Tree-based Methods

Elsa Ferreira Gomes

2024/2025

https://www.youtube.com/watch?v=GNeqvf_JdJ

Decision Trees

```
from sklearn import tree, ensemble
import matplotlib.pyplot as plt
```

Tree-based Methods

Decision trees can be applied to **Regression** and **Classification** problems.

- Tree-based methods are fast and flexible, and they can be applied to a number of data types.
- Trees are all splitting rules used to segment the predictor space (so be summarized a rule).
- Three types of ensemblers are known as decision tree learners.

Pros and Cons of Tree-based Methods

- Our simple and useful for interpretation.
- Inferences they typically are not competitive with the best supervised learning approaches in terms of predictive accuracy.
- Tree-based learners (and dense) given multiple trees combined to yield a prediction.

```
from sklearn import tree, ensemble
import matplotlib as mpl, pandas as pd
import numpy as np
import matplotlib.pyplot as plt, seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('/kaggle/input/boston-housing/boston.csv')
df.shape
```

```
(506, 14)
```

```
df.info()
```

```
<class> pandas.core.frame.DataFrame
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   crim    506 non-null    float64
 1   zn      506 non-null    float64
 2   indus   506 non-null    float64
 3   chas    506 non-null    int64
 4   nox     506 non-null    float64
 5   rm      506 non-null    float64
 6   age     506 non-null    float64
 7   dis     506 non-null    float64
 8   rad     506 non-null    int64
 9   tax     506 non-null    int64
 10  ptratio 506 non-null    float64
 11  black   506 non-null    float64
 12  lstat   506 non-null    float64
 13  medv    506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.4 KB
```

```
df.describe()
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455434	356.674032	12.653063	22.532806
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.141062	9.197104
min	0.006320	0.000000	0.46000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000	5.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	7.073500	17.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.360000	21.200000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955000	25.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970000	50.000000

```
df.hist(figsize=(20,20))
```

Regression Trees

- Can provide output **regression problem**

```
df = df.drop(['zn', 'chas', 'rad', 'black'], axis=1)
df.shape
```

```
(506, 10)
```

```
sns.pairplot(df, kind='reg')
```

How would you stratify it?

Using a standard tree one does, grows a large tree (yellow, red)

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
df = df.dropna()
X = df[['Years', 'Hits']]
y = df['Salary']
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
dtreg = DecisionTreeRegressor(max_leaf_nodes=3)
dtreg.fit(X_train, y_train)
print("Regression tree with 3 terminal nodes")
print("RMSE: {:f}".format(np.sqrt(mean_squared_error(y_test, dtreg.predict(X_test)))))
```

```
Regression tree with 3 terminal nodes
RMSE: 417.962160
```

```
df.shape
```

```
(263, 20)
```

```
pd.qcut(df['Salary'], 3).head()
```

```
0    (67.500, 425.000]
1    (425.000, 1000.0]
2    (425.000, 1000.0]
4    (425.000, 1000.0]
5    (425.000, 1000.0]
Name: Salary, dtype: category
Categories (3, interval): [(67.500, 425.000], (425.000, 1000.0], (1000.0, 2459.000]]
```

```
df['Salary'].hist(bins=50)
```

How can the tree be learned?

- Growing trees are difficult, **can also exhaustively**.
- We use **heuristic search** to grow the tree.
- Typically **greedy search**:
  - **left dividing**
  - **top-down** approach
  - **recursive** partitioning (Recall Binary Search Algorithm of Decision Trees)
  - CART (DC.C4.5)

We use decision tree to:

- Decision trees are typically shared upside down - the leaves are at the bottom of the tree
- Decision trees are typically shared upside down - the leaves are at the bottom of the tree
- The tree is known value as the terminal node or leaves (in the figure, R_1, R_2 and R_3)

Easy to interpret

- Trees are much closer human-inspired decision-making, and pair well with tree-explained and linear-explained. Most explainable models.
- Decision trees can be approximated as the number of splits that leads to a unique prediction. Trees are easy to play little rules to the data.
  - Not strong players: that learn in the magic leagues for first or more, plus, the number of little rules in the process **pure leaf**. Pure leaf, and plus the rules learn the rule pure leaf is more higher rules.

How can the tree be learned?

- Growing trees are difficult, **can also exhaustively**.
- We use **heuristic search** to grow the tree.
- Typically **greedy search**:
  - **left dividing**
  - **top-down** approach
  - **recursive** partitioning (Recall Binary Search Algorithm of Decision Trees)
  - CART (DC.C4.5)

We use decision tree to:

- Decision trees are typically shared upside down - the leaves are at the bottom of the tree
- Decision trees are typically shared upside down - the leaves are at the bottom of the tree
- The tree known value as the terminal node or leaves (in the figure, R_1, R_2 and R_3)

Easy to interpret

- Trees are much closer human-inspired decision-making, and pair well with tree-explained and linear-explained. Most explainable models.
- Decision trees can be approximated as the number of splits that leads to a unique prediction. Trees are easy to play little rules to the data.
  - Not strong players: that learn in the magic leagues for first or more, plus, the number of little rules in the process **pure leaf**. Pure leaf, and plus the rules learn the rule pure leaf is more higher rules.

How can the tree be learned?

- Growing trees are difficult, **can also exhaustively**.
- We use **heuristic search** to grow the tree.
- Typically **greedy search**:
  - **left dividing**
  - **top-down** approach
  - **recursive** partitioning (Recall Binary Search Algorithm of Decision Trees)
  - CART (DC.C4.5)

We use decision tree to:

- Decision trees are typically shared upside down - the leaves are at the bottom of the tree
- Decision trees are typically shared upside down - the leaves are at the bottom of the tree
- The tree known value as the terminal node or leaves (in the figure, R_1, R_2 and R_3)

Easy to interpret

- Trees are much closer human-inspired decision-making, and pair well with tree-explained and linear-explained. Most explainable models.
- Decision trees can be approximated as the number of splits that leads to a unique prediction. Trees are easy to play little rules to the data.
  - Not strong players: that learn in the magic leagues for first or more, plus, the number of little rules in the process **pure leaf**. Pure leaf, and plus the rules learn the rule pure leaf is more higher rules.

Terminology:

The regions R_1, R_2 and R_3 are known as **terminal nodes**

Example - Regression tree analysis

Regression tree analysis for the Hitters data. The training, black curve, cross validation (green curve), and test MSE (orange) are shown as a function of the number of terminal nodes in the pruned tree. As we can see, the minimum cross validation error occurs at a tree size of three.

Example - Heart data

- Data set: Heart data set, Cleveland Clinic Foundation cohort, training and test series, for different sizes of the pruned tree
- Bottom Right: The pruned tree corresponding to the minimal cross validation error

Error measures - Classification

- Misclassification error averaged over regions
- Each region has a class label, class with probability p_k
  - **Gini index**, a measure of total variance across the K classes. Indeed, purity - a small value indicates that a node contains predominantly observations from a single class. $G = \sum_{k=1}^{K} p_k(1-p_k)$
  - **Entropy**
    - As a measure of information or number between 0 and 1 it used to evaluate the homogeneity of sets (measure of disorder).

$$\text{Entropy} = -\sum_{k=1}^{K} p_k \log(p_k)$$

  - Tree loss (mean squared) Distributed Entropy ( )

```
Note: Classification tree: Deviance, Entropy, Gini index.
Regression tree: Residual sum of squares.
```

(https://stanfordmlgroup.github.io/projects/docs/lecture-error-vs-entropy.html)

Error measures - Classification

- The measure that represents error minimization of a tree if leaf splits a set of n cases S is called a **information gain**

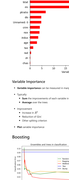
$$IG(S, T) = Entropy(S) - \sum_{t \in T} \left(\frac{|t|}{|S|} \cdot Entropy(t) \right)$$

- When building a classification tree, often the first criteria the strategy will typically seek to evaluate the purity of a particular split (the extent to which it separates the classes in the data).
- The approaches might be used when pruning the tree, but the classification error rate is preferable if prediction accuracy of the final pruned tree is the goal.

```
>> CART (Classification and Regression Trees)
```

```
Note: Decision Entropy, Gini Index, Deviance.
Regression Residual sum of squares.
```

Sex	p_1	p_2	$-p_1 \log_2 p_1$	$-p_2 \log_2 p_2$	Entropy(S)
M	0.5	0.5	0.5	0.5	1
F	0.2	0.8	0.464	0.258	0.722



Bias variance

- Typically we have to sacrifice low bias to have low variance and vice versa
- Bias and variance are complementary
- Ensemble is a way **Decreases** variance
  - Bagging **Reduces variance** and **Increases** bias
  - Bagging **Reduces Future** and Boosting
- Ensemble methods are a way of combining more than one prediction
- Ensemble can consist of different methods, from different samples.
  - Ensemble Decision is made by combining the decisions of each model

The decision trees suffer from high variance.

- If we split the training data into two parts at random, and fit a decision tree to both halves, the results that we get could be quite **different**.
- In contrast, a procedure with low variance will yield similar results if applied repeatedly to distinct data sets.

Ensembles

- A **ensemble** method combines many simple "building block" models to obtain a single and potentially very powerful model.
  - These simple building blocks models are commonly known as weak learners, since they may lead to mediocre predictions on their own.
  - Bagging, random forests, Boosting, and Bayesian additive regression trees are ensemble methods for which the simple building blocks are regression or classification trees.

Random Forests

- Random forests provide an improvement over bagged trees. - decorrelates the trees.
- A very powerful classification and regression method.
- Random forests consists of decision trees are decorrelated from one another.
- Are when building these decision trees,
  - each time a split in a tree is considered, a random selection of m predictors is chosen as split candidates from the full set of p predictors.
  - The split is allowed to use only one of these m predictors.
  - A fresh sample of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$
  - The number of predictors considered at each split is approximately equal to the square root of the total number of predictors (4 out of the 10 in the Heart data).

Random Forests

- The Random Forest approach
  - Build an Ensemble of trees then between 25 with 25 - 1000 and p predictors
  - Each tree is grown as follows:
    - Choose **bootstrap** n samples from D
    - Learn a tree $T(n)$ from D using the **Bootstrap** selected predictors
    - Choose m predictors, $m \approx \sqrt{p}$ and select split on $m \ll p$
- The final model is ensemble

$$f_{ensemble}(\cdot) = \sum_{t=1}^{T} f_t(\cdot)$$

- In classification, the decision is by **majority vote**.

Why Random Forests?

- Selecting $\sqrt{p}$ predictors of the variables
  - What if the **best predictor** is not included?
- Bagging is very strong predictor with high **Bias**
  - Averaging highly correlated trees has **smaller effect on variance**
- In Random Forest **weaker predictors** also have a big
  - This **reduces correlation** between trees

```
>> # Random Forf 10 Predictors in the decision
>> # Random Forest
>> # Fitting 10 random decision trees
>> RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
max_depth=None, max_features='auto',
max_leaf_nodes=None, min_impurity_decrease=0.0,
min_impurity_split=None, min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0,
n_estimators=10, n_jobs=1, oob_score=False,
random_state=None, verbose=0, warm_start=False)
>> [0.85245902 0.81666667 0.83333333]
```

```
>> # Random Forests using 10 decision
>> RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
max_depth=None, max_features='auto',
max_leaf_nodes=None, min_impurity_decrease=0.0,
min_impurity_split=None, min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0,
n_estimators=10, n_jobs=1, oob_score=False,
random_state=None, verbose=0, warm_start=False)
>> [0.85245902 0.81666667 0.83333333]
```



```
>> # Random Forests using 10 decision
>> RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
max_depth=None, max_features='auto',
max_leaf_nodes=None, min_impurity_decrease=0.0,
min_impurity_split=None, min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0,
n_estimators=10, n_jobs=1, oob_score=False,
random_state=None, verbose=0, warm_start=False)
>> [0.85245902 0.81666667 0.83333333]
>> [('Sex', 0.02000000000000002), ('Slope', 0.080000000000000002), ('RestECG', 0.
09999999999999998), ('Oldpeak', 0.14999999999999997), ('MaxHR', 0.16999999999
999999), ('Ca', 0.17999999999999994), ('Chol', 0.18999999999999995), ('Age', 0.
20999999999999996), ('RestBP', 0.22999999999999998), ('Thal:a', 0.
26000000000000001), ('Thal:b', 0.34000000000000008)]
```


Variable Importance

- **Variable Importance** can be measured in many **different** ways
  - Gini index
  - Deviance
  - **Bias** the improvements at each variable in a tree
  - **Coverage** over the trees
- Variable importance
  - increases if
  - Reduction of Gini
  - Reduction of Deviance
- **Bias** variable importance

Boosting

Summary of Random Forests

Random forests are a powerful method.

- Advantages
  - Good possible ability
  - Random forests are accurate and very fast, because of the simplicity of the underlying decision trees.
    - Both tasks can be straightforwardly parallelized, because the individual trees are entirely independent of the
  - Random forests can handle thousands of input variables without variable deletion, and they can also estimate variable probability, successor's Gini index used with the greatest predict method.
  - The independently created is extremely flexible and can thus perform well on tasks that are unable for lip other estimators.
- Cons
  - the results are not easily interpretable. that is, if you would like to draw conclusions about the meaning of the classification model, random forests may not be the best choice.

Summary

- Random trees are simple and interpretable models for regression and classification.
  - But often are competitive with other methods in terms of prediction accuracy.
- Random forest Bagged and Boosting is a good substitute for improving the prediction accuracy of trees

References

- https://arxiv.org/pdf/1407.7502.pdf
- ...