Shivam Gupta, 2017HT13384@wilp.bits-pilani.ac.in, 59.49%

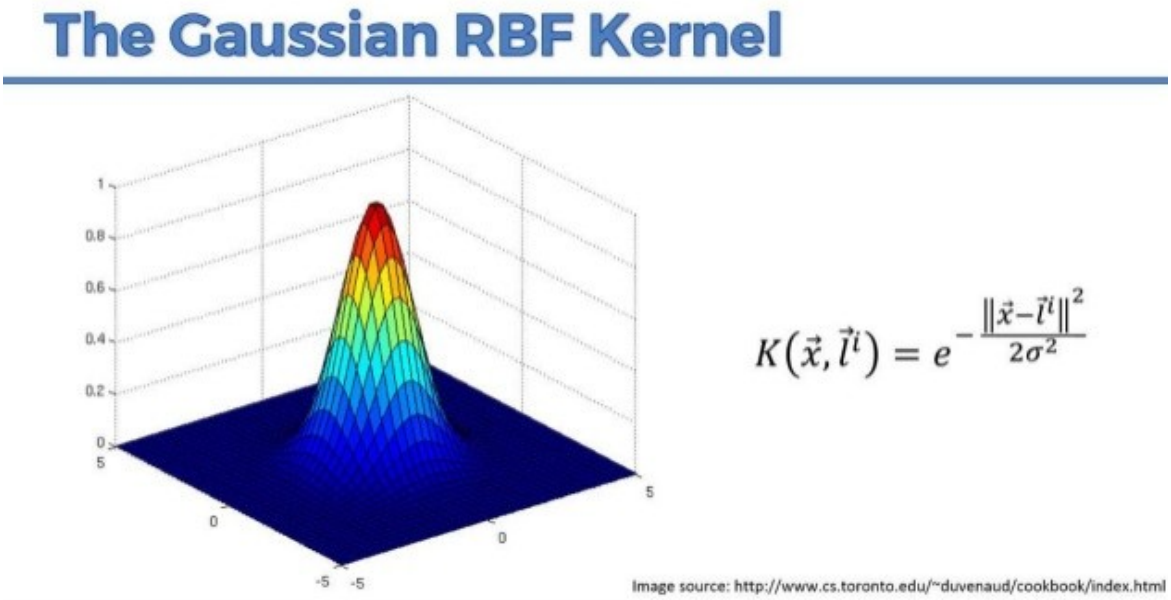
Problem Statement
For given data set, model a classifier with best accuracy.

Solution Approach
Below concepts are used to achieve 59.49% accuracy.

Feature scaling
Feature scaling is a method used to standardize the range of independent variables or features of data. **SVM classifier is not affine transformation invariant**. In other words, if you multiply one feature by a 1000 than a solution given by SVM will be completely different. It has nearly nothing to do with the underlying optimization techniques (although they are affected by these scales problems, they should still converge to global optimum).

Principal Component Analysis
Linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space.
For **n_components = 3, provides best accuracy**. And it covers (0.433 + 0.177 + 0.102) = 71.2% variance for given data set. If n_components value is increased or decreased the accuracy decreases.

Kernel Support Vector Machine (Gaussian RBF Kernel)
RBF Kernel is used beacuse data is better non-linearly separable. In below section data visualization its shown that most of red points are behind green points. So if we increase a dimension than it might be better separable. Below is the Kernel function and it's plot.



SVM's C Parameter

```
clf = SVC(kernel='rbf', random_state=0, C=9.8)
```

The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example.For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very

small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable.

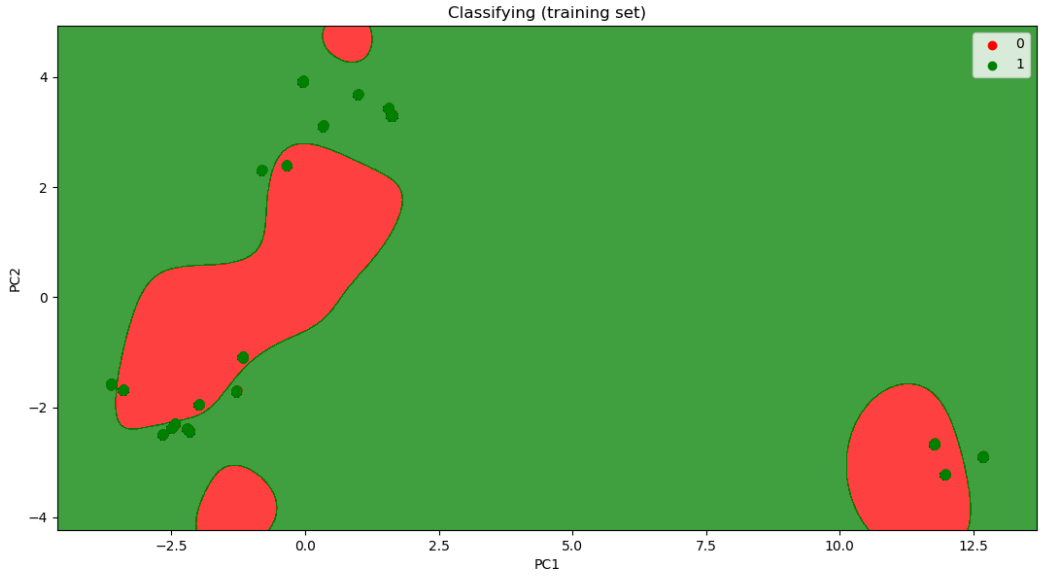
Data visualization

Data visualization for Training Set and Testing Set is shown in below sections for PCA with n_components = 2, using SVM classifier with rbf kernel.

Data set is devided into **18768 examples for Training set** and **2086 examples for Test set**.

Training Set Classification Visualization

For below plot n_components attribute of PCA is 2, here most of red points are behind green points. Code used to visualize Training Set is in appendix.
Here we can see that linear classification would be difficult.



Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

For our classifier Test set contains 2086 examples below are the results:

	0	1
0	**313**	**559**
1	**286**	**928**

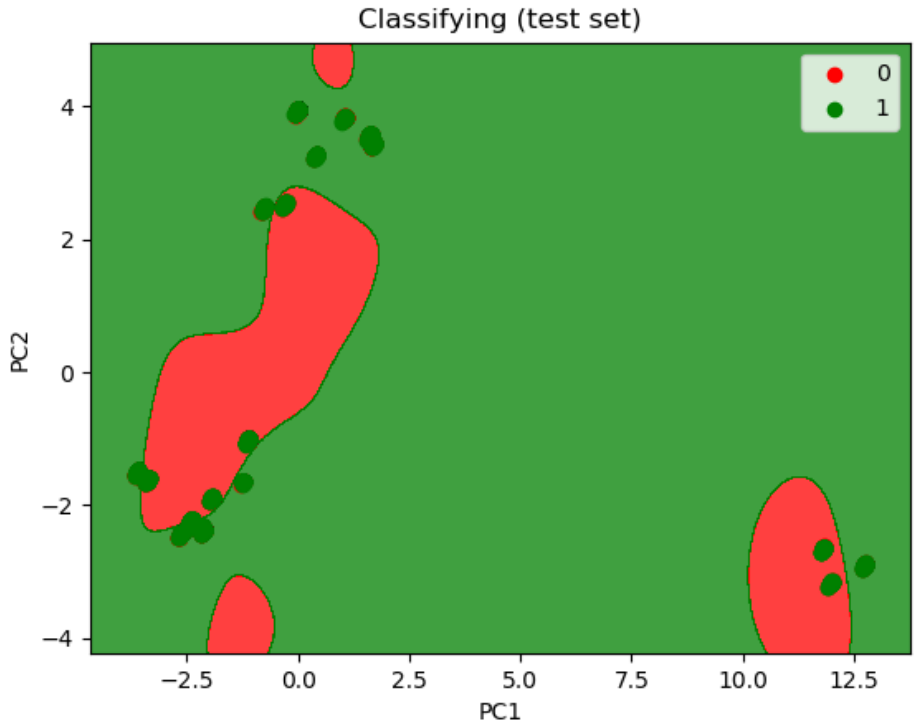
Correctly Classified are (313 + 928) = 1241
Incorrectly classified are (286 + 559) = 845

Accuracy

Accuracy achieved is 59.49%.

Test Set Classification Visualization

For below plot n_components used in PCA is 2. Code used to visualize Training Set is in appendix.



Appendix

Main Code for classification

```
#Start of code--------------------------------------
```

#Importing libraries

```
import pandas as pd
import numpy as np
```

#Importing the dataset

```
Location = 'ML_assignment.csv'
df = pd.read_csv(Location, names=
      ['v1','v2','v3','v4','v5','v6','v7','v8','v9','v10',
       'v11','v12','v13','v14','v15','v16','v17','v18','v19','v20',
       'v21','v22','v23','v24','v25','v26','v27','v28','v29','v30',
       'v31','v32','v33','v34','v35','v36','v37','v38','v39','v40',
       'v41','v42'])
y  = df['v42'].values
x  = df.iloc[:,:-1].values
```

#Splitting the data set into training set and test set

```
from sklearn.model_selection import train_test_split
X_tr, X_tst, y_tr, y_tst = train_test_split(x, y, test_size=0.1, random_state=0)
```

 # Feature scaling

```
from sklearn.preprocessing import StandardScaler
sc    = StandardScaler()
X_tr  = sc.fit_transform(X_tr)
X_tst = sc.transform(X_tst)
```

#Applying PCA

```
from sklearn.decomposition import PCA
pca   = PCA(n_components=3)
X_tr  = pca.fit_transform(X_tr)
```

```
X_tst = pca.transform(X_tst)
```

#Fitting Kernel SVM to Training Set
```
from sklearn.svm import SVC
clf = SVC(kernel='rbf', random_state=0, C=9.8)
clf.fit(X_tr, y_tr)
```

#Predicting the Test Set Results
```
predicted = clf.predict(X_tst)
```

#Measuring Accuracy
```
from sklearn import metrics
print("SVC\n", metrics.accuracy_score(y_tst, predicted))
```

#Making the Confusion Matrix
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_tst, predicted)
```

```
#End of code-------------------------------------
```

Visualizing Training set Results

#Visualizing the Training Set Results
```
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
X_set, Y_set = X_tr, y_tr
X1, X2 = np.meshgrid(np.arange(start=X_set[:,0].min() -1, stop=X_set[:,0].max()+1, step=0.01),
                     np.arange(start=X_set[:,1].min() -1, stop=X_set[:,1].max()+1, step=0.01))
plt.contourf(X1, X2, clf.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha=0.75, cmap=ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i,j in enumerate(np.unique(Y_set)):
    plt.scatter(X_set[Y_set == j, 0], X_set[Y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Classifying (training set)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.show()
```

Visualizing Test Set Results

#Visualizing the Test Set Results
```
X_set, Y_set = X_tst, y_tst
X1, X2 = np.meshgrid(np.arange(start=X_set[:,0].min() -1, stop=X_set[:,0].max()+1, step=0.01),
                     np.arange(start=X_set[:,1].min() -1, stop=X_set[:,1].max()+1, step=0.01))
plt.contourf(X1, X2, clf.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha=0.75, cmap=ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i,j in enumerate(np.unique(Y_set)):
    plt.scatter(X_set[Y_set == j, 0], X_set[Y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Classifying (test set)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.show()
```