In [ ]:
```
import urllib.request
import bs4 as BeautifulSoup
import nltk
from string import punctuation
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize
```

In [ ]:
```
text = urllib.request.urlopen('https://en.wikipedia.org/wiki/Machine_learning')
```

In [ ]:
```
article = text.read()
```

In [ ]:
```
article
```

In [ ]:
```
article_parsed = BeautifulSoup.BeautifulSoup(article,'html.parser')
```

In [ ]:
```
paragraphs = article_parsed.find_all('p')
```

In [ ]:
```
article_content = ''
for p in paragraphs:
    article_content += p.text
```

In [ ]:
```
article_content
```

In [ ]:
```
tokens = word_tokenize(article_content)
```

In [ ]:
```
nltk.download("stopwords")
stop_words = stopwords.words('english')
```

In [ ]:
```
punctuation = punctuation + '\n'
punctuation
```

In [ ]:
```
word_frequencies = {}
for word in tokens:
    if word.lower() not in stop_words:
        if word.lower() not in punctuation:
            if word not in word_frequencies.keys():
                word_frequencies[word] = 1
            else:
                word_frequencies[word] += 1
```

In [ ]:
```
word_frequencies
```

In [ ]:
```
max_frequency = max(word_frequencies.values())
print(max_frequency)
```

In [ ]:
```
for word in word_frequencies.keys():
    word_frequencies[word] = word_frequencies[word]/max_frequency
```

In [ ]:
```
print(word_frequencies)
```

In [ ]:
```
sent_token = sent_tokenize(article_content)
sent_token
```

In [ ]:
```
sentence_scores = {}
for sent in sent_token:
    sentence = sent.split(" ")
    for word in sentence:
        if word.lower() in word_frequencies.keys():
            if sent not in sentence_scores.keys():
                sentence_scores[sent] = word_frequencies[word.lower()]
            else:
                sentence_scores[sent] += word_frequencies[word.lower()]
```

In [ ]:
```
sentence_scores
```

In [ ]:
```
from heapq import nlargest
```

In [ ]:
```
select_length = int(len(sent_token)*0.3)
select_length
```

In [ ]:
```
summary = nlargest(select_length, sentence_scores, key = sentence_scores.get)
```

In [ ]:
```
final_summary = [word for word in summary]
summary = ' '.join(final_summary)
```

In [ ]:
```
summary
```

In [ ]:
```
len(article_content)
```

In [ ]:
```
len(summary)
```

In [ ]:
```

```