Capstone 1 - Machine Learning Results

Burns, Echelle
Jan 2020

Methods

All categorical variables (Zone, Month, Year, Lunar Phase) underwent one-hot encoding via pandas get_dummies() and data were split into random training and testing data prior to being analyzed by machine learning algorithms. Decision Tree, Gradient Boosting, Stochastic Gradient Descent, K Nearest Neighbors, Naive Bayes, and Gaussian Process classifiers via sklearn were all attempted to analyze this dataset (*SpringBoard-Capstone1/Machine_ Learning/Machine_Learning)*. Originally, machine learning techniques were run using the jackknifed dataset of values that were generated previously. Data were not balanced prior to initial machine learning runs, which allowed for parameter testing of balanced and unbalanced sample weights. Grid searches and for loops with several potential hyperparameter values were used to optimize the hyperparameters of the models. Whether grid searches or for loops were used depended on the computational time required to conduct a grid search with five folds. Precision and recall estimates were used to identify the best model.

After initial models were run, models were refined using a dataset that was created using a different jackknife technique that only kept rows of data in which receivers were present (receiver density > 0, *SpringBoard-Capstone1/Machine_Learning/Machine_Learning_2 -Refined_Models*). This helped to reduce the probability of false negatives within the dataset. Even after the new jackknifing technique was implemented, there were nearly twice as many rows without sharks present compared to rows with sharks present. Therefore, instances in which sharks were present were sampled with replacement until there was an equal number of rows with sharks present and sharks absent. This technique was chosen because the entire dataset had less than 10,000 rows. Decision Trees, Gradient Boosting, and K Nearest Neighbor Classifiers were used to analyze this dataset.

A Decision Tree was also used to determine the minimum required sample size for optimal machine learning results by randomly jackknifing the dataset for $2^8 \ldots 2^n \ldots 2^{14}$ random samples and running the Decision Tree Classifier. Model output precision values were calculated using cross_val_score from sklearn with a total of three folds to ensure that each category (sharks vs no sharks) had at least 5 samples in each model run.

Results

Preliminary In-Depth Analysis Using Machine Learning

See: SpringBoard-Capstone1/Machine_Learning/Machine_Learning

Incorporating class weights as ‘balanced’ or ‘None’ in a Decision Tree Classifier for a range of max depth parameters, produced two very different trends in accuracy plots. The non-balanced weight accuracy values appeared to plateau at max depth values around 8, while the balanced weight accuracy values appeared to show a consistent, positive linear relationship as max depth values increased. Using a grid search showed that the best model using a

Decision Tree Classifier is one without class weights and with a maximum depth of 10. This model led to precision and recall values of 0.996 and 0.985 for the testing dataset, respectively.

Precision, recall and accuracy values that were calculated for a Gradient Boosting Classifier with different levels of learning rates indicated higher scores at learning rates less than 0.5, and much lower scores at learning rates larger than 0.5. However, all scores were greater than 0.90. Using a grid search that incorporated a variety of values for learning rate, max depth and number of estimators indicated that the best model using a Gradient Boosting Classifier had a learning rate of 0.075, a max depth of 3 and a number of estimators equal to 100.

A grid search was attempted for a Stochastic Gradient Descent Classifier for several parameters. However, due to the large size of the dataset (once one-hot encoding took place) and the large lists of hyperparameters that were possible, the code took more than 12 hours to run and was, therefore, terminated. However, running a Stochastic Gradient Descent Classifier with the default hyperparameters produced a model with 0.99 precision and 0.99 recall for the testing dataset.

Models were assessed using K Nearest Neighbor Classifiers at a range of possible number of neighbor values from 1 to 30 and using uniform and distance weighting. Precision and recall values for uniformly weighted data points found that a value of 10 nearest neighbors produced sufficiently high recall and accuracy (~0.991) for the testing dataset, while the precision and recall values for distance weighted points found a value of around 9 nearest neighbors produced a sufficiently high recall around 0.9925 and accuracy around 0.9945 for the testing dataset. A grid search was attempted using a variety of more neighbor estimates, but the code was running for five hours before it was terminated.

Gaussian Naive Bayes Classifier precision and recall values were calculated using the default hyperparameters of the classifier and produce a precision of 0.9999 and a recall of 0.9337 for the testing dataset.

The Gaussian Process Classifier was also run using default hyperparameters, but the dataset appeared to be too large for the classifier to handle, even after taking precautions suggested by other model users, such as adding ‘copy_X_train = False’ into the model before running. To combat this, the data were split into groups of 5000 points and sequentially fitted to the classifier. This model resulted in a precision value of 0.95 and a recall value of 0.998 for the testing dataset.

Refined Machine Learning Classification

See: SpringBoard-Capstone1/Machine_Learning/Machine_Learning_2-Refined_Models

Upon further inspection of the data, it became clear that the model was performing better than expected, perhaps because there were data that included regions in which no receivers were present. Because of this, it appeared that all models were automatically assigning 0s to rows that had no receiver present. Therefore, data were re-jackknifed in order to include only data in which receivers were present.

Using these data, the Decision Tree Classifier grid search was run on the full dataset using a range of max depths between 3 and 10, and results found that the best model had a hyperparameter of max depth equal to 10. However, upon plotting the precision scores for this model, it appeared that all models showed precision values that ranged between approximately

0.65 and 0.85, which was much lower than the models using the previous dataset. By looking at the plotted precision values, it appeared as though there was a large difference in precision between maximum depths of 4 and 5, but that the spacing between precision became slightly smaller at higher max depths. In order to keep a model that had a decent precision (0.72) and was relatively easy to understand, a decision tree was built using a max depth of 6. When predicting values for the training and testing data using this model, precision values were 0.928 and 0.931, respectively. The most important features in this model appeared to be depth gradient, the year 2019, particular zones, temperature, and receiver densities of 2.

A similar approach was conducted for the Gradient Boosting Classifier. A grid search used a variety of possible values for learning rate and max depth and the best model was suggested to have a learning rate of 0.75, a max depth of 8, and 100 estimators. However, upon plotting the precision scores, it was found that all scores ranged between 0.65 and 0.99 and most scores were greater than 0.80. Therefore, a model that showed about 0.85 precision was chosen to predict the testing dataset. This model used a learning rate of 0.075, a max depth of 3, and 100 estimates. The precision for the training and testing data for this model were 0.904 and 0.903, respectively. The most important features in this model included Depth Gradient, the year 2019, 2015 and 2018, a receiver density of 2, temperature, and particular zones.

Although these results are beneficial when receivers are in the water, the model should also perform well when receivers are not present. Therefore, another Gradient Boosting Classifier was run without the receiver density predictor. Results from a grid search indicated the best model should have the hyperparameters of a learning rate of 0.75, a max depth of 8, and 100 estimators. However, a similar trend was seen in which most model runs were between 0.65 and 0.99 accuracy. The model with a predicted precision score of about 0.85 was again chosen (learning rate = 0.75, max depth = 3, number of estimators = 100). This model produced precision values of 0.916 when predicting the training data and 0.9156 when predicting the testing data. The most important features here included depth gradient, years 2019, 2015 and 2019, temperature, and particular zones.

A K Nearest Neighbor Classifier was also run on the dataset for which the receiver density factor was removed. A grid search for this model indicated that the best model would have one nearest neighbor with uniform weights. Using this model to predict the testing dataset yielded a precision of 0.999.

Finally, an optimal sample size was calculated by running 100 iterations for each potential sample size and precision estimates for each model were predicted using cross validation with three folds. All values were averaged for each sample size range and results were plotted for precision on the training and testing datasets. From these plots, it appears that the precision values begin to plateau around a precision of 0.70 at approximately 4000 samples.