```
from google.colab import drive
drive.mount('/content/drive')
```

```
cd drive/My Drive/Colab Notebooks/CF_Project
```

```
!pip3 install surprise
```

```
import time
import scipy.io as sio
import numpy as np
import random
import pandas as pd
import math
from math import sqrt
from pandas import DataFrame
from numpy import mean

from sklearn import metrics, preprocessing, dummy
from sklearn.model_selection import train_test_split, KFold
from sklearn.svm import SVC
from sklearn.preprocessing import Imputer, MinMaxScaler
from sklearn.ensemble import RandomForestClassifier as RF, AdaBoostClassifier as
from sklearn.cross_validation import cross_val_score as CV, StratifiedShuffleSpl
from sklearn.metrics import mean_squared_error, mean_absolute_error, accuracy_sc

from operator import itemgetter
from surprise import Reader
from surprise import SVD
from surprise import Dataset
from surprise.model_selection import cross_validate
from surprise import NormalPredictor
```

Read Labels

```
train1 = ['rating', 'user1', 'user2']
train_label = pd.read_csv('EH-training-labels.csv', sep=',', names=train1, encod
test1 = ['rating', 'user1', 'user2']
test_label = pd.read_csv('EH-testing-labels.csv', sep=',', names=test1, encoding
```

Read Features

```
train_features = pd.read_csv('EH-training-data.csv', sep=',', encoding='latin-1'
test_features = pd.read_csv('EH-testing-data.csv', sep=',', encoding='latin-1',
```

```
df = pd.DataFrame(train_label)

# A reader is still needed but only the rating_scale param is requiered.
reader = Reader(rating_scale=(0, 1))
train1 = ['user1', 'user2', 'rating']
df = df[train1]
# The columns must correspond to user id, item id and ratings (in that order).
data = Dataset.load_from_df(df[['user1', 'user2', 'rating']], reader)
```

The 3 baseline algorithms used as a part of th Surprise package

SVD Algorithm

```
from surprise import SVD
cross_validate(SVD(), data, cv=5)
```


```
{'fit_time': (39.474026918411255,
  40.0351836681366,
  39.750672817230225,
  39.60735297203064,
  39.92824125289917),
 'test_mae': array([0.21349853, 0.21373741, 0.21484803, 0.21470225, 0.2140
 'test_rmse': array([0.33564875, 0.33577263, 0.3380962 , 0.33712515, 0.336
 'test_time': (1.278069019317627,
  1.2538127899169922,
  1.2632677555084229,
  1.2543981075286865,
  0.9666924476623535)}
```


Negative Matrix Factorisation

```
from surprise import NMF
cross_validate(NMF(), data, cv=5)
```


```
{'fit_time': (59.71754860877991,
  59.99049210548401,
  60.202138900756836,
  60.95259618759155,
  60.08173108100891),
 'test_mae': array([0.17997403, 0.17940758, 0.18050502, 0.17939329, 0.1800
 'test_rmse': array([0.3412432 , 0.34054614, 0.3418656 , 0.33989704, 0.341
 'test_time': (1.19136643409729,
  0.8434107303619385,
  0.858839750289917,
  1.1541829109191895,
  1.1682100296020508)}
```


Normal predictor based on user item based method

```
cross_validate(NormalPredictor(), data, cv=5)
```

```
{'fit_time': (0.6336076259613037,
  0.8408927917480469,
  0.8551182746887207,
  0.8374414443969727,
  0.8400321006774902),
 'test_mae': array([0.27348941, 0.27599809, 0.27484957, 0.27556959, 0.2761
 'test_rmse': array([0.40785794, 0.40971539, 0.40896619, 0.40982819, 0.409
 'test_time': (1.2137830257415771,
  1.140387773513794,
  1.152226209640503,
  1.154350757598877,
  1.1366682052612305)}
```

DTI - Pred

Since the data was too huge the loading was taking too much time even on a TPU hence the data was reduced to half

```
s0 = [train_label.iloc[0,0]]
s0.extend(train_features.iloc[train_label.iloc[0,1] - 1, 1:].tolist())
s0.extend(train_features.iloc[train_label.iloc[0,2] - 1, 1:].tolist())
data = [s0]
train = pd.DataFrame(data)
ltrain = [0]
for j in range(1, 30000):
  t = 0
  while(t!=1):
    d = random.randint(1, len(train_label))
    if d not in ltrain:
      t = 1
      i = d
  s0 = [train_label.iloc[i,0]]
  s0.extend(train_features.iloc[train_label.iloc[i,1] - 1, 1:].tolist())
  s0.extend(train_features.iloc[train_label.iloc[i,2] - 1, 1:].tolist())
  data = [s0]
  train = train.append(data)
```

```
s0 = [test_label.iloc[0,0]]
s0.extend(test_features.iloc[test_label.iloc[0,1] - 1, 1:].tolist())
s0.extend(test_features.iloc[test_label.iloc[0,2] - 1, 1:].tolist())
data = [s0]
test = pd.DataFrame(data)
ltrain = [0]
for j in range(1, 30000):
  t = 0
  while(t!=1):
    d = random.randint(1, len(test_label))
    if d not in ltrain:
      t = 1
      i = d
  s0 = [test_label.iloc[i,0]]
  s0.extend(test_features.iloc[test_label.iloc[i,1] - 1, 1:].tolist())
  s0.extend(test_features.iloc[test_label.iloc[i,2] - 1, 1:].tolist())
  data = [s0]
  test = test.append(data)


start = time.time()


def classifier(train,test):
    print("Preprocessing data...")
    print ('Train set', train.shape)
    train_y = train.iloc[:,0]
    train = train.iloc[:,1:]

    print ('Full Test: ', test.shape)
    test = test.dropna()
    test_y = test.iloc[:,0]
    test = test.iloc[:,1:]

    ### Normalization and scaling steps ###
    scaler = MinMaxScaler(feature_range=(-1,1))

    train = preprocessing.normalize(train)
    test = preprocessing.normalize(test)

    ### Cross Validation parameters ###
    state = 'y' # 'y' to perform cross validation, anything else to skip
    n_folds = 5 # desired number of folds
    n_features = 116

    if state == 'y':
        ### Construct and fit classifiers ###
        forest = RF(n_estimators = 150, max_features = n_features, n_jobs = -1, 
        elapsed = time.time() - start
        print("\nTime elapsed: " + str(int(elapsed/60)) + " minutes.")
        print('\nCross-validating classification with ' + str(n_folds) + ' folds
        elapsed = time.time() - start
        print("\nTime elapsed: " + str(int(elapsed/60)) + " minutes.")
        scores = CV(forest, train, train_y, cv=n_folds, n_jobs=-1, scoring='roc_
        print("CV Scores: ")
        for i in scores:
            print (i)
        print('RF Cross-validation RMSE: %0.2f (+/- %0.2f)' % (scores.mean(), sc
        elapsed = time.time() - start
        print("\nTime elapsed: " + str(int(elapsed/60)) + " minutes.")
        print('RF CROSS-VALIDATION RESULTS (' + str(n_folds) + ' folds):\n')
        print('AUC: %0.2f (+/- %0.2f)\n' % (scores.mean(), scores.std() * 2))
        print("\nFitting Classifiers...")
```

```
        forest = forest.fit(train, train_y)
    else:
        ### Construct and fit classifiers ###
        forest = RF(n_estimators = 150, max_features = n_features, n_jobs = -1, r
        elapsed = time.time() - start
        print("\nTime elapsed: " + str(int(elapsed/60)) + " minutes.")
        print("Fitting Classifiers...")
        forest = forest.fit(train, train_y)
        print (forest.score(train, train_y))
        print('RF training score: %s\n\n' % (forest.score(train, train_y)))
        # dummy_clf = dummy.DummyClassifier(strategy='stratified')
        # dummy_clf = dummy_clf.fit(train, train_y)
        elapsed = time.time() - start
        print("\nTime elapsed: " + str(int(elapsed/60)) + " minutes.")

    ### Perform predictions on test data ###
    print('\nClassifying external validation set...')
    elapsed = time.time() - start
    print("\nTime elapsed: " + str(int(elapsed/60)) + " minutes.")

    rf_drugbank_pred = forest.predict(test)
    rms = sqrt(mean_squared_error(test_y, rf_drugbank_pred))
    print( "RMSE: ", rms)
    print("MAE: ",mae(test_y, rf_drugbank_pred))
    fpr, tpr, thresholds = metrics.roc_curve(test_y, rf_drugbank_pred, pos_label=
    cm = confusion_matrix(test_y, rf_drugbank_pred)
    auc = metrics.auc(fpr, tpr)

    print('\nRF DrugBank EXTERNAL VALIDATION RESULTS\nAUC: %s\n' % auc)
    print(cm)


def mae(ground_truth, prediction):
    d = float(mean(ground_truth)*mean(prediction))
    return (float(mean_absolute_error(ground_truth, prediction)))
```

```
classifier(train, test)
```

```
Preprocessing data...
Train set (30000, 117)
Full Test:  (30000, 117)

Time elapsed: 0 minutes.

Cross-validating classification with 5 folds...

Time elapsed: 0 minutes.
CV Scores:
0.6410888811926428
0.6263587250837221
0.6446280075565923
0.6411244317328963
0.6428140380546234
RF Cross-validation RMSE: 0.64 (+/- 0.01)

Time elapsed: 28 minutes.
RF CROSS-VALIDATION RESULTS (5 folds):

AUC: 0.64 (+/- 0.01)


Fitting Classifiers...

Classifying external validation set...

Time elapsed: 36 minutes.
RMSE:  0.34813790371058423
MAE:  0.1212

RF DrugBank EXTERNAL VALIDATION RESULTS
AUC: 0.5000730697226489

[[26361    18]
 [ 3618     3]]
```