Lecture “Advanced Data Analytics”

Problem Set 7

Simon Scheidegger

Exercise 1

Active Subspace

a) Find the active subspace of the test function below on the domain $[-1 , 1]^{10}$
based 1000 random observations out of the domain.
Plot the Eigenvalues of the mean squared directional derivative matrix **C**, and the projection matrix **W** of the active subspace.

b) approximate the 10-dimensional function stated below with Gaussian process regression based 10, 50, 100, 500 points randomly sampled from $[-1 , 1]^{10}$. Compute the average and maximum error.

c) approximate the same function with the combination of active subspaces and Gaussian process regression. Compute the average and maximum error.

d) Compare the solution computed in b) with the solution computed in c).

$$f(x_1, ..., x_{10}) = x_9 \cdot x_{10} \cdot \exp(0.01x_1 + 0.7x_2 + 0.02x_3 + 0.03x_4 + 0.04x_5 + 0.05x_6 + 0.06x_7 + 0.08x_8 + 0.09x_9 + 0.1x_{10}).$$

Exercise 2

Principal Component Analysis (PCA)

In this exercise, we look at PCA as a tool for reducing the dimension of the feature space.
To do so, we analyze step-by-step a data set (of breast cancer).

```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
```

Print the content of the data by using this command:

```
print cancer.DESCR
```

a) How many features does the data-set have?

b) How many instances are there?

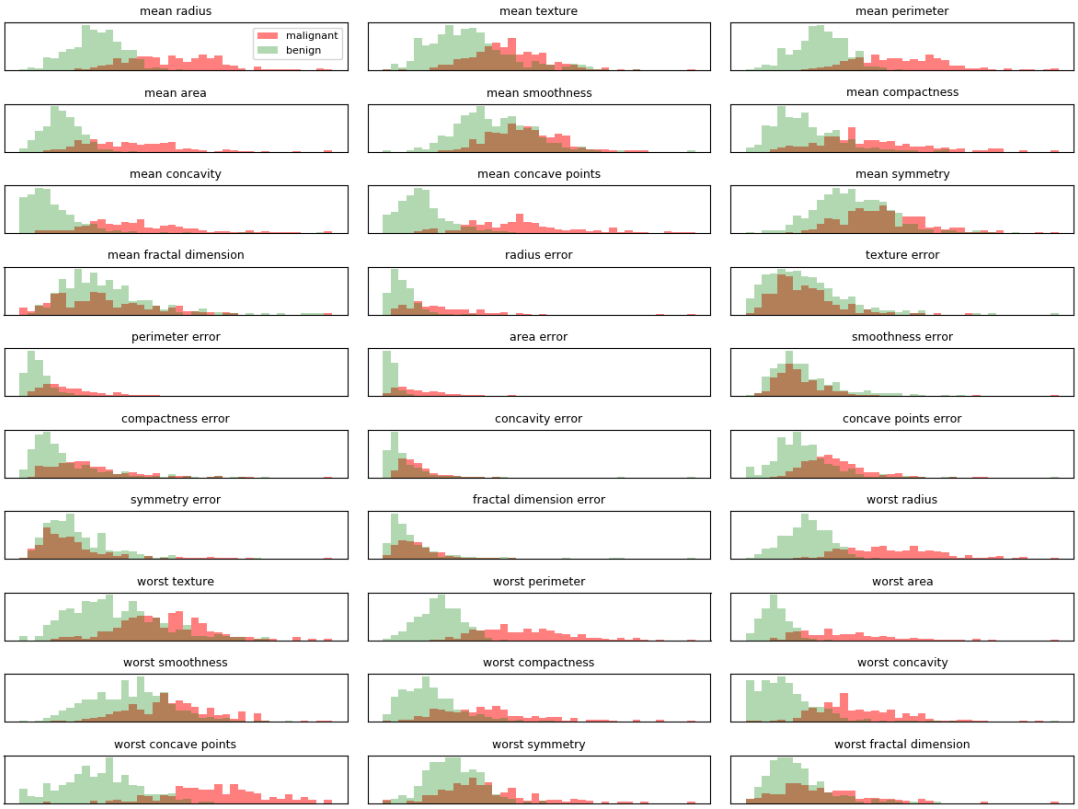
c) How many are malignang, and benign?

You can check this by typing

```
print len(cancer.data[cancer.target==1])
```

d) To know more about how the features affect the target, we next plot histograms of malignant and benign classes. The resulting plot should look similar to what you see as an example below

Note: If the two histograms are separated based on the feature, then we can say that the feature is



important to discern the instances.

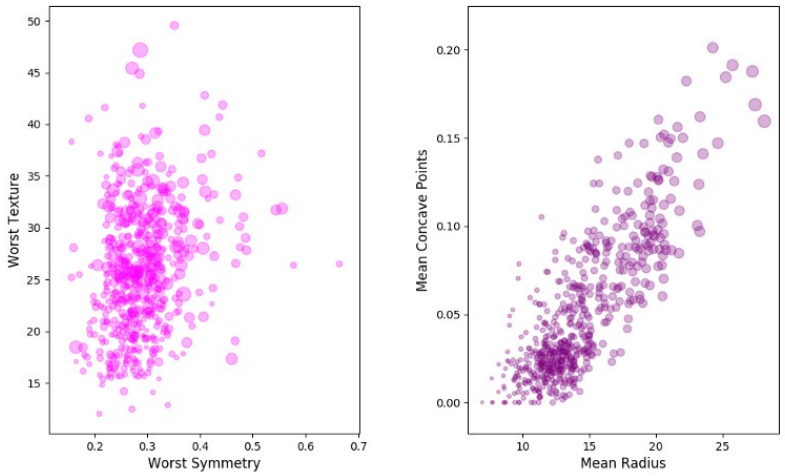
Now from these histograms we see that features like- mean fractal dimension has very little role to play in discerning malignant from benign, but worst concave points or worst perimeter are useful features that can give us strong hint about the classes of cancer data-set.

Before using PCA to these cancer data-set, let's understand very simply what PCA actually does. We know that in a data-set there are high possibilities for some features to be correlated. Let's see some example plots from cancer data set by typing those commands:

e)

```
import pandas as pd

cancer_df=pd.DataFrame(cancer.data,columns=cancer.feature_names)# just convert the scikit learn data-set to pandas data-frame.
plt.subplot(1,2,1)#fisrt plot
plt.scatter(cancer_df['worst symmetry'], cancer_df['worst texture'], s=cancer_df['worst area']*0.05, color='magenta', label='check', alpha=0.3)
plt.xlabel('Worst Symmetry',fontsize=12)
plt.ylabel('Worst Texture',fontsize=12)
plt.subplot(1,2,2)# 2nd plot
plt.scatter(cancer_df['mean radius'], cancer_df['mean concave points'], s=cancer_df['mean area']*0.05, color='purple', label='check', alpha=0.3)
plt.xlabel('Mean Radius',fontsize=12)
plt.ylabel('Mean Concave Points',fontsize=12)
plt.tight_layout()
plt.show()
```



Now hopefully you can already understand which plot shows strong correlation between the features.

f) Reduce the features of the data set down to 3 principal features.

g) Visualize the scatter plot of these new independent variables.

Note: Before applying PCA, scale our data such that each feature has unit variance.
This is necessary because fitting algorithms highly depend on the scaling of the features.

To do so, use the

`StandardScaler`

module for scaling the features individually. `StandardScaler` subtracts the mean from each features and then scale to unit variance.

We first instantiate the module and then fit to the data.

h) Now we're ready to apply PCA on this scaled data-set. We start as before with `StandardScaler` where we instantiate, then fit and finally transform the scaled data. While applying PCA you can mention how many principal components you want to keep.
Use those commands:

```
pca=PCA(n_components=3)
pca.fit(X_scaled)
X_pca=pca.transform(X_scaled)
#let's check the shape of X_pca array
print "shape of X_pca", X_pca.shape
```

i) Now one important point to note is that we have chosen 3 components instead of 2, which could have reduced the dimension of the data-set even more.

Can you choose `n_components=2`?

Check this by measuring the variance ratio of the principal components (3 and 2 components)

j) Now, since the PCA components are orthogonal to each other and they are not correlated, we can expect to see malignant and benign classes as distinct. Let's plot the malignant and benign classes based on the first two principal components. The plot you produce should look similar to the one displayed below: