**Dimensionality Reduction**

**Dimensionality reduction refers to techniques for reducing the number of input variables in training data.**

**When dealing with high dimensional data, it is often useful to reduce the dimensionality by projecting the data to a lower dimensional subspace which captures the “essence” of the data. This is called dimensionality reduction.**

**High-dimensionality might mean hundreds, thousands, or even millions of input variables.**

**Fewer input dimensions often mean correspondingly fewer parameters or a simpler structure in the machine learning model, referred to as degrees of freedom. A model with too many degrees of freedom is likely to overfit the training dataset and therefore may not perform well on new data.**

**It is desirable to have simple models that generalize well, and in turn, input data with few input variables. This is particularly true for linear models where the number of inputs and the degrees of freedom of the model are often closely related.**

**The fundamental reason for the curse of dimensionality is that high-dimensional functions have the potential to be much more complicated than low-dimensional ones, and that those complications are harder to discern. The only way to beat the curse is to incorporate knowledge about the data that is correct.**

**Methods of Dimensionality Reduction:-**

* **Principal Component Analysis (PCA)**
* **Linear Discriminant Analysis (LDA)**
* **Generalized Discriminant Analysis (GDA)**

**Principle Component Analysis(PCA):-**

**Principal component analysis (PCA) simplifies the complexity in high-dimensional data while retaining trends and patterns. It does this by transforming the data into fewer dimensions, which act as summaries of features. High-dimensional data are very common in biology and arise when multiple features, such as expression of many genes, are measured for each sample. This type of data presents several challenges that PCA mitigates: computational expense and an increased error rate due to multiple test correction when testing each feature for association with an outcome. PCA is an unsupervised learning method and is similar to clustering—it finds patterns without reference to prior knowledge about whether the samples come from different treatment groups or have phenotypic differences. PCA reduces data by geometrically projecting them onto lower dimensions called principal components (PCs), with the goal of finding the best summary of the data using a limited number of PCs. The first PC is chosen to minimize the total distance between the data and their projection onto the PC . By minimizing this distance, we also maximize the variance of the projected points, σ2 . The second (and subsequent) PCs are selected similarly, with the additional requirement that they be uncorrelated with all previous PCs. For example, projection onto PC1 is uncorrelated with projection onto PC2, and we can think of the PCs as geometrically orthogonal. This requirement of no correlation means that the maximum number of PCs possible is either the number of samples or the number of features, whichever is smaller. The PC selection process has the effect of maximizing the correlation (r2) between data and their projection and is equivalent to carrying out multiple linear regression on the projected data against each variable of the original data. For example, the projection onto PC2 has maximum *r*2 when used in multiple regression with PC1.**

**Linear Discriminant Analysis(LDA):-**

**Logistic regression is a classification algorithm traditionally limited to only two-class classification problems.**

**If you have more than two classes then Linear Discriminant Analysis is the preferred linear classification technique.**

**In this post you will discover the Linear Discriminant Analysis (LDA) algorithm for classification predictive modeling problems. After reading this post you will know:**

**The limitations of logistic regression and the need for linear discriminant analysis.**

**The representation of the model that is learned from data and can be saved to file.**

**How the model is estimated from your data.**

**How to make predictions from a learned LDA model.**

**How to prepare your data to get the most from the LDA model.**

**This post is intended for developers interested in applied machine learning, how the models work and how to use them well. As such no background in statistics or linear algebra is required, although it does help if you know about the mean and variance of a distribution.**

**LDA is a simple model in both preparation and application. There is some interesting statistics behind how the model is setup and how the prediction equation is derived, but is not covered in this post.**

**Generalized Discriminant Analysis(GDA):-**

**Linear discriminant analysis (LDA), normal discriminant analysis (NDA), or discriminant function analysis is a generalization of Fisher's linear discriminant, a method used in statistics and other fields, to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification.**

**LDA is closely related to analysis of variance (ANOVA) and regression analysis, which also attempt to express one dependent variable as a linear combination of other features or measurements.However, ANOVA uses categorical independent variables and a continuous dependent variable, whereas discriminant analysis has continuous independent variables and a categorical dependent variable (*i.e.* the class label).Logistic regression and probit regression are more similar to LDA than ANOVA is, as they also explain a categorical variable by the values of continuous independent variables. These other methods are preferable in applications where it is not reasonable to assume that the independent variables are normally distributed, which is a fundamental assumption of the LDA method.**

**LDA is also closely related to principal component analysis (PCA) and factor analysis in that they both look for linear combinations of variables which best explain the data.LDA explicitly attempts to model the difference between the classes of data. PCA, in contrast, does not take into account any difference in class, and factor analysis builds the feature combinations based on differences rather than similarities. Discriminant analysis is also different from factor analysis in that it is not an interdependence technique: a distinction between independent variables and dependent variables (also called criterion variables) must be made.**

**LDA works when the measurements made on independent variables for each observation are continuous quantities. When dealing with categorical independent variables, the equivalent technique is discriminant correspondence analysis.**

**Discriminant analysis is used when groups are known a priori (unlike in cluster analysis). Each case must have a score on one or more quantitative predictor measures, and a score on a group measure.In simple terms, discriminant function analysis is classification - the act of distributing things into groups, classes or categories of the same type.**

**Implementation:-**

**Import Libraries**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**Here we declare the libraries**

**DataSet Imported**

**dataset = pd.read\_csv('Wine.csv')**

**X = dataset.iloc[:, :-1].values**

**y = dataset.iloc[:, -1].values**

**Here we took the dataset and splited into x and y.**

**Train Test Split**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)**

**Here we split the dataset into xtrain,xtest,ytrain,ytest and we took the test size 0.2 took the random state=0**

**Feature Scalling**

**Here we did the feature scalling to standardized the dataset.**

**from sklearn.preprocessing import StandardScaler**

**sc = StandardScaler()**

**X\_train = sc.fit\_transform(X\_train)**

**X\_test = sc.transform(X\_test)**

**Apply PCA**

**Principle Component Analysis is use to reduced the dimensionality**

**Of the dataset.**

**from sklearn.decomposition import PCA**

**pca = PCA(n\_components = 2)**

**X\_train = pca.fit\_transform(X\_train)**

**X\_test = pca.transform(X\_test)**

**Logistic Regression Model on Training Set**

**After PCA we use the logistic regression**

**from sklearn.linear\_model import LogisticRegression**

**classifier = LogisticRegression(random\_state = 0)**

**classifier.fit(X\_train, y\_train)**

**Making the confusion matrix**

**Here we check the accuracy score**

**from sklearn.metrics import confusion\_matrix, accuracy\_score**

**y\_pred = classifier.predict(X\_test)**

**cm = confusion\_matrix(y\_test, y\_pred)**

**print(cm)**

**accuracy\_score(y\_test, y\_pred)**

**Visualizing the training set result:-**

**from matplotlib.colors import ListedColormap**

**X\_set, y\_set = X\_train, y\_train**

**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, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),**

**alpha = 0.75, cmap = ListedColormap(('red', 'green', 'blue')))**

**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', 'blue'))(i), label = j)**

**plt.title('Logistic Regression (Training set)')**

**plt.xlabel('PC1')**

**plt.ylabel('PC2')**

**plt.legend()**

**plt.show()**

**Visualizing test set result:-**

**from matplotlib.colors import ListedColormap**

**X\_set, y\_set = X\_test, y\_test**

**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, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),**

**alpha = 0.75, cmap = ListedColormap(('red', 'green', 'blue')))**

**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', 'blue'))(i), label = j)**

**plt.title('Logistic Regression (Test set)')**

**plt.xlabel('PC1')**

**plt.ylabel('PC2')**

**plt.legend()**

**plt.show()**