Stellar Classification

Dharma Hoy

What is Stellar Classification

- The classification of stars based on their spectral characteristics
- This project classifies galaxies, stars, and quasars
- Galaxy: a system of millions or billions of stars together with gas and dust held together by gravitational attraction
- Star: a fixed luminous point in the night sky which is a large, remote incandescent body like the sun
- Quasar: A massive and extremely remote celestial object emitting exceptionally large amounts of energy and typically having a starlike image in a telescope

Project Goal

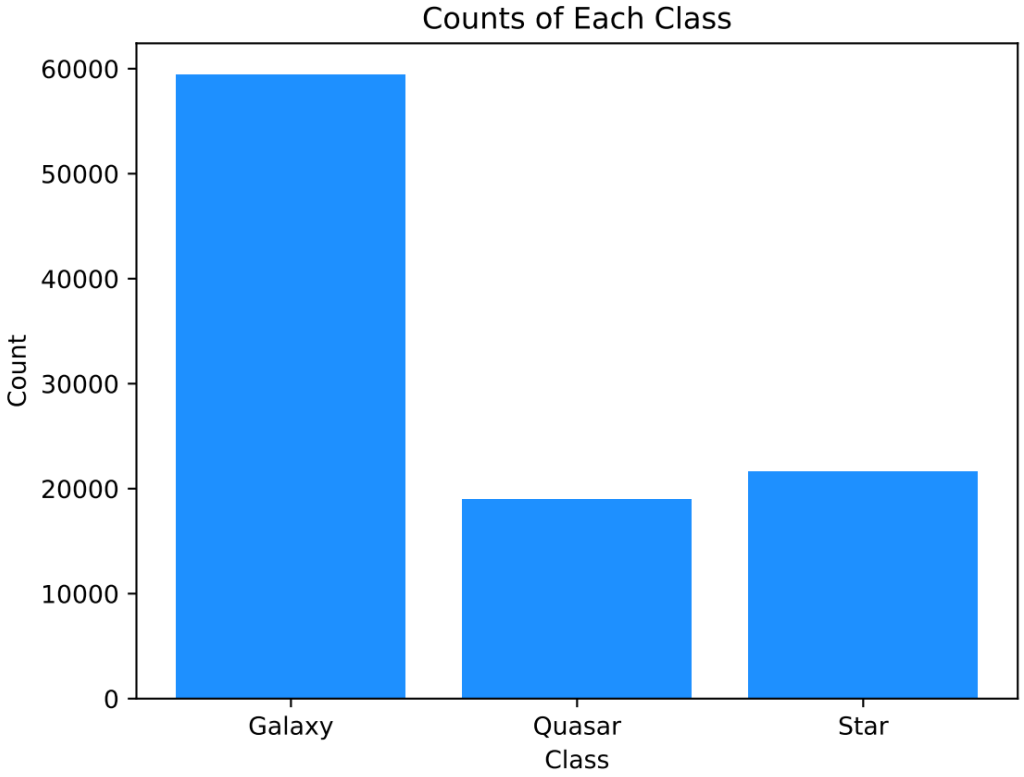
- Find a machine learning algorithm with at least 90% accuracy in predicting the class of an observation
- Find which features are the most important in predicting the class of the observation

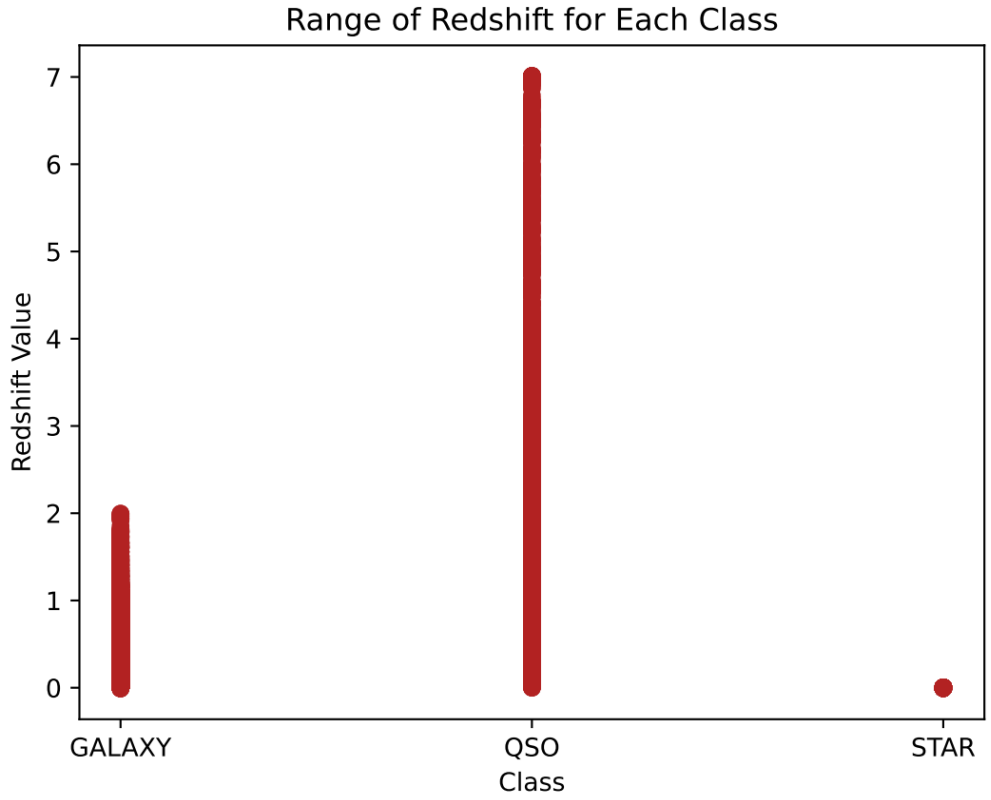
The Data

- 100,000 observations of space taken by the Sloan Digital Sky Survey found on Kaggle
- Every observation is described by 17 features and is identified to be a star, galaxy, or quasar
- The features used for analysis
  - Alpha = Right Ascension angle
  - Delta = declination angle
  - U, G, R, I, Z = ultraviolet, green, red, near infrared, and infrared filter in the photometric system respectively
  - Redshift = redshift value based on the increase in wavelength
  - MJD = modified Julian Date, used to indicate when the piece of data was taken

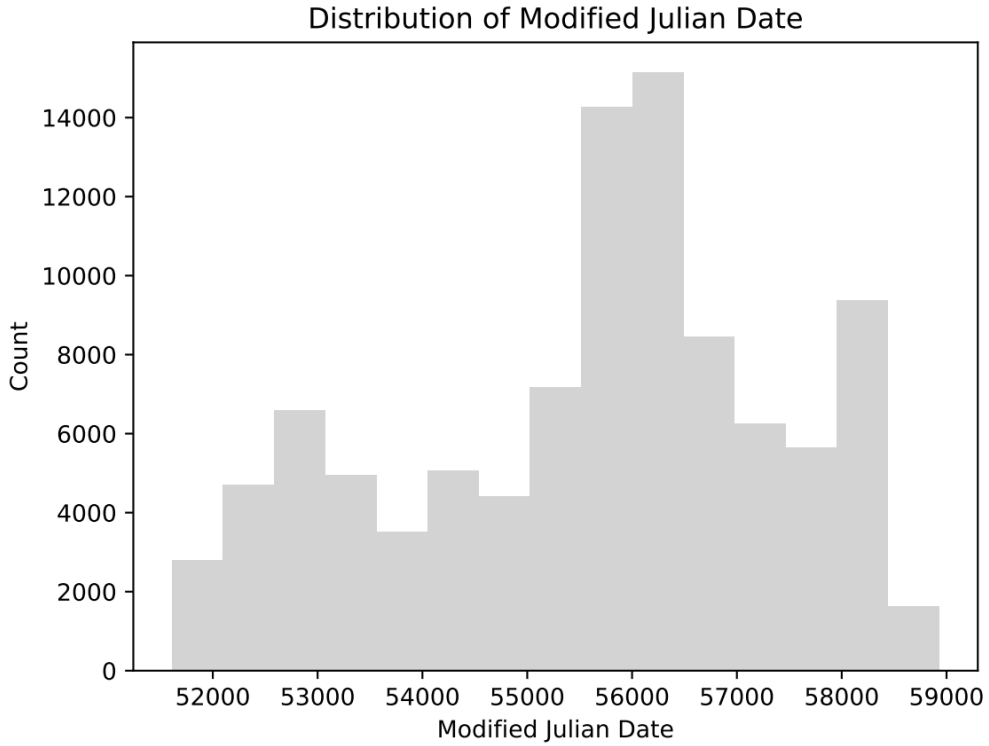
Summary Statistics

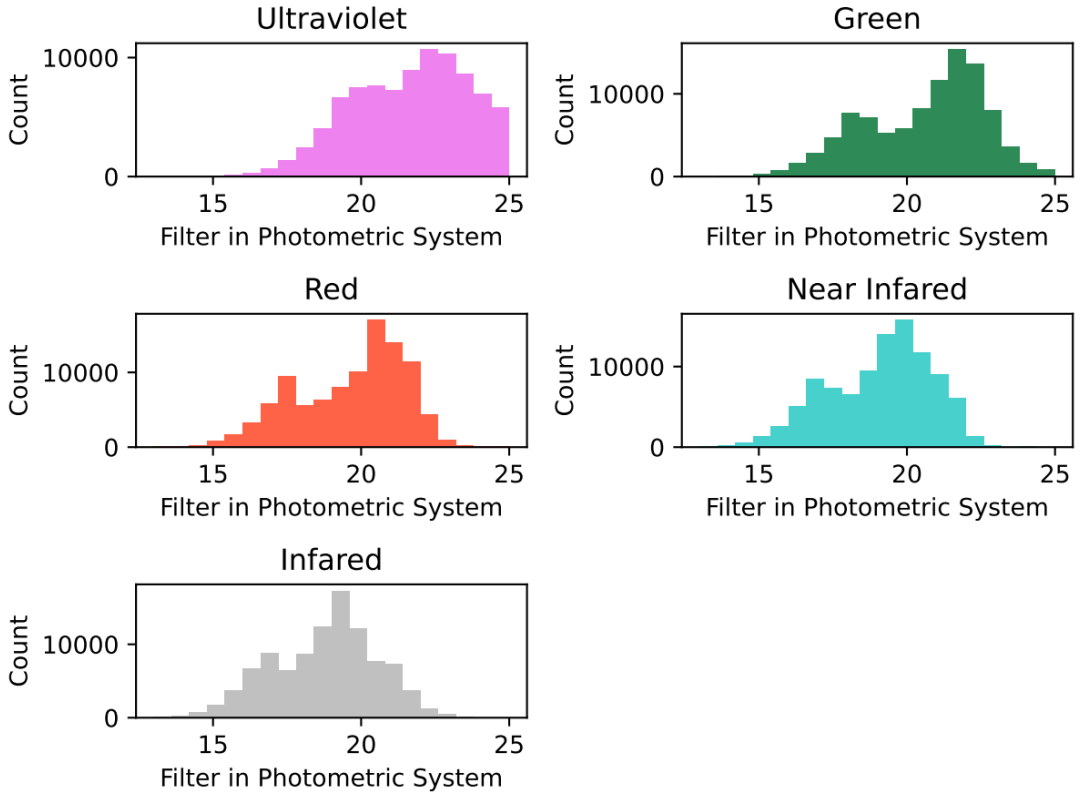
- Mean redshift: 0.5767
- Most common classification: Galaxy





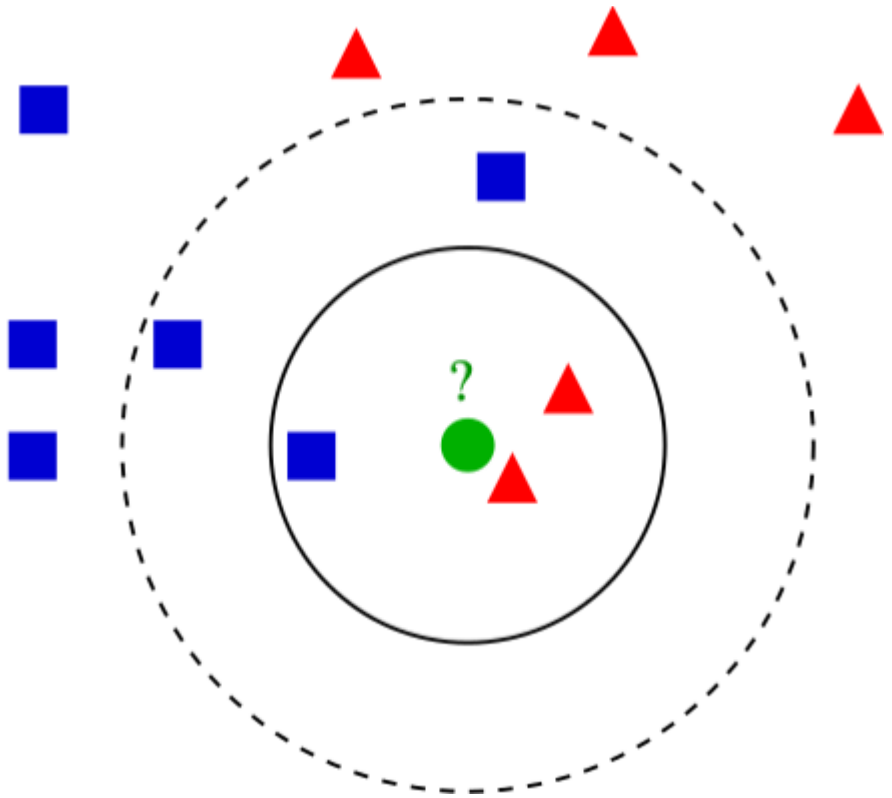
Summary Statistics





K Nearest Neighbors

- Assumes that similar data points are close to each other
- Choose a value for k
- What are the k closest values to the point?



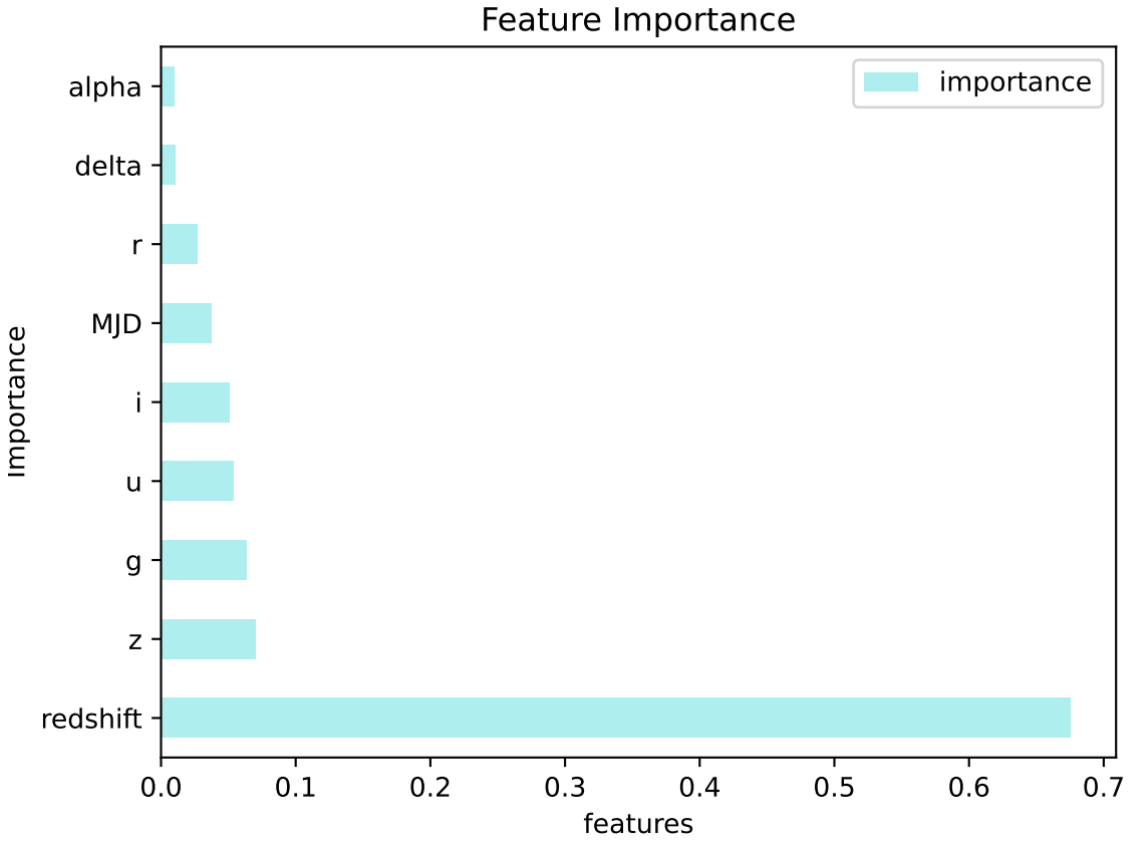
K Nearest Neighbors

- Used normalized data
  - stars[col] = ((stars[col]-stars[col].min()) / (stars[col].max()-stars[col].min()))
  - Changes the values to the same scale
- Parameters
  - parameters: {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'minkowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 5, 'p': 2, 'weights': 'uniform'}
  - parameters: {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'minkowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 10, 'p': 2, 'weights': 'uniform'}
- Accuracy
  - Five neighbors: 0.93215
  - Ten neighbors: 0.9233

Random Forest

- Parameters
  - {'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'max_features': 'auto', 'max_leaf_nodes': None, 'max_samples': None, 'min_impurity_decrease': 0.0,'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0,'n_estimators': 100, 'n_jobs': None, 'oob_score': False, 'random_state': None, 'verbose': 0, 'warm_start': False}
- Accuracy: 0.9777
  - Highest out of all the models used

Random Forest



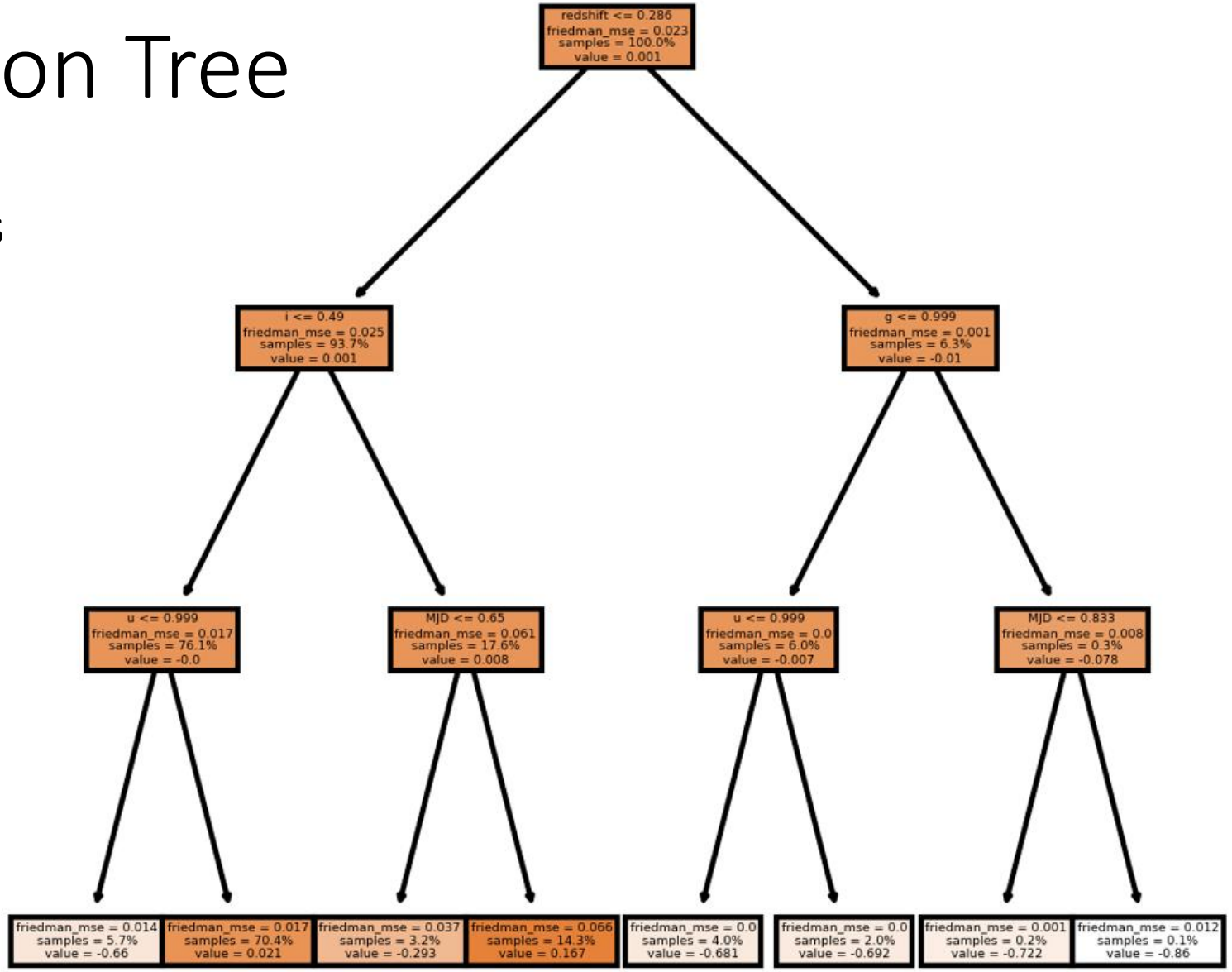

```
44  # find the feature importance
45  feature_importances = pd.DataFrame(randomForest.feature_importances_,index = X_train.columns, \
46  columns=['importance']).sort_values('importance',ascending=False)
47
48  fig, ax = plt.subplots()
49  feature_importances.plot(kind = 'barh', color = 'paleturquoise')
50  plt.title("Feature Importance")
51  plt.xlabel("features")
52  plt.ylabel("Importance")
53  plt.savefig("output/featureImportance.pdf")
54  plt.show()
55  plt.close()
```

Boosted Decision Tree

- A combination of multiple decision trees that attempt to reduce the error of the tree before it
- Parameters
  - parameters: {'ccp_alpha': 0.0, 'criterion': 'friedman_mse', 'init': None, learning_rate': 1.0, 'loss': 'deviance', 'max_depth': 3, 'max_features': None,'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_iter_no_change': None, 'random_state': 0, 'subsample': 1.0, 'tol': 0.0001, 'validation_fraction': 0.1, 'verbose': 0, 'warm_start': False}
- Accuracy: 0.971

Boosted Decision Tree

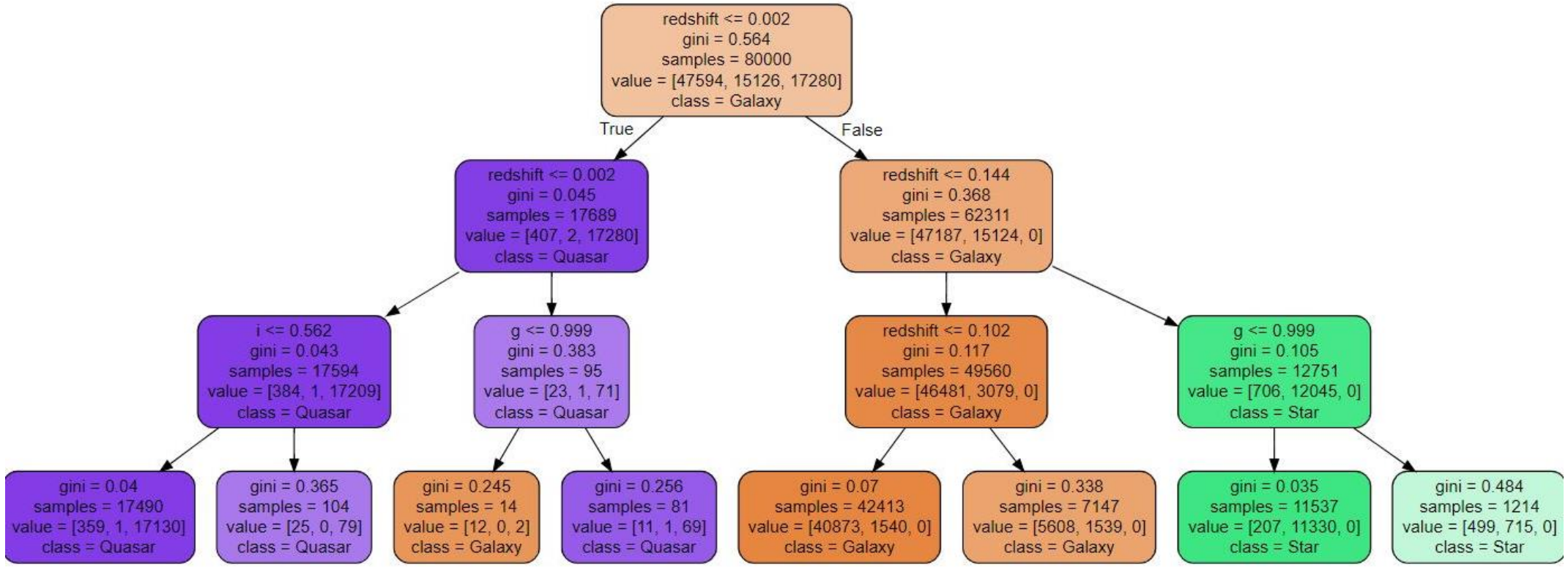
- This is one of the trees that was created



Decision Tree

- Parameters
  - parameters: {'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini', 'max_depth': 3, 'max_features': None, 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'random_state': None, 'splitter': 'best’}
- Accuracy: 0.9465
- Created a dot file and used the website below to create a visualization of the decision tree
  - https://dreampuf.github.io/GraphvizOnline

Decision Tree



Conclusion

- The random forest model had the highest accuracy followed by the boosted decision tree and the k nearest neighbors model
- The k nearest neighbor model that used a k value of 5 was slightly more accurate than the one that used a k value of 10
- The most important feature was redshift

Citations

fedesoriano. (January 2022). Stellar Classification Dataset - SDSS17. Retrieved [04/17/2022] from https://www.kaggle.com/fedesoriano/stellar-classification-dataset-sdss17.

“Stellar Classification.” *Wikipedia*, Wikimedia Foundation, 28 Apr. 2022, https://en.wikipedia.org/wiki/Stellar_classification#:~:text=In%20astronomy%2C%20stellar%20classification%20is,colors%20interspersed%20with%20spectral%20lines.

“K-Nearest Neighbors Algorithm.” *Wikipedia*, Wikimedia Foundation, 15 Mar. 2022, https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.

Neural Network

- The least accurate model attempted
- Neural Network One
  - hyperparameters: {'name': 'Adam', 'learning_rate': 0.001, 'decay': 0.0, 'beta_1': 0.9, 'beta_2': 0.999, 'epsilon': 1e-07, 'amsgrad': False}
  - Final test set loss: nan
  - Final test set accuracy: 0.592550

```
38  neuralNet = Sequential()
39  neuralNet.add(Dense(7, input_shape = (9, ), activation = 'relu'))
40  neuralNet.add(Dense(7, activation = 'relu'))
41  neuralNet.add(Dense(4, activation = 'relu'))
42  neuralNet.add(Dense(1, activation = 'relu'))
```

Neural Network

- Neural Network Two
  - hyperparameters: {'name': 'Adam', 'learning_rate': 0.001, 'decay': 0.0, 'beta_1': 0.9, 'beta_2': 0.999, 'epsilon': 1e-07, 'amsgrad': False}
  - Final test set loss: 0.000000
  - Final test set accuracy: 0.592550

```
90   neuralNet = Sequential()
91   neuralNet.add(Dense(7, input_shape = (9, ), activation = 'relu'))
92   neuralNet.add(Dense(7, activation = 'relu'))
93   neuralNet.add(Dense(4, activation = 'relu'))
94   neuralNet.add(Dense(2, activation = 'relu'))
95   neuralNet.add(Dense(1, activation = 'relu'))
```

- There was no significant difference between the two neural networks