# RANDOM FOREST

Individual decision tree algorithms can be prone to problems, such as bias and over-fitting. A more practical approach is to construct multiple decision trees and combine the results into a single output. This approach is termed **ensemble methods**. The most well-known ensemble method is **bagging** (also known as **bootstrap aggregation**, **bagging=bootstrap+aggregation**). This method was introduced in 1996 by Leo Breiman. In this method, data points in the training set are sampled with replacement (producing a **bootstrap sample**), one-third of which, known as the **out-of-bag (OOB) sample**, is set aside for cross-validation, and the remaining two-thirds of the sample is used to build a decision tree. Decision trees are generated for each bootstrap sample independently from others. The results are then aggregated depending on the type of trees used. If regression trees are trained, the average of predicted values are computed. If classification trees are fitted, the majority of the predictions define the final predicted class. The ensemble method reduces variance and, as the result, yields more accurate predictions than individual decision trees.

The **random forest algorithm** is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees. **Feature randomness** (also known as **feature bagging** or the **random subspace method**) generates a random subset of variables (also called **features**), which ensures low correlation among decision trees. This is a key difference between decision trees and random forests. While decision trees consider all the possible variable splits, random forests only select a subset of those variables.

Random forest algorithms have three main hyper-parameters that need to be set before training. These are node size, the number of trees, and the number of variables sampled.

## Variable Importance

Random forest makes it easy to evaluate the **variable importance** (or the **contribution** of each splitting variable to the model). It is sometimes termed the **feature importance**. There are two commonly used ways to evaluate variable importance that are characteristically different from each other. One method is termed the **loss reduction** or **Gini increase** or **Gini importance** or **impurity reduction** or **mean decrease in impurity (MDI)**. It is used to measure how much the model's accuracy decreases when a given variable is excluded. For a random forest with regression trees, the loss functions are the **mean squared error** $MSE = RSS/n$, and the **absolute error** $\frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$. The second feature importance method is the **permutation importance** (or the **mean decrease accuracy (MDA)**), which identifies the average decrease in accuracy by randomly permuting the variable values in OOB samples.

**Example.** Consider the data in the file "housing_data.csv". We construct random forest regres-

sion for this data set.

In SAS:

```
proc import out=housing
datafile="./housing_data.csv" dbms=csv replace;
run;

/*SPLITTING DATA INTO 80% TRAINING AND 20% TESTING*/
proc surveyselect data=housing rate=0.8 seed=502305
out=housing outall method=srs;
run;

/*BUILDING RANDOM FOREST REGRESSION*/
proc hpforest data=housing seed=829743
maxtrees=60 vars_to_try=4 trainfraction=0.7
maxdepth=50;
target median_house_value/level=interval;
input ocean_proximity/level=nominal;
input housing_median_age total_rooms total_bedrooms
population households median_income/level=interval;
partition rolevar=selected(train='1');
save file='C:/Users/000110888/Desktop/random_forest.bin';
run;
```

Loss Reduction Variable Importance

| Variable | Number of Rules | MSE | OOB MSE | Absolute Error | OOB Absolute Error |
|---|---|---|---|---|---|
| median_income | 35554 | 5.2672E9 | 4.3027E9 | 36134 | 23887 |
| ocean_proximity | 336 | 1.6836E9 | 1.7055E9 | 12553 | 12796 |
| total_rooms | 7375 | 4.5494E8 | -5.897E7 | 4665.515799 | -423.229467 |
| housing_median_age | 7987 | 5.285E8 | -7.79E7 | 5698.148519 | 141.283900 |
| total_bedrooms | 8196 | 3.22E8 | -1.709E8 | 4098.769334 | -795.029912 |
| households | 20105 | 5.2346E8 | -2.364E8 | 6723.747118 | -1030.371514 |
| population | 14896 | 4.4438E8 | -2.694E8 | 5805.772403 | -1414.692176 |

```
/*COMPUTING PREDICTED VALUES FOR TESTING DATA*/
data test;
set housing;
if(selected='0');
```

```
run;

proc hp4score data=test;
id median_house_value;
score file='C:/Users/000110888/Desktop/random_forest.bin'
out=predicted;
run;

/*DETERMINING 10%, 15%, AND 20% ACCURACY*/
data accuracy;
set predicted;
if(abs(median_house_value-P_median_house_value)
<0.10*median_house_value)
then ind10=1; else ind10=0;
if(abs(median_house_value-P_median_house_value)
<0.15*median_house_value)
then ind15=1; else ind15=0;
if(abs(median_house_value-P_median_house_value)
<0.20*median_house_value)
then ind20=1; else ind20=0;
run;

proc sql;
 select sum(ind10)/count(*) as accuracy10,
sum(ind15)/count(*) as accuracy15,
 sum(ind20)/count(*) as accuracy20
 from accuracy;
 quit;
```

| accuracy10 | accuracy15 | accuracy20 |
|---|---|---|
| 0.352113 | 0.485915 | 0.612676 |

In R:

```
#install.packages("randomForest")
library(randomForest)
```

```
housing.data<- read.csv(file="./housing_data.csv", header=TRUE, sep=",")

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
set.seed(902881)
sample <- sample(c(TRUE, FALSE), nrow(housing.data), replace=TRUE, prob=c(0.8,0.2))
train<- housing.data[sample,]
test<- housing.data[!sample,]

#BUILDING RANDOM FOREST REGRESSION
rf.reg<- randomForest(median_house_value ~ housing_median_age + total_rooms + total_bedrooms
+ population + households + median_income + ocean_proximity, data=train, ntree=150, mtry=5,
maxnodes=30)

#DISPLAYING FEATURE IMPORTANCE
print(importance(rf.reg,type=2))


                   IncNodePurity
housing_median_age  4.241691e+11
total_rooms         4.108330e+11
total_bedrooms      2.028437e+11
population          2.153154e+11
households          2.400010e+11
median_income       1.245747e+13
ocean_proximity     2.914203e+12

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
P_median_house_value<- predict(rf.reg, newdata=test)

#accuracy within 10%
accuracy10<-
ifelse(abs(test$median_house_value-P_median_house_value)<0.10*test$median_house_value,1,0)
print(accuracy10<- sum(accuracy10)/length(accuracy10))

0.2857143

#accuracy within 15%
accuracy15<-
ifelse(abs(test$median_house_value-P_median_house_value)<0.15*test$median_house_value,1,0)
print(accuracy15<- sum(accuracy15)/length(accuracy15))

0.4303797
```

```
#accuracy within 20%
accuracy20<-
ifelse(abs(test$median_house_value-P_median_house_value)<0.20*test$median_house_value,1,0)
print(accuracy20<- sum(accuracy20)/length(accuracy20))
```

0.5334539

In Python:

```python
import pandas
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split

housing=pandas.read_csv('C:/Users/000110888/Desktop/housing_data.csv')
coding={'<1H OCEAN': 1, 'INLAND': 2, 'NEAR BAY': 3, 'NEAR OCEAN': 4}
housing['ocean_proximity']=housing['ocean_proximity'].map(coding)
X=housing.iloc[:,0:7].values
y=housing.iloc[:,7].values

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.20,
random_state=348644)

#FITTING RANDOM FOREST REGRESSION TREE
rf_reg=RandomForestRegressor(n_estimators=100, random_state=323445,
max_depth=50, max_features=4)
rf_reg.fit(X_train, y_train)

#DISPLAYING VARIABLE IMPORTANCE
from sklearn.ensemble import ExtraTreesClassifier

var_names=pandas.DataFrame(['housing_median_age','total_rooms','total_bedrooms','population',
'households','median_income','ocean_proximity'], columns=['var_name'])
loss_reduction=pandas.DataFrame(rf_reg.feature_importances_, columns=['loss_reduction'])
var_importance=pandas.concat([var_names, loss_reduction], axis=1)
var_importance=var_importance.sort_values("loss_reduction", axis=0, ascending=False)
print(var_importance)
```

```
             var_name  loss_reduction
5       median_income        0.592797
6     ocean_proximity        0.154105
1         total_rooms        0.061499
0  housing_median_age        0.053247
```

```
3          population        0.051611
4          households        0.044158
2      total_bedrooms        0.042583
```

```python
#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=rf_reg.predict(X_test)

ind10=[]
ind15=[]
ind20=[]

for sub1, sub2 in zip(y_pred, y_test):
    ind10.append(1) if abs(sub1-sub2)<0.10*sub2 else ind10.append(0)
    ind15.append(1) if abs(sub1-sub2)<0.15*sub2 else ind15.append(0)
    ind20.append(1) if abs(sub1-sub2)<0.20*sub2 else ind20.append(0)

#accuracy within 10%
accuracy10=sum(ind10)/len(ind10)
print(accuracy10)

#accuracy within 15%
accuracy15=sum(ind15)/len(ind15)
print(accuracy15)

#accuracy within 20%
accuracy20=sum(ind20)/len(ind20)
print(accuracy20)
```

```
0.36203866432337434
0.5202108963093146
0.648506151142355
```

□

**Example.** Consider the data in the file "pneumonia_data.csv". We construct a random forest binary classifier for this data set.

In SAS:

```
proc import out=pneumonia datafile="./pneumonia_data.csv" dbms=csv replace;

/*SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS*/
proc surveyselect data=pneumonia rate=0.8 seed=6132208
```

```
out=pneumonia outall method=srs;
run;

/*BUILDING RANDOM FOREST BINARY CLASSIFIER*/
proc hpforest data=pneumonia seed=115607
maxtrees=60 vars_to_try=4 trainfraction=0.7
maxdepth=50;
target pneumonia/level=binary;
input gender tobacco_use/level=nominal;
input age PM2_5/level=interval;
partition rolevar=selected(train='1');
save file='C:/Users/000110888/Desktop/random_forest.bin';
run;

/*COMPUTING PREDICTED VALUES FOR TESTING DATA*/
data test;
set pneumonia;
if(selected='0');
run;

proc hp4score data=test;
id pneumonia;
score file='C:/Users/000110888/Desktop/random_forest.bin'
out=predicted;
run;
```

Loss Reduction Variable Importance

| Variable | Number of Rules | Gini | OOB Gini | Margin | OOB Margin |
|---|---|---|---|---|---|
| gender | 84 | 0.056581 | 0.05607 | 0.113161 | 0.112805 |
| tobacco_use | 263 | 0.044094 | 0.04157 | 0.088188 | 0.086383 |
| PM2_5 | 7647 | 0.271965 | 0.00917 | 0.543929 | 0.279844 |
| age | 3208 | 0.085176 | -0.05001 | 0.170351 | 0.035585 |

```
/*COMPUTING PREDICTION ACCURACY FOR TESTING DATA*/
data predicted;
set predicted;
match=(pneumonia=lowcase(I_pneumonia));
run;

proc sql;
select sum(match)/count(*) as accuracy
from predicted;
quit;
```

| accuracy |
|---|
| 0.823188 |

In R:

```
pneumonia.data<- read.csv(file="./pneumonia_data.csv", header=TRUE, sep=",")

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
set.seed(447558)
sample <- sample(c(TRUE, FALSE), nrow(pneumonia.data), replace=TRUE, prob=c(0.8,0.2))
train<- pneumonia.data[sample,]
test<- pneumonia.data[!sample,]

#BUILDING RANDOM FOREST BINARY CLASSIFIER
library(randomForest)
rf.class<- randomForest(as.factor(pneumonia) ~ age + gender + tobacco_use + PM2_5, data=train,
ntree=150, mtry=4, maxnodes=30)

#DISPLAYING FEATURE IMPORTANCE
print(importance(rf.class,type=2))
```

```
            MeanDecreaseGini
age                 36.39492
gender              67.32295
tobacco_use         56.93959
PM2_5              151.83923
```

```
#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
predclass<- predict(rf.class, newdata=test)
test<- cbind(test,predclass)

accuracy<- c()
n<- nrow(test)
for (i in 1:n)
    accuracy[i]<- ifelse(test$pneumonia[i]==test$predclass[i],1,0)

print(accuracy<- sum(accuracy)/length(accuracy))
```

0.815864

In Python:

```
import pandas
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

pneumonia_data=pandas.read_csv('C:/Users/000110888/Desktop/pneumonia_data.csv')
code_gender={'M':1,'F':0}
code_tobacco_use={'yes':1,'no':0}
code_pneumonia={'yes':1,'no':0}

pneumonia_data['gender']=pneumonia_data['gender'].map(code_gender)
pneumonia_data['tobacco_use']=pneumonia_data['tobacco_use'].map(code_tobacco_use)
pneumonia_data['pneumonia']=pneumonia_data['pneumonia'].map(code_pneumonia)

X=pneumonia_data.iloc[:,0:4].values
y=pneumonia_data.iloc[:,4].values

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.20,
random_state=786756)

#FITTING RANDOM FOREST BINARY CLASSIFIER
rf_class=RandomForestClassifier(n_estimators=150, criterion='entropy',
random_state=778554, max_depth=50, max_features=4)
rf_class.fit(X_train, y_train)

#DISPLAYING VARIABLE IMPORTANCE
from sklearn.ensemble import ExtraTreesClassifier

var_names=pandas.DataFrame(['gender','age','tobacco_use','PM2_5'], columns=['var_name'])
loss_reduction=pandas.DataFrame(rf_class.feature_importances_, columns=['loss_reduction'])
var_importance=pandas.concat([var_names, loss_reduction], axis=1)
var_importance=var_importance.sort_values("loss_reduction", axis=0, ascending=False)
print(var_importance)
```

```
      var_name  loss_reduction
3        PM2_5        0.581401
1          age        0.233343
0       gender        0.095434
2  tobacco_use        0.089823
```

```
#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=rf_class.predict(X_test)
y_test=pandas.DataFrame(y_test,columns=['pneumonia'])
y_pred=pandas.DataFrame(y_pred,columns=['predicted'])
df=pandas.concat([y_test,y_pred],axis=1)

match=[]
for i in range(len(df)):
    if df['pneumonia'][i]==df['predicted'][i]:
        match.append(1)
    else:
        match.append(0)

accuracy=sum(match)/len(match)

print(accuracy)
```

```
0.8121387283236994
```

□

**Example.** Consider the data in the file "movie_data.csv". We construct random forest multinomial classifier for this data set.

In SAS:

```
proc import out=movie
datafile="./movie_data.csv" dbms=csv replace;
run;

/*SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS*/
proc surveyselect data=movie rate=0.8 seed=550040
out=movie outall method=srs;
```

```
run;

/*BUILDING RANDOM FOREST MULTINOMIAL CLASSIFIER*/
proc hpforest data=movie seed=454545
maxtrees=150 vars_to_try=4 trainfraction=0.7
maxdepth=10;
target rating/level=ordinal;
input age member/level=nominal;
input age nmovies/level=interval;
partition rolevar=selected(train='1');
save file='C:/Users/000110888/Desktop/random_forest.bin';
run;
```

Loss Reduction Variable Importance

| Variable | Number of Rules | Gini | OOB Gini | Margin | OOB Margin |
|---|---|---|---|---|---|
| member | 378 | 0.004665 | -0.00439 | 0.002648 | -0.00475 |
| age | 1234 | 0.049053 | -0.02003 | 0.038625 | -0.01943 |
| nmovies | 3134 | 0.051944 | -0.02365 | 0.059274 | -0.00833 |

```
/*COMPUTING PREDICTED VALUES FOR TESTING DATA*/
data test;
set movie;
if(selected='0');
run;

proc hp4score data=test;
id rating;
score file='C:/Users/000110888/Desktop/random_forest.bin'
out=predicted;
run;

/*COMPUTING PREDICTION ACCURACY FOR TESTING DATA*/
```

```
data predicted;
set predicted;
match=(rating=lowcase(I_rating));
run;

proc sql;
select sum(match)/count(*) as accuracy
from predicted;
quit;
```

| accuracy |
|---|
| 0.271523 |

In R:

```
movie.data<- read.csv(file="./movie_data.csv", header=TRUE, sep=",")

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
set.seed(566222)
sample <- sample(c(TRUE, FALSE), nrow(movie.data),replace=TRUE, prob=c(0.8,0.2))
train<- movie.data[sample,]
test<- movie.data[!sample,]

#BUILDING RANDOM FOREST MULTINOMIAL CLASSIFIER
library(randomForest)
rf.class<- randomForest(as.factor(rating) ~ age + gender + member + nmovies, data=train, ntree=150,
mtry=4, maxnodes=30)

#DISPLAYING FEATURE IMPORTANCE
print(importance(rf.class,type=2))
```

```
        MeanDecreaseGini
age             64.973483
gender          10.542498
member           5.531971
nmovies         25.470534
```

```
#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
predclass<- predict(rf.class, newdata=test)
```

```
test<- cbind(test,predclass)

accuracy<- c()
n<- nrow(test)
for (i in 1:n)
    accuracy[i]<- ifelse(test$rating[i]==test$predclass[i],1,0)

print(accuracy<- sum(accuracy)/length(accuracy))
```
0.3212435

In Python:

```
import pandas
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

movie_data=pandas.read_csv('C:/Users/000110888/Desktop/movie_data.csv')
code_gender={'M':1,'F':0}
code_member={'yes':1,'no':0}
code_rating={'very bad':1,'bad':2,'okay':3,'good':4,'very good':5}

movie_data['gender']=movie_data['gender'].map(code_gender)
movie_data['member']=movie_data['member'].map(code_member)
movie_data['rating']=movie_data['rating'].map(code_rating)

X=movie_data.iloc[:,0:4].values
y=movie_data.iloc[:,4].values

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.20, random_state=599555)

#FITTING RANDOM FOREST FOR MULTINOMIAL CLASSIFIER
rf_class=RandomForestClassifier(n_estimators=150, random_state=663474, max_depth=50, max_features=4)
rf_class.fit(X_train, y_train)

#DISPLAYING VARIABLE IMPORTANCE
from sklearn.ensemble import ExtraTreesClassifier

var_names=pandas.DataFrame(['age','gender','member','nmovies'], columns=['var_name'])
loss_reduction=pandas.DataFrame(rf_class.feature_importances_, columns=['loss_reduction'])
var_importance=pandas.concat([var_names, loss_reduction], axis=1)
var_importance=var_importance.sort_values("loss_reduction", axis=0, ascending=False)
print(var_importance)
```

```
  var_name  loss_reduction
0      age        0.572954
3  nmovies        0.269890
2   member        0.084660
1   gender        0.072495
```

```
#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=rf_class.predict(X_test)
y_test=pandas.DataFrame(y_test,columns=['rating'])
y_pred=pandas.DataFrame(y_pred,columns=['predicted'])
df=pandas.concat([y_test,y_pred],axis=1)

match=[]
for i in range(len(df)):
    if df['rating'][i]==df['predicted'][i]:
        match.append(1)
    else:
        match.append(0)

accuracy=sum(match)/len(match)

print(accuracy)
```

0.3026315789473684

□