Machine Learning Assignment

# Ravi Shekhar (ID: - 2017HT13059)

# Accuracy Achieved – 60.244%

I have applied several Machine Learning algorithm. The algorithms used are as follows: -

1. Random Forest
2. Decision Tree
3. K-Nearest Neighbor
4. Naive Bayes Bernoulli
5. Naive Bayes Gaussian
6. SVM LinearSVC
7. SVM rbf
8. SVM poly
9. SVM linear
10. Logistic Regression
11. K-Means
12. Gradient Boosting Classifier
13. Neural Network

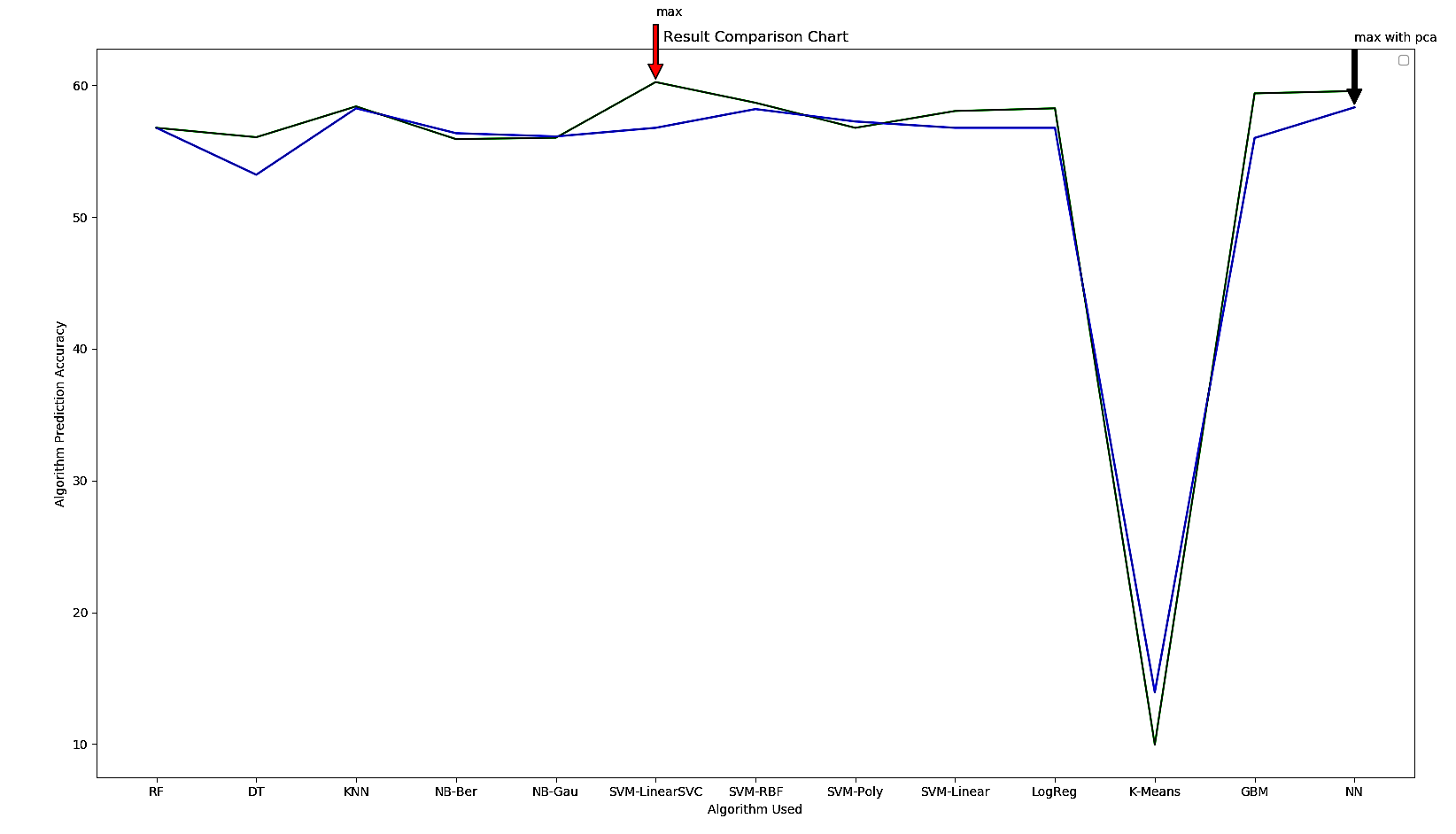
These algorithms are used on processed data, both, with and without PCA decomposition. At the end, all **the accuracy was compared and converted to % format by multiplying 100.** The result is shown below. At the end, the graph shows the accuracy of each model without PCA decomposition in Green and with PCA decomposition in Blue.

The best accuracy obtained in both categories are: -

1. Without PCA decomposition: - SVM – Linear SVC (60.244%)
2. With PCA decomposition: - Neural Network with 3 layer each with 40 units. (59.56%)

**The best predictor was SVM with Linear SVC with accuracy of 0.60244 or >60% accuracy.**

The Graph below compares the accuracy achieved by various ML algorithms. The Green color line is without PCA decomposition while the green color after the PCA decomposition.



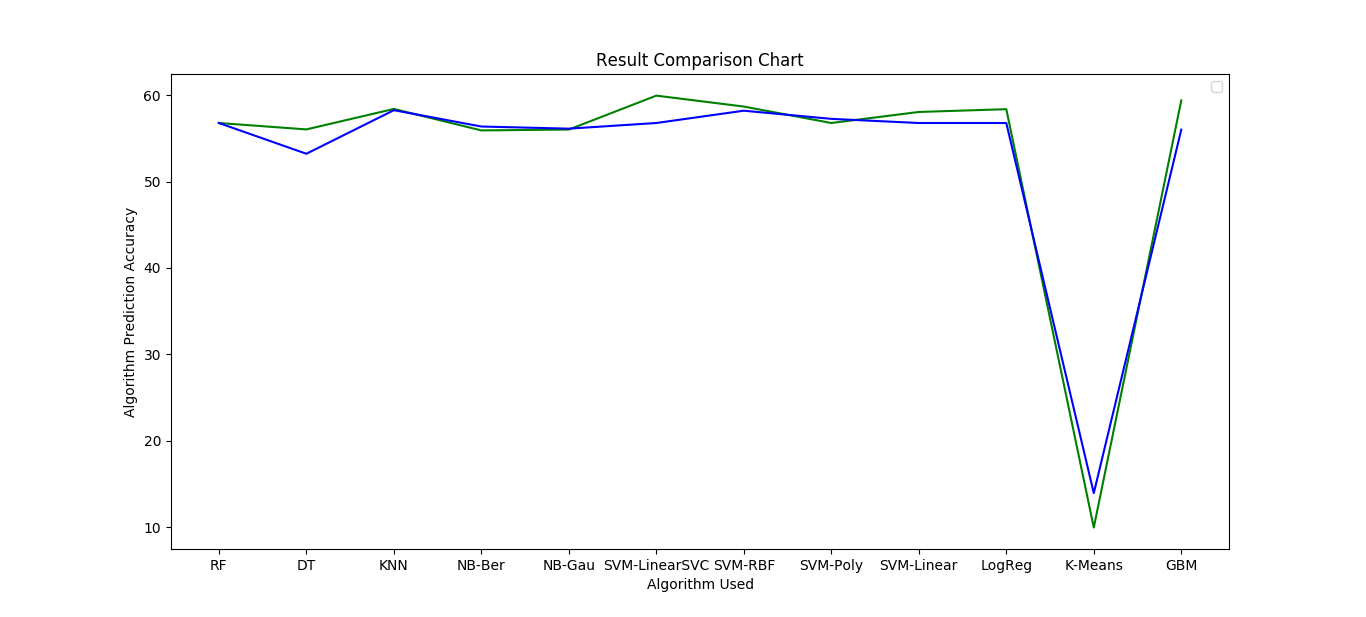
**Results: -**

|  |  |  |
| --- | --- | --- |
| **Without PCA Decomposition** | **With PCA Decomposition** |  |
| Random Forest :  Accuracy: 56.7712874164487  Classification Report:  precision recall f1-score support  0 0.00 0.00 0.00 2975  1 0.57 1.00 0.72 3907  avg / total 0.32 0.57 0.41 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  Decision Tree :  Accuracy: 56.05928509154315  Classification Report:  precision recall f1-score support  0 0.49 0.53 0.51 2975  1 0.62 0.58 0.60 3907  avg / total 0.56 0.56 0.56 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  K-Nearest Neighbor :  Accuracy: **58.39872130194711**  Classification Report:  precision recall f1-score support  0 0.53 0.33 0.40 2975  1 0.60 0.78 0.68 3907  avg / total 0.57 0.58 0.56 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  Naive Bayes Bernoulli:  Accuracy: 55.91397849462365  Classification Report:  precision recall f1-score support  0 0.49 0.52 0.51 2975  1 0.62 0.59 0.60 3907  avg / total 0.56 0.56 0.56 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  Naive Bayes Gaussian :  Accuracy: 56.015693112467304  Classification Report:  precision recall f1-score support  0 0.49 0.52 0.51 2975  1 0.62 0.59 0.60 3907  avg / total 0.56 0.56 0.56 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  SVM LinearSVC :  Accuracy: **60.244115082824756**  Classification Report:  precision recall f1-score support  0 0.54 0.57 0.56 2975  1 0.66 0.62 0.64 3907  avg / total 0.61 0.60 0.60 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  SVM rbf :  Accuracy: **58.674803836094156**  Classification Report:  precision recall f1-score support  0 0.54 0.32 0.40 2975  1 0.60 0.79 0.68 3907  avg / total 0.58 0.59 0.56 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  SVM poly :  Accuracy: 56.7712874164487  Classification Report:  precision recall f1-score support  0 0.00 0.00 0.00 2975  1 0.57 1.00 0.72 3907  avg / total 0.32 0.57 0.41 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  SVM linear :  Accuracy: **58.04998546934031**  Classification Report:  precision recall f1-score support  0 0.53 0.27 0.36 2975  1 0.59 0.82 0.69 3907  avg / total 0.57 0.58 0.55 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  Logistic Regression :  Accuracy: **58.25341470502761**  Classification Report:  precision recall f1-score support  0 0.53 0.29 0.37 2975  1 0.60 0.81 0.69 3907  avg / total 0.57 0.58 0.55 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  K-Means:  Accuracy: 9.953501888985759  Classification Report:  precision recall f1-score support  0 0.46 0.02 0.05 2975  1 0.59 0.16 0.25 3907  2 0.00 0.00 0.00 0  3 0.00 0.00 0.00 0  4 0.00 0.00 0.00 0  5 0.00 0.00 0.00 0  6 0.00 0.00 0.00 0  7 0.00 0.00 0.00 0  8 0.00 0.00 0.00 0  9 0.00 0.00 0.00 0  avg / total 0.53 0.10 0.16 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  Gradient Boosting Classifier:  Accuracy: **59.386806160999704**  Classification Report:  precision recall f1-score support  0 0.54 0.39 0.45 2975  1 0.62 0.75 0.68 3907  avg / total 0.58 0.59 0.58 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  Neural Network:  Accuracy: **59.561174077303114**  Classification Report:  precision recall f1-score support  0 0.54 0.41 0.47 2975  1 0.62 0.74 0.67 3907  avg / total 0.59 0.60 0.58 6882 | Random Forest :  Accuracy: 56.7712874164487  Classification Report:  precision recall f1-score support  0 0.00 0.00 0.00 2975  1 0.57 1.00 0.72 3907  avg / total 0.32 0.57 0.41 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  Decision Tree :  Accuracy: 53.211275791920954  Classification Report:  precision recall f1-score support  0 0.45 0.39 0.42 2975  1 0.58 0.64 0.61 3907  avg / total 0.52 0.53 0.53 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  K-Nearest Neighbor :  Accuracy: **58.25341470502761**  Classification Report:  precision recall f1-score support  0 0.53 0.31 0.39 2975  1 0.60 0.79 0.68 3907  avg / total 0.57 0.58 0.56 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  Naive Bayes Bernoulli :  Accuracy: 56.36442894507411  Classification Report:  precision recall f1-score support  0 0.49 0.42 0.45 2975  1 0.60 0.68 0.64 3907  avg / total 0.56 0.56 0.56 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  Naive Bayes Gaussian :  Accuracy: 56.11740773031095  Classification Report:  precision recall f1-score support  0 0.49 0.49 0.49 2975  1 0.61 0.62 0.62 3907  avg / total 0.56 0.56 0.56 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  SVM LinearSVC :  Accuracy: 56.7712874164487  Classification Report:  precision recall f1-score support  0 0.00 0.00 0.00 2975  1 0.57 1.00 0.72 3907  avg / total 0.32 0.57 0.41 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  SVM rbf :  Accuracy: **58.19529206625981**  Classification Report:  precision recall f1-score support  0 0.53 0.26 0.35 2975  1 0.59 0.83 0.69 3907  avg / total 0.57 0.58 0.54 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  SVM poly :  Accuracy: 57.250799186283054  Classification Report:  precision recall f1-score support  0 0.58 0.04 0.08 2975  1 0.57 0.98 0.72 3907  avg / total 0.57 0.57 0.44 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  SVM linear :  Accuracy: 56.7712874164487  Classification Report:  precision recall f1-score support  0 0.00 0.00 0.00 2975  1 0.57 1.00 0.72 3907  avg / total 0.32 0.57 0.41 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  Logistic Regression :  Accuracy: 56.7712874164487  Classification Report:  precision recall f1-score support  0 0.00 0.00 0.00 2975  1 0.57 1.00 0.72 3907  avg / total 0.32 0.57 0.41 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  K-Means:  Accuracy: 13.934902644580063  Classification Report:  precision recall f1-score support  0 0.47 0.10 0.17 2975  1 0.52 0.17 0.25 3907  2 0.00 0.00 0.00 0  3 0.00 0.00 0.00 0  4 0.00 0.00 0.00 0  5 0.00 0.00 0.00 0  6 0.00 0.00 0.00 0  7 0.00 0.00 0.00 0  8 0.00 0.00 0.00 0  9 0.00 0.00 0.00 0  avg / total 0.50 0.14 0.22 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  Gradient Boosting Classifier:  Accuracy: 56.001162452775354  Classification Report:  precision recall f1-score support  0 0.49 0.32 0.38 2975  1 0.59 0.75 0.66 3907  avg / total 0.54 0.56 0.54 6882  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  Neural Network:  Accuracy: **58.326068003487364**  Classification Report:  precision recall f1-score support  0 0.54 0.26 0.35 2975  1 0.60 0.83 0.69 3907  avg / total 0.57 0.58 0.54 6882 |  |

|  |
| --- |
| **# -\*- coding: utf-8 -\*-**  **"""**  **Spyder Editor**  **Name : Ravi Shekhar Singh**  **"""**  **import warnings**  **warnings.filterwarnings('ignore')**  **import pandas as pd**  **import numpy as np**  **from math import sqrt**  **from sklearn.preprocessing import MinMaxScaler**  **from sklearn.cross\_validation import train\_test\_split**  **from sklearn.metrics import confusion\_matrix**  **from sklearn.metrics import accuracy\_score**  **from sklearn.metrics import classification\_report**  **seed = 110**  **np.random.seed(seed)**  **algo\_num = []**  **algo\_result = []**  **algo\_result\_with\_pca = []**  **df = pd.read\_csv("data.csv",header=None)**  **# process the data before applying ML algorithm**  **first\_quartile = df.quantile(0.25)**  **third\_quartile = df.quantile(0.75)**  **IQR = third\_quartile - first\_quartile**  **clean\_data = df[~((df < (first\_quartile - 1.5 \* IQR)) | (df > (third\_quartile + 1.5 \* IQR))).any(axis=1)]**  **X = df.ix[:,0:40].values**  **Y = df.ix[:,41].values**  **X\_tr, X\_tst, Y\_tr, Y\_tst = train\_test\_split (X, Y, test\_size=0.33, random\_state=24)**  **scaling = MinMaxScaler(feature\_range=(0, 1))**  **X\_train = scaling.fit\_transform(X\_tr)**  **X\_test = scaling.fit\_transform(X\_tst)**  **from sklearn.ensemble import RandomForestClassifier**  **clf = RandomForestClassifier(max\_depth=2, random\_state=0)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("Random Forest : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **algo\_num.append("RF")**  **algo\_result.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.tree import DecisionTreeClassifier**  **clf = DecisionTreeClassifier()**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("Decision Tree : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **algo\_num.append("DT")**  **algo\_result.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.neighbors import KNeighborsClassifier**  **clf = KNeighborsClassifier(n\_neighbors=int(sqrt(df.shape[0])))**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("K-Nearest Neighbor : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **algo\_num.append("KNN")**  **algo\_result.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.naive\_bayes import BernoulliNB**  **clf = BernoulliNB()**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("Naive Bayes Bernoulli: \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **algo\_num.append("NB-Ber")**  **algo\_result.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.naive\_bayes import GaussianNB**  **clf = GaussianNB()**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("Naive Bayes Gaussian : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **algo\_num.append("NB-Gau")**  **algo\_result.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.svm import LinearSVC**  **clf = LinearSVC(C=5.0, dual=True, tol=1e-06)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("SVM LinearSVC : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **algo\_num.append("SVM-LinearSVC")**  **algo\_result.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.svm import SVC**  **clf = SVC(kernel='rbf', gamma=0.7, C=1.0)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("SVM rbf : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\nClassification Report: \n", classification\_report(Y\_tst, predicted))**  **algo\_num.append("SVM-RBF")**  **algo\_result.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **clf = SVC(kernel='poly', degree=3, C=1.0)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("SVM poly : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **algo\_num.append("SVM-Poly")**  **algo\_result.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **clf = SVC(kernel='linear', C=1.0)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("SVM linear : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **algo\_num.append("SVM-Linear")**  **algo\_result.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.linear\_model import LogisticRegression**  **clf = LogisticRegression(solver = 'lbfgs')**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("Logistic Regression : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **algo\_num.append("LogReg")**  **algo\_result.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.cluster import KMeans**  **clf = KMeans(n\_clusters=10, random\_state=0)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("K-Means: \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **algo\_num.append("K-Means")**  **algo\_result.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.ensemble import GradientBoostingClassifier**  **clf = GradientBoostingClassifier(n\_estimators=100, learning\_rate=1.0, max\_depth=1, random\_state=0)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("Gradient Boosting Classifier: \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **algo\_num.append("GBM")**  **algo\_result.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.neural\_network import MLPClassifier**  **clf = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden\_layer\_sizes=(40,40,40), random\_state=1)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("Neural Network: \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **algo\_num.append("NN")**  **algo\_result.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **###############################################################################**  **'''''''''''''''''''''' After PCA Dicomposition '''''''''''''''''''''''''''**  **###############################################################################**  **from sklearn.decomposition import PCA**  **X\_tr, X\_tst, Y\_tr, Y\_tst = train\_test\_split (X, Y, test\_size=0.33, random\_state=24)**  **#Scaling for featrure normalization for setting values between 0 & 1**  **scaling = MinMaxScaler(feature\_range=(0, 1))**  **# Minmax scaling of training & test data**  **X\_train = scaling.fit\_transform(X\_tr)**  **X\_test = scaling.fit\_transform(X\_tst)**  **pca = PCA(0.95)**  **X\_train = pca.fit(X\_train).transform(X\_train)**  **X\_test = pca.fit(X\_test).transform(X\_test)**  **from sklearn.ensemble import RandomForestClassifier**  **clf = RandomForestClassifier(max\_depth=2, random\_state=0)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("Random Forest : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **#algo\_num.append("RF")**  **algo\_result\_with\_pca.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.tree import DecisionTreeClassifier**  **clf = DecisionTreeClassifier()**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("Decision Tree : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **#algo\_num.append("DT")**  **algo\_result\_with\_pca.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.neighbors import KNeighborsClassifier**  **clf = KNeighborsClassifier(n\_neighbors=int(sqrt(df.shape[0])))**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("K-Nearest Neighbor : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **#algo\_num.append("KNN")**  **algo\_result\_with\_pca.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.naive\_bayes import BernoulliNB**  **clf = BernoulliNB()**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("Naive Bayes Bernoulli : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **#algo\_num.append("NB-Ber")**  **algo\_result\_with\_pca.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.naive\_bayes import GaussianNB**  **clf = GaussianNB()**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("Naive Bayes Gaussian : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **#algo\_num.append("NB-Gau")**  **algo\_result\_with\_pca.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.svm import LinearSVC**  **clf = LinearSVC(C=5.0)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("SVM LinearSVC : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **#algo\_num.append("SVM-LinearSVC")**  **algo\_result\_with\_pca.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.svm import SVC**  **clf = SVC(kernel='rbf', gamma=0.7, C=1.0)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("SVM rbf : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\nClassification Report: \n", classification\_report(Y\_tst, predicted))**  **#algo\_num.append("SVM-RBF")**  **algo\_result\_with\_pca.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **clf = SVC(kernel='poly', degree=3, C=1.0)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("SVM poly : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **#algo\_num.append("SVM-Poly")**  **algo\_result\_with\_pca.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **clf = SVC(kernel='linear', C=1.0)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("SVM linear : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **#algo\_num.append("SVM-Linear")**  **algo\_result\_with\_pca.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.linear\_model import LogisticRegression**  **clf = LogisticRegression(solver = 'lbfgs')**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("Logistic Regression : \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **#algo\_num.append("LogReg")**  **algo\_result\_with\_pca.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.cluster import KMeans**  **clf = KMeans(n\_clusters=10, random\_state=0)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("K-Means: \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **#algo\_num.append("K-Means")**  **algo\_result\_with\_pca.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.ensemble import GradientBoostingClassifier**  **clf = GradientBoostingClassifier(n\_estimators=100, learning\_rate=1.0, max\_depth=1, random\_state=0)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("Gradient Boosting Classifier: \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **#algo\_num.append("GBM")**  **algo\_result\_with\_pca.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **from sklearn.neural\_network import MLPClassifier**  **clf = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden\_layer\_sizes=(40,40,40), random\_state=1)**  **clf.fit(X\_train, Y\_tr)**  **predicted = clf.predict(X\_test)**  **print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")**  **print("Neural Network: \n")**  **conf\_mat = confusion\_matrix(Y\_tst, predicted)**  **#print(conf\_mat)**  **print("Accuracy: ", 100\*accuracy\_score(Y\_tst, predicted))**  **print("\n Classification Report: \n", classification\_report(Y\_tst, predicted))**  **#algo\_num.append("NN")**  **algo\_result\_with\_pca.append(100\*accuracy\_score(Y\_tst, predicted))**  **print("")**  **# Plot the graph to compare all the accuracy**  **import matplotlib.pyplot as plt**  **plt.plot(algo\_num, algo\_result, color='green')**  **plt.plot(algo\_num, algo\_result\_with\_pca, color='blue')**  **plt.xlabel('Algorithm Used')**  **plt.ylabel('Algorithm Prediction Accuracy')**  **plt.title('Result Comparison Chart')**  **ymax = max(algo\_result)**  **xpos = algo\_result.index(ymax)**  **xmax = algo\_num[xpos]**  **plt.annotate('max', xy=(xmax, ymax), xytext=(xmax, ymax+5),**  **arrowprops=dict(facecolor='red', shrink=0.05),**  **)**  **ymax = max(algo\_result\_with\_pca)**  **xpos = algo\_result\_with\_pca.index(ymax)**  **xmax = algo\_num[xpos]**  **plt.annotate('max with pca', xy=(xmax, ymax), xytext=(xmax, ymax+5),**  **arrowprops=dict(facecolor='black', shrink=0.05),**  **)**  **plt.legend()**  **plt.show()** |

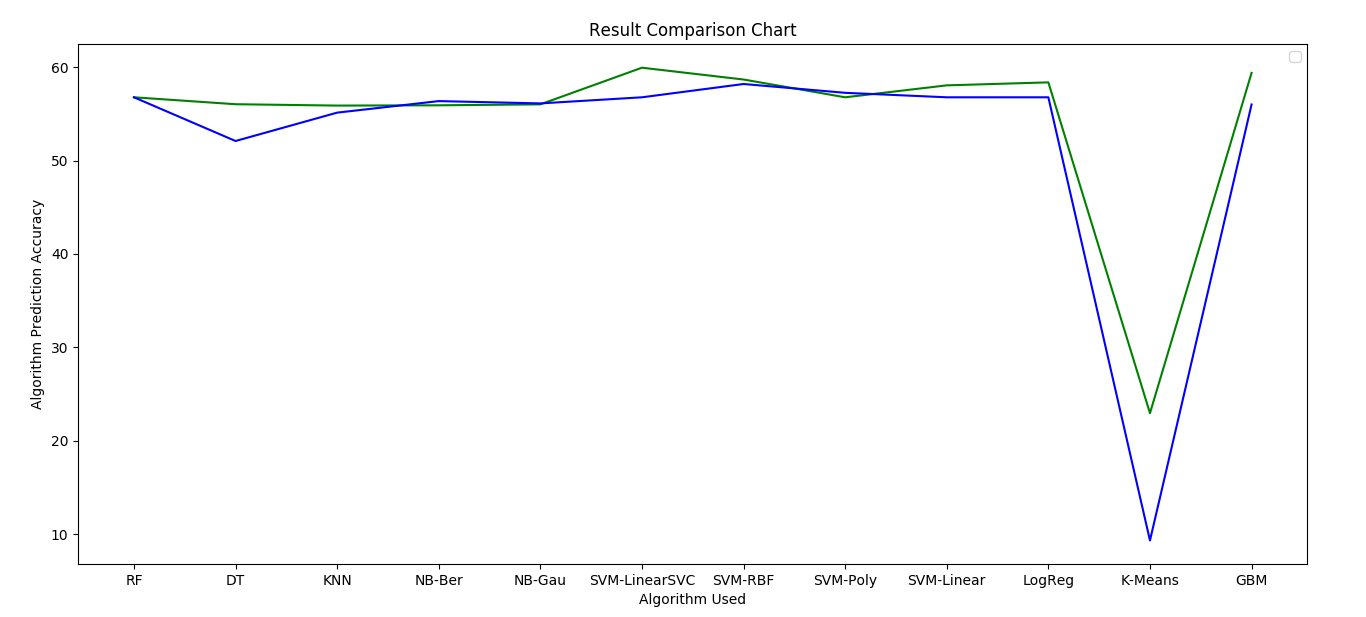
**With KNN – square root of N**

Using No of Neighbors as square root of N, the Accuracy improved but it is still less than LinearSVC.

****

**High KNN High K-Means**

Using High values for No of Neighbors in KNN and higher value in K-Means improved results but still K-Means fared most poorly.

****