Classification on Unseen Data (total 8 points)

In this final task, you should read the feather file 'TestQuestionsDF.feather.zstd' into a pandas dataframe. Hereafter this will be referred to as the test_set.
You can assume that the test_set is a random sample from the same dataset as 'TrainQuestionsDF.feather.zstd' (hereafter train_set). Your goal is to classify the data in the test_set and achieve the best **average f1-score** using the train_set. You are allowed to utilize any technique and model available in the scikit-learn library or the standard python libraries to do so. Pay particular attention to the lessons learned from your experiments in the Classification notebook -- any of these approaches can be used to construct the model you use for prediction. You can additionally choose to generate and/or construct any features from the available data. Remember that the test_set should be represented with the same feature space as the train_set. For example, features based on text should be constructed with the same vocabulary on the test_set as the train_set.
To achieve a high f1 score on unseen data, remember to utilize all the techniques you've learned in the lectures, lectorials and practicals.

For this task, you are expected to submit the following:

1. This notebook with your code, the code should be well documented and must run without errors. There is no time limit, but it is a good practice to save the parameters of the best model and add an option to generate a model with those parameters. Without running the full tuning of the hyper-parameters.
2. Up to 4 prediction files, each predictions file will have exactly two columns: "Id" and "Label" with these headers and no other columns (e.g. index).
   The file names should be SXXXXXXX-A2-predictions-\<n>.csv - where n is a running integer {1,2,3,4}.

Your mark in this task will depend on the following:

1. The code is well documented, and the entire notebook runs without errors (1 points).
2. The submitted solutions are reproducible, i.e. the submitted code can generate the submitted prediction files (2 points).
3. The highest (out of the 4 prediction files) achieved average f1-score is in the following range:
   - (0.8, 1] (5 points)
   - (0.7, 0.8] (4 points)
   - (0.65, 0.7] (3 points)
   - (0, 0.65] (1 point)

To support the reproducibility of your solution, use the random seed anywhere where the solution involves a random process.

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```

```

```
import warnings
warnings.filterwarnings("ignore")


import io
import requests
import matplotlib.pyplot as plt
import sklearn.metrics as metrics
from sklearn.model_selection import cross_val_score, RepeatedStratifiedKFold, GridSe
from sklearn.ensemble import RandomForestClassifier
from sklearn import preprocessing
from sklearn import feature_selection as fs
from sklearn.tree import DecisionTreeClassifier
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.feature_selection import SelectKBest, f_classif, mutual_info_classif
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import PowerTransformer
from scipy import stats
```

In [3]:

```
# TODO: Set the random seed as your student id (only numbers)
RANDOM_SEED = 3955778
np.random.seed(RANDOM_SEED)
```

In [4]:

```
def read_feather_to_df(feather_file_name):
    """
    The function expects to receive a path to feather file,
    it will read the file from the disk into a pandas dataframe
    """
    return pd.read_feather(feather_file_name)
```

In [21]:

```
train_df = read_feather_to_df('TrainQuestionsDF.feather.zstd')
test_df = read_feather_to_df('TestQuestionsDF.feather.zstd')
test_df.head()
```

Out[21]:

	Id	PostTypeId	AcceptedAnswerId	CreationDate	Score	ViewCount	Body	OwnerUserId
0	2	1	59	2010-07-19 19:12:57.157	31	30036	<p>In many different statistical methods there...	24
1	30	1	55	2010-07-19 19:28:34.220	13	1620	<p>Which methods are used for testing random v...	69
2	298	1	-1	2010-07-20 13:11:50.297	161	312967	<p>Am I looking for a better behaved distribut...	125

	Id	PostTypeId	AcceptedAnswerId	CreationDate	Score	ViewCount	Body	OwnerUserId
3	870	1	956	2010-07-28 03:54:56.447	22	26750	<p>Given a list of p-values generated from ind...	520
4	881	1	1189	2010-07-28 08:15:51.733	5	919	<p>Here's something I've wondered about for a ...	34

In [17]:
```
def select_numeric_non_id_columns(df):

    Z = (df.select_dtypes(include=['int64', 'object'], exclude=["string", "datetime6

    return Z
```

In [10]:
```
train_df = select_numeric_non_id_columns(train_df)

Data = train_df.drop(columns = 'Label').values
target = train_df['Label']
```

I hope the use of a different algorithm might give me some luck so I'm going to use the probability based Naive Baise technique. Using my gridsearch function from before I look for the optimal hyperparameter (Var-Smoothing) and train the model.

In [11]:
```
cv_method = RepeatedStratifiedKFold(n_splits=5,
                                    n_repeats=3,
                                    random_state=RANDOM_SEED)


nb_classifier = GaussianNB()

params_NB = {'var_smoothing': np.logspace(0,-9, num=100)}

gs_NB = GridSearchCV(estimator=nb_classifier,
                     param_grid=params_NB,
                     cv=cv_method,
                     verbose=1,
                     scoring='accuracy')

Data_transformed = PowerTransformer().fit_transform(Data)

gs_NB.fit(Data_transformed, target);
```

```
Fitting 15 folds for each of 100 candidates, totalling 1500 fits
```

In [12]:
```
print(gs_NB.best_params_)
print(gs_NB.best_score_)
```

```
{'var_smoothing': 0.04328761281083057}
0.20925532781088466
```

In [22]:
```
pred_data = select_numeric_non_id_columns(test_df)

print(set(gs_NB.predict(pred_data)))

predictions_NB = gs_NB.predict(pred_data)
test_df['Label'] = predictions_NB
output_file = test_df[['Id', 'Label']]

#pred_NB = pd.DataFrame(predictions_NB, columns = ['ID','Label'])

predictions_NB
#I tried lol
```

Out[22]:
```
{'logistic', 'hypothesis-testing', 'distributions', 'bayesian', 'probability', 'self
-study', 'time-series'}
array(['time-series', 'time-series', 'time-series', ..., 'bayesian',
       'bayesian', 'hypothesis-testing'], dtype='<U18')
```

In [24]:
```
output_file.to_csv('S3955778-A2-predictions-1.csv', mode='a', index=False)
```