Supplementary Content

Appendix A

```
# loading dataset into Pandas DataFrame
url=https://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data
df = pd.read_csv(url, names=['sepal length','sepal width','petal
length','                                  petal width','target'])
df.head()

# standardize data as its in feature subspace where variance along
axes
# must be maximized
features = ['sepal length', 'sepal width', 'petal length', 'petal
width']
x = df.loc[:, features].values
x = StandardScaler().fit_transform(x)
y = df.loc[:,['target']].values
pd.DataFrame(data = x, columns = features).head()

# project PCA data to 2D space
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents,
       columns = ['principal component 1', 'principal component 2'])
principalDf.head(5)
df[['target']].head()
finalDf = pd.concat([principalDf, df[['target']]], axis = 1)
finalDf.head(5)

# PCA projection used to visualize 2D plot using different colors for
each
# class of iris plants
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 Component PCA', fontsize = 20)
targets = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['target'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , c = color
```

```
                , s = 50)
ax.legend(targets)
ax.grid()

# calculate variance from the data in each plant species
pca.explained_variance_ratio_
```

Appendix B

```
# arbitrary number of points with neighbors chosen within range of fig
ure size
# number of components = axes/parameters
n_points = 500
n_neighbors = 10
n_components = 3

# plot figures
fig = plt.figure(figsize=(16, 8))
plt.suptitle("Manifold Learning with %i points, %i neighbors, %i compo
nents"
             % (n_points, n_neighbors, n_components), fontsize=14)

# create isomap from point data distances
# calculate time to construct manifold
t0 = time()
Y = manifold.Isomap(n_neighbors, n_components).fit_transform(X)
t1 = time()
print("Isomap: %.2g sec" % (t1 - t0))
ax = fig.add_subplot(257)
plt.scatter(Y[:, 0], Y[:, 1], c=color, cmap=plt.cm.Spectral)
plt.title("Isomap (%.2g sec)" % (t1 - t0))
ax.xaxis.set_major_formatter(NullFormatter())
ax.yaxis.set_major_formatter(NullFormatter())
plt.axis('tight')

# create MDS
# show time to accomplish, compare to isomap
t0 = time()
mds = manifold.MDS(n_components, max_iter=100, n_init=1)
Y = mds.fit_transform(X)
t1 = time()
print("MDS: %.2g sec" % (t1 - t0))
ax = fig.add_subplot(258)
plt.scatter(Y[:, 0], Y[:, 1], c=color, cmap=plt.cm.Spectral)
plt.title("MDS (%.2g sec)" % (t1 - t0))
ax.xaxis.set_major_formatter(NullFormatter())
ax.yaxis.set_major_formatter(NullFormatter())
plt.axis('tight')
```