Machine Learning Applications with Python

Neda Hantehzadeh

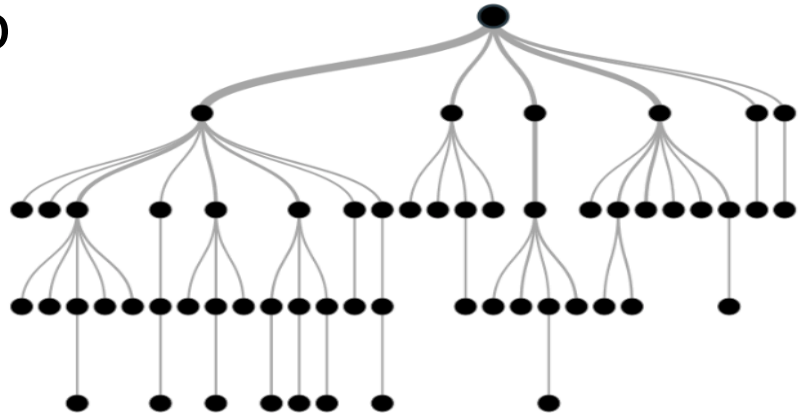
Week7

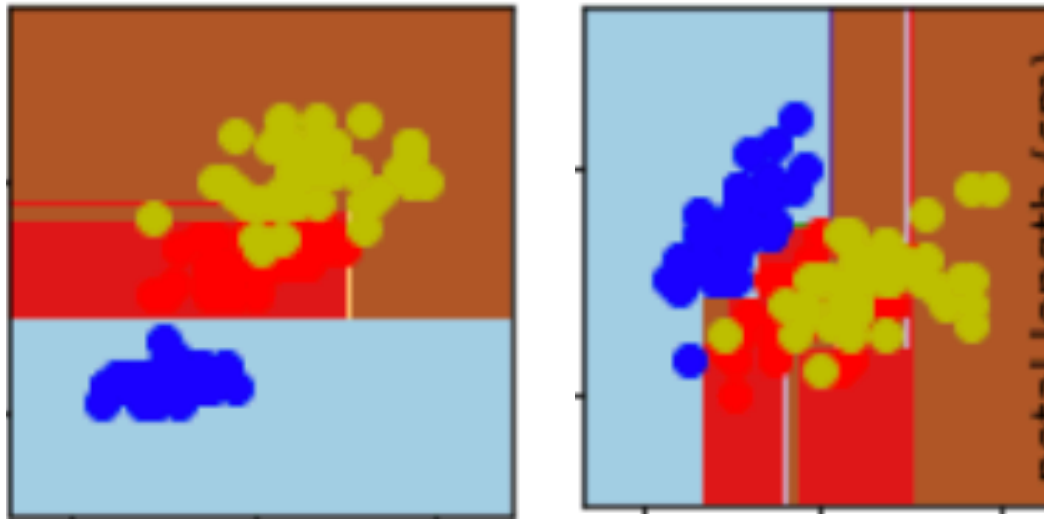
outline

- Decision trees
- Type of optimizing decision tree and improving the accuracies: bagging, random forest and boosting
- Explain each
- Parameters in gradient boosting to tune for
- Comparison of bagging, gradient boosting and random forest
- Ensemble of models
- Python code examples

Decision Trees

- Decision trees are supervised technique that segments or stratifies the variables space into a simple number of regions.
- Tree-based methods can be applied to both classification and regression techniques.





Advantages and Disadvantages of using decision trees

Advantages:

- Simple to understand/interpret and visualize
- Can handle both categorical and numerical variables
- Can handle multioutput problems

Disadvantages:

- Overfitting
- Tend to not be as accurate as other general supervised techniques

Iris data set Example

- Number of classes: 3

-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica

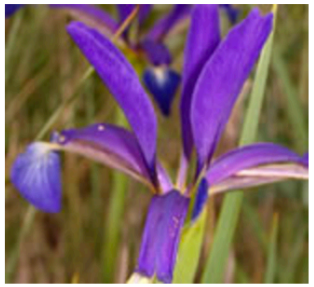
- Attributes: 4
    1. sepal length in cm
    2. sepal width in cm
    3. petal length in cm
    4. petal width in cm

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Iris Data Set

Download: Data Folder, Data Set Description

Abstract: Famous database; from Fisher, 1936



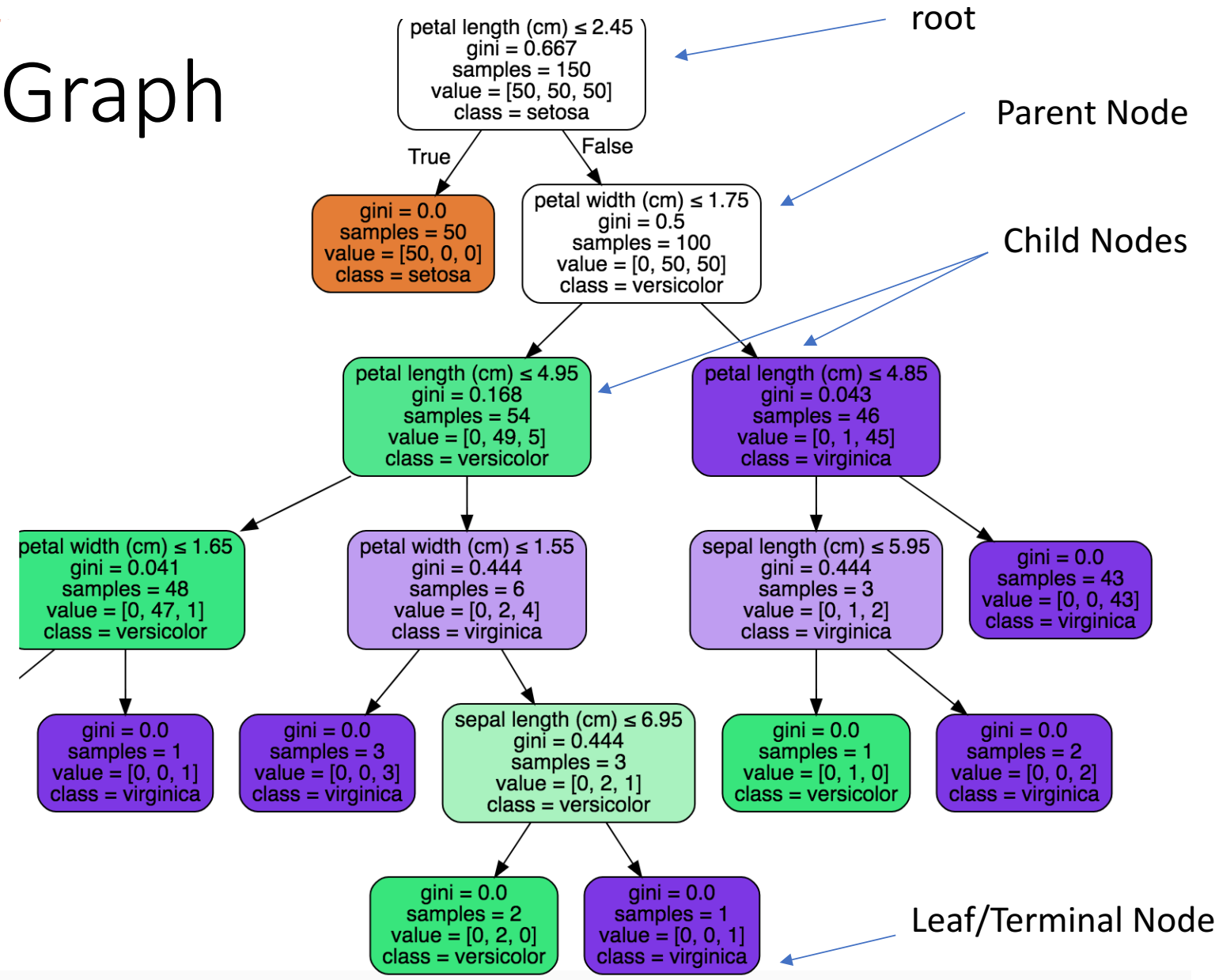
Data Set Characteristics:	Multivariate	**Number of Instances:**	150	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	4	**Date Donated**	1988-07-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	1476008

https://archive.ics.uci.edu/ml/datasets/iris

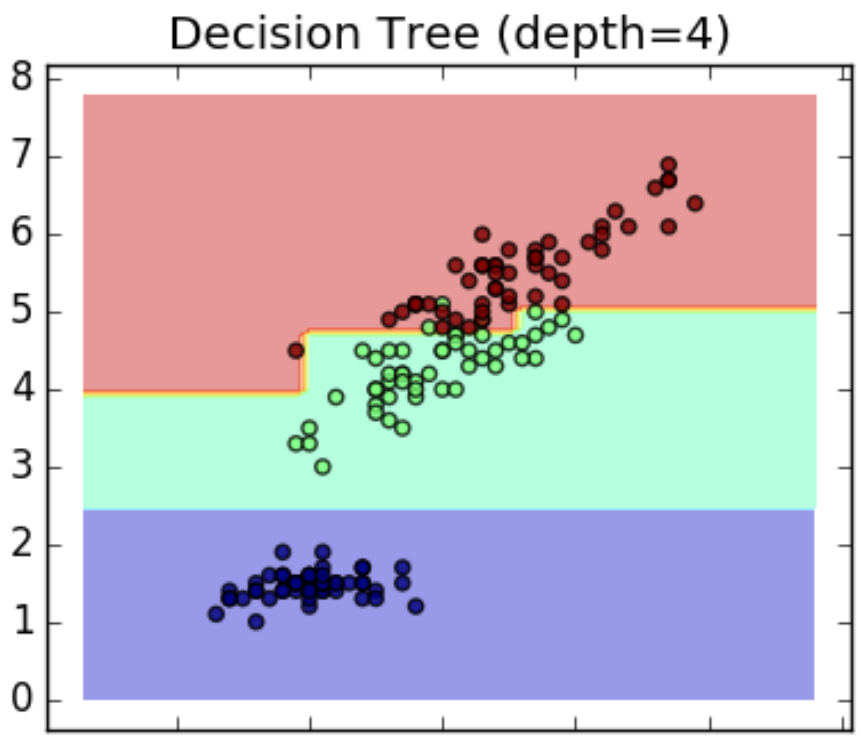
Decision Tree Classification Example

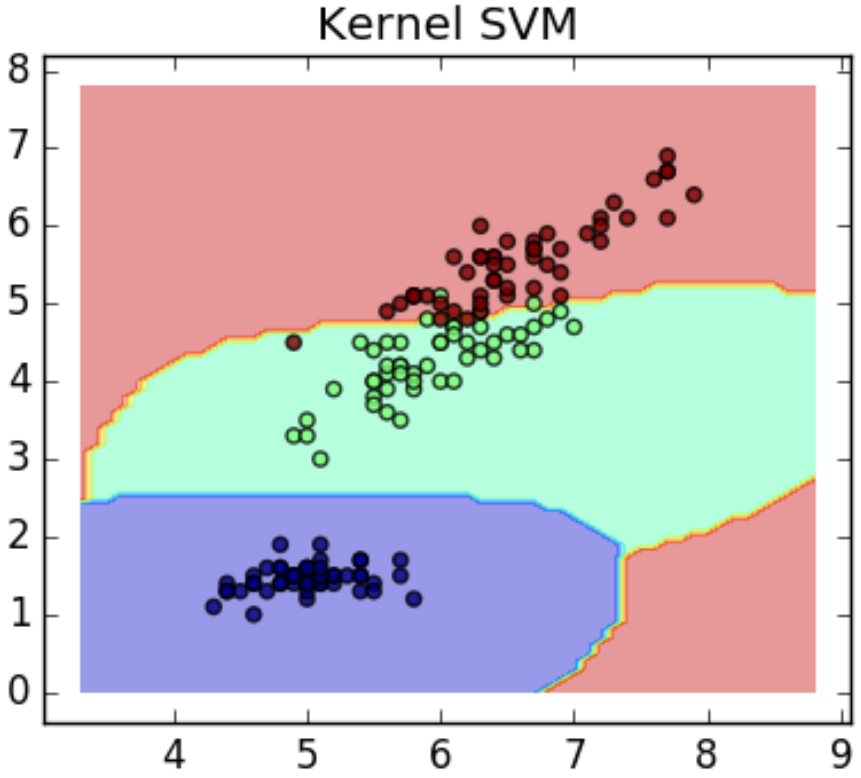
```
from sklearn.datasets import load_iris
from sklearn import tree
iris = load_iris()
clf = tree.DecisionTreeClassifier()
clf = clf.fit(iris.data, iris.target)
```

Decision Tree Graph



Region segmentations DT vs SVM

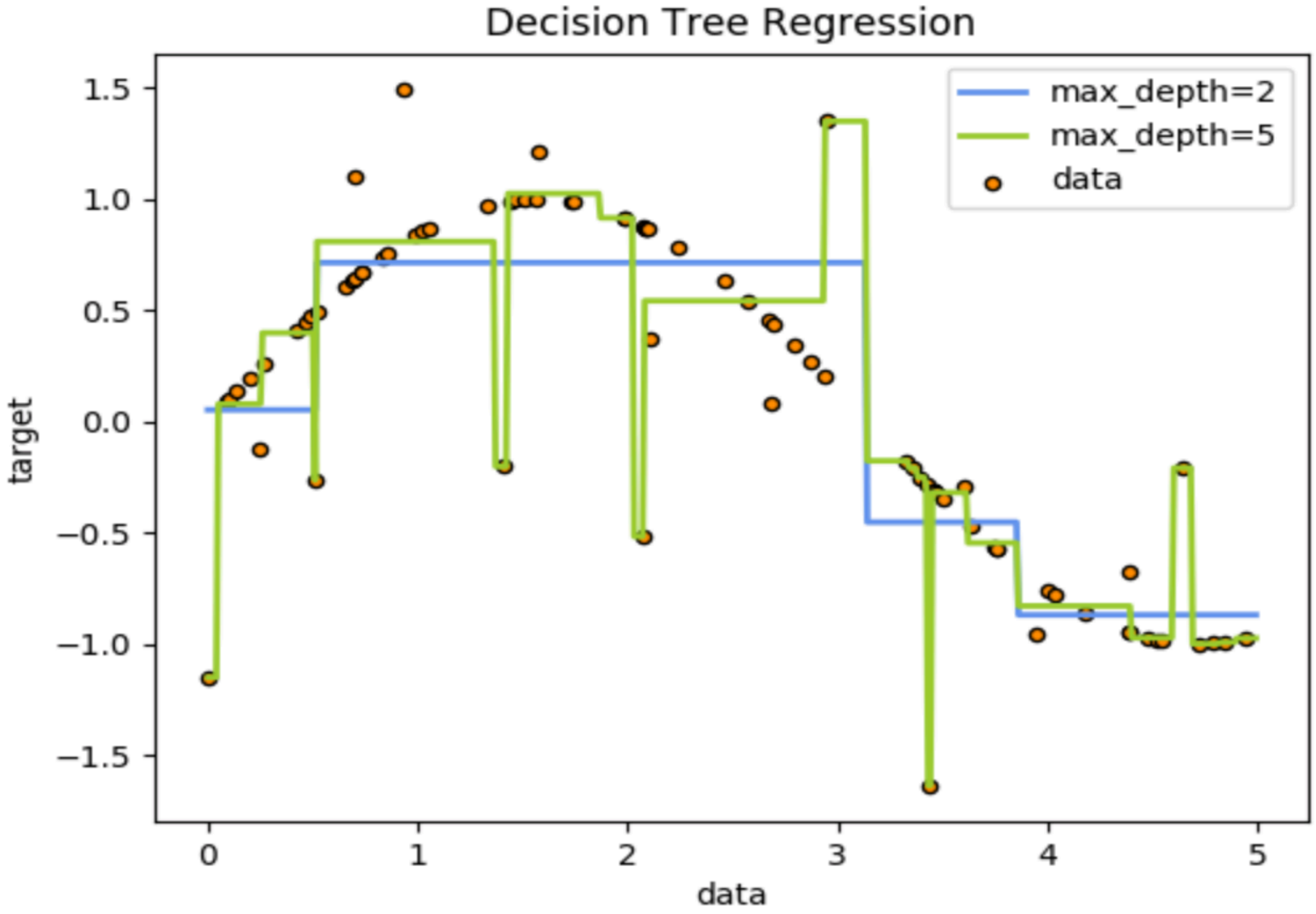




Classification Trees versus Regression Trees

1. In case of regression tree, prediction is made by calculating the mean response of observation in that region
2. In case of classification tree, the value (class) obtained by calculating the mode response of observations in that region
3. Both the trees follow a top-down greedy approach known as recursive binary splitting. This splitting process is continued until a user defined stopping criteria is reached. For example: we can tell the the algorithm to stop once the number of observations per node becomes less than 50.
4. In both the cases, the splitting process results in fully grown trees until the stopping criteria is reached.

Decision Tree Regression Example



Some examples of Decision Trees

- ID3
- C4.5
- C5
- CART (Latest of all, available for both regression and classification)

How does a tree make a decision on split?

The decision criteria is different for classification and regression trees... the nodes are split on all available variables and then the split which results in most homogeneous sub-nodes is selected.

Classification tree: Gini, Cross-Entropy and Misclassification

Regression tree: Mean Squared Error and Mean Absolute Error

Gini Index

- The impurity (or purity) measure used in building decision tree in CART is Gini Index. The decision tree built by CART algorithm is always a binary decision tree (each node will have only two child nodes).It works with categorical target variable “Success” or “Failure”.
- It performs only Binary splits
- lower the value of Gini higher the purity of a node
- CART (Classification and Regression Tree) uses Gini method to create binary splits.

Split Criteria for Classification

Probability of target k at node m

Gini $$H(X_m) = \sum_k p_{mk}(1 - p_{mk})$$

Cross-Entropy $$H(X_m) = -\sum_k p_{mk} \log(p_{mk})$$

Misclassification $$H(X_m) = 1 - \max(p_{mk})$$

Split Criteria for Regression

Mean Square Error

$$c_m = \frac{1}{N_m} \sum_{i \in N_m} y_i$$

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} (y_i - c_m)^2$$

Mean Absolute Error

$$\bar{y_m} = \frac{1}{N_m} \sum_{i \in N_m} y_i$$

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} |y_i - \bar{y_m}|$$

Important Tips on Using Decision Trees

- Decision trees tend to overfit on data with a large number of features. Solution: dimensionality reduction (PCA, ICA, or Feature selection)

- Find the best maximum depth of the tree to control the size of the tree to prevent overfitting. Solution: Use max_depth parameter and Cross Validation to find the best value, rule of thumb is max_depth=3

Important Tips on Using Decision Trees

- Setting the incorrect ‘Minimum samples for a node’ and ‘minimum samples per leaf’ split can lead to overfitting. Solution: Use cross validation to find the optimum numbers.

- Balance your dataset before training to prevent the tree from being biased toward the classes that are dominant. Solution: Class balancing can be done by sampling an equal number of samples from each class, or preferably by normalizing the sum of the sample weights (sample_weight in Skitlearn. DecisionTreeClassifier) for each class to the same value.

Important Tips on Using Decision Trees

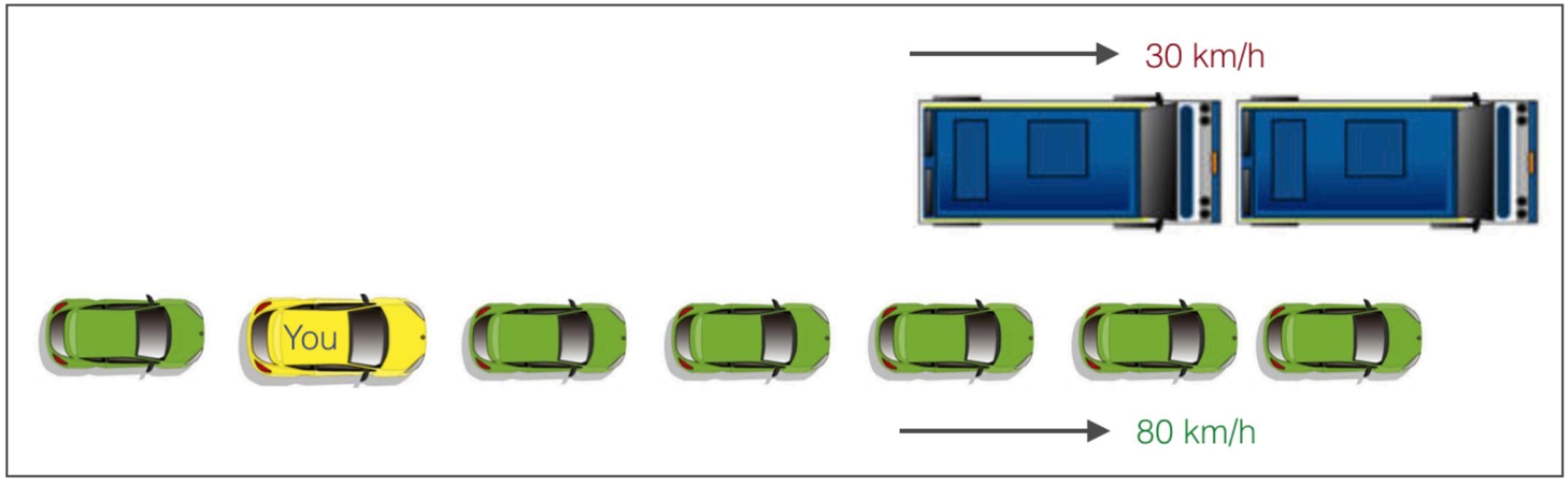
- All decision trees use np.float32 arrays internally. If training data is not in this format, a copy of the dataset will be made.

- If the input matrix X is very sparse, it is recommended to convert to sparse ’csc_matrix’ before calling fit and sparse ’csr_matrix’ before calling predict. Training time can be orders of magnitude faster for a sparse matrix input compared to a dense matrix when features have zero values in most of the samples.

How to avoid overfitting?

- Pruning
- Ensemble models: Bagging, Random Forest, Gradient Boosting

What is Pruning?



Implementation of Pruning for Decision Tree

- We first make the decision tree to a large depth.
- Then we start at the bottom and start removing leaves which are giving us negative returns when compared from the top.

Suppose a split is giving us a gain of say -10 (loss of 10) and then the next split on that gives us a gain of 20. A simple decision tree will stop at step 1 but in pruning, we will see that the overall gain is +10 and keep both leaves.

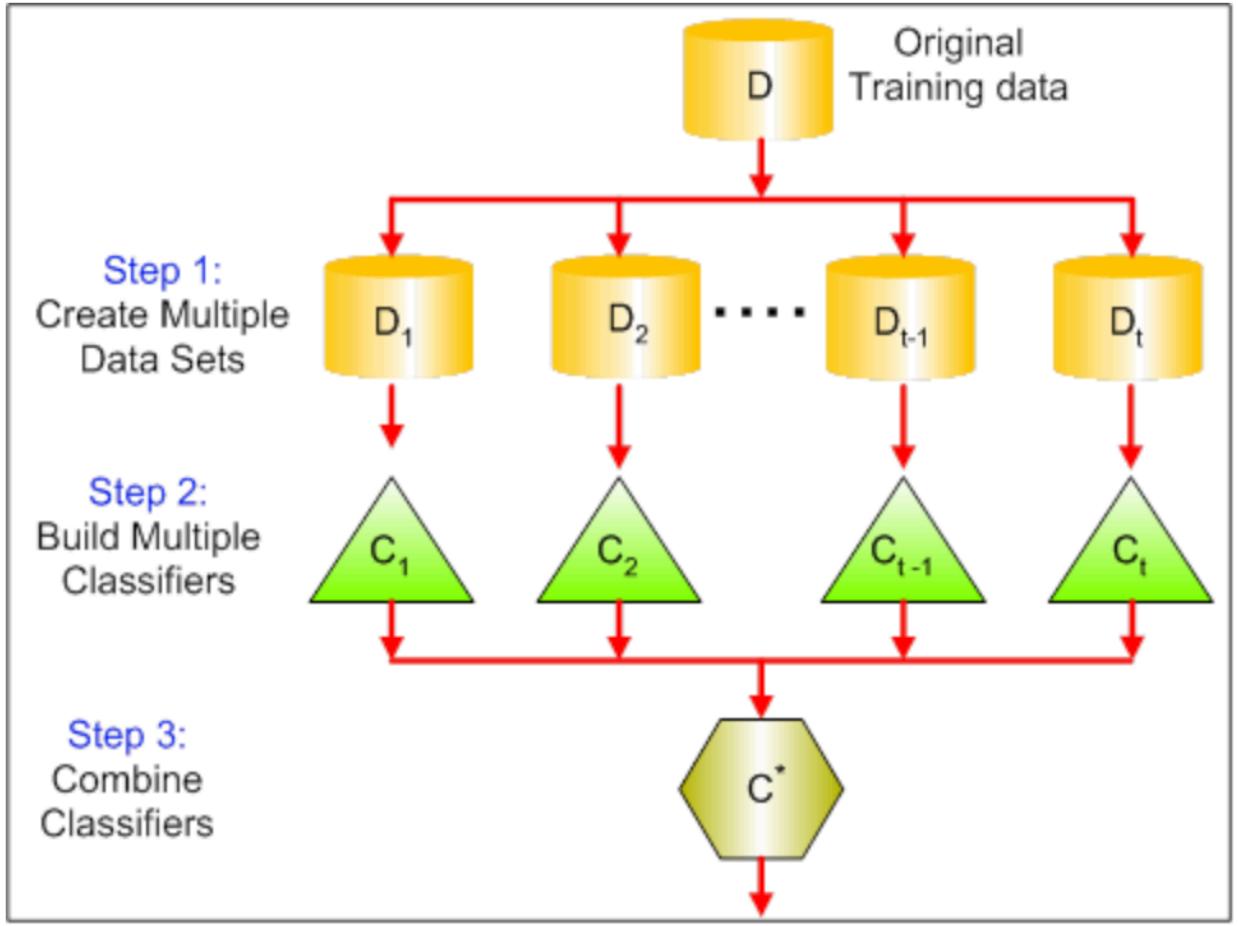
Note that sklearn's decision tree classifier does not currently support pruning. Advanced packages like xgboost have adopted tree pruning in their implementation.

QUESTION

If I can use logistic regression for classification problems and linear regression for regression problems, why is there a need to use trees?

Bagging

1. Create multiple datasets
2. Build a classifier (Decision Tree or SVM, …)
3. Combine Classifiers



Bagging Classification Example

```
from sklearn.ensemble import BaggingClassifier
from sklearn import tree
```

bagging=BaggingClassifier(tree.DecisionTreeClassifier(),n_estimators=200,oob_score=True,max_samples=0.5, max_features=0.5).fit(X_train,y_train)

Bagging Classification Example

In [91]:
```
####perform classification with train-test split####
X, y = shuffle(X, y, random_state=0)

# Split the dataset in two equal parts
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
bagging = BaggingClassifier(tree.DecisionTreeClassifier(),n_estimators=200,oob_score=True,max_samples=0.5, m
Results=bagging.predict(X_test)
print (classification_report(Results,y_test))
```

```
             precision    recall  f1-score   support

          0       1.00      0.88      0.93        16
          1       0.58      0.78      0.67         9
          2       0.75      0.69      0.72        13

avg / total       0.82      0.79      0.80        38
```

Random Forest

1. Create random subsets of data
2. Create decision trees for each subset by randomly selecting m selection of M variables in data. The best split on these m is used to split the node
3. Predict new data by aggregating the predictions of the ntree trees (i.e., majority votes for classification, average for regression)

Pros and Cons of Random Forest

Pros:

- Can be used as feature selection method. the model outputs **Importance of variable,** which can be a very handy feature
- Can be used for both classification and regression problems
- Can handle data with thousands of variables

Cons:

- Regression accuracy is not always as good as classification accuracy
- Random Forest can feel like a black box approach for statistical modelers

Random Forest Classification Example

```
from sklearn.ensemble import RandomForestClassifier
```

class sklearn.ensemble.RandomForestClassifier(*n_estimators=10*, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *min_impurity_split=None*, *bootstrap=True*, *oob_score=False*, *n_jobs=1*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*)

Boosting

The term ‘Boosting’ refers to a family of algorithms which converts weak learner to strong learners.

Step 1: The base learner takes all the distributions and assign equal weight or attention to each observation.

Step 2: If there is any prediction error caused by first base learning algorithm, then we pay higher attention to observations having prediction error. Then, we apply the next base learning algorithm.

Step 3: Iterate Step 2 till the limit of base learning algorithm is reached or higher accuracy is achieved.

Example of Boosting : Gradient Boosting (GBM)

History of Boosting

- Invent Adaboost, the first successful boosting algorithm [Freund et al., 1996, Freund and Schapire, 1997]

- Formulate Adaboost as gradient descent with a special loss function[Breiman et al., 1998, Breiman, 1999]

- Generalize Adaboost to Gradient Boosting in order to handle a variety of loss functions [Friedman et al., 2000, Friedman, 2001]

Gradient Boosting

- Gradient Boosting = Gradient Descent + Boosting

1. Fit an additive model (ensemble) in a forward stage-wise manner.
2. In each stage, introduce a weak learner to compensate the shortcomings of existing weak learners.
3. In Gradient Boosting,“shortcomings” are identified by gradients.
4. Recall that, in Adaboost,“shortcomings” are identified by high-weight data points.
5. Both high-weight data points and gradients tell us how to improve our model.

Gradient Boosting Classification Example

```
from sklearn.ensemble import GradientBoostingClassifier
```

```
cv_params = {'n_estimators':[10,50,100,200],'max_depth': [3,5,7]}
optimized_GBM = GridSearchCV(estimator=GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,random_state
```

Gradient Boosting Regression Example

```
from sklearn import ensemble
```

```
# ##############################################################################
# Load data
boston = datasets.load_boston()
X, y = shuffle(boston.data, boston.target, random_state=13)
X = X.astype(np.float32)
offset = int(X.shape[0] * 0.9)
X_train, y_train = X[:offset], y[:offset]
X_test, y_test = X[offset:], y[offset:]

# ##############################################################################
# Fit regression model
params = {'n_estimators': 500, 'max_depth': 2, 'min_samples_split': 2,
          'learning_rate': 0.01, 'loss': 'ls'}
clf = ensemble.GradientBoostingRegressor(**params)

clf.fit(X_train, y_train)
mse = mean_squared_error(y_test, clf.predict(X_test))
print("MSE: %.4f" % mse)
```

Homework 2

(1) Use SVR (SVM for regression) for Boston dataset as following: Fit a regression three different models with three different kernels ‘rbf’,’poly’ and ‘linear’ with C=100, GAMMA=0.1 AND DEGREE=2. Which model has lower error?
(2) Use Breast cancer dataset to fit a gradient boosting model for classification as following: Use Gridsearch to find the best n_estimators=[10,100,200,500] and max_depth=[2,3,5,7]. What are the best parameters? Use ‘classification_report’ to report the accuracy of classification.
(3) Using Breast cancer dataset apply bagging method with decision tree for the following: Use Gridsearch to find the best parameters for n_estimator=[10,100,200,500], max_depth=[2,3,5,7]. What are the best parameters? Use ‘classification_report’ to report the accuracy.

Homework 2 (bonus)

Using Breast cancer dataset apply random forest for the following: Use Gridsearch to find the best parameters for n_estimator=[10,100,200,500], max_depth=[2,3,5,7]. What are the best parameters? Use ‘classification_report’ to report the accuracy.