Random Forests & XGBoost

Fartash Faghri

University of Toronto

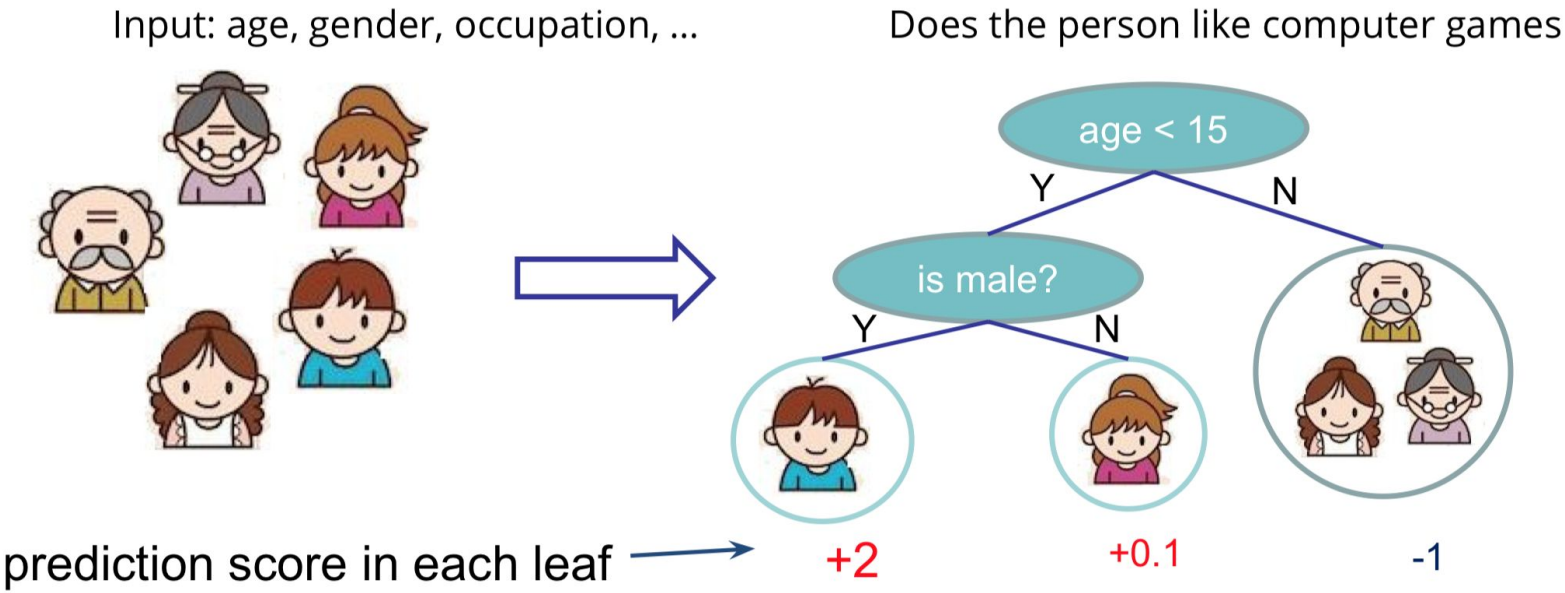
CSC2515, Fall 2019

Decision Trees

HW1

- Handles tabular data
- Features can be of any type (discrete, categorical, raw text, etc)
- Features can be of different types
- No need to “normalize” features
- Too many features? DTs can be efficient by looking at only a few.
- Easy to interpret

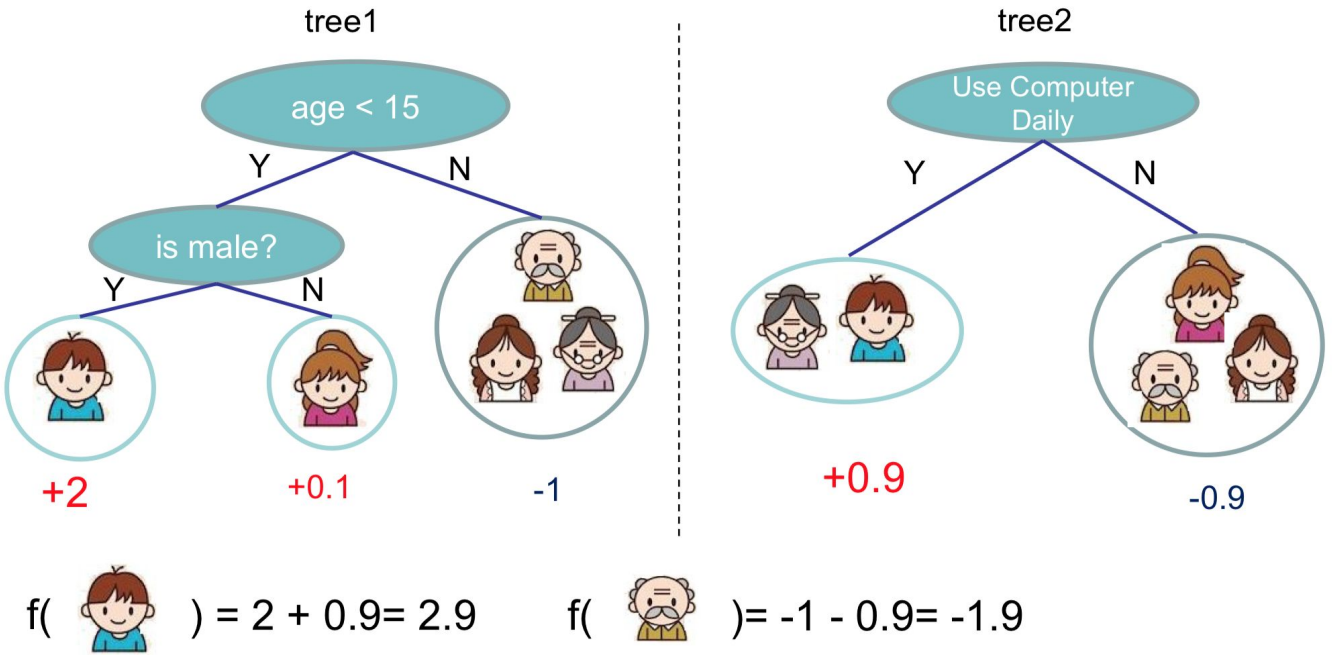
Decision Trees



(XGBoost slides)

Random Forests

Average multiple decision trees



f( ) = 2 + 0.9= 2.9 f( )= -1 - 0.9= -1.9

(XGBoost slides)

Random Forests

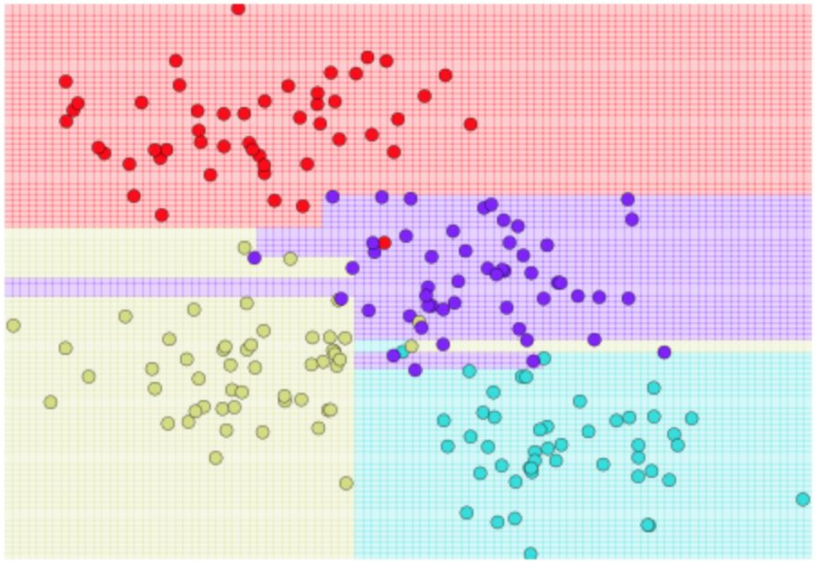
Scikit-learn ipynb

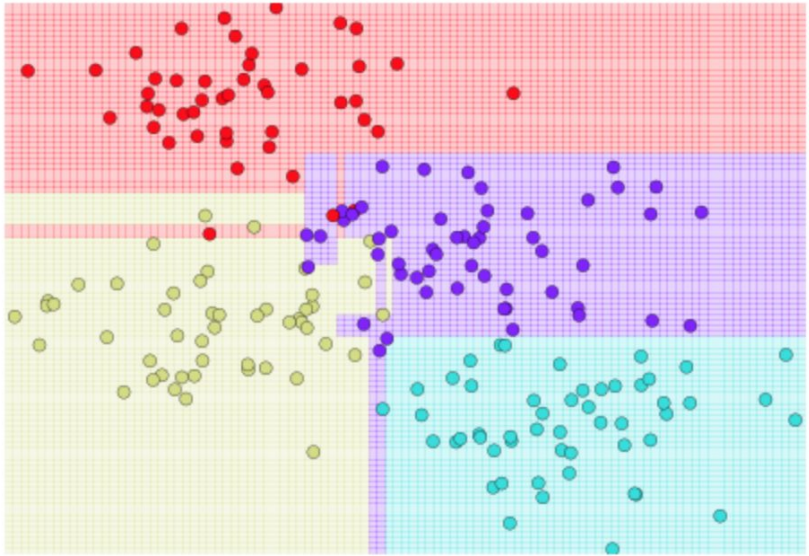
Scikit-learn official docs

Tabular data example and ipynb

One DT Overfits

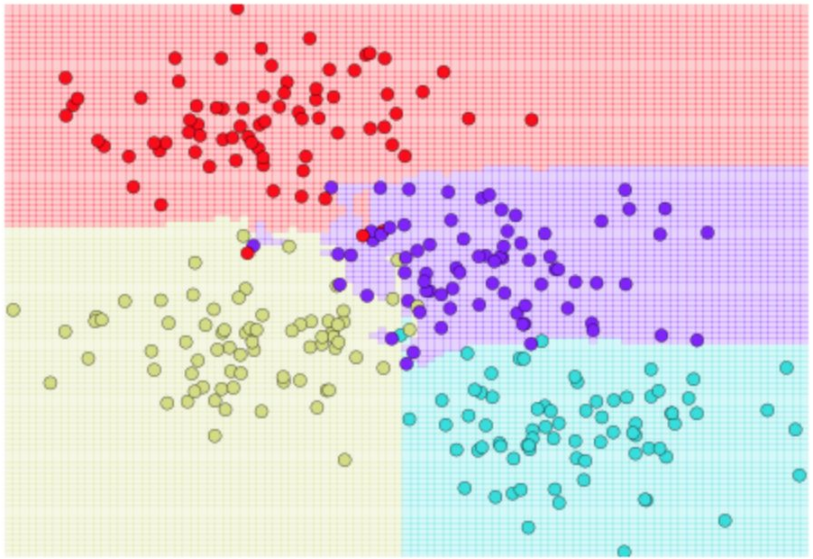
```
>>> from sklearn import tree
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(X, Y)
```





Averaging DTs in a Random Forest

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = RandomForestClassifier(n_estimators=10)
>>> clf = clf.fit(X, Y)
```



Train Model

```
# Import the model we are using
from sklearn.ensemble import RandomForestRegressor

# Instantiate model
rf = RandomForestRegressor(n_estimators= 1000, random_state=42)

# Train the model on training data
rf.fit(train_features, train_labels);
```

max_depth : The maximum depth of the tree.

Make Predictions on Test Data

In [14]:
```
# Use the forest's predict method on the test data
predictions = rf.predict(test_features)

# Calculate the absolute errors
errors = abs(predictions - test_labels)

# Print out the mean absolute error (mae)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')
```

```
Mean Absolute Error: 3.83 degrees.
```

Randomization methods

Each DT is trained on:

- Random subset of training data (sklearn.ensemble.RandomForestClassifier)
- Random subset of features (sklearn.ensemble.RandomForestClassifier)
- Noisy thresholds (sklearn.ensemble.ExtraTreesClassifier)

*Parallel construction and prediction

Tabular Data

Ipython notebook example end post

Data Acqusition

In [1]:
```
# Pandas is used for data manipulation
import pandas as pd

# Read in data as pandas dataframe and display first 5 rows
features = pd.read_csv('data/temps.csv')
features.head(5)
```

Out[1]:

	year	month	day	week	temp_2	temp_1	average	actual	friend
0	2016	1	1	Fri	45	45	45.6	45	29
1	2016	1	2	Sat	44	45	45.7	44	61
2	2016	1	3	Sun	45	44	45.8	41	56
3	2016	1	4	Mon	44	41	45.9	40	53
4	2016	1	5	Tues	41	40	46.0	44	41

In [2]:
```
print('The shape of our features is:', features.shape)
```

```
The shape of our features is: (348, 9)
```

In [3]:
```
# Descriptive statistics for each column
features.describe()
```

Out[3]:

	year	month	day	temp_2	temp_1	average	actual	friend
count	348.0	348.000000	348.000000	348.000000	348.000000	348.000000	348.000000	348.000000
mean	2016.0	6.477011	15.514368	62.511494	62.560345	59.760632	62.543103	60.034483
std	0.0	3.498380	8.772982	11.813019	11.767406	10.527306	11.794146	15.626179
min	2016.0	1.000000	1.000000	35.000000	35.000000	45.100000	35.000000	28.000000
25%	2016.0	3.000000	8.000000	54.000000	54.000000	49.975000	54.000000	47.750000
50%	2016.0	6.000000	15.000000	62.500000	62.500000	58.200000	62.500000	60.000000
75%	2016.0	10.000000	23.000000	71.000000	71.000000	69.025000	71.000000	71.000000
max	2016.0	12.000000	31.000000	92.000000	92.000000	77.400000	92.000000	95.000000

Data preparation

One-hot encoding:

week
Mon
Tue
Wed
Thu
Fri

Mon	Tue	Wed	Thu	Fri
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

```
# One-hot encode categorical features
features = pd.get_dummies(features)
```

Data preparation

Why should be prepare data?

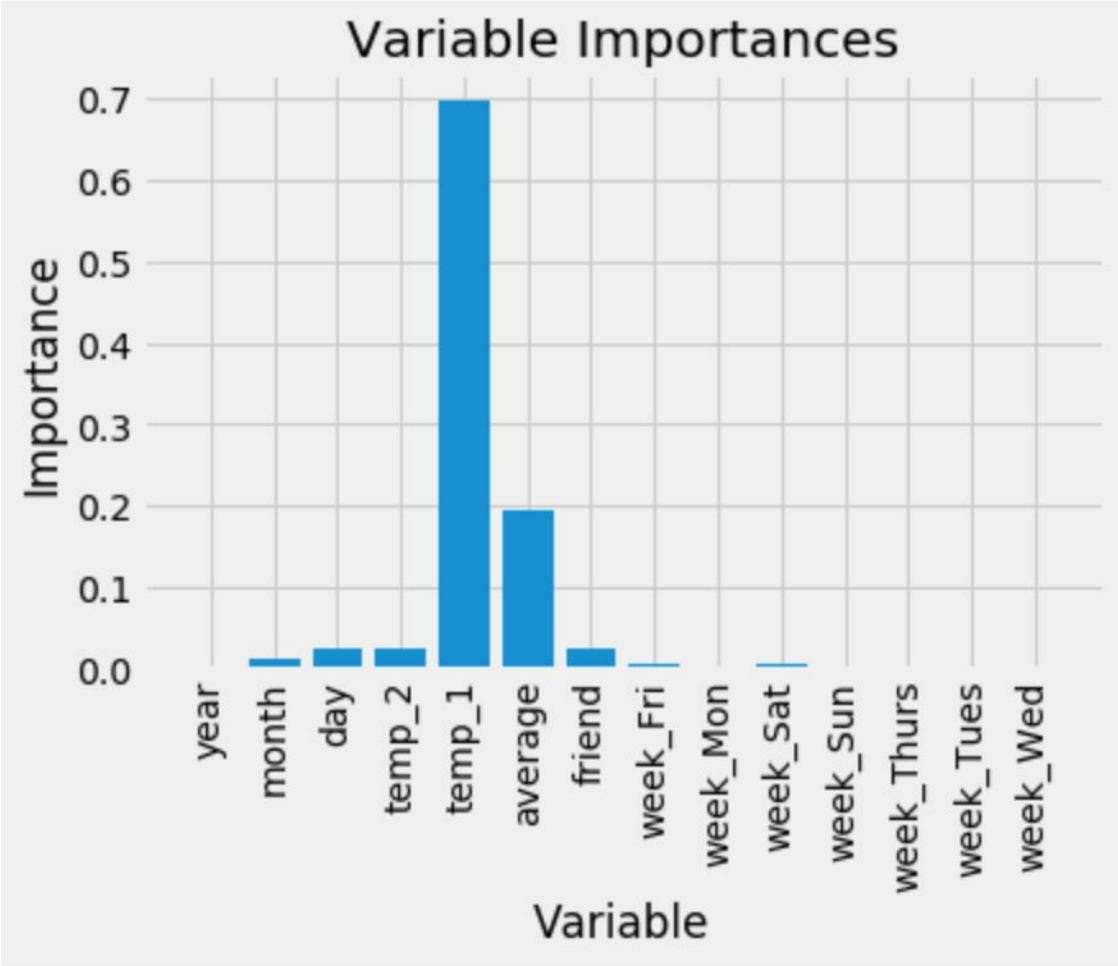
Any other encoding?

We also use one-hot encoding in cross-entropy, remember why? (Tutorial 2)

Feature importance

Feature importance is calculated as the decrease in node impurity weighted by the probability of reaching that node.

```
# Get numerical feature importances
importances = list(rf.feature_importances_)
```


Variable Importances
Importance
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
year
month
day
temp_2
temp_1
average
friend
week_Fri
week_Mon
week_Sat
week_Sun
week_Thurs
week_Tues
week_Wed
Variable

Model with Two Most Important Features

:

```
# New random forest with only the two most important variables
rf_most_important = RandomForestRegressor(n_estimators= 1000, random_state=42)

# Extract the two most important features
important_indices = [feature_list.index('temp_1'), feature_list.index('average')]
train_important = train_features[:, important_indices]
test_important = test_features[:, important_indices]

# Train the random forest
rf_most_important.fit(train_important, train_labels)
```

AdaBoost

Lecture 4 (Slides 43-70)

- Additive models
- Exponential loss $L(y, \hat{y}) = \exp(-\hat{y}y)$

Weak-learners can be any function, e.g. decision trees and decision stumps.

Gradient Boosting

Additive model with loss L:

$$\min_{\alpha_{n=1:N},\beta_{n=1:N}} L\left(y, \sum_{n=1}^{N} \alpha_n f(x, \beta_n)\right)$$

GB approximately solves this objective iteratively and greedily:

$$\min_{\alpha_n,\beta_n} L\left(y, f_{n-1}((x) + \alpha_n f_n(x, \beta_n)\right)$$

Elements of Statistical learning (Chapter 10.10)

XGBoost (A Scalable Tree Boosting System)

Gradient Tree Boosting with Regularization

Parallelization construction on CPU cores

Distributed training on a cluster of machines (large models)

Out-of-Core computing (large datasets that do not fit in memory)

XGBoost (Getting Started)

```
import xgboost as xgb

# read in data
dtrain = xgb.DMatrix('demo/data/agaricus.txt.train')
dtest = xgb.DMatrix('demo/data/agaricus.txt.test')

# specify parameters via map
param = {'max_depth':2, 'eta':1, 'silent':1, 'objective':'binary:logistic' }
num_round = 2
bst = xgb.train(param, dtrain, num_round)

# make prediction
preds = bst.predict(dtest)
```

XGBoost

Slides (formulation on slides 17-26)

Recorded Presentation

Official Examples

XGBoost Official Documentation

Basic walk-through