This note book will demonstrate training of two multi-label prediction models

In [3]:
```
def concate_columns(dataFrame, columns, new_col_name):
    outDataFrame = pd.DataFrame()
    outDataFrame[new_col_name]=dataFrame[columns[0]].map(lambda x: x.strip())
    for i in columns[1:]:
        outDataFrame[new_col_name]+=". "+dataFrame[i].map(lambda x: x.strip())
    return pd.concat([outDataFrame,dataFrame[list([x for x in dataFrame.columns if x not in columns])]],axis=1)

import numpy as np
import pandas as pd
pd.set_option('display.max_columns', 500)

outputColumns = ['Computer Science','Physics','Mathematics','Statistics','Quantitative Biology','Quantitative Finance']
csv = pd.read_csv('dataset/train.csv')
csv = concate_columns(csv, new_col_name = "Text", columns = ["TITLE","ABSTRACT"])

trainCount = int(len(csv)*0.7//1)
train = csv[:trainCount]
```

We load a dataset containting research paper Title and Abstract and the categories the paper belongs to. We concatenate the Title and Abstract into a single column of 'Text'

In [4]:
```
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS, TfidfVectorizer
from sklearn.pipeline import Pipeline
from xgboost import XGBClassifier
import xgboost as xgb
from joblib import dump
from sklearn.multioutput import MultiOutputClassifier

xgbPipeline = Pipeline(steps= [('tfidf', TfidfVectorizer(lowercase=True,
                                                         max_features=1000,
                                                         stop_words= ENGLISH_STOP_WORDS)),
                               ('model', MultiOutputClassifier(xgb.XGBClassifier(objective='binary:logistic', eval_metric="auc")))])

xgbPipeline.fit(train["Text"],train[outputColumns])
```

```
C:\Users\moham\anaconda3\envs\py2021-with-dill\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be re
moved in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your label
s (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

Out[4]:
```
Pipeline(steps=[('tfidf',
                 TfidfVectorizer(max_features=1000,
                                 stop_words=frozenset({'a', 'about', 'above',
                                                       'across', 'after',
                                                       'afterwards', 'again',
                                                       'against', 'all',
                                                       'almost', 'alone',
                                                       'along', 'already',
                                                       'also', 'although',
                                                       'always', 'am', 'among',
                                                       'amongst', 'amoungst',
                                                       'amount', 'an', 'and',
                                                       'another', 'any',
                                                       'anyhow', 'anyone',
                                                       'anything', 'anyway',
                                                       'anywhere', ...}))),
                ('mode...
                               importance_type='gain',
                               interaction_constraints=None,
                               learning_rate=None,
                               max_delta_step=None,
                               max_depth=None,
                               min_child_weight=None,
                               missing=nan,
                               monotone_constraints=None,
                               n_estimators=100,
                               n_jobs=None,
                               num_parallel_tree=None,
                               random_state=None,
                               reg_alpha=None,
                               reg_lambda=None,
                               scale_pos_weight=None,
                               subsample=None,
                               tree_method=None,
                               validate_parameters=None,
                               verbosity=None)))])
```

We build here a XGBoost model with a binary logistic objective. The model is trained with 70% of the data

In [5]:
```
from sklearn.multioutput import MultiOutputClassifier
from sklearn import ensemble
rfPipeLine = Pipeline(steps= [('tfidf', TfidfVectorizer(lowercase=True,
                                                        max_features=1000,
                                                        stop_words= ENGLISH_STOP_WORDS)),
                              ('model', ensemble.RandomForestClassifier(n_estimators=75, random_state=71))])

rfPipeLine.fit(train["Text"],train[outputColumns])
```

Out[5]:
```
Pipeline(steps=[('tfidf',
                 TfidfVectorizer(max_features=1000,
                                 stop_words=frozenset({'a', 'about', 'above',
                                                       'across', 'after',
                                                       'afterwards', 'again',
                                                       'against', 'all',
                                                       'almost', 'alone',
                                                       'along', 'already',
                                                       'also', 'although',
                                                       'always', 'am', 'among',
                                                       'amongst', 'amoungst',
                                                       'amount', 'an', 'and',
                                                       'another', 'any',
                                                       'anyhow', 'anyone',
                                                       'anything', 'anyway',
                                                       'anywhere', ...}))),
                ('model',
                 RandomForestClassifier(n_estimators=75, random_state=71))])
```

We build here a Random Forest model. We use a larger number of features in the TfidfVectorizer than in the previous case. This model now works on a differnet featurization and a different prediction algorithm than the first one.

In [6]:
```
dump(xgbPipeline, filename="xgbPipeline.joblib")
dump(rfPipeLine, filename="rfPipeLine.joblib")
```

Out[6]:
```
['rfPipeLine.joblib']
```

We dump both pipelines/models into seperate files so that they can be loaded into a model serving application.

We have thus created two *portable* models that can be loaded into a serving solution. By creating portable piplelines, the serving solution is decoupled from the data analysis and subsequent model training. This lends to easier continous deployment