Implementing_PCA

February 2, 2020

0.1 Implementing Principal Component Analysis

- Author: Vincent Lee
- Date: 2 Feb 2020

```
In [135]: import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt

          from sklearn.model_selection import train_test_split
          from sklearn import preprocessing
          from sklearn import datasets
          from sklearn.decomposition import PCA
          from sklearn.metrics import r2_score

          import warnings
          warnings.filterwarnings("ignore")
          %matplotlib inline
```

0.1.1 Read data

```
In [2]: GermanCredit = pd.read_table("http://archive.ics.uci.edu/ml/machine-learning-databases/
                     sep=" ", header=None)

        GermanCredit.columns = ["check_account", "Duration", "Credit_history", "Purpose",
                                 "Amount", "Saving_acct", "Present_employment",
                                 "Installment_rate", "Sex", "Other_debtor",
                                 "Present_resident", "Property", "Age", "Other_installment"
                                 "Housing", "N_credits", "Job", "N_people", "Telephone",
                             "Foreign", "Response"]
```

```
In [3]: GermanCredit = GermanCredit.select_dtypes(include=['int64'])
        GermanCredit.head()
```

```
Out[3]:    Duration  Amount  Installment_rate  Present_resident  Age  N_credits  \
        0         6    1169                 4                 4   67          2
        1        48    5951                 2                 2   22          1
```

```
2        12     2096                 2                 3    49             1
3        42     7882                 2                 4    45             1
4        24     4870                 3                 4    53             2

   N_people  Response
0         1         1
1         1         2
2         2         1
3         2         1
4         2         2
```

In [4]:
```
GermanCredit.columns
```

Out[4]:
```
Index(['Duration', 'Amount', 'Installment_rate', 'Present_resident', 'Age',
       'N_credits', 'N_people', 'Response'],
      dtype='object')
```

0.1.2 Scaled GermanCredit data

In [5]:
```
scaler = preprocessing.StandardScaler()
GC_scaled = scaler.fit_transform(GermanCredit)
GC_scaled = pd.DataFrame(GC_scaled, columns=GermanCredit.columns)
GC_scaled.head()
```

Out[5]:
```
   Duration    Amount  Installment_rate  Present_resident       Age  \
0 -1.236478 -0.745131          0.918477          1.046987  2.766456
1  2.248194  0.949817         -0.870183         -0.765977 -1.191404
2 -0.738668 -0.416562         -0.870183          0.140505  1.183312
3  1.750384  1.634247         -0.870183          1.046987  0.831502
4  0.256953  0.566664          0.024147          1.046987  1.535122

   N_credits  N_people  Response
0   1.027079 -0.428290 -0.654654
1  -0.704926 -0.428290  1.527525
2  -0.704926  2.334869 -0.654654
3  -0.704926  2.334869 -0.654654
4   1.027079  2.334869  1.527525
```

0.1.3 Split data into train and test

In [6]:
```
GC_train,GC_test =train_test_split(GC_scaled, test_size=0.3, random_state= 43)
```

In [7]:
```
print(GC_train.shape)
print(GC_test.shape)
```

```
(700, 8)
(300, 8)
```

0.2 Implement PCA until 6th components

```
In [8]: # to get the loadings
        pca_GC = PCA(n_components=6)

        # to get the scores
        X_reduced = pca_GC.fit_transform(GC_train)
```

Explained variance

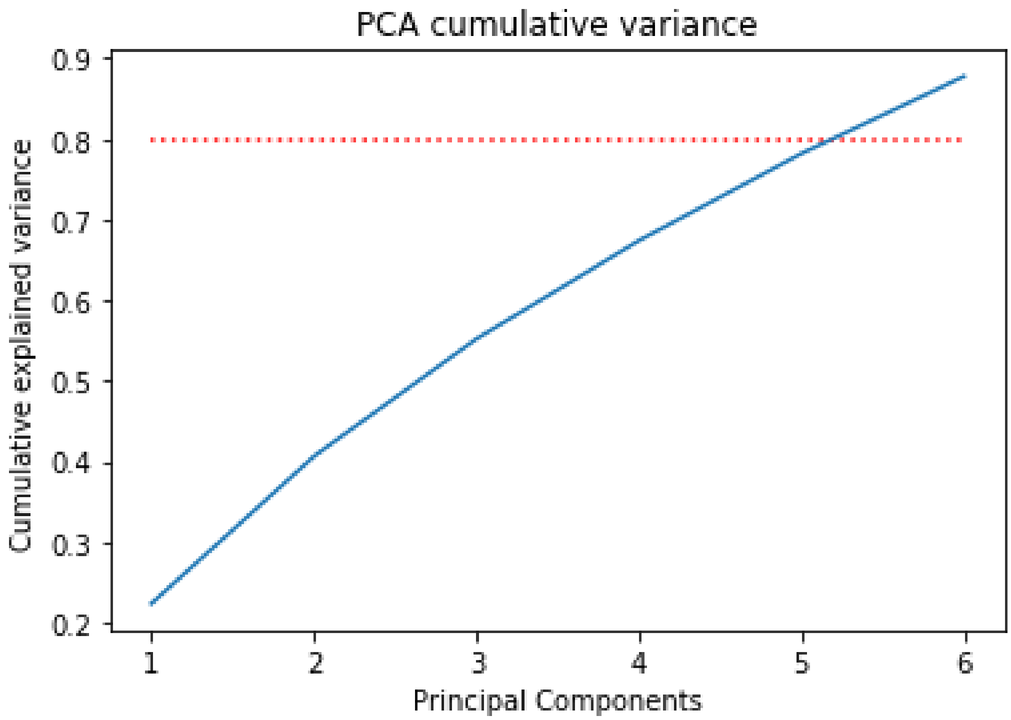
```
In [9]: pca_GC.explained_variance_ratio_

Out[9]: array([0.22364151, 0.1826132 , 0.14573814, 0.12183342, 0.10803265,
               0.09559986])
```

Cumulative explained variance

```
In [10]: cumsum=np.cumsum(pca_GC.explained_variance_ratio_)
         print('Cumsum is', cumsum)

Cumsum is [0.22364151 0.40625471 0.55199286 0.67382627 0.78185892 0.87745878]
```

Scree plot

```
In [11]: x = [1,2,3,4,5,6]
         plt.plot(x,cumsum)
         plt.title("PCA cumulative variance")
         plt.xlabel("Principal Components")
         plt.ylabel("Cumulative explained variance")
         plt.hlines(y=0.8,xmin=1, xmax=6,colors='red',linestyles='dotted')
         plt.show()
```



0.3 Interpret the loadings

```
In [126]: print(pca_GC.components_.T)
```

```
[[ 0.66320283 -0.14101746 -0.10326735  0.15056495 -0.19110495 -0.30857092]
 [ 0.61606633  0.07243651  0.13172021 -0.10988157 -0.18769314 -0.13199853]
 [-0.11387621 -0.15647886 -0.64782663  0.63143035 -0.02654414 -0.16098663]
 [ 0.09119126  0.23668758 -0.57487584 -0.59208254  0.04066465  0.15722942]
 [-0.00570777  0.57621083 -0.35705102 -0.03659787 -0.11063598 -0.15383142]
 [ 0.27765578  0.40042329  0.07269226  0.42204347  0.03682717  0.74910728]
 [-0.01493779  0.55029563  0.23484549  0.17647082  0.50404633 -0.48480587]
 [ 0.2863428  -0.31492439 -0.18412512 -0.0734041   0.81131542  0.12984623]]
```

```
In [131]: # place the loadings into data frame
          LoadingsMatrix = pd.DataFrame(pca_GC.components_,
                                        columns=GermanCredit.columns)
          LoadingsMatrix = np.transpose(LoadingsMatrix)
          LoadingsMatrix
```

Out[131]:

	0	1	2	3	4	5
Duration	0.663203	-0.141017	-0.103267	0.150565	-0.191105	-0.308571
Amount	0.616066	0.072437	0.131720	-0.109882	-0.187693	-0.131999
Installment_rate	-0.113876	-0.156479	-0.647827	0.631430	-0.026544	-0.160987

```
Present_resident  0.091191  0.236688 -0.574876 -0.592083  0.040665  0.157229
Age              -0.005708  0.576211 -0.357051 -0.036598 -0.110636 -0.153831
N_credits         0.277656  0.400423  0.072692  0.422043  0.036827  0.749107
N_people         -0.014938  0.550296  0.234845  0.176471  0.504046 -0.484806
Response          0.286343 -0.314924 -0.184125 -0.073404  0.811315  0.129846
```

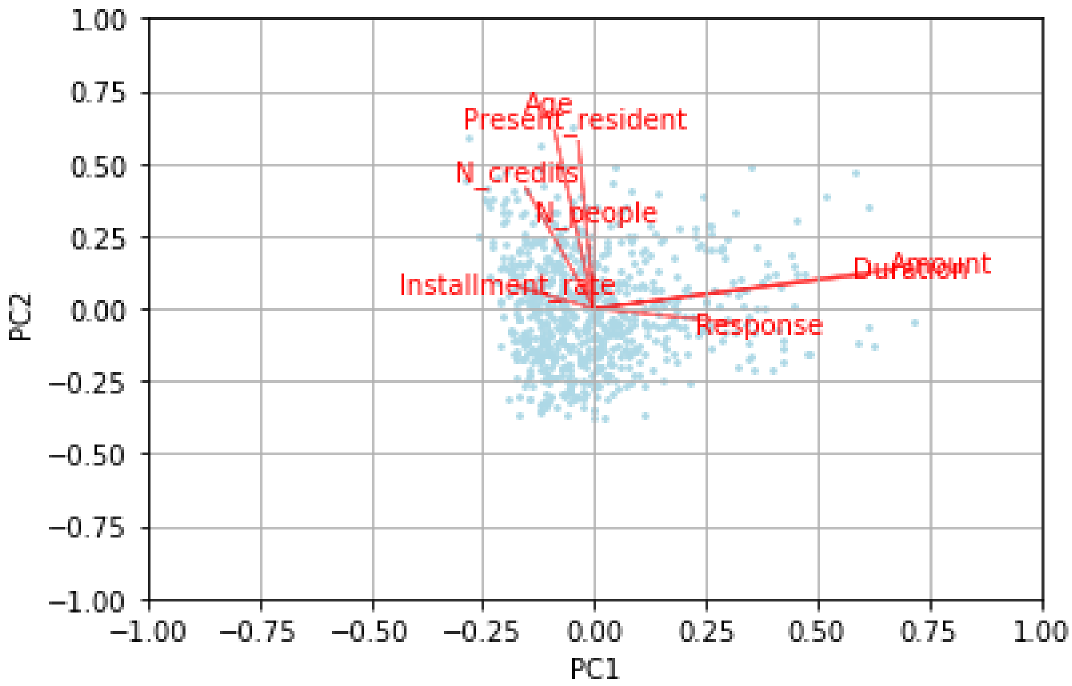
Draw the biplot Print Reduced X matrix

```
In [44]: print("Reduced X matrix is", X_reduced)
```

```
Reduced X matrix is [[-0.32368606 -0.95123701 -1.39317808  0.52649094 -0.48663725 -0.63681898]
 [ 2.13011452 -0.4960478  -1.67104291  0.15155452 -0.1254104  -0.10183121]
 [-1.14120024  1.42450464  0.01164134  1.15783004 -1.74257897  1.27623733]
 ...
 [ 3.31043653 -1.36022506  1.43232873 -0.72896949  0.75993065  1.24353535]
 [ 0.96124022 -0.43015936 -1.40671536  0.85672279  1.43836418 -0.43254932]
 [-1.14776842 -1.75386904 -0.48177965 -0.09587215  0.03015272  0.65960971]]
```

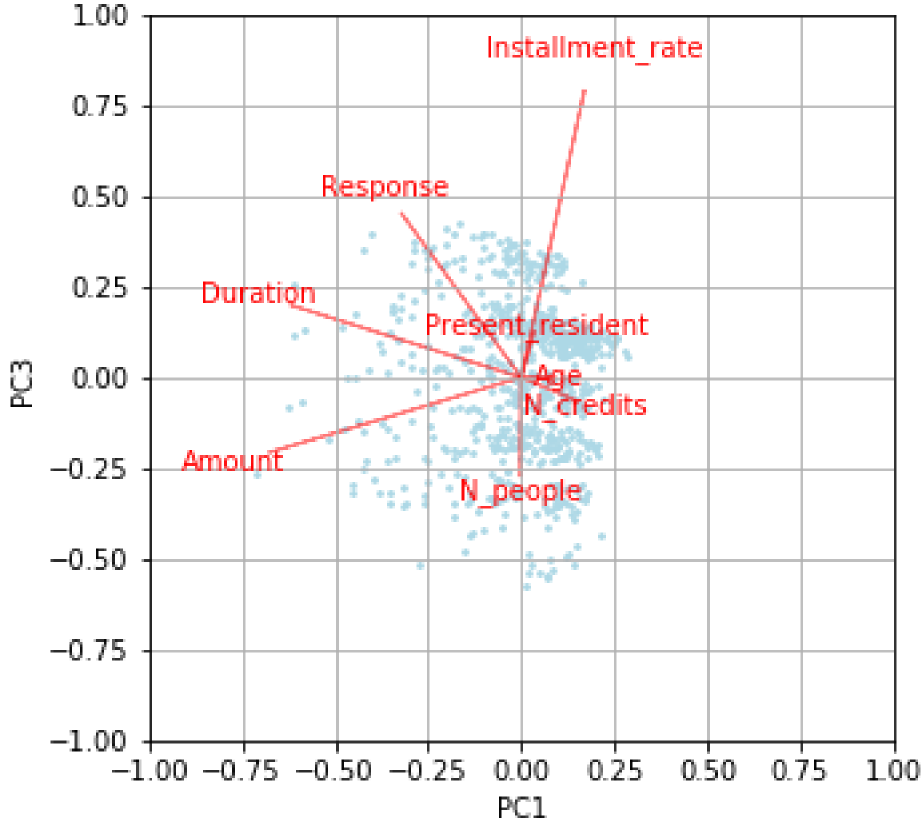
Plot PC1 versus PC2

```
In [40]: def bi_plot(score, loadings,labels=None):
             xs = score[:,0]
             ys = score[:,1]
             n  = loadings.shape[0]
             scalex = 1.0/(xs.max() - xs.min())
             scaley = 1.0/(ys.max() - ys.min())
             plt.scatter(xs * scalex,ys * scaley,s=3, color='lightblue')
             for i in range(n):
                 plt.arrow(0, 0, loadings[i,0], loadings[i,1],color = 'r',alpha = 0.5)
                 if labels is None:
                     plt.text(loadings[i,0]* 1.15, loadings[i,1] * 1.15, "Var"+str(i+1),
                              color = 'red', ha = 'center', va = 'center')
                 else:
                     plt.text(loadings[i,0]* 1.15, loadings[i,1] * 1.15, labels[i],
                              color = 'red', ha = 'center', va = 'center')

             plt.xlabel("PC{}".format(1))
             plt.xlim(-1,1)
             plt.ylim(-1,1)
             plt.grid()
             plt.show()

         plt.ylabel("PC2")
         bi_plot(X_reduced[:,0:2],np.transpose(pca_GC.components_[0:2, :]),list(GermanCredit.c
```



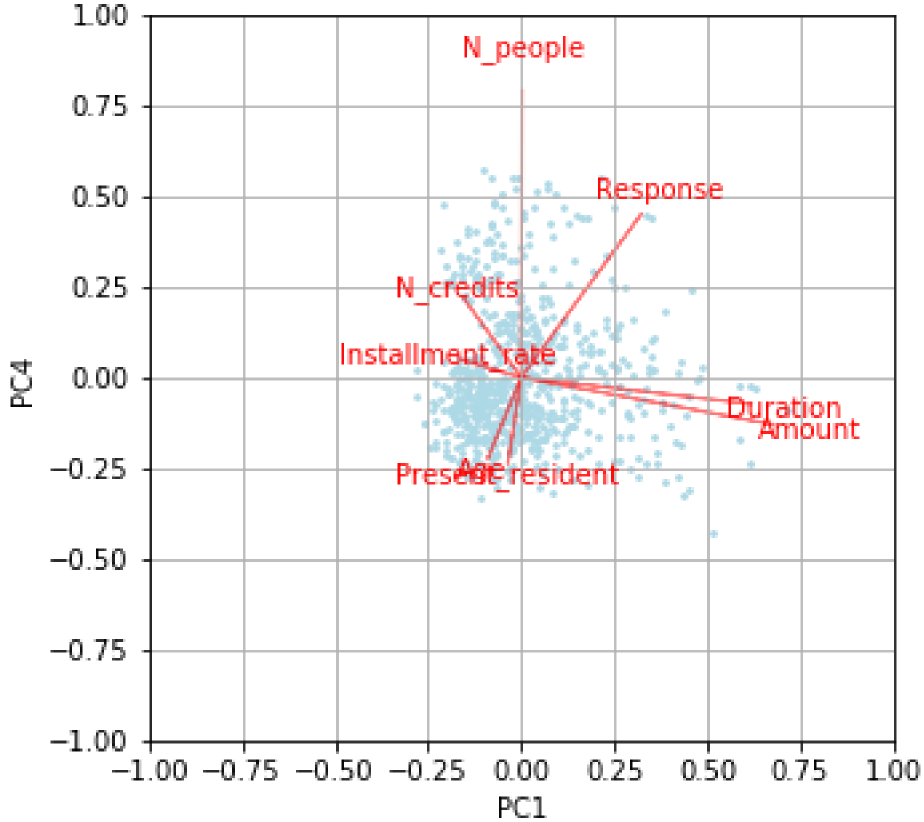
Plot PC1 and PC3

In [41]:
```
plt.figure(figsize=(5,5))
plt.ylabel("PC3")
bi_plot(-X_reduced[:,[0,2]],-np.transpose(pca_GC.components_[[0,2], :]),list(GermanCre
```



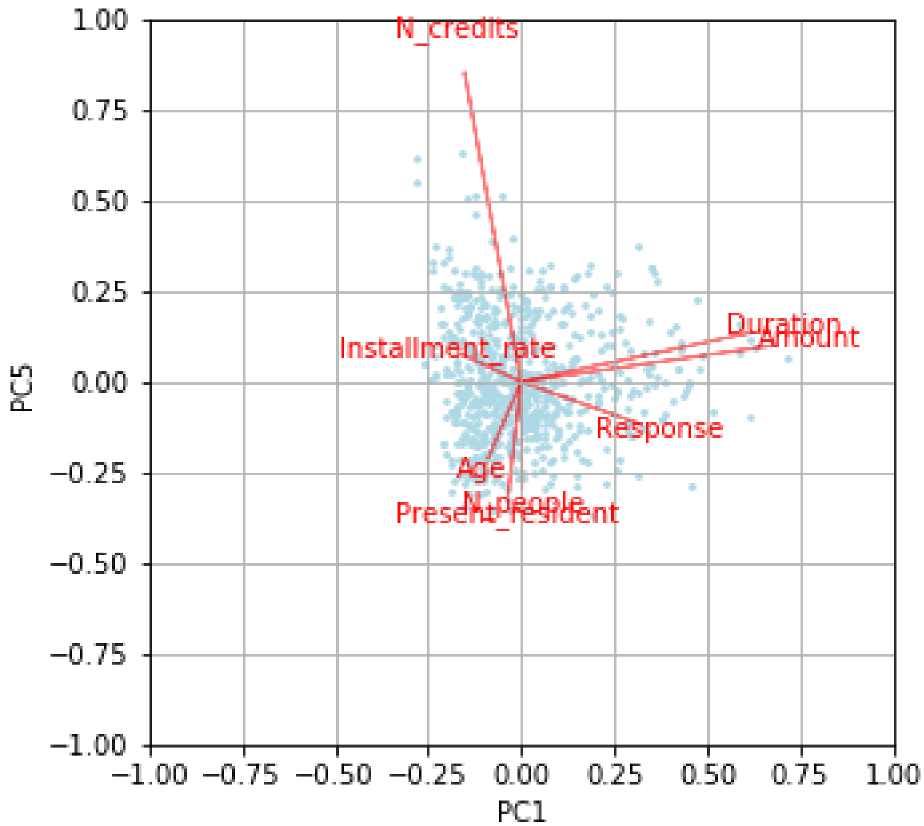
Plot PC 1 and PC4

```
In [45]: plt.figure(figsize=(5,5))
         plt.ylabel("PC4")
         bi_plot(X_reduced[:,[0,3]],np.transpose(pca_GC.components_[[0,3], :]),list(GermanCredi
```



Plot PC 1 and PC5

```
In [44]: plt.figure(figsize=(5,5))
         plt.ylabel("PC5")
         bi_plot(X_reduced[:,[0,4]],np.transpose(pca_GC.components_[[0,4], :]),list(GermanCredi
```



0.4 Show component loadings are orthogonal

In [59]:
```
# dot product of pca loadings and pca loadings transpose
np.round_(np.dot(pca_GC.components_, pca_GC.components_.T),2)
```

Out[59]:
```
array([[ 1.,  0., -0., -0., -0., -0.],
       [ 0.,  1.,  0.,  0., -0., -0.],
       [-0.,  0.,  1.,  0.,  0.,  0.],
       [-0.,  0.,  0.,  1.,  0., -0.],
       [-0., -0.,  0.,  0.,  1., -0.],
       [-0., -0.,  0., -0., -0.,  1.]])
```

0.5 Show component scores are orthogonal

In [66]:
```
np.round_(np.dot(X_reduced/(700-1), X_reduced.T),2)
```

Out[66]:
```
array([[ 0.01,  0.  , -0.  , ..., -0.  ,  0.  ,  0.  ],
       [ 0.  ,  0.01, -0.  , ...,  0.01,  0.01, -0.  ],
       [-0.  , -0.  ,  0.01, ..., -0.01, -0.01, -0.  ],
       ...,
       [-0.  ,  0.01, -0.01, ...,  0.03,  0.  , -0.  ],
```

```
[ 0.  ,  0.01, -0.01, ...,  0.  ,  0.01,  0.  ],
[ 0.  , -0.  , -0.  , ..., -0.  ,  0.  ,  0.01]])
```

0.6 Perform holdout validation of PC solution

```
In [67]: # to get the scores for test data
         X_test_reduced = pca_GC.fit_transform(GC_test)
```

```
In [116]: # dot product test scores and transpose of train loadings
          matrix_multiplication = np.dot(X_test_reduced[:,0:5],
                                         pca_GC.components_[0:5,:])
```

```
In [132]: np.round_(r2_score(GC_test, matrix_multiplication),2)
```

```
Out[132]: 0.77
```

Rsquared at holdout is around 77%

0.7 Rotate component loadings using varimax rotation

```
In [149]: # Source: https://en.wikipedia.org/wiki/Talk%3AVarimax_rotation

          from numpy import eye, asarray, dot, sum, diag
          from numpy.linalg import svd

          def varimax(Phi, gamma = 1, q = 20, tol = 1e-6):
              p,k = Phi.shape
              R = eye(k)
              d=0
              for i in range(q):
                  d_old = d
                  Lambda = dot(Phi, R)
                  u,s,vh = svd(dot(Phi.T,asarray(Lambda)**3 - (gamma/p) * dot(Lambda, diag(dia
                  R = dot(u,vh)
                  d = sum(s)
                  if d/d_old < tol: break
              return dot(Phi, R)
```

```
In [162]: # pre rotation loadings
          # 0 = PC1, 1= PC2, ..., 5=PC6
          LoadingsMatrix
```

Out[162]:

	0	1	2	3	4	5
Duration	0.663203	-0.141017	-0.103267	0.150565	-0.191105	-0.308571
Amount	0.616066	0.072437	0.131720	-0.109882	-0.187693	-0.131999
Installment_rate	-0.113876	-0.156479	-0.647827	0.631430	-0.026544	-0.160987
Present_resident	0.091191	0.236688	-0.574876	-0.592083	0.040665	0.157229
Age	-0.005708	0.576211	-0.357051	-0.036598	-0.110636	-0.153831
N_credits	0.277656	0.400423	0.072692	0.422043	0.036827	0.749107
N_people	-0.014938	0.550296	0.234845	0.176471	0.504046	-0.484806
Response	0.286343	-0.314924	-0.184125	-0.073404	0.811315	0.129846

In [159]: rotated_components = varimax(pca_GC.components_.T)

In [160]: rotated_components

```
Out[160]: array([[ 7.70270313e-01, -2.65996115e-02, -1.55092797e-01,
                   4.82985869e-02,  4.16942195e-02, -5.14407601e-02],
                 [ 6.35010230e-01,  1.89691118e-02,  2.34019977e-01,
                  -5.14407897e-02, -4.08434438e-02,  6.46819648e-02],
                 [ 1.93062925e-02, -3.04049621e-02, -9.38076355e-01,
                   2.38601152e-02,  2.45371083e-02,  1.59052380e-03],
                 [-2.35119708e-02, -1.27927392e-01,  6.51699223e-02,
                  -8.56856463e-01,  1.20637674e-01, -4.28221492e-02],
                 [ 4.69404791e-02,  3.32136045e-01, -1.86308715e-01,
                  -5.01640770e-01, -3.00190806e-01,  8.82840905e-02],
                 [-7.22797571e-03, -1.66272253e-02, -8.49187721e-04,
                   1.65046605e-02,  1.69611811e-02,  9.91193640e-01],
                 [-1.11731204e-02,  9.31324214e-01,  3.74398671e-02,
                   6.86260471e-02,  6.53566835e-02, -2.38705896e-02],
                 [ 1.18159128e-02,  6.07218734e-02, -2.88635970e-02,
                  -6.01945886e-02,  9.41681575e-01,  2.24745973e-02]])
```

In [165]:
```
# place the loadings into data frame
LoadingsRotated = pd.DataFrame(rotated_components.T,
                               columns=GermanCredit.columns)
LoadingsRotated = np.round_(np.transpose(LoadingsRotated),2)
LoadingsRotated
```

Out[165]:

	0	1	2	3	4	5
Duration	0.77	-0.03	-0.16	0.05	0.04	-0.05
Amount	0.64	0.02	0.23	-0.05	-0.04	0.06
Installment_rate	0.02	-0.03	-0.94	0.02	0.02	0.00
Present_resident	-0.02	-0.13	0.07	-0.86	0.12	-0.04
Age	0.05	0.33	-0.19	-0.50	-0.30	0.09
N_credits	-0.01	-0.02	-0.00	0.02	0.02	0.99
N_people	-0.01	0.93	0.04	0.07	0.07	-0.02
Response	0.01	0.06	-0.03	-0.06	0.94	0.02