be downloaded just to extract image chips sometimes less than 10 pixels wide. Furthermore, each object is imaged in five different bands, so each object requires 5 images at 1489x2048 resolution to be associated with an image chip. This process took many days, and this study still was unable to make use of all the images available for the 100,000 objects.

13 Standards

This study makes use of quite a bit of third-party Python packages. Machine Learning and Neural Network training and evaluation was performed using Tensorflow's Keras API. Neural Network hyper-parameters were tuned using Keras-Tuner. Traditional Machine Learning models were trained using SciKit-Learn. Additionally, once the photometry data set was downloaded, I leveraged AstroPy's coordinate transformations (i.e. going from world coordinates to pixel coordinates) and AstroQuery to download data directly from the SDSS API. Finally, I used AstroPy's PhotUtils package to perform much of the image processing, including background subtraction, noise reduction, and image segmentation via source finding.

14 Comparison

Figure 15 shows the ROC curve for the `VotingClassifier` model, a combination of both the `RandomForestClassifier` and the `GradientBoostingClassifier` The high AUC values for each class indicate phenomenal performance. However, when we compare to the application of the CNN-HyperModel in Figure 16, we see a significant amount of noise and potential incorrect classifications.

It is important to note that these two approaches use vastly different data sets to arrive at similar conclusions, but with significantly different accuracies. Additionally, it is worth noting that the quality of data that the CNN-HyperModel is trained on is quite variable. This variation in image chips is partly responsible for the relatively poor performance of the neural networks as a whole. Should we invest in improving the data quality the neural networks are trained on, I would expect their respective accuracies to increase.

`media/VotingClassifier - ROC Curve.png`

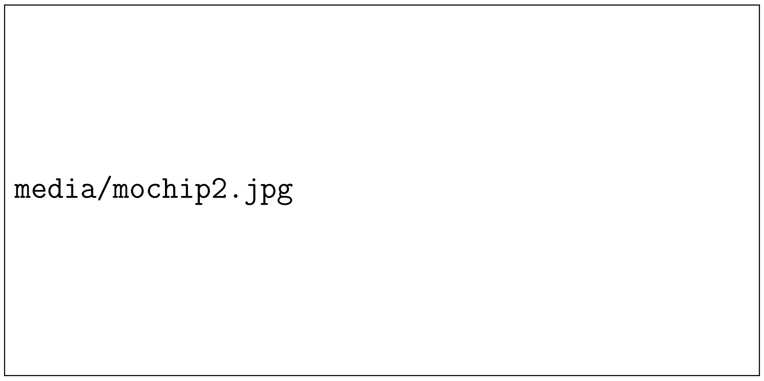
Figure 15: ROC Curve for `VotingClassifier`

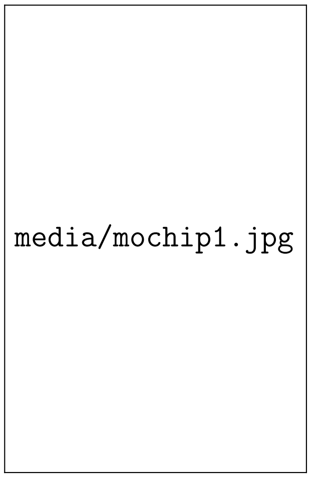
`media/normalized_example_image_bboxes_classified_`

Figure 16: ROC Curve for CNN-HyperModel

15 Limitations

Many of the limitations in this study are due my inexperience in image processing for astronomy. I have learned quite a bit along the way in order to produce better quality images that can eventually be fed into a neural network, through background subtraction and noise reduction, however there are plenty more techniques and algorithms I can take advantage of to improve the quality of the data further. For example, Figure 17 shows an example image chip that was fed into the neural networks that contains multiple objects spaced closely together. There exist advanced algorithms to work on "deblending" these two objects, however





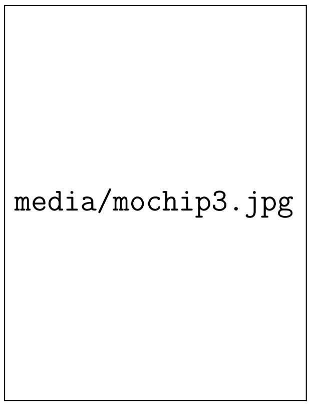


Figure 17: Examples of image chips with multiple objects.

given time considerations, this additional preprocessing lies beyond the scope of this project. However, I can confidently claim that the doing this additional preprocessing would result in a substantial increase in the above neural network performances.

16 Future Work

I would like to continue this analysis as I improve my understanding of astronomical image processing. While I am impressed by the 80% accuracy of image classification, I would love to investigate further how an improved data quality affects these results. I would also be interested in expanding the scope of this problem to include different and more specific types of astronomical objects. For example, differentiate between spiral, elliptical, and irregular galaxies, finding sources of gravitational lensing in images, and more. There a countless applications of data science in astronomy, and I am excited to continue learning more!