Atelier 3 : Classification

1. Objectif :

L'objectif de cet atelier est de découvrir la classification de documents texte à travers plusieure classificateur et que nous allons explorer avec le dataset contenant les news apparus sur le fil de presse Reuters en 1987.

Le datset peut être téléchargé à partir de ce lien :https://archive.ics.uci.edu/dataset/137/reuters+21578+text+categorization+collection

Le dataset est disponible egalement sur nltk et scikitlearn:

```
import nltk
nltk.download('reuters')

#from sklearn.datasets import fetch_rcv1
```

```
[nltk_data] Downloading package reuters to
[nltk_data]     C:\Users\dscon\AppData\Roaming\nltk_data...
[nltk_data]   Package reuters is already up-to-date!
```

```
True
```

2. Chargement des données

Nous considerons le dataset reuters de NLTK.

```
from nltk.corpus import reuters

#recuperation du vocabulaire du corpus
vocabulaire=reuters.words()

#recuperation de toutes les categories
categories=reuters.categories()

#recuperation de tous les id des fichiers appartenant à une categorie
bien determinée
ids_coffe=reuters.fileids("coffee")

#recuperation des mots contenus dans les documents d'une categorie
bien determinee
coffe_words=reuters.words(reuters.fileids("coffee"))
```

```
#recuperation du texte brut des documents d'une categorie bien
determinee
cofee_docs=reuters.raw(reuters.fileids("coffee")[0])

#recuperation de toutes les autres classe d'un document annoté avec
une classe bien determinee
classes_Annotated_coffee=reuters.categories(reuters.fileids("coffee"))

#recuperer le dataset d'apprentissage
train_categories=[ reuters.categories(i) for i in reuters.fileids() if
i.startswith('training/')]
train_documents = [reuters.raw(i) for i in reuters.fileids() if
i.startswith('training/')]

#recuperer le dataset de test
test_documents=[reuters.raw(i)  for i in reuters.fileids() if
i.startswith('test/')]
test_categories = [reuters.categories(i) for i in reuters.fileids() if
i.startswith('test/')]

# recuperer tout le corpus
whole_docs=[reuters.raw(i)for i in reuters.fileids()]
whole_cats = [ reuters.categories(i) for i in reuters.fileids()]

len(test_documents)

3019

len(categories)

90

len(whole_cats)

10788

len(reuters.raw(reuters.fileids("coffee")[0]))

2530

len(reuters.raw(reuters.fileids("coffee")))

203702

reuters.categories()[:10]

['acq',
 'alum',
 'barley',
 'bop',
 'carcass',
 'castor-oil',
```

```
 'cocoa',
 'coconut',
 'coconut-oil',
 'coffee']
```

3. Prétraitements

```python
len(reuters.categories())
```

```
90
```

```python
len(whole_docs)
```

```
10788
```

```python
whole_docs[:10]
```

```
['ASIAN EXPORTERS FEAR DAMAGE FROM U.S.-JAPAN RIFT\n  Mounting trade
friction between the\n  U.S. And Japan has raised fears among many of
Asia\'s exporting\n  nations that the row could inflict far-reaching
economic\n  damage, businessmen and officials said.\n      They told
Reuter correspondents in Asian capitals a U.S.\n  Move against Japan
might boost protectionist sentiment in the\n  U.S. And lead to curbs
on American imports of their products.\n      But some exporters said
that while the conflict would hurt\n  them in the long-run, in the
short-term Tokyo\'s loss might be\n  their gain.\n      The U.S. Has
said it will impose 300 mln dlrs of tariffs on\n  imports of Japanese
electronics goods on April 17, in\n  retaliation for Japan\'s alleged
failure to stick to a pact not\n  to sell semiconductors on world
markets at below cost.\n      Unofficial Japanese estimates put the
impact of the tariffs\n  at 10 billion dlrs and spokesmen for major
electronics firms\n  said they would virtually halt exports of
products hit by the\n  new taxes.\n      "We wouldn\'t be able to do
business," said a spokesman for\n  leading Japanese electronics firm
Matsushita Electric\n  Industrial Co Ltd &lt;MC.T>.\n      "If the
tariffs remain in place for any length of time\n  beyond a few months
it will mean the complete erosion of\n  exports (of goods subject to
tariffs) to the U.S.," said Tom\n  Murtha, a stock analyst at the
Tokyo office of broker &lt;James\n  Capel and Co>.\n      In Taiwan,
businessmen and officials are also worried.\n      "We are aware of
the seriousness of the U.S. Threat against\n  Japan because it serves
as a warning to us," said a senior\n  Taiwanese trade official who
asked not to be named.\n      Taiwan had a trade trade surplus of 15.6
billion dlrs last\n  year, 95 pct of it with the U.S.\n      The
surplus helped swell Taiwan\'s foreign exchange reserves\n  to 53
billion dlrs, among the world\'s largest.\n      "We must quickly open
our markets, remove trade barriers and\n  cut import tariffs to allow
```

imports of U.S. Products, if we\n want to defuse problems from
possible U.S. Retaliation," said\n Paul Sheen, chairman of textile
exporters <Taiwan Safe Group>.\n   A senior official of South
Korea\'s trade promotion\n association said the trade dispute between
the U.S. And Japan\n might also lead to pressure on South Korea,
whose chief exports\n are similar to those of Japan.\n   Last year
South Korea had a trade surplus of 7.1 billion\n dlrs with the U.S.,
Up from 4.9 billion dlrs in 1985.\n   In Malaysia, trade officers
and businessmen said tough\n curbs against Japan might allow hard-hit
producers of\n semiconductors in third countries to expand their
sales to the\n U.S.\n   In Hong Kong, where newspapers have
alleged Japan has been\n selling below-cost semiconductors, some
electronics\n manufacturers share that view. But other businessmen
said such\n a short-term commercial advantage would be outweighed by\
n further U.S. Pressure to block imports.\n   "That is a very
short-term view," said Lawrence Mills,\n director-general of the
Federation of Hong Kong Industry.\n   "If the whole purpose is to
prevent imports, one day it will\n be extended to other sources. Much
more serious for Hong Kong\n is the disadvantage of action
restraining trade," he said.\n   The U.S. Last year was Hong
Kong\'s biggest export market,\n accounting for over 30 pct of
domestically produced exports.\n   The Australian government is
awaiting the outcome of trade\n talks between the U.S. And Japan with
interest and concern,\n Industry Minister John Button said in
Canberra last Friday.\n   "This kind of deterioration in trade
relations between two\n countries which are major trading partners of
ours is a very\n serious matter," Button said.\n   He said
Australia\'s concerns centred on coal and beef,\n Australia\'s two
largest exports to Japan and also significant\n U.S. Exports to that
country.\n   Meanwhile U.S.-Japanese diplomatic manoeuvres to solve
the\n trade stand-off continue.\n   Japan\'s ruling Liberal
Democratic Party yesterday outlined\n a package of economic measures
to boost the Japanese economy.\n   The measures proposed include a
large supplementary budget\n and record public works spending in the
first half of the\n financial year.\n   They also call for
stepped-up spending as an emergency\n measure to stimulate the
economy despite Prime Minister\n Yasuhiro Nakasone\'s avowed fiscal
reform program.\n   Deputy U.S. Trade Representative Michael Smith
and Makoto\n Kuroda, Japan\'s deputy minister of International Trade
and\n Industry (MITI), are due to meet in Washington this week in an\
n effort to end the dispute.\n \n\n',
 "CHINA DAILY SAYS VERMIN EAT 7-12 PCT GRAIN STOCKS\n A survey of 19
provinces and seven cities\n showed vermin consume between seven and
12 pct of China's grain\n stocks, the China Daily said.\n   It
also said that each year 1.575 mln tonnes, or 25 pct, of\n China's
fruit output are left to rot, and 2.1 mln tonnes, or up\n to 30 pct,
of its vegetables. The paper blamed the waste on\n inadequate storage
and bad preservation methods.\n   It said the government had
launched a national programme to\n reduce waste, calling for improved

technology in storage and\n preservation, and greater production of
additives. The paper\n gave no further details.\n \n\n",
 "JAPAN TO REVISE LONG-TERM ENERGY DEMAND DOWNWARDS\n The Ministry of
International Trade and\n Industry (MITI) will revise its long-term
energy supply/demand\n outlook by August to meet a forecast downtrend
in Japanese\n energy demand, ministry officials said.\n   MITI is
expected to lower the projection for primary energy\n supplies in the
year 2000 to 550 mln kilolitres (kl) from 600\n mln, they said.\n
The decision follows the emergence of structural changes in\n
Japanese industry following the rise in the value of the yen\n and a
decline in domestic electric power demand.\n   MITI is planning to
work out a revised energy supply/demand\n outlook through
deliberations of committee meetings of the\n Agency of Natural
Resources and Energy, the officials said.\n   They said MITI will
also review the breakdown of energy\n supply sources, including oil,
nuclear, coal and natural gas.\n   Nuclear energy provided the bulk
of Japan's electric power\n in the fiscal year ended March 31,
supplying an estimated 27\n pct on a kilowatt/hour basis, followed by
oil (23 pct) and\n liquefied natural gas (21 pct), they noted.\n \n\
n",
 "THAI TRADE DEFICIT WIDENS IN FIRST QUARTER\n Thailand's trade
deficit widened to 4.5\n billion baht in the first quarter of 1987
from 2.1 billion a\n year ago, the Business Economics Department
said.\n   It said Janunary/March imports rose to 65.1 billion baht\
n from 58.7 billion. Thailand's improved business climate this\n
year resulted in a 27 pct increase in imports of raw materials\n and
semi-finished products.\n   The country's oil import bill, however,
fell 23 pct in the\n first quarter due to lower oil prices.\n
The department said first quarter exports expanded to 60.6\n billion
baht from 56.6 billion.\n   Export growth was smaller than expected
due to lower\n earnings from many key commodities including rice
whose\n earnings declined 18 pct, maize 66 pct, sugar 45 pct, tin 26\
n pct and canned pineapples seven pct.\n   Products registering
high export growth were jewellery up\n 64 pct, clothing 57 pct and
rubber 35 pct.\n \n\n",
 "INDONESIA SEES CPO PRICE RISING SHARPLY\n Indonesia expects crude
palm oil (CPO)\n prices to rise sharply to between 450 and 550 dlrs a
tonne FOB\n sometime this year because of better European demand and
a fall\n in Malaysian output, Hasrul Harahap, junior minister for
tree\n crops, told Indonesian reporters.\n   Prices of Malaysian
and Sumatran CPO are now around 332\n dlrs a tonne CIF for delivery
in Rotterdam, traders said.\n   Harahap said Indonesia would
maintain its exports, despite\n making recent palm oil purchases from
Malaysia, so that it\n could possibly increase its international
market share.\n   Indonesia, the world's second largest producer of
palm oil\n after Malaysia, has been forced to import palm oil to
ensure\n supplies during the Moslem fasting month of Ramadan.\n
Harahap said it was better to import to cover a temporary\n shortage
than to lose export markets.\n   Indonesian exports of CPO in

calendar 1986 were 530,500\n tonnes, against 468,500 in 1985,
according to central bank\n figures.\n \n\n",
 "AUSTRALIAN FOREIGN SHIP BAN ENDS BUT NSW PORTS HIT\n Tug crews in
New South Wales (NSW),\n Victoria and Western Australia yesterday
lifted their ban on\n foreign-flag ships carrying containers but NSW
ports are still\n being disrupted by a separate dispute, shipping
sources said.\n   The ban, imposed a week ago over a pay claim, had
prevented\n the movement in or out of port of nearly 20 vessels, they
said.\n   The pay dispute went before a hearing of the Arbitration\
n Commission today.\n   Meanwhile, disruption began today to cargo
handling in the\n ports of Sydney, Newcastle and Port Kembla, they
said.\n   The industrial action at the NSW ports is part of the
week\n of action called by the NSW Trades and Labour Council to\n
protest changes to the state's workers' compensation laws.\n   The
shipping sources said the various port unions appear to\n be taking
it in turn to work for a short time at the start of\n each shift and
then to walk off.\n   Cargo handling in the ports has been
disrupted, with\n container movements most affected, but has not
stopped\n altogether, they said.\n   They said they could not say
how long the disruption will\n go on and what effect it will have on
shipping movements.\n \n\n",
 'INDONESIAN COMMODITY EXCHANGE MAY EXPAND\n The Indonesian Commodity
Exchange is\n likely to start trading in at least one new commodity,
and\n possibly two, during calendar 1987, exchange chairman Paian\n
Nainggolan said.\n   He told Reuters in a telephone interview that
trading in\n palm oil, sawn timber, pepper or tobacco was being
considered.\n   Trading in either crude palm oil (CPO) or refined
palm oil\n may also be introduced. But he said the question was
still\n being considered by Trade Minister Rachmat Saleh and no\n
decision on when to go ahead had been made.\n   The fledgling
exchange currently trades coffee and rubber\n physicals on an open
outcry system four days a week.\n   "Several factors make us move
cautiously," Nainggolan said.\n "We want to move slowly and safely so
that we do not make a\n mistake and undermine confidence in the
exchange."\n   Physical rubber trading was launched in 1985, with
coffee\n added in January 1986. Rubber contracts are traded FOB, up
to\n five months forward. Robusta coffee grades four and five are\n
traded for prompt delivery and up to five months forward,\n exchange
officials said.\n   The trade ministry and exchange board are
considering the\n introduction of futures trading later for rubber,
but one\n official said a feasibility study was needed first. No\n
decisions are likely until after Indonesia\'s elections on April\n
23, traders said.\n   Trade Minister Saleh said on Monday that
Indonesia, as the\n world\'s second largest producer of natural
rubber, should\n expand its rubber marketing effort and he hoped
development of\n the exchange would help this.\n   Nainggolan said
that the exchange was trying to boost\n overseas interest by building
up contacts with end-users.\n   He said teams had already been to
South Korea and Taiwan to\n encourage direct use of the exchange,

while a delegation would\n also visit Europe, Mexico and some Latin
American states to\n encourage participation.\n   Officials say
the infant exchange has made a good start\n although trading in
coffee has been disappointing.\n   Transactions in rubber between
the start of trading in\n April 1985 and December 1986 totalled 9,595
tonnes, worth 6.9\n mln dlrs FOB, plus 184.3 mln rupiah for rubber
delivered\n locally, the latest exchange report said.\n   Trading
in coffee in calendar 1986 amounted to only 1,905\n tonnes in 381
lots, valued at 6.87 billion rupiah.\n   Total membership of the
exchange is now nine brokers and\n 44 traders.\n \n\n',
 'SRI LANKA GETS USDA APPROVAL FOR WHEAT PRICE\n Food Department
officials said the U.S.\n Department of Agriculture approved the
Continental Grain Co\n sale of 52,500 tonnes of soft wheat at 89 U.S.
Dlrs a tonne C\n and F from Pacific Northwest to Colombo.\n   They
said the shipment was for April 8 to 20 delivery.\n \n\n',
 'WESTERN MINING TO OPEN NEW GOLD MINE IN AUSTRALIA\n Western Mining
Corp Holdings Ltd\n <WMNG.S> (WMC) said it will establish a new
joint venture gold\n mine in the Northern Territory at a cost of
about 21 mln dlrs.\n   The mine, to be known as the Goodall
project, will be owned\n 60 pct by WMC and 40 pct by a local W.R.
Grace and Co <GRA>\n unit. It is located 30 kms east of the
Adelaide River at Mt.\n Bundey, WMC said in a statement\n   It
said the open-pit mine, with a conventional leach\n treatment plant,
is expected to produce about 50,000 ounces of\n gold in its first
year of production from mid-1988. Annual ore\n capacity will be about
750,000 tonnes.\n \n\n',
 'SUMITOMO BANK AIMS AT QUICK RECOVERY FROM MERGER\n Sumitomo Bank
Ltd <SUMI.T> is certain to\n lose its status as Japan\'s most
profitable bank as a result of\n its merger with the Heiwa Sogo Bank,
financial analysts said.\n   Osaka-based Sumitomo, with desposits
of around 23.9\n trillion yen, merged with Heiwa Sogo, a small,
struggling bank\n with an estimated 1.29 billion dlrs in
unrecoverable loans, in\n October.\n   But despite the link-up,
Sumitomo President Koh Komatsu\n told Reuters he is confident his
bank can quickly regain its\n position.\n   "We\'ll be back in
position in first place within three\n years," Komatsu said in an
interview.\n   He said that while the merger will initially reduce\
n Sumitomo\'s profitability and efficiency, it will vastly expand\n
Sumitomo\'s branch network in the Tokyo metropolitan area where\n it
has been relatively weak.\n   But financial analysts are divided on
whether and how\n quickly the gamble will pay off.\n   Some said
Sumitomo may have paid too much for Heiwa Sogo in\n view of the
smaller bank\'s large debts. Others argue the merger\n was more cost
effective than creating a comparable branch\n network from scratch.\n
The analysts agreed the bank was aggressive. It has\n expanded
overseas, entered the lucrative securities business\n and geared up
for domestic competition, but they questioned the\n wisdom of some of
those moves.\n   "They\'ve made bold moves to put everything in
place. Now\n it\'s largely out of their hands," said Kleinwort Benson

```
Ltd\n  financial analyst Simon Smithson.\n      Among Sumitomo\'s
problems are limits placed on its move to\n  enter U.S. Securities
business by taking a share in American\n  investment bank Goldman,
Sachs and Co.\n      Sumitomo last August agreed to pay 500 mln dlrs
for a 12.5\n  pct limited partnership in the bank, but for the time
being at\n  least, the Federal Reserve Board has forbidden them to
exchange\n  personnel, or increase the business they do with each
other.\n      "The tie-up is widely looked on as a lame duck because
the\n  Fed was stricter than Sumitomo expected," said one analyst.\n
But Komatsu said the move will pay off in time.\n      "U.S.
Regulations will change in the near future and if so,\n  we can do
various things. We only have to wait two or three\n  years, not until
the 21st century," Komatsu said.\n      Komatsu is also willing to be
patient about possible routes\n  into the securities business at
home.\n      Article 65 of the Securities and Exchange Act, Japan\'s\n
version of the U.S. Glass-Steagall Act, separates commercial\n  from
investment banking.\n      But the walls between the two are crumbling
and Komatsu\n  said he hopes further deregulation will create new\n
opportunities.\n      "We need to find new business chances," Komatsu
said. "In some\n  cases these will be securities related, in some
cases trust\n  bank related. That\'s the kind of deregulation we
want."\n      Until such changes occur, Sumitomo will focus on such\n
domestic securities business as profitable government bond\n  dealing
and strengthening relations with Meiko Securities Co\n  Ltd, in which
it holds a five pct share, Komatsu said.\n      He said Sumitomo is
cautiously optimistic about entering\n  the securities business here
through its Swiss universal bank\n  subsidiary, Banca del Gottardo.\n
The Finance Ministry is expected to grant licences to\n  securities
subsidiaries of U.S. Commercial banks soon,\n  following a similar
decision for subsidiaries of European\n  universal banks in which the
parent holds a less than 50 pct.\n      But Komatsu is reluctant to
push hard for a similar\n  decision on a Gottardo subsidiary.\n
"We don\'t want to make waves. We expect this will be allowed\n  in
two or three years," he said.\n      Like other city banks, Sumitomo
is also pushing to expand\n  lending to individuals and small and
medium businesses to\n  replace disappearing demand from big business,
he added.\n      The analysts said Sumitomo will have to devote a lot
of\n  time to digesting its most recent initiatives, including the\n
merger with ailing Heiwa Sogo.\n      "It\'s (Sumitomo) been bold in
its strategies," said\n  Kleinwort\'s Smithson.\n      "After that,
it\'s a question of absorbing and juggling\n  around. It will be the
next decade before we see if the\n  strategy is right or wrong."\n  \
n\n']
```

```python
from sklearn.feature_extraction.text import TfidfVectorizer
import nltk, re
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import string
```

```python
vectorizer = TfidfVectorizer(stop_words = 'english')

#Generer le vocabulaire à partir du whole_docs
#Realiser les differentes operations NLP permettant d'unifier la
representation
#vectorielle de tout le corpus


def preprocessing(text):
    text = text.lower()
    text = re.sub(r'\d+', '', text)
    text = text.translate(str.maketrans('', '', string.punctuation))
    words = [token for token in word_tokenize(text) if token not in
stopwords.words('english')]
    return ' '.join(words)

# Ensure whole_docs is a list of documents
whole_docs = list(whole_docs)
whole_docs = [preprocessing(doc) for doc in whole_docs]
train_documents = [preprocessing(doc) for doc in train_documents]
test_documents = [preprocessing(doc) for doc in test_documents]

vect_whole_docs = vectorizer.fit_transform(whole_docs)
print(vectorizer.get_feature_names_out())
print(f'len {len(vectorizer.get_feature_names_out())}')

#vectoriser les datasets d'apprentissage et de test
vect_train_docs = vectorizer.transform(train_documents)
vect_test_docs = vectorizer.transform(test_documents)
print(vect_train_docs.toarray())
```

```
['aa' 'aaa' 'aabex' ... 'zverev' 'zwermann' 'zzzz']
len 34509
[[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.]]
```

```python
lens = [ len(doc.split(' ')) for doc in whole_docs ]
print(f'Nombre de mots dans chaque doc {lens}')
print(f"Nombres de docs {len(lens)}")
sum(lens)
```

```
Nombre de mots dans chaque doc [447, 64, 109, 100, 103, 116, 241, 34,
69, 391, 60, 148, 166, 53, 301, 60, 82, 60, 248, 163, 69, 67, 104,
132, 60, 49, 62, 117, 60, 65, 21, 169, 453, 416, 152, 20, 109, 51,
133, 58, 105, 144, 117, 52, 22, 48, 459, 113, 60, 62, 124, 55, 264,
```

```
51, 81, 17, 98, 16, 74, 66, 27, 79, 16, 126, 36, 18, 37, 61, 18, 39,
16, 34, 28, 27, 21, 105, 59, 27, 265, 48, 120, 17, 22, 198, 96, 115,
19, 40, 41, 19, 17, 33, 56, 17, 18, 16, 117, 20, 35, 126, 16, 40, 60,
13, 54, 45, 90, 25, 94, 24, 44, 336, 76, 2, 131, 63, 40, 20, 102, 54,
53, 35, 19, 529, 109, 27, 14, 16, 18, 15, 16, 23, 16, 41, 160, 14, 65,
76, 179, 35, 76, 253, 38, 145, 81, 17, 131, 30, 107, 17, 24, 23, 135,
78, 12, 30, 58, 22, 49, 54, 39, 21, 39, 17, 73, 69, 322, 19, 18, 239,
98, 16, 58, 62, 65, 20, 20, 75, 30, 16, 39, 98, 50, 106, 348, 17, 59,
124, 33, 59, 165, 18, 17, 53, 66, 111, 239, 61, 104, 72, 35, 20, 66,
117, 151, 133, 57, 48, 68, 182, 239, 109, 16, 18, 55, 193, 26, 69,
115, 333, 174, 15, 35, 18, 189, 167, 12, 102, 58, 33, 64, 60, 25, 40,
77, 16, 84, 70, 72, 51, 23, 44, 18, 16, 22, 73, 81, 52, 130, 45, 133,
28, 72, 131, 94, 42, 24, 94, 65, 182, 50, 38, 52, 89, 16, 35, 16, 53,
76, 120, 32, 166, 114, 479, 39, 30, 100, 28, 149, 352, 48, 68, 119,
60, 392, 174, 46, 20, 107, 16, 200, 68, 55, 60, 128, 39, 95, 65, 56,
111, 60, 42, 63, 83, 25, 180, 46, 92, 196, 28, 163, 105, 52, 110, 20,
49, 52, 60, 56, 46, 141, 2, 467, 231, 41, 120, 50, 52, 238, 207, 174,
186, 213, 111, 18, 41, 350, 52, 30, 52, 59, 58, 133, 33, 27, 54, 42,
58, 35, 55, 74, 41, 20, 49, 35, 29, 32, 62, 80, 75, 34, 22, 158, 37,
92, 28, 16, 54, 47, 25, 121, 31, 16, 40, 198, 46, 24, 88, 22, 61, 51,
19, 510, 47, 268, 186, 17, 382, 41, 20, 43, 31, 51, 27, 146, 18, 168,
27, 140, 19, 107, 30, 107, 60, 55, 39, 26, 31, 207, 40, 12, 66, 52,
34, 63, 34, 40, 30, 63, 51, 36, 48, 23, 94, 20, 79, 54, 43, 14, 59,
65, 253, 71, 165, 86, 41, 94, 241, 17, 39, 43, 12, 39, 19, 314, 33,
95, 47, 78, 18, 20, 20, 24, 103, 32, 134, 32, 70, 57, 36, 28, 35, 175,
65, 35, 20, 40, 30, 32, 145, 17, 57, 73, 43, 56, 19, 79, 57, 14, 110,
75, 17, 16, 22, 54, 84, 18, 35, 18, 93, 18, 27, 65, 56, 53, 39, 47,
24, 39, 129, 61, 55, 353, 22, 33, 38, 21, 16, 107, 34, 74, 20, 26, 34,
24, 17, 31, 23, 22, 58, 40, 167, 54, 40, 62, 76, 18, 28, 46, 85, 64,
97, 82, 18, 50, 25, 22, 42, 23, 24, 24, 43, 64, 144, 325, 108, 22, 20,
22, 16, 20, 62, 18, 176, 49, 43, 63, 55, 42, 41, 82, 129, 114, 66, 70,
12, 46, 46, 65, 824, 144, 66, 693, 35, 50, 54, 25, 45, 31, 26, 42, 16,
44, 45, 129, 120, 16, 33, 20, 26, 41, 38, 17, 25, 18, 16, 216, 30, 34,
128, 67, 72, 55, 143, 67, 57, 40, 48, 77, 69, 78, 72, 94, 60, 33, 74,
17, 117, 16, 16, 18, 191, 17, 30, 70, 52, 50, 83, 99, 111, 22, 71, 61,
44, 94, 94, 35, 35, 48, 107, 196, 22, 20, 20, 62, 26, 16, 18, 42, 218,
72, 47, 48, 39, 45, 35, 58, 18, 18, 20, 158, 224, 50, 282, 19, 31, 2,
13, 48, 58, 45, 141, 56, 73, 50, 12, 36, 2, 23, 20, 61, 376, 30, 57,
16, 18, 127, 34, 89, 470, 91, 194, 111, 90, 50, 54, 53, 64, 124, 61,
222, 228, 219, 54, 161, 25, 134, 33, 156, 18, 64, 168, 25, 55, 284,
47, 69, 182, 110, 35, 34, 92, 59, 58, 279, 50, 38, 67, 320, 115, 99,
45, 48, 177, 54, 191, 105, 56, 62, 57, 54, 97, 378, 68, 113, 83, 353,
121, 170, 66, 246, 20, 70, 141, 226, 89, 126, 49, 38, 52, 55, 268, 19,
24, 144, 116, 85, 431, 93, 18, 61, 32, 124, 45, 181, 41, 90, 198, 279,
343, 44, 21, 14, 68, 318, 18, 26, 63, 92, 28, 49, 205, 107, 20, 41,
54, 35, 105, 79, 63, 100, 31, 122, 154, 52, 16, 41, 19, 143, 51, 32,
233, 23, 60, 147, 43, 49, 145, 163, 33, 18, 53, 54, 62, 37, 69, 46,
20, 27, 18, 27, 27, 18, 17, 21, 2, 24, 20, 29, 187, 56, 31, 70, 53,
52, 57, 27, 60, 22, 59, 43, 38, 51, 201, 138, 25, 50, 56, 84, 63, 50,
```

```
44, 41, 158, 228, 108, 67, 27, 34, 39, 118, 234, 62, 88, 55, 18, 25,
22, 43, 44, 28, 65, 19, 17, 17, 16, 23, 24, 25, 67, 28, 77, 130, 38,
17, 100, 52, 43, 27, 18, 35, 62, 36, 57, 28, 16, 35, 58, 31, 40, 23,
24, 88, 16, 51, 189, 60, 23, 119, 23, 60, 18, 24, 74, 157, 50, 13, 62,
35, 55, 29, 30, 46, 54, 16, 31, 31, 22, 22, 21, 22, 23, 22, 107, 28,
31, 49, 39, 119, 133, 19, 32, 74, 33, 38, 50, 104, 90, 42, 46, 74, 55,
37, 363, 19, 45, 47, 17, 39, 71, 62, 18, 38, 44, 16, 18, 30, 39, 22,
39, 18, 84, 37, 46, 210, 473, 40, 57, 50, 53, 166, 57, 32, 21, 39, 37,
107, 166, 53, 58, 24, 50, 66, 23, 20, 154, 57, 74, 124, 20, 178, 30,
18, 93, 57, 72, 49, 44, 15, 16, 33, 2, 20, 58, 193, 13, 41, 348, 21,
32, 73, 118, 28, 54, 66, 184, 46, 64, 25, 18, 26, 50, 120, 164, 24,
23, 29, 25, 34, 31, 27, 32, 16, 36, 83, 17, 16, 386, 138, 66, 53, 21,
68, 41, 28, 55, 30, 51, 40, 14, 59, 58, 32, 19, 26, 57, 35, 266, 28,
16, 308, 52, 18, 53, 136, 97, 16, 32, 64, 20, 133, 67, 55, 41, 99, 60,
14, 55, 302, 42, 59, 173, 97, 96, 46, 77, 58, 168, 134, 59, 346, 67,
117, 299, 174, 179, 173, 58, 276, 29, 34, 30, 35, 27, 17, 34, 20, 30,
67, 39, 38, 38, 19, 27, 40, 55, 24, 15, 22, 15, 63, 24, 27, 41, 28,
60, 227, 16, 17, 15, 17, 14, 18, 39, 61, 21, 140, 174, 116, 39, 205,
32, 42, 85, 224, 54, 48, 46, 73, 17, 53, 31, 62, 28, 79, 63, 59, 37,
46, 26, 22, 18, 41, 26, 32, 17, 13, 33, 36, 62, 424, 100, 58, 33, 31,
79, 42, 353, 47, 92, 117, 47, 82, 14, 18, 18, 121, 96, 62, 42, 58,
182, 58, 26, 60, 43, 122, 18, 61, 83, 141, 242, 49, 127, 67, 50, 107,
56, 40, 402, 177, 95, 175, 10, 20, 176, 12, 109, 19, 49, 35, 14, 66,
159, 18, 144, 178, 210, 91, 150, 78, 95, 119, 26, 199, 101, 32, 80,
12, 99, 48, 35, 18, 18, 21, 103, 16, 32, 32, 50, 58, 67, 16, 74, 50,
208, 282, 65, 47, 65, 54, 2, 140, 47, 134, 55, 55, 27, 53, 25, 181,
15, 16, 121, 43, 381, 67, 60, 39, 65, 31, 59, 43, 73, 41, 50, 187, 28,
87, 24, 33, 22, 45, 27, 49, 69, 183, 20, 62, 21, 125, 33, 74, 14, 48,
121, 39, 35, 51, 16, 488, 107, 16, 77, 43, 231, 33, 29, 204, 52, 32,
16, 66, 20, 31, 384, 20, 34, 20, 39, 41, 55, 27, 235, 20, 31, 31, 65,
18, 37, 64, 276, 18, 108, 90, 18, 23, 68, 253, 33, 66, 134, 56, 131,
99, 160, 57, 45, 25, 44, 67, 20, 36, 30, 29, 159, 34, 59, 41, 21, 197,
107, 91, 45, 111, 39, 43, 54, 22, 26, 102, 332, 25, 34, 210, 25, 36,
62, 382, 60, 63, 21, 148, 400, 63, 156, 171, 50, 242, 236, 70, 20, 16,
364, 244, 27, 112, 59, 63, 450, 51, 42, 67, 151, 197, 103, 204, 100,
402, 26, 67, 365, 61, 184, 62, 83, 128, 264, 20, 122, 20, 68, 107,
111, 166, 41, 113, 46, 64, 88, 106, 128, 107, 131, 429, 38, 62, 396,
19, 49, 19, 48, 106, 386, 25, 74, 321, 35, 112, 51, 170, 10, 16, 30,
18, 37, 60, 95, 35, 108, 149, 64, 89, 29, 20, 50, 42, 161, 40, 34, 47,
30, 21, 83, 24, 208, 151, 56, 43, 129, 478, 65, 198, 51, 142, 56, 144,
18, 124, 108, 45, 44, 83, 60, 162, 271, 117, 57, 233, 235, 84, 226,
42, 38, 35, 19, 58, 35, 106, 170, 20, 59, 111, 209, 52, 47, 47, 41,
117, 145, 119, 20, 120, 54, 36, 98, 114, 166, 18, 38, 35, 60, 38, 25,
40, 19, 261, 49, 50, 102, 41, 70, 136, 107, 54, 69, 214, 62, 17, 17,
35, 97, 62, 239, 18, 73, 29, 21, 27, 20, 47, 279, 48, 59, 106, 50,
150, 144, 111, 58, 22, 102, 91, 130, 37, 82, 182, 93, 54, 274, 28, 69,
99, 123, 268, 67, 18, 62, 37, 39, 47, 69, 136, 61, 137, 22, 167, 69,
32, 152, 52, 80, 64, 66, 142, 60, 48, 61, 45, 142, 35, 55, 107, 102,
77, 103, 20, 104, 155, 265, 18, 22, 18, 18, 20, 99, 102, 43, 47, 59,
```

```
64, 73, 64, 80, 75, 130, 18, 61, 117, 106, 91, 95, 38, 45, 85, 55, 59,
57, 69, 35, 262, 34, 38, 186, 48, 302, 41, 380, 234, 24, 79, 186, 20,
18, 48, 37, 105, 31, 16, 108, 77, 107, 16, 122, 67, 18, 20, 318, 96,
50, 20, 22, 72, 49, 179, 68, 24, 38, 39, 118, 26, 30, 18, 18, 20, 18,
37, 24, 39, 55, 36, 56, 41, 57, 21, 47, 42, 43, 20, 289, 64, 16, 45,
45, 18, 57, 134, 45, 34, 35, 42, 68, 63, 44, 110, 61, 2, 232, 67, 47,
16, 47, 116, 31, 36, 89, 33, 18, 66, 50, 23, 221, 55, 45, 421, 63, 47,
14, 48, 117, 36, 67, 77, 29, 62, 119, 17, 64, 155, 62, 47, 117, 37,
50, 34, 54, 73, 97, 66, 51, 49, 200, 336, 68, 59, 42, 164, 66, 42,
358, 16, 53, 68, 20, 38, 26, 17, 37, 54, 16, 67, 51, 258, 98, 186, 41,
124, 41, 127, 187, 208, 131, 129, 124, 238, 33, 295, 16, 124, 262, 54,
141, 18, 18, 53, 39, 342, 91, 109, 117, 272, 169, 69, 55, 28, 16, 144,
45, 87, 50, 72, 122, 72, 176, 181, 105, 129, 49, 20, 32, 61, 252, 125,
35, 65, 87, 120, 194, 35, 14, 16, 37, 100, 166, 98, 26, 50, 39, 101,
44, 63, 126, 105, 84, 20, 20, 24, 48, 36, 66, 22, 381, 50, 58, 84, 63,
16, 51, 45, 127, 14, 46, 61, 14, 173, 47, 124, 36, 35, 177, 55, 42,
60, 61, 39, 66, 109, 73, 21, 62, 14, 75, 20, 412, 58, 53, 133, 46,
148, 136, 36, 41, 52, 19, 49, 237, 60, 48, 18, 35, 44, 57, 51, 45, 48,
103, 14, 16, 71, 20, 24, 52, 66, 29, 231, 64, 33, 59, 105, 42, 36,
231, 115, 36, 57, 60, 48, 48, 184, 60, 116, 168, 16, 69, 25, 148, 55,
20, 36, 59, 42, 27, 27, 34, 34, 128, 82, 475, 119, 53, 35, 35, 117,
49, 45, 31, 48, 29, 18, 19, 18, 30, 14, 46, 20, 36, 180, 56, 32, 32,
64, 389, 71, 39, 300, 20, 121, 71, 75, 281, 314, 53, 20, 183, 76, 116,
269, 129, 168, 87, 20, 31, 44, 113, 20, 175, 180, 118, 16, 138, 20,
67, 147, 111, 251, 119, 245, 84, 41, 48, 22, 20, 150, 47, 24, 45, 48,
16, 36, 26, 28, 60, 36, 34, 38, 56, 25, 49, 59, 42, 23, 20, 70, 64,
68, 39, 33, 81, 161, 56, 20, 157, 41, 47, 225, 305, 14, 39, 36, 35,
53, 37, 55, 53, 21, 18, 20, 31, 26, 178, 46, 156, 25, 16, 46, 28, 50,
130, 65, 33, 59, 19, 18, 172, 58, 51, 287, 59, 67, 35, 74, 35, 116,
42, 54, 105, 49, 103, 29, 62, 85, 27, 25, 59, 76, 58, 28, 92, 57, 65,
28, 73, 16, 22, 33, 233, 42, 100, 237, 94, 85, 69, 42, 110, 33, 51,
20, 27, 59, 45, 23, 49, 160, 154, 50, 16, 34, 218, 57, 175, 31, 81,
37, 54, 14, 45, 88, 39, 60, 19, 40, 224, 20, 76, 273, 36, 40, 18, 59,
110, 229, 177, 88, 30, 67, 63, 36, 161, 18, 23, 210, 34, 29, 55, 102,
63, 53, 16, 123, 87, 16, 44, 52, 100, 256, 297, 64, 16, 61, 90, 22,
41, 49, 114, 62, 169, 34, 18, 45, 53, 60, 274, 99, 161, 300, 54, 18,
33, 14, 16, 14, 98, 82, 75, 14, 18, 14, 59, 42, 71, 64, 132, 297, 20,
168, 64, 44, 22, 17, 61, 143, 33, 28, 67, 18, 68, 26, 70, 61, 67, 29,
47, 18, 43, 20, 115, 41, 33, 42, 170, 43, 113, 72, 49, 37, 295, 78,
33, 44, 14, 23, 42, 40, 132, 36, 44, 2, 28, 131, 72, 42, 28, 33, 93,
51, 53, 210, 176, 32, 27, 42, 74, 49, 91, 39, 2, 33, 17, 33, 53, 40,
22, 23, 307, 66, 380, 59, 140, 16, 32, 67, 31, 260, 38, 47, 22, 29,
56, 32, 67, 63, 20, 52, 60, 86, 28, 121, 18, 80, 38, 18, 115, 29, 18,
92, 52, 84, 24, 58, 125, 37, 214, 33, 46, 63, 94, 14, 56, 14, 53, 65,
28, 29, 43, 122, 21, 27, 33, 22, 32, 51, 24, 28, 29, 27, 30, 24, 43,
86, 46, 18, 166, 64, 38, 34, 21, 134, 42, 110, 100, 39, 125, 65, 35,
38, 65, 30, 126, 40, 40, 26, 55, 41, 34, 60, 29, 56, 40, 152, 30, 58,
22, 50, 43, 46, 58, 37, 46, 113, 63, 177, 283, 43, 125, 63, 46, 47,
119, 28, 33, 35, 112, 38, 38, 36, 58, 34, 58, 29, 47, 50, 60, 49, 34,
```

```
34, 390, 36, 42, 233, 16, 64, 131, 40, 45, 33, 79, 36, 67, 89, 66, 72,
122, 30, 52, 22, 56, 20, 35, 39, 31, 34, 23, 71, 37, 29, 40, 28, 28,
65, 44, 36, 41, 59, 70, 74, 105, 79, 28, 104, 264, 98, 14, 66, 28, 48,
54, 49, 35, 117, 43, 54, 34, 80, 20, 42, 44, 48, 45, 48, 88, 58, 48,
55, 32, 40, 38, 25, 64, 50, 23, 67, 89, 16, 67, 16, 142, 57, 91, 24,
68, 45, 30, 61, 37, 110, 54, 69, 58, 39, 18, 37, 29, 35, 44, 278, 52,
30, 45, 18, 22, 102, 127, 12, 50, 434, 52, 43, 26, 75, 87, 39, 109,
159, 55, 23, 93, 33, 31, 149, 130, 40, 43, 22, 55, 99, 53, 49, 28, 27,
21, 44, 22, 35, 123, 39, 31, 70, 22, 75, 114, 31, 50, 111, 115, 48,
59, 34, 31, 41, 38, 44, 16, 16, 257, 429, 146, 94, 31, 152, 93, 27,
61, 57, 40, 115, 76, 47, 47, 50, 45, 22, 95, 41, 71, 116, 62, 97, 159,
48, 49, 55, 46, 43, 36, 20, 74, 113, 22, 35, 28, 53, 28, 53, 54, 86,
22, 28, 18, 46, 41, 72, 22, 65, 35, 22, 84, 45, 29, 84, 41, 27, 47,
54, 2, 123, 76, 21, 35, 55, 133, 105, 14, 42, 273, 20, 271, 44, 26,
239, 32, 37, 35, 61, 65, 27, 66, 60, 70, 55, 22, 56, 27, 30, 24, 28,
82, 33, 33, 34, 36, 136, 36, 29, 53, 67, 48, 39, 2, 16, 63, 16, 22,
85, 49, 52, 65, 40, 20, 158, 135, 39, 2, 16, 37, 47, 57, 52, 33, 87,
21, 40, 44, 56, 32, 47, 53, 52, 160, 18, 53, 52, 61, 334, 19, 28, 27,
45, 27, 85, 45, 76, 43, 52, 20, 114, 18, 66, 95, 18, 52, 37, 34, 22,
18, 36, 108, 40, 18, 43, 52, 18, 19, 45, 28, 27, 23, 40, 71, 35, 37,
37, 83, 30, 116, 121, 25, 28, 18, 18, 35, 36, 103, 23, 2, 41, 51, 69,
41, 39, 45, 37, 49, 30, 63, 18, 38, 32, 42, 51, 48, 49, 22, 73, 91,
81, 18, 94, 213, 19, 30, 21, 40, 16, 46, 171, 88, 28, 32, 51, 40, 67,
31, 29, 18, 92, 299, 43, 49, 20, 54, 32, 47, 123, 52, 34, 139, 36, 42,
23, 82, 46, 74, 42, 22, 54, 56, 48, 65, 99, 172, 67, 12, 69, 53, 42,
16, 137, 111, 57, 61, 68, 63, 29, 21, 28, 57, 47, 143, 54, 112, 35,
56, 53, 146, 212, 130, 66, 26, 33, 18, 152, 45, 20, 131, 32, 143, 29,
62, 119, 76, 44, 54, 63, 46, 104, 42, 37, 80, 16, 75, 85, 24, 43, 38,
37, 53, 53, 14, 204, 18, 40, 30, 34, 46, 44, 63, 20, 48, 83, 70, 50,
18, 60, 55, 36, 143, 60, 55, 16, 118, 16, 111, 203, 61, 145, 72, 16,
22, 42, 25, 22, 20, 421, 41, 53, 78, 106, 71, 16, 156, 77, 118, 67,
66, 494, 87, 14, 114, 92, 41, 66, 56, 196, 47, 109, 149, 67, 119, 36,
91, 189, 390, 104, 38, 38, 18, 114, 240, 181, 42, 114, 119, 132, 58,
124, 270, 47, 195, 58, 274, 134, 56, 79, 57, 20, 115, 43, 87, 81, 65,
43, 37, 57, 48, 53, 37, 18, 31, 66, 55, 80, 24, 118, 55, 63, 195, 22,
32, 23, 61, 44, 34, 16, 49, 14, 20, 55, 56, 38, 33, 74, 108, 17, 14,
17, 289, 22, 61, 419, 101, 53, 25, 26, 30, 52, 15, 31, 258, 17, 49,
99, 35, 58, 39, 26, 2, 39, 133, 30, 83, 239, 77, 22, 17, 87, 90, 28,
70, 23, 35, 43, 169, 274, 43, 43, 27, 54, 105, 2, 77, 99, 61, 73, 45,
20, 20, 52, 35, 41, 39, 16, 34, 59, 222, 23, 39, 49, 57, 43, 178, 20,
24, 24, 30, 29, 53, 37, 16, 230, 177, 114, 180, 31, 57, 2, 57, 65, 22,
36, 38, 139, 45, 65, 123, 22, 125, 52, 129, 142, 84, 31, 62, 28, 271,
61, 238, 24, 140, 23, 19, 78, 63, 20, 16, 23, 56, 363, 71, 49, 316,
62, 170, 18, 110, 28, 16, 171, 88, 453, 128, 132, 16, 67, 24, 65, 20,
117, 130, 53, 42, 307, 244, 20, 18, 182, 85, 2, 54, 38, 131, 62, 414,
60, 66, 303, 127, 39, 54, 62, 97, 46, 53, 47, 14, 39, 106, 27, 52, 32,
95, 130, 12, 31, 77, 128, 195, 103, 477, 151, 65, 85, 22, 88, 63, 393,
50, 74, 93, 56, 113, 173, 22, 60, 249, 14, 14, 55, 56, 150, 351, 155,
48, 76, 164, 18, 224, 33, 57, 32, 22, 25, 78, 42, 332, 16, 194, 16,
```

```
87, 42, 16, 18, 219, 58, 167, 47, 14, 53, 70, 57, 50, 79, 28, 34, 59,
17, 45, 72, 81, 68, 17, 98, 63, 38, 59, 215, 41, 33, 52, 109, 36, 22,
474, 35, 25, 97, 44, 53, 116, 163, 16, 99, 256, 59, 16, 18, 64, 148,
38, 24, 16, 109, 61, 40, 127, 28, 21, 37, 160, 39, 45, 31, 39, 54,
205, 80, 77, 90, 18, 66, 21, 109, 162, 40, 44, 18, 334, 38, 99, 43,
17, 107, 102, 267, 45, 169, 59, 191, 111, 50, 63, 40, 111, 29, 16, 30,
100, 150, 33, 26, 18, 16, 55, 255, 87, 14, 63, 14, 100, 61, 32, 20,
361, 59, 42, 33, 38, 29, 17, 49, 37, 22, 84, 196, 155, 29, 29, 34, 39,
30, 53, 61, 355, 143, 34, 62, 170, 146, 327, 55, 201, 31, 159, 159,
47, 93, 72, 151, 109, 277, 224, 126, 236, 17, 14, 18, 10, 20, 54, 12,
14, 82, 17, 54, 124, 75, 111, 2, 108, 120, 207, 54, 18, 80, 20, 47,
95, 18, 119, 59, 454, 109, 16, 207, 147, 294, 145, 37, 33, 125, 106,
175, 233, 102, 52, 62, 36, 62, 49, 42, 83, 122, 104, 66, 110, 90, 44,
247, 14, 170, 65, 174, 51, 49, 51, 158, 170, 86, 162, 68, 71, 73, 62,
94, 25, 81, 113, 35, 101, 410, 95, 65, 123, 131, 82, 65, 52, 454, 170,
124, 277, 60, 518, 111, 53, 280, 40, 33, 49, 82, 56, 54, 38, 190, 18,
130, 30, 166, 46, 145, 68, 328, 74, 18, 84, 26, 64, 132, 126, 112, 92,
276, 61, 47, 24, 36, 32, 141, 73, 55, 63, 26, 68, 45, 29, 21, 91, 38,
32, 26, 55, 26, 18, 54, 37, 93, 52, 32, 45, 16, 45, 68, 22, 92, 59,
38, 85, 67, 50, 36, 16, 49, 80, 27, 30, 90, 37, 69, 25, 26, 256, 16,
59, 296, 68, 87, 337, 38, 49, 56, 22, 42, 29, 98, 77, 21, 28, 54, 19,
62, 26, 56, 497, 98, 44, 35, 102, 25, 264, 45, 102, 225, 46, 28, 34,
37, 34, 65, 34, 73, 42, 22, 75, 48, 100, 36, 22, 48, 53, 50, 72, 43,
72, 65, 46, 39, 357, 93, 230, 16, 212, 57, 16, 63, 29, 34, 50, 117,
21, 53, 156, 101, 41, 93, 47, 46, 29, 59, 55, 55, 71, 68, 48, 41, 43,
73, 77, 32, 18, 61, 15, 47, 47, 56, 65, 121, 29, 18, 58, 80, 204, 220,
69, 41, 539, 63, 36, 67, 26, 18, 20, 83, 68, 44, 18, 52, 171, 56, 95,
212, 66, 45, 64, 75, 84, 40, 47, 45, 21, 30, 16, 228, 74, 121, 126,
64, 399, 126, 35, 20, 20, 37, 24, 32, 189, 62, 159, 16, 59, 210, 110,
194, 22, 19, 27, 2, 16, 131, 130, 14, 112, 14, 38, 68, 19, 54, 128,
36, 86, 300, 342, 104, 62, 164, 85, 29, 73, 45, 98, 25, 96, 50, 395,
16, 69, 55, 101, 259, 47, 50, 169, 65, 55, 53, 366, 63, 24, 106, 51,
104, 818, 425, 55, 47, 2, 41, 55, 133, 125, 27, 345, 162, 159, 53,
126, 36, 55, 66, 174, 99, 93, 36, 70, 64, 131, 155, 57, 34, 58, 20,
123, 151, 67, 203, 40, 59, 66, 58, 76, 189, 41, 17, 16, 80, 115, 28,
74, 32, 19, 34, 111, 127, 14, 165, 254, 20, 90, 168, 53, 39, 55, 58,
58, 34, 31, 37, 128, 58, 63, 73, 36, 80, 86, 111, 210, 296, 50, 53,
36, 30, 35, 20, 112, 152, 111, 48, 162, 21, 238, 169, 44, 57, 68, 77,
51, 57, 109, 23, 56, 51, 29, 84, 60, 107, 26, 58, 51, 79, 73, 59, 16,
69, 93, 126, 161, 105, 136, 32, 26, 86, 50, 162, 76, 68, 55, 96, 35,
140, 51, 76, 35, 79, 20, 123, 20, 20, 16, 22, 36, 107, 41, 21, 2, 82,
54, 26, 26, 83, 85, 56, 44, 36, 58, 57, 42, 140, 52, 44, 91, 48, 79,
175, 58, 65, 82, 61, 45, 89, 68, 20, 48, 40, 47, 24, 26, 20, 33, 75,
22, 30, 308, 16, 425, 202, 267, 28, 20, 105, 287, 30, 15, 24, 22, 486,
97, 44, 64, 70, 18, 47, 66, 39, 71, 146, 78, 51, 69, 31, 34, 21, 175,
104, 96, 47, 58, 106, 65, 57, 158, 46, 27, 30, 45, 35, 14, 128, 67,
67, 123, 56, 28, 20, 68, 41, 62, 46, 23, 27, 15, 17, 159, 114, 20, 50,
27, 142, 12, 128, 42, 101, 49, 27, 142, 52, 143, 18, 14, 16, 14, 14,
16, 21, 51, 29, 20, 20, 34, 92, 67, 16, 29, 82, 63, 49, 2, 47, 38,
```

```
167, 68, 163, 18, 107, 39, 44, 58, 122, 60, 30, 58, 14, 194, 141, 16,
59, 30, 28, 22, 22, 68, 55, 40, 18, 42, 31, 64, 18, 18, 26, 77, 48,
39, 24, 18, 15, 46, 20, 14, 53, 86, 61, 59, 400, 98, 69, 16, 183, 42,
60, 64, 36, 74, 67, 21, 30, 33, 58, 26, 20, 23, 263, 18, 47, 273, 59,
84, 44, 449, 61, 20, 47, 346, 445, 47, 57, 16, 42, 49, 18, 274, 16,
103, 22, 442, 189, 28, 117, 16, 397, 41, 47, 215, 57, 47, 55, 39, 54,
16, 64, 74, 66, 86, 59, 91, 46, 56, 75, 41, 35, 215, 51, 120, 124, 34,
181, 20, 96, 22, 68, 93, 301, 174, 28, 48, 43, 57, 64, 104, 80, 96,
89, 210, 55, 72, 124, 32, 97, 104, 41, 41, 16, 68, 74, 60, 128, 43,
244, 23, 153, 57, 18, 61, 68, 18, 61, 65, 16, 54, 68, 285, 57, 59,
150, 61, 97, 150, 73, 90, 41, 54, 42, 82, 34, 16, 41, 20, 37, 17, 36,
47, 86, 63, 29, 334, 52, 83, 125, 22, 49, 24, 56, 195, 15, 60, 54, 24,
26, 186, 20, 86, 19, 47, 23, 20, 62, 18, 146, 77, 63, 129, 26, 20, 64,
78, 110, 39, 57, 66, 177, 27, 37, 28, 16, 22, 48, 119, 32, 52, 61, 55,
140, 13, 44, 25, 20, 39, 56, 54, 83, 75, 25, 405, 36, 25, 53, 20, 64,
116, 26, 407, 21, 43, 32, 14, 16, 29, 20, 49, 43, 59, 68, 18, 51, 20,
17, 39, 38, 166, 33, 53, 68, 63, 29, 114, 102, 69, 58, 75, 61, 31, 14,
21, 29, 69, 76, 93, 19, 80, 47, 42, 166, 58, 54, 105, 198, 39, 42, 59,
31, 48, 28, 29, 63, 158, 48, 19, 43, 134, 45, 19, 25, 90, 35, 18, 174,
168, 112, 26, 131, 15, 190, 25, 97, 39, 83, 459, 116, 18, 12, 110,
241, 52, 52, 114, 32, 86, 28, 43, 22, 202, 71, 34, 330, 79, 33, 19,
44, 81, 25, 34, 35, 21, 80, 84, 39, 20, 119, 48, 37, 34, 19, 15, 54,
25, 45, 76, 398, 154, 51, 70, 44, 221, 21, 47, 26, 354, 404, 26, 56,
23, 71, 31, 67, 21, 26, 88, 71, 35, 15, 22, 28, 17, 66, 47, 42, 58,
43, 38, 21, 144, 22, 130, 16, 85, 39, 29, 63, 46, 111, 16, 161, 46,
20, 33, 67, 21, 18, 48, 52, 38, 50, 122, 22, 12, 86, 26, 36, 20, 43,
39, 40, 76, 37, 25, 22, 16, 125, 51, 54, 68, 20, 34, 61, 25, 76, 135,
36, 96, 59, 17, 54, 64, 129, 33, 22, 281, 54, 68, 17, 223, 57, 52, 42,
25, 44, 55, 55, 16, 26, 297, 50, 63, 34, 33, 2, 58, 2, 355, 46, 38,
69, 77, 41, 47, 31, 38, 114, 21, 105, 19, 131, 44, 300, 160, 172, 310,
61, 134, 41, 89, 115, 20, 119, 80, 237, 20, 309, 60, 20, 60, 49, 113,
76, 116, 233, 18, 137, 172, 81, 59, 108, 55, 114, 37, 259, 16, 119,
61, 63, 49, 15, 72, 147, 333, 390, 18, 200, 69, 70, 147, 10, 29, 142,
53, 130, 123, 147, 23, 84, 39, 57, 179, 105, 61, 40, 29, 153, 142, 50,
20, 68, 114, 167, 147, 45, 105, 43, 64, 16, 63, 41, 47, 52, 65, 28,
39, 41, 20, 21, 53, 16, 52, 16, 63, 34, 34, 313, 34, 288, 19, 31, 59,
50, 58, 16, 51, 96, 110, 39, 21, 19, 39, 18, 57, 18, 97, 29, 54, 71,
60, 20, 55, 69, 113, 74, 34, 69, 38, 25, 38, 64, 105, 108, 35, 77, 16,
39, 34, 71, 14, 30, 12, 39, 56, 114, 14, 32, 16, 252, 63, 34, 205, 72,
28, 66, 126, 68, 60, 135, 47, 84, 61, 56, 28, 97, 172, 73, 36, 58, 65,
50, 64, 59, 59, 111, 107, 129, 17, 212, 149, 92, 96, 106, 281, 150,
45, 16, 59, 81, 169, 41, 242, 309, 55, 25, 61, 189, 27, 39, 35, 115,
114, 16, 383, 52, 22, 39, 33, 350, 18, 196, 57, 62, 59, 47, 390, 93,
58, 145, 39, 42, 64, 12, 44, 30, 74, 147, 38, 58, 429, 197, 91, 66,
26, 23, 102, 94, 32, 101, 26, 55, 29, 39, 112, 70, 143, 19, 289, 52,
45, 2, 63, 70, 63, 329, 27, 37, 69, 53, 42, 16, 47, 12, 32, 30, 29,
65, 171, 43, 49, 22, 26, 48, 25, 334, 39, 141, 53, 48, 29, 17, 24, 57,
50, 82, 138, 131, 49, 74, 62, 62, 22, 130, 88, 38, 299, 60, 24, 42,
17, 69, 110, 19, 159, 104, 50, 318, 79, 64, 45, 78, 92, 56, 84, 54,
```

16, 43, 134, 55, 20, 57, 30, 148, 116, 103, 50, 31, 25, 27, 78, 49,
26, 35, 64, 150, 60, 38, 122, 401, 73, 446, 53, 15, 378, 61, 44, 53,
40, 41, 206, 60, 32, 47, 374, 122, 46, 48, 15, 18, 54, 76, 61, 52, 57,
49, 38, 32, 23, 30, 32, 15, 449, 70, 38, 65, 489, 17, 12, 202, 112,
16, 41, 36, 166, 18, 68, 17, 23, 58, 55, 49, 20, 67, 2, 154, 66, 44,
289, 2, 37, 85, 57, 32, 53, 84, 21, 70, 50, 36, 45, 32, 277, 50, 54,
88, 125, 44, 118, 33, 67, 44, 54, 50, 61, 26, 34, 37, 62, 2, 33, 62,
43, 19, 16, 31, 39, 106, 57, 57, 43, 18, 17, 16, 193, 22, 76, 38, 82,
16, 30, 137, 377, 121, 137, 122, 57, 338, 184, 19, 508, 68, 297, 191,
312, 89, 36, 64, 118, 2, 66, 13, 51, 2, 16, 122, 44, 16, 54, 112, 58,
72, 22, 65, 56, 56, 57, 47, 16, 28, 56, 45, 15, 16, 91, 18, 37, 16,
104, 481, 24, 90, 37, 65, 63, 85, 66, 22, 13, 36, 86, 302, 122, 43,
62, 112, 58, 44, 36, 67, 79, 156, 52, 141, 35, 132, 39, 39, 27, 101,
35, 245, 116, 82, 17, 73, 44, 204, 86, 23, 16, 89, 30, 25, 41, 22, 38,
28, 34, 35, 59, 67, 60, 94, 239, 64, 29, 48, 49, 90, 135, 20, 20, 38,
43, 166, 85, 33, 45, 26, 17, 14, 46, 20, 13, 20, 16, 68, 35, 93, 43,
109, 176, 49, 46, 53, 127, 20, 105, 14, 60, 159, 158, 210, 67, 59, 23,
49, 22, 126, 418, 35, 61, 150, 18, 136, 171, 68, 58, 92, 46, 33, 41,
45, 37, 16, 18, 172, 97, 75, 14, 112, 67, 15, 16, 33, 18, 237, 57,
146, 53, 167, 21, 14, 28, 40, 20, 16, 65, 43, 32, 18, 283, 18, 16, 64,
110, 218, 138, 191, 16, 194, 28, 49, 96, 42, 63, 38, 20, 45, 64, 80,
51, 17, 35, 148, 193, 70, 36, 17, 128, 35, 86, 90, 22, 64, 22, 54, 12,
66, 22, 68, 33, 53, 57, 263, 73, 17, 33, 66, 39, 30, 237, 50, 113, 66,
36, 18, 23, 72, 91, 69, 62, 23, 112, 99, 46, 77, 53, 78, 20, 16, 27,
36, 17, 64, 31, 61, 49, 32, 17, 15, 16, 17, 17, 39, 238, 28, 24, 65,
35, 95, 47, 29, 16, 33, 31, 36, 21, 50, 37, 109, 60, 35, 51, 67, 75,
42, 44, 44, 2, 173, 131, 19, 58, 105, 2, 2, 17, 189, 41, 117, 308, 50,
2, 18, 74, 2, 15, 57, 28, 90, 51, 34, 23, 91, 116, 129, 53, 17, 31,
19, 191, 41, 346, 112, 264, 45, 18, 52, 207, 30, 24, 50, 20, 123, 18,
43, 25, 39, 25, 15, 40, 170, 115, 19, 119, 38, 50, 18, 35, 41, 24, 40,
19, 76, 22, 29, 26, 24, 16, 61, 33, 62, 17, 66, 330, 16, 32, 60, 58,
17, 17, 105, 269, 49, 17, 81, 231, 297, 70, 50, 29, 27, 16, 15, 27,
98, 68, 51, 61, 117, 61, 432, 60, 291, 43, 65, 17, 59, 128, 74, 54,
28, 92, 55, 64, 40, 38, 272, 327, 56, 29, 49, 37, 104, 63, 41, 17, 75,
55, 173, 46, 38, 36, 87, 41, 64, 30, 16, 36, 20, 86, 23, 98, 18, 35,
131, 54, 42, 26, 15, 42, 51, 19, 42, 58, 38, 59, 152, 130, 58, 27, 37,
55, 30, 77, 42, 414, 51, 43, 13, 93, 43, 107, 63, 150, 160, 88, 16,
63, 204, 16, 151, 17, 223, 18, 47, 83, 295, 36, 145, 118, 56, 65, 168,
127, 38, 35, 52, 82, 41, 2, 45, 196, 34, 34, 15, 58, 50, 20, 50, 20,
129, 141, 64, 183, 17, 40, 103, 112, 277, 35, 112, 18, 17, 70, 22,
151, 253, 43, 45, 218, 266, 60, 19, 73, 64, 14, 15, 28, 52, 57, 457,
31, 105, 147, 24, 79, 44, 69, 142, 37, 26, 18, 30, 18, 75, 118, 21,
282, 95, 62, 57, 26, 35, 47, 16, 58, 41, 120, 89, 85, 63, 55, 287, 22,
125, 76, 65, 55, 107, 210, 127, 53, 121, 42, 14, 52, 44, 92, 43, 182,
415, 96, 50, 366, 29, 82, 55, 51, 172, 68, 48, 136, 120, 41, 96, 111,
85, 113, 58, 273, 17, 165, 24, 512, 138, 46, 34, 231, 80, 96, 29, 48,
65, 93, 51, 40, 20, 18, 16, 51, 269, 148, 93, 199, 276, 41, 191, 67,
24, 45, 47, 59, 56, 29, 175, 58, 89, 37, 49, 20, 58, 81, 131, 45, 19,
73, 30, 363, 56, 18, 38, 30, 50, 54, 32, 113, 16, 118, 38, 41, 297,

```
309, 18, 20, 68, 114, 39, 106, 160, 16, 437, 35, 83, 82, 22, 36, 73,
27, 26, 44, 115, 23, 59, 158, 56, 52, 273, 56, 218, 50, 221, 22, 44,
174, 98, 100, 41, 16, 33, 75, 17, 38, 46, 16, 19, 58, 18, 54, 2, 17,
20, 34, 166, 20, 20, 79, 80, 22, 59, 14, 16, 20, 50, 16, 94, 49, 45,
140, 38, 55, 63, 124, 69, 393, 92, 22, 119, 67, 259, 36, 73, 16, 94,
17, 48, 17, 18, 55, 94, 18, 316, 86, 68, 37, 138, 54, 407, 27, 69, 25,
50, 17, 30, 17, 41, 41, 14, 112, 97, 47, 167, 281, 40, 76, 227, 39,
69, 39, 51, 35, 61, 365, 239, 19, 115, 70, 34, 34, 187, 36, 51, 52,
100, 115, 2, 46, 38, 33, 126, 37, 107, 36, 160, 34, 52, 116, 68, 58,
281, 18, 24, 26, 37, 182, 18, 71, 18, 38, 54, 20, 44, 16, 16, 15, 54,
36, 17, 17, 17, 251, 70, 278, 44, 367, 166, 16, 182, 263, 46, 88, 264,
266, 68, 66, 57, 352, 146, 18, 72, 32, 60, 374, 23, 44, 127, 108, 15,
36, 65, 17, 17, 31, 49, 28, 20, 75, 115, 30, 56, 18, 56, 203, 51, 91,
32, 50, 57, 356, 49, 73, 35, 101, 68, 33, 18, 20, 20, 57, 126, 130,
57, 192, 43, 25, 193, 18, 54, 20, 24, 113, 51, 311, 114, 81, 52, 296,
85, 37, 25, 214, 137, 15, 64, 47, 28, 96, 21, 131, 111, 17, 97, 17,
20, 73, 249, 17, 99, 244, 104, 60, 246, 67, 59, 33, 37, 114, 580, 252,
47, 92, 52, 30, 47, 59, 66, 45, 186, 68, 73, 69, 52, 55, 141, 42, 112,
163, 25, 42, 57, 46, 47, 135, 324, 52, 92, 41, 21, 119, 48, 29, 38,
16, 40, 15, 57, 47, 29, 63, 41, 59, 66, 21, 49, 19, 28, 22, 28, 15,
48, 18, 79, 30, 16, 16, 94, 18, 177, 18, 92, 67, 45, 30, 63, 15, 59,
36, 37, 59, 469, 53, 102, 16, 14, 31, 19, 82, 2, 83, 36, 123, 46, 76,
50, 20, 44, 180, 90, 55, 40, 76, 42, 47, 54, 46, 48, 50, 27, 53, 18,
76, 36, 156, 37, 114, 73, 31, 129, 55, 293, 38, 44, 49, 56, 207, 55,
60, 40, 101, 69, 47, 27, 2, 397, 86, 65, 56, 249, 49, 91, 79, 166, 20,
148, 97, 97, 50, 85, 180, 63, 35, 27, 35, 21, 33, 66, 32, 86, 50, 100,
147, 62, 103, 46, 17, 16, 16, 413, 149, 25, 18, 49, 171, 16, 84, 15,
76, 83, 190, 254, 78, 12, 65, 18, 168, 28, 20, 202, 119, 78, 43, 15,
70, 62, 22, 63, 46, 77, 98, 29, 20, 33, 63, 257, 18, 389, 35, 23, 53,
39, 57, 46, 31, 81, 17, 32, 72, 58, 61, 74, 47, 52, 204, 41, 36, 20,
37, 15, 23, 50, 77, 43, 16, 81, 64, 50, 52, 55, 64, 66, 57, 62, 15,
18, 63, 153, 19, 111, 127, 56, 62, 37, 81, 58, 222, 269, 19, 32, 17,
29, 59, 47, 25, 204, 55, 16, 14, 119, 50, 56, 30, 95, 152, 56, 50, 47,
48, 31, 56, 148, 126, 127, 51, 56, 35, 35, 36, 37, 57, 227, 49, 208,
102, 50, 128, 249, 125, 111, 49, 41, 48, 399, 18, 64, 69, 37, 45, 78,
70, 88, 95, 47, 180, 34, 20, 58, 69, 50, 20, 58, 28, 79, 78, 228, 45,
113, 63, 345, 250, 358, 337, 330, 57, 148, 128, 112, 204, 179, 16, 68,
20, 51, 93, 119, 38, 18, 33, 58, 61, 48, 68, 248, 26, 81, 57, 187, 31,
22, 14, 284, 63, 389, 34, 111, 81, 57, 62, 137, 123, 26, 23, 174, 246,
42, 49, 18, 196, 37, 171, 18, 16, 69, 127, 45, 36, 108, 117, 63, 47,
45, 53, 61, 93, 78, 47, 56, 45, 433, 118, 278, 16, 273, 170, 55, 69,
48, 59, 93, 16, 66, 59, 128, 66, 54, 64, 18, 131, 193, 51, 39, 63, 97,
63, 22, 142, 52, 84, 61, 59, 196, 111, 16, 141, 20, 75, 36, 72, 62,
27, 114, 96, 60, 50, 54, 75, 39, 14, 97, 46, 18, 82, 175, 42, 144, 58,
69, 55, 69, 46, 47, 64, 49, 64, 176, 49, 121, 102, 36, 48, 35, 34, 69,
16, 52, 64, 210, 38, 158, 20, 116, 64, 37, 43, 29, 41, 22, 46, 214,
59, 61, 50, 271, 130, 136, 110, 55, 64, 49, 80, 106, 107, 71, 228, 91,
263, 40, 101, 44, 196, 16, 144, 41, 85, 16, 147, 46, 192, 68, 18, 20,
51, 91, 43, 71, 66, 22, 87, 61, 50, 61, 63, 86, 117, 173, 18, 65, 46,
```

```
107, 91, 118, 58, 64, 61, 22, 45, 24, 79, 28, 62, 47, 46, 108, 167,
107, 20, 53, 56, 20, 20, 111, 12, 117, 22, 50, 18, 139, 116, 22, 27,
54, 141, 65, 21, 73, 17, 99, 73, 330, 125, 30, 275, 56, 59, 78, 60,
16, 90, 47, 202, 156, 67, 52, 39, 48, 18, 349, 81, 253, 52, 33, 217,
21, 102, 65, 51, 183, 230, 104, 97, 109, 447, 18, 110, 57, 121, 63,
189, 188, 111, 55, 104, 111, 61, 47, 107, 69, 49, 42, 54, 88, 40, 52,
16, 46, 105, 14, 58, 57, 144, 419, 72, 190, 72, 90, 173, 97, 29, 12,
255, 64, 105, 91, 274, 190, 59, 73, 88, 198, 47, 110, 66, 123, 111,
39, 481, 314, 18, 126, 163, 322, 81, 184, 82, 65, 92, 104, 84, 159,
51, 70, 268, 320, 228, 39, 59, 18, 337, 58, 55, 159, 35, 41, 39, 31,
16, 45, 52, 37, 18, 161, 139, 22, 124, 319, 210, 31, 62, 64, 33, 63,
18, 17, 114, 61, 29, 51, 97, 73, 65, 20, 18, 42, 146, 300, 22, 86,
335, 38, 18, 47, 52, 31, 104, 50, 34, 107, 44, 54, 29, 72, 29, 88, 52,
67, 249, 159, 17, 27, 16, 30, 84, 61, 50, 62, 28, 35, 55, 45, 49, 17,
42, 71, 32, 29, 69, 74, 69, 55, 53, 33, 35, 107, 52, 80, 32, 103, 49,
61, 68, 146, 42, 33, 315, 60, 43, 61, 18, 33, 18, 37, 44, 29, 96, 64,
14, 82, 107, 35, 70, 108, 117, 37, 52, 16, 91, 150, 27, 28, 454, 16,
15, 15, 20, 93, 64, 66, 52, 20, 16, 100, 52, 153, 41, 14, 28, 43, 45,
16, 18, 37, 59, 180, 50, 96, 64, 154, 457, 35, 101, 67, 60, 18, 18,
63, 52, 52, 146, 46, 16, 66, 21, 49, 53, 63, 52, 66, 54, 44, 216, 18,
12, 36, 88, 48, 30, 17, 47, 107, 266, 46, 41, 151, 254, 53, 81, 28,
84, 39, 16, 20, 41, 74, 90, 56, 62, 67, 98, 121, 60, 43, 18, 14, 36,
76, 159, 84, 88, 143, 70, 75, 26, 153, 72, 61, 69, 121, 14, 37, 15,
68, 18, 55, 39, 44, 104, 151, 47, 290, 63, 63, 56, 28, 170, 17, 59,
58, 77, 50, 57, 54, 45, 29, 105, 133, 56, 16, 24, 38, 42, 14, 18, 49,
28, 57, 16, 33, 158, 14, 78, 335, 119, 16, 236, 45, 29, 34, 34, 176,
65, 86, 119, 34, 76, 58, 52, 18, 54, 285, 83, 106, 50, 57, 272, 162,
64, 26, 92, 39, 19, 330, 232, 131, 60, 129, 20, 57, 53, 39, 52, 131,
35, 57, 117, 115, 163, 47, 109, 449, 25, 50, 185, 64, 126, 57, 172,
120, 22, 202, 18, 112, 129, 33, 50, 492, 113, 26, 296, 116, 105, 56,
30, 92, 107, 51, 244, 55, 33, 54, 20, 73, 94, 57, 32, 27, 59, 371, 51,
108, 50, 114, 70, 59, 17, 52, 239, 56, 23, 212, 79, 97, 62, 126, 20,
52, 65, 68, 108, 49, 117, 185, 131, 63, 177, 125, 58, 57, 52, 26, 25,
155, 18, 87, 66, 112, 16, 36, 20, 208, 87, 48, 208, 33, 129, 35, 65,
262, 40, 60, 14, 404, 58, 41, 2, 73, 46, 76, 110, 299, 40, 15, 16, 26,
22, 87, 31, 20, 38, 21, 53, 14, 28, 82, 59, 52, 77, 48, 116, 84, 31,
54, 17, 62, 64, 56, 58, 29, 97, 30, 272, 85, 17, 130, 18, 62, 311,
152, 41, 32, 48, 47, 68, 15, 15, 16, 228, 60, 60, 16, 46, 18, 21, 115,
26, 241, 339, 28, 50, 28, 123, 22, 20, 18, 24, 14, 16, 16, 68, 22,
206, 449, 154, 43, 18, 61, 144, 36, 43, 24, 229, 97, 36, 27, 57, 160,
46, 146, 188, 49, 52, 25, 42, 45, 17, 68, 47, 61, 32, 229, 23, 59, 17,
60, 81, 73, 52, 105, 27, 41, 129, 29, 35, 50, 56, 41, 53, 108, 118,
36, 161, 171, 36, 27, 56, 50, 29, 50, 20, 19, 147, 16, 34, 48, 16, 17,
16, 20, 77, 17, 17, 16, 58, 14, 41, 57, 216, 162, 63, 18, 87, 51, 47,
17, 28, 46, 38, 201, 121, 67, 130, 10, 20, 88, 36, 35, 18, 41, 16, 2,
18, 82, 18, 32, 17, 47, 22, 14, 31, 70, 20, 20, 16, 79, 18, 67, 2, 45,
380, 32, 16, 55, 63, 89, 390, 50, 173, 18, 36, 18, 18, 16, 22, 121,
62, 298, 38, 56, 28, 20, 226, 52, 94, 32, 24, 60, 22, 89, 19, 112, 80,
20, 18, 45, 210, 52, 17, 32, 143, 72, 24, 24, 18, 56, 38, 71, 60, 371,
```

```
73, 16, 47, 213, 40, 111, 16, 38, 17, 53, 303, 79, 155, 53, 129, 61,
16, 61, 64, 34, 60, 50, 21, 22, 59, 70, 46, 164, 125, 60, 67, 62, 374,
286, 62, 48, 38, 138, 19, 114, 44, 28, 62, 65, 2, 172, 88, 71, 432,
61, 371, 37, 105, 72, 68, 20, 139, 40, 60, 22, 102, 58, 56, 53, 46,
64, 18, 61, 20, 52, 16, 61, 26, 42, 73, 77, 326, 60, 34, 33, 14, 67,
115, 66, 61, 58, 60, 44, 163, 111, 209, 297, 113, 47, 62, 20, 111, 18,
30, 66, 53, 72, 65, 62, 73, 77, 61, 49, 87, 21, 101, 71, 97, 131, 55,
58, 68, 61, 42, 18, 281, 49, 41, 58, 63, 82, 58, 67, 16, 18, 18, 16,
37, 24, 68, 20, 215, 141, 29, 120, 41, 39, 79, 64, 30, 62, 27, 121,
22, 66, 123, 32, 93, 45, 52, 52, 92, 71, 44, 48, 30, 40, 18, 39, 41,
15, 100, 73, 364, 119, 30, 31, 127, 55, 31, 27, 16, 54, 202, 73, 66,
62, 29, 18, 2, 57, 74, 29, 54, 28, 53, 62, 43, 39, 101, 62, 50, 105,
135, 41, 18, 92, 96, 47, 263, 36, 154, 48, 75, 17, 60, 43, 14, 83, 62,
85, 24, 59, 39, 24, 56, 20, 20, 71, 133, 45, 57, 149, 118, 30, 110,
537, 221, 52, 235, 36, 36, 14, 14, 298, 26, 130, 56, 49, 58, 48, 96,
46, 93, 31, 37, 55, 157, 55, 274, 16, 25, 55, 18, 13, 86, 28, 50, 47,
149, 79, 128, 16, 20, 48, 14, 25, 119, 63, 17, 28, 68, 103, 48, 74,
92, 71, 44, 49, 56, 20, 34, 71, 38, 37, 50, 28, 18, 44, 65, 209, 45,
20, 16, 37, 184, 12, 14, 516, 117, 38, 58, 48, 21, 16, 60, 39, 45, 17,
36, 16, 24, 62, 18, 40, 51, 105, 74, 62, 42, 16, 46, 43, 305, 18, 37,
116, 32, 90, 148, 45, 129, 118, 17, 54, 93, 75, 52, 278, 58, 18, 87,
158, 22, 263, 18, 48, 26, 132, 41, 92, 101, 190, 65, 100, 97, 161, 92,
115, 97, 18, 372, 163, 256, 18, 368, 210, 60, 375, 167, 173, 88, 151,
130, 91, 48, 118, 16, 193, 53, 162, 144, 52, 115, 96, 377, 172, 22,
54, 293, 54, 68, 17, 127, 51, 624, 41, 87, 53, 76, 36, 61, 59, 59,
132, 48, 53, 44, 186, 203, 50, 26, 41, 57, 37, 168, 66, 223, 39, 185,
110, 24, 111, 22, 112, 95, 60, 503, 56, 76, 56, 165, 36, 181, 235, 56,
371, 76, 65, 46, 181, 111, 28, 130, 93, 102, 14, 241, 169, 48, 116,
16, 427, 110, 53, 85, 141, 32, 41, 78, 16, 78, 18, 30, 34, 23, 46, 20,
49, 37, 41, 127, 44, 56, 476, 39, 18, 69, 109, 16, 60, 62, 16, 28, 74,
78, 37, 31, 47, 146, 2, 154, 155, 22, 42, 45, 20, 330, 19, 50, 26, 25,
48, 138, 43, 18, 45, 36, 27, 78, 15, 31, 2, 72, 143, 30, 122, 20, 16,
501, 189, 22, 92, 56, 207, 62, 61, 51, 66, 20, 95, 202, 118, 302, 38,
65, 66, 19, 54, 42, 71, 44, 34, 54, 88, 316, 16, 42, 42, 49, 48, 76,
55, 46, 66, 61, 58, 14, 69, 65, 62, 35, 67, 30, 39, 62, 49, 20, 122,
87, 145, 18, 59, 114, 72, 64, 110, 206, 18, 20, 32, 41, 77, 279, 28,
51, 29, 330, 137, 436, 89, 50, 18, 29, 18, 65, 17, 143, 60, 16, 54,
15, 30, 27, 20, 22, 64, 80, 51, 44, 17, 52, 82, 203, 88, 78, 101, 155,
53, 89, 29, 49, 18, 102, 101, 103, 18, 46, 38, 70, 31, 41, 18, 40, 28,
17, 24, 90, 282, 68, 77, 17, 18, 14, 45, 25, 167, 17, 18, 88, 59, 445,
44, 80, 34, 31, 60, 16, 53, 244, 22, 22, 66, 31, 64, 28, 27, 50, 38,
112, 68, 45, 52, 38, 23, 55, 46, 75, 151, 102, 27, 46, 36, 16, 16, 16,
36, 57, 118, 70, 35, 20, 101, 63, 37, 49, 19, 79, 50, 18, 75, 64, 16,
72, 58, 60, 179, 56, 70, 70, 18, 18, 42, 75, 58, 42, 46, 62, 61, 18,
18, 18, 45, 101, 21, 63, 104, 57, 51, 30, 16, 22, 55, 36, 146, 149,
65, 16, 18, 51, 270, 153, 53, 20, 59, 16, 201, 134, 22, 53, 241, 230,
119, 65, 16, 16, 141, 96, 16, 15, 16, 40, 113, 56, 181, 18, 26, 44,
58, 54, 42, 95, 53, 67, 30, 25, 102, 102, 25, 138, 56, 201, 20, 144,
115, 45, 21, 338, 45, 22, 228, 90, 20, 191, 365, 241, 40, 46, 34, 54,
229, 340, 130, 128, 55, 161, 178, 178, 22, 52, 46, 358, 16, 14, 177,
```

65, 409, 127, 66, 180, 24, 50, 57, 55, 53, 42, 271, 111, 37, 104, 63,
18, 37, 147, 102, 18, 101, 162, 70, 214, 128, 36, 365, 23, 56, 123,
131, 59, 28, 29, 289, 46, 22, 94, 140, 348, 64, 59, 210, 218, 32, 78,
130, 59, 33, 72, 19, 20, 84, 50, 63, 35, 20, 13, 17, 16, 71, 15, 44,
19, 28, 56, 38, 27, 59, 69, 27, 28, 18, 327, 32, 49, 64, 43, 36, 31,
122, 57, 51, 39, 13, 118, 100, 33, 33, 41, 81, 177, 18, 27, 38, 26,
16, 16, 24, 24, 563, 79, 44, 35, 29, 49, 111, 63, 37, 32, 67, 29, 20,
50, 38, 16, 25, 112, 115, 16, 46, 41, 17, 98, 52, 125, 20, 30, 17, 18,
37, 142, 27, 39, 66, 15, 62, 108, 42, 201, 169, 74, 60, 63, 18, 16,
60, 71, 98, 23, 161, 58, 72, 128, 71, 94, 346, 35, 278, 312, 62, 40,
31, 54, 49, 20, 48, 52, 45, 26, 36, 337, 62, 81, 104, 44, 73, 150, 21,
129, 56, 52, 184, 108, 156, 40, 29, 26, 40, 169, 75, 84, 69, 26, 58,
209, 17, 18, 390, 295, 133, 18, 161, 55, 58, 157, 28, 15, 25, 22, 247,
56, 64, 123, 59, 126, 17, 16, 59, 28, 107, 408, 62, 38, 226, 172, 2,
17, 85, 276, 36, 20, 67, 72, 183, 182, 67, 59, 58, 34, 47, 196, 20,
29, 42, 26, 71, 17, 16, 40, 36, 36, 23, 48, 61, 73, 17, 96, 50, 50,
166, 71, 111, 75, 124, 66, 29, 45, 27, 29, 58, 36, 36, 18, 80, 45, 45,
44, 55, 20, 15, 113, 43, 38, 36, 95, 16, 14, 86, 71, 18, 47, 53, 40,
99, 48, 18, 52, 203, 62, 20, 55, 161, 338, 48, 172, 45, 22, 119, 456,
20, 112, 103, 73, 241, 172, 37, 16, 42, 452, 36, 101, 97, 20, 16, 20,
18, 106, 88, 55, 38, 120, 95, 234, 129, 54, 38, 120, 14, 18, 36, 34,
420, 148, 94, 18, 254, 128, 291, 48, 71, 67, 22, 18, 48, 38, 61, 46,
88, 361, 104, 190, 110, 155, 91, 94, 16, 69, 49, 125, 24, 31, 114, 16,
52, 89, 16, 56, 108, 54, 187, 43, 35, 76, 16, 46, 20, 16, 60, 318,
116, 82, 77, 261, 216, 19, 43, 44, 55, 23, 240, 70, 22, 58, 82, 52,
37, 306, 145, 56, 36, 39, 50, 38, 18, 31, 44, 68, 91, 54, 17, 17, 32,
17, 67, 19, 94, 21, 25, 49, 43, 35, 58, 51, 18, 47, 401, 29, 28, 21,
55, 44, 31, 54, 45, 341, 24, 65, 101, 108, 40, 51, 40, 39, 113, 55,
108, 81, 24, 17, 33, 17, 169, 30, 71, 158, 46, 24, 88, 50, 298, 52,
84, 146, 184, 65, 80, 48, 29, 85, 53, 47, 318, 57, 188, 18, 22, 20,
44, 30, 97, 31, 61, 81, 61, 58, 18, 59, 52, 76, 43, 48, 42, 150, 47,
29, 17, 22, 67, 117, 14, 38, 464, 628, 106, 28, 62, 20, 399, 20, 38,
34, 73, 71, 19, 38, 20, 60, 40, 31, 47, 45, 18, 16, 16, 18, 28, 194,
50, 27, 15, 33, 87, 35, 37, 19, 119, 340, 28, 31, 48, 74, 2, 76, 237,
156, 246, 149, 393, 41, 91, 181, 37, 18, 29, 61, 17, 37, 50, 66, 44,
33, 68, 32, 281, 28, 135, 89, 53, 58, 35, 261, 81, 66, 38, 91, 26, 47,
36, 59, 40, 42, 138, 212, 40, 28, 21, 16, 19, 60, 38, 26, 78, 27, 127,
79, 57, 39, 24, 121, 18, 87, 67, 28, 38, 59, 16, 17, 59, 14, 37, 35,
47, 147, 38, 28, 145, 163, 59, 56, 91, 40, 23, 24, 34, 27, 16, 78, 64,
120, 112, 62, 404, 175, 36, 125, 125, 37, 23, 33, 31, 52, 21, 25, 21,
17, 21, 113, 21, 15, 86, 132, 28, 18, 57, 25, 54, 18, 390, 73, 83, 52,
15, 128, 65, 47, 85, 26, 8, 325, 119, 24, 49, 23, 62, 21, 74, 72, 74,
66, 108, 21, 22, 43, 50, 411, 14, 22, 35, 221, 275, 183, 232, 52, 22,
59, 14, 104, 97, 85, 16, 156, 22, 146, 120, 112, 54, 22, 29, 56, 54,
120, 49, 83, 186, 131, 51, 96, 60, 86, 86, 20, 70, 184, 18, 43, 83,
18, 55, 37, 18, 121, 18, 21, 42, 112, 42, 443, 137, 68, 23, 16, 185,
90, 2, 22, 20, 18, 80, 18, 23, 18, 165, 131, 94, 210, 106, 22, 17, 83,
120, 53, 142, 40, 96, 29, 163, 212, 161, 20, 175, 132, 197, 52, 36,
60, 51, 43, 26, 18, 96, 79, 58, 166, 377, 26, 23, 46, 182, 90, 45, 16,

```
41, 264, 38, 90, 39, 161, 17, 19, 44, 26, 38, 52, 38, 59, 39, 420, 16,
43, 34, 39, 31, 98, 16, 224, 19, 33, 31, 169, 179, 51, 16, 113, 51,
60, 56, 52, 42, 20, 75, 48, 92, 18, 90, 245, 86, 59, 127, 17, 33, 546,
66, 52, 426, 41, 133, 19, 17, 58, 65, 139, 22, 104, 197, 36, 37, 46,
44, 248, 50, 27, 35, 186, 63, 191, 43, 37, 22, 19, 27, 25, 41, 40,
118, 24, 16, 20, 37, 16, 27, 71, 37, 87, 49, 36, 15, 44, 56, 156, 42,
45, 44, 35, 17, 58, 128, 26, 29, 32, 48, 17, 59, 16, 39, 197, 48, 206,
76, 17, 16, 51, 17, 29, 17, 18, 18, 62, 25, 123, 26, 20, 292, 29, 314,
56, 240, 67, 181, 69, 111, 325, 52, 65, 23, 45, 134, 60, 17, 18, 43,
116, 334, 16, 68, 58, 34, 126, 125, 57, 40, 24, 35, 26, 149, 34, 16,
33, 43, 59, 25, 29, 185, 392, 71, 42, 111, 61, 39, 24, 412, 60, 24,
241, 58, 63, 83, 16, 58, 14, 39, 16, 48, 14, 391, 98, 75, 85, 45, 71,
42, 68, 56, 138, 39, 29, 17, 20, 183, 20, 52, 173, 148, 20, 24, 101,
179, 57, 73, 36, 40, 51, 57, 37, 30, 117, 108, 67, 55, 36, 22, 39, 34,
79, 27, 63, 37, 35, 75, 24, 22, 48, 58, 232, 13, 18, 53, 27, 159, 456,
77, 55, 39, 49, 20, 27, 55, 41, 42, 56, 91, 31, 19, 363, 52, 148, 83,
42, 32, 11, 95, 176, 51, 111, 55, 14, 16, 17, 16, 113, 16, 85, 56, 90,
120, 107, 166, 128, 62, 34, 143, 60, 439, 64, 105, 14, 28, 83, 97, 43,
16, 60, 18, 63, 331, 53, 67, 29, 74, 63, 18, 135, 65, 66, 21, 22, 20,
198, 82, 31, 85, 20, 34, 90, 167, 151, 48, 20, 118, 2, 57, 57, 179,
91, 63, 85, 85, 38, 65, 49, 35, 18, 53, 109, 113, 91, 328, 169, 28,
163, 106, 40, 96, 26, 12, 16, 63, 34, 62, 16, 113, 232, 120, 169, 63,
131, 51, 17, 113, 136, 124, 39, 34, 74, 65, 57, 52, 34, 64, 39, 182,
18, 17, 90, 30, 136, 21, 94, 16, 40, 25, 20, 76, 16, 53, 18, 17, 228,
86, 34, 113, 45, 41, 24, 30, 32, 25, 34, 16, 201, 34, 36, 51, 40, 20,
52, 72, 316, 22, 2, 17, 131, 63, 100, 20, 29, 14, 16, 52, 85, 16, 232,
23, 217, 17, 36, 168, 24, 34, 181, 17, 26, 18, 18, 22, 72, 45, 110,
217, 14, 113, 109, 20, 36, 184, 175, 18, 20, 61, 27, 95, 41, 29, 59,
42, 58, 18, 60, 34, 109, 70, 38, 34, 22, 29, 49, 66, 18, 47, 20, 54,
59, 26, 66, 65, 17, 23, 58, 41, 16, 17, 62, 34, 22, 16, 20, 20, 18,
16, 48, 59, 34, 50, 41, 56, 127, 103, 100, 242, 56, 57, 101, 133, 16,
127, 29, 14, 56, 27, 27, 52, 39, 149, 18, 42, 53, 43, 66, 113, 80, 87,
178, 109, 67, 48, 17, 24, 206, 214, 20, 28, 14, 64, 345, 45, 73, 76,
289, 38, 273, 75, 190, 16, 18, 46, 33, 135, 59, 70, 40, 19, 18, 59,
27, 133, 69, 64, 65, 67, 55, 32, 81, 12, 57, 57, 20, 204, 28, 29, 159,
29, 74, 37, 28, 19, 59, 13, 17, 2, 40, 60, 30, 40, 52, 82, 103, 36,
16, 120, 324, 35, 27, 181, 319, 110, 16, 44, 171, 15, 40, 57, 85, 133,
57, 2, 42, 75, 16, 234, 38, 63, 52, 96, 117, 271, 292, 53, 43, 28, 40,
40, 431, 83, 357, 106, 63, 166, 362, 286, 261, 207, 106, 39, 151, 48,
266, 103, 131, 140, 401, 60, 193, 95, 68, 117, 220, 97, 194, 32, 105,
34, 113, 371, 206, 50, 24, 20, 67, 35, 166, 153, 99, 155, 174, 470,
60, 86, 153, 16, 41, 53, 64, 159, 93, 79, 117, 106, 61, 67, 139, 113,
180, 77, 117, 181, 75, 71, 2, 31, 104, 170, 54, 154, 73, 104, 149, 28,
14, 18, 68, 220, 194, 140, 73, 122, 44, 29, 261, 39, 55, 59, 295, 81,
123, 67, 78, 323, 47, 44, 58, 68, 45, 18, 21, 45, 44, 114, 16, 59,
162, 33, 29, 94, 64, 34, 82, 75, 16, 48, 59, 326, 16, 17, 25, 103,
152, 60, 123, 38, 16, 18, 79, 58, 45, 14, 14, 31, 39, 64, 56, 155, 53,
36, 35, 28, 38, 17, 44, 50, 33, 16, 57, 20, 30, 76, 43, 131, 53, 186,
32, 30, 52, 52, 57, 16, 44, 150, 124, 46, 40, 49, 17, 40, 53, 33, 104,
```

```
16, 36, 63, 84, 114, 30, 38, 32, 53, 105, 115, 42, 27, 31, 22, 34, 12,
86, 32, 89, 75, 16, 20, 53, 91, 60, 54, 60, 21, 95, 176, 45, 22, 110,
71, 31, 17, 155, 57, 215, 79, 301, 94, 35, 21, 14, 208, 45, 14, 317,
106, 31, 18, 43, 176, 65, 203, 383, 16, 105, 48, 26, 112, 52, 110, 19,
61, 16, 29, 37, 82, 56, 134, 48, 34, 47, 25, 22, 37, 44, 35, 16, 16,
75, 78, 152, 16, 54, 16, 26, 252, 106, 63, 179, 111, 45, 43, 141, 90,
81, 18, 57, 155, 112, 35, 212, 27, 12, 14, 132, 18, 48, 76, 32, 44,
22, 29, 66, 83, 83, 64, 20, 20, 28, 38, 145, 325, 25, 288, 30, 60,
348, 68, 60, 42, 48, 45, 155, 64, 51, 79, 40, 14, 236, 18, 18, 64, 56,
18, 54, 195, 336, 117, 222, 215, 154, 15, 169, 159, 82, 92, 12, 166,
112, 135, 14, 53, 71, 129, 20, 79, 68, 166, 22, 18, 227, 71, 319, 159,
54, 299, 55, 227, 132, 158, 66, 144, 62, 63, 16, 124, 29, 71, 85, 54,
112, 17, 178, 195, 158, 319, 16, 299, 158, 108, 119, 76, 137, 71, 16,
63, 189, 135, 137, 44, 66, 97, 82, 52, 62, 74, 123, 86, 53, 134, 64,
30, 60, 64, 32, 47, 22, 142, 111, 212, 140, 175, 130, 16, 112, 168,
46, 381, 131, 56, 58, 22, 131, 20, 153, 37, 42, 137, 78, 16, 41, 28,
49, 62, 18, 38, 152, 42, 37, 38, 40, 64, 512, 222, 65, 120, 313, 42,
129, 102, 98, 22, 84, 30, 96, 165, 36, 19, 117, 53, 136, 111, 46, 36,
16, 16, 16, 72, 38, 45, 43, 287, 32, 128, 139, 140, 66, 49, 16, 29,
18, 22, 207, 117, 395, 14, 115, 39, 61, 56, 30, 94, 52, 20, 43, 142,
51, 67, 53, 46, 109, 122, 110, 114, 68, 16, 17, 18, 56, 111, 127, 37,
111, 29, 91, 107, 67, 62, 74, 229, 313, 230, 54, 54, 46, 85, 17, 53,
17, 45, 212, 94, 66, 98, 16, 55, 20, 17, 68, 37, 54, 18, 28, 79, 29,
16, 18, 247, 130, 48, 172, 189, 123, 487, 220, 78, 38, 35, 67, 25, 99,
38, 63, 83, 134, 330, 89, 192, 36, 16, 40, 48, 34, 211, 16, 208, 47,
83, 41, 39, 115, 51, 140, 16, 18, 24, 63, 22, 71, 128, 46, 59, 40, 25,
32, 64, 151, 39, 26, 37, 93, 26, 18, 47, 41, 165, 41, 44, 65, 460, 49,
44, 20, 156, 47, 69, 191, 48, 55, 101, 20, 44, 24, 16, 19, 45, 302,
118, 29, 73, 43, 18, 210, 18, 401, 38, 158, 87, 64, 118, 33, 14, 18,
222, 103, 16, 17, 99, 249, 99, 44, 103, 66, 97, 75, 48, 33, 29, 29,
25, 80, 66, 58, 53, 52, 17, 64, 35, 58, 321, 33, 167, 22, 19, 17, 14,
16, 61, 46, 77, 16, 65, 55, 20, 14, 36, 181, 60, 16, 116, 320, 249,
47, 60, 74, 223, 86, 20, 67, 129, 155, 100, 16, 131, 61, 22, 160, 51,
30, 117, 492, 53, 96, 129, 31, 61, 141, 90, 212, 116, 41, 77, 107, 53,
42, 108, 31, 80, 33, 20, 81, 47, 106, 416, 63, 126, 252, 84, 180, 177,
104, 223, 61, 116, 25, 28, 302, 181, 117, 102, 2, 55, 133, 60, 52, 52,
59, 51, 40, 22, 48, 29, 66, 16, 102, 435, 24, 38, 167, 18, 44, 57,
164, 15, 26, 30, 12, 32, 23, 41, 25, 25, 18, 17, 49, 33, 104, 27, 293,
52, 12, 117, 20, 49, 23, 26, 45, 58, 104, 54, 185, 123, 31, 28, 58,
46, 46, 20, 98, 19, 20, 28, 45, 18, 48, 162, 45, 76, 24, 35, 100, 190,
63, 36, 40, 51, 26, 289, 58, 47, 45, 2, 39, 99, 22, 110, 36, 56, 18,
44, 14, 55, 43, 59, 30, 22, 27, 17, 161, 45, 45, 78, 55, 20, 54, 126,
53, 117, 41, 15, 85, 28, 73, 42, 79, 16, 66, 51, 22, 112, 135, 35,
210, 179, 58, 45, 16, 42, 37, 105, 19, 164, 18, 80, 39, 20, 52, 22,
36, 21, 28, 51, 56, 47, 16, 362, 120, 22, 27, 38, 81, 28, 218, 55, 99,
36, 91, 22, 126, 22, 428, 50, 50, 32, 17, 35, 197, 60, 42, 156, 146,
34, 20, 2, 20, 27, 45, 22, 24, 46, 16, 66, 95, 31, 18, 52, 87, 137,
317, 70, 139, 60, 45, 18, 19, 186, 16, 144, 48, 77, 112, 144, 18, 162,
176, 83, 32, 183, 65, 48, 24, 150, 108, 63, 78, 40, 18, 39, 28, 20,
```

```
20, 32, 23, 18, 2, 15, 268, 201, 81, 17, 15, 144, 99, 16, 18, 30, 42,
280, 20, 93, 102, 57, 28, 213, 37, 59, 46, 16, 17, 20, 52, 223, 71,
29, 56, 20, 99, 63, 172, 16, 258, 326, 207, 62, 128, 66, 28, 30, 267,
40, 66, 157, 98, 49, 390, 65, 20, 118, 56, 40, 28, 45, 87, 63, 38, 18,
183, 154, 57, 18, 16, 110, 86, 164, 181, 60, 273, 108, 111, 67, 59,
20, 110, 20, 16, 80, 53, 115, 20, 394, 202, 18, 186, 24, 60, 141, 67,
189, 38, 42, 118, 22, 20, 329, 88, 63, 55, 347, 187, 35, 21, 242, 207,
172, 403, 22, 24, 166, 161, 154, 110, 22, 24, 67, 90, 57, 20, 267, 57,
329, 35, 88, 285, 193, 142, 136, 39, 59, 112, 16, 45, 33, 18, 45, 43,
35, 24, 57, 77, 113, 128, 93, 16, 20, 30, 18, 39, 218, 93, 41, 103,
20, 66, 49, 140, 46, 70, 415, 87, 18, 39, 24, 16, 136, 56, 20, 48, 53,
47, 107, 14, 167, 49, 179, 54, 19, 131, 124, 61, 133, 25, 19, 99, 33,
33, 68, 148, 47, 36, 40, 88, 2, 18, 15, 53, 23, 126, 83, 218, 35, 120,
75, 67, 84, 60, 39, 47, 36, 195, 33, 60, 82, 120, 52, 52, 117, 16, 15,
69, 20, 175, 25, 14, 62, 233, 152, 42, 116, 185, 178, 16, 88, 97, 24,
137, 41, 43, 18, 17, 88, 38, 92, 18, 41, 59, 418, 67, 57, 419, 65, 55,
58, 61, 18, 35, 59, 26, 16, 16, 44, 21]
Nombres de docs 10788

855324
```

```
#recuperer des labels uniques pour les categories
from sklearn.preprocessing import MultiLabelBinarizer
mlb = MultiLabelBinarizer()

train_labels = mlb.fit_transform(train_categories)
test_labels = mlb.transform(test_categories)
whole_labels = mlb.fit_transform(whole_cats)

print(mlb.classes_)
print(whole_labels)
```

```
['acq' 'alum' 'barley' 'bop' 'carcass' 'castor-oil' 'cocoa' 'coconut'
 'coconut-oil' 'coffee' 'copper' 'copra-cake' 'corn' 'cotton' 'cotton-
oil'
 'cpi' 'cpu' 'crude' 'dfl' 'dlr' 'dmk' 'earn' 'fuel' 'gas' 'gnp'
'gold'
 'grain' 'groundnut' 'groundnut-oil' 'heat' 'hog' 'housing' 'income'
 'instal-debt' 'interest' 'ipi' 'iron-steel' 'jet' 'jobs' 'l-cattle'
 'lead' 'lei' 'lin-oil' 'livestock' 'lumber' 'meal-feed' 'money-fx'
 'money-supply' 'naphtha' 'nat-gas' 'nickel' 'nkr' 'nzdlr' 'oat'
'oilseed'
 'orange' 'palladium' 'palm-oil' 'palmkernel' 'pet-chem' 'platinum'
 'potato' 'propane' 'rand' 'rape-oil' 'rapeseed' 'reserves' 'retail'
 'rice' 'rubber' 'rye' 'ship' 'silver' 'sorghum' 'soy-meal' 'soy-oil'
 'soybean' 'strategic-metal' 'sugar' 'sun-meal' 'sun-oil' 'sunseed'
'tea'
 'tin' 'trade' 'veg-oil' 'wheat' 'wpi' 'yen' 'zinc']
[[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]]
```

4. Classification avec SVM

```python
#SVM
import numpy as np
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report

classifier_svm = OneVsRestClassifier(LinearSVC())
classifier_svm.fit(vect_train_docs,train_labels)
test_labels_predict=classifier_svm.predict(vect_test_docs)
report = classification_report(test_labels,test_labels_predict)
print(report)
scores=classifier_svm.score(vect_test_docs,test_labels)
print(scores)
```

```
              precision    recall  f1-score   support

           0       0.98      0.95      0.97       719
           1       1.00      0.43      0.61        23
           2       1.00      0.64      0.78        14
           3       0.91      0.67      0.77        30
           4       0.88      0.39      0.54        18
           5       0.00      0.00      0.00         1
           6       1.00      0.94      0.97        18
           7       1.00      0.50      0.67         2
           8       0.00      0.00      0.00         3
           9       0.96      0.96      0.96        28
          10       1.00      0.83      0.91        18
          11       0.00      0.00      0.00         1
          12       0.95      0.75      0.84        56
          13       1.00      0.55      0.71        20
          14       0.00      0.00      0.00         2
          15       0.92      0.43      0.59        28
          16       0.00      0.00      0.00         1
          17       0.92      0.84      0.88       189
          18       0.00      0.00      0.00         1
          19       0.87      0.61      0.72        44
          20       0.00      0.00      0.00         4
          21       0.99      0.98      0.98      1087
          22       1.00      0.20      0.33        10
          23       1.00      0.47      0.64        17
```

24	1.00	0.69	0.81	35
25	0.91	0.67	0.77	30
26	0.98	0.82	0.89	149
27	0.00	0.00	0.00	4
28	0.00	0.00	0.00	1
29	1.00	0.60	0.75	5
30	1.00	0.33	0.50	6
31	1.00	0.75	0.86	4
32	1.00	0.43	0.60	7
33	0.00	0.00	0.00	1
34	0.88	0.65	0.75	131
35	1.00	0.83	0.91	12
36	0.90	0.64	0.75	14
37	0.00	0.00	0.00	1
38	0.92	0.52	0.67	21
39	0.00	0.00	0.00	2
40	0.00	0.00	0.00	14
41	1.00	1.00	1.00	3
42	0.00	0.00	0.00	1
43	0.75	0.38	0.50	24
44	0.00	0.00	0.00	6
45	1.00	0.32	0.48	19
46	0.81	0.74	0.77	179
47	0.93	0.74	0.82	34
48	0.00	0.00	0.00	4
49	0.76	0.53	0.63	30
50	0.00	0.00	0.00	1
51	0.00	0.00	0.00	2
52	0.00	0.00	0.00	2
53	1.00	0.17	0.29	6
54	0.81	0.55	0.66	47
55	1.00	0.64	0.78	11
56	0.00	0.00	0.00	1
57	1.00	0.60	0.75	10
58	0.00	0.00	0.00	1
59	0.00	0.00	0.00	12
60	0.00	0.00	0.00	7
61	1.00	0.33	0.50	3
62	0.00	0.00	0.00	3
63	0.00	0.00	0.00	1
64	0.00	0.00	0.00	3
65	1.00	0.44	0.62	9
66	0.92	0.61	0.73	18
67	1.00	0.50	0.67	2
68	0.88	0.29	0.44	24
69	1.00	0.75	0.86	12
70	0.00	0.00	0.00	1
71	0.92	0.64	0.75	89
72	0.00	0.00	0.00	8

```
          73       0.75      0.30      0.43        10
          74       1.00      0.23      0.38        13
          75       0.00      0.00      0.00        11
          76       0.76      0.48      0.59        33
          77       0.00      0.00      0.00        11
          78       1.00      0.75      0.86        36
          79       0.00      0.00      0.00         1
          80       0.00      0.00      0.00         2
          81       1.00      0.20      0.33         5
          82       0.00      0.00      0.00         4
          83       1.00      0.58      0.74        12
          84       0.88      0.74      0.81       117
          85       0.94      0.43      0.59        37
          86       0.92      0.76      0.83        71
          87       1.00      0.60      0.75        10
          88       0.00      0.00      0.00        14
          89       1.00      0.46      0.63        13

   micro avg       0.95      0.79      0.86      3744
   macro avg       0.59      0.37      0.44      3744
weighted avg       0.92      0.79      0.84      3744
 samples avg       0.87      0.86      0.86      3744

0.8065584630672408

c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

```python
import pandas as pd
columns= report.split('\n')[0]
data = report.split('\n')[:-1]
df = pd.read_csv('c:\\Users\\dscon\\Documents\\COURS UM6P\\S3\\TEXT-
MINING\\score.csv')

df
```

	id	precision	recall	f1-score	support
0	0	0.99	0.95	0.97	719
1	1	1.00	0.43	0.61	23
2	2	1.00	0.64	0.78	14

```
3    3       0.95    0.60      0.73       30
4    4       0.88    0.39      0.54       18
..   ..       ...     ...       ...      ...
85  85       0.94    0.43      0.59       37
86  86       0.93    0.75      0.83       71
87  87       1.00    0.60      0.75       10
88  88       0.00    0.00      0.00       14
89  89       1.00    0.46      0.63       13

[90 rows x 5 columns]
```

5. Classification SVM avec cross validation

```python
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate
from sklearn.metrics import recall_score
scoring = ['precision_samples', 'recall_samples', 'f1_samples']
scores_svm = cross_validate(classifier_svm, vect_whole_docs,
whole_labels, cv=3, scoring=scoring)
print(scores_svm['fit_time'])
print(scores_svm['score_time'])
print(scores_svm['test_precision_samples'])
print(scores_svm['test_recall_samples'])
print(scores_svm['test_f1_samples'])
```

```
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\multiclass.py:87:
UserWarning: Label not 5 is present in all training examples.
  warnings.warn(
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))

[1.00844145 1.02443314 0.96584797]
[0.06988668 0.07264233 0.07170463]
[0.86643493 0.88022803 0.88324527]
[0.84632651 0.86370184 0.86209956]
[0.84841431 0.86375468 0.86505655]
```

```
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

6. Exercice

Q1. Réaliser le même processus avec le KNN et comparer les performances obtenues avec SVM.

Q2. Réaliser le même processus avec une méthode ensembliste et comparer les performances obtenues avec KNN et SVM.

Q3. Réaliser le même processus avec un perceptron multicouche (activation=segmoid logitic, solver=sgd)

Q4. Realiser les même processus en appliquant un features selection SelectKBest de la librairie sklearn

```python
import matplotlib.pyplot as plt

def evaluate_classifier(classifier, train_docs, train_labels,
test_docs, test_labels):
    scoring = ['precision_samples', 'recall_samples', 'f1_samples']
    scores = cross_validate(classifier, train_docs, train_labels,
cv=3, scoring=scoring, return_train_score=False)
    classifier.fit(train_docs, train_labels)
    test_predict_labels = classifier.predict(test_docs)
    report = classification_report(test_labels, test_labels_predict,
output_dict=True)
    return scores, report

def compare_classifiers(classifier1_scores, classifier2_scores,
names):
    metrics = ['fit_time', 'score_time', 'test_precision_samples',
'test_recall_samples', 'test_f1_samples']

    classifier1_values = [
        np.mean(classifier1_scores['fit_time']),
        np.mean(classifier1_scores['score_time']),
        np.mean(classifier1_scores['test_precision_samples']),
        np.mean(classifier1_scores['test_recall_samples']),
        np.mean(classifier1_scores['test_f1_samples'])
    ]

    classifier2_values = [
        np.mean(classifier2_scores['fit_time']),
```

```
        np.mean(classifier2_scores['score_time']),
        np.mean(classifier2_scores['test_precision_samples']),
        np.mean(classifier2_scores['test_recall_samples']),
        np.mean(classifier2_scores['test_f1_samples'])
    ]

    # Affichage des résultats
    x = np.arange(len(metrics))
    width = 0.35  # Largeur des barres

    fig, ax = plt.subplots(figsize=(10, 6))
    rects1 = ax.bar(x - width/2, classifier1_values, width,
label=f'{names[0]}', color='skyblue')
    rects2 = ax.bar(x + width/2, classifier2_values, width,
label=f'{names[1]}', color='salmon')

    # Configurations du graphique
    ax.set_xlabel('Métriques')
    ax.set_ylabel('Scores')
    ax.set_title(f'Comparaison des performances entre {names[0]} et
{names[1]}')
    ax.set_xticks(x)
    ax.set_xticklabels(metrics, rotation=45)
    ax.legend()

    # Afficher les valeurs sur les barres
    for rect in rects1 + rects2:
        height = rect.get_height()
        ax.annotate(f'{height:.2f}', xy=(rect.get_x() +
rect.get_width() / 2, height),
                    xytext=(0, 3), textcoords="offset points",
ha='center', va='bottom')

    plt.tight_layout()
    plt.show()


def display_classifier_results(scores, classifier_name, report):
    """Affiche les résultats du classificateur."""
    print(f"\n--- Résultats pour {classifier_name} ---")
    print(f"Fit time : {scores['fit_time']}")
    print(f"Score time : {scores['score_time']}")
    print(f"test_precision_samples :
{scores['test_precision_samples']}")
    print(f"test_recall_samples :  {scores['test_recall_samples']}")
    print(f"test_f1_samples : {scores['test_f1_samples']}")

    print("\nRapport de classification :")
    print(report)
```

```
import seaborn as sns

def plot_heatmap(report):
    metrics = ['precision', 'recall', 'f1-score']
    class_labels = list(report.keys())

    # Créer une matrice pour les métriques
    data = np.array([[report[cls][metric] for metric in metrics] for
cls in class_labels])

    # Créer une heatmap
    plt.figure(figsize=(10, 6))
    sns.heatmap(data, annot=True, fmt=".2f", cmap='YlGnBu',
xticklabels=metrics, yticklabels=class_labels)

    plt.title('Heatmap des Métriques de Classification')
    plt.xlabel('Métriques')
    plt.ylabel('Classes')
    plt.show()
```

Q1 : Processus avec KNN et comparaison avec SVM

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report


knn_classifier = KNeighborsClassifier(n_neighbors=5)
scores_knn, report_knn = evaluate_classifier(knn_classifier,
train_docs=vect_train_docs, train_labels=train_labels,
test_docs=vect_test_docs, test_labels=test_labels)
# Affichage des résultats pour KNN
display_classifier_results(scores_knn, "KNN", report=report_knn)
pd.DataFrame(report_knn)
```

```
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
```

```
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))


--- Résultats pour KNN ---
Fit time : [0.01603484 0.0191133  0.01532984]
Score time : [0.51683474 0.51292562 0.50662208]
test_precision_samples : [0.78557272 0.80507079 0.8292069 ]
test_recall_samples :  [0.77133756 0.78746461 0.81302672]
test_f1_samples : [0.7679172  0.78770204 0.81245928]

Rapport de classification :
{'0': {'precision': 0.9827338129496402, 'recall': 0.9499304589707928,
'f1-score': 0.9660537482319661, 'support': 719.0}, '1': {'precision':
1.0, 'recall': 0.43478260869565216, 'f1-score': 0.6060606060606061,
'support': 23.0}, '2': {'precision': 1.0, 'recall':
0.6428571428571429, 'f1-score': 0.782608695652174, 'support': 14.0},
'3': {'precision': 0.9090909090909091, 'recall': 0.6666666666666666,
'f1-score': 0.7692307692307693, 'support': 30.0}, '4': {'precision':
0.875, 'recall': 0.3888888888888889, 'f1-score': 0.5384615384615384,
'support': 18.0}, '5': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 1.0}, '6': {'precision': 1.0, 'recall':
0.9444444444444444, 'f1-score': 0.9714285714285714, 'support': 18.0},
'7': {'precision': 1.0, 'recall': 0.5, 'f1-score': 0.6666666666666666,
'support': 2.0}, '8': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 3.0}, '9': {'precision': 0.9642857142857143, 'recall':
0.9642857142857143, 'f1-score': 0.9642857142857143, 'support': 28.0},
'10': {'precision': 1.0, 'recall': 0.8333333333333334, 'f1-score':
0.9090909090909091, 'support': 18.0}, '11': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '12': {'precision':
0.9545454545454546, 'recall': 0.75, 'f1-score': 0.84, 'support':
56.0}, '13': {'precision': 1.0, 'recall': 0.55, 'f1-score':
0.7096774193548387, 'support': 20.0}, '14': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 2.0}, '15': {'precision':
0.9230769230769231, 'recall': 0.42857142857142855, 'f1-score':
0.5853658536585366, 'support': 28.0}, '16': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '17': {'precision':
0.9186046511627907, 'recall': 0.8359788359788359, 'f1-score':
0.8753462603878116, 'support': 189.0}, '18': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '19': {'precision':
0.8709677419354839, 'recall': 0.6136363636363636, 'f1-score': 0.72,
'support': 44.0}, '20': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 4.0}, '21': {'precision': 0.9888475836431226,
'recall': 0.9788408463661453, 'f1-score': 0.9838187702265372,
'support': 1087.0}, '22': {'precision': 1.0, 'recall': 0.2, 'f1-
score': 0.3333333333333333, 'support': 10.0}, '23': {'precision': 1.0,
'recall': 0.47058823529411764, 'f1-score': 0.64, 'support': 17.0},
'24': {'precision': 1.0, 'recall': 0.6857142857142857, 'f1-score':
```

```
0.8135593220338984, 'support': 35.0}, '25': {'precision':
0.9090909090909091, 'recall': 0.6666666666666666, 'f1-score':
0.7692307692307693, 'support': 30.0}, '26': {'precision': 0.976,
'recall': 0.8187919463087249, 'f1-score': 0.8905109489051095,
'support': 149.0}, '27': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 4.0}, '28': {'precision': 0.0, 'recall': 0.0, 'f1-
score': 0.0, 'support': 1.0}, '29': {'precision': 1.0, 'recall': 0.6,
'f1-score': 0.75, 'support': 5.0}, '30': {'precision': 1.0, 'recall':
0.3333333333333333, 'f1-score': 0.5, 'support': 6.0}, '31':
{'precision': 1.0, 'recall': 0.75, 'f1-score': 0.8571428571428571,
'support': 4.0}, '32': {'precision': 1.0, 'recall':
0.42857142857142855, 'f1-score': 0.6, 'support': 7.0}, '33':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0},
'34': {'precision': 0.8762886597938144, 'recall': 0.648854961832061,
'f1-score': 0.7456140350877193, 'support': 131.0}, '35': {'precision':
1.0, 'recall': 0.8333333333333334, 'f1-score': 0.9090909090909091,
'support': 12.0}, '36': {'precision': 0.9, 'recall':
0.6428571428571429, 'f1-score': 0.75, 'support': 14.0}, '37':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0},
'38': {'precision': 0.9166666666666666, 'recall': 0.5238095238095238,
'f1-score': 0.6666666666666666, 'support': 21.0}, '39': {'precision':
0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 2.0}, '40':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 14.0},
'41': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support':
3.0}, '42': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 1.0}, '43': {'precision': 0.75, 'recall': 0.375, 'f1-
score': 0.5, 'support': 24.0}, '44': {'precision': 0.0, 'recall': 0.0,
'f1-score': 0.0, 'support': 6.0}, '45': {'precision': 1.0, 'recall':
0.3157894736842105, 'f1-score': 0.48, 'support': 19.0}, '46':
{'precision': 0.8148148148148148, 'recall': 0.7374301675977654, 'f1-
score': 0.7741935483870968, 'support': 179.0}, '47': {'precision':
0.9259259259259259, 'recall': 0.7352941176470589, 'f1-score':
0.819672131147541, 'support': 34.0}, '48': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 4.0}, '49': {'precision':
0.7619047619047619, 'recall': 0.5333333333333333, 'f1-score':
0.6274509803921569, 'support': 30.0}, '50': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '51': {'precision':
0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 2.0}, '52':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 2.0},
'53': {'precision': 1.0, 'recall': 0.16666666666666666, 'f1-score':
0.2857142857142857, 'support': 6.0}, '54': {'precision': 0.8125,
'recall': 0.5531914893617021, 'f1-score': 0.6582278481012658,
'support': 47.0}, '55': {'precision': 1.0, 'recall':
0.6363636363636364, 'f1-score': 0.7777777777777778, 'support': 11.0},
'56': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support':
1.0}, '57': {'precision': 1.0, 'recall': 0.6, 'f1-score': 0.75,
'support': 10.0}, '58': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 1.0}, '59': {'precision': 0.0, 'recall': 0.0, 'f1-
score': 0.0, 'support': 12.0}, '60': {'precision': 0.0, 'recall': 0.0,
```

```
'f1-score': 0.0, 'support': 7.0}, '61': {'precision': 1.0, 'recall':
0.3333333333333333, 'f1-score': 0.5, 'support': 3.0}, '62':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 3.0},
'63': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support':
1.0}, '64': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 3.0}, '65': {'precision': 1.0, 'recall':
0.4444444444444444, 'f1-score': 0.6153846153846154, 'support': 9.0},
'66': {'precision': 0.9166666666666666, 'recall': 0.6111111111111112,
'f1-score': 0.7333333333333333, 'support': 18.0}, '67': {'precision':
1.0, 'recall': 0.5, 'f1-score': 0.6666666666666666, 'support': 2.0},
'68': {'precision': 0.875, 'recall': 0.2916666666666667, 'f1-score':
0.4375, 'support': 24.0}, '69': {'precision': 1.0, 'recall': 0.75,
'f1-score': 0.8571428571428571, 'support': 12.0}, '70': {'precision':
0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '71':
{'precision': 0.9193548387096774, 'recall': 0.6404494382022472, 'f1-
score': 0.7549668874172185, 'support': 89.0}, '72': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 8.0}, '73': {'precision':
0.75, 'recall': 0.3, 'f1-score': 0.42857142857142855, 'support':
10.0}, '74': {'precision': 1.0, 'recall': 0.23076923076923078, 'f1-
score': 0.375, 'support': 13.0}, '75': {'precision': 0.0, 'recall':
0.0, 'f1-score': 0.0, 'support': 11.0}, '76': {'precision':
0.7619047619047619, 'recall': 0.48484848484848486, 'f1-score':
0.5925925925925926, 'support': 33.0}, '77': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 11.0}, '78': {'precision':
1.0, 'recall': 0.75, 'f1-score': 0.8571428571428571, 'support': 36.0},
'79': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support':
1.0}, '80': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 2.0}, '81': {'precision': 1.0, 'recall': 0.2, 'f1-score':
0.3333333333333333, 'support': 5.0}, '82': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 4.0}, '83': {'precision':
1.0, 'recall': 0.5833333333333334, 'f1-score': 0.7368421052631579,
'support': 12.0}, '84': {'precision': 0.8787878787878788, 'recall':
0.7435897435897436, 'f1-score': 0.8055555555555556, 'support': 117.0},
'85': {'precision': 0.9411764705882353, 'recall': 0.43243243243243246,
'f1-score': 0.5925925925925926, 'support': 37.0}, '86': {'precision':
0.9152542372881356, 'recall': 0.7605633802816901, 'f1-score':
0.8307692307692308, 'support': 71.0}, '87': {'precision': 1.0,
'recall': 0.6, 'f1-score': 0.75, 'support': 10.0}, '88': {'precision':
0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 14.0}, '89':
{'precision': 1.0, 'recall': 0.46153846153846156, 'f1-score':
0.631578947368421, 'support': 13.0}, 'micro avg': {'precision':
0.9528728211749515, 'recall': 0.7884615384615384, 'f1-score':
0.8629055831628178, 'support': 3744.0}, 'macro avg': {'precision':
0.5887609931425809, 'recall': 0.3650654059513509, 'f1-score':
0.436502821543132, 'support': 3744.0}, 'weighted avg': {'precision':
0.9156073554970184, 'recall': 0.7884615384615384, 'f1-score':
0.8368355156799253, 'support': 3744.0}, 'samples avg': {'precision':
0.8740973832394833, 'recall': 0.8560203368086773, 'f1-score':
0.8575567002281145, 'support': 3744.0}}
```

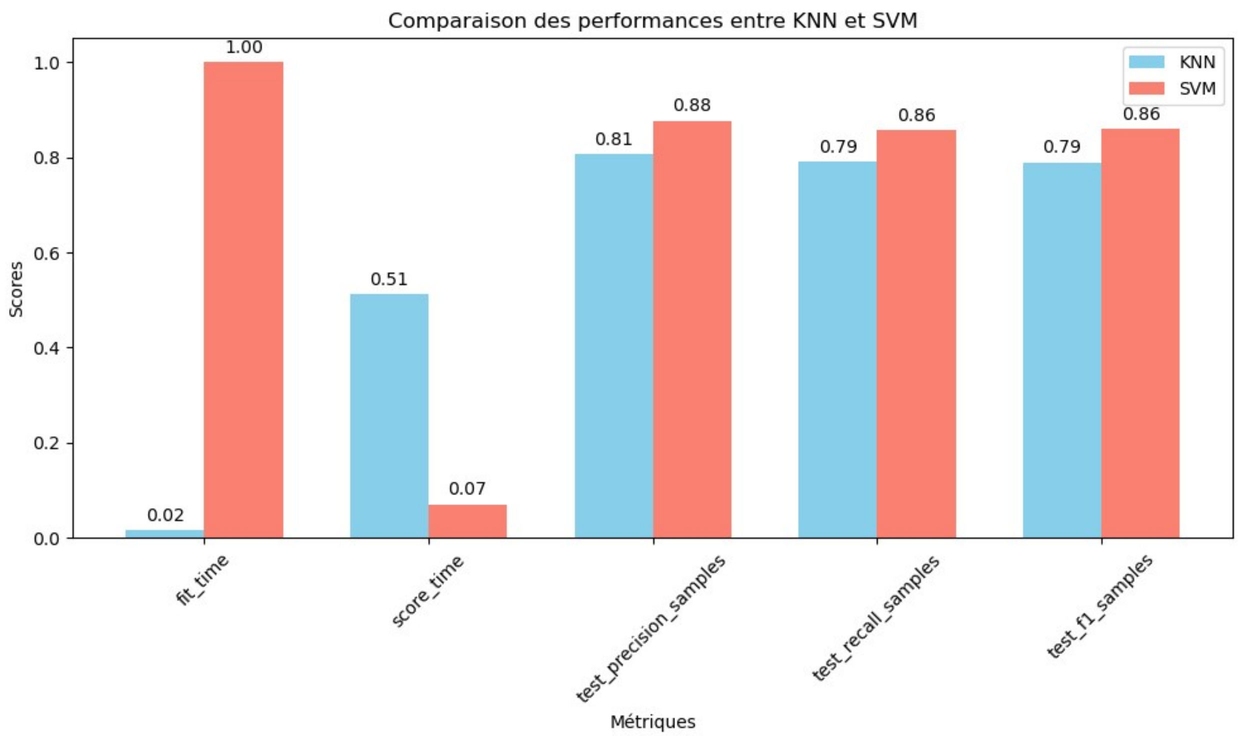
```
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

```
                    0          1          2          3          4    5
\
precision    0.982734   1.000000   1.000000   0.909091   0.875000  0.0

recall       0.949930   0.434783   0.642857   0.666667   0.388889  0.0

f1-score     0.966054   0.606061   0.782609   0.769231   0.538462  0.0

support    719.000000  23.000000  14.000000  30.000000  18.000000  1.0


                   6         7    8          9  ...          84
85  \
precision   1.000000  1.000000  0.0   0.964286  ...    0.878788
0.941176
recall      0.944444  0.500000  0.0   0.964286  ...    0.743590
0.432432
f1-score    0.971429  0.666667  0.0   0.964286  ...    0.805556
0.592593
support    18.000000  2.000000  3.0  28.000000  ...  117.000000
37.000000

                  86     87    88          89    micro avg    macro avg
\
precision   0.915254   1.00   0.0   1.000000     0.952873     0.588761

recall      0.760563   0.60   0.0   0.461538     0.788462     0.365065

f1-score    0.830769   0.75   0.0   0.631579     0.862906     0.436503

support    71.000000  10.00  14.0  13.000000  3744.000000  3744.000000


           weighted avg  samples avg
precision      0.915607     0.874097
recall         0.788462     0.856020
f1-score       0.836836     0.857557
support     3744.000000  3744.000000
```

```
[4 rows x 94 columns]
```

Comparaison avec SMV

```
compare_classifiers(classifier1_scores=scores_knn,
classifier2_scores=scores_svm, names=['KNN', 'SVM'])
```



Q2 : Random forest

```
from sklearn.ensemble import RandomForestClassifier

rf_classifier = RandomForestClassifier(n_estimators=100)
scores_rf, report_rf = evaluate_classifier(rf_classifier,
train_docs=vect_train_docs, train_labels=train_labels,
test_docs=vect_test_docs, test_labels=test_labels)
# Affichage des résultats pour Random forest
display_classifier_results(scores_rf, "RF", report=report_rf)
pd.DataFrame(report_rf)

c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
```

```
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))


--- Résultats pour RF ---
Fit time : [21.01025486 22.21801066 21.797086  ]
Score time : [0.41162515 0.36629033 0.35211968]
test_precision_samples : [0.63561776 0.63949807 0.66747779]
test_recall_samples :  [0.61295735 0.61791077 0.65150315]
test_f1_samples : [0.61807014 0.62348716 0.65483854]

Rapport de classification :
{'0': {'precision': 0.9827338129496402, 'recall': 0.9499304589707928,
'f1-score': 0.9660537482319661, 'support': 719.0}, '1': {'precision':
1.0, 'recall': 0.43478260869565216, 'f1-score': 0.6060606060606061,
'support': 23.0}, '2': {'precision': 1.0, 'recall':
0.6428571428571429, 'f1-score': 0.782608695652174, 'support': 14.0},
'3': {'precision': 0.9090909090909091, 'recall': 0.6666666666666666,
'f1-score': 0.7692307692307693, 'support': 30.0}, '4': {'precision':
0.875, 'recall': 0.3888888888888889, 'f1-score': 0.5384615384615384,
'support': 18.0}, '5': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 1.0}, '6': {'precision': 1.0, 'recall':
0.9444444444444444, 'f1-score': 0.9714285714285714, 'support': 18.0},
'7': {'precision': 1.0, 'recall': 0.5, 'f1-score': 0.6666666666666666,
'support': 2.0}, '8': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 3.0}, '9': {'precision': 0.9642857142857143, 'recall':
0.9642857142857143, 'f1-score': 0.9642857142857143, 'support': 28.0},
'10': {'precision': 1.0, 'recall': 0.8333333333333334, 'f1-score':
0.9090909090909091, 'support': 18.0}, '11': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '12': {'precision':
0.9545454545454546, 'recall': 0.75, 'f1-score': 0.84, 'support':
56.0}, '13': {'precision': 1.0, 'recall': 0.55, 'f1-score':
0.7096774193548387, 'support': 20.0}, '14': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 2.0}, '15': {'precision':
0.9230769230769231, 'recall': 0.42857142857142855, 'f1-score':
0.5853658536585366, 'support': 28.0}, '16': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '17': {'precision':
0.9186046511627907, 'recall': 0.8359788359788359, 'f1-score':
0.8753462603878116, 'support': 189.0}, '18': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '19': {'precision':
0.8709677419354839, 'recall': 0.6136363636363636, 'f1-score': 0.72,
```

'support': 44.0}, '20': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 4.0}, '21': {'precision': 0.9888475836431226,
'recall': 0.9788408463661453, 'f1-score': 0.9838187702265372,
'support': 1087.0}, '22': {'precision': 1.0, 'recall': 0.2, 'f1-
score': 0.3333333333333333, 'support': 10.0}, '23': {'precision': 1.0,
'recall': 0.47058823529411764, 'f1-score': 0.64, 'support': 17.0},
'24': {'precision': 1.0, 'recall': 0.6857142857142857, 'f1-score':
0.8135593220338984, 'support': 35.0}, '25': {'precision':
0.9090909090909091, 'recall': 0.6666666666666666, 'f1-score':
0.7692307692307693, 'support': 30.0}, '26': {'precision': 0.976,
'recall': 0.8187919463087249, 'f1-score': 0.8905109489051095,
'support': 149.0}, '27': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 4.0}, '28': {'precision': 0.0, 'recall': 0.0, 'f1-
score': 0.0, 'support': 1.0}, '29': {'precision': 1.0, 'recall': 0.6,
'f1-score': 0.75, 'support': 5.0}, '30': {'precision': 1.0, 'recall':
0.3333333333333333, 'f1-score': 0.5, 'support': 6.0}, '31':
{'precision': 1.0, 'recall': 0.75, 'f1-score': 0.8571428571428571,
'support': 4.0}, '32': {'precision': 1.0, 'recall':
0.42857142857142855, 'f1-score': 0.6, 'support': 7.0}, '33':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0},
'34': {'precision': 0.8762886597938144, 'recall': 0.648854961832061,
'f1-score': 0.7456140350877193, 'support': 131.0}, '35': {'precision':
1.0, 'recall': 0.8333333333333334, 'f1-score': 0.9090909090909091,
'support': 12.0}, '36': {'precision': 0.9, 'recall':
0.6428571428571429, 'f1-score': 0.75, 'support': 14.0}, '37':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0},
'38': {'precision': 0.9166666666666666, 'recall': 0.5238095238095238,
'f1-score': 0.6666666666666666, 'support': 21.0}, '39': {'precision':
0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 2.0}, '40':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 14.0},
'41': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support':
3.0}, '42': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 1.0}, '43': {'precision': 0.75, 'recall': 0.375, 'f1-
score': 0.5, 'support': 24.0}, '44': {'precision': 0.0, 'recall': 0.0,
'f1-score': 0.0, 'support': 6.0}, '45': {'precision': 1.0, 'recall':
0.3157894736842105, 'f1-score': 0.48, 'support': 19.0}, '46':
{'precision': 0.8148148148148148, 'recall': 0.7374301675977654, 'f1-
score': 0.7741935483870968, 'support': 179.0}, '47': {'precision':
0.9259259259259259, 'recall': 0.7352941176470589, 'f1-score':
0.819672131147541, 'support': 34.0}, '48': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 4.0}, '49': {'precision':
0.7619047619047619, 'recall': 0.5333333333333333, 'f1-score':
0.6274509803921569, 'support': 30.0}, '50': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '51': {'precision':
0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 2.0}, '52':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 2.0},
'53': {'precision': 1.0, 'recall': 0.16666666666666666, 'f1-score':
0.2857142857142857, 'support': 6.0}, '54': {'precision': 0.8125,
'recall': 0.5531914893617021, 'f1-score': 0.6582278481012658,

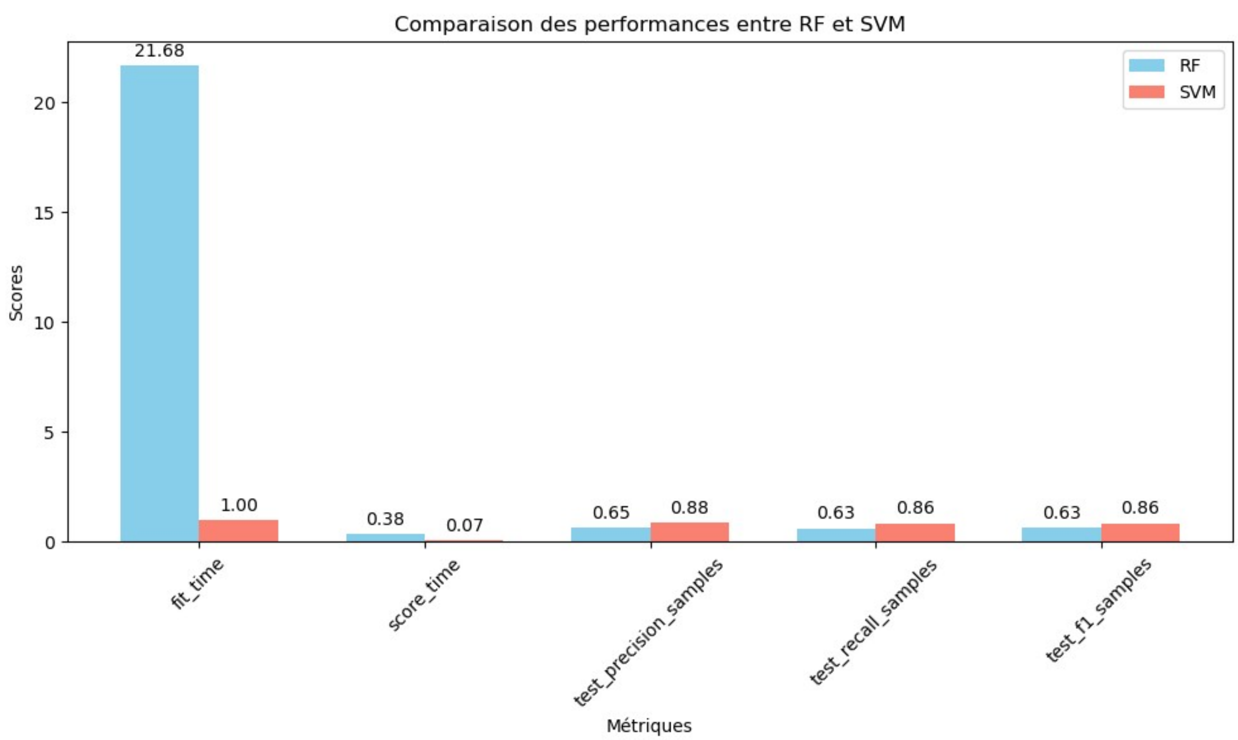
```
'support': 47.0}, '55': {'precision': 1.0, 'recall':
0.6363636363636364, 'f1-score': 0.7777777777777778, 'support': 11.0},
'56': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support':
1.0}, '57': {'precision': 1.0, 'recall': 0.6, 'f1-score': 0.75,
'support': 10.0}, '58': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 1.0}, '59': {'precision': 0.0, 'recall': 0.0, 'f1-
score': 0.0, 'support': 12.0}, '60': {'precision': 0.0, 'recall': 0.0,
'f1-score': 0.0, 'support': 7.0}, '61': {'precision': 1.0, 'recall':
0.3333333333333333, 'f1-score': 0.5, 'support': 3.0}, '62':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 3.0},
'63': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support':
1.0}, '64': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 3.0}, '65': {'precision': 1.0, 'recall':
0.4444444444444444, 'f1-score': 0.6153846153846154, 'support': 9.0},
'66': {'precision': 0.9166666666666666, 'recall': 0.6111111111111112,
'f1-score': 0.7333333333333333, 'support': 18.0}, '67': {'precision':
1.0, 'recall': 0.5, 'f1-score': 0.6666666666666666, 'support': 2.0},
'68': {'precision': 0.875, 'recall': 0.2916666666666667, 'f1-score':
0.4375, 'support': 24.0}, '69': {'precision': 1.0, 'recall': 0.75,
'f1-score': 0.8571428571428571, 'support': 12.0}, '70': {'precision':
0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '71':
{'precision': 0.9193548387096774, 'recall': 0.6404494382022472, 'f1-
score': 0.7549668874172185, 'support': 89.0}, '72': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 8.0}, '73': {'precision':
0.75, 'recall': 0.3, 'f1-score': 0.42857142857142855, 'support':
10.0}, '74': {'precision': 1.0, 'recall': 0.23076923076923078, 'f1-
score': 0.375, 'support': 13.0}, '75': {'precision': 0.0, 'recall':
0.0, 'f1-score': 0.0, 'support': 11.0}, '76': {'precision':
0.7619047619047619, 'recall': 0.48484848484848486, 'f1-score':
0.5925925925925926, 'support': 33.0}, '77': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 11.0}, '78': {'precision':
1.0, 'recall': 0.75, 'f1-score': 0.8571428571428571, 'support': 36.0},
'79': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support':
1.0}, '80': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 2.0}, '81': {'precision': 1.0, 'recall': 0.2, 'f1-score':
0.3333333333333333, 'support': 5.0}, '82': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 4.0}, '83': {'precision':
1.0, 'recall': 0.5833333333333334, 'f1-score': 0.7368421052631579,
'support': 12.0}, '84': {'precision': 0.8787878787878788, 'recall':
0.7435897435897436, 'f1-score': 0.8055555555555556, 'support': 117.0},
'85': {'precision': 0.9411764705882353, 'recall': 0.43243243243243246,
'f1-score': 0.5925925925925926, 'support': 37.0}, '86': {'precision':
0.9152542372881356, 'recall': 0.7605633802816901, 'f1-score':
0.8307692307692308, 'support': 71.0}, '87': {'precision': 1.0,
'recall': 0.6, 'f1-score': 0.75, 'support': 10.0}, '88': {'precision':
0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 14.0}, '89':
{'precision': 1.0, 'recall': 0.46153846153846156, 'f1-score':
0.631578947368421, 'support': 13.0}, 'micro avg': {'precision':
0.9528728211749515, 'recall': 0.7884615384615384, 'f1-score':
```

```
0.8629055831628178, 'support': 3744.0}, 'macro avg': {'precision':
0.5887609931425809, 'recall': 0.3650654059513509, 'f1-score':
0.436502821543132, 'support': 3744.0}, 'weighted avg': {'precision':
0.9156073554970184, 'recall': 0.7884615384615384, 'f1-score':
0.8368355156799253, 'support': 3744.0}, 'samples avg': {'precision':
0.8740973832394833, 'recall': 0.8560203368086773, 'f1-score':
0.8575567002281145, 'support': 3744.0}}

c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))

                    0          1          2          3          4    5
\
precision    0.982734   1.000000   1.000000   0.909091   0.875000  0.0

recall       0.949930   0.434783   0.642857   0.666667   0.388889  0.0

f1-score     0.966054   0.606061   0.782609   0.769231   0.538462  0.0

support    719.000000  23.000000  14.000000  30.000000  18.000000  1.0


                   6         7    8          9  ...          84
85  \
precision   1.000000  1.000000  0.0   0.964286  ...    0.878788
0.941176
recall      0.944444  0.500000  0.0   0.964286  ...    0.743590
0.432432
f1-score    0.971429  0.666667  0.0   0.964286  ...    0.805556
0.592593
support    18.000000  2.000000  3.0  28.000000  ...  117.000000
37.000000

                  86     87    88          89    micro avg    macro avg
\
precision   0.915254   1.00   0.0   1.000000     0.952873     0.588761

recall      0.760563   0.60   0.0   0.461538     0.788462     0.365065

f1-score    0.830769   0.75   0.0   0.631579     0.862906     0.436503
```

```
support    71.000000  10.00  14.0  13.000000  3744.000000  3744.000000


            weighted avg  samples avg
precision       0.915607     0.874097
recall          0.788462     0.856020
f1-score        0.836836     0.857557
support      3744.000000  3744.000000

[4 rows x 94 columns]
```

Comparaison avec SVM

```
compare_classifiers(classifier1_scores=scores_rf,
classifier2_scores=scores_svm, names=['RF', 'SVM'])
```



Q3 : Perceptron multicouche

```
from sklearn.neural_network import MLPClassifier
from sklearn.decomposition import TruncatedSVD
from sklearn.metrics import classification_report, accuracy_score
import numpy as np

svd = TruncatedSVD(n_components=500, random_state=42)
reduced_train_docs = svd.fit_transform(vect_train_docs)
```

```python
reduced_test_docs = svd.transform(vect_test_docs)

# Step 2: Convert train_labels and test_labels to 1D
if train_labels.ndim == 2:
    train_labels = np.argmax(train_labels, axis=1)
if test_labels.ndim == 2:
    test_labels = np.argmax(test_labels, axis=1)

mlp_classifier = MLPClassifier(
    hidden_layer_sizes=(50,),
    activation='logistic',
    solver='adam',
    random_state=42,
    max_iter=50,
    early_stopping=True
)
mlp_classifier.fit(reduced_train_docs, train_labels)

predicted_labels = mlp_classifier.predict(reduced_test_docs)

accuracy = accuracy_score(test_labels, predicted_labels)
print(f"Accuracy: {accuracy}")
print(classification_report(test_labels, predicted_labels))
```

```
Accuracy: 0.5836369658827426
              precision    recall  f1-score   support

           0       0.40      1.00      0.57       719
           1       0.00      0.00      0.00        22
           2       0.00      0.00      0.00        14
           3       0.00      0.00      0.00        30
           4       0.00      0.00      0.00        17
           5       0.00      0.00      0.00         1
           6       0.00      0.00      0.00        17
           7       0.00      0.00      0.00         2
           8       0.00      0.00      0.00         2
           9       0.00      0.00      0.00        25
          10       0.00      0.00      0.00        15
          12       0.00      0.00      0.00        48
          13       0.00      0.00      0.00        14
          15       0.00      0.00      0.00        24
          16       0.00      0.00      0.00         1
          17       0.00      0.00      0.00       182
          18       0.00      0.00      0.00         1
          19       0.00      0.00      0.00        43
          20       0.00      0.00      0.00         1
          21       0.84      0.96      0.90      1083
          22       0.00      0.00      0.00         9
          23       0.00      0.00      0.00         9
          24       0.00      0.00      0.00        19
```

25	0.00	0.00	0.00	26
26	0.00	0.00	0.00	77
27	0.00	0.00	0.00	3
29	0.00	0.00	0.00	4
30	0.00	0.00	0.00	4
31	0.00	0.00	0.00	3
32	0.00	0.00	0.00	5
33	0.00	0.00	0.00	1
34	0.00	0.00	0.00	124
35	0.00	0.00	0.00	11
36	0.00	0.00	0.00	14
37	0.00	0.00	0.00	1
38	0.00	0.00	0.00	13
39	0.00	0.00	0.00	2
40	0.00	0.00	0.00	12
41	0.00	0.00	0.00	3
43	0.00	0.00	0.00	6
44	0.00	0.00	0.00	5
45	0.00	0.00	0.00	6
46	0.00	0.00	0.00	96
47	0.00	0.00	0.00	29
48	0.00	0.00	0.00	1
49	0.00	0.00	0.00	13
50	0.00	0.00	0.00	1
54	0.00	0.00	0.00	13
55	0.00	0.00	0.00	9
56	0.00	0.00	0.00	1
57	0.00	0.00	0.00	4
59	0.00	0.00	0.00	6
60	0.00	0.00	0.00	3
61	0.00	0.00	0.00	3
62	0.00	0.00	0.00	2
64	0.00	0.00	0.00	1
66	0.00	0.00	0.00	14
67	0.00	0.00	0.00	1
68	0.00	0.00	0.00	1
69	0.00	0.00	0.00	9
71	0.00	0.00	0.00	39
75	0.00	0.00	0.00	2
76	0.00	0.00	0.00	2
77	0.00	0.00	0.00	6
78	0.00	0.00	0.00	25
82	0.00	0.00	0.00	3
83	0.00	0.00	0.00	10
84	0.00	0.00	0.00	76
85	0.00	0.00	0.00	11
87	0.00	0.00	0.00	9
88	0.00	0.00	0.00	6
89	0.00	0.00	0.00	5

```
    accuracy                           0.58      3019
   macro avg       0.02      0.03      0.02      3019
weighted avg       0.40      0.58      0.46      3019


c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))

from sklearn.model_selection import cross_validate
from sklearn.metrics import classification_report

def evaluate_classifier2(classifier, train_docs, train_labels,
test_docs, test_labels):
    # Ensure labels are in single-label format
    if train_labels.ndim == 2:
        train_labels = np.argmax(train_labels, axis=1)
    if test_labels.ndim == 2:
        test_labels = np.argmax(test_labels, axis=1)

    # Perform cross-validation with macro metrics for multi-class
classification
    scoring = ['precision_macro', 'recall_macro', 'f1_macro']
    scores = cross_validate(
        classifier, train_docs, train_labels, cv=3, scoring=scoring,
return_train_score=False
    )

    # Fit the classifier on the entire training data
    classifier.fit(train_docs, train_labels)

    # Predict on the test data
    test_predict_labels = classifier.predict(test_docs)
```

```
    # Generate classification report
    report = classification_report(test_labels, test_predict_labels,
output_dict=True)

    return scores, report

scores_mlp, report_mlp = evaluate_classifier2(mlp_classifier,
train_docs=vect_train_docs, train_labels=train_labels,
test_docs=vect_test_docs, test_labels=test_labels)
```

```
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\model_selection\
_split.py:776: UserWarning: The least populated class in y has only 1
members, which is less than n_splits=3.
  warnings.warn(
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\neural_network\
_multilayer_perceptron.py:690: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (50) reached and the optimization hasn't
converged yet.
  warnings.warn(
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\neural_network\
_multilayer_perceptron.py:690: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (50) reached and the optimization hasn't
converged yet.
  warnings.warn(
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\neural_network\
_multilayer_perceptron.py:690: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (50) reached and the optimization hasn't
converged yet.
  warnings.warn(
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
```

```
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

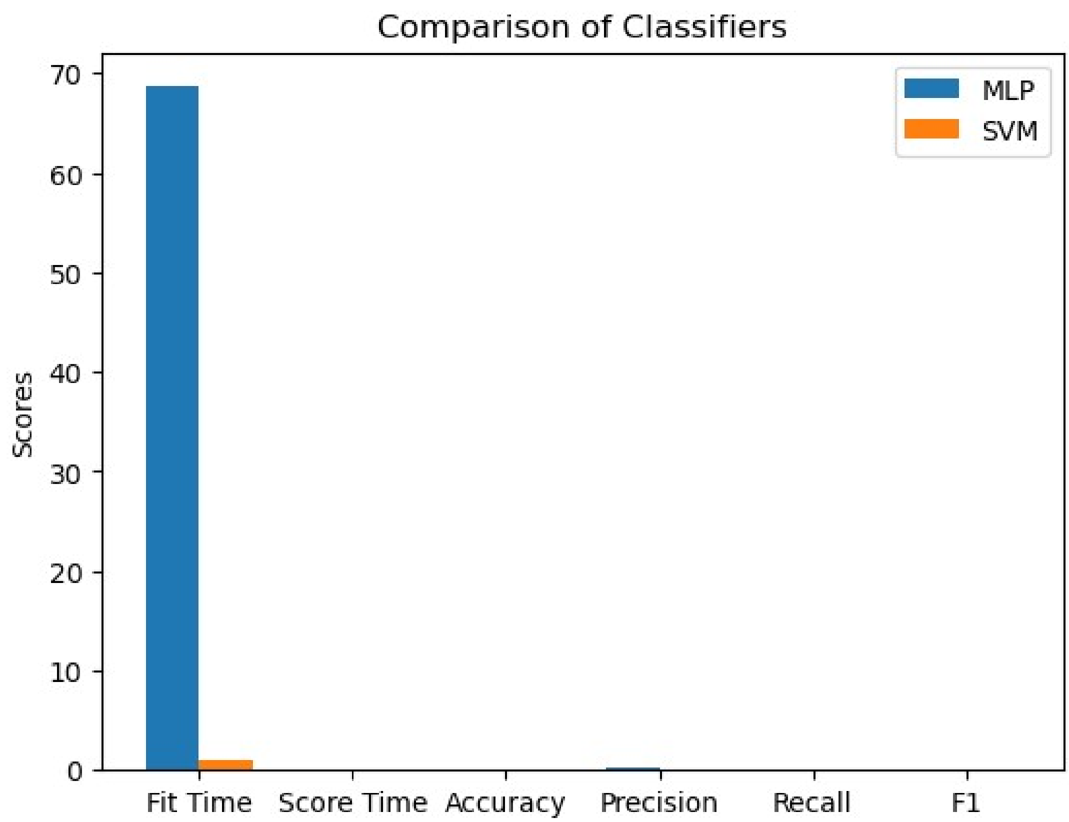
```
scores_mlp
```

```
{'fit_time': array([ 65.64952683,  17.96768069, 122.32284617]),
 'score_time': array([0.01563287, 0.01339674, 0.0194273 ]),
 'test_precision_macro': array([0.25580788, 0.01843737, 0.27007634]),
 'test_recall_macro': array([0.19642831, 0.02764762, 0.21067415]),
 'test_f1_macro': array([0.20441517, 0.02051383, 0.21370082])}
```

```
def compare_classifiers2(classifier1_scores, classifier2_scores,
names):
    metrics = ['fit_time', 'score_time', 'test_accuracy',
'test_precision_macro', 'test_recall_macro', 'test_f1_macro']

    classifier1_values = [
        np.mean(classifier1_scores.get('fit_time', 0)),
        np.mean(classifier1_scores.get('score_time', 0)),
        np.mean(classifier1_scores.get('test_accuracy', 0)),
        np.mean(classifier1_scores.get('test_precision_macro', 0)),
        np.mean(classifier1_scores.get('test_recall_macro', 0)),
        np.mean(classifier1_scores.get('test_f1_macro', 0))
    ]

    classifier2_values = [
        np.mean(classifier2_scores.get('fit_time', 0)),
        np.mean(classifier2_scores.get('score_time', 0)),
        np.mean(classifier2_scores.get('test_accuracy', 0)),
        np.mean(classifier2_scores.get('test_precision_macro', 0)),
        np.mean(classifier2_scores.get('test_recall_macro', 0)),
        np.mean(classifier2_scores.get('test_f1_macro', 0))
    ]

    # Display results
    print(f"{names[0]}: {classifier1_values}")
    print(f"{names[1]}: {classifier2_values}")
```

```
    # Optional visualization
    import matplotlib.pyplot as plt
    labels = ['Fit Time', 'Score Time', 'Accuracy', 'Precision',
'Recall', 'F1']
    x = np.arange(len(labels))  # Label locations
    width = 0.35  # Bar width

    fig, ax = plt.subplots()
    rects1 = ax.bar(x - width / 2, classifier1_values, width,
label=names[0])
    rects2 = ax.bar(x + width / 2, classifier2_values, width,
label=names[1])

    ax.set_ylabel('Scores')
    ax.set_title('Comparison of Classifiers')
    ax.set_xticks(x)
    ax.set_xticklabels(labels)
    ax.legend()

    plt.show()


compare_classifiers2(classifier1_scores=scores_mlp,
classifier2_scores=scores_svm, names=['MLP', 'SVM'])
```

```
MLP: [68.64668456713359, 0.016152302424112957, 0.0,
0.18144052831421323, 0.14491669546749816, 0.14620993974324525]
SVM: [0.9995741844177246, 0.0714112122853597, 0.0, 0.0, 0.0, 0.0]
```



Q4 : Sélection de caractéristiques avec SelectKBest

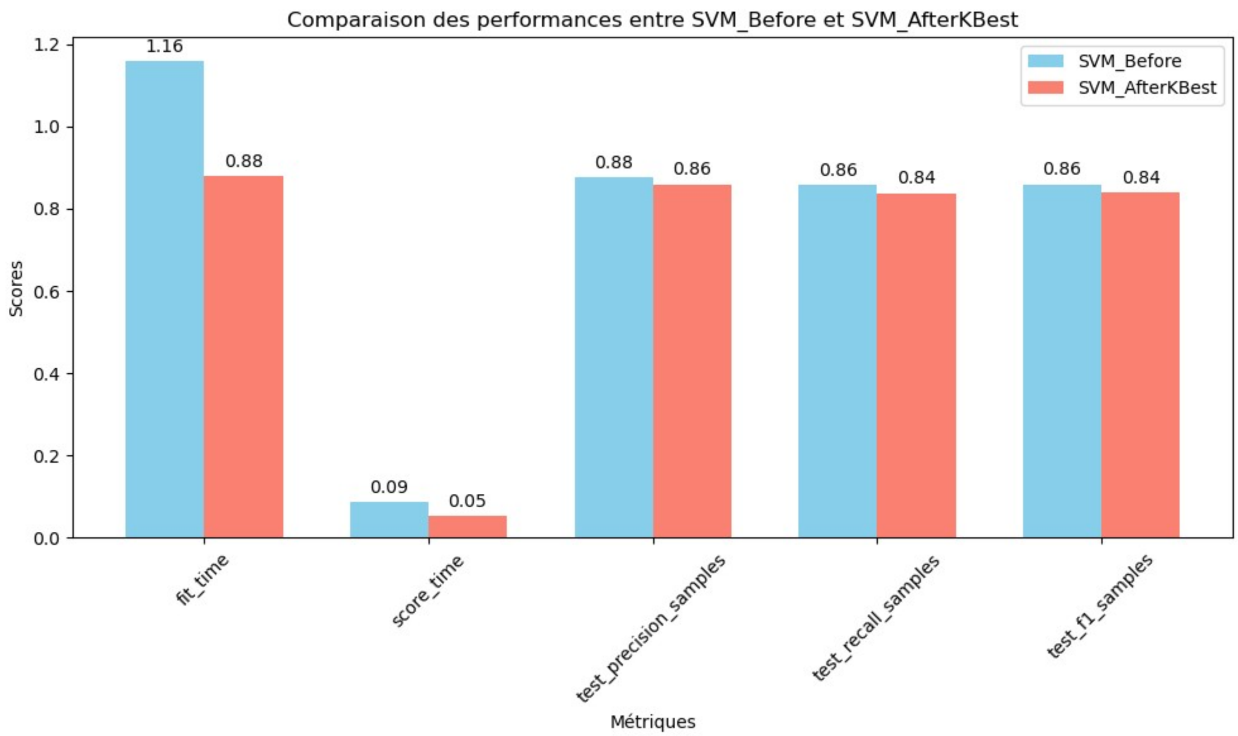
```
from sklearn.feature_selection import SelectKBest, chi2

# Sélectionner les 1000 meilleures caractéristiques
selector = SelectKBest(score_func=chi2, k=1000)
vect_train_selected = selector.fit_transform(vect_train_docs,
train_labels)
vect_test_selected = selector.transform(vect_test_docs)

vect_train_docs.shape
```

```
(7769, 34509)
```

SVM

```
import numpy as np
from sklearn.multiclass import OneVsRestClassifier
```

```
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report

classifier_svm = OneVsRestClassifier(LinearSVC())

scores_SVM, report_SVM = evaluate_classifier(classifier_svm,
train_docs=vect_train_docs, train_labels=train_labels,
test_docs=vect_test_docs, test_labels=test_labels)
# Affichage des résultats pour SVM
display_classifier_results(scores_SVM, "SVM", report=report_SVM)
pd.DataFrame(report_SVM)


compare_classifiers(classifier1_scores=scores_svm,
classifier2_scores=scores_SVM, names=['SVM_Before', 'SVM_AfterKBest'])
```



KNN

```
knn_classifier = KNeighborsClassifier(n_neighbors=5)
scores_knn_skb, report_knn_skb = evaluate_classifier(knn_classifier,
train_docs=vect_train_docs, train_labels=train_labels,
test_docs=vect_test_docs, test_labels=test_labels)
# Affichage des résultats pour KNN
display_classifier_results(scores_knn_skb, "KNN",
report=report_knn_skb)
pd.DataFrame(report_knn_skb)
```

```
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))


--- Résultats pour KNN ---
Fit time : [0.01972294 0.02276611 0.03185987]
Score time : [0.79732585 0.71018267 0.57976723]
test_precision_samples : [0.78557272 0.80507079 0.8292069 ]
test_recall_samples :  [0.77133756 0.78746461 0.81302672]
test_f1_samples : [0.7679172  0.78770204 0.81245928]

Rapport de classification :
{'0': {'precision': 0.9827338129496402, 'recall': 0.9499304589707928,
'f1-score': 0.9660537482319661, 'support': 719.0}, '1': {'precision':
1.0, 'recall': 0.43478260869565216, 'f1-score': 0.6060606060606061,
'support': 23.0}, '2': {'precision': 1.0, 'recall':
0.6428571428571429, 'f1-score': 0.782608695652174, 'support': 14.0},
'3': {'precision': 0.9090909090909091, 'recall': 0.6666666666666666,
'f1-score': 0.7692307692307693, 'support': 30.0}, '4': {'precision':
0.875, 'recall': 0.3888888888888889, 'f1-score': 0.5384615384615384,
'support': 18.0}, '5': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 1.0}, '6': {'precision': 1.0, 'recall':
0.9444444444444444, 'f1-score': 0.9714285714285714, 'support': 18.0},
'7': {'precision': 1.0, 'recall': 0.5, 'f1-score': 0.6666666666666666,
'support': 2.0}, '8': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 3.0}, '9': {'precision': 0.9642857142857143, 'recall':
0.9642857142857143, 'f1-score': 0.9642857142857143, 'support': 28.0},
'10': {'precision': 1.0, 'recall': 0.8333333333333334, 'f1-score':
0.9090909090909091, 'support': 18.0}, '11': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '12': {'precision':
0.9545454545454546, 'recall': 0.75, 'f1-score': 0.84, 'support':
56.0}, '13': {'precision': 1.0, 'recall': 0.55, 'f1-score':
0.7096774193548387, 'support': 20.0}, '14': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 2.0}, '15': {'precision':
```

0.9230769230769231, 'recall': 0.42857142857142855, 'f1-score':
0.5853658536585366, 'support': 28.0}, '16': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '17': {'precision':
0.9186046511627907, 'recall': 0.8359788359788359, 'f1-score':
0.8753462603878116, 'support': 189.0}, '18': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '19': {'precision':
0.8709677419354839, 'recall': 0.6136363636363636, 'f1-score': 0.72,
'support': 44.0}, '20': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 4.0}, '21': {'precision': 0.9888475836431226,
'recall': 0.9788408463661453, 'f1-score': 0.9838187702265372,
'support': 1087.0}, '22': {'precision': 1.0, 'recall': 0.2, 'f1-
score': 0.3333333333333333, 'support': 10.0}, '23': {'precision': 1.0,
'recall': 0.47058823529411764, 'f1-score': 0.64, 'support': 17.0},
'24': {'precision': 1.0, 'recall': 0.6857142857142857, 'f1-score':
0.8135593220338984, 'support': 35.0}, '25': {'precision':
0.9090909090909091, 'recall': 0.6666666666666666, 'f1-score':
0.7692307692307693, 'support': 30.0}, '26': {'precision': 0.976,
'recall': 0.8187919463087249, 'f1-score': 0.8905109489051095,
'support': 149.0}, '27': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 4.0}, '28': {'precision': 0.0, 'recall': 0.0, 'f1-
score': 0.0, 'support': 1.0}, '29': {'precision': 1.0, 'recall': 0.6,
'f1-score': 0.75, 'support': 5.0}, '30': {'precision': 1.0, 'recall':
0.3333333333333333, 'f1-score': 0.5, 'support': 6.0}, '31':
{'precision': 1.0, 'recall': 0.75, 'f1-score': 0.8571428571428571,
'support': 4.0}, '32': {'precision': 1.0, 'recall':
0.42857142857142855, 'f1-score': 0.6, 'support': 7.0}, '33':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0},
'34': {'precision': 0.8762886597938144, 'recall': 0.648854961832061,
'f1-score': 0.7456140350877193, 'support': 131.0}, '35': {'precision':
1.0, 'recall': 0.8333333333333334, 'f1-score': 0.9090909090909091,
'support': 12.0}, '36': {'precision': 0.9, 'recall':
0.6428571428571429, 'f1-score': 0.75, 'support': 14.0}, '37':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0},
'38': {'precision': 0.9166666666666666, 'recall': 0.5238095238095238,
'f1-score': 0.6666666666666666, 'support': 21.0}, '39': {'precision':
0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 2.0}, '40':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 14.0},
'41': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support':
3.0}, '42': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 1.0}, '43': {'precision': 0.75, 'recall': 0.375, 'f1-
score': 0.5, 'support': 24.0}, '44': {'precision': 0.0, 'recall': 0.0,
'f1-score': 0.0, 'support': 6.0}, '45': {'precision': 1.0, 'recall':
0.3157894736842105, 'f1-score': 0.48, 'support': 19.0}, '46':
{'precision': 0.8148148148148148, 'recall': 0.7374301675977654, 'f1-
score': 0.7741935483870968, 'support': 179.0}, '47': {'precision':
0.9259259259259259, 'recall': 0.7352941176470589, 'f1-score':
0.819672131147541, 'support': 34.0}, '48': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 4.0}, '49': {'precision':
0.7619047619047619, 'recall': 0.5333333333333333, 'f1-score':

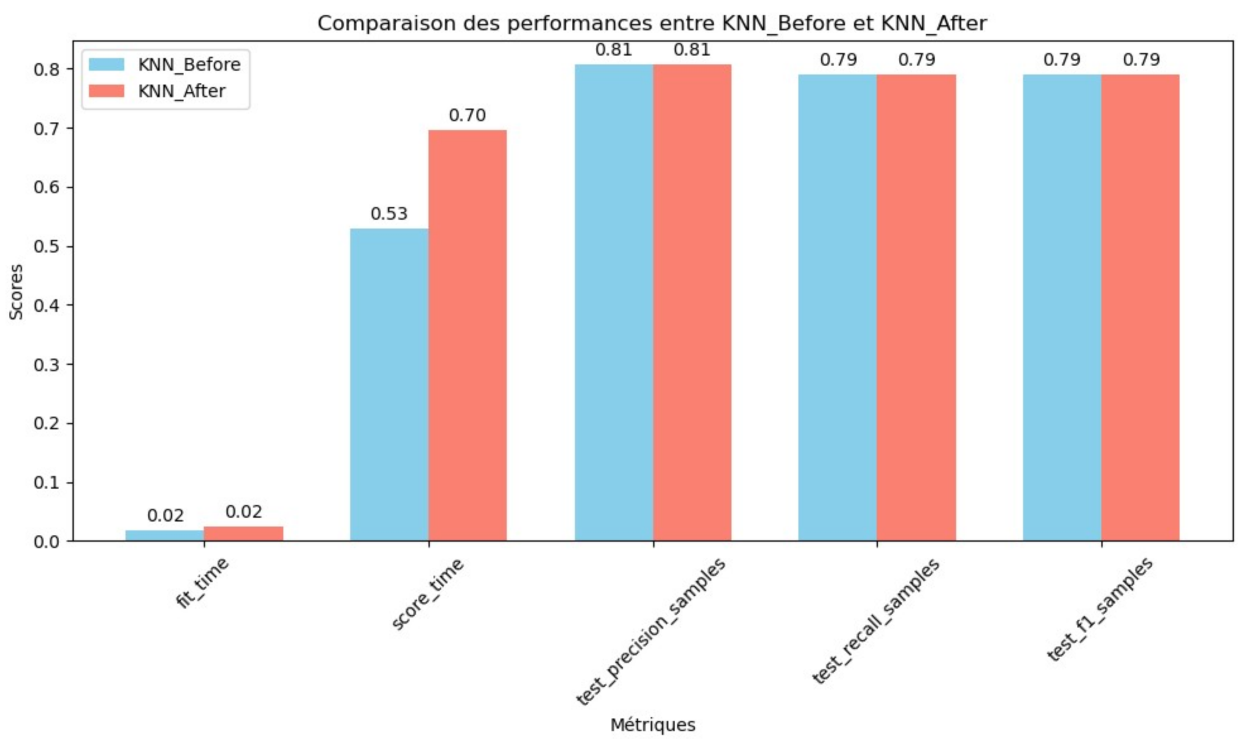
```
0.6274509803921569, 'support': 30.0}, '50': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '51': {'precision':
0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 2.0}, '52':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 2.0},
'53': {'precision': 1.0, 'recall': 0.16666666666666666, 'f1-score':
0.2857142857142857, 'support': 6.0}, '54': {'precision': 0.8125,
'recall': 0.5531914893617021, 'f1-score': 0.6582278481012658,
'support': 47.0}, '55': {'precision': 1.0, 'recall':
0.6363636363636364, 'f1-score': 0.7777777777777778, 'support': 11.0},
'56': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support':
1.0}, '57': {'precision': 1.0, 'recall': 0.6, 'f1-score': 0.75,
'support': 10.0}, '58': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 1.0}, '59': {'precision': 0.0, 'recall': 0.0, 'f1-
score': 0.0, 'support': 12.0}, '60': {'precision': 0.0, 'recall': 0.0,
'f1-score': 0.0, 'support': 7.0}, '61': {'precision': 1.0, 'recall':
0.3333333333333333, 'f1-score': 0.5, 'support': 3.0}, '62':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 3.0},
'63': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support':
1.0}, '64': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 3.0}, '65': {'precision': 1.0, 'recall':
0.4444444444444444, 'f1-score': 0.6153846153846154, 'support': 9.0},
'66': {'precision': 0.9166666666666666, 'recall': 0.6111111111111112,
'f1-score': 0.7333333333333333, 'support': 18.0}, '67': {'precision':
1.0, 'recall': 0.5, 'f1-score': 0.6666666666666666, 'support': 2.0},
'68': {'precision': 0.875, 'recall': 0.2916666666666667, 'f1-score':
0.4375, 'support': 24.0}, '69': {'precision': 1.0, 'recall': 0.75,
'f1-score': 0.8571428571428571, 'support': 12.0}, '70': {'precision':
0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '71':
{'precision': 0.9193548387096774, 'recall': 0.6404494382022472, 'f1-
score': 0.7549668874172185, 'support': 89.0}, '72': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 8.0}, '73': {'precision':
0.75, 'recall': 0.3, 'f1-score': 0.42857142857142855, 'support':
10.0}, '74': {'precision': 1.0, 'recall': 0.23076923076923078, 'f1-
score': 0.375, 'support': 13.0}, '75': {'precision': 0.0, 'recall':
0.0, 'f1-score': 0.0, 'support': 11.0}, '76': {'precision':
0.7619047619047619, 'recall': 0.48484848484848486, 'f1-score':
0.5925925925925926, 'support': 33.0}, '77': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 11.0}, '78': {'precision':
1.0, 'recall': 0.75, 'f1-score': 0.8571428571428571, 'support': 36.0},
'79': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support':
1.0}, '80': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 2.0}, '81': {'precision': 1.0, 'recall': 0.2, 'f1-score':
0.3333333333333333, 'support': 5.0}, '82': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 4.0}, '83': {'precision':
1.0, 'recall': 0.5833333333333334, 'f1-score': 0.7368421052631579,
'support': 12.0}, '84': {'precision': 0.8787878787878788, 'recall':
0.7435897435897436, 'f1-score': 0.8055555555555556, 'support': 117.0},
'85': {'precision': 0.9411764705882353, 'recall': 0.43243243243243246,
'f1-score': 0.5925925925925926, 'support': 37.0}, '86': {'precision':
```

```
0.9152542372881356, 'recall': 0.7605633802816901, 'f1-score':
0.8307692307692308, 'support': 71.0}, '87': {'precision': 1.0,
'recall': 0.6, 'f1-score': 0.75, 'support': 10.0}, '88': {'precision':
0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 14.0}, '89':
{'precision': 1.0, 'recall': 0.46153846153846156, 'f1-score':
0.631578947368421, 'support': 13.0}, 'micro avg': {'precision':
0.9528728211749515, 'recall': 0.7884615384615384, 'f1-score':
0.8629055831628178, 'support': 3744.0}, 'macro avg': {'precision':
0.5887609931425809, 'recall': 0.3650654059513509, 'f1-score':
0.436502821543132, 'support': 3744.0}, 'weighted avg': {'precision':
0.9156073554970184, 'recall': 0.7884615384615384, 'f1-score':
0.8368355156799253, 'support': 3744.0}, 'samples avg': {'precision':
0.8740973832394833, 'recall': 0.8560203368086773, 'f1-score':
0.8575567002281145, 'support': 3744.0}}
```

```
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

```
                    0          1          2          3          4    5  \
precision    0.982734   1.000000   1.000000   0.909091   0.875000  0.0   

recall       0.949930   0.434783   0.642857   0.666667   0.388889  0.0   

f1-score     0.966054   0.606061   0.782609   0.769231   0.538462  0.0   

support    719.000000  23.000000  14.000000  30.000000  18.000000  1.0   


                   6         7    8          9  ...          84  
85  \
precision   1.000000  1.000000  0.0   0.964286  ...    0.878788   
0.941176   
recall      0.944444  0.500000  0.0   0.964286  ...    0.743590   
0.432432   
f1-score    0.971429  0.666667  0.0   0.964286  ...    0.805556   
0.592593   
support    18.000000  2.000000  3.0  28.000000  ...  117.000000  
37.000000
```

```
                    86     87    88          89     micro avg     macro avg  \
precision   0.915254   1.00   0.0   1.000000      0.952873      0.588761

recall      0.760563   0.60   0.0   0.461538      0.788462      0.365065

f1-score    0.830769   0.75   0.0   0.631579      0.862906      0.436503

support    71.000000  10.00  14.0  13.000000   3744.000000   3744.000000


           weighted avg  samples avg
precision      0.915607     0.874097
recall         0.788462     0.856020
f1-score       0.836836     0.857557
support     3744.000000  3744.000000

[4 rows x 94 columns]

compare_classifiers(classifier1_scores=scores_knn,
classifier2_scores=scores_knn_skb, names=['KNN_Before', 'KNN_After'])
```



Random Forest

```
rf_classifier = RandomForestClassifier(n_estimators=100)
scores_rf_kbs, report_rf_kbs = evaluate_classifier(rf_classifier,
train_docs=vect_train_docs, train_labels=train_labels,
```

```
test_docs=vect_test_docs, test_labels=test_labels)
display_classifier_results(scores_rf_kbs, "RF", report=report_rf_kbs)
pd.DataFrame(report_rf_kbs)
```

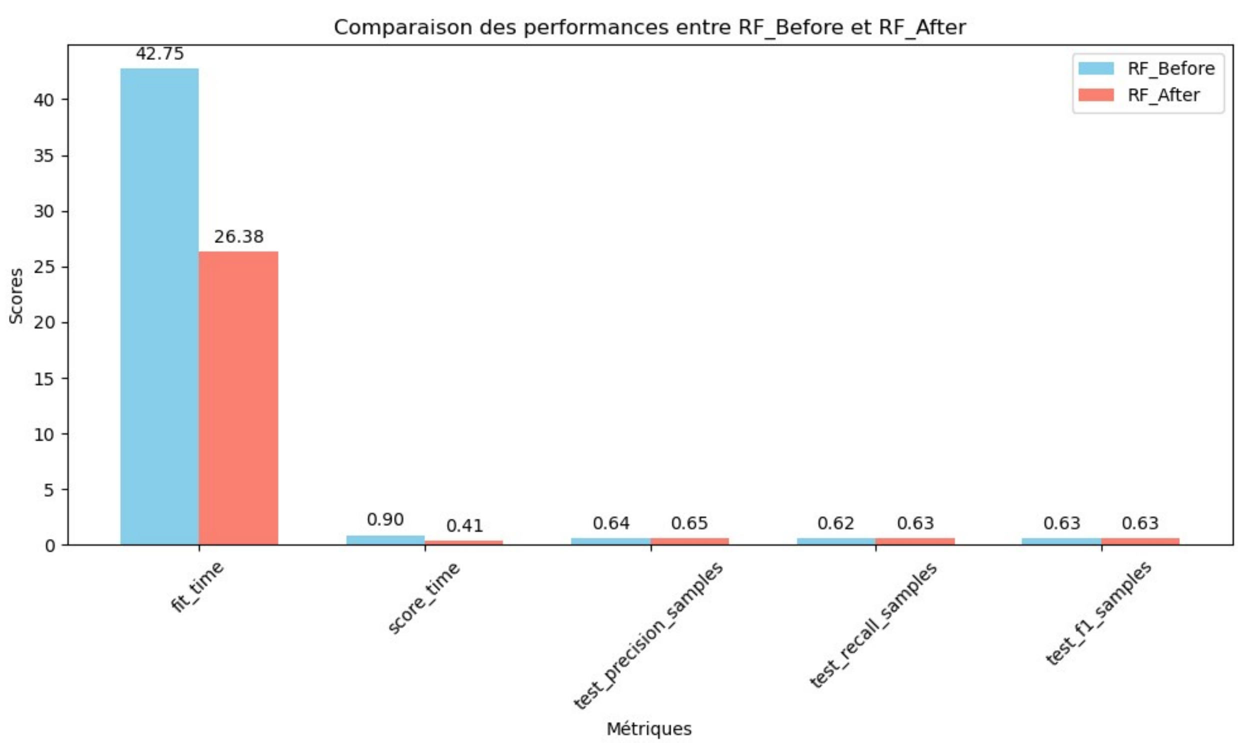
```
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))


--- Résultats pour RF ---
Fit time : [26.9270649  26.2157197  25.99020052]
Score time : [0.4175036  0.42927027 0.37000227]
test_precision_samples : [0.64295367 0.63229086 0.66825029]
test_recall_samples :  [0.62135457 0.61108966 0.65086233]
test_f1_samples : [0.62600055 0.61628162 0.65482567]

Rapport de classification :
{'0': {'precision': 0.9827338129496402, 'recall': 0.9499304589707928,
'f1-score': 0.9660537482319661, 'support': 719.0}, '1': {'precision':
1.0, 'recall': 0.43478260869565216, 'f1-score': 0.6060606060606061,
'support': 23.0}, '2': {'precision': 1.0, 'recall':
0.6428571428571429, 'f1-score': 0.782608695652174, 'support': 14.0},
'3': {'precision': 0.9090909090909091, 'recall': 0.6666666666666666,
'f1-score': 0.7692307692307693, 'support': 30.0}, '4': {'precision':
0.875, 'recall': 0.3888888888888889, 'f1-score': 0.5384615384615384,
'support': 18.0}, '5': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 1.0}, '6': {'precision': 1.0, 'recall':
0.9444444444444444, 'f1-score': 0.9714285714285714, 'support': 18.0},
'7': {'precision': 1.0, 'recall': 0.5, 'f1-score': 0.6666666666666666,
'support': 2.0}, '8': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 3.0}, '9': {'precision': 0.9642857142857143, 'recall':
0.9642857142857143, 'f1-score': 0.9642857142857143, 'support': 28.0},
'10': {'precision': 1.0, 'recall': 0.8333333333333334, 'f1-score':
0.9090909090909091, 'support': 18.0}, '11': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '12': {'precision':
```

```
0.9545454545454546, 'recall': 0.75, 'f1-score': 0.84, 'support':
56.0}, '13': {'precision': 1.0, 'recall': 0.55, 'f1-score':
0.7096774193548387, 'support': 20.0}, '14': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 2.0}, '15': {'precision':
0.9230769230769231, 'recall': 0.42857142857142855, 'f1-score':
0.5853658536585366, 'support': 28.0}, '16': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '17': {'precision':
0.9186046511627907, 'recall': 0.8359788359788359, 'f1-score':
0.8753462603878116, 'support': 189.0}, '18': {'precision': 0.0,
'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '19': {'precision':
0.8709677419354839, 'recall': 0.6136363636363636, 'f1-score': 0.72,
'support': 44.0}, '20': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 4.0}, '21': {'precision': 0.9888475836431226,
'recall': 0.9788408463661453, 'f1-score': 0.9838187702265372,
'support': 1087.0}, '22': {'precision': 1.0, 'recall': 0.2, 'f1-
score': 0.3333333333333333, 'support': 10.0}, '23': {'precision': 1.0,
'recall': 0.47058823529411764, 'f1-score': 0.64, 'support': 17.0},
'24': {'precision': 1.0, 'recall': 0.6857142857142857, 'f1-score':
0.8135593220338984, 'support': 35.0}, '25': {'precision':
0.9090909090909091, 'recall': 0.6666666666666666, 'f1-score':
0.7692307692307693, 'support': 30.0}, '26': {'precision': 0.976,
'recall': 0.8187919463087249, 'f1-score': 0.8905109489051095,
'support': 149.0}, '27': {'precision': 0.0, 'recall': 0.0, 'f1-score':
0.0, 'support': 4.0}, '28': {'precision': 0.0, 'recall': 0.0, 'f1-
score': 0.0, 'support': 1.0}, '29': {'precision': 1.0, 'recall': 0.6,
'f1-score': 0.75, 'support': 5.0}, '30': {'precision': 1.0, 'recall':
0.3333333333333333, 'f1-score': 0.5, 'support': 6.0}, '31':
{'precision': 1.0, 'recall': 0.75, 'f1-score': 0.8571428571428571,
'support': 4.0}, '32': {'precision': 1.0, 'recall':
0.42857142857142855, 'f1-score': 0.6, 'support': 7.0}, '33':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0},
'34': {'precision': 0.8762886597938144, 'recall': 0.648854961832061,
'f1-score': 0.7456140350877193, 'support': 131.0}, '35': {'precision':
1.0, 'recall': 0.8333333333333334, 'f1-score': 0.9090909090909091,
'support': 12.0}, '36': {'precision': 0.9, 'recall':
0.6428571428571429, 'f1-score': 0.75, 'support': 14.0}, '37':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0},
'38': {'precision': 0.9166666666666666, 'recall': 0.5238095238095238,
'f1-score': 0.6666666666666666, 'support': 21.0}, '39': {'precision':
0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 2.0}, '40':
{'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 14.0},
'41': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support':
3.0}, '42': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,
'support': 1.0}, '43': {'precision': 0.75, 'recall': 0.375, 'f1-
score': 0.5, 'support': 24.0}, '44': {'precision': 0.0, 'recall': 0.0,
'f1-score': 0.0, 'support': 6.0}, '45': {'precision': 1.0, 'recall':
0.3157894736842105, 'f1-score': 0.48, 'support': 19.0}, '46':
{'precision': 0.8148148148148148, 'recall': 0.7374301675977654, 'f1-
score': 0.7741935483870968, 'support': 179.0}, '47': {'precision':
```

0.9259259259259259, 'recall': 0.7352941176470589, 'f1-score': 0.819672131147541, 'support': 34.0}, '48': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 4.0}, '49': {'precision': 0.7619047619047619, 'recall': 0.5333333333333333, 'f1-score': 0.6274509803921569, 'support': 30.0}, '50': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '51': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 2.0}, '52': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 2.0}, '53': {'precision': 1.0, 'recall': 0.16666666666666666, 'f1-score': 0.2857142857142857, 'support': 6.0}, '54': {'precision': 0.8125, 'recall': 0.5531914893617021, 'f1-score': 0.6582278481012658, 'support': 47.0}, '55': {'precision': 1.0, 'recall': 0.6363636363636364, 'f1-score': 0.7777777777777778, 'support': 11.0}, '56': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '57': {'precision': 1.0, 'recall': 0.6, 'f1-score': 0.75, 'support': 10.0}, '58': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '59': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 12.0}, '60': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 7.0}, '61': {'precision': 1.0, 'recall': 0.3333333333333333, 'f1-score': 0.5, 'support': 3.0}, '62': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 3.0}, '63': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '64': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 3.0}, '65': {'precision': 1.0, 'recall': 0.4444444444444444, 'f1-score': 0.6153846153846154, 'support': 9.0}, '66': {'precision': 0.9166666666666666, 'recall': 0.6111111111111112, 'f1-score': 0.7333333333333333, 'support': 18.0}, '67': {'precision': 1.0, 'recall': 0.5, 'f1-score': 0.6666666666666666, 'support': 2.0}, '68': {'precision': 0.875, 'recall': 0.2916666666666667, 'f1-score': 0.4375, 'support': 24.0}, '69': {'precision': 1.0, 'recall': 0.75, 'f1-score': 0.8571428571428571, 'support': 12.0}, '70': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '71': {'precision': 0.9193548387096774, 'recall': 0.6404494382022472, 'f1-score': 0.7549668874172185, 'support': 89.0}, '72': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 8.0}, '73': {'precision': 0.75, 'recall': 0.3, 'f1-score': 0.42857142857142855, 'support': 10.0}, '74': {'precision': 1.0, 'recall': 0.23076923076923078, 'f1-score': 0.375, 'support': 13.0}, '75': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 11.0}, '76': {'precision': 0.7619047619047619, 'recall': 0.48484848484848486, 'f1-score': 0.5925925925925926, 'support': 33.0}, '77': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 11.0}, '78': {'precision': 1.0, 'recall': 0.75, 'f1-score': 0.8571428571428571, 'support': 36.0}, '79': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 1.0}, '80': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 2.0}, '81': {'precision': 1.0, 'recall': 0.2, 'f1-score': 0.3333333333333333, 'support': 5.0}, '82': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 4.0}, '83': {'precision': 1.0, 'recall': 0.5833333333333334, 'f1-score': 0.7368421052631579,

```
'support': 12.0}, '84': {'precision': 0.8787878787878788, 'recall':
0.7435897435897436, 'f1-score': 0.8055555555555556, 'support': 117.0},
'85': {'precision': 0.9411764705882353, 'recall': 0.43243243243243246,
'f1-score': 0.5925925925925926, 'support': 37.0}, '86': {'precision':
0.9152542372881356, 'recall': 0.7605633802816901, 'f1-score':
0.8307692307692308, 'support': 71.0}, '87': {'precision': 1.0,
'recall': 0.6, 'f1-score': 0.75, 'support': 10.0}, '88': {'precision':
0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 14.0}, '89':
{'precision': 1.0, 'recall': 0.46153846153846156, 'f1-score':
0.631578947368421, 'support': 13.0}, 'micro avg': {'precision':
0.9528728211749515, 'recall': 0.7884615384615384, 'f1-score':
0.8629055831628178, 'support': 3744.0}, 'macro avg': {'precision':
0.5887609931425809, 'recall': 0.3650654059513509, 'f1-score':
0.436502821543132, 'support': 3744.0}, 'weighted avg': {'precision':
0.9156073554970184, 'recall': 0.7884615384615384, 'f1-score':
0.8368355156799253, 'support': 3744.0}, 'samples avg': {'precision':
0.8740973832394833, 'recall': 0.8560203368086773, 'f1-score':
0.8575567002281145, 'support': 3744.0}}

c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))

                    0          1          2          3          4    5
\
precision    0.982734   1.000000   1.000000   0.909091   0.875000  0.0

recall       0.949930   0.434783   0.642857   0.666667   0.388889  0.0

f1-score     0.966054   0.606061   0.782609   0.769231   0.538462  0.0

support    719.000000  23.000000  14.000000  30.000000  18.000000  1.0


                   6         7    8          9  ...          84
85  \
precision   1.000000  1.000000  0.0   0.964286  ...    0.878788
0.941176
recall      0.944444  0.500000  0.0   0.964286  ...    0.743590
0.432432
f1-score    0.971429  0.666667  0.0   0.964286  ...    0.805556
```

```
0.592593
support    18.000000  2.000000  3.0  28.000000  ...  117.000000
37.000000

                  86     87    88          89    micro avg    macro avg
\
precision   0.915254   1.00   0.0   1.000000     0.952873     0.588761

recall      0.760563   0.60   0.0   0.461538     0.788462     0.365065

f1-score    0.830769   0.75   0.0   0.631579     0.862906     0.436503

support    71.000000  10.00  14.0  13.000000  3744.000000  3744.000000


           weighted avg  samples avg
precision      0.915607     0.874097
recall         0.788462     0.856020
f1-score       0.836836     0.857557
support     3744.000000  3744.000000

[4 rows x 94 columns]

compare_classifiers(classifier1_scores=scores_rf,
classifier2_scores=scores_rf_kbs, names=['RF_Before', 'RF_After'])
```



Perceptron multicouche

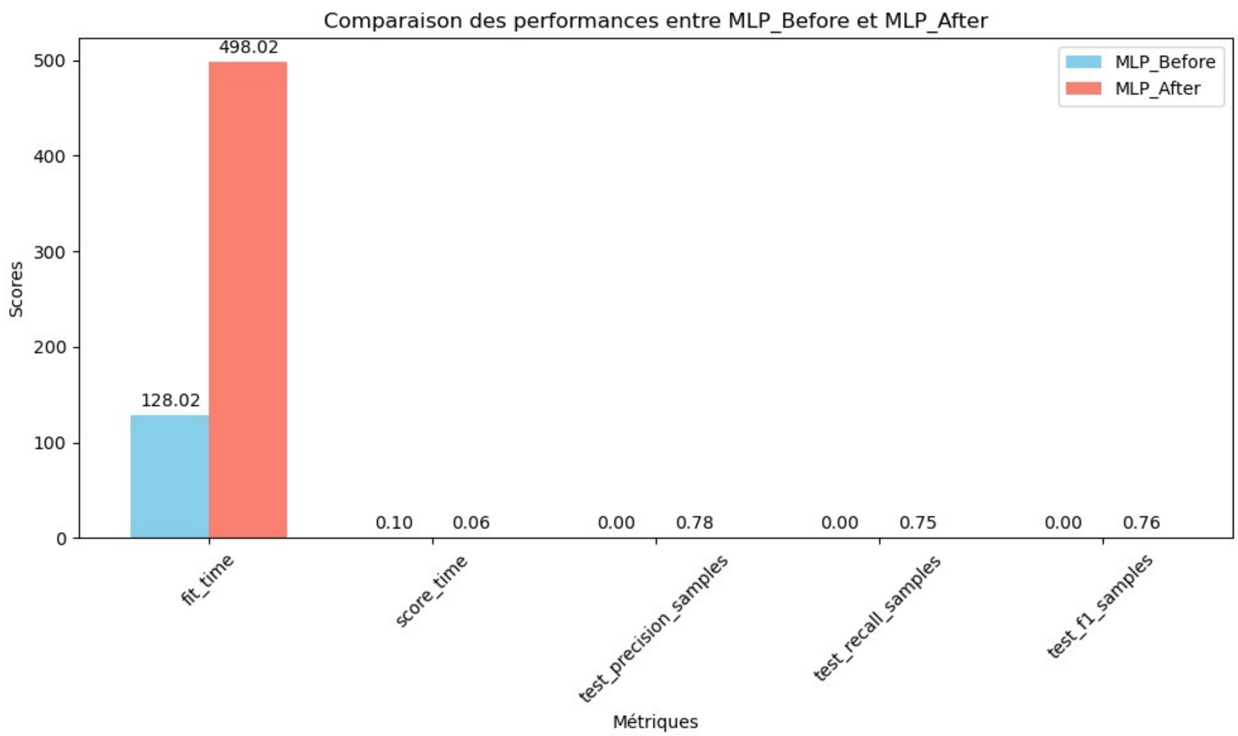
```
mlp_classifier =  MLPClassifier(activation='relu', solver='adam',
random_state=42, max_iter=100)
scores_mlp_skb, report_mlp_skb = evaluate_classifier(mlp_classifier,
train_docs=vect_train_docs, train_labels=train_labels,
test_docs=vect_test_docs, test_labels=test_labels)
display_classifier_results(scores_mlp_skb, "mlp",
report=report_mlp_skb)
pd.DataFrame(report_mlp_skb)
```

```
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\neural_network\
_multilayer_perceptron.py:697: UserWarning: Training interrupted by
user.
  warnings.warn("Training interrupted by user.")
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\neural_network\
_multilayer_perceptron.py:697: UserWarning: Training interrupted by
user.
  warnings.warn("Training interrupted by user.")
c:\Users\dscon\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1531: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

```
report_mlp_skb
```

```
----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[1], line 1
----> 1 report_mlp_skb

NameError: name 'report_mlp_skb' is not defined
```

```
compare_classifiers(classifier1_scores=scores_mlp,
classifier2_scores=scores_mlp_skb, names=['MLP_Before', 'MLP_After'])
```



Cross-Validation

Summary

```
import pandas as pd
import matplotlib.pyplot as plt

# Liste des classificateurs et initialisation des résultats
classifiers = ['SVM', 'KNN', 'Random Forest', 'MLP']
results_before = []
results_after = []

# Helper function pour obtenir la moyenne de la précision ou traiter
les valeurs float
def get_precision_mean(scores, key='test_precision_samples'):
    return scores[key].mean()

# Ajout des résultats de précision avant la sélection de
caractéristiques
results_before.append(get_precision_mean(scores_svm))
results_before.append(get_precision_mean(scores_knn))
results_before.append(get_precision_mean(scores_rf))
results_before.append(get_precision_mean(scores_mlp))
```

```python
# Ajout des résultats de précision après la sélection de
caractéristiques
results_after.append(get_precision_mean(scores_SVM))
results_after.append(get_precision_mean(scores_knn_skb))
results_after.append(get_precision_mean(scores_rf_kbs))
results_after.append(get_precision_mean(scores_mlp_skb))

# Création du DataFrame récapitulatif
summary_data = {
    'Classifier': classifiers,
    'Precision Before': results_before,
    'Precision After': results_after
}

summary_df = pd.DataFrame(summary_data)

# Affichage du tableau récapitulatif
print("Résumé de la Précision Avant et Après la Sélection de
Caractéristiques")
summary_df
```

```
Résumé de la Précision Avant et Après la Sélection de Caractéristiques
```

	Classifier	Precision Before	Precision After
0	SVM	0.876636	0.859441
1	KNN	0.806617	0.806617
2	Random Forest	0.648175	0.644635
3	MLP	0.000000	0.775588

```python
scores_svm
```

```
{'fit_time': array([1.01328444, 1.14886451, 1.00400114]),
 'score_time': array([0.09167933, 0.07272625, 0.07480359]),
 'test_precision_samples': array([0.86643493, 0.88022803,
0.88324527]),
 'test_recall_samples': array([0.84632651, 0.86370184, 0.86209956]),
 'test_f1_samples': array([0.84841431, 0.86375468, 0.86505655])}
```