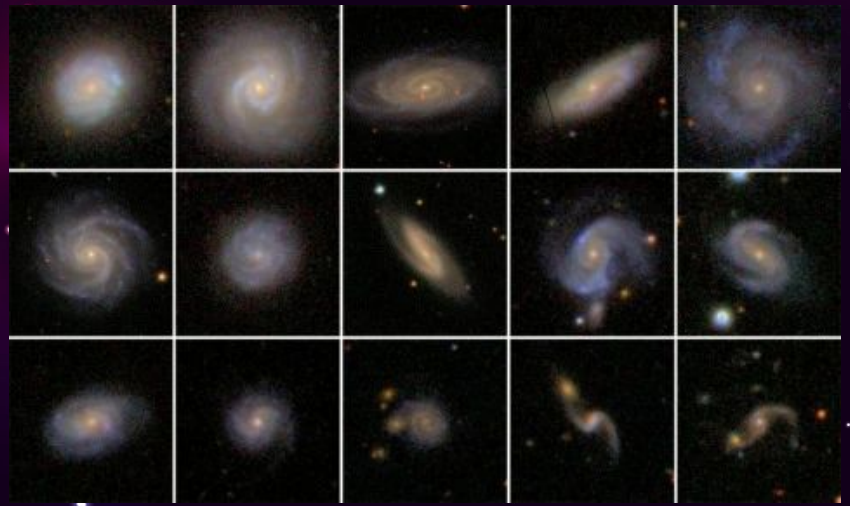
NSDC Project Fall 2023: Galaxy Classification

Gianna Pedroza, Aidan Nguyen, Tishi Avvaru, Benjamin Yu

The Dataset: Galaxy Zoo

In the original Galaxy Zoo project, volunteers classified images of Sloan Digital Sky Survey galaxies as belonging to one of six categories - elliptical, clockwise spiral, anticlockwise spiral, edge-on , star/don't know, or merger.



The Dataset: Galaxy Zoo Table

The table gives classifications of galaxies with the fraction of the vote in each of the six categories is given flags identifying systems as classified as spiral, elliptical or uncertain.

The Galaxy Zoo project collected simple classifications of nearly 900,000 galaxies drawn from the Sloan Digital Sky Survey with classifications given by hundreds of thousands of volunteers.

Goals:

1. Classify the galaxies based off of physical features as either, spiral, elliptical, or uncertain.
2. Find the model that best fit the dataset and tested out multiple classification models: kNN, Logistic Regression, Naive Bayes, and a Neural Network.
3. Learn about different libraries that can be used for classification and machine learning.

Cleaning the Data:

- To clean the data we first created a dataset using pandas to view the different features and removed features that were not useful for classification:
  - OBJID: is an id number for the given galaxy
  - RA: Right Ascension coordinate of the galaxy
  - DEC: Declination coordinate of the galaxy
  - NVOTE: How many people voted on the given galaxy
- There were also individual columns for the three classifications so we combined them into one column with 1 = spiral, 2 = elliptical, and 3 = uncertain.

	OBJID	RA	DEC	NVOTE
	587727178986356823	00:00:00.41	-10:22:25.7	59
	587727227300741210	00:00:00.74	-09:13:20.2	18
	587727225153257596	00:00:01.03	-10:56:48.0	68
	587730774962536596	00:00:01.38	+15:30:35.3	52
	587731186203885750	00:00:01.55	-00:05:33.3	59
	...	...	...	...

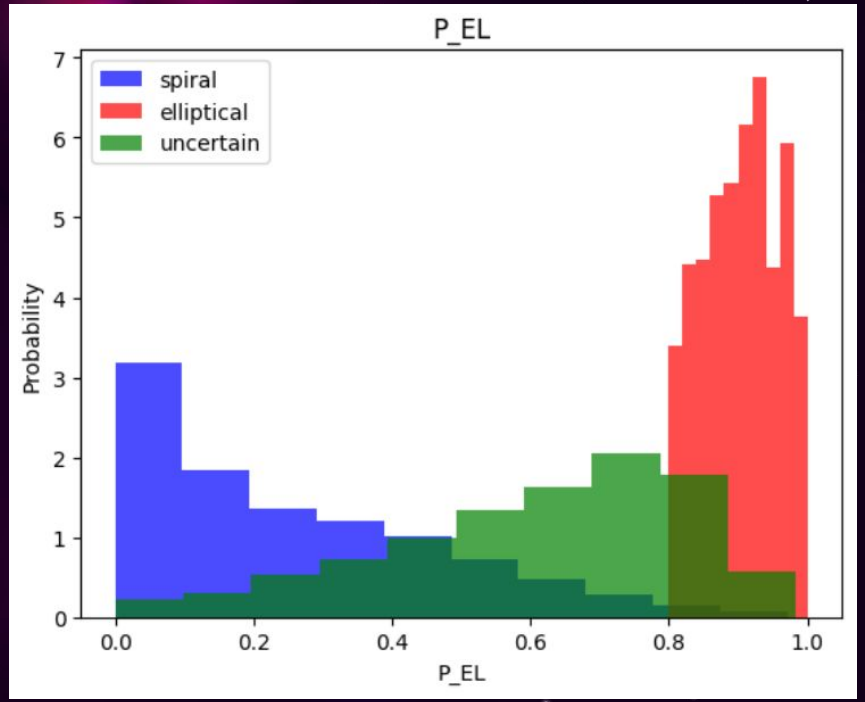
SPIRAL	ELLIPTICAL	UNCERTAIN
0	0	1
1	0	0
0	0	1
0	1	0
0	0	1
...	...	...

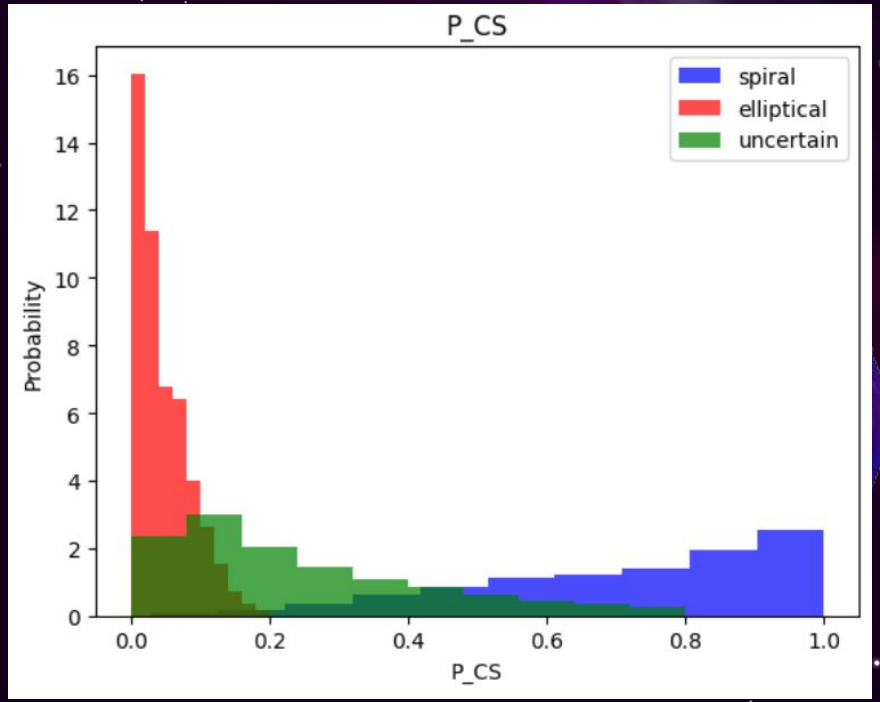
```
df['CLASS'] = df.apply(lambda x: 1 if x['SPIRAL'] == 1 else (2 if x['ELLIPTICAL'] == 1 else 3), axis=1)
```

	P_EL	P_CW	P_ACW	P_EDGE	P_DK	P_MG	P_CS	P_EL_DEBIASED	P_CS_DEBIASED	CLASS
0	0.610	0.034	0.000	0.153	0.153	0.051	0.186	0.610	0.186	3
1	0.611	0.000	0.167	0.222	0.000	0.000	0.389	0.203	0.797	1
2	0.735	0.029	0.000	0.147	0.074	0.015	0.176	0.432	0.428	3
3	0.885	0.019	0.000	0.058	0.019	0.019	0.077	0.885	0.077	2
4	0.712	0.000	0.000	0.220	0.068	0.000	0.220	0.640	0.290	3

Visualization:

- To visualize the features of all three classifications we created color coded histograms.





Model 1: K-Nearest Neighbors

- K-Nearest Neighbors (KNN) - a supervised learning classifier which uses proximity to make predictions or classifications about a data point
  - K refers to the amount of “neighbors” that will be checked to determine the classification of a certain point

```
[ ]  from sklearn.neighbors import KNeighborsClassifier

[ ]  knn_model = KNeighborsClassifier(n_neighbors = 5)
     knn_model.fit(x_train, y_train)

[ ]  y_pred = knn_model.predict(x_test)

[ ]  print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
1	0.80	0.92	0.86	37957
2	0.68	0.91	0.78	12298
3	0.95	0.84	0.89	83334
accuracy			0.87	133589
macro avg	0.81	0.89	0.84	133589
weighted avg	0.88	0.87	0.87	133589

Model 2: Logistic Regression

- Logistic Regression model - a supervised learning model often used for classification and predictive analysis
  - estimates the probability of an event occurring based on a given dataset of independent variables

```
[ ]  from sklearn.linear_model import LogisticRegression

[ ]  lg_model = LogisticRegression()
     lg_model.fit(x_train, y_train)

[ ]  y_pred = lg_model.predict(x_test)
     print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
1	0.81	0.93	0.87	37957
2	0.59	0.96	0.73	12298
3	0.96	0.81	0.88	83334
accuracy			0.86	133589
macro avg	0.79	0.90	0.83	133589
weighted avg	0.88	0.86	0.86	133589

Model 3: Naive Bayes

- Naive Bayes is a classification algorithm that uses the Bayes Theorem to predict the probability of an item to be a specific class.
  - This classification model is called “naive” because it assumes that the data features are independent of one another

$$P(class \mid data) = \frac{P(data \cap class)}{P(data)} = \frac{P(data \mid class) * P(class)}{P(data)}$$

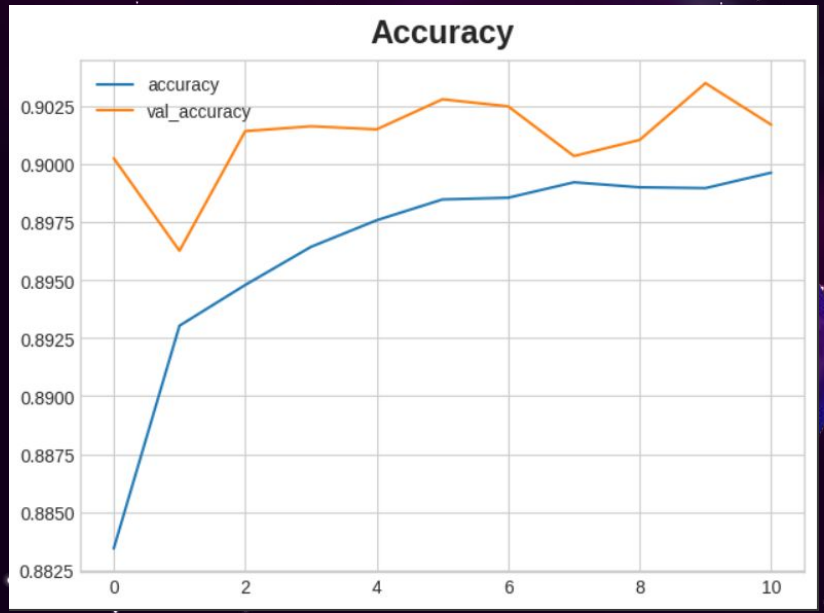
```
from sklearn.naive_bayes import GaussianNB

nb_model = GaussianNB()
nb_model.fit(x_train, y_train)
```

▾ GaussianNB
GaussianNB()

	precision	recall	f1-score	support
1	0.79	0.89	0.84	37679
2	0.46	0.97	0.62	12325
3	0.93	0.72	0.82	83585
accuracy			0.79	133589
macro avg	0.73	0.86	0.76	133589
weighted avg	0.85	0.79	0.80	133589

Model 4: Neural Network

- Network of neurons with associated weights designed to predict output
- Final Design: Tensorflow Sequential Model
- 90% accuracy on validation data



Comparisons:

	Model 1	Model 2	Model 3	Model 4
Accuracy	87%	86%	79%	90.35%
Precision	80% 68% 95%	81% 59% 96%	79% 46% 93%	

Citations

https://data.galaxyzoo.org/

https://ui.adsabs.harvard.edu/abs/2008MNRAS.389.1179L/abstract

https://ui.adsabs.harvard.edu/abs/2011MNRAS.410..166L/abstract

https://www.ibm.com/topics/naive-bayes#:~:text=the%20next%20step-,Na%C3%AFve%20Bayes%20classifiers,a%20given%20class%20or%20category.

https://youtu.be/i_LwzRVP7bg?si=DA-c40mJZ2RMKkaA