IT 350

Assignment 1 - Dimensionality Reduction

Name: Niraj Nandish

Roll no.: 191IT234

```
import sys
from math import sqrt

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.datasets import make_classification
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

pd.set_option('display.max_rows', 30)
```

1. Download a dataset of your choice

Dataset link - https://www.kaggle.com/iamsouravbanerjee/nifty50-stocks-df

NOTE - Rename the `csv` file to `stock.csv`

```
df = pd.read_csv('stock.csv')
df = df.select_dtypes(exclude=['object'])
print(df)
```

```
         Open      High       Low  ...    52w L  365 d % chng  30 d % chng
0      750.0    766.00    713.25  ...   384.40         79.22        -4.65
1     3101.0   3167.35   3091.00  ...  2117.15         45.66         5.66
2      669.0    674.90    660.45  ...   568.40         10.19       -21.49
3     3370.0   3383.50   3320.00  ...  3041.00          9.30       -12.05
4    17200.0  17237.20  16610.00  ...  8273.70         91.38        -9.10
..       ...       ...       ...  ...      ...           ...          ...
45    1544.0   1550.00   1510.15  ...   846.70         76.17        -2.83
46    2377.8   2385.10   2285.05  ...  1300.35         75.45        -6.59
47    7550.0   7599.00   7370.10  ...  4770.00         53.50         1.78
48     726.0    726.00    701.00  ...   414.15         68.06        -1.37
49     632.0    634.40    619.65  ...   346.25         77.51        -7.01

[50 rows x 12 columns]
```

2. Visualize it using multiple dimensions and say why SVD and PCA should be used here

The following df has 12 features with 50 rows. Each of the features are different ways to depict the values of change for different stocks. We need to use PCA and SVD for dimensionality reduction so that we can avoid overfitting when the dataset is trained for a model. There is also chances for the data to be redundant. Using these reduction techniques leads to lesser computational cost for the model and helps in removal of noise thereby improving the overall performance of the model.

3a. Implement PCA logic on your own and find the appropriate k-dimensions to represent this data

```
scalar = StandardScaler()
flag = 1

for c in df.columns:
    std = sqrt(((df[c] - df[c].mean())**2).sum() / df.shape[0])
    if std == 0:
        print("Invalid input")
    df[c] = (df[c] - df[c].mean())/std

if df.shape[0] >= df.shape[1] and flag == 1:
    df.fillna(0, inplace=True)
    means = []
    for c in df.columns:
        test = df[c].mean()
        means.append(test)
    covariance = []
    for j in range(df.shape[1]):
        cov = []
        for k in range(df.shape[1]):
            sum = 0
            for i in range(df.shape[0]):
                sum += (df.iloc[i,j] - means[j]) * (df.iloc[i,k] - means[k])
            result = sum / (df.shape[0]-1)
            cov.append(result)
        covariance.append(cov)
    eigen_values, eigen_vectors = np.linalg.eig(covariance)
    idx = eigen_values.argsort()[::-1]
    eigen_values = eigen_values[idx]
    eigen_vectors = eigen_vectors[:,idx]
    pc = df.dot(eigen_vectors)
    print("Principle Components using step by step approach: \n")
    pc = np.array(pc)
```

```
print(pc)
pca = PCA()
df_scaled = pd.DataFrame(scalar.fit_transform(df), columns=df.columns)
result = pca.fit_transform(df_scaled)
print("\nPrinciple Components using built in function: \n")
print(result)
```

```
 Principle Components using step by step approach:

[[ 1.22698749e+00 -7.38534155e-01  4.18614594e-01 -1.18079313e+00
   8.70022165e-02 -5.51851932e-02 -5.13284678e-01  2.42828351e-02
  -3.46332439e-02  2.50701999e-03  6.72567811e-04  3.67833608e-04]
 [-2.45921564e-01  1.76609070e+00 -1.01005894e+00 -1.06167594e+00
   2.55267077e-01 -1.89555412e-02 -6.26816325e-02 -2.29498377e-02
  -1.97349787e-02 -6.19441932e-03  2.83446687e-03 -1.98066045e-03]
 [ 1.25934456e+00 -6.10770946e-01  1.20333605e+00  2.00762957e+00
  -5.78136824e-01  5.82907277e-01  4.82582999e-01 -1.37518358e-03
   3.47269604e-02 -1.11085450e-03  5.48040787e-04  9.08052422e-05]
 [-5.01686409e-01  8.77736553e-01  1.12956653e+00  7.48553675e-01
  -4.85694150e-01  4.29639406e-01  3.57885596e-01  9.96231793e-02
  -2.38691638e-02 -5.09197261e-03 -1.29907090e-03  3.25072570e-04]
 [-6.97206821e+00 -2.57358177e+00  2.59056324e-01 -1.07302983e+00
   7.92340915e-01  7.56347455e-01  1.10034895e+00 -6.68654108e-01
  -1.58866319e-02 -7.93419962e-04 -2.40420630e-03 -1.75528040e-04]
 [-2.07557691e+00 -1.83241386e+00  6.50997756e-01  6.76906871e-01
   1.12251212e+00  6.78965740e-01 -1.41598419e-03 -3.58860273e-02
   2.29657035e-02  1.14757727e-03  4.12070746e-03 -6.23332428e-04]
 [ 1.18692995e+00  2.03570870e-01 -1.16276052e+00 -1.12746903e+00
   7.35807508e-01 -8.34708817e-01 -5.20203787e-01 -4.08244273e-02
  -3.55412671e-02 -3.41473782e-03 -1.04374894e-03  5.79377501e-04]
 [ 1.28929706e+00 -1.50630861e-01  1.47669830e+00  9.19156578e-01
  -2.44873259e-01 -7.18599754e-01 -2.00282634e-01 -1.58437096e-01
  -2.27047057e-02  1.52855575e-03 -5.67252912e-05 -2.44557533e-05]
 [-6.23580665e-01  1.86079126e+00  1.59024933e-01  3.86504805e-01
  -2.28641242e-01 -3.72470387e-02  1.54311803e-01  5.30209218e-02
   1.53222752e-02  9.64970802e-05  2.02930439e-03  2.13226694e-03]
 [ 1.07804968e+00  2.08711771e+00 -3.93783588e+00  9.28142606e-01
   6.79544775e-01  4.14285680e-01  1.09731914e+00  1.39737449e-01
   5.31026447e-02  4.29603385e-04 -2.16113294e-03 -1.27069641e-03]
 [ 1.38773394e+00  4.79300968e-01  5.45055072e-01  5.47789615e-01
  -7.56021875e-01 -8.58702394e-02  7.98671013e-01  5.03815202e-03
   4.29519092e-02  7.79816955e-04  5.51486335e-04  1.68415114e-04]
 [-6.93344004e-01  1.65267262e+00 -1.82591096e+00  4.43382402e-01
  -4.45225723e-01  1.18507694e+00 -1.67930429e-01 -1.29639602e-01
  -5.30964657e-02  2.15974968e-02  3.39533919e-03  7.65941383e-04]
 [-9.10291506e-01  2.74203477e+00 -1.62255370e+00  8.02593195e-01
  -4.18615968e-01  3.57461935e-01 -1.98591995e-01 -2.85354532e-02
  -8.64769995e-02  1.37069732e-02 -1.56632577e-03 -7.99352883e-04]
 [-1.53776971e-01  1.18136718e+00  1.10422685e+00  3.34264016e-01
   1.61050500e-01 -4.48691613e-01 -2.65804120e-02  1.00991902e-03
  -7.81051492e-03  6.31389179e-04 -3.14933016e-04 -1.17665934e-03]
 [ 5.58826259e-01 -7.69244599e-02  5.14773712e-01 -1.98652971e+00
  -1.10474978e-01  5.66750762e-01 -5.87726069e-02  1.06692997e-01
   1.51607511e-02  1.83823744e-03 -1.30806179e-03  3.05419210e-04]
```

```
[ 7.23132393e-01  1.17162348e+00  2.09555332e-01 -1.79868604e-01
 -8.22073517e-02  3.04948099e-01  3.18716096e-01  5.74578158e-02
  6.84338511e-03 -3.73052598e-03 -1.08793195e-03  3.11367598e-04]
[-1.35162581e-02 -2.38332358e-01  2.21227667e-01  3.31642724e-01
  8.97497886e-01 -8.34806506e-02 -4.90746298e-01  2.58925713e-03
  2.73720802e-02  2.93111054e-03  2.69183213e-03 -1.60841298e-03]
[ 8.39979313e-01 -3.68220625e-01 -4.38199024e-01  1.72854797e+00
  7.92524929e-01  1.03163399e-01 -3.38958829e-01 -1.84408889e-02
 -1.28413118e-03 -8.50869122e-03  5.96063245e-04 -6.34576739e-04]
[ 8.31938712e-01  1.48766900e+00  7.38983625e-01  4.50900812e-02
```

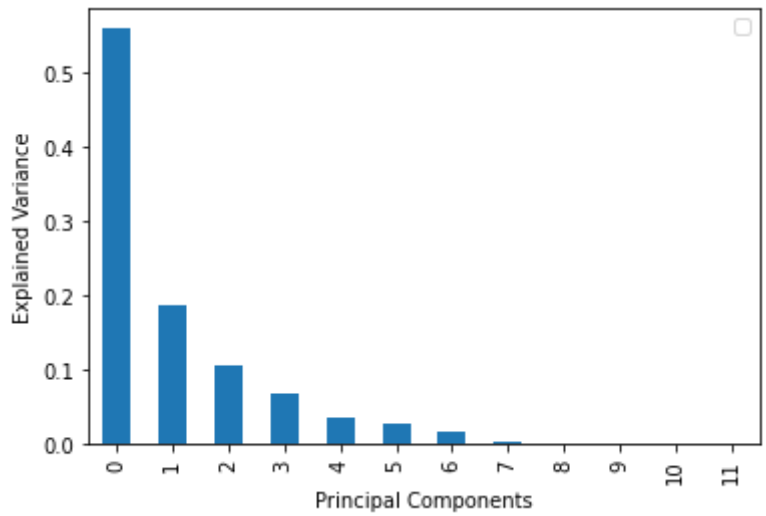
```
pca.explained_variance_ratio_
```

```
array([5.60901870e-01, 1.87045436e-01, 1.05629426e-01, 6.66227232e-02,
       3.47641348e-02, 2.76054048e-02, 1.56149964e-02, 1.71180554e-03,
       1.01241588e-04, 2.68416795e-06, 2.19702211e-07, 5.79315132e-08])
```

Below is the plot of how much variance each principle component captures. Since the first 5 features captures about 95 percent of the information, I selected the first 5 features to appropriately represent the data.

```
pd.DataFrame(pca.explained_variance_ratio_).plot.bar()
plt.legend('')
plt.xlabel('Principal Components')
plt.ylabel('Explained Variance')
```

```
Text(0, 0.5, 'Explained Variance')
```



4a. Visualize the data after applying PCA

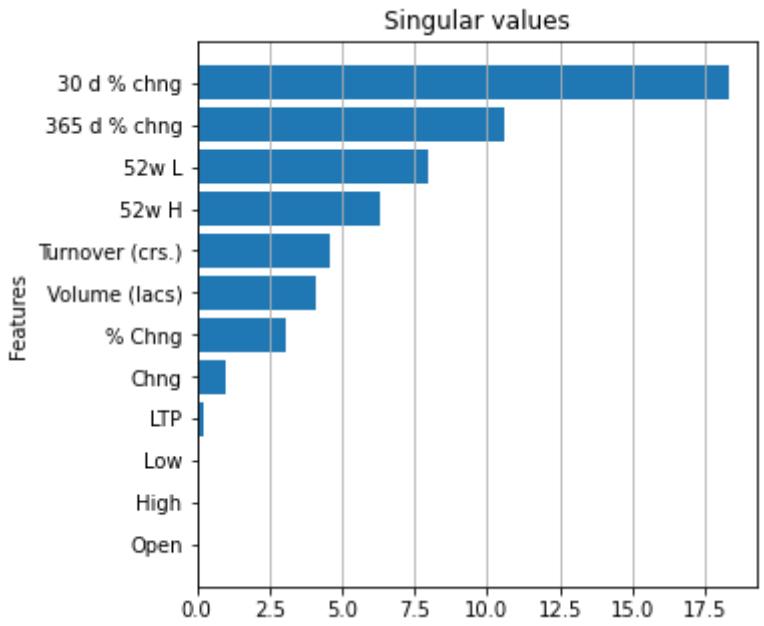
After principal component analysis it was decided to replace the original 12 features of dataset with the new 5 features by retaining 95 percent of the information.

3b. Implement SVD logic on your own and find the appropriate k-dimensions to represent this data

```
def svd(A, tol=1e-5):
    eigs, V = np.linalg.eig(A.T.dot(A))
    sing_vals = np.sqrt(eigs)
    idx = np.argsort(sing_vals)
    sing_vals = sing_vals[idx[::-1]]
    V = V[:, idx[::-1]]
    sing_vals_trunc = sing_vals[sing_vals > tol]
    V = V[:, sing_vals > tol]
    sigma = sing_vals_trunc
    U = A @ V / sing_vals_trunc
    return U.real, sigma.real, V.T.real


def truncate(U, S, V, k):
    U_trunc = U[:, :k]
    S_trunc = S[:k]
    V_trunc = V[:k, :]
    return U_trunc, S_trunc, V_trunc


A = df.values
U, S, Vt = svd(A)
plt.figure(figsize=(5, 5))
plt.barh(df.columns[:12], S[::-1])
plt.ylabel("Features")
plt.title("Singular values")
plt.gca().xaxis.grid(True)
```



Here we will take the first 7 singular values, ie. the first 7 features in the bar graph shown above.

4b. Visualize the data after applying SVD

The dataset is reduced to 7 features namely 30d % chng, 365d % chng, 52w L, 52w H, Turnover etc. of the stocks. The initial 12 columns of the dataset is reduced to now 7 columns.

5. State your conclusions as to how SVD and PCA have helped here

In this dataset, we successfully reduced the number of dimensions using SVD and PCA. After performing reduction, the dataset was reduced to 7 features using SVD and 5 features using PCA, hence making the dataset much more noise free and easier to work with while training the model.