**INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR**

**MACHINE LEARNING**

**ASSIGNMENT 3**

# SUPPORT VECTOR MACHINE CLASSIFIER

Prepared by :

Samar Pratap Singh (19EE10069)

Sawale Saurabh Suresh (19CS10054)

Group No.: 47 (E)

## Running the Code:

The .ipynb along with the dataset diabetes.csv file is already attached in the zip folder, which can be downloaded and run on Google Colab.

The code is recommended to run on Colab to avoid problems with installation packages or version incompatibility due to outdated versions.

The view link code is written on the Google colab and the link for the same is here : https://colab.research.google.com/drive/1RbNe7qHjQJerfr8KrdBt9ecCybhvnGHn

- **Importing Dataset**

  1. The dataset used is https://www.kaggle.com/mathchi/diabetes-data-set
  2. We have split the dataset as **70:10:20 splits as train, validation and test set** respectively.

  3. Attribute Information:
     ------------------------
     -- 1. Pregnancies : Number of times pregnant
     -- 2. Glucose : Plasma glucose concentration 2 hours in an oral glucose tolerance test
     -- 3 BloodPressure : Diastolic blood pressure (mm Hg)
     -- 4. SkinThickness : Triceps skin fold thickness (mm)
     -- 5. Insulin : 2-Hour serum insulin (mu U/ml)
     -- 6. BMI : Body mass index (weight in kg/(height in m)^2)
     -- 7. DiabetesPedigreeFunction : Diabetes pedigree function
     -- 8. Age : Age (years)

     Value to be predicted(having diabetes or not) :
     -- 9. Outcome : Class variable (0 or 1) where class value 1 is interpreted as "tested positive for diabetes"

- **Splitting Dataset for training and validation:**

  We split the data set into 70:10:20 splits for training, validation and testing, respectively, using the **train_valid_test_split** function which uses **sklearn's train_test_split function** twice to get the required 70:10:20 split of the dataset.

- **Data Cleaning :**

1. For missing data, we replaced the missing data with the mean value of that feature. For this, we used the pandas function : df.fillna(df.mean()). Although our dataset didn't have any missing features. We have just mentioned the function as caution.
2. We normalized the data using sklearn's MinMaxScalar().

## Functions used in the project:

1. **train_valid_test_split :** this function splits the dataset as 70:10:20 split for training validation and testing respectively

For all the plotting functions, we have used **plotly.express** library.
For min, max functions, we have directly used Python's min() and max() functions. We have used other python libraries as well in this project.

## Theory:

- **PCA(principal component analysis):**
    - Principal Component Analysis is an unsupervised learning algorithm that is used for dimensionality reduction in machine learning.
    - It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**.
    - Steps involved in PCA:
        - Standardize the range of continuous initial variables
        - Compute the covariance matrix to identify correlations
        - Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components
        - Create a feature vector to decide which principal components to keep
        - Recast the data along the principal components axes.

- **SVM(Support Vector machine):**
    - **Support vector machine (SVM)** is a supervised machine learning algorithm that can be used for both classification or regression challenge
    - SVM is mostly used in classification problems
    - SVM are simply the coordinates of individual observation. The SVM classifier is the frontier that best segregates the two classes (hyper plane/line).

- **LDA (Linear Discriminant analysis):**
    - **Linear Discriminant Analysis** or **Normal Discriminant Analysis** or **Discriminant Function Analysis** is a dimensionality reduction technique that is commonly used for supervised classification problems
    - It is used for modelling differences in groups, i.e. separating two or more classes.
    - It is used to project the features in higher dimension space into a lower dimension space.
    - LDA is like PCA, but it focuses on maximizing the separation among the known categories.

## Implementation and results:

- **Get training, validation and testing sets for operations:**
    - Get the training, validation and Testing sets for operations by calling the train_test_valid_split function.
    - since, random rows have been selected... we need to reset the indices so that the training, tests and validation X and Y are aligned w.r.t each other.

- **Using PCA (principal component analysis) to reduce dimensions:**
    - We have reduced the feature dimensions of the given data to two-dimensional feature space using PCA (principal component analysis), that is, we have reduced features from 8 to 2.
    - We have used sklearn.decomposition and imported PCA from it in the code.
    - Operations done in our code while doing dimension reduction using PCA::
        - Checking the size of X_train before applying PCA
        - Fit the training set with PCA with n_components = 2.
        - Stored the PCA metrics generated to transform our validation and test data as we had to do PCA only once and generate PCA metrics using that metrics only we had to transform our validation and test dataset.
        - See the variance ratio contributed by PC1 and PC2.
        - Then we converted the fitted train set using the X-train back into Dataframe for plotting and computational purposes.
        - For observing the obtained data : Concatenate the Y_train data frame to get a combined dataframe with PC1, PC2 and the Y_train for plotting the reduced dimensional graph.

- We have used **plotly.express** library for plotting the reduced dimensional data into a 2-D place where all data points of a single class have the same color and data points from different classes have different colors.

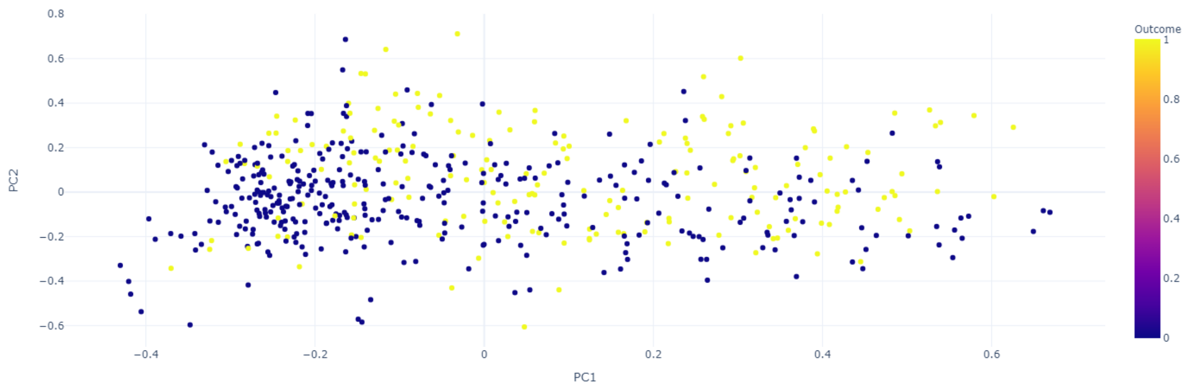
- **Result**:
  - `X_train dimension before applying PCA = (537, 8)`
  - `X_train dimension after applying PCA = (537, 2)`
  - `Explained_variance_ratio = [0.31411269 0.21153548]`

- Table after reducing to two dimensions:

| | PC1 | PC2 | Outcome |
|---|---|---|---|
| 0 | -0.095965 | 0.207448 | 0 |
| 1 | -0.254032 | 0.068722 | 1 |
| 2 | 0.192237 | -0.136709 | 1 |
| 3 | -0.153368 | -0.053380 | 1 |
| 4 | -0.334383 | -0.234876 | 0 |
| ... | ... | ... | ... |
| 532 | 0.163619 | 0.007829 | 1 |
| 533 | -0.266531 | 0.078115 | 0 |
| 534 | -0.167429 | 0.047636 | 0 |
| 535 | -0.026553 | -0.157012 | 0 |
| 536 | -0.186370 | -0.139380 | 0 |

537 rows × 3 columns

- Graph of PC1 vs PC2




*Plot of reduced dimensional data. All data points of a single class have the same color and data points from different classes have different colors.*

- **Training SVM Classifier on the reduced dimension by PCA:**

    - We have trained an SVM classifier on reduced dimensional data generated above, for which we have used the sklearn.svm and imported SVC from that.
    - Steps performed for this process:
        - create a list of hyperparameters 'C', 'gamma' and the 'degree' for a list of kernel values(linear, ploy, rbf and sigmoid) to find out the combination giving the maximum accuracy on the validation set.
        - Create a list to store the scores for different combinations.
        - generating and storing scores by iterating over the hyperparameters for different kernel types.
        - convert the score list into a dataframe for better visualization.
        - getting the row(details of parameters) with the maximum validation score.
        - storing the best model in the list for testing the test_set.
    - Validation accuracy for each combination is shown in a tabular form.
    - Some parts of the obtained result are attached :

| | kernel | C | gamma | degree | validation score |
|---|---|---|---|---|---|
| 0 | linear | 1 | 1 | 1 | 0.688312 |
| 1 | linear | 1 | 1 | 3 | 0.688312 |
| 2 | linear | 1 | 1 | 5 | 0.688312 |
| 3 | linear | 1 | 3 | 1 | 0.688312 |
| 4 | linear | 1 | 3 | 3 | 0.688312 |
| 5 | linear | 1 | 3 | 5 | 0.688312 |
| 6 | linear | 1 | 5 | 1 | 0.688312 |
| 7 | linear | 1 | 5 | 3 | 0.688312 |
| 8 | linear | 1 | 5 | 5 | 0.688312 |
| 9 | linear | 3 | 1 | 1 | 0.701299 |
| 10 | linear | 3 | 1 | 3 | 0.701299 |
| 11 | linear | 3 | 1 | 5 | 0.701299 |
| 12 | linear | 3 | 3 | 1 | 0.701299 |
| 13 | linear | 3 | 3 | 3 | 0.701299 |
| 14 | linear | 3 | 3 | 5 | 0.701299 |
| 15 | linear | 3 | 5 | 1 | 0.701299 |
| 16 | linear | 3 | 5 | 3 | 0.701299 |
| 17 | linear | 3 | 5 | 5 | 0.701299 |
| 18 | linear | 5 | 1 | 1 | 0.688312 |
| 19 | linear | 5 | 1 | 3 | 0.688312 |
| 20 | linear | 5 | 1 | 5 | 0.688312 |
| 21 | linear | 5 | 3 | 1 | 0.688312 |
| 22 | linear | 5 | 3 | 3 | 0.688312 |
| 23 | linear | 5 | 3 | 5 | 0.688312 |
| 24 | linear | 5 | 5 | 1 | 0.688312 |
| 25 | linear | 5 | 5 | 3 | 0.688312 |
| 26 | linear | 5 | 5 | 5 | 0.688312 |
| 27 | linear | 7 | 1 | 1 | 0.688312 |
| 28 | linear | 7 | 1 | 3 | 0.688312 |
| 29 | linear | 7 | 1 | 5 | 0.688312 |
| 30 | linear | 7 | 3 | 1 | 0.688312 |
| 64 | poly | 5 | 1 | 3 | 0.649351 |
| 65 | poly | 5 | 1 | 5 | 0.649351 |
| 66 | poly | 5 | 3 | 1 | 0.688312 |
| 67 | poly | 5 | 3 | 3 | 0.688312 |
| 68 | poly | 5 | 3 | 5 | 0.649351 |
| 69 | poly | 5 | 5 | 1 | 0.688312 |
| 70 | poly | 5 | 5 | 3 | 0.727273 |
| 71 | poly | 5 | 5 | 5 | 0.675325 |
| 72 | poly | 7 | 1 | 1 | 0.688312 |
| 73 | poly | 7 | 1 | 3 | 0.649351 |
| 74 | poly | 7 | 1 | 5 | 0.649351 |
| 75 | poly | 7 | 3 | 1 | 0.688312 |
| 76 | poly | 7 | 3 | 3 | 0.714286 |
| 77 | poly | 7 | 3 | 5 | 0.662338 |
| 78 | poly | 7 | 5 | 1 | 0.688312 |
| 79 | poly | 7 | 5 | 3 | 0.714286 |
| 80 | poly | 7 | 5 | 5 | 0.688312 |
| 81 | poly | 9 | 1 | 1 | 0.688312 |
| 82 | poly | 9 | 1 | 3 | 0.662338 |
| 83 | poly | 9 | 1 | 5 | 0.649351 |
| 84 | poly | 9 | 3 | 1 | 0.688312 |
| 85 | poly | 9 | 3 | 3 | 0.727273 |
| 86 | poly | 9 | 3 | 5 | 0.662338 |
| 87 | poly | 9 | 5 | 1 | 0.688312 |
| 88 | poly | 9 | 5 | 3 | 0.714286 |
| 89 | poly | 9 | 5 | 5 | 0.688312 |
| 90 | rbf | 1 | 1 | 1 | 0.688312 |
| 91 | rbf | 1 | 1 | 3 | 0.688312 |
| 92 | rbf | 1 | 1 | 5 | 0.688312 |
| 93 | rbf | 1 | 3 | 1 | 0.675325 |
| 94 | rbf | 1 | 3 | 3 | 0.675325 |
| 95 | rbf | 1 | 3 | 5 | 0.675325 |
| 96 | rbf | 1 | 5 | 1 | 0.688312 |
| 129 | rbf | 9 | 3 | 1 | 0.688312 |
| 130 | rbf | 9 | 3 | 3 | 0.688312 |
| 131 | rbf | 9 | 3 | 5 | 0.688312 |
| 132 | rbf | 9 | 5 | 1 | 0.662338 |
| 133 | rbf | 9 | 5 | 3 | 0.662338 |
| 134 | rbf | 9 | 5 | 5 | 0.662338 |
| 135 | sigmoid | 1 | 1 | 1 | 0.688312 |
| 136 | sigmoid | 1 | 1 | 3 | 0.688312 |
| 137 | sigmoid | 1 | 1 | 5 | 0.688312 |
| 138 | sigmoid | 1 | 3 | 1 | 0.688312 |
| 139 | sigmoid | 1 | 3 | 3 | 0.688312 |
| 140 | sigmoid | 1 | 3 | 5 | 0.688312 |
| 141 | sigmoid | 1 | 5 | 1 | 0.584416 |
| 142 | sigmoid | 1 | 5 | 3 | 0.584416 |
| 143 | sigmoid | 1 | 5 | 5 | 0.584416 |
| 144 | sigmoid | 3 | 1 | 1 | 0.701299 |
| 145 | sigmoid | 3 | 1 | 3 | 0.701299 |
| 146 | sigmoid | 3 | 1 | 5 | 0.701299 |
| 147 | sigmoid | 3 | 3 | 1 | 0.584416 |
| 148 | sigmoid | 3 | 3 | 3 | 0.584416 |
| 149 | sigmoid | 3 | 3 | 5 | 0.584416 |
| 150 | sigmoid | 3 | 5 | 1 | 0.597403 |
| 151 | sigmoid | 3 | 5 | 3 | 0.597403 |
| 152 | sigmoid | 3 | 5 | 5 | 0.597403 |
| 153 | sigmoid | 5 | 1 | 1 | 0.688312 |
| 154 | sigmoid | 5 | 1 | 3 | 0.688312 |
| 155 | sigmoid | 5 | 1 | 5 | 0.688312 |
| 156 | sigmoid | 5 | 3 | 1 | 0.584416 |
| 157 | sigmoid | 5 | 3 | 3 | 0.584416 |
| 158 | sigmoid | 5 | 3 | 5 | 0.584416 |
| 159 | sigmoid | 5 | 5 | 1 | 0.597403 |
| 160 | sigmoid | 5 | 5 | 3 | 0.597403 |

- We have chosen the kernel for which validation accuracy is highest
- Result is:

```
Maximum validation score is obtained for
: kernel              poly
C                     3
gamma                 5
degree                3
model                   SVC(C=3, break_ties=False,
                      cache_size=200, cla...
validation score      0.727273
Name: 61, dtype: object
```

**Accuracy on the test set based on the max validation parameters:**

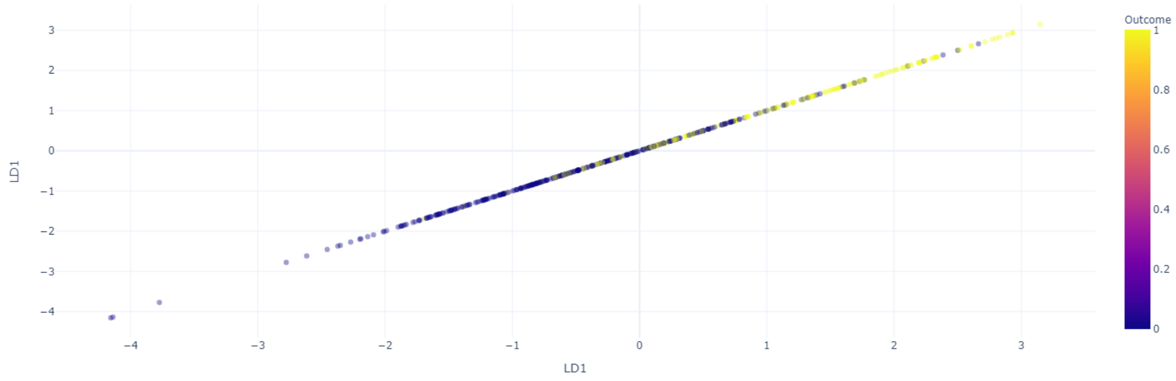
```
0.6818181818181818
```

- **Using LDA(Linear Discriminant Analysis) for reducing dimensions:**
    - We have reduced the feature dimension of the above data into a one dimensional feature space using Linear Discriminant Analysis (LDA), for which we used sklearn.discriminant_analysis and imported LDA from it.
    - We have done following things
        - Updated X_train dimensions after applying the LDA to reduce its dimensions
        - See the variance ratio contributed by LD1
        - converting the X_train_LDA back into Dataframe for computational purposes
        - Concatenate the Y_train data frame to get a combined dataframe with LD1 and the Y_train for plotting and further usage.
    - Table after reducing the dimension:

| | LD1 | Outcome |
|---|---|---|
| **0** | 0.723125 | 0 |
| **1** | -1.087577 | 1 |
| **2** | 1.454740 | 1 |
| **3** | 0.754996 | 1 |
| **4** | -0.302462 | 0 |
| **...** | ... | ... |
| **532** | -0.213644 | 1 |
| **533** | 0.170812 | 0 |
| **534** | -0.895499 | 0 |
| **535** | -1.080423 | 0 |
| **536** | -1.450775 | 0 |

537 rows × 2 columns

    - Graph after reducing dimensions:




*Plot of reduced dimensional data. All data points of a single class have the same color and data points from different classes have different colors.*

- **Training SVM Classifier on the reduced dimension by LDA:**
  - We have trained an SVM classifier on reduced dimensional data generated above, for which we have used sklearn.svm and imported SVC from that.
  - Steps performed are:
    - create a list of hyperparameters 'C', 'gamma' and the 'degree' for a list of kernel values(linear, ploy, rbf and sigmoid) to find out the combination giving the maximum accuracy on the validation set.
    - Create a list to store the scores for different combinations.
    - generating and storing scores by iterating over the hyperparameters for different kernel types.
    - convert the score list into a dataframe for better visualization.
    - getting the row(details of parameters) with the maximum validation score.
    - storing the best model in the list for testing the test_set.
  - validation accuracy for each combination is shown in a tabular form.
  - Some parts of the obtained result are attached :

| | kernel | C | gamma | degree | validation score |
|---|---|---|---|---|---|
| 0 | linear | 1 | 0.1 | 1 | 0.779221 |
| 1 | linear | 1 | 0.1 | 3 | 0.779221 |
| 2 | linear | 1 | 0.1 | 5 | 0.779221 |
| 3 | linear | 1 | 0.3 | 1 | 0.779221 |
| 4 | linear | 1 | 0.3 | 3 | 0.779221 |
| 5 | linear | 1 | 0.3 | 5 | 0.779221 |
| 6 | linear | 1 | 0.5 | 1 | 0.779221 |
| 7 | linear | 1 | 0.5 | 3 | 0.779221 |
| 8 | linear | 1 | 0.5 | 5 | 0.779221 |
| 9 | linear | 1 | 0.7 | 1 | 0.779221 |
| 10 | linear | 1 | 0.7 | 3 | 0.779221 |
| 11 | linear | 1 | 0.7 | 5 | 0.779221 |
| 12 | linear | 1 | 0.9 | 1 | 0.779221 |
| 13 | linear | 1 | 0.9 | 3 | 0.779221 |
| 14 | linear | 1 | 0.9 | 5 | 0.779221 |
| 15 | linear | 3 | 0.1 | 1 | 0.779221 |
| 16 | linear | 3 | 0.1 | 3 | 0.779221 |
| 17 | linear | 3 | 0.1 | 5 | 0.779221 |
| 18 | linear | 3 | 0.3 | 1 | 0.779221 |
| 19 | linear | 3 | 0.3 | 3 | 0.779221 |
| 20 | linear | 3 | 0.3 | 5 | 0.779221 |
| 21 | linear | 3 | 0.5 | 1 | 0.779221 |
| 22 | linear | 3 | 0.5 | 3 | 0.779221 |
| 23 | linear | 3 | 0.5 | 5 | 0.779221 |
| 24 | linear | 3 | 0.7 | 1 | 0.779221 |
| 25 | linear | 3 | 0.7 | 3 | 0.779221 |
| 26 | linear | 3 | 0.7 | 5 | 0.779221 |
| 27 | linear | 3 | 0.9 | 1 | 0.779221 |
| 28 | linear | 3 | 0.9 | 3 | 0.779221 |
| 29 | linear | 3 | 0.9 | 5 | 0.779221 |
| 30 | linear | 5 | 0.1 | 1 | 0.779221 |
| 127 | poly | 7 | 0.5 | 3 | 0.740260 |
| 128 | poly | 7 | 0.5 | 5 | 0.701299 |
| 129 | poly | 7 | 0.7 | 1 | 0.779221 |
| 130 | poly | 7 | 0.7 | 3 | 0.740260 |
| 131 | poly | 7 | 0.7 | 5 | 0.701299 |
| 132 | poly | 7 | 0.9 | 1 | 0.779221 |
| 133 | poly | 7 | 0.9 | 3 | 0.740260 |
| 134 | poly | 7 | 0.9 | 5 | 0.701299 |
| 135 | poly | 9 | 0.1 | 1 | 0.779221 |
| 136 | poly | 9 | 0.1 | 3 | 0.727273 |
| 137 | poly | 9 | 0.1 | 5 | 0.701299 |
| 138 | poly | 9 | 0.3 | 1 | 0.779221 |
| 139 | poly | 9 | 0.3 | 3 | 0.740260 |
| 140 | poly | 9 | 0.3 | 5 | 0.701299 |
| 141 | poly | 9 | 0.5 | 1 | 0.779221 |
| 142 | poly | 9 | 0.5 | 3 | 0.740260 |
| 143 | poly | 9 | 0.5 | 5 | 0.701299 |
| 144 | poly | 9 | 0.7 | 1 | 0.779221 |
| 145 | poly | 9 | 0.7 | 3 | 0.740260 |
| 146 | poly | 9 | 0.7 | 5 | 0.701299 |
| 147 | poly | 9 | 0.9 | 1 | 0.779221 |
| 148 | poly | 9 | 0.9 | 3 | 0.740260 |
| 149 | poly | 9 | 0.9 | 5 | 0.701299 |
| 150 | rbf | 1 | 0.1 | 1 | 0.792208 |
| 151 | rbf | 1 | 0.1 | 3 | 0.792208 |
| 152 | rbf | 1 | 0.1 | 5 | 0.792208 |
| 153 | rbf | 1 | 0.3 | 1 | 0.792208 |
| 154 | rbf | 1 | 0.3 | 3 | 0.792208 |
| 155 | rbf | 1 | 0.3 | 5 | 0.792208 |
| 156 | rbf | 1 | 0.5 | 1 | 0.792208 |
| 157 | rbf | 1 | 0.5 | 3 | 0.792208 |
| 158 | rbf | 1 | 0.5 | 5 | 0.792208 |
| 223 | rbf | 9 | 0.9 | 3 | 0.792208 |
| 224 | rbf | 9 | 0.9 | 5 | 0.792208 |
| 225 | sigmoid | 1 | 0.1 | 1 | 0.792208 |
| 226 | sigmoid | 1 | 0.1 | 3 | 0.792208 |
| 227 | sigmoid | 1 | 0.1 | 5 | 0.792208 |
| 228 | sigmoid | 1 | 0.3 | 1 | 0.701299 |
| 229 | sigmoid | 1 | 0.3 | 3 | 0.701299 |
| 230 | sigmoid | 1 | 0.3 | 5 | 0.701299 |
| 231 | sigmoid | 1 | 0.5 | 1 | 0.675325 |
| 232 | sigmoid | 1 | 0.5 | 3 | 0.675325 |
| 233 | sigmoid | 1 | 0.5 | 5 | 0.675325 |
| 234 | sigmoid | 1 | 0.7 | 1 | 0.675325 |
| 235 | sigmoid | 1 | 0.7 | 3 | 0.675325 |
| 236 | sigmoid | 1 | 0.7 | 5 | 0.675325 |
| 237 | sigmoid | 1 | 0.9 | 1 | 0.675325 |
| 238 | sigmoid | 1 | 0.9 | 3 | 0.675325 |
| 239 | sigmoid | 1 | 0.9 | 5 | 0.675325 |
| 240 | sigmoid | 3 | 0.1 | 1 | 0.753247 |
| 241 | sigmoid | 3 | 0.1 | 3 | 0.753247 |
| 242 | sigmoid | 3 | 0.1 | 5 | 0.753247 |
| 243 | sigmoid | 3 | 0.3 | 1 | 0.688312 |
| 244 | sigmoid | 3 | 0.3 | 3 | 0.688312 |
| 245 | sigmoid | 3 | 0.3 | 5 | 0.688312 |
| 246 | sigmoid | 3 | 0.5 | 1 | 0.675325 |
| 247 | sigmoid | 3 | 0.5 | 3 | 0.675325 |
| 248 | sigmoid | 3 | 0.5 | 5 | 0.675325 |
| 249 | sigmoid | 3 | 0.7 | 1 | 0.662338 |
| 250 | sigmoid | 3 | 0.7 | 3 | 0.662338 |
| 251 | sigmoid | 3 | 0.7 | 5 | 0.662338 |
| 252 | sigmoid | 3 | 0.9 | 1 | 0.675325 |
| 253 | sigmoid | 3 | 0.9 | 3 | 0.675325 |
| 254 | sigmoid | 3 | 0.9 | 5 | 0.675325 |

- We have chosen the kernel for which validation accuracy is highest
- Result is:

```
Maximum validation score is obtained for
: kernel                poly
C                          1
gamma                    0.1
degree                   1
model                   SVC(C=1, break_ties=False,
                        cache_size=200, cla...
validation score         0.792208
Name: 75, dtype: object
```

Accuracy on the test set based on the max validation parameters :0.7857142857142857

## Analysis:

**Checking if there are any specific difference between final test accuracy of step 3 (Applying SVM on the reduced dimension by PCA) and step 5 (Applying SVM on the reduced dimension by LDA)** :

- For step 3 (Applying SVM on the reduced dimension by PCA):
  - accuracy on the test set based on the max validation parameters: `0.6818181818181818`
    **Validation score :** `0.727273`
- For step 5 (Applying SVM on the reduced dimension by LDA):
  - accuracy on the test set based on the max validation parameters :`0.7857142857142857`
    **Validation score :** `0.792208`
- For most of the validation test scores : there isn't a significant difference between the accuracy scores.
- However, there are minor changes in test accuracy which are expected as the number of components in both the cases are different.
- The best Kernel obtained most of the time is either **poly** or **rbf**.
- We can see that there is not much significant change before(i.e. only PCA) and after performing only LDA.
- This might be because one of the components(PC1 or PC2) in particular play a major role in determining the classification, and the single component LD1 covers the maximum margin/variation in the data.
- The 2-dimension PCA identifies the two most significant dimensions. If one component does not contribute much to the maximum split, the other component can play its role and lead to a better split, Hence, while training an SVM classifier it tries to find a maximum margin split and 2.-component helps in the same.
- In case of LDA, the SVM classifier has to rely on the single component along which the variance is maximum to get the maximum margin, as The 1-component LDA does not have any such liberty as compared to the 2-component PCA described.
- However, we can see that the accuracy values in both the cases are nearly the same.

## Conclusion :

- In this assignment, we have learned to use several dimensionality reduction techniques(using PCA and LDA) to reduce the feature dimension of a data and understand the impact of dimensionality reduction of data on accuracy.