**Decision Tree**

1. Gini Index: <https://blog.quantinsti.com/gini-index/>
2. Entropy and information gain: <https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8>

<https://www.saedsayad.com/decision_tree.htm>

1. **Summary**: The Gini Index is calculated by subtracting the sum of the squared probabilities of each class from one. It favors larger partitions. Information Gain multiplies the probability of the class times the log (base=2) of that class probability. Information Gain favors smaller partitions with many distinct values. Ultimately, you have to experiment with your data and the splitting criterion.

|  |  |  |
| --- | --- | --- |
| **Algo / Split Criterion** | **Description** | **Tree Type** |
| Gini Split / Gini Index | Favors larger partitions. Very simple to implement. | CART |
| Information Gain / Entropy | Favors partitions that have small counts but many distinct values. | ID3 / C4.5 |

1. **GridSearch CV and Hyperparameters Optimization - In order to choose the perfect parameters for the model to get more accurate results:** Grid search is an approach to parameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid. Use to choose the correct set of hyperparameters for the best accurate result. **- see random forest Jupiter notebook**

# Hyperparameters Optimization

Utilizing the GridSearchCV functionality, let's create a dictionary with parameters we are looking to optimize to create the best model for our data. Setting the n\_jobs to 3 tells the grid search to run three jobs in parallel, reducing the time the function will take to compute the best parameters. I included the timer to see how long different jobs took; that led me to ultimately decide to use three parallel jobs.

This will help set the parameters we will use to tune one final parameter: the number of trees in our forest.

**# Algorithm tuning**

**np.random.seed(42)**

**start = time.time()**

**param\_dist = {'max\_depth': [2, 3, 4],**

**'bootstrap': [True, False],**

**'max\_features': ['auto', 'sqrt', 'log2', None],**

**'criterion': ['gini', 'entropy']}**

**cv\_rf = GridSearchCV(fit\_rf, cv = 10,**

**param\_grid=param\_dist,**

**n\_jobs = 3)**

**cv\_rf.fit(training\_set, class\_set)**

**print('Best Parameters using grid search: \n',**

**cv\_rf.best\_params\_)**

**end = time.time()**

**print('Time taken in grid search: {0: .2f}'.format(end - start))**

1. **K-Fold Cross Validation?**

* [**https://scikit-learn.org/stable/modules/cross\_validation.html**](https://scikit-learn.org/stable/modules/cross_validation.html)
* [**https://stackoverflow.com/questions/38151615/specific-cross-validation-with-random-forest**](https://stackoverflow.com/questions/38151615/specific-cross-validation-with-random-forest)

1. **Out of Bag Error Rate?** To determine the number of trees (n\_estimators) for better accuracy. We generate graphs to see for which value of “n\_estimator” we are getting less OOB error rate. **- see random forest Jupiter notebook**

**min\_estimators = 15**

**max\_estimators = 1000**

**error\_rate = {}**

**for i in range(min\_estimators, max\_estimators + 1):**

**fit\_rf.set\_params(n\_estimators=i)**

**fit\_rf.fit(training\_set, class\_set)**

**oob\_error = 1 - fit\_rf.oob\_score\_**

**error\_rate[i] = oob\_error**

**# Convert dictionary to a pandas series for easy plotting**

**oob\_series = pd.Series(error\_rate)**

**fig, ax = plt.subplots(figsize=(10, 10))**

**ax.set\_facecolor('#fafafa')**

**oob\_series.plot(kind='line',**

**color = 'red')**

**plt.axhline(0.055,**

**color='#875FDB',**

**linestyle='--')**

**plt.axhline(0.05,**

**color='#875FDB',**

**linestyle='--')**

**plt.xlabel('n\_estimators')**

**plt.ylabel('OOB Error Rate')**

**plt.title('OOB Error Rate Across various Forest sizes \n(From 15 to 1000 trees)')**

# The OOB error rate starts to oscillate at around 400 trees, so I will go ahead and use my judgment to use 400 trees in my forest.

#Using the pandas series object I can easily find the OOB error rate for the estimator as follows:

**print('OOB Error rate for 400 trees is: {0:.5f}'.format(oob\_series[400]))**

**Jupyter Notebooks for code :**

1. Decision Trees (Titanic dataset) - prepared from Kaggle
2. Decision\_Tree\_ML(classifier)