STEVENS INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY

PHP Webshell detection

Huajie Xu

Yuchen zeng

Xuanzhu Luo

Yuanxin Liu

Instructor: Khasha Dehnad

CS-513

MALICIOUS CODE

introduction&background

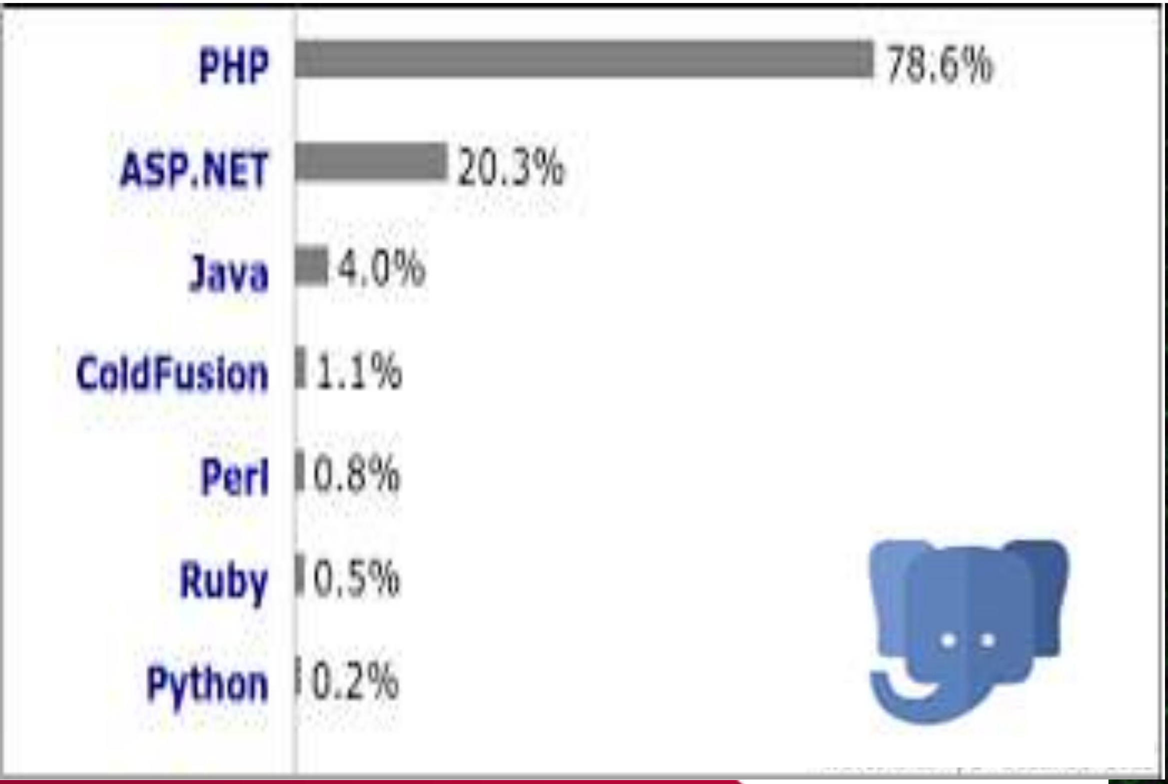
Harm of webshell

1. leave backdoor
2. access root permissions
3. monitor the network traffic

Detect PHP Webshell

PHP is the most common server-side programming language, so a lot of malicious codes are written in PHP language.

Data source:
https://www.kaggle.com/zavadskyy/lots-of-code



Preparation

The dataset we expected..

```
Age,Attrition,BusinessTravel,DistanceFromHome,Education,Emplo
NumCompaniesWorked,OverTime,TotalWorkingYears
41,Yes,Travel_Rarely,1,2,1,2,Female,Single,5993,8,Yes,8
49,No,Travel_Frequently,8,1,2,3,Male,Married,5130,1,No,10
37,Yes,Travel_Rarely,2,2,4,4,Male,Single,2090,6,Yes,7
33,No,Travel_Frequently,3,4,5,4,Female,Married,2909,1,Yes,8
27,No,Travel_Rarely,2,1,7,1,Male,Married,3468,9,No,6
```

The dataset we have…….

```php
<?php
/* Autoloader for composer/ca-bundle and its dependencies */

if (!class_exists('Fedora\\Autoloader\\Autoload', false)) {
    require_once '/usr/share/php/Fedora/Autoloader/autoload.php';
}

\Fedora\Autoloader\Autoload::addPsr4('Composer\\CaBundle\\', __DIR__);
```

But HOW?

- **File Size**
- **Longest String**
  - Malicious code is often stored as **a long string of encoded text** within a file. Many popular encoding methods, such as base64 encoding, will produce a long string without space characters.
  - Typical text and script files will be composed of relatively short length words; identifying files with uncharacteristically long strings may help to identify files with obfuscated code.
- **Entropy**
  - Measuring entropy is useful in locating encrypted shellcode. Encryption can often introduce a large amount of entropy into a text string.
- **Keywords**
  - eval, shell_exec, fwrite, chr, str_replace…...

Example

```php
<?php
/* Autoloader for composer/ca-bundle and its dependencies */

if (!class_exists('Fedora\\Autoloader\\Autoload', false)) {
    require_once '/usr/share/php/Fedora/Autoloader/autoload.php';
}

\Fedora\Autoloader\Autoload::addPsr4('Composer\\CaBundle\\', __DIR__);
```

```php
// default: linux
// linux shellcode loader (x86)
$shellcode_loader=
"f0VMRgEBAQAAAAAAAAAAAAMAAwABAAAAIAQAADQAAACIEQAAAAAAADQAIAAGACgAGwAYAAEAAAAA
AAAAAAAAAAAAAAABIBgAASAYAAAUAAAAAEAAAAQAAAAwPAAAMHwAADB8AABABAAAYAQAABgAAAAAQ
AAACAAAAIA8AACAfAAAgHwAAyAAAAMgAAAAGAAAABAAAAFHldGQAAAAAAAAAAAAAAAAAAAAAAAAA
AAYAAAAEAAAAUuV0ZAwPAAAMHwAADB8AAPQAAAD0AAAABAAAAAEAAACAFQRlAAAAAAAAAAAAAAAA
AAAAAAAAAAAKAAABAAAAAMAAAAOAAAADAAAAAcAAAAGAAAAAAAAAAAAAAAAAAAAAAgAAAAAAAAAN
AAAACwAAAAkAAAADAAAABQAAAAgAAAABAAAACgAAAAQAAAADAAAACAAAAAIAAAAGAAAAiAAhAQDE
QAkIAAAACwAAAA0AAAAGpIf/uuOSfENF1ezYcVgcuY3xDuvT7w4AAAAAAAAAAAAAAAAAAAAATwAA
AAAAAAB6AAAAEgAAAAEAAAAAAAAAAAAAAACAAAAArAAAAAAAAAAAAAAAgAAAARgAAAAAAAAD+AAAA
EgAAAFkAAAAAAAAAegAAABIAAAAcAAAAAAAAAAAsBAAAiAAAAVAAAAAAAAAAD9AAAAEgAAAD8AAAAM
BQAAvQAAABIACwB7AAAAJCAAAAAAAAAAQAPH/aAAAABwgAAAAAAAAAEADx/28AAAAcIAAAAAAAABAA
```

CSV generated

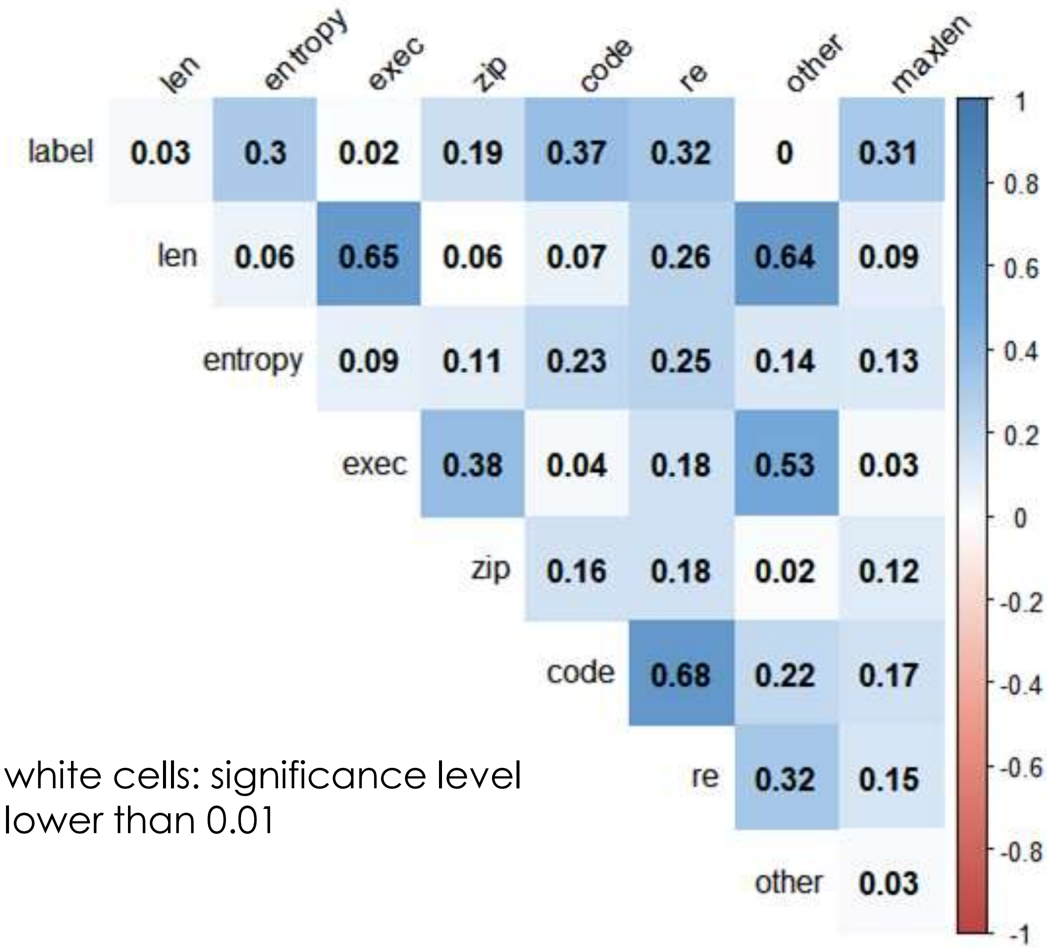
275 Webshells, 6000 Ordinary PHP codes.

```
ID,label,len,entropy,exec,zip,code,chr,re,other,maxlen
0,1,1405,5.433242758321368,4,0,0,0,0,0,26
1,1,93545,6.011169553454867,1,1,1,0,0,0,93123
2,1,37391,5.45186781406762,6,0,0,0,17,0,148
```

ID	
label	1 for webshell, 0 for normal
length	The length of the code file
count_*	Amount of sensitive functions
...	
maxlen	The longest word

correlation analysis

most relevant 3 factors with label: code, re, maxlen



white cells: significance level lower than 0.01

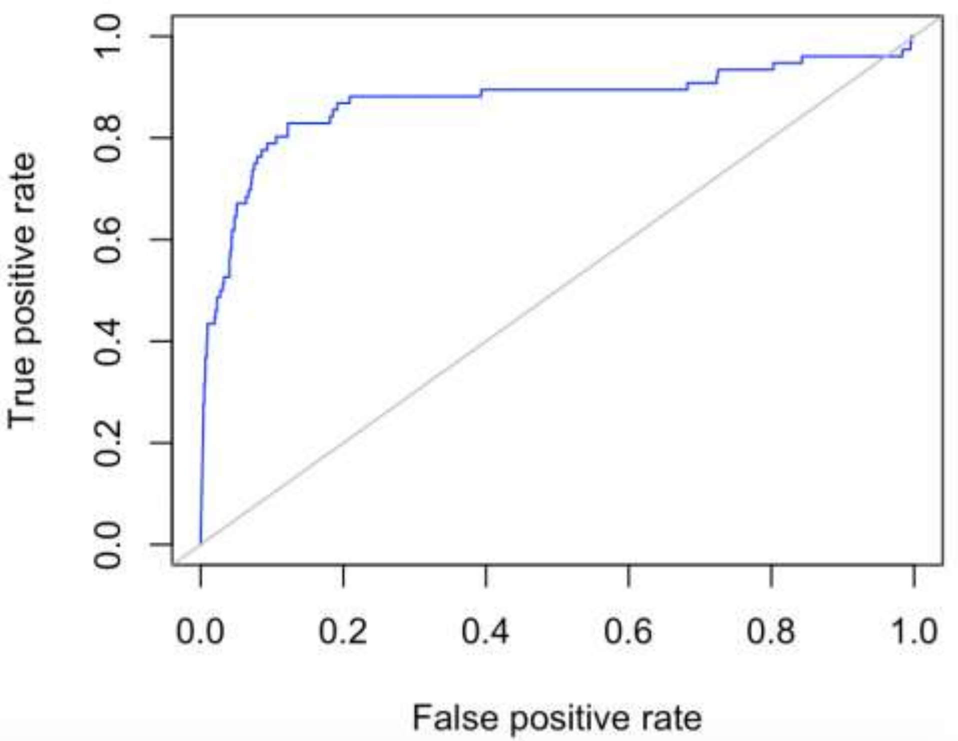
characteristics for malicious code:

1.frequent calling function: base64_encode/decode, str_replace

2.maxlen higher than normal codes

Model 1 Naive Bayes Results

```
            y_pred
y_true    Normal  Bad
  Normal    1742   33
  Bad         43   33
> Accuracy(pred, ytest)
[1] 0.9589411
> Precision(ytest, pred)
[1] 0.9759104
> Recall(ytest, pred)
[1] 0.9814085
> F1_Score(ytest, pred)
[1] 0.9786517
```



Model 2 K-Means Results

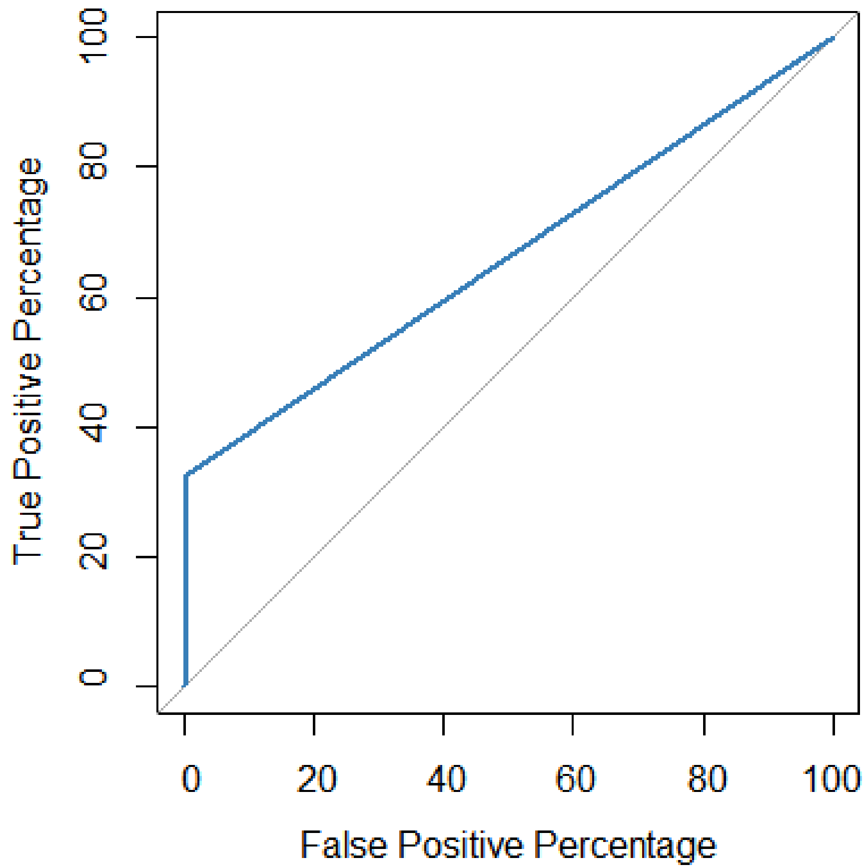
```
TP 47
FP 85
FN 228
TN 5915
Acc 0.9501195219123506
Recall 0.17090909090909
Precision 0.3560606060606061
F1 0.23095823095823095
```

This model is not suitable for our condition

Model 3 KNN

Area under the curve: 66.11%

correctRate_k5	0.965737051792829
erroRate_k5	0.0342629482071714



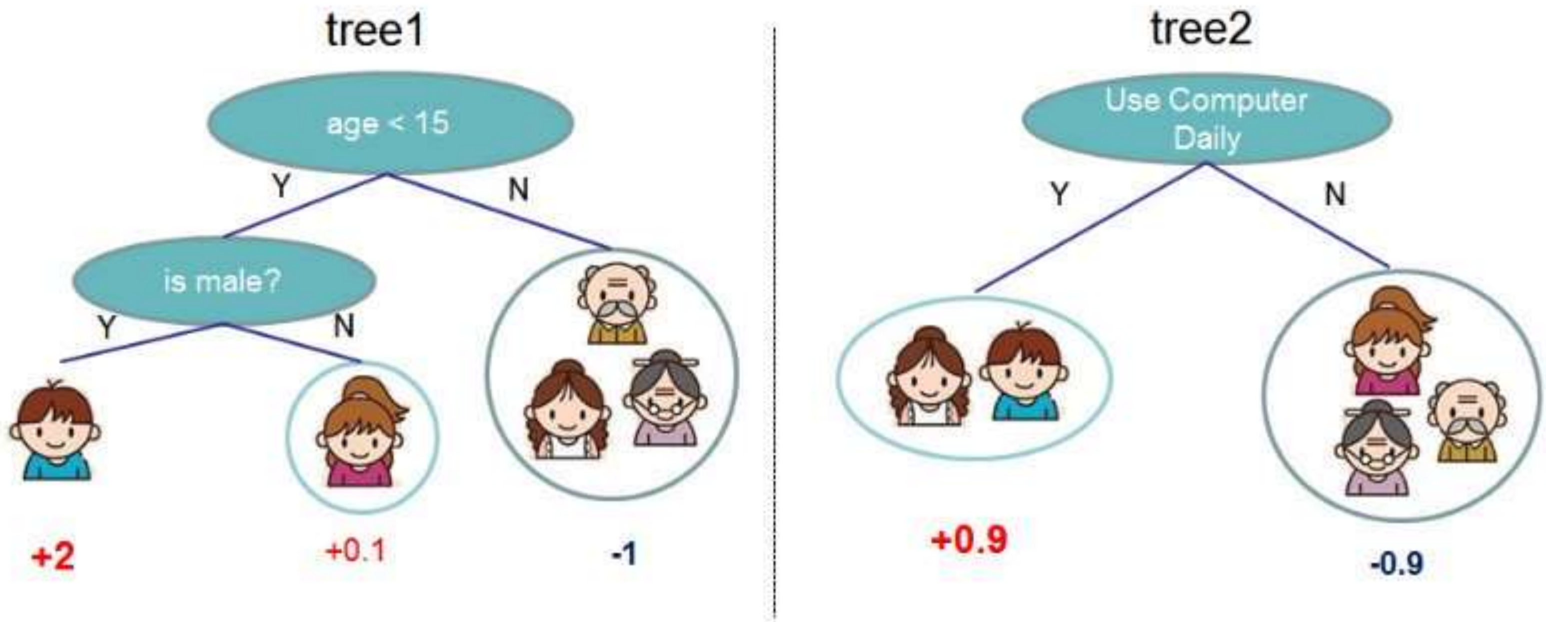

```
         y_pred
y_true     0    1
     0  1194    6
     1    37   18
> Accuracy(predict_K5, test$label)
[1] 0.9657371
> Precision(test$label, predict_K5)
[1] 0.9699431
> Recall(test$label, predict_K5)
[1] 0.995
> F1_Score(test$label, predict_K5)
[1] 0.9823118
>
```

Model 4 Gradient Boosted Trees

- Gradient boosting is a supervised learning algorithm, which attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models.

- When using gradient boosting for regression, the weak learners are regression trees, and each regression tree maps an input data point to one of its leafs that contains a continuous score.

- XGBoost minimizes a regularized (L1 and L2) objective function that combines a convex loss function (based on the difference between the predicted and target outputs) and a penalty term for model complexity (in other words, the regression tree functions). The training proceeds iteratively, adding new trees that predict the residuals or errors of prior trees that are then combined with previous trees to make the final prediction.

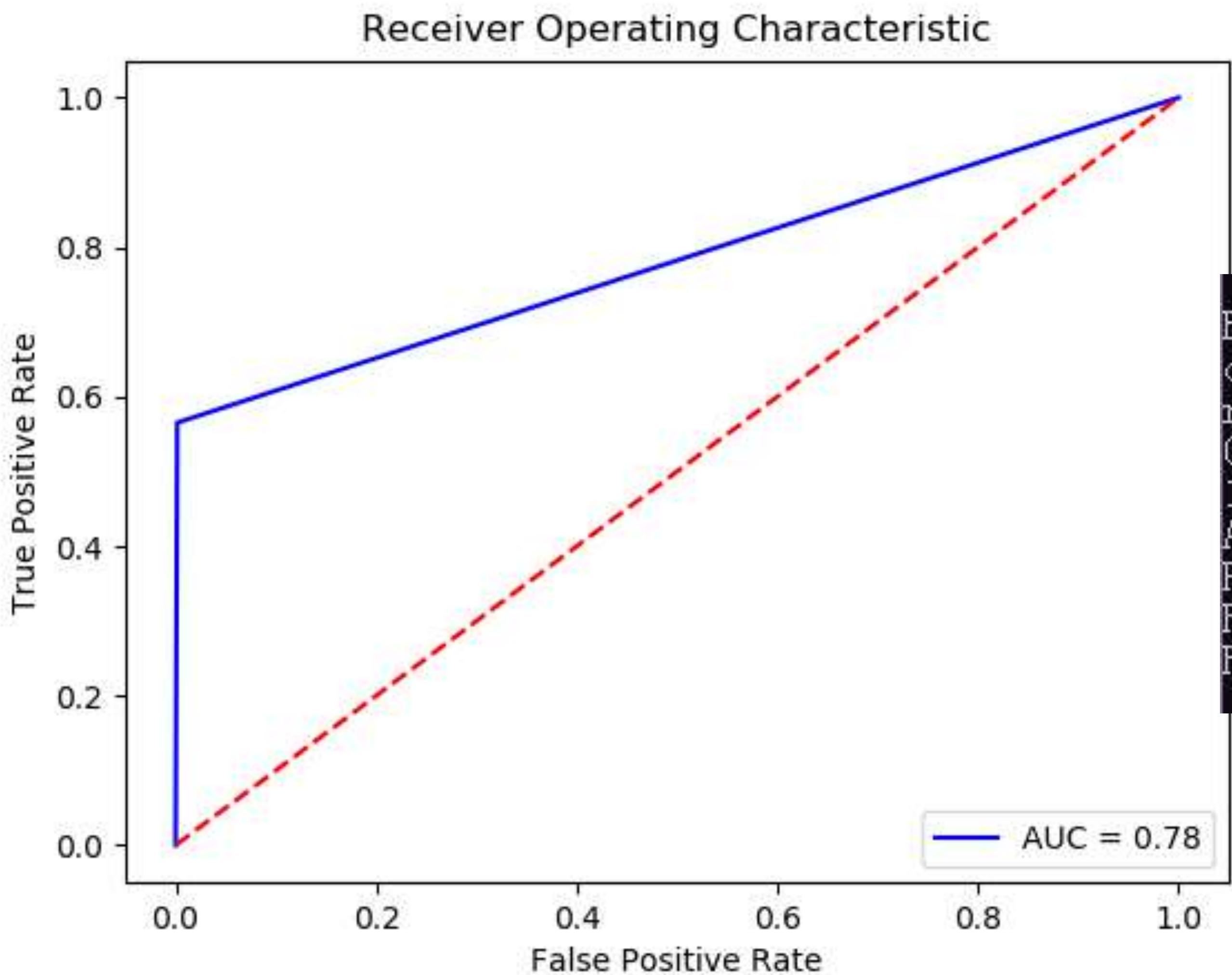


f() = 2 + 0.9= 2.9 f()= -1 - 0.9 = -1.9

Here is an example of a tree ensemble of two trees. The prediction scores of each individual tree are summed up to get the final score. If you look at the example, an important fact is that the two trees try to *complement* each other. Mathematically, we can write our model in the form

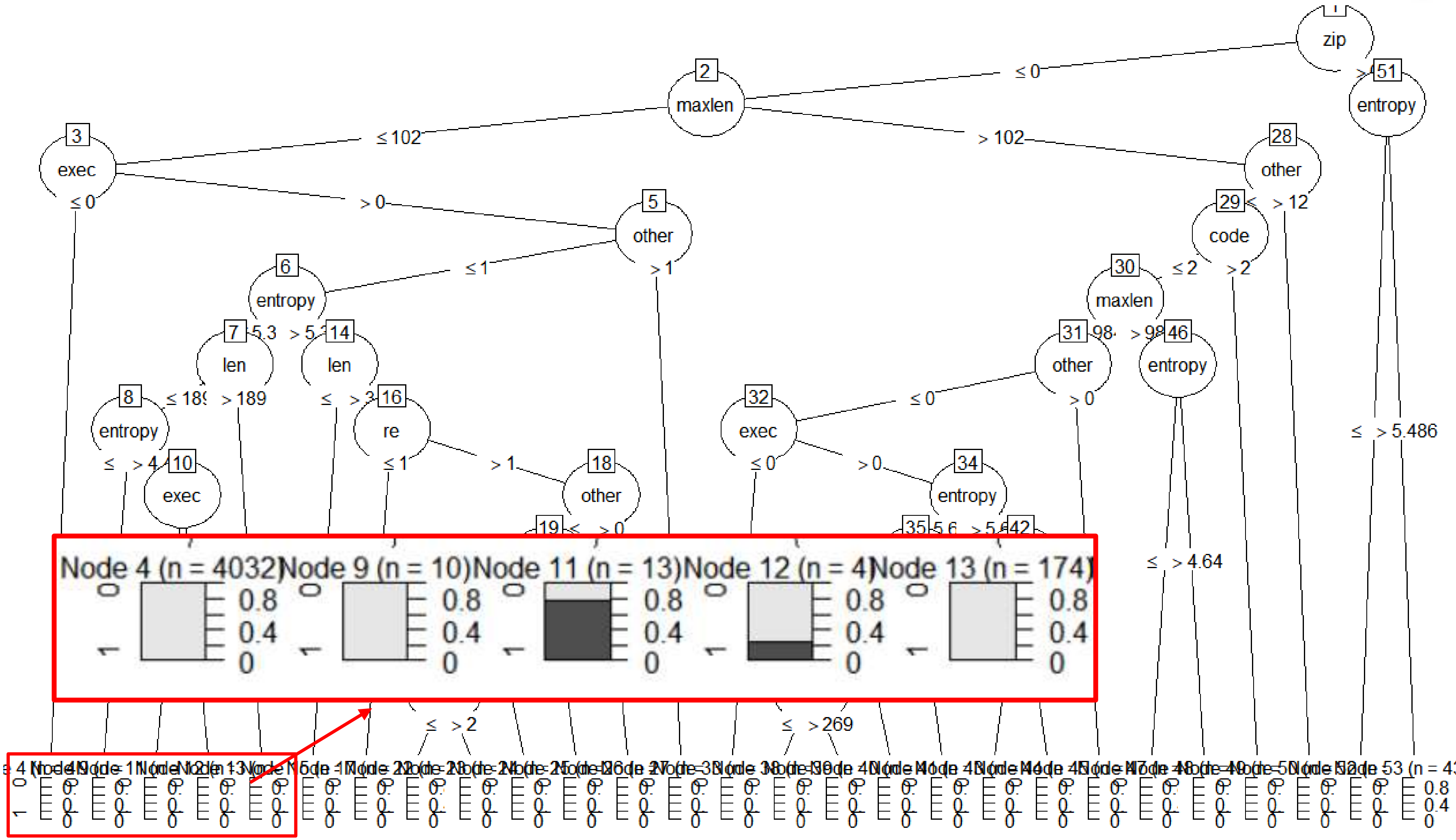
$$\hat{y}_i = \sum^{K} f_k(x_i), f_k \in \mathcal{F}$$

Results




```
E:\Work\10-CS513\Final>pytho
col_0  0.0  1.0
row_0
0.0    604    1
1.0     10   13
Accurancy:  0.98248407643312
Precision:  0.92857142857142
Recall:  0.5652173913043478
F1:  0.7027027027027025
```

Model 5 C5.0


zip
≤ 0
> 51
2
maxlen
≤ 102
> 102
3
exec
≤ 0
> 0
5
other
≤ 1
> 1
6
entropy
7
len
14
len
8
entropy
10
exec
16
re
18
other
19
28
other
> 12
29
code
≤ 2
> 2
30
maxlen
31
other
46
entropy
32
exec
34
entropy
35
42
≤ 0
> 0
> 4.64
> 5.486
> 269
> 2
Node 4 (n = 4032)
Node 9 (n = 10)
Node 11 (n = 13)
Node 12 (n = 4)
Node 13 (n = 174)
0.8
0.4
0
Node 53 (n = 43

Model 5 C5.0

Evaluation on training data (4706 cases):

```
   Decision Tree
  ----------------
  Size      Errors

    27   82( 1.7%)   <<


   (a)   (b)    <-classified as
  ----  ----
  4485    10    (a): class 0
    72   139    (b): class 1
```

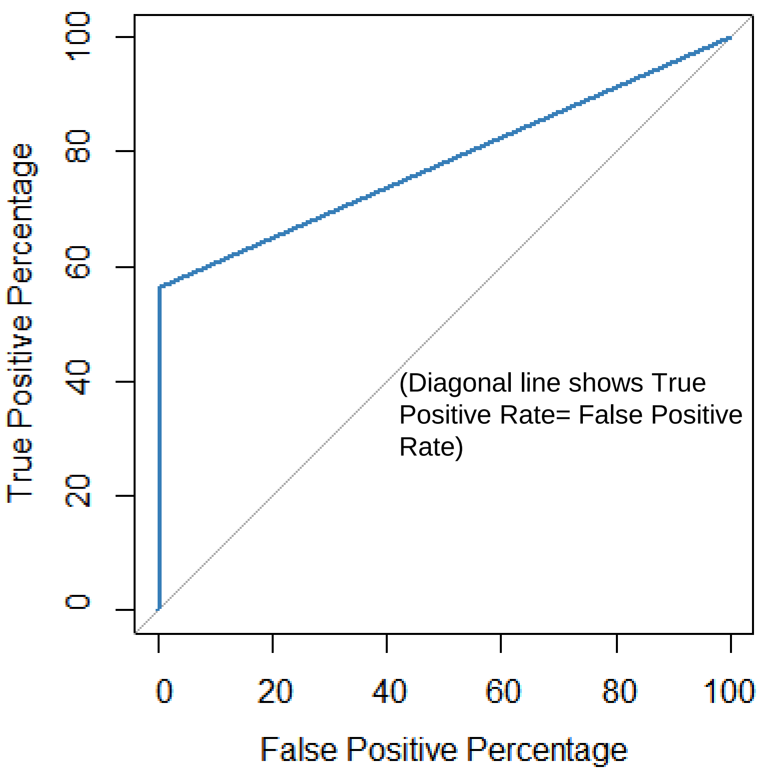
Model 5 C5.0

evaluation using test data

Area under the curve: 0.7786

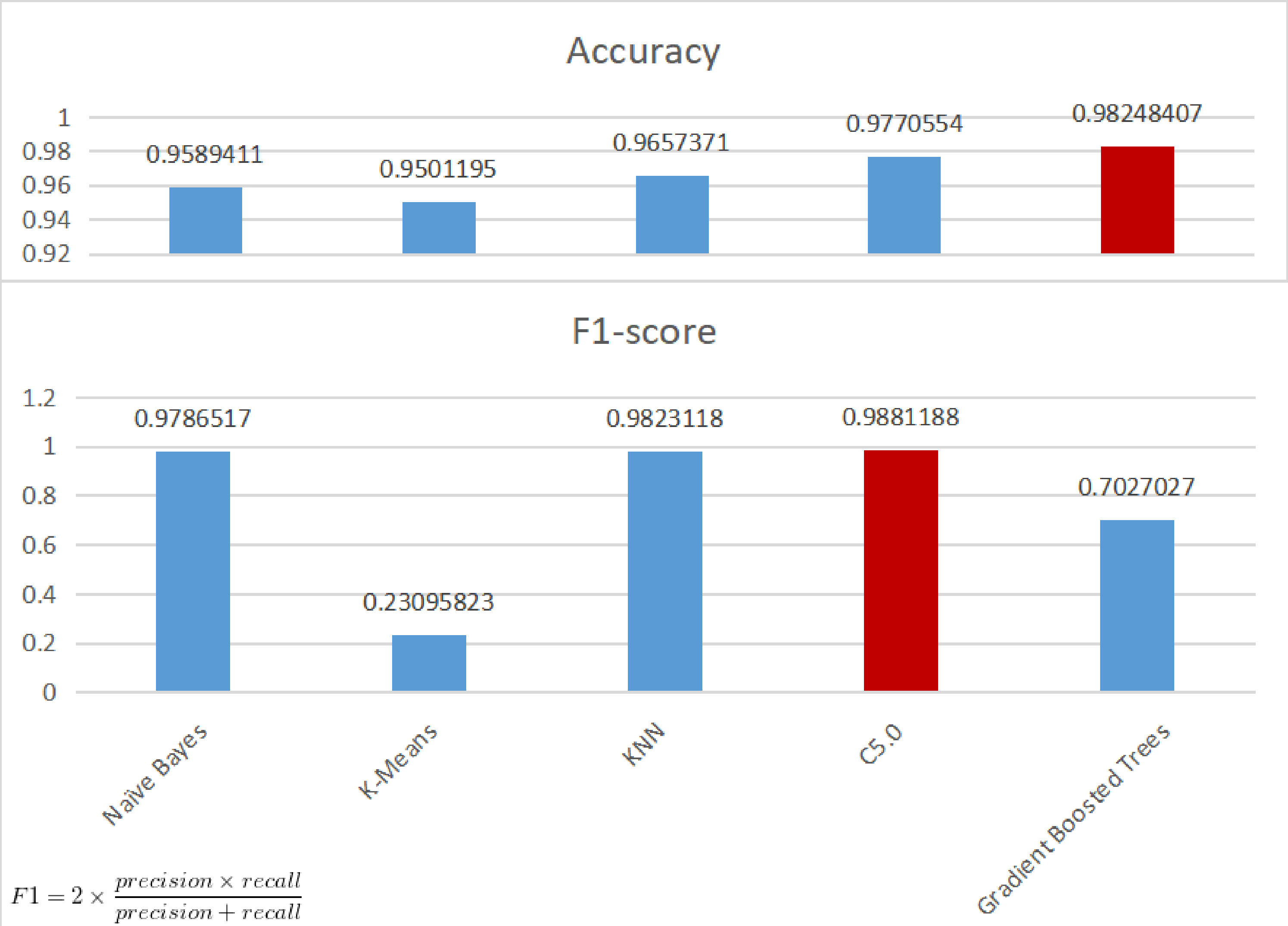
correctRate_c5.0	0.977055449330784
erroRate_c5.0	0.0229445506692161

```
       prediction
actual    0    1
     0 1497    8
     1   28   36
```




```
> Accuracy(prediction1, test1)
[1] 0.9770554
> Precision(test1,prediction1)
[1] 0.9816393
> Recall(test1,prediction1)
[1] 0.9946844
> F1_Score(test1,prediction1)
[1] 0.9881188
```

Conclusion



$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

THANK YOU!

Q&A

Instructor: Khasha Dehnad
Course: CS513 Group Project