In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelBinarizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize,sent_tokenize
from bs4 import BeautifulSoup
import re,string,unicodedata
from nltk.tokenize.toktok import ToktokTokenizer
from nltk.stem import LancasterStemmer,WordNetLemmatizer
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
import math
import os
import warnings
warnings.filterwarnings('ignore')
import gensim
from gensim.utils import simple_preprocess
import nltk
from nltk.corpus import stopwords
import pandas as pd
import re
from wordcloud import WordCloud
import gensim.corpora as corpora
import pyLDAvis.gensim_models
import pickle
import pyLDAvis
np.random.seed(0)
import pandas as pd
import re
from wordcloud import WordCloud
from pprint import pprint
```

3 Latent Dirichlet Allocation

We'll utilize the dataset of papers from the NeurIPS (NIPS) conference https://www.kaggle.com/datasets/ benhamner/nips-papers, one of the most esteemed annual gatherings in the machine learning field, for this assignment. The CSV data file includes details on the many NeurIPS articles that have been published between 1 Lab Course Machine Learning Exercise Sheet 11 – Prof. Dr. Dr. Lars Schmidt-Thieme 2/2 Kiran Madhusudhanan 1987 and 2016 (that's 29 years!). These articles cover a wide range of machine learning issues, including neural networks, optimization techniques, and many more. You are free to use sklearn/gensim/nltk or any library for this task. Perform topic modelling using LDA and present a comprehensive analysis of the task. You can use word clouds and present analysis using the LDA visualization library pyLDAvis. Points will be awarded based on the depth of results!

In [ ]:
```
# Load the data
data = pd.read_csv("archive/papers.csv")

# Select the columns of interest and take a sample of 100 rows
data = data[['paper_text']].sample(100)

# Preprocess the data
def preprocess_text(text):
    # Remove punctuation
    text = re.sub('[,\.!?]', '', text)
    # Convert text to lowercase
    text = text.lower()
    return text

data['paper_text_processed'] = data['paper_text'].apply(preprocess_text)
```

```
# Join the preprocessed text into a single string
text = ' '.join(data['paper_text_processed'].tolist())
```

In [ ]:
```
stop_words = stopwords.words('english')
# Function to convert sentences to words
def sent_to_words(sentences):
    for sentence in sentences:
        # Remove punctuations using simple_preprocess and deacc=True
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))
# Function to remove stop words
def remove_stopwords(texts):
    # Return a list of words that are not in stop_words list
    return [[word for word in simple_preprocess(str(doc))
             if word not in stop_words] for doc in texts]
# Convert data to list
data = data.paper_text_processed.values.tolist()
# Convert sentences to words using sent_to_words function
data_words = list(sent_to_words(data))
# Remove stop words from data_words
data_words = remove_stopwords(data_words)
```

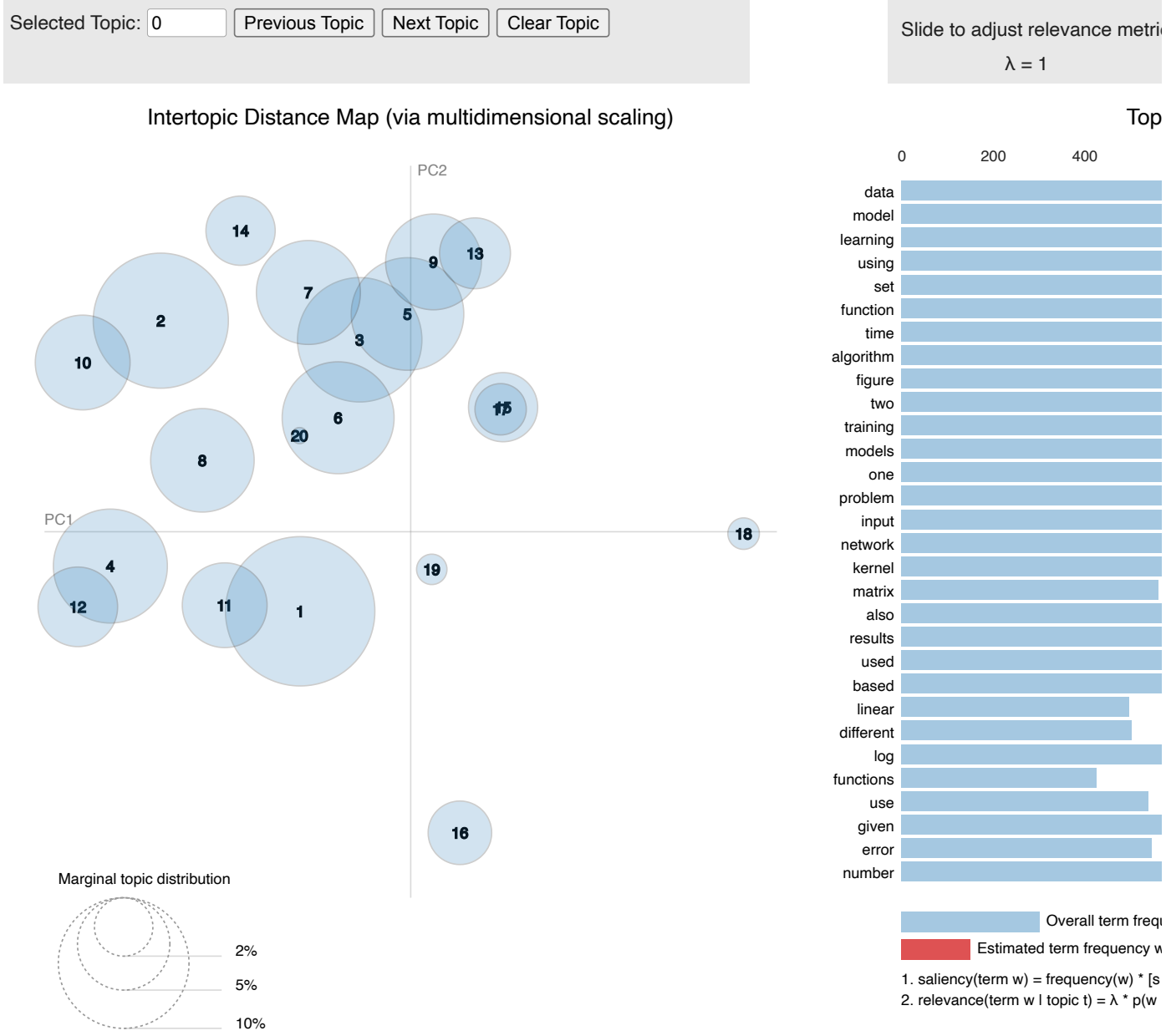
In [ ]:
```
# Number of topics to be generated
num_topics = 20
# Creating a dictionary from the data_words
id2word = corpora.Dictionary(data_words)
# Creating a corpus of documents, where each document is represented as a bag of words
corpus = [id2word.doc2bow(text) for text in data_words]
# Creating the LDA model
lda_model = gensim.models.LdaMulticore(corpus=corpus, id2word=id2word, num_topics=num_topics)
# Printing the top 5 topics and their keywords
pprint(lda_model.print_topics()[:5])
```

```
[(0,
  '0.006*"model" + 0.005*"function" + 0.004*"using" + 0.004*"learning" + '
  '0.004*"set" + 0.004*"data" + 0.003*"figure" + 0.003*"one" + 0.003*"input" + '
  '0.003*"network"'),
 (1,
  '0.006*"data" + 0.005*"model" + 0.005*"learning" + 0.005*"algorithm" + '
  '0.004*"function" + 0.004*"log" + 0.004*"time" + 0.004*"number" + '
  '0.004*"one" + 0.004*"models"'),
 (2,
  '0.007*"data" + 0.006*"model" + 0.006*"learning" + 0.005*"using" + '
  '0.004*"algorithm" + 0.004*"set" + 0.004*"models" + 0.003*"function" + '
  '0.003*"time" + 0.003*"problem"'),
 (3,
  '0.008*"learning" + 0.007*"model" + 0.005*"data" + 0.004*"log" + '
  '0.004*"algorithm" + 0.004*"one" + 0.004*"using" + 0.004*"set" + '
  '0.003*"number" + 0.003*"function"'),
 (4,
  '0.006*"data" + 0.006*"model" + 0.005*"learning" + 0.004*"algorithm" + '
  '0.004*"time" + 0.004*"using" + 0.004*"models" + 0.003*"matrix" + '
  '0.003*"one" + 0.003*"function"')]
```

HTML Not showing in the pdf, attaching it in the submission

In [ ]:
```
pyLDAvis.enable_notebook()
LDAvis = pyLDAvis.gensim_models.prepare(lda_model, corpus, id2word)
LDAvis
```

Out[ ]:


Selected Topic: 0
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metri
λ = 1
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
14
13
9
7
2
5
3
10
6
20
8
18
4
19
12
11
1
16
Marginal topic distribution
2%
5%
10%
Top
0
200
400
data
model
learning
using
set
function
time
algorithm
figure
two
training
models
one
problem
input
network
kernel
matrix
also
results
used
based
linear
different
log
functions
use
given
error
number
Overall term frequ
Estimated term frequency w
1. saliency(term w) = frequency(w) * [s
2. relevance(term w | topic t) = λ * p(w

(2)