

```
In [1]: import pandas as pd
import numpy as np
from sklearn.decomposition import TruncatedSVD
import seaborn as sns
import random
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split

In [2]: !pip install -U -g PyDrive
```

```
import os
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
```

```
[3]: auth.authenticate_user()
      gauth = GoogleAuth()
      gauth.credentials = GoogleCredentials.get_application_default()
      drive = GoogleDrive(gauth)
      fileDownloaded = drive.CreateFile({'id':'https://drive.google.com/file/d/1uTHL59_tid1
      file_list = drive.ListFile({'q': '"1ATVDSyJ_YhpzSmYhV5X-q21CYzujQ2As' in parents and
      for file1 in file_list:
          print('title: %s, id: %s' % (file1['title'], file1['id']))
          data_downloaded_user_likes = drive.CreateFile({'id': '1uTHL59_tid1Tu409_T2zHZYIYu0D6b
          data_downloaded_user_likes.GetContentFile('users-likes.csv')
          data_downloaded_users = drive.CreateFile({'id': '1XRppHA9JDYGwIgio1Y5XpzXT9uwmJiU1'})
          data_downloaded_users.GetContentFile('users.csv')
          data_downloaded_likes = drive.CreateFile({'id': '1cHKETaDrTQBFC6MCDcFMP_wu8AgrwK4D'})
          data_downloaded_likes.GetContentFile('likes.csv')

      users=pd.read_csv('users.csv')
      likes=pd.read_csv('likes.csv')
      ul=pd.read_csv('users-likes.csv',error_bad_lines=False,engine='python')

      title: likes.csv, id: 1cHKETaDrTQBFC6MCDcFMP_wu8AgrwK4D
      title: users.csv, id: 1XRppHA9JDYGwIgio1Y5XpzXT9uwmJiU1
      title: users-likes.csv, id: 1uTHL59_tid1Tu409_T2zHZYIYu0D6bit

In [4]: #users=pd.read_csv('./sample_dataset/users.csv')
         #likes=pd.read_csv('./sample_dataset/likes.csv')
```

```
#ul=pd.read_csv('./sample_dataset/users-likes.csv')
users.head()
```

Out[4]:

		userid	gender	age	political	ope	con	ext	agr	neu
0	54f34605aebd63f7680e37ffd299af79	0	33	0.0	1.26	1.65	1.17	-1.76	0.61	

1	86399f8c44ba54224b2e60177ca89fa9	1	35	0.0	1.07	0.17	-0.14	1.49	0.30
2	84fab50f3c60d1fdc83aa91b5e584a78	1	36	0.0	0.89	1.28	0.86	1.07	0.99
3	f3b8fdaccce12ef6352bfad4d6052fe9	0	39	NaN	0.33	-1.01	-0.33	-0.68	0.92
4	8b06ea5e9cb87c61da387995450607f7	0	31	NaN	0.15	0.47	1.17	-1.01	-0.32

```
In [5]: ul.head()
```

```
Out[5]:
```

	userid	likeid
0	71bc7c0901488aec6d30f0add257e7c5	3c1636c878e6b2acfd0c06b108638
1	978ab8e90c4d6ad1a48ef5c973b62f4d	feca46ddb8ef04f86172ace0cb7e004c

2	85123b0e358907725cf19a2cb0ec3983	b65f46d64c688fe98bdbcf93a76a71fc
3	ce110562b3e2f7e5cad3775b32d9caa5	b65f46d64c688fe98bdbcf93a76a71fc
4	8188d20745471273fa69ba44a5b28473	b65f46d64c688fe98bdbcf93a76a71fc

```
In [6]: likes.head()
```

	likeid	name
0	3c1636c878e6eb2acf00c6b1086e38	REIGN by Paul Gibson
1	feca46ddb8ef04f86172ace0cb7e004c	Cupcake Wishes & Birthday Dreams

2	b65f466d4c688fe98dbcf93a76a71fc	Yo también me rei de la caída de otro jejeje
3	9c5c8bb82d2cd46fbd7582f944fe370e	Abraham Joshua Heschel Day School- Alumni Network
4	2d82fa84ad79b085dc516dde154327a2	Kennesaw Farmer's Market

```
In [7]: df=ul.iloc[:50957]
```

```
In [8]: sparse_matrix=df.groupby(['userid', 'likeid']).size().unstack(fill_value=0)

In [9]: sparse_matrix.head()

Out[9]:
```

userid		
0002abedba29d5a7fc1d34834b8846d7	0	0
00035a29fa913610d9dfd1c6d6a15fd6	0	0
000769fb960a5900187f6631c4bb7264	0	0

0007ef89a1a6d9440c3863bad4202f21	0	0
00082a96ca78b2883a3e24h9e8823567	0	0

5 rows × 7062 columns

```
n [10]: user_ids=sparse_matrix.index.values.tolist()

n [11]: sparse_matrix_n=np.array(sparse_matrix)

n [12]: #Trimming
sparse_matrix_n1=sparse_matrix_n[:,np.sum(sparse_matrix_n, axis=0)>1]
```

```
print(sparse_matrix_n1.shape)
sparse_matrix_n2=sparse_matrix_n1[np.sum(sparse_matrix_n, axis=1)>1]
print(sparse_matrix_n2.shape)

(25257, 2342)
(9992, 2342)
```

```
In [13]: #print(user_ids.shape)
          user_ids=np.array(user_ids)
          user_ids=user_ids[np.sum(sparse_matrix_n, axis=1)>1]

In [14]: sparse_matrix_n2

Out[14]: array([[0, 0, 0, ..., 0, 0, 0],
```

```

[0, 0, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 0, 0, 0],
...
[0, 0, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 0, 0, 0]])

random.seed(60)

```

```
[15]: random.seed(68)

n [16]: n_components=5
svd = TruncatedSVD(n_components=5)
X_reduced = svd.fit_transform(sparse_matrix_n2)
df_svd = pd.DataFrame(data=X_reduced, index=[i for i in range(len(user_ids))], columns=[
df_svd['userid']=user_ids
```

```
df_svd.head()
```

	svd_1	svd_2	svd_3	svd_4	svd_5	userid
0	1.576169	0.202196	0.460868	-0.335003	-0.079080	00035a29fa913610d9dfdc6d6a15fd6
1	0.834194	-0.528804	-0.372666	0.003381	-0.114872	00082a96ca78b2883a3e24b9e8823567

```

2 0.056317 -0.017725 0.082691 0.780414 -0.204151 00217f065b47f79902cb8b57b897608
3 0.005412 0.004482 0.005724 0.006332 0.010378 0026109987824baee6d0251ff52f039e
4 0.004538 0.000962 -0.004915 0.007960 -0.006291 002cff3e5a5e1e3a4874d3768dd5e6be

```

[illegible]

c6a9a43058c8cc8398ca6e97324c0fae	0	47	NaN	-0.31	-0.57	-0.89	0.41	1.17	0.000000	-0.000000
172e5d8611cb33a8b466a29705b1b1bda	0	28	NaN	0.79	1.06	-0.89	-1.01	-0.01	0.003406	0.002000
f9ed42fd1c0e0e1ecd2ba3fdb54ce6fa	1	29	0.0	-0.31	-0.94	-0.77	-1.76	1.05	0.084353	-0.009000
eca69bfad8f42b193b2592248101b7f1	1	28	0.0	-0.68	0.54	-0.52	-1.01	-0.51	0.835180	-0.528000

```

c045fd404e1dfdd0dfcd9c30f690f84      1  30      0.0  1.53  1.65 -0.14  1.41 -0.38  0.004506  0.002
n [18]: combined_table.corr()
out[18]:
```

gender	1.000000	0.006989	-0.026141	-0.021108	-0.000021	-0.013182	0.033783	0.225765	-0.062500	-0.000000
age	0.006989	1.000000	-0.025313	0.073102	0.164837	0.043922	0.091851	-0.074951	-0.163858	0.000000
political	-0.026141	-0.025313	1.000000	-0.419679	0.142928	0.026795	0.036373	-0.088177	0.048995	0.000000
ope	-0.021108	0.073102	-0.419679	1.000000	0.089158	0.236131	0.120279	-0.115254	-0.047349	0.000000
acc	0.000021	0.164837	0.142928	0.089158	1.000000	0.250246	0.250923	0.204704	0.026546	0.000000

con	-0.000021	0.164837	0.142928	0.089158	1.000000	0.239248	0.238072	-0.394704	-0.026848	-0.000000
ext	-0.013182	0.043922	0.026795	0.236131	0.259246	1.000000	0.250768	-0.444976	0.061324	-0.000000
agr	0.033783	0.091851	0.036373	0.120279	0.258072	0.250768	1.000000	-0.435083	0.004109	-0.000000
neu	0.225765	-0.074951	-0.088177	-0.115254	-0.394704	-0.444976	-0.435083	1.000000	-0.029401	-0.000000
svd_1	-0.062500	-0.163858	0.048995	-0.047349	-0.026646	0.061324	0.004109	-0.029401	1.000000	-0.000000

svd_2	-0.095983	0.048733	-0.034834	0.027041	-0.010614	0.031871	-0.037772	-0.012849	-0.058391	1
svd_3	0.089164	-0.090183	0.037325	-0.031276	-0.037400	-0.041914	-0.041975	0.058966	-0.090121	-0
svd_4	0.178241	-0.001163	-0.018295	-0.025144	0.025299	-0.010348	0.001294	0.098115	-0.289917	-0
svd_5	0.034980	-0.158644	0.056226	-0.081085	-0.072351	-0.006450	-0.049853	0.049106	-0.043586	-0

```

n [19]: plt.figure(figsize=(16, 6))
heatmap = sns.heatmap(combined_table.corr(), vmin=-1, vmax=1, annot=True)
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':18}, pad=12);

# save heatmap as .png file
# dpi - sets the resolution of the saved image in dots/inches
# bbox_inches when set to 'tight' does not allow the labels to be cropped

```

```
# bbox_inches = when set to 'tight' does not allow the labels to be cropped
plt.savefig('heatmap.png', dpi=300, bbox_inches='tight')
```

	gender	age	political
gender	1	0.007	0.026
age	0.007	1	-0.025
political	0.026	-0.025	1

	0.021	0.028	0.032	0.034	0.036	0.038	0.040	0.042	0.044	0.046	0.048	0.050	0.052	0.054	0.056	0.058	0.060	0.062	0.064	0.066	0.068	0.070	0.072	0.074	0.076	0.078	0.080	0.082	0.084	0.086	0.088	0.090	0.092	0.094	0.096	0.098	0.100	0.102	0.104	0.106	0.108	0.110	0.112	0.114	0.116	0.118	0.120	0.122	0.124	0.126	0.128	0.130	0.132	0.134	0.136	0.138	0.140	0.142	0.144	0.146	0.148	0.150	0.152	0.154	0.156	0.158	0.160	0.162	0.164	0.166	0.168	0.170	0.172	0.174	0.176	0.178	0.180	0.182	0.184	0.186	0.188	0.190	0.192	0.194	0.196	0.198	0.200	0.202	0.204	0.206	0.208	0.210	0.212	0.214	0.216	0.218	0.220	0.222	0.224	0.226	0.228	0.230	0.232	0.234	0.236	0.238	0.240	0.242	0.244	0.246	0.248	0.250	0.252	0.254	0.256	0.258	0.260	0.262	0.264	0.266	0.268	0.270	0.272	0.274	0.276	0.278	0.280	0.282	0.284	0.286	0.288	0.290	0.292	0.294	0.296	0.298	0.300	0.302	0.304	0.306	0.308	0.310	0.312	0.314	0.316	0.318	0.320	0.322	0.324	0.326	0.328	0.330	0.332	0.334	0.336	0.338	0.340	0.342	0.344	0.346	0.348	0.350	0.352	0.354	0.356	0.358	0.360	0.362	0.364	0.366	0.368	0.370	0.372	0.374	0.376	0.378	0.380	0.382	0.384	0.386	0.388	0.390	0.392	0.394	0.396	0.398	0.400	0.402	0.404	0.406	0.408	0.410	0.412	0.414	0.416	0.418	0.420	0.422	0.424	0.426	0.428	0.430	0.432	0.434	0.436	0.438	0.440	0.442	0.444	0.446	0.448	0.450	0.452	0.454	0.456	0.458	0.460	0.462	0.464	0.466	0.468	0.470	0.472	0.474	0.476	0.478	0.480	0.482	0.484	0.486	0.488	0.490	0.492	0.494	0.496	0.498	0.500	0.502	0.504	0.506	0.508	0.510	0.512	0.514	0.516	0.518	0.520	0.522	0.524	0.526	0.528	0.530	0.532	0.534	0.536	0.538	0.540	0.542	0.544	0.546	0.548	0.550	0.552	0.554	0.556	0.558	0.560	0.562	0.564	0.566	0.568	0.570	0.572	0.574	0.576	0.578	0.580	0.582	0.584	0.586	0.588	0.590	0.592	0.594	0.596	0.598	0.600	0.602	0.604	0.606	0.608	0.610	0.612	0.614	0.616	0.618	0.620	0.622	0.624	0.626	0.628	0.630	0.632	0.634	0.636	0.638	0.640	0.642	0.644	0.646	0.648	0.650	0.652	0.654	0.656	0.658	0.660	0.662	0.664	0.666	0.668	0.670	0.672	0.674	0.676	0.678	0.680	0.682	0.684	0.686	0.688	0.690	0.692	0.694	0.696	0.698	0.700	0.702	0.704	0.706	0.708	0.710	0.712	0.714	0.716	0.718	0.720	0.722	0.724	0.726	0.728	0.730	0.732	0.734	0.736	0.738	0.740	0.742	0.744	0.746	0.748	0.750	0.752	0.754	0.756	0.758	0.760	0.762	0.764	0.766	0.768	0.770	0.772	0.774	0.776	0.778	0.780	0.782	0.784	0.786	0.788	0.790	0.792	0.794	0.796	0.798	0.800	0.802	0.804	0.806	0.808	0.810	0.812	0.814	0.816	0.818	0.820	0.822	0.824	0.826	0.828	0.830	0.832	0.834	0.836	0.838	0.840	0.842
--	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

	gender	age	political	ope	con	ext	agr	neu	svd_1	svd_2	svd_3	svd_4	svd_5
svd_1	-0.062	-0.16	0.049	-0.047	-0.027	0.061	0.0041	-0.029	1	-0.058	-0.09	-0.29	0.044
svd_2	-0.096	0.049	-0.035	-0.027	-0.011	0.032	-0.038	-0.013	-0.058	1	-0.066	-0.021	-0.0032
svd_3	0.089	-0.09	0.037	-0.031	-0.037	-0.042	-0.042	0.059	-0.09	-0.0066	1	-0.033	-0.0049
svd_4	0.18	-0.0012	-0.018	-0.025	0.025	-0.01	0.0013	0.098	-0.29	-0.021	-0.033	1	-0.016
svd_5	0.035	-0.16	0.056	-0.081	-0.072	-0.0065	-0.05	0.049	-0.044	-0.0032	-0.049	-0.016	1

```
n [20]: #X_reduced[0]
print(svd.explained_variance_ratio_)
print(svd.explained_variance_ratio_.sum())
print(svd.singular_values_)

[0.06633252 0.05456201 0.03567127 0.02201882 0.02138327]
```

```
0.19996789606195053
[60.20281345 40.76576925 33.05804358 27.16310336 25.4964577 ]

n [42]: n_components_svd=50
response_columns=8
result=np.zeros((n_components_svd,response_columns))
features = ['Gender','Age','Political','ope','con','ext','agr','neu']
for s in range(n_components_svd):
```

```
for n_c in range(1, n_components_svd+1):
    #print('model learning in progress for components', n_c)
    svd = TruncatedSVD(n_components=n_c)
    X_reduced = svd.fit_transform(sparse_matrix_n2)
    df_svd = pd.DataFrame(data=X_reduced, index=[i for i in range(len(user_ids))], columns=[f'userid_{n_c}' for n_c in range(1, n_components_svd+1)])
    df_svd['userid'] = user_ids
    left = users.set_index('userid')
    right = df_svd.set_index('userid')
```

```
right = df_svd.set_index('user_id')
combined_table=left.join(right, how='inner')
#for all y response variable except political one
c_t=np.array(combined_table)
X,y=c_t[:,response_columns:],c_t[:,response_columns]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_
#for political response variable
c_t=np.array(combined_table['political'])
```

```
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1, test_size=0.33, r
for y_col in range(response_columns):
    #for categorical columns
    if y_col in [0,2]:
        if y_col==2:
            clf = LogisticRegression() fit(X_train1, y_train1)
```

```

clf = LogisticRegression().fit(X_train, y_train)
auc=roc_auc_score(y_test1, clf.predict_proba(X_test1)[:, 1])
result[n_c-1,y_col]=auc
if(n_c==50):
    print('no of component',n_c,'Political response variable auc',auc)
else:
    clf = LogisticRegression().fit(X_train, y_train[:,y_col])
    auc=roc_auc_score(y_test1, clf.predict_proba(X_test1)[:, 1])

```

```

auc = roc_auc_score(y_test[:, y_col], clf.predict_proba(X_test)[:, 1])
result[n_c-1, y_col] = auc
if (n_c==50):
    print('no of component', n_c, 'Gender response variable auc', auc)

else:
    reg = LinearRegression().fit(X_train, y_train[:, y_col])
    y_gender_pred = reg.predict(X_test)

```

```

y_pred=pred.predict(X_test)
r=np.corrcoef(y_test[:,y_col],y_pred)[0][1]
result[n_c-1,y_col]=r
if(n_c==50):
    print('Number of component:',n_c,' y_col:',titles[y_col],'Accuracy(co
no of component 50 Gender response variable auc 0.7130780795947227

```

```

Number of component: 50 y_col: Age Accuracy(correlation) 0.3841129040422152
no of component 50 Political response variable auc 0.585892264305858
Number of component: 50 y_col: ope Accuracy(correlation) 0.14891999258026333
Number of component: 50 y_col: con Accuracy(correlation) 0.1383674364366912
Number of component: 50 y_col: ext Accuracy(correlation) 0.08209152468220343
Number of component: 50 y_col: agr Accuracy(correlation) 0.09505173136741513
Number of component: 50 y_col: neu Accuracy(correlation) 0.133291114524524114

```

```

n [43]: titles = ['Gender', 'Age', 'Political', 'ope', 'con', 'ext', 'agr', 'neu'] #title
fig, axs = plt.subplots(2,4,figsize=(10, 10))
fig.suptitle('AUC, Accuracy(Correlation) vs Number of components used', fontsize=16)
for kk, (ax,yy) in enumerate(zip(axs.reshape(-1),zip(*result))):
    #print(yy)
    ax.plot([i+1 for i in range(50)],yy)

```

```
ax.set_title(titles[kk])
ax.set_xlabel('number of components')
if kk in [0,2]:
    ax.set_ylabel('AUC')
else:
    ax.set_ylabel('Accuracy(correlation)')
#fig.delaxes(axes[1][1])
```

```
plt.show()
plt.savefig('result.png', dpi=300, bbox_inches='tight')
```

Figure 1 consists of four subplots, (a) through (d), each showing the accuracy of a different model as a function of the number of iterations. The x-axis for all plots represents the number of iterations, ranging from 0 to 35. The y-axis represents accuracy, with scales varying between plots: (a) Lasso ranges from 0.02 to 0.06; (b) Ridge ranges from 0.050 to 0.065; (c) Elastic Net ranges from 0.00 to 0.02; and (d) Adaptive Elastic Net ranges from 0.04 to 0.08. In all cases, the accuracy increases rapidly in the first few iterations and then stabilizes. The Adaptive Elastic Net (d) achieves the highest accuracy, peaking at approximately 0.085, while the Lasso (a) and Ridge (b) models reach around 0.065, and the Elastic Net (c) reaches around 0.02.

<Figure size 432x288 with 0 Axes>

