Configuration Manual

MSc Research Project
Data Analytics

Sarath Kumar Samynathan
Student ID: x20185774

School of Computing
National College of Ireland

Supervisor: Giovani Estrada

National College of Ireland
Project Submission Sheet School of Computing

Student Name:	Sarath Kumar Samynathan
Student ID:	X20185774
Programme:	Data Analytics
Year:	2021-2022
Module:	MSc Research Project
Supervisor:	Giovani Estrada
Submission Due Date:	16/12/2021
Project Title:	Configuration Manual
Word Count:	982
Page Count:	7

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sarath Kumar Samynathan
Date:	16th December 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	Q
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	Q
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	Q

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sarath Kumar Samynathan

X20185774

MSc Research Project in Data Analytics
16th December 2021

1 Introduction

The software requirements, hardware requirements, and system setup would all be covered in this configuration manual. In addition, the following codes have been utilized for programming that has been created for the purpose of putting the research study into action:

”Improvised ICD-10 (International Classification of Diseases 10th Revision) Code Prediction using Machine Learning”

2 System Configuration

2.1 Hardware

Processor: AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz; RAM:8GB Storage: 512 GB SSD; Operating system: Windows 10, 64-bit.

2.2 Software

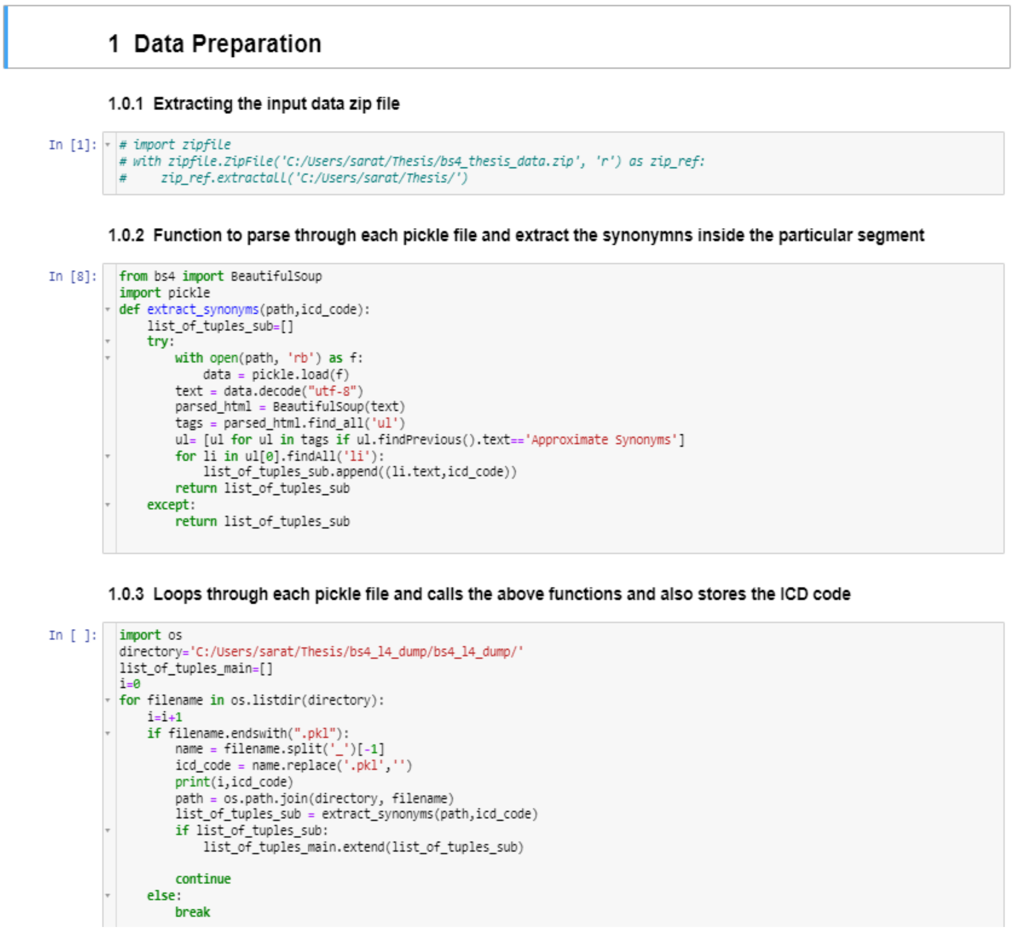
* Python, utilizing the Jupyter notebook: data analysis, data cleansing, pre-processing, and manipulation are all covered. Cross-validation, as well as the development of machine learning algorithms and word embeddings, were accomplished through the use of Python modules.
* Microsoft Excel: Used for saving of data, data exploration, and plots for explorations.

3 Project Development

The following are the steps involved in project development: data extraction from a pickle file, data preparation, model creation, model validation, word embedding setup, and cross validation to determine which model is the most effective.

3.1 Data Extraction

Our data originates from Kaggle, which is a freely available source of information. Many pieces of information are contained within HTML files that have been scraped from the icd10data.com website. Each HTML file is allocated to a specific ICD code category, which may be found here. This HTML file contains the whole set of ICD code detail information. In the html file, there is a section titled 'Synonym' where we will submit this synonyms data as input for the model to train based on the synonyms text to predict the ICD code, and this data will be stored in a database.



1 Data Preparation

1.0.1 Extracting the input data zip file

```
In [1]:
# import zipfile
# with zipfile.ZipFile('C:/Users/sarat/Thesis/bs4_thesis_data.zip', 'r') as zip_ref:
#     zip_ref.extractall('C:/Users/sarat/Thesis/')
```

1.0.2 Function to parse through each pickle file and extract the synonymns inside the particular segment

```
In [8]:
from bs4 import BeautifulSoup
import pickle
def extract_synonyms(path,icd_code):
    list_of_tuples_sub=[]
    try:
        with open(path, 'rb') as f:
            data = pickle.load(f)
        text = data.decode("utf-8")
        parsed_html = BeautifulSoup(text)
        tags = parsed_html.find_all('ul')
        ul= [ul for ul in tags if ul.findPrevious().text=='Approximate Synonyms']
        for li in ul[0].findAll('li'):
            list_of_tuples_sub.append((li.text,icd_code))
        return list_of_tuples_sub
    except:
        return list_of_tuples_sub
```

1.0.3 Loops through each pickle file and calls the above functions and also stores the ICD code

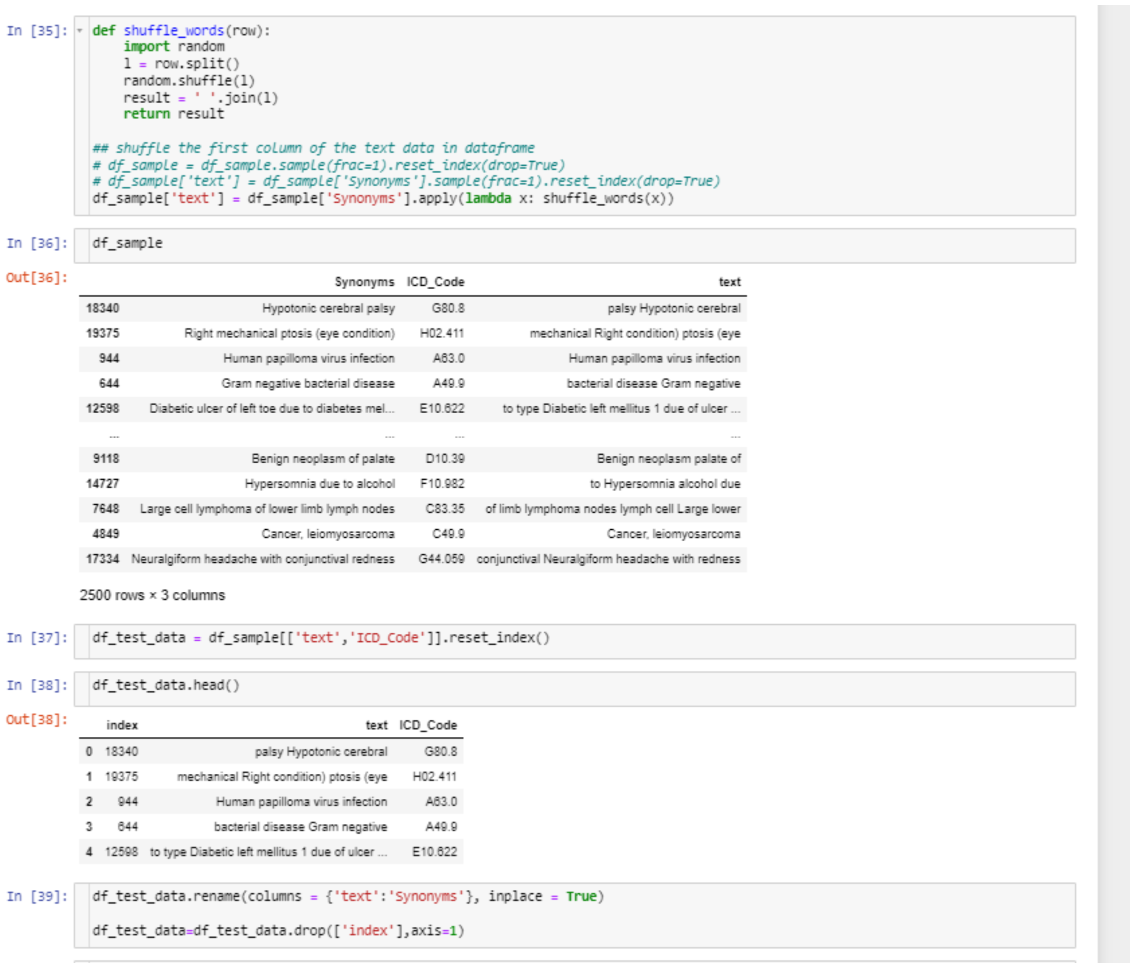
```
In [ ]:
import os
directory='C:/Users/sarat/Thesis/bs4_14_dump/bs4_14_dump/'
list_of_tuples_main=[]
i=0
for filename in os.listdir(directory):
    i=i+1
    if filename.endswith(".pkl"):
        name = filename.split('_')[-1]
        icd_code = name.replace('.pkl','')
        print(i,icd_code)
        path = os.path.join(directory, filename)
        list_of_tuples_sub = extract_synonyms(path,icd_code)
        if list_of_tuples_sub:
            list_of_tuples_main.extend(list_of_tuples_sub)

        continue
    else:
        break
```

Figure 1: Data Extraction

3.2 Data Preparation

For the sake of this stage, we'll divide the data into two sections: one for training and another for testing (Botta-Dukát, 2008) . In all, we have around 80,000 records. We will create our test data from the train data because we do not have enough entries for each class to use the train data as is currently. Selecting 20,000 records at random from train data, we'll switch the wording in the synonyms to make it difficult for the machine to predict the outcome.



```
In [35]: def shuffle_words(row):
             import random
             l = row.split()
             random.shuffle(l)
             result = ' '.join(l)
             return result

         ## shuffle the first column of the text data in dataframe
         # df_sample = df_sample.sample(frac=1).reset_index(drop=True)
         # df_sample['text'] = df_sample['Synonyms'].sample(frac=1).reset_index(drop=True)
         df_sample['text'] = df_sample['Synonyms'].apply(lambda x: shuffle_words(x))

In [36]: df_sample

Out[36]:
                                          Synonyms  ICD_Code                                           text
18340                        Hypotonic cerebral palsy     G80.8                       palsy Hypotonic cerebral
19375          Right mechanical ptosis (eye condition)   H02.411         mechanical Right condition) ptosis (eye
  944                    Human papilloma virus infection     A63.0                Human papilloma virus infection
  644                     Gram negative bacterial disease     A49.9                bacterial disease Gram negative
12598        Diabetic ulcer of left toe due to diabetes mel...   E10.622   to type Diabetic left mellitus 1 due of ulcer ...
  ...                                              ...       ...                                            ...
 9118                          Benign neoplasm of palate    D10.39                      Benign neoplasm palate of
14727                         Hypersomnia due to alcohol    F10.982                     to Hypersomnia alcohol due
 7648      Large cell lymphoma of lower limb lymph nodes    C83.35   of limb lymphoma nodes lymph cell Large lower
 4849                           Cancer, leiomyosarcoma     C49.9                        Cancer, leiomyosarcoma
17334    Neuralgiform headache with conjunctival redness   G44.059   conjunctival Neuralgiform headache with redness

2500 rows × 3 columns

In [37]: df_test_data = df_sample[['text','ICD_Code']].reset_index()

In [38]: df_test_data.head()

Out[38]:
   index                                          text  ICD_Code
0  18340                       palsy Hypotonic cerebral     G80.8
1  19375       mechanical Right condition) ptosis (eye   H02.411
2    944               Human papilloma virus infection     A63.0
3    644               bacterial disease Gram negative     A49.9
4  12598  to type Diabetic left mellitus 1 due of ulcer ...   E10.622

In [39]: df_test_data.rename(columns = {'text':'Synonyms'}, inplace = True)

         df_test_data=df_test_data.drop(['index'],axis=1)
```

Figure 2: Data Preparation

4 Code used for Machine Learning Models

This study has involved the implementation of six machine learning models and one deep learning model in total. The coding for this study was completed using the Jupyter Notebook programming language. Because the entire code was written in Python, it was only necessary to perform the correlation, cleaning, and model functions. It is intended that the following programs for explanation be used in the following ways: cross validation, imbalance of classes, word embeddings, and experiments on models.

4.2 Logistic Regression Classifier

```
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', LogisticRegression())])

model = pipe.fit(x_train, y_train)
prediction = model.predict(x_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

```
accuracy: 45.5%
```

4.3 Support Vector Classifier

```
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', LinearSVC())])

model = pipe.fit(x_train, y_train)
prediction = model.predict(x_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

```
accuracy: 85.12%
```

4.4 Multinomial Naive Bayes Classifier

```
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', MultinomialNB())])

model = pipe.fit(x_train, y_train)
prediction = model.predict(x_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

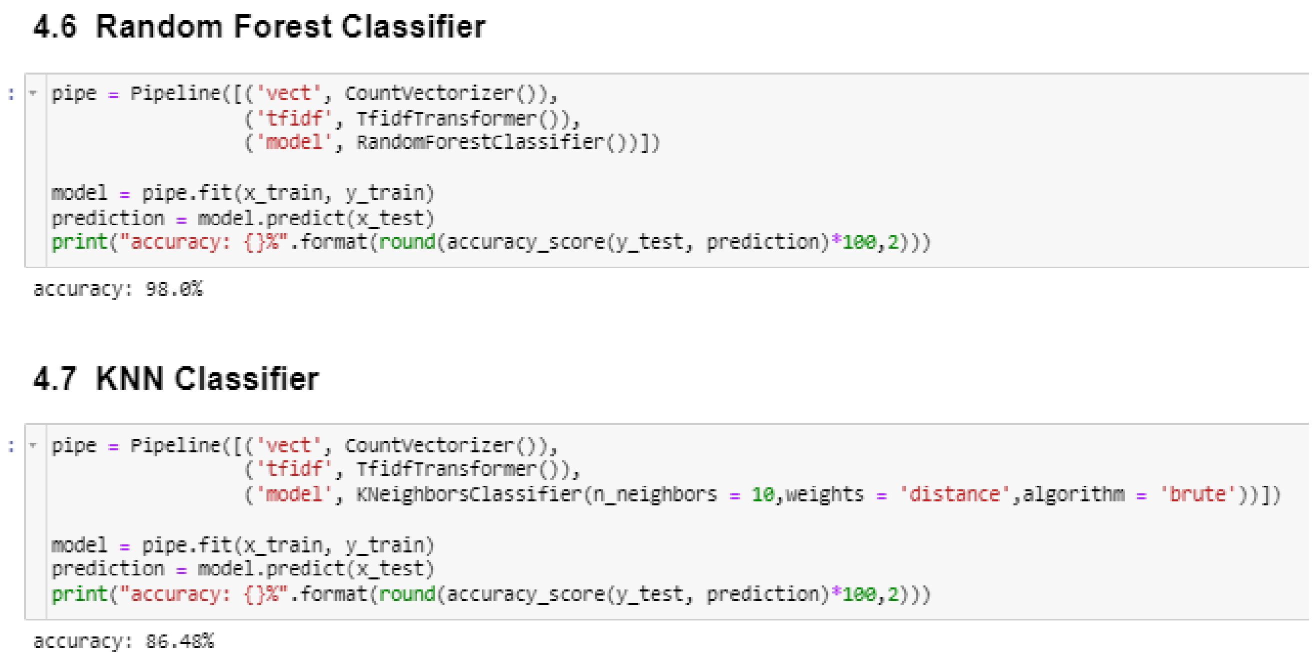
```
accuracy: 15.12%
```

4.5 Bernoulli Naive Bayes Classifier

```
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', BernoulliNB())])

model = pipe.fit(x_train, y_train)
prediction = model.predict(x_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

```
accuracy: 0.88%
```



4.6 Random Forest Classifier

```
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', RandomForestClassifier())])

model = pipe.fit(x_train, y_train)
prediction = model.predict(x_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

```
accuracy: 98.0%
```

4.7 KNN Classifier

```
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', KNeighborsClassifier(n_neighbors = 10,weights = 'distance',algorithm = 'brute'))])

model = pipe.fit(x_train, y_train)
prediction = model.predict(x_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

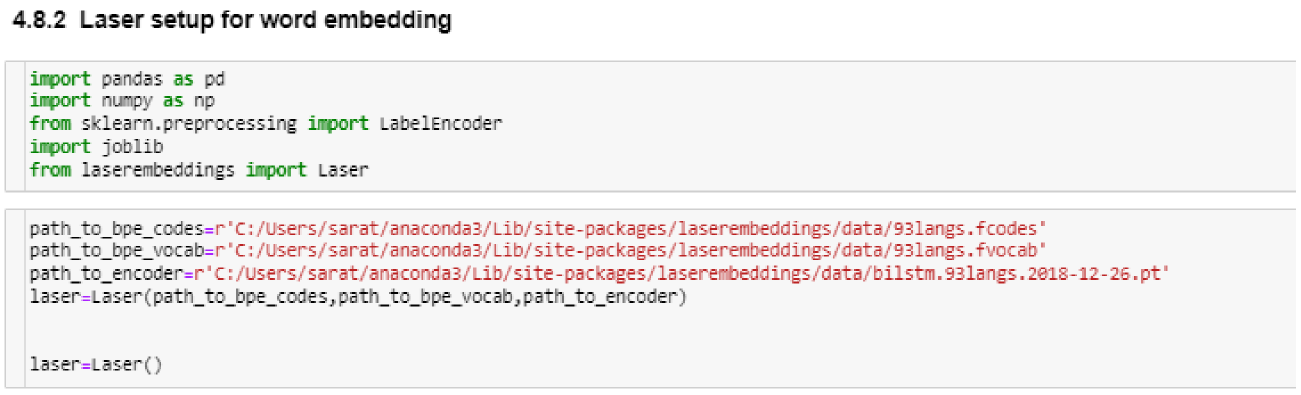
```
accuracy: 86.48%
```

Figure 3: Machine Learning Models

5 Deep Learning Model

We have implemented the ICD-10 classification (Maier, Philipp and Zaudig, 1990) with on the basic deep learning technique called MLP classifier. Before performing multilayer perceptron model, we will perform LASER embedding (Aluru, Mathew, Saha and Mukherjee, 2021) in our features which is considered as a language model published by Facebook known as Language-Agnostic Sentence Representation.

5.1 Laser setup:



4.8.2 Laser setup for word embedding

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
import joblib
from laserembeddings import Laser
```

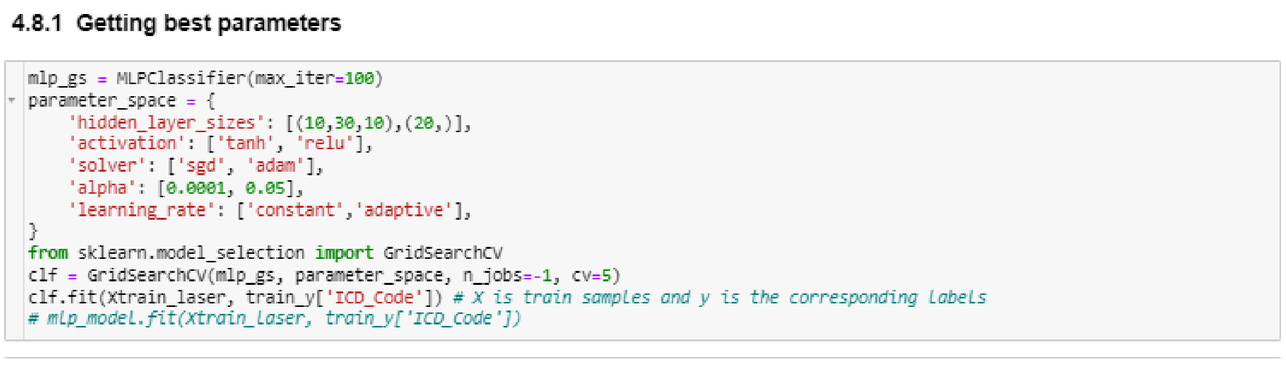
```
path_to_bpe_codes=r'C:/Users/sarat/anaconda3/Lib/site-packages/laserembeddings/data/93langs.fcodes'
path_to_bpe_vocab=r'C:/Users/sarat/anaconda3/Lib/site-packages/laserembeddings/data/93langs.fvocab'
path_to_encoder=r'C:/Users/sarat/anaconda3/Lib/site-packages/laserembeddings/data/bilstm.93langs.2018-12-26.pt'
laser=Laser(path_to_bpe_codes,path_to_bpe_vocab,path_to_encoder)


laser=Laser()
```

Figure 4: Laser setup for word embedding

5.2 Cross validation:

With the use of the gridsearchCV crossvalidation (Browne, 2021) approach, we were able to determine which parameters would work best for the mlp classifier.



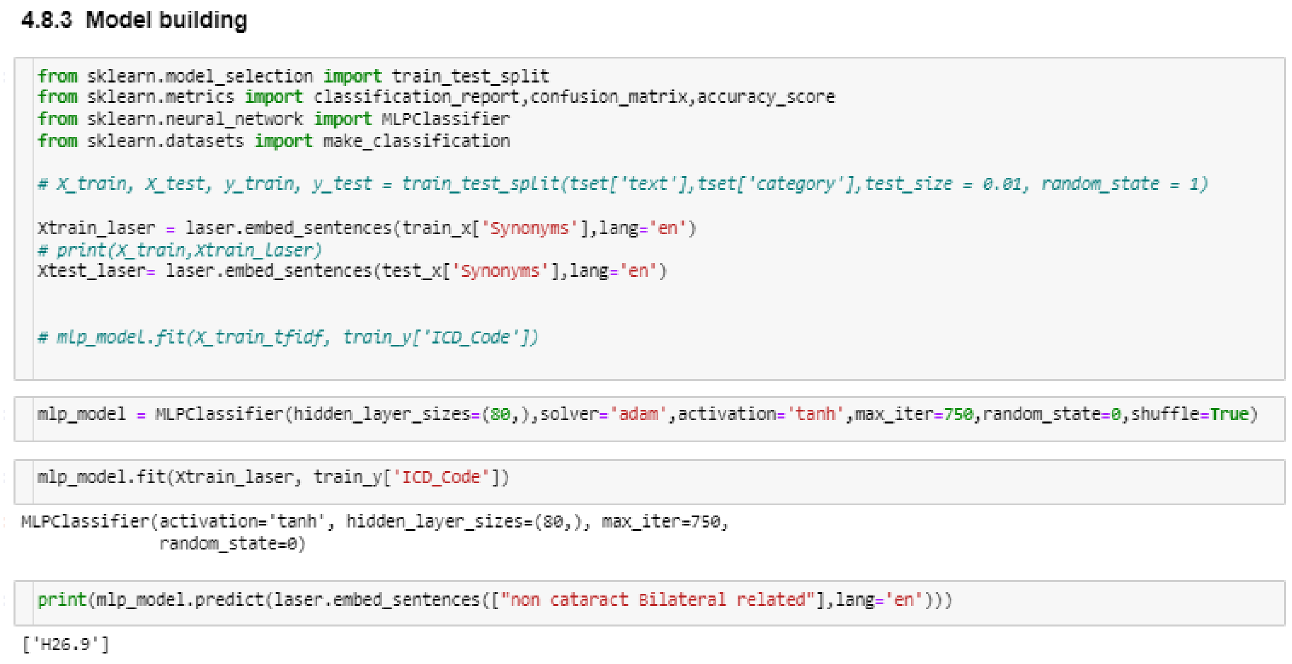
4.8.1 Getting best parameters

```
mlp_gs = MLPClassifier(max_iter=100)
parameter_space = {
    'hidden_layer_sizes': [(10,30,10),(20,)],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant','adaptive'],
}
from sklearn.model_selection import GridSearchCV
clf = GridSearchCV(mlp_gs, parameter_space, n_jobs=-1, cv=5)
clf.fit(Xtrain_laser, train_y['ICD_Code']) # X is train samples and y is the corresponding labels
# mlp_model.fit(Xtrain_laser, train_y['ICD_Code'])
```

Figure 5: GridSearch Cross Validation

5.3 MLP Classifier:

As a pickle file, we have stored our trained model (Patil and Yardi, 2012) based on the best parameters that we have found. As a result, every time we run the model, it will automatically choose the file and provide the forecasts.



4.8.3 Model building

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import make_classification

# X_train, X_test, y_train, y_test = train_test_split(tset['text'],tset['category'],test_size = 0.01, random_state = 1)

Xtrain_laser = laser.embed_sentences(train_x['Synonyms'],lang='en')
# print(X_train,Xtrain_laser)
Xtest_laser= laser.embed_sentences(test_x['Synonyms'],lang='en')


# mlp_model.fit(X_train_tfidf, train_y['ICD_Code'])
```

```
mlp_model = MLPClassifier(hidden_layer_sizes=(80,),solver='adam',activation='tanh',max_iter=750,random_state=0,shuffle=True)
```

```
mlp_model.fit(Xtrain_laser, train_y['ICD_Code'])
```

```
MLPClassifier(activation='tanh', hidden_layer_sizes=(80,), max_iter=750,
              random_state=0)
```

```
print(mlp_model.predict(laser.embed_sentences(["non cataract Bilateral related"],lang='en')))
```

```
['H26.9']
```

Figure 6: MLP Classifier

References:

Browne, M. (2000) "Cross-Validation Methods", Journal of Mathematical Psychology, 44(1), pp. 108-132. doi: 10.1006/jmps.1999.1279.

Aluru, S.S., Mathew, B., Saha, P. and Mukherjee, A., 2020. Deep learning models for multilingual hate speech detection. arXiv preprint arXiv:2004.06465.

Botta-Dukát, Z., 2008. Validation of hierarchical classifications by splitting dataset. Acta Botanica Hungarica, 50(1-2), pp.73-80.

Patil, M. and Yardi, A., 2012. MLP Classifier for Dementia Levels. International Journal of Modeling and Optimization, pp.567-569.

Maier, W., Philipp, M. and Zaudig, M., 1990. Comparison of the ICD-10-Classification with the ICD-9- and the DSM-III-Classification of Mental Disorders. Pharmacopsychiatry, 23(S 4), pp.183-187.