FAKE NEWS DETECTION

Rachata Kaewviset
6410406843

Goals

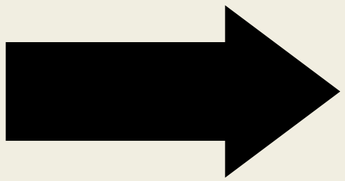
Logistic Regression

because it can estimate the likelihood of a given text being either real or fake news. It's especially useful for analyzing text data, performs well in binary classification tasks

Text Cleansing

1.Lower case all text

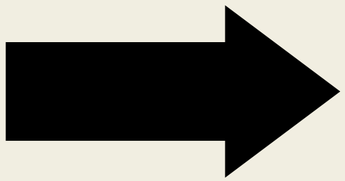
```
data['text'] = data['text'].str.lower()
```

Get  get

Text Cleansing

2.Removing words with only one letter

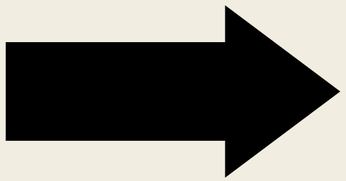
```
data['text'] = data['text'].apply(lambda x: ' '.join([word for word in x.split() if len(word) > 1]))
```

A 

Text Cleansing

3.Remove words that contain numbers

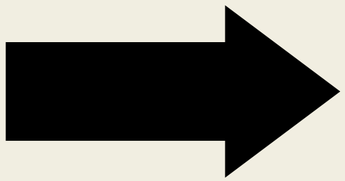
```
data['text'] = data['text'].apply(lambda x: ' '.join([word for word in x.split() if not re.match( pattern: r'.*\d.*', word)
```

25th 

Text Cleansing

4.Tokenizing the text ,Remove empty tokens and removing punctuation

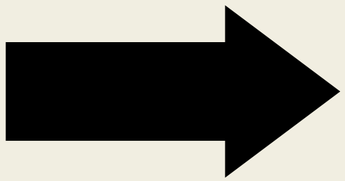
```
from nltk.tokenize import word_tokenize
data['text'].apply(lambda x: ' '.join([word for word in word_tokenize(x) if word.isalnum()]))
```

"get  get

Text Cleansing

5.Remove all stop words

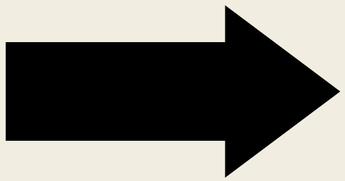
```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
data['text'] = data['text'].apply(lambda x: ' '.join([word for word in x.split() if word not in stop_words]))
```

latest from today  latest today

Text Cleansing

6.POS tagging the text

```
from nltk import pos_tag
data['text'] = data['text'].apply(lambda x: pos_tag(word_tokenize(x)))
```

get latest today  ('get', 'VB'), ('latest', 'JJS'), ('today', 'NN')

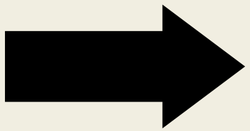
Text Cleansing

7.Lemmatizing the text

```
data['text'] = data['text'].apply(lambda x: ' '.join([lemmatizer.lemmatize(word, pos=get_wordnet_pos(tag)) for word, tag in x]))
```

```
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return 'a'  # Adjective
    elif tag.startswith('V'):
        return 'v'  # Verb
    elif tag.startswith('N'):
        return 'n'  # Noun
    elif tag.startswith('R'):
        return 'r'  # Adverb
    else:
        return 'n'
```

('get', 'VB'), ('latest', 'JJS'), ('today', 'NN')



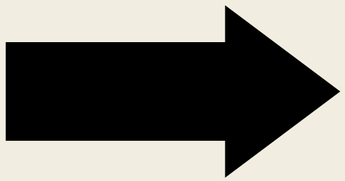
get late today

Feature Extraction

TF-IDF vectorizer = TF(t,d) * IDF(t)

TF(t,d) = number of t appear in Document/all word in Document

IDF(t) = log [(1 + n) / (1 + number of Document that appear t)] + 1

texts  numbers

Feature Extraction

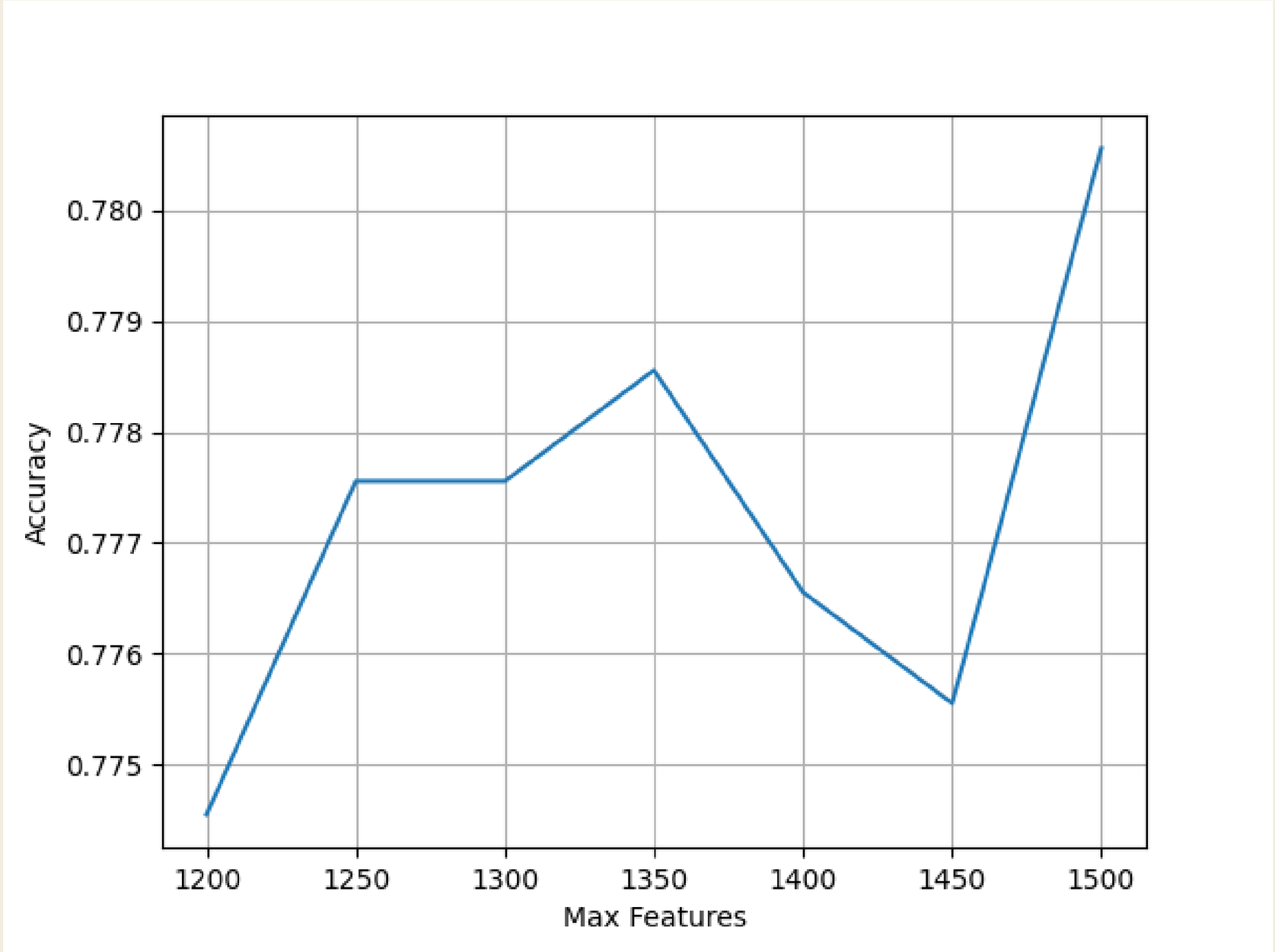
limit feature

some word are rarely to appear it might be consider as a noise

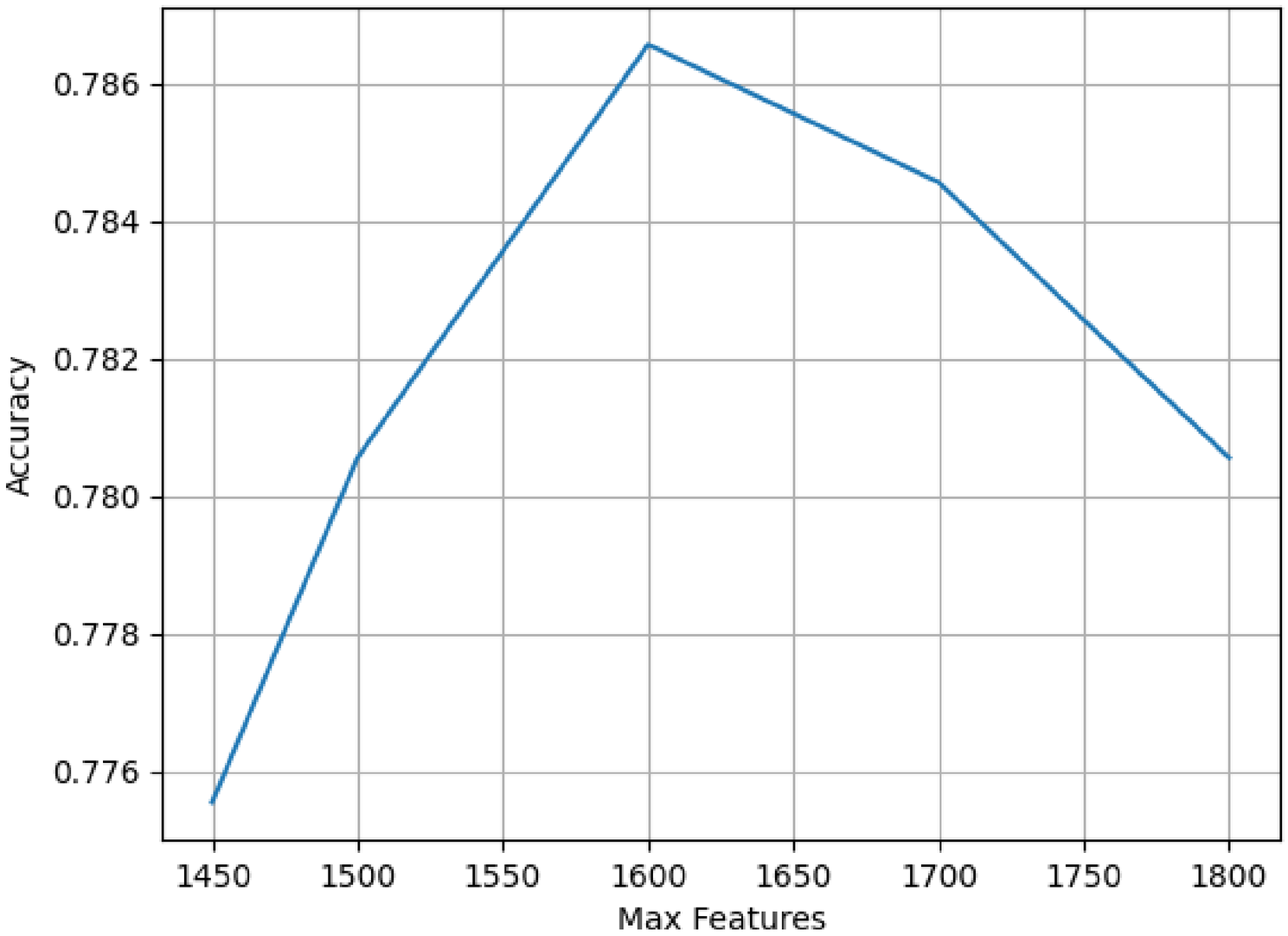
```
vectorizer = TfidfVectorizer(stop_words='english',max_features=1600)
```

*use only top 1600 word that appear on document

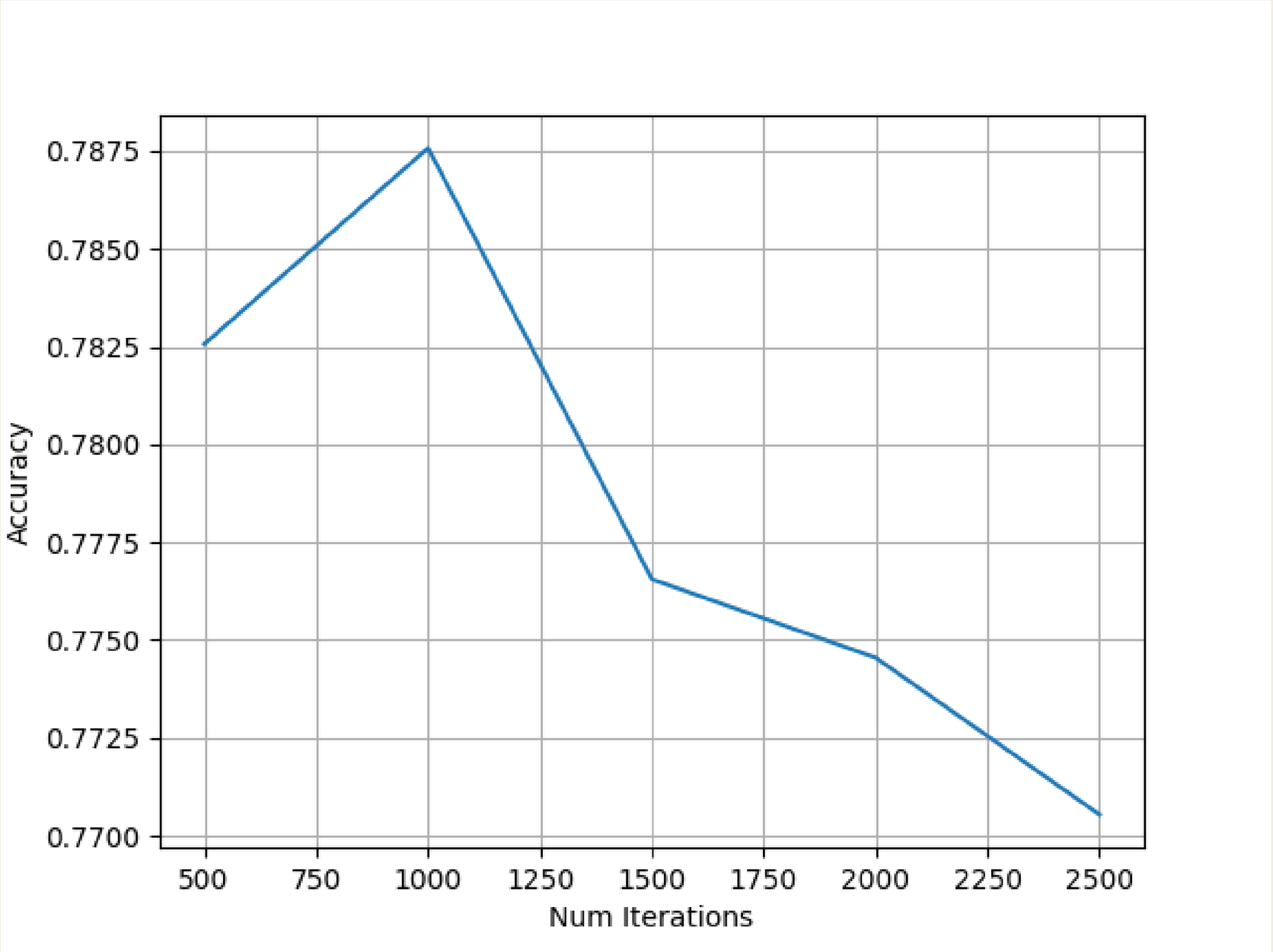
Tuning Parameter Max Features



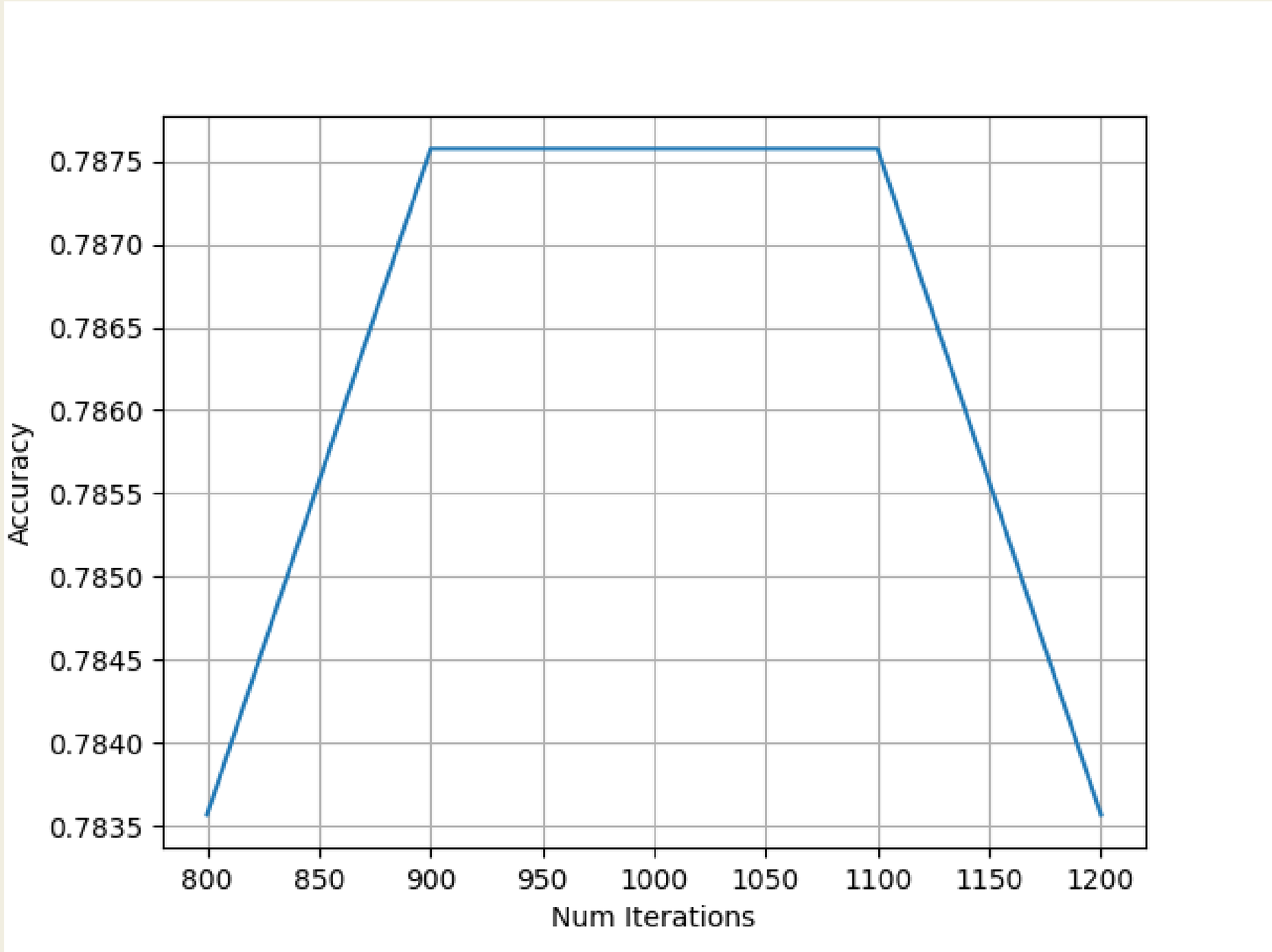
Tuning Parameter Max Features



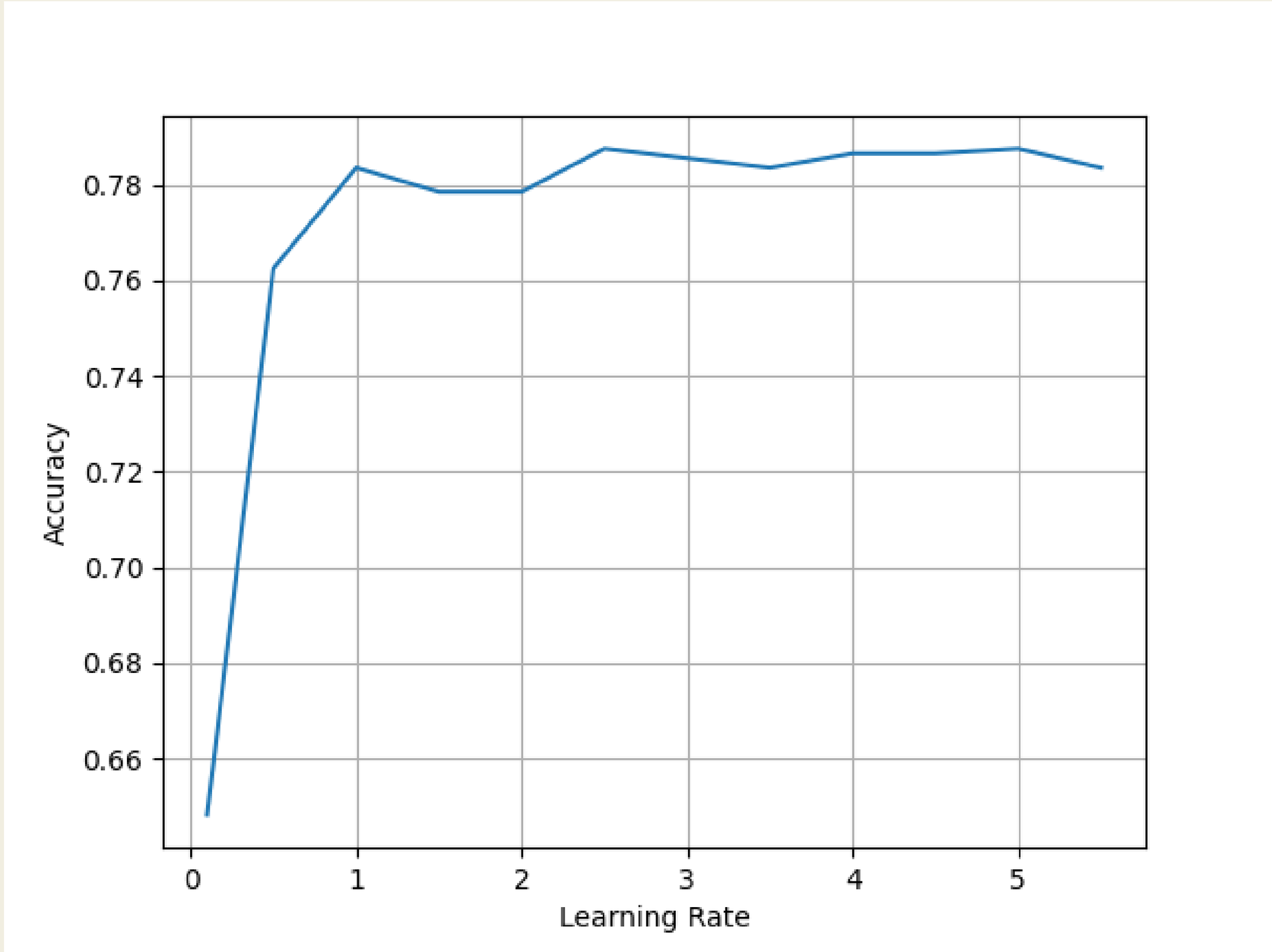
Tuning Parameter Num iterations



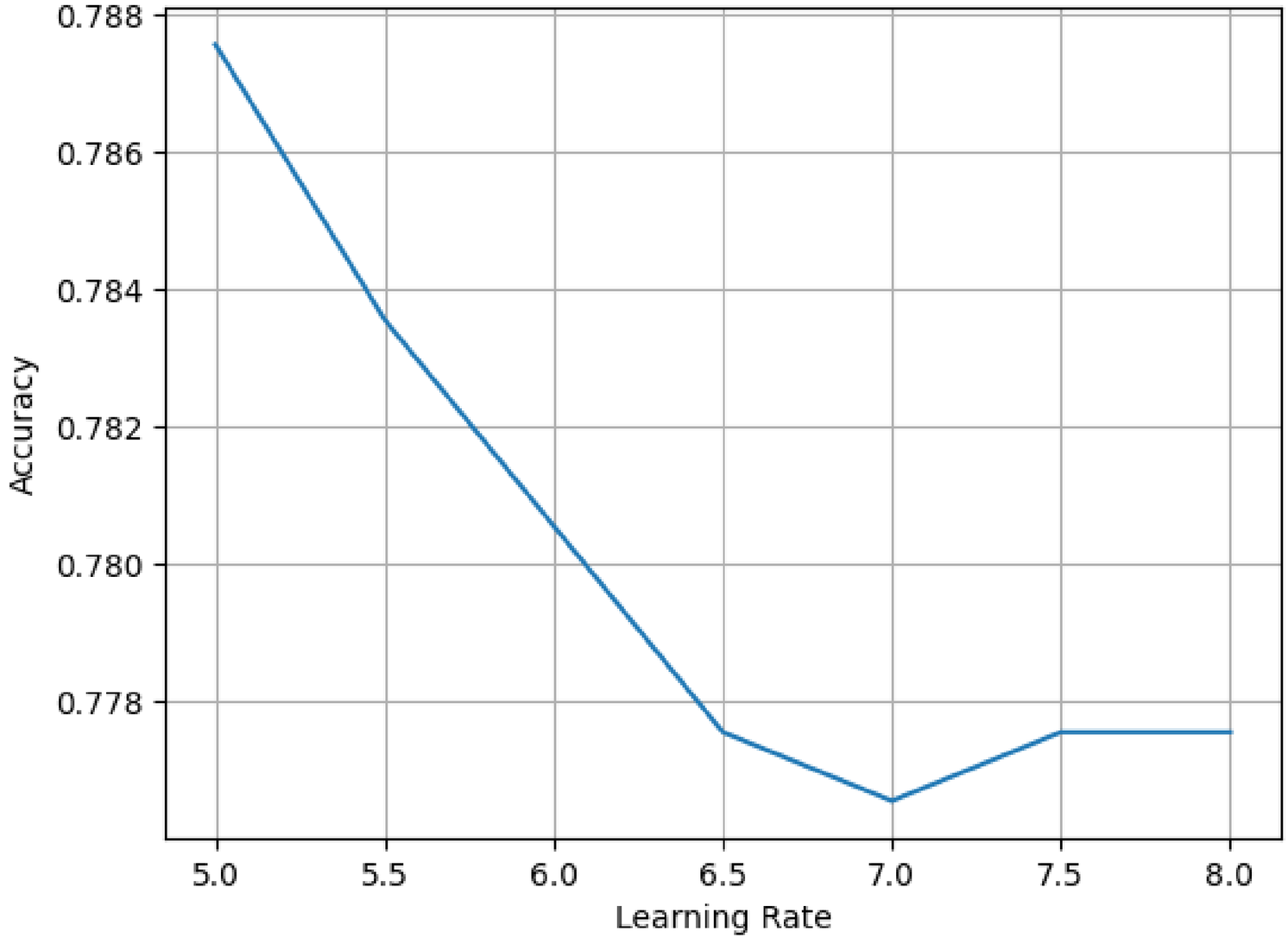
Tuning Parameter Num iterations



Tuning Parameter Learning Rate



Tuning Parameter Learning Rate



Conclusion

max features = 1600
learning rate = 5
num iterations = 1000

accacuracy ≈ 0.787