```
!pip install -U -q PyDrive
import os
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

# 1. Authenticate and create the PyDrive client.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

# choose a local (colab) directory to store the data.
local_download_path = os.path.expanduser('~/data')
try:
  os.makedirs(local_download_path)
except: pass

# 2. Auto-iterate using the query syntax
#    https://developers.google.com/drive/v2/web/search-parameters
file_list = drive.ListFile(
    {'q': "'1imx6kKXQA1S3my_bnNlixt_XfWd3kJDi' in parents"}).GetList()

for f in file_list:
  # 3. Create & download by id.
  print('title: %s, id: %s' % (f['title'], f['id']))
  fname = os.path.join(local_download_path, f['title'])
  print('downloading to {}'.format(fname))
  f_ = drive.CreateFile({'id': f['id']})
  f_.GetContentFile(fname)
```

```
title: X_test_picked.pckl, id: 1vfG5TfcIXOWvK9dZCxgibVotnW9-YSTp
downloading to /content/data/X_test_picked.pckl
title: X_train_picked.pckl, id: 1SaUIV5ered822WzND17ozXHgfrZrLVin
downloading to /content/data/X_train_picked.pckl
title: y_train_picked.pckl, id: 17oKssXRY4NC2p8DAko5Me0N_IJRVhbzd
downloading to /content/data/y_train_picked.pckl
title: y_test_picked.pckl, id: 1bsXNqaZusrHnXVlUWsFNpyzwcSMV-PMK
downloading to /content/data/y_test_picked.pckl
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import mutual_info_classif
from scipy.stats import pearsonr
from sklearn import svm
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
import pickle
from sklearn.naive_bayes import GaussianNB
```

```
path_dataset_save = '/content/data/'

file = open(path_dataset_save+'X_train_picked.pckl','rb')
```

```
X_train_picked = pickle.load(file); file.close()
file = open(path_dataset_save+'X_test_picked.pckl','rb')
X_test_picked = pickle.load(file); file.close()
file = open(path_dataset_save+'y_train_picked.pckl','rb')
y_train_picked = pickle.load(file); file.close()
file = open(path_dataset_save+'y_test_picked.pckl','rb')
y_test_picked = pickle.load(file); file.close()



A=pd.DataFrame(X_train_picked)
B=pd.DataFrame(y_train_picked)
C=pd.DataFrame(X_test_picked)
D=pd.DataFrame(y_test_picked)


A.to_csv('X_train_picked.csv')
B.to_csv('Y_train_picked.csv')
C.to_csv('X_test_picked.csv')
D.to_csv('Y_test_picked.csv')



from google.colab import files

files.download('X_train_picked.csv')
#files.download('Y_train_picked.csv')
#files.download('X_test_picked.csv')
#files.download('Y_test_picked.csv')


df1 = pd.DataFrame(X_train_picked)

Y=pd.DataFrame(y_train_picked)
Y.columns=['f1']


#--------------------------------CHI-SQUARED-------------------------------------------

X=df1;
#Using all the samples for
c=SelectKBest(chi2, k=400)
c.fit_transform(X,Y)
names1 = X.columns.values[c.get_support()]


names1
```

```
array([ 67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  93,  94,
        95,  96,  97,  98,  99, 100, 101, 102, 103, 104, 105, 119, 120,
       121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 134,
       135, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156,
       157, 158, 159, 160, 161, 162, 163, 164, 165, 173, 174, 175, 176,
       177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189,
       190, 191, 192, 193, 200, 201, 202, 203, 204, 205, 206, 207, 208,
       209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221,
       222, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238,
       239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 255, 256,
       257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269,
       270, 271, 272, 273, 274, 275, 276, 277, 283, 284, 285, 286, 287,
       288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300,
       301, 302, 303, 304, 311, 312, 313, 314, 315, 316, 317, 318, 319,
       320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332,
       339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351,
       352, 353, 354, 355, 356, 357, 358, 359, 360, 369, 370, 371, 372,
       373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385,
       386, 387, 388, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406,
       407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 424, 425, 426,
       427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439,
       440, 441, 442, 443, 444, 452, 453, 454, 455, 456, 457, 458, 459,
       460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472,
```

```
#tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],'C': [1, 10, 100,1000]}]
#clf = GridSearchCV( LassoLarsCV(), parameters, cv=6, n_jobs=4, verbose=1)
#clf1 = GridSearchCV(GaussianNB(),  cv=3)
clf1=GaussianNB()
clf1.fit(X[names1],Y)
#clf1.fit(df1[names1],Y)
#clf1.fit(X,Y)
#clf.grid_scores_
#print (clf1.grid_scores_)
#model = clf1.best_estimator_
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:578: DataConversionl
  y = column_or_1d(y, warn=True)
GaussianNB(priors=None)
```

```
array([721, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745,
```

```
X1=pd.DataFrame(X_test_picked)
Y1=pd.DataFrame(y_test_picked)

y_pred = clf1.predict(X1[names1])


accuracy_score(Y1, y_pred)
```

```
0.6746
```

```
#---------------------------------------MI-----------------------------------------
c2=SelectKBest(mutual_info_classif, k=400)
c2.fit_transform(X,Y)
names2 = X.columns.values[c2.get_support()]
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:578: DataConversionl
  y = column_or_1d(y, warn=True)
```

```
clf2=GaussianNB()
clf2.fit(X[names2],Y)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:578: DataConversionW
  y = column_or_1d(y, warn=True)
GaussianNB(priors=None)
```

```
names2
```

```
array([  4,  38,  39,  43,  68,  69,  70,  71,  72,  73,  74,  75,  76,
        94,  95,  96,  97,  98,  99, 100, 101, 102, 103, 104, 105, 118,
       119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131,
       132, 133, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 173, 174, 175, 176,
       177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189,
       190, 191, 192, 193, 200, 201, 202, 203, 204, 205, 206, 207, 208,
       209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221,
       227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239,
       240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 256, 257, 258,
       259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271,
       272, 273, 274, 275, 276, 277, 280, 282, 283, 284, 285, 286, 287,
       288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300,
       301, 302, 303, 304, 312, 313, 314, 315, 316, 317, 318, 319, 320,
       321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 340,
       341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353,
       354, 355, 356, 357, 358, 359, 360, 368, 369, 370, 371, 372, 373,
       374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386,
       387, 388, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406,
       407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 425, 426, 427,
       428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440,
       441, 442, 443, 444, 447, 452, 453, 454, 455, 456, 457, 458, 459,
       460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472,
       473, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489,
       490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 507,
       508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520,
       521, 522, 523, 524, 525, 526, 527, 528, 529, 531, 534, 535, 536,
       537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549,
       550, 551, 552, 553, 554, 555, 556, 557, 563, 564, 565, 566, 567,
       568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580,
       581, 582, 583, 584, 585, 591, 592, 593, 594, 595, 596, 597, 598,
       599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611,
       612, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630,
       631, 632, 633, 634, 635, 636, 637, 639, 649, 650, 651, 652, 653,
       654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 667,
       678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690,
       691, 692, 693, 695, 696, 701, 707, 708, 709, 710, 711, 712, 713,
       714, 715, 716, 717, 718, 719, 720, 735, 736, 737, 738, 739, 740,
       741, 742, 743, 744, 745, 746])
```

```
X1=pd.DataFrame(X_test_picked)
Y1=pd.DataFrame(y_test_picked)
y_pred = clf2.predict(X1[names2])
```

```
Y1.shape
```

```
(5000, 1)
```

```
accuracy_score(Y1, y_pred)
```

```
0.677
```

```
result = X.join(Y,how='outer')
result
X3=result.loc[1:10001,abs(result.corr()['f1']>0.01)]
X3=X3.drop(['f1'],axis=1)
len(X3.columns)
```

```
290
```

```
clf3=GaussianNB()
clf3.fit(X3,Y[1:10000])
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:578: DataConversionl
  y = column_or_1d(y, warn=True)
GaussianNB(priors=None)
```

```
X1=pd.DataFrame(X_test_picked)
Y1=pd.DataFrame(y_test_picked)
y_pred = clf3.predict(X1[X3.columns])
```

```
accuracy_score(Y1, y_pred)
```

```
0.4916
```