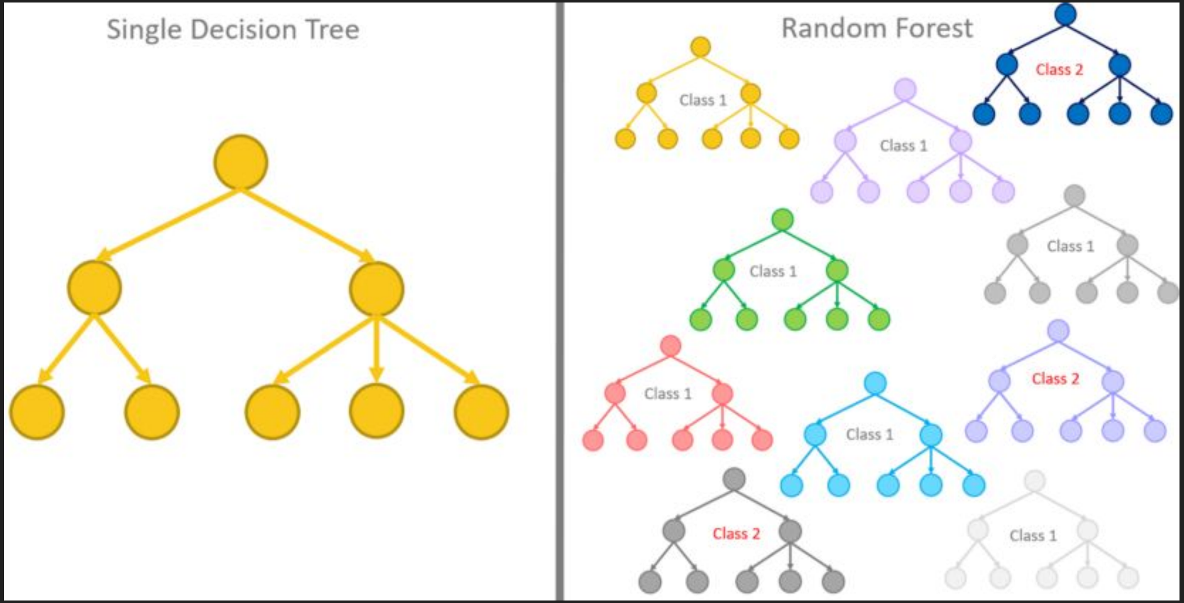
RANDOM FOREST ALGORITHM

Talia
Isabelle

David
Fritz

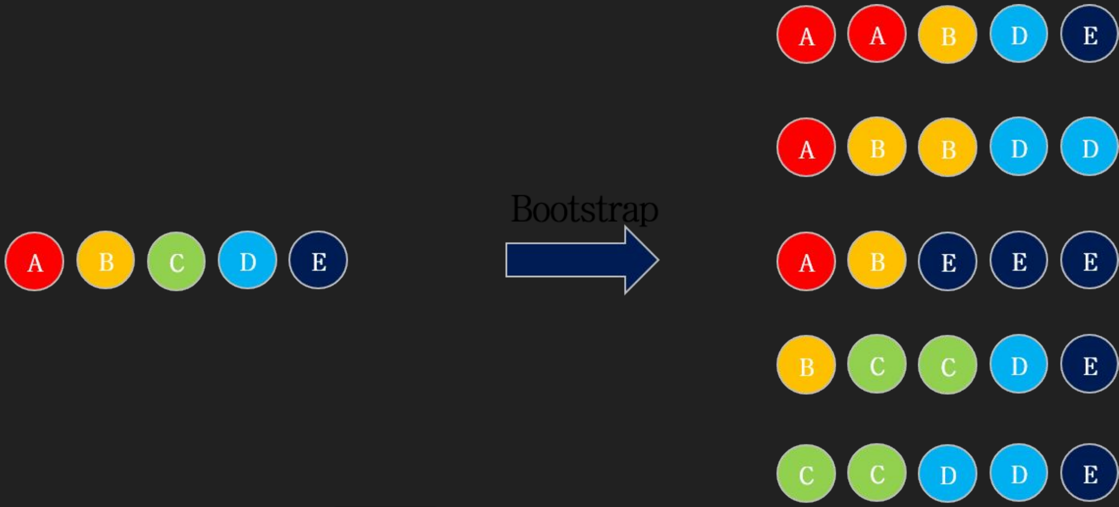
Overview

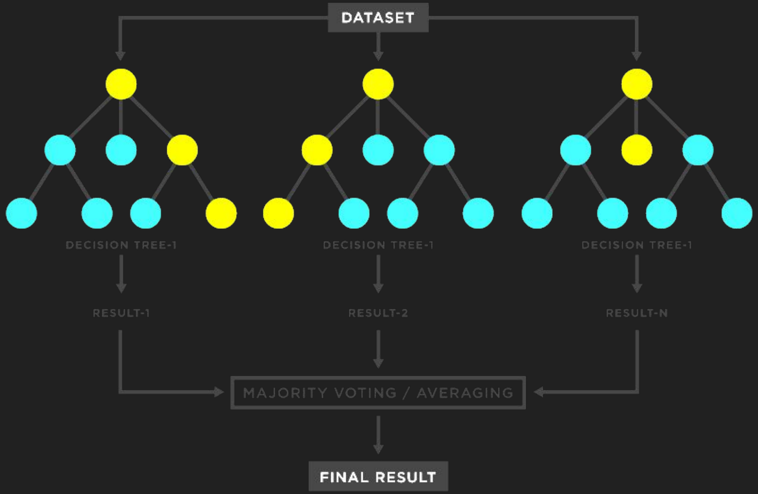


- Random Forest: an ensemble machine learning algorithm that uses bagging to combine the output of multiple decision trees to reach a single result
- Each tree independently makes a prediction, the values are then averaged (Regression) / Max voted (Classification) to arrive at the final value.
- One of the most popular tree-based supervised learning algorithms
  - Flexible and easy to use

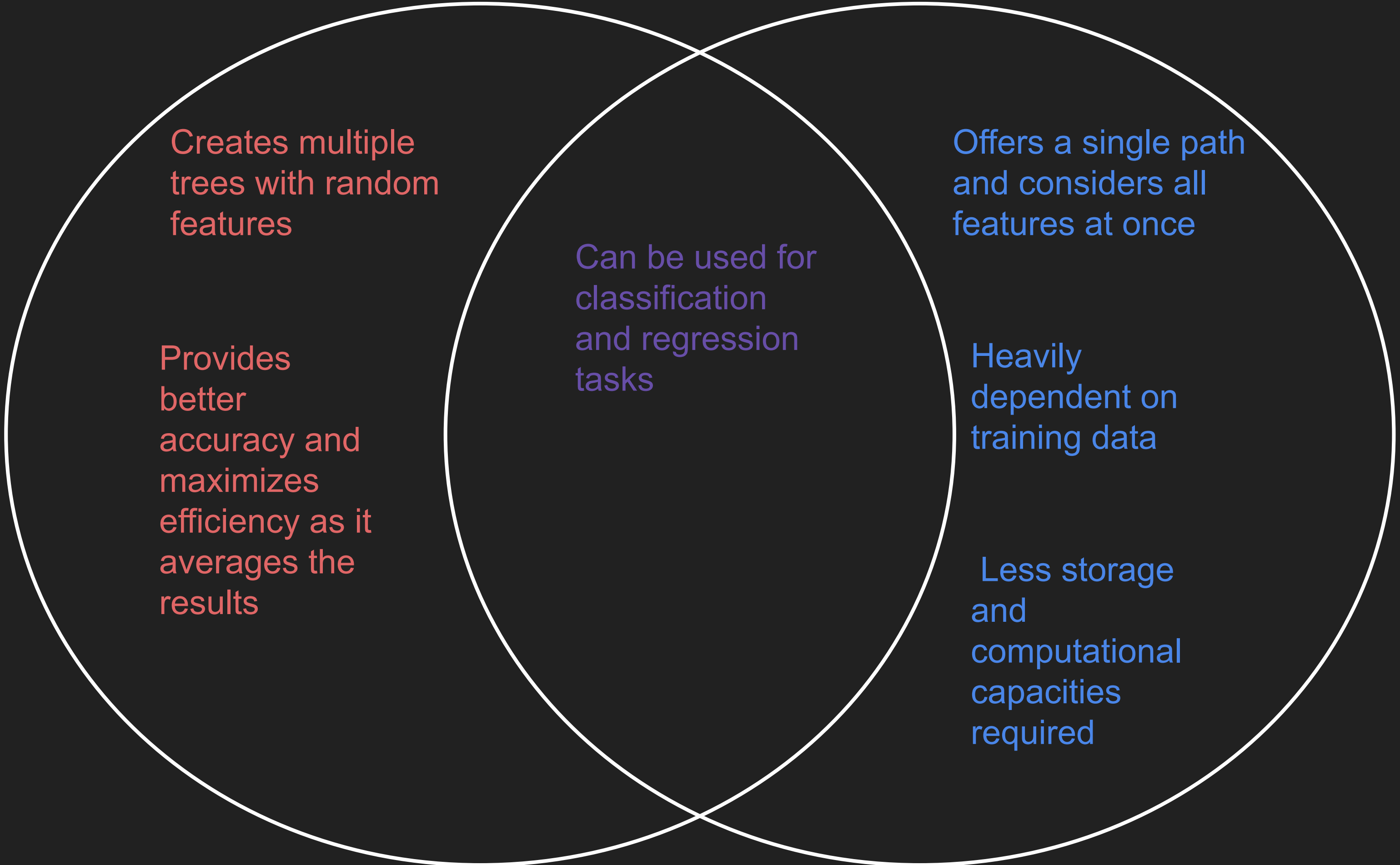
Bagging

- Bootstrapping: creating many datasets from one original by randomly selecting observations with repetition
- Aggregation: combining the predictions of many algorithms
- Bagging = Bootstrapping + Aggregation





Random Forest vs Decision Tree



Creates multiple trees with random features

Provides better accuracy and maximizes efficiency as it averages the results

Can be used for classification and regression tasks

Offers a single path and considers all features at once

Heavily dependent on training data

Less storage and computational capacities required

Random Forest - How Does it Work?

Multiple decision trees are combined to form an ultimate decision maker based on the output from each tree

- The data for each tree is selected using a method called *bootstrapping,* which selects a random set of data points from the dataset for each tree.
- Each tree randomly picks the features based on the subset of data provided and independently executes to provide its decision.
- Once we receive the output from every decision tree we use the majority vote taken to arrive at the decision. To use this as a regression model, we would take an average of the values.

Classifier vs Regressor

- Random forest classifiers work with data having discrete labels, or classes.

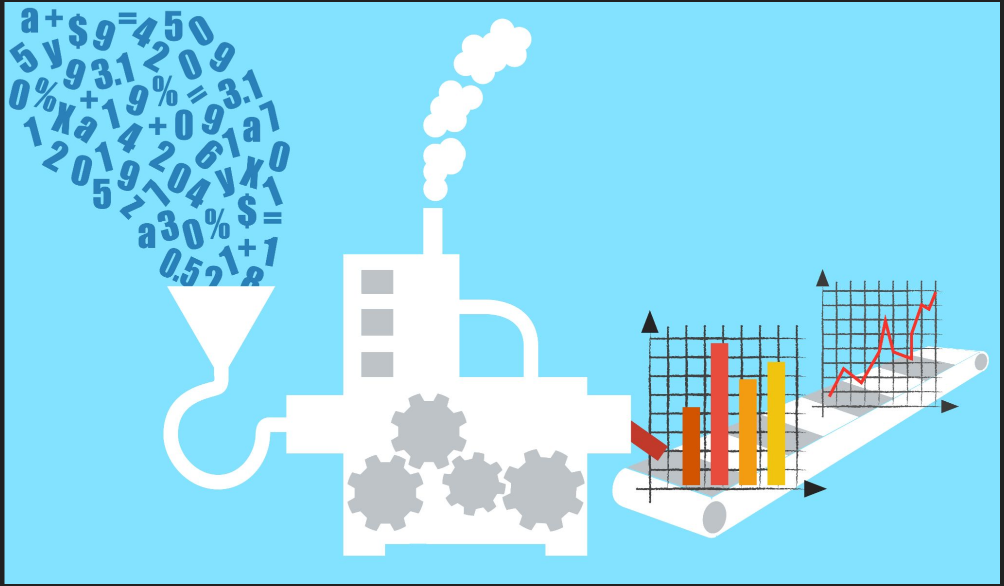
  I.e gender, type of flower, whether or not a player made a basketball team, etc

- Random forest regressors work with data having numeric or continuous output

  I.e the price of sneakers, gross income, etc

Data Processing Steps

- Zero pre-processing steps necessary other than selecting the target variable (separating y from X)
- Do not need train-test-split (though recommended)
  - each tree is only seeing a subset of the training set
- Scaling and regularization can help with accuracy, but not necessary either

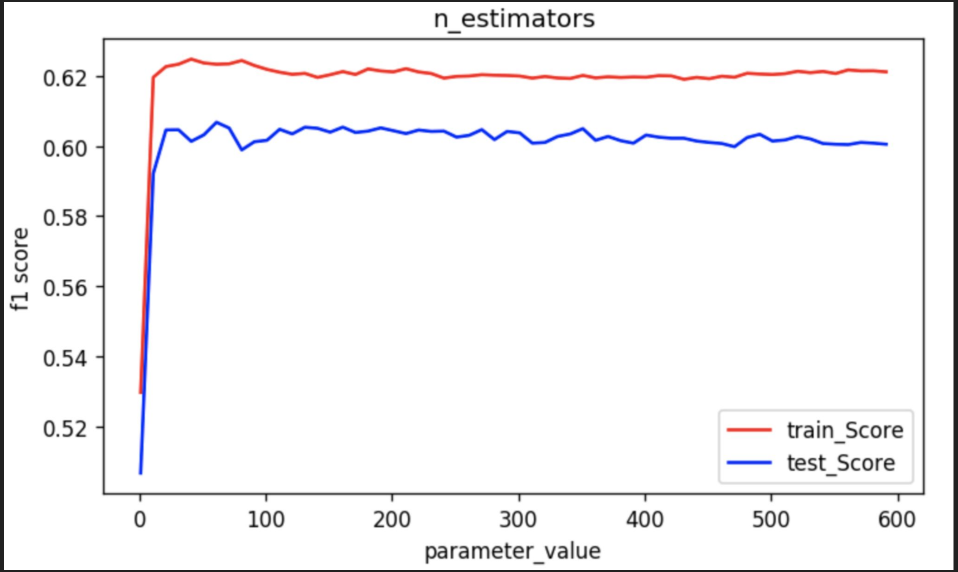


- Works on datasets with numerical and categorical features
- Works on sparse datasets with lots of missing values

Hyperparameters

Random Forests include all of the hyperparameters from their base estimator, decision trees, with some tweaks and additions.

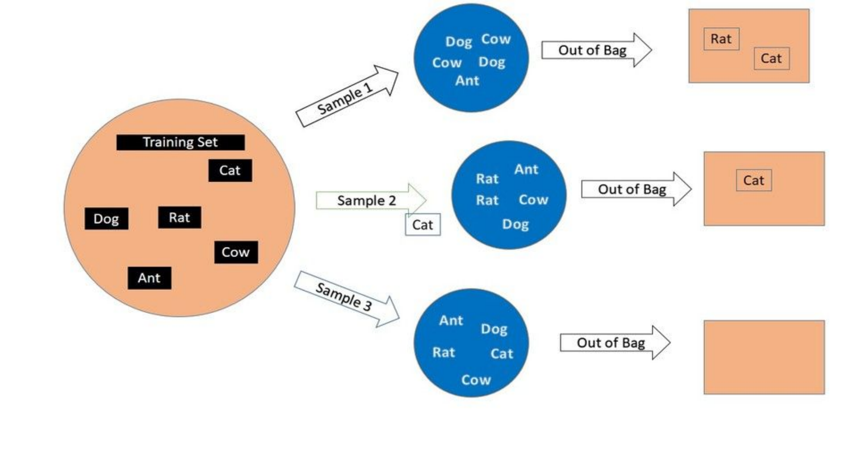
- n_estimators: int, default = 100
  - Number of decision trees in the forest
  - Larger number will hurt time complexity
  - More trees help generalize the model, but usually plateaus after a certain point

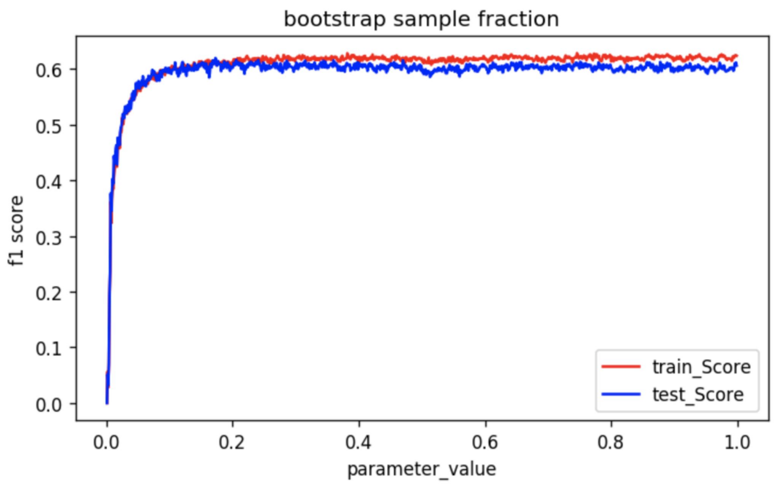


Hyperparameters - Bootstrapping

- bootstrap: bool, default=True
  - Whether to use bootstrapping. If False, uses the whole dataset for each tree instance
- oob_score: bool, default=False
  - Validation technique using out-of-bag (oob) samples.

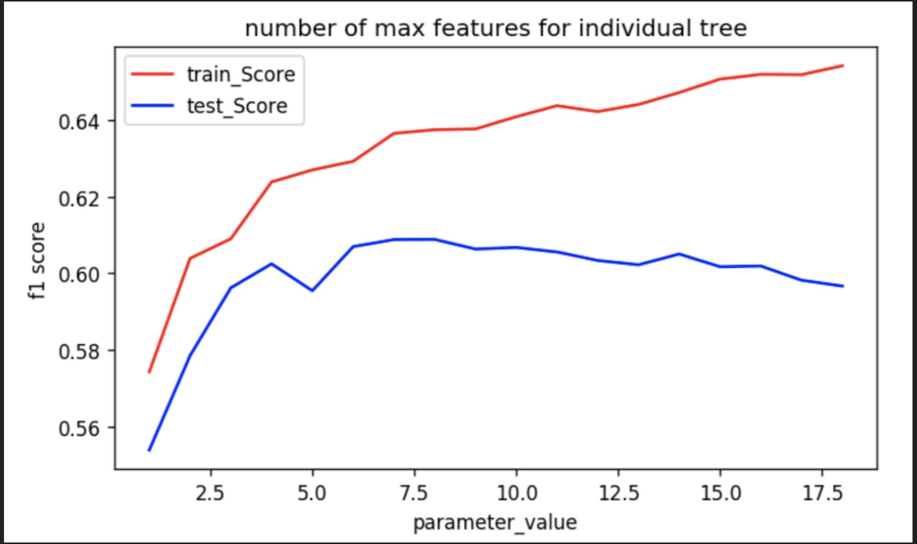
- max_samples: int or float, default=None
  - Number of samples to pull from the original dataset for each bootstrapped set (if float, then this is the fraction of total samples that will be pulled)





Hyperparameters - Other parameters

- max_features: {“sqrt”, “log2”, None}, int or float, default = ”sqrt”
    - Unlike the standard decision tree, this hyperparameter defaults to the square root of the total number of features



- random_state: int or None, default=None
    - Used with bootstrapping and max_features if enabled. Works in the same way as random_state in train_test_split, so that the randomness can be repeated in different tests.

ADVANTAGES

- It reduces overfitting in decision trees and helps to improve the accuracy
- It is flexible to both classification and regression problems
- It works well with both categorical and continuous values

DISADVANTAGES

- It requires much computational power as it builds numerous trees to combine their outputs.
- In massive databases, it requires much time for training as it combines a lot of decision trees to determine the class.

Real World Applications

Banking:

- Credit Card Fraud Detection
- Customer Segmentation

HealthCare and Medicine:

- Disease Prediction

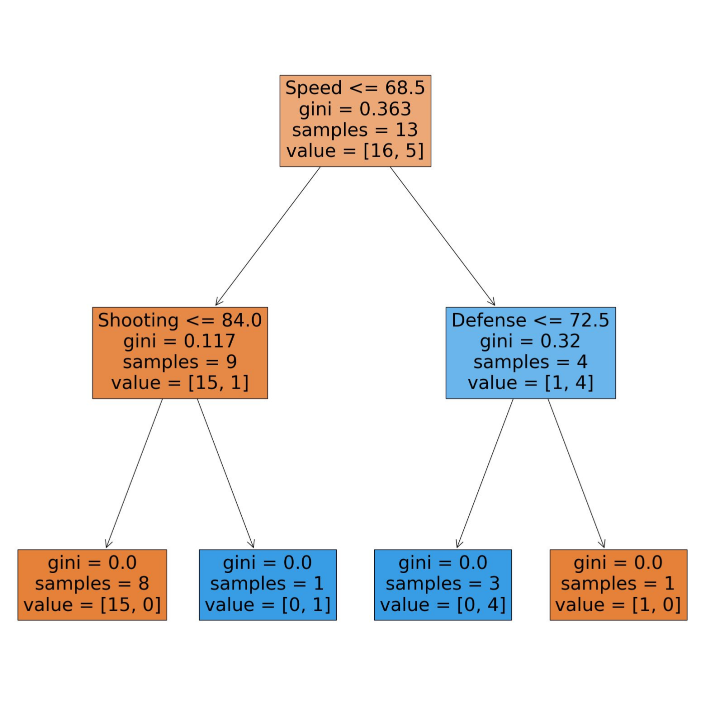
Stock Market:

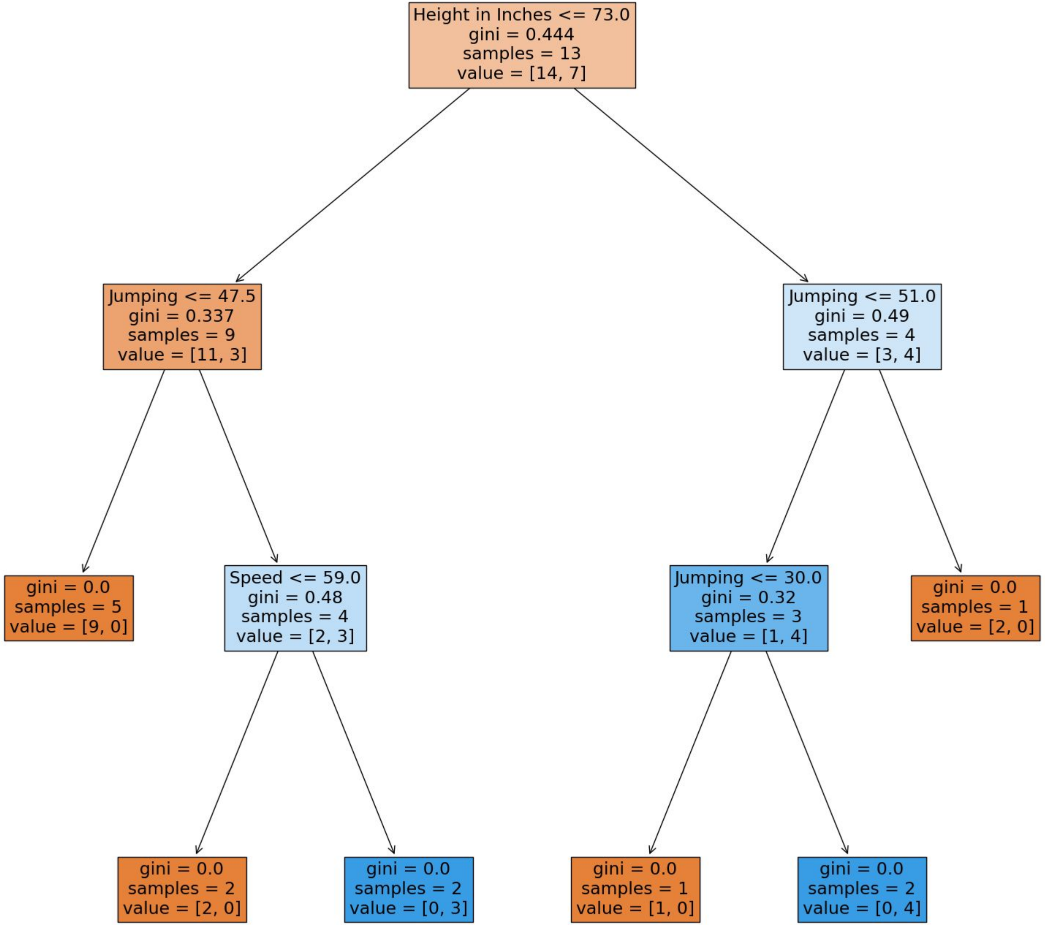
- Stock Market Prediction
- Cryptocurrency Detection

E-Commerce

- Product Recommendation
- Price Optimization

Toy Dataset Demo





Old Dataset Demo

New Dataset Demo

Appendix

Helpful links

- sklearn.ensemble.RandomForestClassifier — scikit-learn 1.2.0 documentation
- Decision Tree Classification Clearly Explained! - YouTube
- Random Forest Algorithm Clearly Explained! - YouTube
- Random Forest Classifier: Overview, How Does it Work, Pros & Cons | upGrad blog
- How to Improve Accuracy of Random Forest ? Tune Classifier In 7 Steps (datasciencelearner.com)
- Random Forest Regression. A basic explanation and use case in 7… | by Nima Beheshti | Towards Data Science
- Guide to Random Forest Classification and Regression Algorithms (serokell.io)
- machine learning - Feature importances in random forest - Cross Validated (stackexchange.com)
- P2: Random Forest tuning | GridSearchCV | Kaggle
- Random Forest Regression: When Does It Fail and Why? - neptune.ai
- Understanding the decision tree structure — scikit-learn 1.2.0 documentation
- Random Forest Hyperparameter Tuning in Python | Machine learning (analyticsvidhya.com)
- Bagging, Random Forest and Out-of-Bag Samples - Just Chillin' (liyanxu.blog)
- python - Why does this decision tree's values at each step not sum to the number of samples? - Stack Overflow
- Out of Bag Score | OOB Score Random Forest Machine Learning (analyticsvidhya.com)

Image References

2020-10-07-random-forest-bootstrap.png (1135×508) (tyami.github.io) - bootstrapping diagram

shutterstock_357106388.jpg (5143×3000) (dataireland.ie) - data processing