Malware Detection Level - 2,3

1. Business Problem

1.1. Malware

A malicious software is called a malware. In simple terms, malware is a piece of software that was written with the intent of doing harm to the data or devices.

Source: https://en.wikipedia.org/wiki/Malware

1.2. Problem Statement

The daily improvised cyber crimes by computer system from a malware attack is to identify whether a given piece of files/feature is a malware.

1.3. Data Source

The dataset is provided by Microsoft which contains about 9 classes of malwares.

Source: https://www.kaggle.com/c/malware-classification

1.4. Objectives & Constraints

1. Minimize loss error.
2. Multi-class probability estimates.
3. Interpretable because we need know and block the user's computer. It should threat a new executable or a binary.

2. Machine Learning Problem

2.1. Data Overview

- We have the files in zip every malware.
- Train file size is 50 GB.

- Test size is 15 GB. The data contains the foundational representation of the files binary (unless without the PE header).
- Train size between around 400MB files and 2 Gigs, in total files are 21GB of size in .asm files
- There are total 10K .byte files and 1TB size files train 5GB files.
- Test size is 15 GB. Test contains 10 classes of malwares in .byte given data.

- Type of Malwares are
  1. Ramnit
  2. Lollipop
  3. Kelihos_ver3
  4. Vundo
  5. Simda
  6. Tracur
  7. Kelihos_ver1
  8. Obfuscator.ACY
  9. Gatak

2.2. Data Files Structure

.asm file

```
00401000 55 8B EC 83 EC 50 53 56 57 8B 7D 08 8B F1 83 3E 00 0F 84 ACCCCCCC
00401010 00 00 00 00 8B 46 04 2D 00 10 00 00 3B 07 0F 84 9A CCCCCCCC
00401020 00 00 00 00 8B 4E 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 8A
00401030 CCCCCCCC 8B 56 0C 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 7A CCCCCCCC
00401040 8B 45 0C 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 6A CCCCCCCC
00401050 8B 4D 10 8B 11 8B 42 0C 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F
00401060 0F 84 4A CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74
00401070 3A 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 2A CCCCCCCC
00401080 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F
00401090 84 0A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84
004010A0 FA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 EA
004010B0 CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 DA
004010C0 CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02
004010D0 3B 42 0F 0F 84 BA CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02
004010E0 3B 0F 0F 84 AA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B
004010F0 57 0F 74 9A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47
00401100 0F 74 8A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00
00401110 C1 F8 02 3B 42 0F 0F 84 7A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00
00401120 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1
00401130 FA 02 3B 57 0F 74 5A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8
00401140 02 3B 47 0F 74 4A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00
00401150 00 40 00 C1 F8 02 3B 42 0F 0F 84 3A CCCCCCCC 8B 4D 08 2B 0D 00
00401160 00 40 00 C1 F9 02 3B 0F 0F 84 2A CCCCCCCC 8B 56 08 2B 15 00 00
00401170 40 00 C1 FA 02 3B 57 0F 74 1A CCCCCCCC 8B 45 08 2B 05 00 00 40
00401180 00 C1 F8 02 3B 47 0F 74 0A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08
00401190 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 FA CCCCCCCC 8B 4D 08
004011A0 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 EA CCCCCCCC 8B 56 08 2B
004011B0 15 00 00 40 00 C1 FA 02 3B 57 0F 74 DA CCCCCCCC 8B 45 08 2B 05
004011C0 00 00 40 00 C1 F8 02 3B 47 0F 74 CA CCCCCCCC 8B 4D 08 8B 51 0C
004011D0 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 BA CCCCCCCC
004011E0 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 AA CCCCCCCC 8B
004011F0 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 9A CCCCCCCC 8B 45
00401200 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 8A CCCCCCCC 8B 4D 08
00401210 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 7A
00401220 CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC
00401230 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 5A CCCCCCCC 8B
00401240 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 4A CCCCCCCC 8B 4D
00401250 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84
00401260 3A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 2A
00401270 CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 1A CCCCCCCC
00401280 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 0A CCCCCCCC 8B
00401290 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F
004012A0 84 FA CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84
004012B0 EA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 DA
004012C0 CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 CA
004012D0 CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02
004012E0 3B 42 0F 0F 84 BA CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02
004012F0 3B 0F 0F 84 AA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B
00401300 57 0F 74 9A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47
00401310 0F 74 8A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00
00401320 C1 F8 02 3B 42 0F 0F 84 7A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00
00401330 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1
00401340 FA 02 3B 57 0F 74 5A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8
00401350 02 3B 47 0F 74 4A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00
00401360 00 40 00 C1 F8 02 3B 42 0F 0F 84 3A CCCCCCCC 8B 4D 08 2B 0D 00
00401370 00 40 00 C1 F9 02 3B 0F 0F 84 2A CCCCCCCC 8B 56 08 2B 15 00 00
00401380 40 00 C1 FA 02 3B 57 0F 74 1A CCCCCCCC 8B 45 08 2B 05 00 00 40
00401390 00 C1 F8 02 3B 47 0F 74 0A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08
004013A0 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 FA CCCCCCCC 8B 4D 08
004013B0 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 EA CCCCCCCC 8B 56 08 2B
004013C0 15 00 00 40 00 C1 FA 02 3B 57 0F 74 DA CCCCCCCC 8B 45 08 2B 05
004013D0 00 00 40 00 C1 F8 02 3B 47 0F 74 CA CCCCCCCC 8B 4D 08 8B 51 0C
004013E0 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 BA CCCCCCCC
004013F0 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 AA CCCCCCCC 8B
00401400 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 9A CCCCCCCC 8B 45
00401410 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 8A CCCCCCCC 8B 4D 08
00401420 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 7A
00401430 CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC
00401440 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 5A CCCCCCCC 8B
00401450 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 4A CCCCCCCC 8B 4D
00401460 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84
00401470 3A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 2A
00401480 CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 1A CCCCCCCC
00401490 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 0A CCCCCCCC 8B
004014A0 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F
004014B0 84 FA CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84
004014C0 EA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 DA
004014D0 CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 CA
004014E0 CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02
004014F0 3B 42 0F 0F 84 BA CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02
00401500 3B 0F 0F 84 AA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B
00401510 57 0F 74 9A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47
00401520 0F 74 8A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00
00401530 C1 F8 02 3B 42 0F 0F 84 7A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00
00401540 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1
00401550 FA 02 3B 57 0F 74 5A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8
00401560 02 3B 47 0F 74 4A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00
00401570 00 40 00 C1 F8 02 3B 42 0F 0F 84 3A CCCCCCCC 8B 4D 08 2B 0D 00
00401580 00 40 00 C1 F9 02 3B 0F 0F 84 2A CCCCCCCC 8B 56 08 2B 15 00 00
00401590 40 00 C1 FA 02 3B 57 0F 74 1A CCCCCCCC 8B 45 08 2B 05 00 00 40
004015A0 00 C1 F8 02 3B 47 0F 74 0A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08
004015B0 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 FA CCCCCCCC 8B 4D 08
004015C0 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 EA CCCCCCCC 8B 56 08 2B
004015D0 15 00 00 40 00 C1 FA 02 3B 57 0F 74 DA CCCCCCCC 8B 45 08 2B 05
004015E0 00 00 40 00 C1 F8 02 3B 47 0F 74 CA CCCCCCCC 8B 4D 08 8B 51 0C
004015F0 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 BA CCCCCCCC
00401600 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 AA CCCCCCCC 8B
00401610 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 9A CCCCCCCC 8B 45
00401620 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 8A CCCCCCCC 8B 4D 08
00401630 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 7A
00401640 CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC
00401650 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 5A CCCCCCCC 8B
00401660 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 4A CCCCCCCC 8B 4D
00401670 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84
00401680 3A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 2A
00401690 CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 1A CCCCCCCC
004016A0 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 0A CCCCCCCC 8B
004016B0 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F
004016C0 84 FA CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84
004016D0 EA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 DA
004016E0 CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 CA
004016F0 CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02
00401700 3B 42 0F 0F 84 BA CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02
00401710 3B 0F 0F 84 AA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B
00401720 57 0F 74 9A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47
00401730 0F 74 8A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00
00401740 C1 F8 02 3B 42 0F 0F 84 7A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00
00401750 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1
00401760 FA 02 3B 57 0F 74 5A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8
00401770 02 3B 47 0F 74 4A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00
00401780 00 40 00 C1 F8 02 3B 42 0F 0F 84 3A CCCCCCCC 8B 4D 08 2B 0D 00
00401790 00 40 00 C1 F9 02 3B 0F 0F 84 2A CCCCCCCC 8B 56 08 2B 15 00 00
004017A0 40 00 C1 FA 02 3B 57 0F 74 1A CCCCCCCC 8B 45 08 2B 05 00 00 40
004017B0 00 C1 F8 02 3B 47 0F 74 0A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08
004017C0 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 FA CCCCCCCC 8B 4D 08
004017D0 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 EA CCCCCCCC 8B 56 08 2B
004017E0 15 00 00 40 00 C1 FA 02 3B 57 0F 74 DA CCCCCCCC 8B 45 08 2B 05
004017F0 00 00 40 00 C1 F8 02 3B 47 0F 74 CA CCCCCCCC 8B 4D 08 8B 51 0C
00401800 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 BA CCCCCCCC
00401810 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 AA CCCCCCCC 8B
00401820 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 9A CCCCCCCC 8B 45
00401830 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 8A CCCCCCCC 8B 4D 08
00401840 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 7A
00401850 CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC
00401860 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 5A CCCCCCCC 8B
00401870 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 4A CCCCCCCC 8B 4D
00401880 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84
00401890 3A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 2A
004018A0 CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 1A CCCCCCCC
004018B0 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 0A CCCCCCCC 8B
004018C0 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F
004018D0 84 FA CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84
004018E0 EA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 DA
004018F0 CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 CA
00401900 CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02
00401910 3B 42 0F 0F 84 BA CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02
00401920 3B 0F 0F 84 AA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B
00401930 57 0F 74 9A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47
00401940 0F 74 8A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00
00401950 C1 F8 02 3B 42 0F 0F 84 7A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00
00401960 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1
00401970 FA 02 3B 57 0F 74 5A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8
00401980 02 3B 47 0F 74 4A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00
00401990 00 40 00 C1 F8 02 3B 42 0F 0F 84 3A CCCCCCCC 8B 4D 08 2B 0D 00
004019A0 00 40 00 C1 F9 02 3B 0F 0F 84 2A CCCCCCCC 8B 56 08 2B 15 00 00
004019B0 40 00 C1 FA 02 3B 57 0F 74 1A CCCCCCCC 8B 45 08 2B 05 00 00 40
004019C0 00 C1 F8 02 3B 47 0F 74 0A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08
004019D0 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 FA CCCCCCCC 8B 4D 08
004019E0 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 EA CCCCCCCC 8B 56 08 2B
004019F0 15 00 00 40 00 C1 FA 02 3B 57 0F 74 DA CCCCCCCC 8B 45 08 2B 05
00401A00 00 00 40 00 C1 F8 02 3B 47 0F 74 CA CCCCCCCC 8B 4D 08 8B 51 0C
00401A10 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 BA CCCCCCCC
00401A20 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 AA CCCCCCCC 8B
00401A30 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 9A CCCCCCCC 8B 45
00401A40 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 8A CCCCCCCC 8B 4D 08
00401A50 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 7A
00401A60 CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC
00401A70 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 5A CCCCCCCC 8B
00401A80 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 4A CCCCCCCC 8B 4D
00401A90 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84
00401AA0 3A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 2A
00401AB0 CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 1A CCCCCCCC
00401AC0 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 0A CCCCCCCC 8B
00401AD0 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F
00401AE0 84 FA CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84
00401AF0 EA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 DA
00401B00 CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 CA
00401B10 CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02
00401B20 3B 42 0F 0F 84 BA CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02
00401B30 3B 0F 0F 84 AA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B
00401B40 57 0F 74 9A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47
00401B50 0F 74 8A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00
00401B60 C1 F8 02 3B 42 0F 0F 84 7A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00
00401B70 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1
00401B80 FA 02 3B 57 0F 74 5A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8
00401B90 02 3B 47 0F 74 4A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00
00401BA0 00 40 00 C1 F8 02 3B 42 0F 0F 84 3A CCCCCCCC 8B 4D 08 2B 0D 00
00401BB0 00 40 00 C1 F9 02 3B 0F 0F 84 2A CCCCCCCC 8B 56 08 2B 15 00 00
00401BC0 40 00 C1 FA 02 3B 57 0F 74 1A CCCCCCCC 8B 45 08 2B 05 00 00 40
00401BD0 00 C1 F8 02 3B 47 0F 74 0A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08
00401BE0 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 FA CCCCCCCC 8B 4D 08
00401BF0 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 EA CCCCCCCC 8B 56 08 2B
00401C00 15 00 00 40 00 C1 FA 02 3B 57 0F 74 DA CCCCCCCC 8B 45 08 2B 05
00401C10 00 00 40 00 C1 F8 02 3B 47 0F 74 CA CCCCCCCC 8B 4D 08 8B 51 0C
00401C20 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 BA CCCCCCCC
00401C30 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 AA CCCCCCCC 8B
00401C40 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 9A CCCCCCCC 8B 45
00401C50 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 8A CCCCCCCC 8B 4D 08
00401C60 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 7A
00401C70 CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC
00401C80 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 5A CCCCCCCC 8B
00401C90 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 4A CCCCCCCC 8B 4D
00401CA0 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84
00401CB0 3A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 2A
00401CC0 CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 1A CCCCCCCC
00401CD0 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 0A CCCCCCCC 8B
00401CE0 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F
00401CF0 84 FA CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84
00401D00 EA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 DA
00401D10 CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 CA
00401D20 CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02
00401D30 3B 42 0F 0F 84 BA CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02
00401D40 3B 0F 0F 84 AA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B
00401D50 57 0F 74 9A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47
00401D60 0F 74 8A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00
00401D70 C1 F8 02 3B 42 0F 0F 84 7A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00
00401D80 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1
00401D90 FA 02 3B 57 0F 74 5A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8
00401DA0 02 3B 47 0F 74 4A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00
00401DB0 00 40 00 C1 F8 02 3B 42 0F 0F 84 3A CCCCCCCC 8B 4D 08 2B 0D 00
00401DC0 00 40 00 C1 F9 02 3B 0F 0F 84 2A CCCCCCCC 8B 56 08 2B 15 00 00
00401DD0 40 00 C1 FA 02 3B 57 0F 74 1A CCCCCCCC 8B 45 08 2B 05 00 00 40
00401DE0 00 C1 F8 02 3B 47 0F 74 0A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08
00401DF0 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 FA CCCCCCCC 8B 4D 08
00401E00 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 EA CCCCCCCC 8B 56 08 2B
00401E10 15 00 00 40 00 C1 FA 02 3B 57 0F 74 DA CCCCCCCC 8B 45 08 2B 05
00401E20 00 00 40 00 C1 F8 02 3B 47 0F 74 CA CCCCCCCC 8B 4D 08 8B 51 0C
00401E30 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 BA CCCCCCCC
00401E40 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 AA CCCCCCCC 8B
00401E50 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 9A CCCCCCCC 8B 45
00401E60 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 8A CCCCCCCC 8B 4D 08
00401E70 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 7A
00401E80 CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC
00401E90 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 5A CCCCCCCC 8B
00401EA0 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 4A CCCCCCCC 8B 4D
00401EB0 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84
00401EC0 3A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 2A
00401ED0 CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 1A CCCCCCCC
00401EE0 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 0A CCCCCCCC 8B
00401EF0 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F
00401F00 84 FA CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84
00401F10 EA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 DA
00401F20 CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 CA
00401F30 CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02
00401F40 3B 42 0F 0F 84 BA CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02
00401F50 3B 0F 0F 84 AA CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B
00401F60 57 0F 74 9A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 47
00401F70 0F 74 8A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00 00 40 00
00401F80 C1 F8 02 3B 42 0F 0F 84 7A CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00
00401F90 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC 8B 56 08 2B 15 00 00 40 00 C1
00401FA0 FA 02 3B 57 0F 74 5A CCCCCCCC 8B 45 08 2B 05 00 00 40 00 C1 F8
00401FB0 02 3B 47 0F 74 4A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08 2B 05 00
00401FC0 00 40 00 C1 F8 02 3B 42 0F 0F 84 3A CCCCCCCC 8B 4D 08 2B 0D 00
00401FD0 00 40 00 C1 F9 02 3B 0F 0F 84 2A CCCCCCCC 8B 56 08 2B 15 00 00
00401FE0 40 00 C1 FA 02 3B 57 0F 74 1A CCCCCCCC 8B 45 08 2B 05 00 00 40
00401FF0 00 C1 F8 02 3B 47 0F 74 0A CCCCCCCC 8B 4D 08 8B 51 0C 8B 45 08
00402000 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 FA CCCCCCCC 8B 4D 08
00402010 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 EA CCCCCCCC 8B 56 08 2B
00402020 15 00 00 40 00 C1 FA 02 3B 57 0F 74 DA CCCCCCCC 8B 45 08 2B 05
00402030 00 00 40 00 C1 F8 02 3B 47 0F 74 CA CCCCCCCC 8B 4D 08 8B 51 0C
00402040 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 BA CCCCCCCC
00402050 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 AA CCCCCCCC 8B
00402060 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74 9A CCCCCCCC 8B 45
00402070 08 2B 05 00 00 40 00 C1 F8 02 3B 47 0F 74 8A CCCCCCCC 8B 4D 08
00402080 8B 51 0C 8B 45 08 2B 05 00 00 40 00 C1 F8 02 3B 42 0F 0F 84 7A
00402090 CCCCCCCC 8B 4D 08 2B 0D 00 00 40 00 C1 F9 02 3B 0F 0F 84 6A CCCCCCCC
004020A0 8B 56 08 2B 15 00 00 40 00 C1 FA 02 3B 57 0F 74
```

```
In [48]: ## Hyperparameter tuning for Decision Tree
## Hyperparameter tuning for Decision Tree
DTHyperparameters = {'criterion': ['gini', 'entropy'],
                     'max_depth': [None, 10, 20, 30, 40, 50],
                     'min_samples_split': [2, 5, 10],
                     'min_samples_leaf': [1, 2, 4],
                     'max_features': ['sqrt', 'log2', None],
                     'max_leaf_nodes': [None, 10, 20, 30, 40, 50],
                     'min_impurity_decrease': [0.0, 0.1, 0.2],
                     'ccp_alpha': [0.0, 0.1, 0.2]}

# Initialize GridSearchCV for Decision Tree
dt = DecisionTreeClassifier(random_state=42)
grid_search = GridSearchCV(estimator=dt, param_grid=DTHyperparameters, cv=5, n_jobs=-1, verbose=1)

# Fit GridSearchCV to the training data
grid_search.fit(X_train, y_train)

# Get the best parameters and the best score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

# Print the best parameters and the best score
print("Best parameters: ", best_params)
print("Best score: ", best_score)

# Get the best estimator
best_dt = grid_search.best_estimator_

# Predict on the test set using the best estimator
y_pred = best_dt.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)

# Calculate and print the classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Calculate and print the confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Fitting 5 folds for each of 1296 candidates, totalling 6480 fits
Best parameters:  {'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': None, 'max_features': None, 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2}
Best score:  0.9666666666666666
Accuracy:  0.9666666666666666
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.90      0.95        10
           2       0.91      1.00      0.95        10

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
Confusion Matrix:
[[10  0  0]
 [ 0  9  1]
 [ 0  0 10]]
```

```
In [49]: ## Cross Validation score for Decision Tree
cvs = cross_val_score(best_dt, X, y, cv=5)
print(cvs)
print(np.mean(cvs))
print(np.median(cvs))
```

```
[1. 1. 1. 1. 1.]
1.0
1.0
```

```
In [50]: ## Cross validation prediction
cvp = cross_val_predict(best_dt, X, y, cv=5)
print(cvp)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

```
In [51]: ## Confusion matrix
cm = confusion_matrix(y, cvp)
print(cm)
```

```
[[50  0  0]
 [ 0 47  3]
 [ 0  0 50]]
```

```
In [52]: ## Plot the confusion matrix
plt.figure(figure_size=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
In [53]: ## Cross validation score for cat data
cvs = cross_val_score(best_dt, X, y, cv=5)
print(cvs)
print(np.mean(cvs))
print(np.median(cvs))
```

```
[1. 1. 1. 1. 1.]
1.0
1.0
```

```
In [54]: ## Cross validation prediction
cvp = cross_val_predict(best_dt, X, y, cv=5)
print(cvp)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

```
In [55]: ## Confusion matrix
cm = confusion_matrix(y, cvp)
print(cm)
```

```
[[50  0  0]
 [ 0 47  3]
 [ 0  0 50]]
```

```
In [56]: ## Plot the confusion matrix
plt.figure(figure_size=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

5.1.5. Logistic Regression

```
In [57]: ## Hyperparameter tuning for Logistic Regression
## Hyperparameter tuning for Logistic Regression
LRHyperparameters = {'penalty': ['l1', 'l2', 'elasticnet', None],
                     'dual': [True, False],
                     'tol': [1e-4, 1e-5, 1e-6],
                     'C': [1.0, 0.1, 0.01],
                     'fit_intercept': [True, False],
                     'intercept_scaling': [1, 0.1, 0.01],
                     'class_weight': [None, 'balanced'],
                     'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
                     'max_iter': [100, 200, 300],
                     'multi_class': ['auto', 'ovr', 'multinomial'],
                     'verbose': [0, 1, 2],
                     'warm_start': [True, False],
                     'n_jobs': [None, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                     'l1_ratio': [None, 0.1, 0.2]}

# Initialize GridSearchCV for Logistic Regression
lr = LogisticRegression(random_state=42)
grid_search = GridSearchCV(estimator=lr, param_grid=LRHyperparameters, cv=5, n_jobs=-1, verbose=1)

# Fit GridSearchCV to the training data
grid_search.fit(X_train, y_train)

# Get the best parameters and the best score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

# Print the best parameters and the best score
print("Best parameters: ", best_params)
print("Best score: ", best_score)

# Get the best estimator
best_lr = grid_search.best_estimator_

# Predict on the test set using the best estimator
y_pred = best_lr.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)

# Calculate and print the classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Calculate and print the confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Fitting 5 folds for each of 1296 candidates, totalling 6480 fits
Best parameters:  {'C': 1.0, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercept_scaling': 1, 'l1_ratio': None, 'max_iter': 100, 'multi_class': 'auto', 'n_jobs': None, 'penalty': 'l2', 'solver': 'newton-cg', 'tol': 1e-4, 'verbose': 0, 'warm_start': False}
Best score:  0.9666666666666666
Accuracy:  0.9666666666666666
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.90      0.95        10
           2       0.91      1.00      0.95        10

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
Confusion Matrix:
[[10  0  0]
 [ 0  9  1]
 [ 0  0 10]]
```

```
In [58]: ## Cross Validation Score for cat data
cvs = cross_val_score(best_lr, X, y, cv=5)
print(cvs)
print(np.mean(cvs))
print(np.median(cvs))
```

```
[1. 1. 1. 1. 1.]
1.0
1.0
```

```
In [59]: ## Cross validation prediction
cvp = cross_val_predict(best_lr, X, y, cv=5)
print(cvp)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

```
In [60]: ## Confusion matrix
cm = confusion_matrix(y, cvp)
print(cm)
```

```
[[50  0  0]
 [ 0 47  3]
 [ 0  0 50]]
```

```
In [61]: ## Plot the confusion matrix
plt.figure(figure_size=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

5.1.6. Random Forest

```
In [62]: ## Hyperparameter tuning for Random Forest
## Hyperparameter tuning for Random Forest
RFHyperparameters = {'n_estimators': [100, 200, 300],
                     'criterion': ['gini', 'entropy'],
                     'max_depth': [None, 10, 20, 30, 40, 50],
                     'min_samples_split': [2, 5, 10],
                     'min_samples_leaf': [1, 2, 4],
                     'max_features': ['sqrt', 'log2', None],
                     'max_leaf_nodes': [None, 10, 20, 30, 40, 50],
                     'min_impurity_decrease': [0.0, 0.1, 0.2],
                     'bootstrap': [True, False],
                     'oob_score': [True, False],
                     'n_jobs': [None, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                     'ccp_alpha': [0.0, 0.1, 0.2],
                     'max_samples': [None, 0.1, 0.2]}

# Initialize GridSearchCV for Random Forest
rf = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(estimator=rf, param_grid=RFHyperparameters, cv=5, n_jobs=-1, verbose=1)

# Fit GridSearchCV to the training data
grid_search.fit(X_train, y_train)

# Get the best parameters and the best score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

# Print the best parameters and the best score
print("Best parameters: ", best_params)
print("Best score: ", best_score)

# Get the best estimator
best_rf = grid_search.best_estimator_

# Predict on the test set using the best estimator
y_pred = best_rf.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)

# Calculate and print the classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Calculate and print the confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Fitting 5 folds for each of 1296 candidates, totalling 6480 fits
Best parameters:  {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': None, 'max_features': None, 'max_leaf_nodes': None, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100, 'n_jobs': None, 'oob_score': False}
Best score:  0.9666666666666666
Accuracy:  0.9666666666666666
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.90      0.95        10
           2       0.91      1.00      0.95        10

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
Confusion Matrix:
[[10  0  0]
 [ 0  9  1]
 [ 0  0 10]]
```

```
In [63]: ## Cross Validation Score for cat data
cvs = cross_val_score(best_rf, X, y, cv=5)
print(cvs)
print(np.mean(cvs))
print(np.median(cvs))
```

```
[1. 1. 1. 1. 1.]
1.0
1.0
```

```
In [64]: ## Cross validation prediction
cvp = cross_val_predict(best_rf, X, y, cv=5)
print(cvp)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

```
In [65]: ## Confusion matrix
cm = confusion_matrix(y, cvp)
print(cm)
```

```
[[50  0  0]
 [ 0 47  3]
 [ 0  0 50]]
```

```
In [66]: ## Plot the confusion matrix
plt.figure(figure_size=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

5.1.7. AdaBoost Model

```
In [67]: ## Hyperparameter tuning for AdaBoost Classifier
## Hyperparameter tuning for AdaBoost Classifier
ABHyperparameters = {'base_estimator': [None, DecisionTreeClassifier(max_depth=1), DecisionTreeClassifier(max_depth=2), DecisionTreeClassifier(max_depth=3)],
                     'n_estimators': [50, 100, 200],
                     'learning_rate': [1.0, 0.1, 0.01],
                     'algorithm': ['SAMME', 'SAMME.R'],
                     'random_state': [None, 42],
                     'base_estimator__criterion': ['gini', 'entropy'],
                     'base_estimator__splitter': ['best', 'random'],
                     'base_estimator__max_depth': [None, 10, 20, 30, 40, 50],
                     'base_estimator__min_samples_split': [2, 5, 10],
                     'base_estimator__min_samples_leaf': [1, 2, 4],
                     'base_estimator__max_features': ['sqrt', 'log2', None],
                     'base_estimator__max_leaf_nodes': [None, 10, 20, 30, 40, 50],
                     'base_estimator__min_impurity_decrease': [0.0, 0.1, 0.2],
                     'base_estimator__ccp_alpha': [0.0, 0.1, 0.2]}

# Initialize GridSearchCV for AdaBoost Classifier
ab = AdaBoostClassifier(random_state=42)
grid_search = GridSearchCV(estimator=ab, param_grid=ABHyperparameters, cv=5, n_jobs=-1, verbose=1)

# Fit GridSearchCV to the training data
grid_search.fit(X_train, y_train)

# Get the best parameters and the best score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

# Print the best parameters and the best score
print("Best parameters: ", best_params)
print("Best score: ", best_score)

# Get the best estimator
best_ab = grid_search.best_estimator_

# Predict on the test set using the best estimator
y_pred = best_ab.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)

# Calculate and print the classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Calculate and print the confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Fitting 5 folds for each of 1296 candidates, totalling 6480 fits
Best parameters:  {'base_estimator': None, 'base_estimator__ccp_alpha': 0.0, 'base_estimator__criterion': 'gini', 'base_estimator__max_depth': None, 'base_estimator__max_features': None, 'base_estimator__max_leaf_nodes': None, 'base_estimator__min_impurity_decrease': 0.0, 'base_estimator__min_samples_leaf': 1, 'base_estimator__min_samples_split': 2, 'base_estimator__splitter': 'best', 'algorithm': 'SAMME.R', 'learning_rate': 1.0, 'n_estimators': 50, 'random_state': 42}
Best score:  0.9666666666666666
Accuracy:  0.9666666666666666
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.90      0.95        10
           2       0.91      1.00      0.95        10

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
Confusion Matrix:
[[10  0  0]
 [ 0  9  1]
 [ 0  0 10]]
```

```
In [68]: ## Cross Validation Score for cat data
cvs = cross_val_score(best_ab, X, y, cv=5)
print(cvs)
print(np.mean(cvs))
print(np.median(cvs))
```

```
[1. 1. 1. 1. 1.]
1.0
1.0
```

```
In [69]: ## Cross validation prediction
cvp = cross_val_predict(best_ab, X, y, cv=5)
print(cvp)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

```
In [70]: ## Confusion matrix
cm = confusion_matrix(y, cvp)
print(cm)
```

```
[[50  0  0]
 [ 0 47  3]
 [ 0  0 50]]
```

```
In [71]: ## Plot the confusion matrix
plt.figure(figure_size=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

5.1.8. XgBoost Model

```
In [72]: ## Hyperparameter tuning for XgBoost Classifier
## Hyperparameter tuning for XgBoost Classifier
XGBHyperparameters = {'learning_rate': [0.1, 0.01, 0.001],
                      'n_estimators': [100, 200, 300],
                      'max_depth': [3, 4, 5, 6, 7, 8, 9, 10],
                      'min_child_weight': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                      'gamma': [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1],
                      'subsample': [0.6, 0.7, 0.8, 0.9, 1],
                      'colsample_bytree': [0.6, 0.7, 0.8, 0.9, 1],
                      'colsample_bylevel': [0.6, 0.7, 0.8, 0.9, 1],
                      'colsample_bynode': [0.6, 0.7, 0.8, 0.9, 1],
                      'reg_alpha': [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1],
                      'reg_lambda': [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1],
                      'scale_pos_weight': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                      'base_score': [0.5, 0.6, 0.7, 0.8, 0.9, 1],
                      'random_state': [None, 42],
                      'n_jobs': [None, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                      'verbosity': [0, 1, 2, 3],
                      'booster': ['gbtree', 'gblinear', 'dart'],
                      'tree_method': ['auto', 'exact', 'approx', 'hist', 'gpu_hist'],
                      'sampling_method': ['uniform', 'gradient_based'],
                      'grow_policy': ['depthwise', 'lossguide'],
                      'max_leaves': [None, 10, 20, 30, 40, 50],
                      'max_bin': [None, 10, 20, 30, 40, 50],
                      'predictor': ['auto', 'cpu_predictor', 'gpu_predictor'],
                      'num_parallel_tree': [None, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                      'monotone_constraints': [None, '(1,0,-1)', '(0,1,0,-1)', '(1,0,-1,0,1)'],
                      'interaction_constraints': [None, '[[0,1,2]]', '[[0,1,2,3,4]]', '[[0,1,2,3,4,5,6,7,8,9]]'],
                      'num_zero_tree': [None, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                      'num_feature': [None, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                      'num_class': [None, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                      'num_boost_round': [None, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                      'early_stopping_rounds': [None, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                      'verbose_eval': [True, False],
                      'silent': [True, False],
                      'xgb_model': [None, 'model.bin'],
                      'nthread': [None, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                      'seed': [None, 42],
                      'missing': [None, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                      'gpu_id': [None, 0, 1, 2, 3],
                      'gpu_hist': [True, False],
                      'gpu_predictor': [True, False
```

```
In [38]: CHECK_COST = 1.0/(train_size*0.5)
         print("CHECK_COST = %f" %CHECK_COST)
```

```
CHECK_COST = 0.000010
```

```
In [39]: train_data.groupby('Survived').count()
```

Survived	PassengerId	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	549	549	549	424	549	549	549	549	68	547
1	342	342	342	290	342	342	342	342	136	340

```
In [40]: train_data.groupby('Pclass').count()
```

Pclass	PassengerId	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	216	216	216	186	216	216	216	216	176	214
2	184	184	184	173	184	184	184	184	16	184
3	491	491	491	355	491	491	491	491	12	489

```
In [41]: train_data.groupby('Sex').count()
```

Sex	PassengerId	Name	Pclass	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
female	314	314	314	261	314	314	314	314	97	312
male	577	577	577	453	577	577	577	577	77	577

```
In [42]: train_data.groupby('SibSp').count()
```

SibSp	PassengerId	Name	Sex	Age	Parch	Ticket	Fare	Cabin	Embarked
0	608	608	608	471	608	608	608	608	604
1	209	209	209	183	209	209	209	209	209
2	28	28	28	25	28	28	28	28	28
3	16	16	16	12	16	16	16	16	16
4	18	18	18	15	18	18	18	18	18
5	5	5	5	5	5	5	5	5	5
8	7	7	7	6	7	7	7	7	7

```
In [43]: train_data.groupby('Parch').count()
```

Parch	PassengerId	Name	Sex	Age	SibSp	Ticket	Fare	Cabin	Embarked
0	678	678	678	537	678	678	678	678	674
1	118	118	118	108	118	118	118	118	118
2	80	80	80	70	80	80	80	80	80
3	5	5	5	5	5	5	5	5	5
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	1	1	1	1	1	1	1	1	1

```
In [44]: train_data.groupby('Embarked').count()
```

Embarked	PassengerId	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
C	168	168	168	130	168	168	168	168	59
Q	77	77	77	30	77	77	77	77	7
S	644	644	644	554	644	644	644	644	129

```
In [45]: train_data.groupby('Cabin').count()
```

Cabin	PassengerId	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
A10	1	1	1	1	1	1	1	1	1
A14	1	1	1	1	1	1	1	1	1
A16	1	1	1	1	1	1	1	1	1
A19	1	1	1	0	1	1	1	1	1
A20	1	1	1	1	1	1	1	1	1
A23	1	1	1	1	1	1	1	1	1
A24	2	2	2	2	2	2	2	2	2
A26	1	1	1	1	1	1	1	1	1
A31	1	1	1	1	1	1	1	1	1
A32	1	1	1	1	1	1	1	1	1
A34	1	1	1	1	1	1	1	1	1
A36	1	1	1	1	1	1	1	1	1
B101	1	1	1	1	1	1	1	1	1
B102	1	1	1	1	1	1	1	1	1
B18	2	2	2	2	2	2	2	2	2
B19	1	1	1	1	1	1	1	1	1
B20	2	2	2	2	2	2	2	2	2
B22	2	2	2	2	2	2	2	2	2
B28	1	1	1	1	1	1	1	1	1
B30	1	1	1	1	1	1	1	1	1
B35	2	2	2	2	2	2	2	2	2
B37	1	1	1	1	1	1	1	1	1
B38	1	1	1	1	1	1	1	1	1
B39	1	1	1	1	1	1	1	1	1
B41	1	1	1	1	1	1	1	1	1
B42	1	1	1	1	1	1	1	1	1
B49	1	1	1	1	1	1	1	1	1
B50	1	1	1	1	1	1	1	1	1
B51 B53 B55	3	3	3	2	3	3	3	3	3
B57 B59 B63 B66	4	4	4	3	4	4	4	4	4
B58 B60	2	2	2	2	2	2	2	2	2
B69	1	1	1	1	1	1	1	1	1
B71	1	1	1	1	1	1	1	1	1
B73	1	1	1	1	1	1	1	1	1
B77	1	1	1	1	1	1	1	1	1
B78	2	2	2	2	2	2	2	2	2
B79	1	1	1	1	1	1	1	1	1
B80	1	1	1	1	1	1	1	1	1
B82 B84	2	2	2	2	2	2	2	2	2
B86	1	1	1	1	1	1	1	1	1
B96 B98	4	4	4	4	4	4	4	4	4
C101	1	1	1	1	1	1	1	1	1
C103	2	2	2	2	2	2	2	2	2
C104	1	1	1	1	1	1	1	1	1
C106	1	1	1	1	1	1	1	1	1
C110	1	1	1	1	1	1	1	1	1
C111	1	1	1	1	1	1	1	1	1
C116	1	1	1	1	1	1	1	1	1
C118	1	1	1	1	1	1	1	1	1
C123	1	1	1	1	1	1	1	1	1
C125	2	2	2	2	2	2	2	2	2
C126	1	1	1	1	1	1	1	1	1
C128	1	1	1	1	1	1	1	1	1
C130	1	1	1	1	1	1	1	1	1
C132	1	1	1	1	1	1	1	1	1
C148	1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	2	2	2
C21	2	2	2	2	2	2	2	2	2
C22 C26	3	3	3	3	3	3	3	3	3
C23 C25 C27	4	4	4	4	4	4	4	4	4
C30	1	1	1	1	1	1	1	1	1
C31	1	1	1	1	1	1	1	1	1
C32	1	1	1	1	1	1	1	1	1
C33	3	3	3	3	3	3	3	3	3
C34	2	2	2	2	2	2	2	2	2
C35	1	1	1	1	1	1	1	1	1
C37	1	1	1	1	1	1	1	1	1
C39	1	1	1	1	1	1	1	1	1
C45	1	1	1	1	1	1	1	1	1
C46	1	1	1	1	1	1	1	1	1
C47	1	1	1	1	1	1	1	1	1
C49	1	1	1	1	1	1	1	1	1
C50	1	1	1	1	1	1	1	1	1
C52	1	1	1	1	1	1	1	1	1
C54	1	1	1	1	1	1	1	1	1
C55 C57	2	2	2	2	2	2	2	2	2
C62 C64	2	2	2	2	2	2	2	2	2
C65	1	1	1	1	1	1	1	1	1
C68	1	1	1	1	1	1	1	1	1
C70	1	1	1	1	1	1	1	1	1
C78	2	2	2	2	2	2	2	2	2
C80	1	1	1	1	1	1	1	1	1
C82	1	1	1	1	1	1	1	1	1
C83	1	1	1	1	1	1	1	1	1
C85	2	2	2	2	2	2	2	2	2
C86	1	1	1	1	1	1	1	1	1
C87	1	1	1	1	1	1	1	1	1
C90	2	2	2	2	2	2	2	2	2
C91	1	1	1	1	1	1	1	1	1
C92	1	1	1	1	1	1	1	1	1
C93	1	1	1	1	1	1	1	1	1
C95	1	1	1	1	1	1	1	1	1
C96	1	1	1	1	1	1	1	1	1
C99	2	2	2	2	2	2	2	2	2
D	4	4	4	3	4	4	4	4	4
D10 D12	2	2	2	2	2	2	2	2	2
D11	1	1	1	1	1	1	1	1	1
D15	2	2	2	2	2	2	2	2	2
D17	2	2	2	2	2	2	2	2	2
D19	1	1	1	1	1	1	1	1	1
D20	2	2	2	2	2	2	2	2	2
D21	2	2	2	2	2	2	2	2	2
D26	2	2	2	2	2	2	2	2	2
D28	1	1	1	1	1	1	1	1	1
D30	1	1	1	1	1	1	1	1	1
D33	1	1	1	1	1	1	1	1	1
D35	2	2	2	2	2	2	2	2	2
D36	2	2	2	2	2	2	2	2	2
D37	1	1	1	1	1	1	1	1	1
D38	1	1	1	1	1	1	1	1	1
D40	1	1	1	1	1	1	1	1	1
D43	1	1	1	1	1	1	1	1	1
D45	1	1	1	1	1	1	1	1	1
D46	1	1	1	1	1	1	1	1	1
D47	1	1	1	1	1	1	1	1	1
D48	1	1	1	1	1	1	1	1	1
D49	2	2	2	2	2	2	2	2	2
D50	2	2	2	2	2	2	2	2	2
D56	1	1	1	1	1	1	1	1	1
D6	2	2	2	2	2	2	2	2	2
D7	2	2	2	2	2	2	2	2	2
D9	2	2	2	2	2	2	2	2	2
E10	2	2	2	2	2	2	2	2	2
E12	2	2	2	2	2	2	2	2	2
E24	2	2	2	2	2	2	2	2	2
E25	2	2	2	2	2	2	2	2	2
E31	1	1	1	1	1	1	1	1	1
E33	2	2	2	2	2	2	2	2	2
E34	1	1	1	1	1	1	1	1	1
E36	1	1	1	1	1	1	1	1	1
E38	1	1	1	1	1	1	1	1	1
E40	1	1	1	1	1	1	1	1	1
E44	1	1	1	1	1	1	1	1	1
E46	1	1	1	1	1	1	1	1	1
E49	2	2	2	2	2	2	2	2	2
E50	1	1	1	1	1	1	1	1	1
E52	1	1	1	1	1	1	1	1	1
E58	1	1	1	1	1	1	1	1	1
E60	1	1	1	1	1	1	1	1	1
E63	1	1	1	1	1	1	1	1	1
E67	1	1	1	1	1	1	1	1	1
E68	1	1	1	1	1	1	1	1	1
E77	1	1	1	1	1	1	1	1	1
E8	2	2	2	2	2	2	2	2	2

2 rows × 11 columns

```
In [46]: train_data.groupby('Ticket').count()
```

Ticket	PassengerId	Name	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
110152	3	3	3	3	3	3	3	3	3
110413	3	3	3	3	3	3	3	3	3
110465	2	2	2	2	2	2	2	2	2
110564	1	1	1	1	1	1	1	1	1
110813	1	1	1	1	1	1	1	1	1
111240	1	1	1	1	1	1	1	1	1
111320	1	1	1	1	1	1	1	1	1
111361	1	1	1	1	1	1	1	1	1
111369	1	1	1	1	1	1	1	1	1
111426	1	1	1	1	1	1	1	1	1
111428	1	1	1	1	1	1	1	1	1
112050	1	1	1	1	1	1	1	1	1
112052	1	1	1	1	1	1	1	1	1
112053	1	1	1	1	1	1	1	1	1
112058	1	1	1	1	1	1	1	1	1
112059	1	1	1	1	1	1	1	1	1
112379	1	1	1	1	1	1	1	1	1
113028	2	2	2	2	2	2	2	2	2
113043	1	1	1	1	1	1	1	1	1
113050	1	1	1	1	1	1	1	1	1
113051	1	1	1	1	1	1	1	1	1
113055	1	1	1	1	1	1	1	1	1
113056	1	1	1	1	1	1	1	1	1
113059	1	1	1	1	1	1	1	1	1
113060	1	1	1	1	1	1	1	1	1
113073	1	1	1	1	1	1	1	1	1
113075	1	1	1	1	1	1	1	1	1
113076	2	2	2	2	2	2	2	2	2
113077	1	1	1	1	1	1	1	1	1
113083	1	1	1	1	1	1	1	1	1
113087	1	1	1	1	1	1	1	1	1
113088	1	1	1	1	1	1	1	1	1
113089	1	1	1	1	1	1	1	1	1
113090	1	1	1	1	1	1	1	1	1
113091	1	1	1	1	1	1	1	1	1
113094	1	1	1	1	1	1	1	1	1
113096	1	1	1	1	1	1	1	1	1
113098	1	1	1	1	1	1	1	1	1
113099	1	1	1	1	1	1	1	1	1
113100	1	1	1	1	1	1	1	1	1
113101	1	1	1	1	1	1	1	1	1
113102	1	1	1	1	1	1	1	1	1
113103	2	2	2	2	2	2	2	2	2
113104	1	1	1	1	1	1	1	1	1
113105	1	1	1	1	1	1	1	1	1
113107	1	1	1	1	1	1	1	1	1
113108	1	1	1	1	1	1	1	1	1
113109	1	1	1	1	1	1	1	1	1
113110	1	1	1	1	1	1	1	1	1
113111	1	1	1	1	1	1	1	1	1
113112	1	1	1	1	1	1	1	1	1
113113	1	1	1	1	1	1	1	1	1
113114	1	1	1	1	1	1	1	1	1
113116	1	1	1	1	1	1	1	1	1
113117	1	1	1	1	1	1	1	1	1
113118	1	1	1	1	1	1	1	1	1
113119	1	1	1	1	1	1	1	1	1
113120	1	1	1	1	1	1	1	1	1
113121	1	1	1	1	1	1	1	1	1
113122	1	1	1	1	1	1	1	1	1
113123	1	1	1	1	1	1	1	1	1
113124	1	1	1	1	1	1	1	1	1
113125	1	1	1	1	1	1	1	1	1
113126	1	1	1	1	1	1	1	1	1
113128	1	1	1	1	1	1	1	1	1
113129	1	1	1	1	1	1	1	1	1
113130	1	1	1	1	1	1	1	1	1
113131	1	1	1	1	1	1	1	1	1
113132	1	1	1	1	1	1	1	1	1
113134	1	1	1	1	1	1	1	1	1
113135	1	1	1	1	1	1	1	1	1
113136	1	1	1	1	1	1	1	1	1
113137	1	1	1	1	1	1	1	1	1
113138	1	1	1	1	1	1	1	1	1
113139	1	1	1	1	1	1	1	1	1
113140	1	1	1	1	1	1	1	1	1
113141	1	1	1	1	1	1	1	1	1
113142	1	1	1	1	1	1	1	1	1
113143	1	1	1	1	1	1	1	1	1
113144	1	1	1	1	1	1	1	1	1
113145	1	1	1	1	1	1	1	1	1
113146	1	1	1	1	1	1	1	1	1
113147	1	1	1	1	1	1	1	1	1
113148	1	1	1	1	1	1	1	1	1
113149	1	1	1	1	1	1	1	1	1
113150	1	1	1	1	1	1	1	1	1
113151	1	1	1	1	1	1	1	1	1
113152	1	1	1	1	1	1	1	1	1
113153	1	1	1	1	1	1	1	1	1
113154	1	1	1	1	1	1	1	1	1
113155	1	1	1	1	1	1	1	1	1
113156	1	1	1	1	1	1	1	1	1
113157	1	1	1	1	1	1	1	1	1
113158	1	1	1	1	1	1	1	1	1
113159	1	1	1	1	1	1	1	1	1
113160	1	1	1	1	1	1	1	1	1
113161	1	1	1	1	1	1	1	1	1
113162	1	1	1	1	1	1	1	1	1
113163	1	1	1	1	1	1	1	1	1
113164	1	1	1	1	1	1	1	1	1
113165	1	1	1	1	1	1	1	1	1
113166	1	1	1	1	1	1	1	1	1
113167	1	1	1	1	1	1	1	1	1
113168	1	1	1	1	1	1	1	1	1
113169	1	1	1	1	1	1	1	1	1
113170	1	1	1	1	1	1	1	1	1
113171	1	1	1	1	1	1	1	1	1
113172	1	1	1	1	1	1	1	1	1
113173	1	1	1	1	1	1	1	1	1
113174	1	1	1	1	1	1	1	1	1
113175	1	1	1	1	1	1	1	1	1
113176	1	1	1	1	1	1	1	1	1
113177	1	1	1	1	1	1	1	1	1
113178	1	1	1	1	1	1	1	1	1
113179	1	1	1	1	1	1	1	1	1
113180	1	1	1	1	1	1	1	1	1
113181	1	1	1	1	1	1	1	1	1
113182	1	1	1	1	1	1	1	1	1
113183	1	1	1	1	1	1	1	1	1
113184	1	1	1	1	1	1	1	1	1
113185	1	1	1	1	1	1	1	1	1
113186	1	1	1	1	1	1	1	1	1
113187	1	1	1	1	1	1	1	1	1
113188	1	1	1	1	1	1	1	1	1
113189	1	1	1	1	1	1	1	1	1
113190	1	1	1	1	1	1	1	1	1
113191	1	1	1	1	1	1	1	1	1
113192	1	1	1	1	1	1	1	1	1
113193	1	1	1	1	1	1	1	1	1
113194	1	1	1	1	1	1	1	1	1
113195	1	1	1	1	1	1	1	1	1
113196	1	1	1	1	1	1	1	1	1
113197	1	1	1	1	1	1	1	1	1
113198	1	1	1	1	1	1	1	1	1
113199	1	1	1	1	1	1	1	1	1
113200	1	1	1	1	1	1	1	1	1
113201	1	1	1	1	1	1	1	1	1
113202	1	1	1	1	1	1	1	1	1
113203	1	1	1	1	1	1	1	1	1
113204	1	1	1	1	1	1	1	1	1
113205	1	1	1	1	1	1	1	1	1
113206	1	1	1	1	1	1	1	1	1
113207	1	1	1	1	1	1	1	1	1
113208	1	1	1	1	1	1	1	1	1
113209	1	1	1	1	1	1	1	1	1
113210	1	1	1	1	1	1	1	1	1
113211	1	1	1	1	1	1	1	1	1
113212	1	1	1	1	1	1	1	1	1
113213	1	1	1	1	1	1	1	1	1
113214	1	1	1	1	1	1	1	1	1
113215	1	1	1	1	1	1	1	1	1
113216	1	1	1	1	1	1	1	1	1
113217	1	1	1	1	1	1	1	1	1
113218	1	1	1	1	1	1	1	1	1
113219	1	1	1	1	1	1	1	1	1
113220	1	1	1	1	1	1	1	1	1
113221	1	1	1	1	1	1	1	1	1
113222	1	1	1	1	1	1	1	1	1
113223	1	1	1	1	1	1	1	1	1
113224	1	1	1	1	1	1	1	1	1
113225	1	1	1	1	1	1	1	1	1
113226	1	1	1	1	1	1	1	1	1
113227	1	1	1	1	1	1	1	1	1
113228	1	1	1	1	1	1	1	1	1
113229	1	1	1	1	1	1	1	1	1
113230	1	1	1	1	1	1	1	1	1
113231	1	1	1	1	1	1	1	1	1
113232	1	1	1	1	1	1	1	1	1
113233	1	1	1	1	1	1	1	1	1
113234	1	1	1	1	1	1	1	1	1
113235	1	1	1	1	1	1	1	1	1
113236	1	1	1	1	1	1	1	1	1
113237	1	1	1	1	1	1	1	1	1
113238	1	1	1	1	1	1	1	1	1
113239	1	1	1	1	1	1	1	1	1
113240	1	1	1	1	1	1	1	1	1
113241	1	1	1	1	1	1	1	1	1
113242	1	1	1	1	1	1	1	1	1
113243	1	1	1	1	1	1	1	1	1
113244	1	1	1	1	1	1	1	1	1
113245	1	1	1	1	1	1	1	1	1
113246	1	1	1	1	1	1	1	1	1
113247	1	1	1	1	1	1	1	1	1
113248	1	1	1	1	1	1	1	1	1
113249	1	1	1	1	1	1	1	1	1
113250	1	1	1	1	1	1	1	1	1
113251	1	1	1	1	1	1