In [2]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in d:\programs\python\lib\site-packages (3.9.1)
Requirement already satisfied: click in d:\programs\python\lib\site-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in d:\programs\python\lib\site-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in d:\programs\python\lib\site-packages (from nltk) (2024.7.24)
Requirement already satisfied: tqdm in d:\programs\python\lib\site-packages (from nltk) (4.66.5)
Requirement already satisfied: colorama in d:\programs\python\lib\site-packages (from click->nltk) (0.4.6)
```

In [3]:
```
# Importing libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

import random

from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score

from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

In [4]:
```
# nltk - wordnet and lexical database: Setting up WordNet in NLTK
import nltk
nltk.download('wordnet')

# For wordent with different language
nltk.download('omw-1.4')

# For tokenization
nltk.download('punkt')

# POS tagger: part of speech
nltk.download('averaged_perceptron_tagger')
nltk.download('averaged_perceptron_tagger_eng')

# For stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package wordnet to C:\Users\sheila
[nltk_data]     brown\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to C:\Users\sheila
[nltk_data]     brown\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package punkt to C:\Users\sheila
[nltk_data]     brown\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\sheila brown\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     C:\Users\sheila brown\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger_eng is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package stopwords to C:\Users\sheila
[nltk_data]     brown\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[4]: True

In [5]:
```
# Implement lesk Algorithm - is simple knowledge-based WSD method - lesk()
from nltk.wsd import lesk
from nltk.corpus import wordnet as wn
```

In [7]:
```
# Importing CSV file
df = pd.read_csv('WSD_Sentences.csv')
pd.set_option('display.max_rows', None)
df.head(10)
```

Out[7]:

| | sn | sentence/context | polysemy_word | definition |
|---|---|---|---|---|
| 0 | 1 | I have bank account. | bank | depository_financial_institution.n.01 |
| 1 | 2 | Loan amount is approved by the bank. | bank | depository_financial_institution.n.01 |
| 2 | 3 | He returned to office after he deposited cash ... | bank | depository_financial_institution.n.01 |
| 3 | 4 | They started using new software in their bank. | bank | depository_financial_institution.n.01 |
| 4 | 5 | he went to bank balance inquiry. | bank | depository_financial_institution.n.01 |
| 5 | 6 | I wonder why some bank have more interest rate... | bank | depository_financial_institution.n.01 |
| 6 | 7 | You have to deposit certain percentage of your... | bank | depository_financial_institution.n.01 |
| 7 | 8 | He took loan from a Bank. | bank | depository_financial_institution.n.01 |
| 8 | 9 | he is waking along the river bank. | bank | bank.n.01 |
| 9 | 10 | The red boat in the bank is already sold. | bank | bank.n.01 |

In [8]:
```
# Selecting data with 'bank' as the polysemy word
word = 'bank'
bank_df = df[df['polysemy_word'] == word]
bank_df = bank_df.reset_index(drop=True, inplace=False)

# Display the updated DataFrame
bank_df
```

Out[8]:

| | sn | sentence/context | polysemy_word | definition |
|---|---|---|---|---|
| 0 | 1 | I have bank account. | bank | depository_financial_institution.n.01 |
| 1 | 2 | Loan amount is approved by the bank. | bank | depository_financial_institution.n.01 |
| 2 | 3 | He returned to office after he deposited cash ... | bank | depository_financial_institution.n.01 |
| 3 | 4 | They started using new software in their bank. | bank | depository_financial_institution.n.01 |
| 4 | 5 | he went to bank balance inquiry. | bank | depository_financial_institution.n.01 |
| 5 | 6 | I wonder why some bank have more interest rate... | bank | depository_financial_institution.n.01 |
| 6 | 7 | You have to deposit certain percentage of your... | bank | depository_financial_institution.n.01 |
| 7 | 8 | He took loan from a Bank. | bank | depository_financial_institution.n.01 |
| 8 | 9 | he is waking along the river bank. | bank | bank.n.01 |
| 9 | 10 | The red boat in the bank is already sold. | bank | bank.n.01 |
| 10 | 11 | Spending time on the bank of Kaligandaki river... | bank | bank.n.01 |
| 11 | 12 | He was sitting on sea bank with his friend | bank | bank.n.01 |
| 12 | 13 | She has always dreamed of spending a vacation ... | bank | bank.n.01 |
| 13 | 14 | Bank of a river is very pleasant place to enjoy. | bank | bank.n.01 |

In [9]:
```
# Selecting data with 'plant' as the polysemy word
word = 'plant'
plant_df = df[df['polysemy_word'] == word]
plant_df = plant_df.reset_index(drop=True, inplace=False)

# Display the updated DataFrame
plant_df
```

Out[9]:

| | sn | sentence/context | polysemy_word | definition |
|---|---|---|---|---|
| 0 | 493 | They built a large plant to manufacture automo... | plant | plant.n.01 |
| 1 | 494 | He laid at that plant. | plant | plant.n.01 |
| 2 | 495 | Our company is planning to build a new chemica... | plant | plant.n.01 |
| 3 | 496 | Fear of pollution discouraged people from buil... | plant | plant.n.01 |
| 4 | 497 | Plant dies without water. | plant | plant.n.02 |
| 5 | 498 | Take care of your plant in the garden. | plant | plant.n.02 |
| 6 | 499 | You have various plant flower. | plant | plant.n.02 |
| 7 | 500 | Plant grows only in the tropical regions. | plant | plant.n.02 |

In [10]:
```
# Selecting data with 'bark' as sense word
word = 'bark'
bark_df = df[df['polysemy_word'] == word]
bark_df = bark_df.reset_index(drop=True, inplace=False)
```

```
# Display the updated DataFrame
bark_df
```

Out[10]:

| | sn | sentence/context | polysemy_word | definition |
|---|---|---|---|---|
| 0 | 1931 | The bark of this tree is very rough. | bark | bark.n.01 |
| 1 | 1932 | Bears like to scratch their back on tree bark. | bark | bark.n.01 |
| 2 | 1933 | Bears often scratch their backs on the bark of... | bark | bark.n.01 |
| 3 | 1934 | They stripped the tree of its bark. | bark | bark.n.01 |
| 4 | 1935 | The collected bark of tree. | bark | bark.n.01 |
| 5 | 1936 | That dog is trained to bark at stranger. | bark | bark.n.02 |
| 6 | 1937 | The dog ‘s bark is against the intruder. | bark | bark.n.02 |
| 7 | 1938 | Dog that bark a lot usually aren t dangerous. | bark | bark.n.02 |
| 8 | 1939 | The bark of that dog wouldn t even scare off a... | bark | bark.n.02 |
| 9 | 1940 | Sometimes I hear dog ‘s bark in the middle of ... | bark | bark.n.02 |

In [11]:
```
# Selecting data with 'bass' as sense word
word = 'bass'
bass_df = df[df['polysemy_word'] == word]
bass_df = bass_df.reset_index(drop=True, inplace=False)

# Display the updated DataFrame
bass_df
```

Out[11]:

| | sn | sentence/context | polysemy_word | definition |
|---|---|---|---|---|
| 0 | 48 | He can play the bass. | bass | bass.n.01 |
| 1 | 49 | My bass string broke. | bass | bass.n.01 |
| 2 | 50 | It is all about the bass, no treble. | bass | bass.n.01 |
| 3 | 51 | Most bass guitar strings are made of nickel-wr... | bass | bass.n.01 |
| 4 | 52 | Some bass has Cobalt string. | bass | bass.n.01 |
| 5 | 53 | The bass of female voice is different than tha... | bass | bass.n.01 |
| 6 | 54 | I caught a bass in my net. | bass | freshwater_bass.n.01 |
| 7 | 55 | I quite like to eat bass. | bass | freshwater_bass.n.01 |
| 8 | 56 | The bass is very nutritious for the person suf... | bass | freshwater_bass.n.01 |
| 9 | 57 | Bass is a North American delicacy. | bass | freshwater_bass.n.01 |

In [12]:
```
# Selecting data with 'stable' as sense word
word = 'stable'
stable_df = df[df['polysemy_word'] == word]
stable_df = stable_df.reset_index(drop=True, inplace=False)

# Display the updated DataFrame
stable_df
```

Out[12]:

| | sn | sentence/context | polysemy_word | definition |
|---|---|---|---|---|
| 0 | 107 | He owns a horse stable. | stable | stable.n.01 |
| 1 | 108 | The horse is in the stable. | stable | stable.n.01 |
| 2 | 109 | A horse is tied in the stable. | stable | stable.n.01 |
| 3 | 110 | He is not stable after the new circumstance. | stable | stable.s.03 |
| 4 | 111 | The price in the market is stable for the last... | stable | stable.s.03 |
| 5 | 112 | Do not worry, the ladder is stable. | stable | stable.s.03 |
| 6 | 113 | Although there is fluctuation in economy of Ne... | stable | stable.s.03 |
| 7 | 114 | Some elements are stable in chemical reaction. | stable | stable.s.03 |
| 8 | 115 | Most of the intermediate product in chemical r... | stable | stable.s.03 |
| 9 | 116 | Some chemical compounds are very stable. | stable | stable.s.03 |

In [13]:
```
# Keywords
bank_savings_list = ['account', 'loan', 'amount', 'deposited', 'software', 'cash', 'money', 'balance', 'interes
bank_river_list = ['boat', 'river', 'sea', 'vacation']

plant_power_list = ['manufacture', 'laid', 'chemical', 'power']
plant_tree_list = ['water', 'garden', 'flower', 'grows']
```

```
bark_tree_list = ['tree', 'trees']
bark_dog_list = ['dog', 'dogs', "dog's"]

bass_music_list = ['play', 'string', 'treble', 'strings', 'voice']
bass_fish_list = ['caught', 'net', 'eat', 'nutritious', 'delicacy']

stable_horse_list = ['horse']
stable_condition_list = ['circumstance', 'price', 'worry', 'economy', 'chemical']
```

In [14]:
```
def get_best_sense(word, context):
    synsets = wn.synsets(word)
    tokens = nltk.word_tokenize(context)

    match word:
        case 'bank':
            for token in tokens:
                if token in bank_river_list:
                    selected_sense = synsets[0]
                    break
                elif token in bank_savings_list:
                    selected_sense = synsets[1]
                    break
                else:
                    selected_sense = None

        case 'plant':
            for token in tokens:
                if token in plant_power_list:
                    selected_sense = synsets[0]
                    break
                elif token in plant_tree_list:
                    selected_sense = synsets[1]
                    break
                else:
                    selected_sense = None

        case 'bark':
            for token in tokens:
                if token in bark_tree_list:
                    selected_sense = synsets[0]
                    break
                elif token in bark_dog_list:
                    selected_sense = synsets[1]
                    break
                else:
                    selected_sense = None

        case 'bass':
            for token in tokens:
                if token in bass_music_list:
                    selected_sense = synsets[0]
                    break
                elif token in bass_fish_list:
                    selected_sense = synsets[4]
                    break
                else:
                    selected_sense = None

        case 'stable':
            for token in tokens:
                if token in stable_horse_list:
                    selected_sense = synsets[0]
                    break
                elif token in stable_condition_list:
                    selected_sense = synsets[4]
                    break
                else:
                    selected_sense = None

        case default:
            pass

    return selected_sense
```

In [15]:
```
def choose_df(df):
    golden_pred = []

    for index, row in df.iterrows():
        best_sense = get_best_sense(row[2], row[1])
        # print(row[2])
        # print(best_sense)
```

```
        # Check
        if best_sense:
            print(f"Sentence: {row[1]}")
            print(f"Best sense of 'bank': {best_sense.name()}")
            print(f"Definition: {best_sense.definition()}")
            print()
            golden_pred.append(best_sense.name())
        else:
            print(f"Sentence: {row[1]}")
            print(f"No suitable sense found for '{row[2]}'.")
            print()
            golden_pred.append('None')
        print(golden_pred)
        print()

    return golden_pred
```

In [16]:
```
def lesk_algo(df):
    lesk_pred = []
    for index, row in df.iterrows():
        sentence = row[1]
        word_sense = row[2]
        tokens = nltk.word_tokenize(sentence)

        pos_tags = nltk.pos_tag(tokens)

        # n: explicitly disambiguate the noun word 'bank'
        sense = lesk(tokens, word_sense, 'n')

        # Check
        if sense:
            print(f"Sentence: {row[1]}")
            print(f"Best sense of {word_sense}: {sense}")
            print(f"Definition: {sense.definition()}")
            print()
            lesk_pred.append(sense.name())
        else:
            print(f"Sentence: {row[1]}")
            print(f"No sense found for {word_sense} using lesk.")
            lesk_pred.append('None')

    return lesk_pred
```

In [18]:
```
word_sense = input('Choose a word from the following (bank, plant, bark, bass, stable): ')
print(word_sense)
match word_sense:
    case 'bank':
        y_test = bank_df['definition'].tolist()
        y_lesk = lesk_algo(bank_df)
        y_gold = choose_df(bank_df)

    case 'plant':
        y_test = plant_df['definition'].tolist()
        y_lesk = lesk_algo(plant_df)
        y_gold = choose_df(plant_df)

    case 'bark':
        y_test = bark_df['definition'].tolist()
        y_lesk = lesk_algo(bark_df)
        y_gold = choose_df(bark_df)

    case 'bass':
        y_test = bass_df['definition'].tolist()
        y_lesk = lesk_algo(bass_df)
        y_gold = choose_df(bass_df)

    case 'stable':
        y_test = stable_df['definition'].tolist()
        y_lesk = lesk_algo(stable_df)
        y_gold = choose_df(stable_df)

    case default:
        print('Invalid word.')

print(f'''DATA:
y_test: {y_test}\n\ny_lesk: {y_lesk} \n\ny_gold: {y_gold}''')
```

```
bark
Sentence: The bark of this tree is very rough.
Best sense of bark: Synset('bark.n.02')
Definition: a noise resembling the bark of a dog

Sentence: Bears like to scratch their back on tree bark.
```

```
Best sense of bark: Synset('bark.n.02')
Definition: a noise resembling the bark of a dog

Sentence: Bears often scratch their backs on the bark of trees.
Best sense of bark: Synset('bark.n.02')
Definition: a noise resembling the bark of a dog

Sentence: They stripped the tree of its bark.
Best sense of bark: Synset('bark.n.02')
Definition: a noise resembling the bark of a dog

Sentence: The collected bark of tree.
Best sense of bark: Synset('bark.n.02')
Definition: a noise resembling the bark of a dog

Sentence: That dog is trained to bark at stranger.
Best sense of bark: Synset('bark.n.02')
Definition: a noise resembling the bark of a dog

Sentence: The dog ‘s bark is against the intruder.
Best sense of bark: Synset('bark.n.02')
Definition: a noise resembling the bark of a dog

Sentence: Dog that bark a lot usually aren t dangerous.
Best sense of bark: Synset('bark.n.02')
Definition: a noise resembling the bark of a dog

Sentence: The bark of that dog wouldn t even scare off a baby.
Best sense of bark: Synset('bark.n.02')
Definition: a noise resembling the bark of a dog

Sentence: Sometimes I hear dog ‘s bark in the middle of the night.
Best sense of bark: Synset('bark.n.02')
Definition: a noise resembling the bark of a dog

Sentence: The bark of this tree is very rough.
Best sense of 'bank': bark.n.01
Definition: tough protective covering of the woody stems and roots of trees and other woody plants

['bark.n.01']

Sentence: Bears like to scratch their back on tree bark.
Best sense of 'bank': bark.n.01
Definition: tough protective covering of the woody stems and roots of trees and other woody plants

['bark.n.01', 'bark.n.01']

Sentence: Bears often scratch their backs on the bark of trees.
Best sense of 'bank': bark.n.01
Definition: tough protective covering of the woody stems and roots of trees and other woody plants

['bark.n.01', 'bark.n.01', 'bark.n.01']

Sentence: They stripped the tree of its bark.
Best sense of 'bank': bark.n.01
Definition: tough protective covering of the woody stems and roots of trees and other woody plants

['bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01']

Sentence: The collected bark of tree.
Best sense of 'bank': bark.n.01
Definition: tough protective covering of the woody stems and roots of trees and other woody plants

['bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01']

Sentence: That dog is trained to bark at stranger.
Best sense of 'bank': bark.n.02
Definition: a noise resembling the bark of a dog

['bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.02']

Sentence: The dog ‘s bark is against the intruder.
Best sense of 'bank': bark.n.02
Definition: a noise resembling the bark of a dog

['bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.02', 'bark.n.02']

Sentence: Dog that bark a lot usually aren t dangerous.
No suitable sense found for 'bark'.

['bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.02', 'bark.n.02', 'None']

Sentence: The bark of that dog wouldn t even scare off a baby.
```

```
Best sense of 'bank': bark.n.02
Definition: a noise resembling the bark of a dog

['bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.02', 'bark.n.02', 'None', 'bark.n.02']

Sentence: Sometimes I hear dog ‘s bark in the middle of the night.
Best sense of 'bank': bark.n.02
Definition: a noise resembling the bark of a dog

['bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.02', 'bark.n.02', 'None', 'bark.n.02',
'bark.n.02']

DATA:
y_test: ['bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.02', 'bark.n.02', 'bark.n.02',
'bark.n.02', 'bark.n.02']

y_lesk: ['bark.n.02', 'bark.n.02', 'bark.n.02', 'bark.n.02', 'bark.n.02', 'bark.n.02', 'bark.n.02', 'bark.n.02',
'bark.n.02', 'bark.n.02']

y_gold: ['bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.01', 'bark.n.02', 'bark.n.02', 'None', 'bar
k.n.02', 'bark.n.02']
C:\Users\sheila brown\AppData\Local\Temp\ipykernel_13968\3966447291.py:4: FutureWarning: Series.__getitem__ trea
ting keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consis
tent with DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  sentence = row[1]
C:\Users\sheila brown\AppData\Local\Temp\ipykernel_13968\3966447291.py:5: FutureWarning: Series.__getitem__ trea
ting keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consis
tent with DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  word_sense = row[2]
C:\Users\sheila brown\AppData\Local\Temp\ipykernel_13968\3966447291.py:15: FutureWarning: Series.__getitem__ tre
ating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consi
stent with DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  print(f"Sentence: {row[1]}")
C:\Users\sheila brown\AppData\Local\Temp\ipykernel_13968\4256043952.py:11: FutureWarning: Series.__getitem__ tre
ating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consi
stent with DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  print(f"Sentence: {row[1]}")
C:\Users\sheila brown\AppData\Local\Temp\ipykernel_13968\4256043952.py:17: FutureWarning: Series.__getitem__ tre
ating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consi
stent with DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  print(f"Sentence: {row[1]}")
C:\Users\sheila brown\AppData\Local\Temp\ipykernel_13968\4256043952.py:18: FutureWarning: Series.__getitem__ tre
ating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consi
stent with DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  print(f"No suitable sense found for '{row[2]}'.")
```

# Evaluation Metrics

In [78]:
```python
# Lesk: Evaluation
accuracy = accuracy_score(y_test, y_lesk)
precision = precision_score(y_test, y_lesk, average='macro')
recall = recall_score(y_test, y_lesk, average='macro')
f1 = f1_score(y_test, y_lesk, average='macro')

print(f'''EVALUATION METRICS
Accuracy: {accuracy}
Precision: {precision}
Recall Score: {recall}
F1 Score: {f1}''')
```

```
EVALUATION METRICS
Accuracy: 0.3
Precision: 0.15
Recall Score: 0.5
F1 Score: 0.23076923076923078
D:\Programs\Python\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 contro
l this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [79]:
```python
# Gold Standard: Evaluation
accuracy = accuracy_score(y_test, y_gold)
precision = precision_score(y_test, y_gold, average='macro')
recall = recall_score(y_test, y_gold, average='macro')
f1 = f1_score(y_test, y_gold, average='macro')

print(f'''EVALUATION METRICS
Accuracy: {accuracy}
Precision: {precision}
Recall Score: {recall}
F1 Score: {f1}''')
```

```
EVALUATION METRICS
Accuracy: 1.0
Precision: 1.0
Recall Score: 1.0
F1 Score: 1.0
```

In [ ]: