Carrying out Dimensionality Reduction

Shizhi Chen-10307389

1. Dimensionality reduction method

PCA (Principal Components Analysis)

PCA is a very common and effective method of dimensionality reduction. The data is converted from the original feature space to the new feature space, the number of features of the data is changed from n to k (k < n). For example, the original space is three-dimensional (x, y, z), and x, y, and z are the three bases of the original space, respectively. By some means, using the new coordinate system (a, b, c) to represent the original data, then a, b, c are the new base, they form a new feature space. In the new feature space, the projection of all the data on c may be close to 0, which can be ignored, then we can directly use (a, b) to represent the data, so that the data is from three-dimensional (x, y, z) fell to two-dimensional (a, b).

Random Forests

Random forest is a widely used feature selection algorithm. This dimension reduction method is to generate many huge trees for the target features, and then find the feature subset with the largest amount of information according to the statistical results of each feature. For example, we can generate very shallow trees for a very large data set, each tree training only a small number of features. If a feature is often the best split feature, then it is most likely an information feature that needs to be preserved. The statistical scoring of random forest data features reveals to us which feature is the best predictive feature compared to other features. In the case of setting a certain threshold, the best features are retained to achieve dimensionality reduction.

2. Variables selection and Pre-processing

We are going to use the dataset from the 2012 U.S. Army Anthropometric Survey (ANSUR II Male). Including 93 anthropometric measurements which were directly measured, and 15 demographic/administrative variables. The ANSUR II Male working database contains a total sample of 4,082 subjects.

First, import some useful packages and load the dataset.

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
plt.style.use('ggplot')
```

```
df = pd.read_csv('./ANSUR_II_MALE_Public.csv', encoding='latin-1')
```

Then, find if there are any missing values in the dataset,

```
#find the missing values in the dataset
# missing percentage
(df.isnull().sum()/len(df)*100).sort_values(ascending = False).head()
```

```
Ethnicity                        77.902989
WritingPreference                 0.000000
forearmforearmbreadth             0.000000
crotchlengthomphalion             0.000000
crotchlengthposterioromphalion    0.000000
dtype: float64
```

```
df = df.drop(['Ethnicity'],axis = 1)
```

The only feature with missing values is 'Ethnicity' which has 78% in total missing values, so we first remove this feature.

In the remaining features, I am interested in the relationship between the length of the hand and other length data in the anthropometric, so I randomly selected 10 anthropometric length data including the length of the hand from the dataset.

```
#select 10 length data (randomly)
len_data = df[['handlength','acromionradialelength',
               'balloffootlength','earlength',
               'footlength','forearmhandlength',
               'functionalleglength','headlength',
               'palmlength','shoulderlength']]
```

```
len_data.head()
```

	handlength	acromionradialelength	balloffootlength	earlength	footlength	forearmhandlength	functionalleglength	headlength	palmlength	shoulderlength
0	193	337	202	71	273	477	1136	206	113	145
1	195	326	193	62	263	476	1096	201	118	141
2	203	341	196	61	270	491	1115	202	121	167
3	194	310	199	66	267	467	1034	199	118	148
4	218	372	224	56	305	550	1279	197	132	180

Although all features measurements are recorded in millimeters, the data ranges of different dimensions are very different, standardization processing is required.

```
# Standardization
scaler = StandardScaler()
scaled_data = scaler.fit_transform(len_data.values)
scaled_data = pd.DataFrame(scaled_data)
scaled_data.columns = ['handlength','acromionradialelength',
                       'balloffootlength','earlength',
                       'footlength','forearmhandlength',
                       'functionalleglength','headlength',
                       'palmlength','shoulderlength']
```

After standardization, the average value of each column dimension is 0, and the variance is 1.

	handlength	acromionradialelength	balloffootlength	earlength	footlength	forearmhandlength	functionalleglength	headlength	palmlength	shoulderlength
0	-0.027933	0.100426	0.101715	1.505759	0.139047	-0.138031	0.104153	0.924781	-0.554558	-0.454346
1	0.173167	-0.528841	-0.757912	-0.491446	-0.624254	-0.180980	-0.608384	0.212901	0.247776	-0.834863
2	0.977571	0.329250	-0.471370	-0.713357	-0.089943	0.463251	-0.269929	0.355277	0.729177	1.638497
3	0.072617	-1.444139	-0.184827	0.396201	-0.318933	-0.567518	-1.712816	-0.071851	0.247776	-0.168958
4	2.485828	2.102639	2.203024	-1.822916	2.581610	2.997224	2.651473	-0.356602	2.494313	2.875177

Finally, we can see the correlation between those features by the scatter pair plot.

```
#Variable correlation scatter plot
sns.pairplot(scaled_data)
plt.show()
```

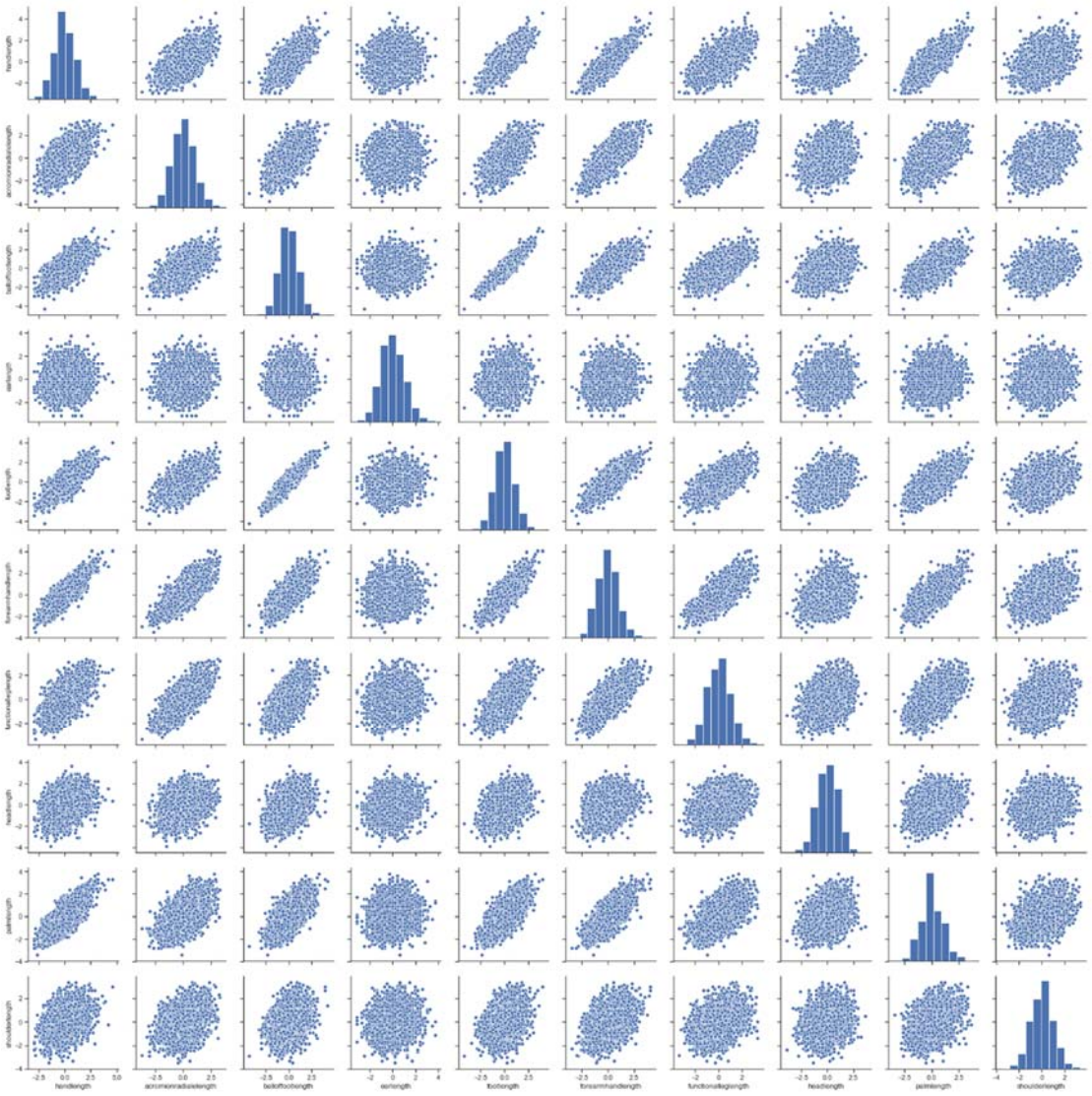

Figure 1: Scatter plot of correlation between features

From the figure, we can see that most features contain a linear relationship from others which is good for us to do the PCA and Random Projection.

3. Description, interpretation and assessment of success of the results of the dimensionality reduction techniques on the data.

The specific implementation process of **PCA** is as follows:

1. Standardize raw dimension data;

2. Calculating a covariance matrix of multidimensional features to obtain eigenvalues and eigenvectors of the matrix;

2. Sorting the feature values from large to small, selecting the first k principal components, and finding the corresponding k feature vectors;

4. Projecting the original dimensional data onto the selected k feature vectors, the dimension of the original data feature becomes k-dimensional;

5. This k-dimensional data can be used to represent the original large-dimensional data for subsequent data processing analysis.

Here we directly call the PCA method of the sklearn package to create an object, input the normalized raw dimension data, and return the dimensionally reduced data object pcaMat.

```
pca=PCA(n_components=0.95)
#return the dimensionality data
pcaMat = pca.fit_transform(scaled_data)
print("Explained Variance:\n",pca.explained_variance_)
print("Explained Variance Ratio:\n",pca.explained_variance_ratio_)
print("Number of components:",pca.n_components_)
```

```
Explained Variance:
 [5.64021335 1.03070471 0.8246139  0.77513973 0.6271615  0.47757242
 0.26116148]
Explained Variance Ratio:
 [0.56388316 0.10304522 0.08244119 0.07749498 0.06270079 0.04774554
 0.02610975]
Number of components: 7
```

n_components can specify the variance of the principal component and the minimum proportional threshold so that the PCA itself can determine the number of dimensions to be dimensioned according to the sample feature variance. In order to retain 95% of the principal component information, we can set the parameter to 0.95.

Visualizing PCA variance contribution rate by the following code:

```
#Visualize PCA variance contribution rate
plt.figure(figsize = (12,8))
plt.bar(['PC1','PC2','PC3','PC4','PC5','PC6','PC7'], pca.explained_variance_ratio_, alpha=0.5, align='center',
       label = "individual explained variance")
plt.step(range(0, 7), np.cumsum(pca.explained_variance_ratio_), where='mid',label = "cumulative explained variance")
plt.ylabel('Explained variance ratio')
plt.xlabel('Principal components')
plt.legend()
plt.show()
```

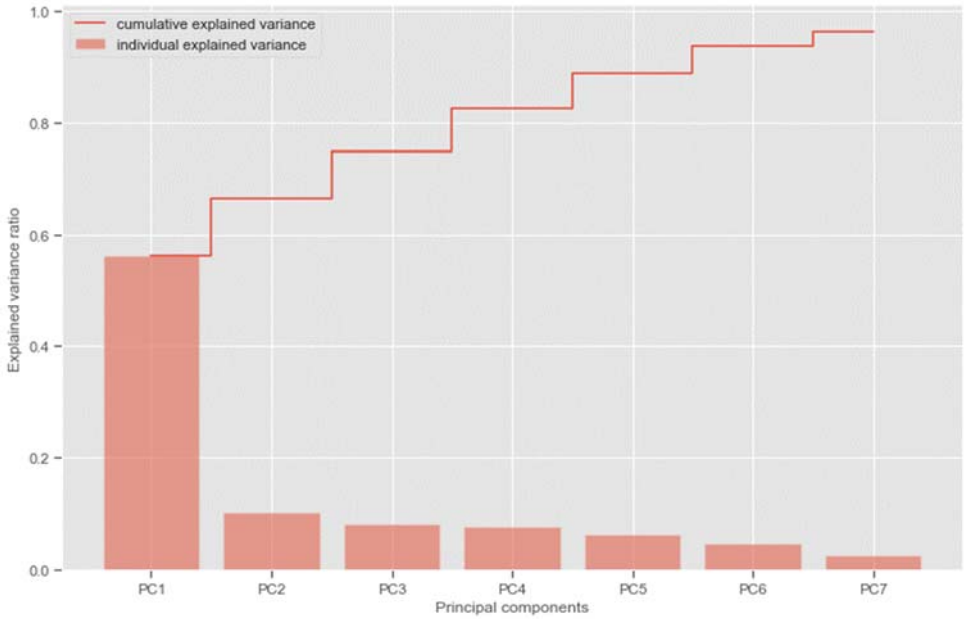

Figure 2: PCA Variance Contribution Rate

As seen from the above figure, the first column of new dimensions PC1 occupies 56% of the total variance, while the remaining six dimensions are not too different, add up to occupy 40% of the total variance. These 7 new dimensions contribute cumulative variance is about 96% of the total variance, this means that the data dimension is reduced from 10 dimensions to 7 dimensions while retaining 96% of the amount of information.

The components can be written as a sum of loadings onto variables, take the first component as an example:

```
pca.components_[0]
```

```
array([0.3728005 , 0.33582578, 0.35944966, 0.08759843, 0.38146973,
       0.39278064, 0.3425964 , 0.19107627, 0.34471081, 0.19712822])
```

PC1 = 0.373 × handlength + 0.336 × acromionradialelength + …… +

0.345 × palmlength + 0.197×shoulderlength

Another dimensionality reduction technique I used is **Random Forests**. Similarly, using the same 10 features as before. However, it is not necessary to normalize or normalize features for the tree-based models, so we use the original length dataset here.

```
len_data.head()
```

	handlength	acromionradialelength	balloffootlength	earlength	footlength	forearmhandlength	functionalleglength	headlength	palmlength	shoulderlength
0	193	337	202	71	273	477	1136	206	113	145
1	195	326	193	62	263	476	1096	201	118	141
2	203	341	196	61	270	491	1115	202	121	167
3	194	310	199	66	267	467	1034	199	118	148
4	218	372	224	56	305	550	1279	197	132	180

We are interested in the relationship between the length of the hand and other length data in the anthropometric, so we choose handlength as our dependent variable.

First, delete the dependent variable and save the remaining variables in the new data set.

```
new_df = len_data.drop('handlength',1)
```

```
new_df.head()
```

	acromionradialelength	balloffootlength	earlength	footlength	forearmhandlength	functionalleglength	headlength	palmlength	shoulderlength
0	337	202	71	273	477	1136	206	113	145
1	326	193	62	263	476	1096	201	118	141
2	341	196	61	270	491	1115	202	121	167
3	310	199	66	267	467	1034	199	118	148
4	372	224	56	305	550	1279	197	132	180

Then, train the model using the RandomForestRegressor in the scikit-learn.

```
from sklearn.ensemble import RandomForestRegressor
```

```
feat_labels = len_data.columns[1:]
model = RandomForestRegressor(random_state=1, max_depth=10)
model.fit(new_df,len_data.handlength)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=10,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
           oob_score=False, random_state=1, verbose=0, warm_start=False)
```

Next, we can see the features importance from the Random Forest model:

```
features = new_df.columns
importances = model.feature_importances_
indices = np.argsort(importances[0:9])[::-1]  # top 10 features
print(" Feature Importance")
for f in range(new_df.shape[1]):
    print("%2d. %-*s %f" % (f+1, 30 ,features[indices[f]], importances[indices[f]]))
```

```
 Feature Importance
 1. forearmhandlength              0.794502
 2. palmlength                     0.070822
 3. footlength                     0.045968
 4. functionalleglength            0.021386
 5. acromionradialelength          0.015593
 6. shoulderlength                 0.014628
 7. headlength                     0.013501
 8. earlength                      0.011976
 9. balloffootlength               0.011624
```

```
plt.figure(figsize = (10,6))
plt.title('Feature Importance')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```

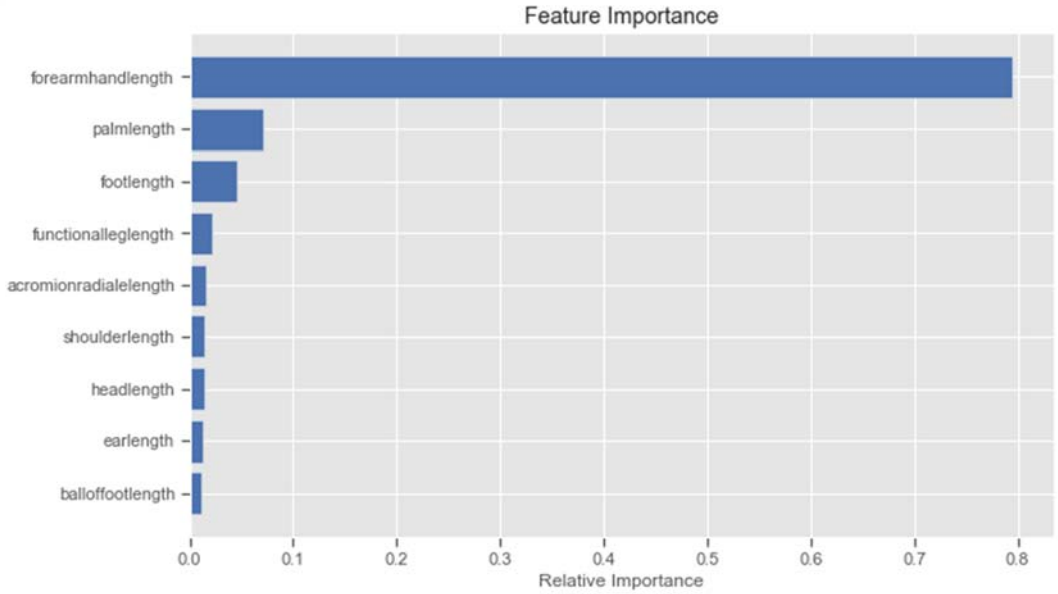

Figure 3: Feature Importance

Finally, we can use SelectFromModel directly to select features based on the importance of weights to reduce the dimensions in the dataset.

```
from sklearn.feature_selection import SelectFromModel
# Set a minimum threshold of 0.02
sfm = SelectFromModel(model, threshold=0.02)
sfm.fit(new_df,len_data.handlength)
n_features = sfm.transform(new_df).shape[1]
print("Number of features:",n_features)
```

```
Number of features: 4
```

From the results, we can see that when we set the threshold as 0.02, four dimensional features, forearmhandlength, palmlength, Footlength, functional-leglength were selected.