1

October 16, 2020

Question 1)

a)

[1]:
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from scipy.io import loadmat
import warnings
warnings.filterwarnings('ignore')
```

Import and read in the iris dataset

[2]:
```
iris = load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
convert_species = np.vectorize(lambda x : "setosa" if x==0 else ("versicolor"␣
 ↪if x==1 else "virginica"))
iris_unnamed = iris_df
iris_df["target"] = convert_species(iris.target)
iris_unnamed["target"] = iris.target
```

[3]:
```
iris_unnamed.head()
```

[3]:
```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
```

```
3       0
4       0
```

[4]:
```
iris_df.head()
```

[4]:
```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0
```

Import and read in the indian pines dataset

[5]:
```
indian = loadmat(os.path.join(os.getcwd(), "indianR.mat"))
data = np.array(indian["X"]).T
targets = np.array(indian["gth"])[0]
indian_df = pd.DataFrame(data=data)
indian_df["target"] = targets
```

[6]:
```
np.unique(indian_df["target"], return_counts=True)
```

[6]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16],
       dtype=uint8),
 array([10776,    46,  1428,   830,   237,   483,   730,    28,   478,
           20,   972,  2455,   593,   205,  1265,   386,    93],
       dtype=int64))
```

[7]:
```
iris_df.head()
```

[7]:
```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
```

```
2        0
3        0
4        0
```

[8]:
```
indian_df.describe()
```

[8]:
```
                  0             1             2             3             4  \
count  21025.000000  21025.000000  21025.000000  21025.000000  21025.000000
mean    3341.313103   4091.957765   4277.523662   4170.075672   4516.685375
std      212.254014    227.885060    257.819760    280.381316    346.041777
min     2632.000000   3477.000000   3649.000000   3578.000000   3840.000000
25%     3179.000000   3889.000000   4066.000000   3954.000000   4214.000000
50%     3343.000000   4107.000000   4237.000000   4126.000000   4478.000000
75%     3510.000000   4247.000000   4479.000000   4350.000000   4772.000000
max     4536.000000   5744.000000   6361.000000   6362.000000   7153.000000

                  5             6             7             8             9  \
count  21025.000000  21025.000000  21025.000000  21025.000000  21025.000000
mean    4790.607134   4848.334269   4714.739073   4668.916528   4439.259453
std      414.386445    469.244664    491.731609    533.233484    539.486030
min     4056.000000   4004.000000   3865.000000   3775.000000   3560.000000
25%     4425.000000   4421.000000   4263.000000   4173.000000   3940.000000
50%     4754.000000   4808.000000   4666.000000   4632.000000   4404.000000
75%     5093.000000   5198.000000   5100.000000   5084.000000   4860.000000
max     7980.000000   8284.000000   8128.000000   8194.000000   7928.000000

       …           193           194           195           196  \
count  …  21025.000000  21025.000000  21025.000000  21025.000000
mean   …   1060.238383   1063.299215   1050.072913   1040.227539
std    …     38.694237     41.906116     34.232348     26.447967
min    …    999.000000    999.000000    993.000000    990.000000
25%    …   1024.000000   1024.000000   1019.000000   1016.000000
50%    …   1052.000000   1054.000000   1043.000000   1033.000000
75%    …   1098.000000   1104.000000   1083.000000   1066.000000
max    …   1289.000000   1315.000000   1258.000000   1201.000000

                197           198           199           200           201  \
count  21025.000000  21025.000000  21025.000000  21025.000000  21025.000000
mean    1043.391011   1030.223496   1015.600856   1008.513864   1006.791011
std       29.788944     20.872378     11.437696      7.052013      6.995153
min      992.000000    989.000000    986.000000    981.000000    980.000000
25%     1016.000000   1012.000000   1006.000000   1004.000000   1003.000000
50%     1037.000000   1026.000000   1014.000000   1009.000000   1005.000000
75%     1072.000000   1050.000000   1024.000000   1014.000000   1010.000000
max     1245.000000   1167.000000   1076.000000   1037.000000   1034.000000

             target
```

```
count  21025.000000
mean       4.224923
std        5.281972
min        0.000000
25%        0.000000
50%        0.000000
75%       10.000000
max       16.000000

[8 rows x 203 columns]
```

[9]:
```python
indexDrop = indian_df[indian_df["target"] == 0].index
indian_df.drop(indexDrop, inplace=True)
indian_df.reset_index(inplace=True)
```

[10]:
```python
indian_df.describe()
```

[10]:
```
              index             0             1             2             3  \
count  10249.000000  10249.000000  10249.000000  10249.000000  10249.000000
mean    9322.297200   3382.435847   4152.747292   4355.246073   4257.888184
std     5218.036087    210.366658    224.041937    256.879124    281.800019
min        0.000000   2640.000000   3491.000000   3739.000000   3601.000000
25%     5016.000000   3196.000000   3990.000000   4137.000000   4029.000000
50%     8723.000000   3358.000000   4130.000000   4394.000000   4270.000000
75%    13560.000000   3526.000000   4264.000000   4510.000000   4431.000000
max    20214.000000   4536.000000   5526.000000   6080.000000   6139.000000

                  4             5             6             7             8  \
count  10249.000000  10249.000000  10249.000000  10249.000000  10249.000000
mean    4630.182750   4930.287931   5010.664357   4887.468241   4859.082349
std      348.119919    417.197677    473.771399    497.665315    539.681324
min     3911.000000   4123.000000   4138.000000   3997.000000   3883.000000
25%     4327.000000   4536.000000   4560.000000   4415.000000   4337.000000
50%     4661.000000   4993.000000   5091.000000   4976.000000   4958.000000
75%     4881.000000   5202.000000   5335.000000   5234.000000   5227.000000
max     7070.000000   7809.000000   8000.000000   7980.000000   8106.000000

       …           193           194           195           196  \
count  …  10249.000000  10249.000000  10249.000000  10249.000000
mean   …   1078.458874   1083.158845   1066.017855   1052.516928
std    …     38.994331     42.108992     34.516822     26.723702
min    …   1000.000000    999.000000    993.000000    995.000000
25%    …   1039.000000   1040.000000   1031.000000   1026.000000
50%    …   1093.000000   1099.000000   1078.000000   1062.000000
75%    …   1110.000000   1117.000000   1094.000000   1074.000000
max    …   1163.000000   1180.000000   1141.000000   1110.000000
```

```
                197           198           199           200           201  \
count  10249.000000  10249.000000  10249.000000  10249.000000  10249.000000
mean    1057.192799   1039.704459   1020.304322   1010.365499   1008.517416
std       30.055980     21.030831     11.539938      7.034534      7.080302
min      992.000000    989.000000    986.000000    985.000000    985.000000
25%     1027.000000   1020.000000   1010.000000   1005.000000   1004.000000
50%     1067.000000   1045.000000   1022.000000   1010.000000   1009.000000
75%     1081.000000   1055.000000   1029.000000   1015.000000   1014.000000
max     1130.000000   1088.000000   1048.000000   1037.000000   1034.000000

             target
count  10249.000000
mean       8.667089
std        4.327987
min        1.000000
25%        5.000000
50%       10.000000
75%       11.000000
max       16.000000

[8 rows x 204 columns]
```

Both indian and iris datasets have successfully been loaded. Now, we need to setup PCA and LDA

First is PCA. PCA needs to scale the data first and then deconstruct the data into its principal components

[11]:
```
def scale(df, scaler="MinMax"):
    inputs = df.iloc[:, :-1].to_numpy()
    if scaler == "MinMax":
        scale = MinMaxScaler()
    elif scaler == "Standard":
        scale = StandardScaler()
    else:
        raise ValueError(f"Unsupported scaler {scaler}")
    scale.fit(inputs.astype(float))
    inputs = scale.transform(inputs)

    scaled_df = pd.DataFrame(data=inputs)

    scaled_df["target"] = df["target"]

    return scaled_df
```

[12]:
```
def fit_pca(df, **kwargs):
    n_components = kwargs.get("n_components", len(df.columns)-1)
    plot_variance = kwargs.get("plot_variance", True)

```

```
    inputs = df.iloc[:, :-1].to_numpy()

    pca = PCA(n_components=n_components)
    pc = pca.fit(inputs)

    return pc
```

[13]:
```
def transform_pca(df, pc):
    inputs = df.iloc[:, :-1].to_numpy()
    transformed_df = pd.DataFrame(data=pc.transform(inputs))
    transformed_df.columns = [*map(lambda y : f"PC-{y}", list(range(1, pc.
 ↪n_components + 1)))]
    transformed_df["target"] = df["target"]
    return transformed_df
```

[55]:
```
def plot_variance(pc, n_components=None):
    pc_cols = lambda x : [*map(lambda y : f"PC-{y}", list(range(1, x+1)))]

    n_comps = n_components if n_components is not None else pc.n_components

    plt.figure(figsize=(18,12))
    plt.style.use("ggplot")
    plt.rcParams.update({'font.size': 18})
    plt.bar(pc_cols(n_comps), pc.explained_variance_ratio_[:n_comps])
    plt.title("Explained Variance Ratio vs. Principal Component")
    plt.ylabel("Explained Variance Ratio")
    plt.xlabel("Principal Component")
    plt.ylim([0, 1])

    return
```

[15]:
```
def plot_pca_lda(df, **kwargs):
    title = kwargs.get("title", "Plot of Data With First Two Components")
    xlabel = kwargs.get("xlabel", "First Component")
    ylabel = kwargs.get("ylabel", "Second Component")
    eigens = kwargs.get("eigens", None)
    alpha = kwargs.get("alpha", 0.75)

    labels = np.unique(df["target"])

    fig = plt.figure(figsize=(18,12))
    plt.style.use("ggplot")
    plt.rcParams.update({'font.size': 18})
    #fig, ax = plt.subplots(1,1, figsize=(18,12), style="ggplot")
    ax = fig.add_subplot(111)
    ax.set_xlabel(xlabel)
    ax.set_ylabel(ylabel)
```

```
    ax.set_title(title)

    colors = ['r', 'g', 'b', 'y', 'm', 'c', 'k', 'r', 'g', 'b', 'y', 'm', 'c',␣
↪'k', 'r', 'g', 'b']
    markers = ['o', 'o', 'o', 'o', 'o', 'o', '*', '*', '*', '*', '*', '*', '+',␣
↪'+', '+', '+', '+']
    for i, label in enumerate(labels):
        first_two = df.loc[df["target"] == label].iloc[:, 0:2].to_numpy()
        ax.scatter(first_two[:, 0], first_two[:, 1], label=label, alpha=alpha,␣
↪color=colors[i], marker=markers[i])

    if eigens is not None:
        eig_vec = eigens[0][:2]
        eig_val = eigens[1][:2]
        for vec, val in zip(eig_vec, eig_val.T):
            ax.plot([0, np.sqrt(vec)*val[0]], [0, np.sqrt(vec)*val[1]], "k-",␣
↪lw=2)

    ax.legend()

    return
```

[16]:
```
def get_eigens(df):
    inputs = df.iloc[:, :-1].to_numpy()
    cov = np.cov(inputs.T)
    eig_vec, eig_val = np.linalg.eig(cov)

    print(f"Eigen Values:\n{eig_val}")
    print(f"Eigen Vectors:\n{eig_vec}")

    return eig_vec, eig_val
```

[17]:
```
def perform_pca(df, **kwargs):

    df_scaled = scale(df)
    pc = fit_pca(df_scaled, **kwargs)
    df_pca = transform_pca(df_scaled, pc)

    return df_pca
```

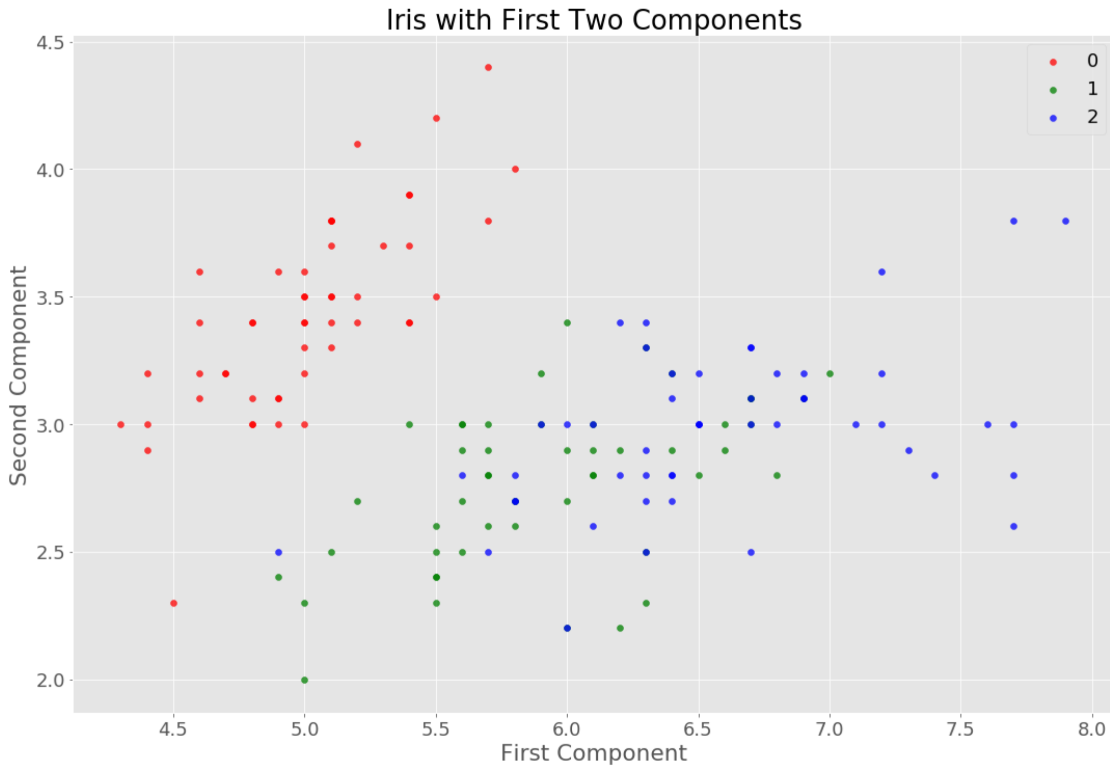
Now to create the same functionality for LDA.

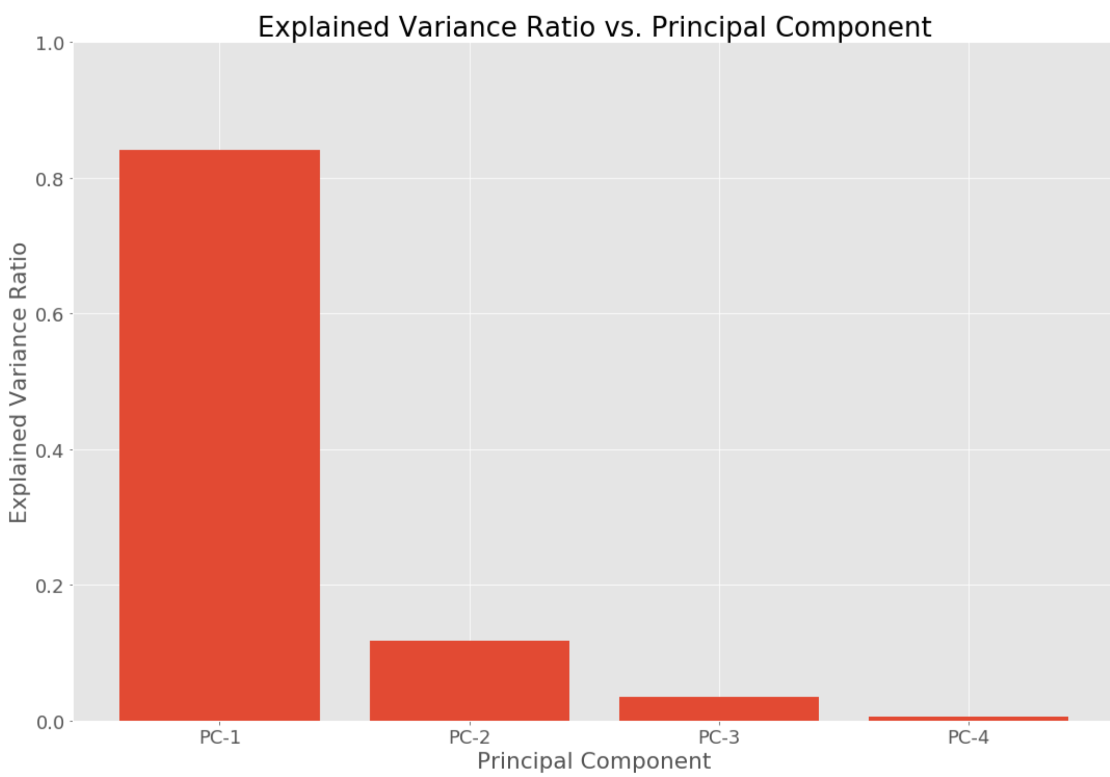
[98]:
```
def perform_lda(df, **kwargs):
    n_components = kwargs.get("n_components", len(np.unique(df["target"].
↪values))-1)

    inputs = df.iloc[:, :-1].to_numpy()
```

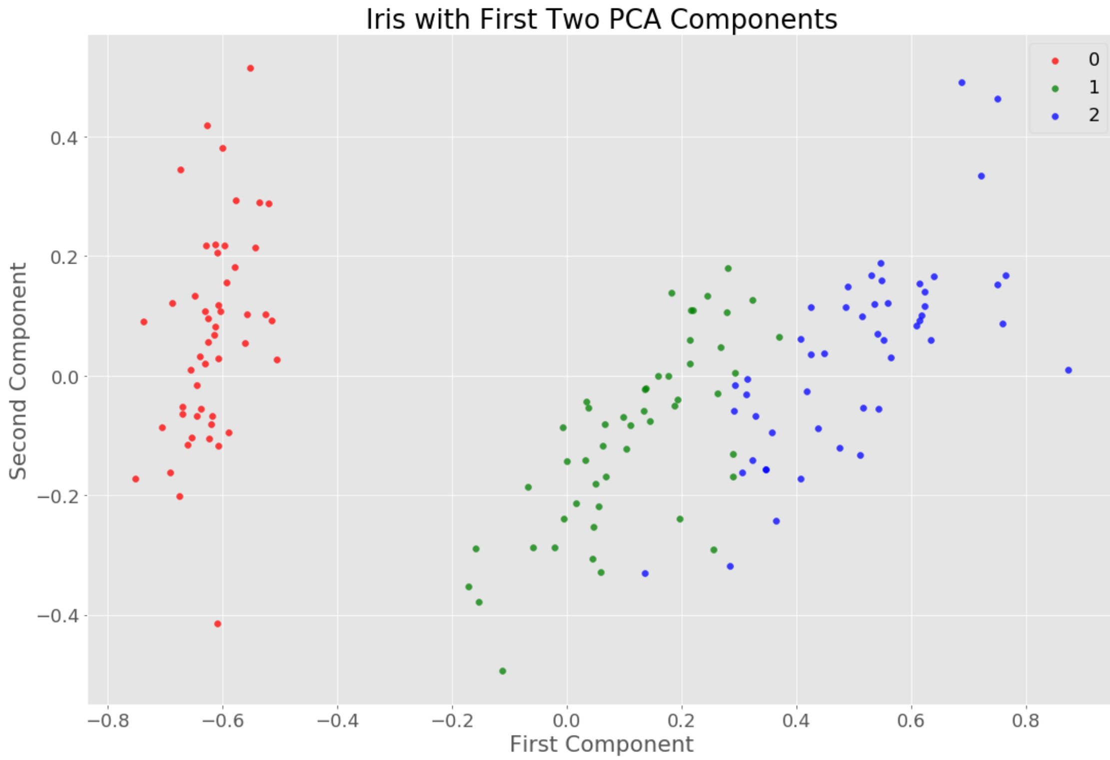
```
    targets = np.array(df["target"].values)

    lda = LinearDiscriminantAnalysis(n_components=n_components)
    transform_df = pd.DataFrame(data=lda.fit(inputs, targets).transform(inputs))
    transform_df["target"] = df["target"]

    return transform_df
```

[56]:
```
plot_pca_lda(iris_df, title="Iris with First Two Components")
iris_scaled = scale(iris_df)
pc = fit_pca(iris_scaled)
print(f"Explained Variance Ratio:\n{pc.explained_variance_ratio_}")
plot_variance(pc)
iris_pc = transform_pca(iris_scaled, pc)
eigens = get_eigens(iris_pc)
plot_pca_lda(iris_pc, title="Iris with First Two PCA Components")
```

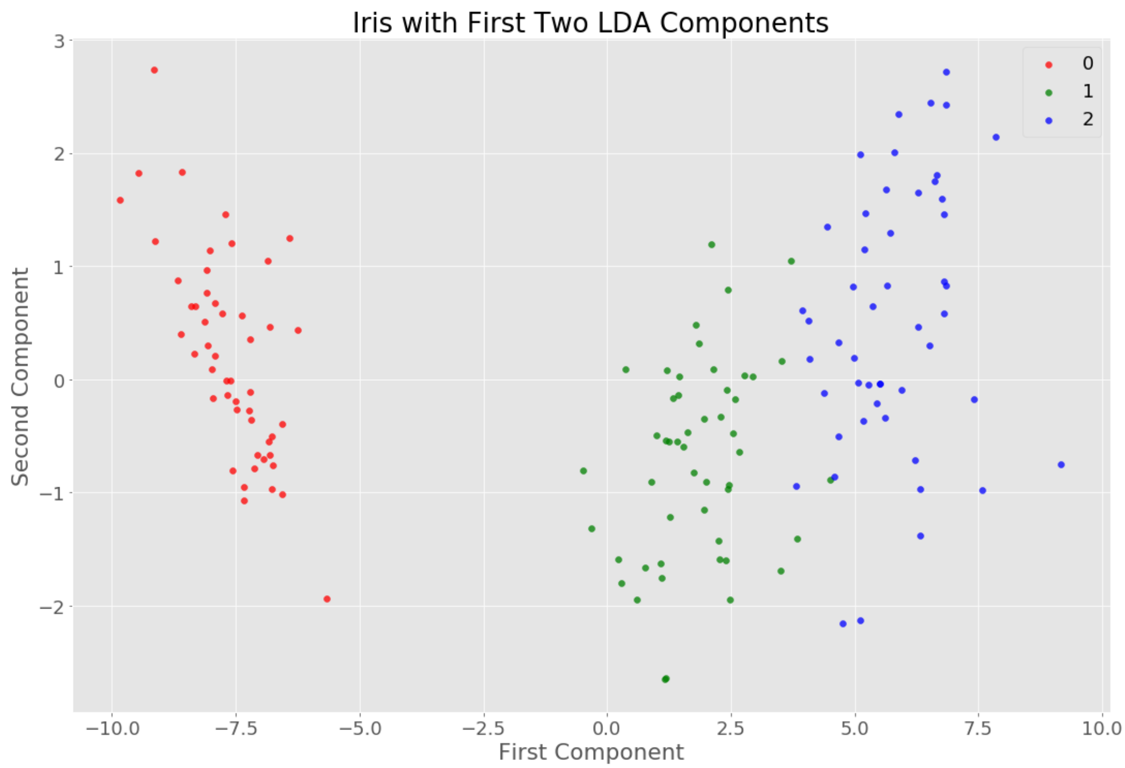
```
Explained Variance Ratio:
[0.84136038 0.11751808 0.03473561 0.00638592]
Eigen Values:
[[ 1.00000000e+00  2.27117367e-16  1.58067971e-16  8.17929707e-17]
 [ 0.00000000e+00  1.00000000e+00 -3.34545298e-16 -3.45648222e-17]
 [ 0.00000000e+00  3.46217222e-16  1.00000000e+00  8.87035040e-17]
 [ 0.00000000e+00  4.27653296e-17 -1.11519743e-16  1.00000000e+00]]
Eigen Vectors:
[0.23245325 0.0324682  0.00959685 0.00176432]
```


Iris with First Two Components
0
1
2
Second Component
First Component
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Explained Variance Ratio vs. Principal Component
Explained Variance Ratio
Principal Component
1.0
0.8
0.6
0.4
0.2
0.0
PC-1
PC-2
PC-3
PC-4



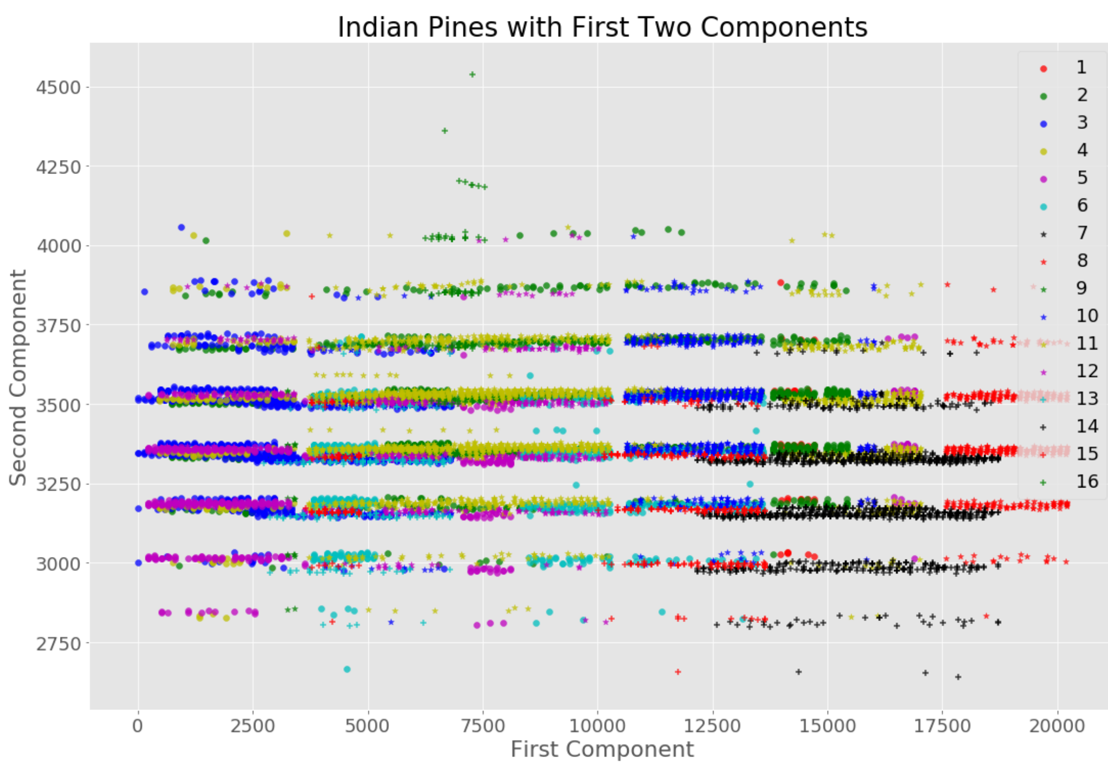
[99]:
```
iris_lda = perform_lda(iris_df)
plot_pca_lda(iris_lda, title="Iris with First Two LDA Components")
```

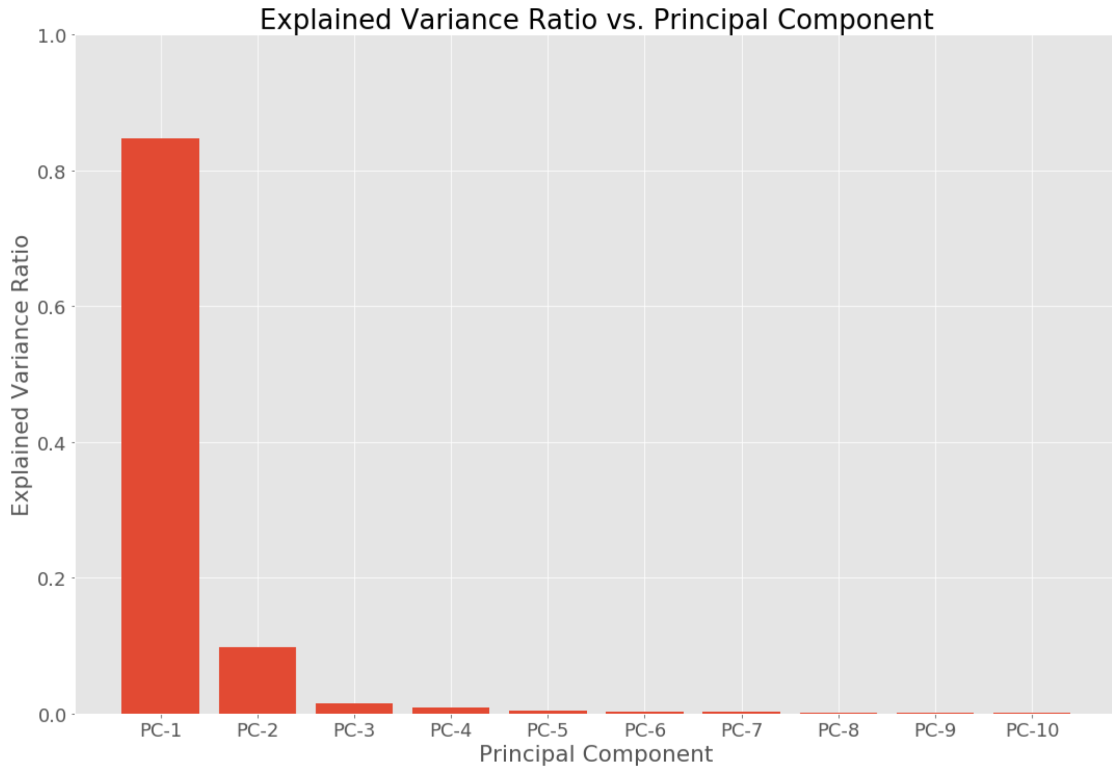


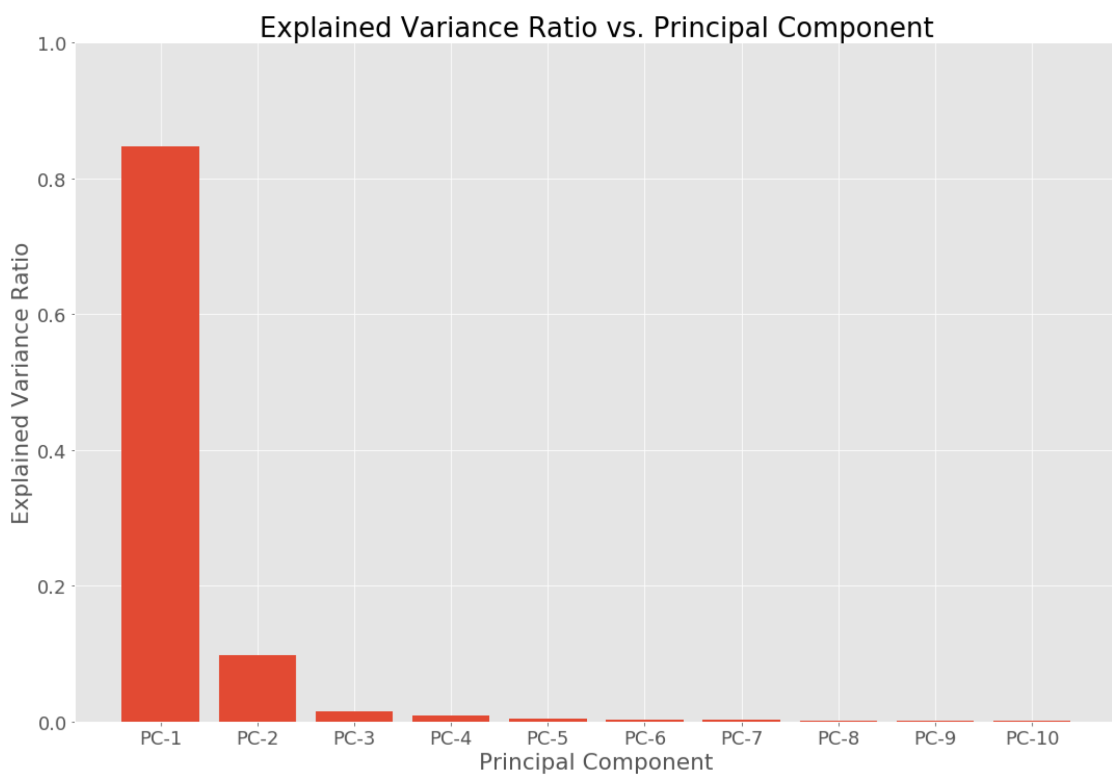
[57]:
```
plot_pca_lda(indian_df, title="Indian Pines with First Two Components")
indian_scaled = scale(indian_df, scaler="MinMax")
pc = fit_pca(indian_scaled, n_components=10)
print(f"Explained Variance Ratio:\n{pc.explained_variance_ratio_}")
plot_variance(pc)
plot_variance(pc, n_components=10)
indian_pc = transform_pca(indian_scaled, pc)
eigens = get_eigens(indian_pc)
plot_pca_lda(indian_pc, title="Indian Pines with First Two PCA Components")
```

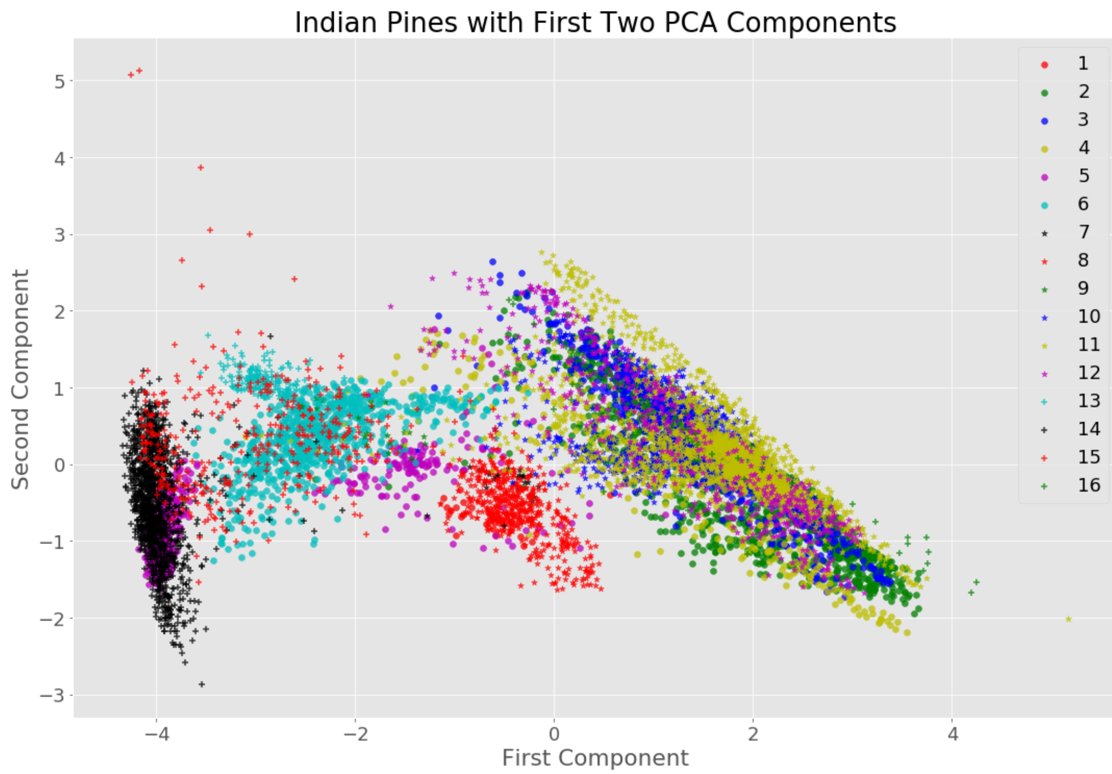
```
Explained Variance Ratio:
[0.84769736 0.0979396  0.01467606 0.00926067 0.00496592 0.00250275
 0.00205987 0.00193326 0.00187837 0.00145054]
Eigen Values:
[[-1.00000000e+00 -1.10880946e-14 -1.40330785e-16  4.85082057e-16
  -5.21843460e-17  7.91244887e-18  1.77887767e-17  4.99162322e-18
  -1.18330066e-17  6.43894374e-18]
 [ 1.09451092e-14 -1.00000000e+00 -2.64710375e-15 -4.82564896e-17
  -1.79793213e-17 -1.88014807e-18  2.17592495e-17 -7.42968871e-17
  -1.69549214e-17  1.06717568e-16]
 [ 1.40298725e-16  2.60253373e-15 -1.00000000e+00 -2.49253600e-15
   3.55477068e-16 -6.68949838e-17 -2.55609923e-17 -1.68382381e-17
  -2.00112835e-17  5.60125718e-17]
```

```
[ 4.85045236e-16 -8.59690482e-17 -2.27078014e-15  1.00000000e+00
  5.88837499e-16  2.33399434e-17 -4.54690348e-15 -2.30954762e-16
 -3.68794125e-16  3.39257181e-16]
[-5.21836566e-17 -1.83984740e-17 -7.68094097e-17 -8.51038433e-16
  1.00000000e+00 -5.18415414e-15 -3.04761817e-13 -1.10657704e-14
 -2.22272638e-14  2.42962006e-14]
[-7.91244474e-18  8.93207144e-19  4.36666978e-17 -3.17809477e-17
 -5.03406042e-15 -1.00000000e+00 -1.05699219e-10 -8.48933554e-12
 -1.68542306e-11  1.06431567e-11]
[ 4.99162301e-18 -7.35152431e-17 -5.13873478e-17  2.33400125e-16
  1.11017702e-14 -8.48958110e-12 -1.24637505e-09  1.00000000e+00
 -5.05969454e-10  2.45900340e-10]
[ 1.18330067e-17  1.79452390e-17  3.62878280e-17 -3.32106266e-16
 -2.23725540e-14  1.68533959e-11  3.88778810e-09 -5.05968335e-10
 -1.00000000e+00 -1.94585398e-09]
[-6.43894382e-18 -1.07635194e-16 -4.21677319e-17  3.08769410e-16
  2.43087216e-14 -1.06429512e-11 -3.28060483e-09  2.45900736e-10
  1.94585304e-09 -1.00000000e+00]
[ 1.77887718e-17  2.47380815e-17 -1.16363735e-16  4.60422907e-15
  3.05012776e-13 -1.05698449e-10  1.00000000e+00  1.24637467e-09
  3.88778805e-09 -3.28060503e-09]]
Eigen Vectors:
[5.49293297 0.63463172 0.09509832 0.06000752 0.03217833 0.01621739
 0.00939925 0.01334759 0.0125272  0.01217151]
```

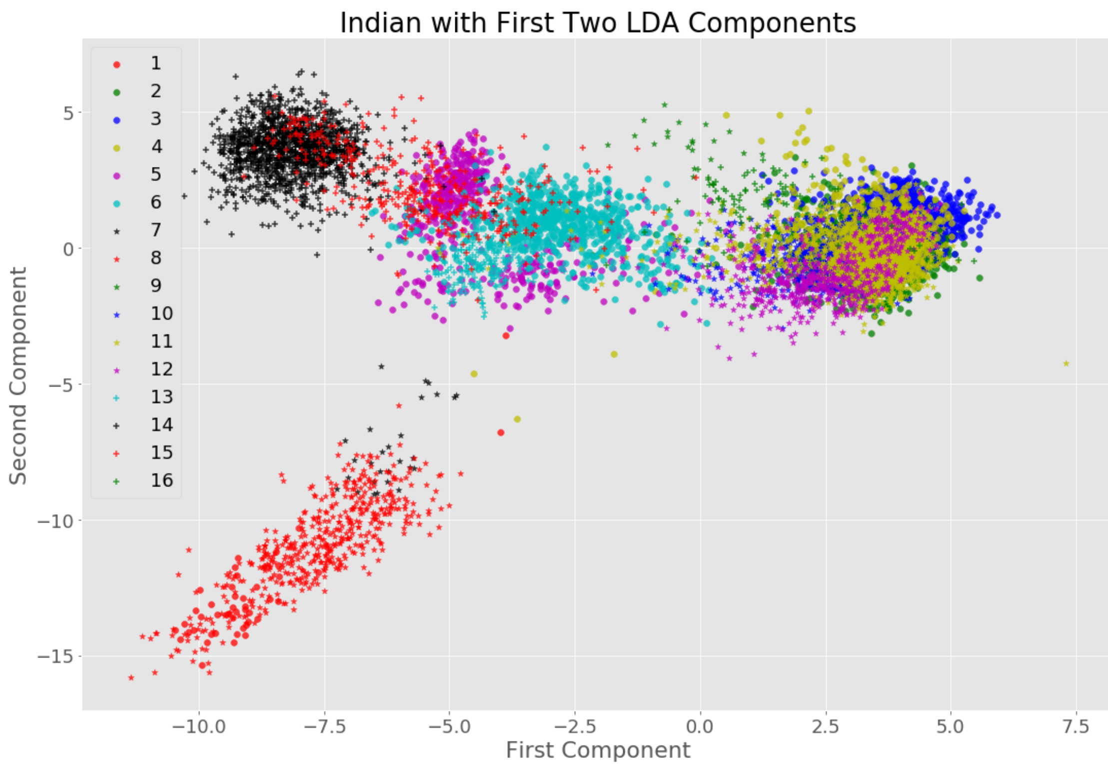



Explained Variance Ratio vs. Principal Component
Explained Variance Ratio
1.0
0.8
0.6
0.4
0.2
0.0
PC-1
PC-2
PC-3
PC-4
PC-5
PC-6
PC-7
PC-8
PC-9
PC-10
Principal Component


Explained Variance Ratio vs. Principal Component
Explained Variance Ratio
1.0
0.8
0.6
0.4
0.2
0.0
PC-1
PC-2
PC-3
PC-4
PC-5
PC-6
PC-7
PC-8
PC-9
PC-10
Principal Component



[105]:
```
indian_lda = perform_lda(indian_df, n_components=2)
plot_pca_lda(indian_lda, title="Indian with First Two LDA Components")
```



b)

i) The three plots of the first two components with no, pca and lda dimenstionality reduction for both the iris and indian datasets demonstrate the usefulness of dimensionality reduction and preojection. First, with the iris, with no projection there is a lot of overlap between the first and second classes. It is not clear how to delineate between the two. After performing either LDA or PCA, the separability because much clearer and easier. This separation also only requires two components/dimensions to be readily apparent. This is further supported by the variances plot which shows that the first two principal components contain over 90% of the iris' data variance.

Looking at the indian pines dataset and plots we see a similar pattern. The first plot shows no dimensionality reduction or projection. There is a lot of overlap between classes. Note that this plot just shows the first two components of the data, and not the best two. The PCA's and LDA's first two components do a much better job of separating the classes. There is still overlap between classes in the plot but we are starting to separate the classes out. It is not surprising that there is still some overlap between the data since we are only taking 2 of 202 components to try and separate all the data. However, the first two components show significant improvement and could make things much better once including several more components. From the variance plot, we see once again that the first two principal components are able to capture over 90% of the indian's data variance. This suggests we can severely decrease the number of components from 202 to a much more reasonable number. This should greatly reduce runtimes of learners on the data (as demonstrated later in question two).

The number of principal components to use for the classifiers seems to suggest 2 for the iris dataset, and possibly around 5-10 for the indian. We only need two for the iris because we can already see the three classes are separated in its plot. However, the indian dataset, while improved, would still need more components since the classes are not able to be completely separated with just the first two components. The same could be said for the LDA analyses.

The LDA and PCA worked about the same for the iris dataset. Most likely since it is a simple dataset and the variance does a good job of explaining the data's separate classes. The LDA appears to perform better on the indian dataset. PCA still works but there appears to still be significant overlap in some regions. The explained varaince plot also shows that minimal variance is explained by principal components greater than 2, but we can see from our plot that we will need more than just two principal components to explain the data. LDA seems to perform slightly better. This is most likely because it is able to take the classes into consideration when finding the best basis for projection to increase separability. PCA can sometimes struggle because it looks for overall variances, and does not consider intra- and inter- class means and variances.

Question 2)

a)

[60]:
```
from sklearn.model_selection import train_test_split, StratifiedKFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
import time
```

[24]:
```
def perform_classification(df_data, model, test_size):
    def split_data(df, test_size):
        X = df.iloc[:, :-1].to_numpy()
        y = df["target"].to_numpy()
        return train_test_split(X, y, test_size=test_size, random_state=42,␣
↪shuffle=True)

    X_train, X_test, Y_train, Y_test = split_data(df_data, test_size)


    model.fit(X_train, Y_train)
    predict_train = model.predict(X_train)

    predict_test = model.predict(X_test)

    return {"predict_train": predict_train,
            "predict_test": predict_test,
            "train": Y_train,
            "test": Y_test
           }
```

[74]:
```
def perform_performance(classifications):
    def get_spec_sens(confusion_matrix):
        specificities = []
```

```
        sensitivities = []

        for iLabel in list(range(len(confusion_matrix))):
            tp, tn, fn, fp = 0, 0, 0, 0
            for i in range(len(confusion_matrix)):
                for j in range(len(confusion_matrix)):
                    if j == iLabel and i == j:
                        tp += confusion_matrix[i, j]
                    elif j == iLabel:
                        fp += confusion_matrix[i, j]
                    elif iLabel == i:
                        fn += confusion_matrix[i, j]
                    else:
                        tn += confusion_matrix[i, j]
            sensitivity = tp / (tp+fn)
            specificity = tn / (tn+fp)
            sensitivities.append( sensitivity)
            specificities.append( specificity)

        return specificities, sensitivities

    def get_score(actual, predictions):
        correct = 0
        for i in range(len(actual)):
            if actual[i] == predictions[i]:
                correct += 1
        return correct / len(actual)

    def confusion_matrix(actual, predict, targets):
        def get_target_index(target):
            for i in range(len(targets)):
                if targets[i] == target:
                    found = True
                    break
            if found == False:
                raise ValueError(f"Target {target} not found in targets {targets}")
            return i

        conf_mat = np.zeros((len(targets), len(targets)), dtype=int)
        for i in range(len(actual)):
            iActual = get_target_index(actual[i])
            iPredict = get_target_index(predict[i])
            conf_mat[iActual, iPredict] += 1

        return conf_mat
```

```
    performance = {}
    targets = np.unique(list(np.unique(classifications["train"])) + list(np.unique(classifications["test"])))
    performance["conf_mat_train"] = confusion_matrix(classifications["train"], classifications["predict_train"], targets)
    performance["specificity_train"], performance["sensitivity_train"] = get_spec_sens(performance["conf_mat_train"])
    performance["accuracy_train"] = get_score(classifications["train"], classifications["predict_train"])

    performance["conf_mat_test"] = confusion_matrix(classifications["test"], classifications["predict_test"], targets)
    performance["specificity_test"], performance["sensitivity_test"] = get_spec_sens(performance["conf_mat_test"])
    performance["accuracy_test"] = get_score(classifications["test"], classifications["predict_test"])

    return performance
```

[26]:
```
def print_conf_mat(conf_mat, labels):
    spacer=4
    width = " "*spacer
    print("Act /" + "    Predictions  ")

    line = width
    for label in labels:
        line += " | " + label
    print(line)
    for i, label in enumerate(labels):
        line = label
        for j in range(len(conf_mat)):
            line += f" | {conf_mat[i,j]:<4}"
        print(line)
```

[27]:
```
def get_targets(dataset, targets):
    def convert(val):
        if val == 0:
            ret = "Seto"
        elif val == 1:
            ret = "Vers"
        else:
            ret = "Virg"
        return ret

    ret_targets = np.empty(len(targets), dtype=object)
    if dataset == "iris":
```

```
        for i in range(len(targets)):
            ret_targets[i] = convert(targets[i])
    else:
        for i in range(len(targets)):
            ret_targets[i] = str(targets[i]).ljust(4)
    return ret_targets
```

[28]:
```
def run_model(dataset, df, model, testing_sizes):
    def get_training_sizes(testing_sizes):
        training_sizes = []
        for i in range(len(testing_sizes)):
            training_size = np.round((1-testing_sizes[i])*10)/10
            training_sizes.append(training_size)
        return training_sizes

    output = {}
    start = time.time()
    output["targets"] = get_targets(dataset, np.unique(df.target))
    output["specs_train"] = np.zeros((len(testing_sizes),␣
 ↪len(output["targets"])))
    output["senss_train"] = np.zeros((len(testing_sizes),␣
 ↪len(output["targets"])))
    output["specs_test"] = np.zeros((len(testing_sizes),␣
 ↪len(output["targets"])))
    output["senss_test"] = np.zeros((len(testing_sizes),␣
 ↪len(output["targets"])))
    output["train_scores"] = []
    output["test_scores"] = []
    output["training_sizes"] = get_training_sizes(testing_sizes)

    for i, size in enumerate(testing_sizes):
        print(size)
        classifications = perform_classification(df, model, size)
        performance = perform_performance(classifications)

        output["test_scores"].append(performance["accuracy_test"])
        output["train_scores"].append(performance["accuracy_train"])
        output["specs_train"][i, :] = performance["specificity_train"]
        output["senss_train"][i, :] = performance["sensitivity_train"]
        output["specs_test"][i, :] = performance["specificity_test"]
        output["senss_test"][i, :] = performance["sensitivity_test"]
        if size == 0.7:
            output["conf_mat_train"] = performance["conf_mat_train"]
            output["conf_mat_test"] = performance["conf_mat_test"]
    output["runtime"] = time.time() - start
    return output
```

Now loop over all possible cases, and run the models.

[61]:
```
testing_sizes = np.array([*range(90, 0, -10)])/100
models = {}
models["svm_rbf"] = SVC(kernel="rbf", gamma="auto")
models["svm_poly"] = SVC(kernel="poly", gamma="auto")
models["svm_linear"] = SVC(kernel="linear", gamma="auto")
models["knn"] = KNeighborsClassifier()
models["naive_bayes"] = GaussianNB()
dim_reds = [("PCA", 3), ("LDA", 6), (None, None)]
```

[88]:
```
iris_pca = perform_pca(iris_unnamed, n_components=2)
iris_lda = perform_lda(iris_unnamed, n_components=2)
indian_pca = perform_pca(indian_df, n_components=4)
indian_lda = perform_lda(indian_df, n_components=15)
dfs = {"iris_PCA_2": iris_pca,
      "iris_LDA_2": iris_lda,
      "iris_None_None": iris_unnamed,
      "indian_PCA_4": indian_pca,
      "indian_LDA_15": indian_lda,
      "indian_None_None": indian_df
    }
```

[89]:
```
total_output = {}
for key, df in dfs.items():
    data, dim_red, dim_num = key.split("_")
    output_classifier = {}
    for classifier, model in models.items():
        print(f"==== {data} - {classifier} - {dim_red} ====")
        output_classifier[classifier] = run_model(data, df, model,␣
 ↪testing_sizes)
    total_output[key] = output_classifier

print("finished")
```

```
==== iris - svm_rbf - PCA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== iris - svm_poly - PCA ====
0.9
0.8
```

```
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== iris - svm_linear - PCA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== iris - knn - PCA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== iris - naive_bayes - PCA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== iris - svm_rbf - LDA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== iris - svm_poly - LDA ====
```

```
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== iris - svm_linear - LDA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== iris - knn - LDA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== iris - naive_bayes - LDA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== iris - svm_rbf - None ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
```

```
0.1
==== iris - svm_poly - None ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== iris - svm_linear - None ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== iris - knn - None ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== iris - naive_bayes - None ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== indian - svm_rbf - PCA ====
0.9
0.8
0.7
0.6
0.5
0.4
```

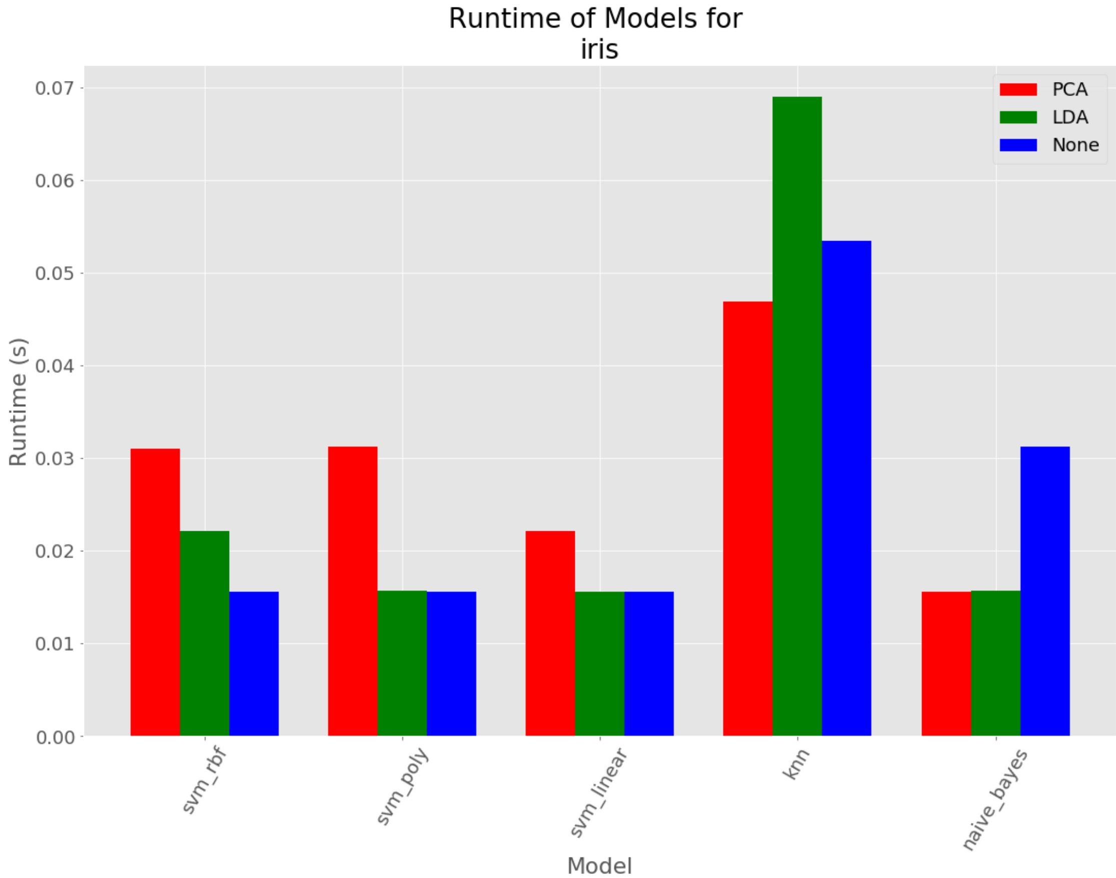
```
0.3
0.2
0.1
==== indian - svm_poly - PCA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== indian - svm_linear - PCA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== indian - knn - PCA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== indian - naive_bayes - PCA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== indian - svm_rbf - LDA ====
0.9
0.8
0.7
0.6
```

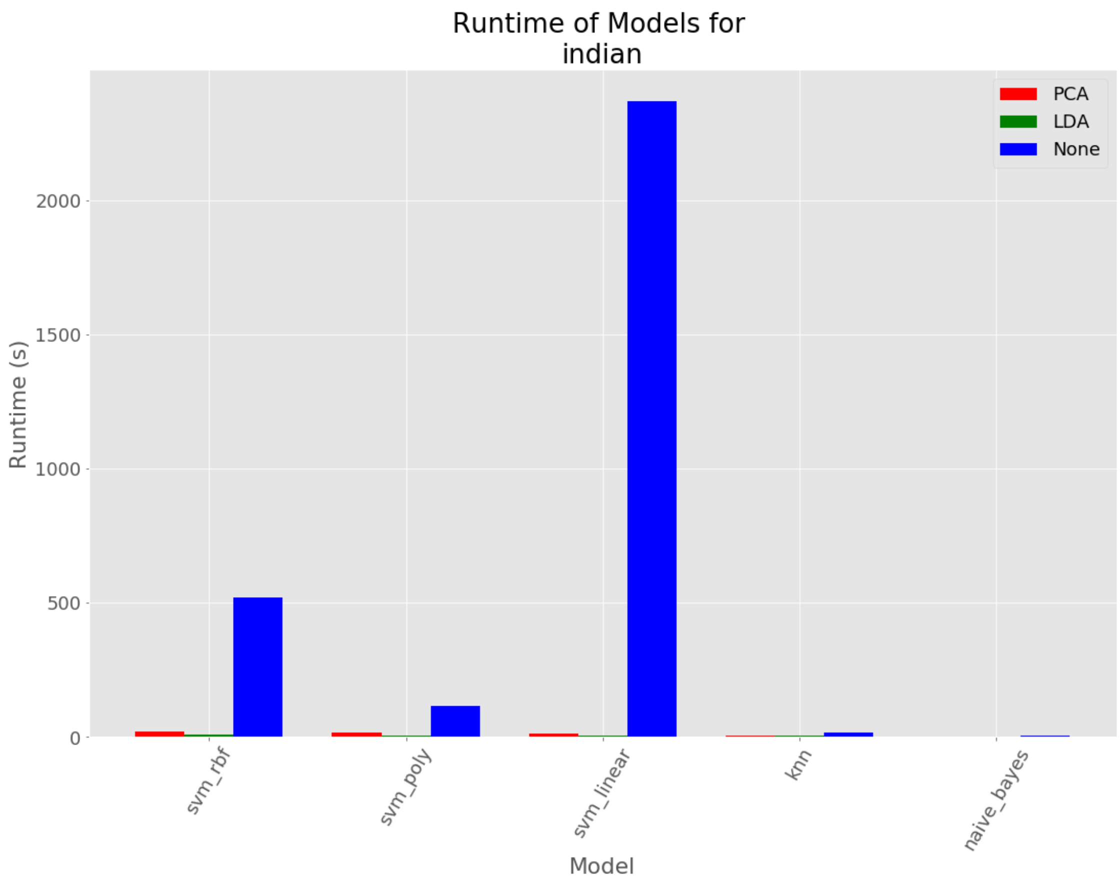
```
0.5
0.4
0.3
0.2
0.1
==== indian - svm_poly - LDA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== indian - svm_linear - LDA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== indian - knn - LDA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== indian - naive_bayes - LDA ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== indian - svm_rbf - None ====
0.9
0.8
```

```
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== indian - svm_poly - None ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== indian - svm_linear - None ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== indian - knn - None ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
==== indian - naive_bayes - None ====
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
finished
```

[78]:

```
for dataset in ["iris", "indian"]:
    count = 0
    fig = plt.figure(figsize=(18,12))
    ax = fig.add_subplot(111)
    ax.set_title(f"Runtime of Models for \n{dataset}")
    ax.set_xlabel("Model")
    ax.set_ylabel("Runtime (s)")
    colors = ["r", "g", "b"]
    for key, df in dfs.items():
        data, dim_red, dim_num = key.split("_")
        if data != dataset:
            continue
        runtimes = []
        classifiers = []
        for classifier, model in models.items():
            runtimes.append(total_output[key][classifier]["runtime"])
            classifiers.append(classifier)
        X = np.arange(len(classifiers))
        ax.bar(X -0.25 + count*0.25, runtimes, color=colors[count], width=0.25,␣
 ↪label=dim_red)
        count +=1
    classifiers = [""] + classifiers
    ax.set_xticklabels(classifiers, Rotation=60)
    ax.legend()
```


Runtime of Models for
iris
PCA
LDA
None
Runtime (s)
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
svm_rbf
svm_poly
svm_linear
knn
naive_bayes
Model



[72]:
```
def get_header(data, dim_red, dim_num, classifier, length):
    mid = f" {data} - {dim_red} - {dim_num} - {classifier} "
    excess = length - len(mid)
    if excess > 0:
        output = "=" * (excess // 2 + excess % 2)
        output += mid
        output += "=" * (excess // 2)
    else:
        output = mid
    return output
```

[90]:
```
for dataset in ["iris", "indian"]:
    fig_scores, ax_scores = plt.subplots(3, 2, figsize=(24,18))
    fig_scores.suptitle(f"{dataset}\nTraining/Testing Scores vs. Training Size")
    fig_sens, ax_sens = plt.subplots(3, 2, figsize=(24,18))
    fig_sens.suptitle(f"{dataset}\nSensisitivity vs Target for each Classifier")
    fig_specs, ax_specs = plt.subplots(3, 2, figsize=(24,18))
    fig_specs.suptitle(f"{dataset}\nSpecificity vs Target for each Classifier")

    for axes in [ax_scores, ax_sens, ax_specs]:
```

```python
        for axis, dim_reduction in zip(range(3), ["None", "PCA", "LDA"]):
            axes[axis, 0].set_title(f"{dim_reduction} Train")
            axes[axis, 1].set_title(f"{dim_reduction} Test")

    for key, df in dfs.items():
        data, dim_red, dim_num = key.split("_")
        if data != dataset:
            continue
        if dim_red == "PCA":
            axis = 1
        elif dim_red == "LDA":
            axis = 2
        else:
            axis = 0

        senss_train = []
        senss_test = []
        specs_train = []
        specs_test = []
        classifiers = []
        for classifier, model in models.items():
            run_output = total_output[key][classifier]
            if True:
                print("============================================")
                print(get_header(data, dim_red, dim_num, classifier, 44))
                print("============================================")
                print(f"======       Train Size: 30%         =======")
                print("----      Training Confusion Matrix     ----")
                print_conf_mat(run_output["conf_mat_train"],␣
↪run_output["targets"])
                print("----      Testing Confusion Matrix      ----")
                print_conf_mat(run_output["conf_mat_test"],␣
↪run_output["targets"])

            label = f"{classifier}"
            ax_scores[axis, 0].plot(run_output["training_sizes"],
                                    run_output["train_scores"], label=label)
            ax_scores[axis, 1].plot(run_output["training_sizes"],
                                    run_output["test_scores"], label=label)
            classifiers.append(classifier)
            senss_train.append(run_output["senss_train"][2,:])
            senss_test.append(run_output["senss_test"][2,:])
            specs_train.append(run_output["specs_train"][2,:])
            specs_test.append(run_output["specs_test"][2,:])

        X = np.arange(1, len(run_output["targets"])+1)
        if dataset == "indian":
```

```
            pass
            #breakpoint()
        width = 1/(len(classifiers)+1)
        offset = width*(len(classifiers)//2) + width*(len(classifiers)%2)/2
        for iClassifier, classifier in enumerate(classifiers):
            ax_sens[axis, 0].bar(X+offset-width*iClassifier,
                                 senss_train[iClassifier],
                                 label=classifier,
                                 width=width)
            ax_sens[axis, 1].bar(X+offset-width*iClassifier,
                                 senss_test[iClassifier],
                                 label=classifier,
                                 width=width)
            ax_specs[axis, 0].bar(X+offset-width*iClassifier,
                                 specs_train[iClassifier],
                                 label=classifier,
                                 width=width)
            ax_specs[axis, 1].bar(X+offset-width*iClassifier,
                                 specs_test[iClassifier],
                                 label=classifier,
                                 width=width)
        targets = [""] + run_output["targets"]
        for i in range(3):
            for j in range(2):
                ax_sens[i, j].set_xticks(X)
                ax_specs[i, j].set_xticks(X)
                ax_sens[i, j].set_xticklabels(targets, Rotation=60)
                ax_specs[i, j].set_xticklabels(targets, Rotation=60)

    for axes, yax, xax in [(ax_scores, " Score", "Training Size"),
                           (ax_sens, " Sensitivity", "Target Class"),
                           (ax_specs, " Specificity", "Target Class")]:
        for i in range(3):
            for j, te_tr in [(0, "Train"), (1, "Test")]:
                axes[i, j].set_ylim([0, 1.1])
                axes[i, j].legend(loc="lower right")
                axes[i, j].set_xlabel(xax)
                axes[i, j].set_ylabel(te_tr + yax)
    for fig in [fig_scores, fig_sens, fig_specs]:
        fig.tight_layout()
```

```
============================================
========= iris - PCA - 2 - svm_rbf =========
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
```

```
     | Seto | Vers | Virg
Seto | 10   | 0    | 0
Vers | 0    | 16   | 1
Virg | 0    | 4    | 14
----      Testing Confusion Matrix      ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 40   | 0    | 0
Vers | 0    | 32   | 1
Virg | 0    | 8    | 24
============================================
========= iris - PCA - 2 - svm_poly ========
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 0    | 0    | 10
Vers | 0    | 0    | 17
Virg | 0    | 0    | 18
----      Testing Confusion Matrix      ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 0    | 0    | 40
Vers | 0    | 0    | 33
Virg | 0    | 0    | 32
============================================
======== iris - PCA - 2 - svm_linear =======
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 10   | 0    | 0
Vers | 0    | 16   | 1
Virg | 0    | 4    | 14
----      Testing Confusion Matrix      ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 40   | 0    | 0
Vers | 0    | 32   | 1
Virg | 0    | 5    | 27
============================================
=========== iris - PCA - 2 - knn ===========
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
```

```
     | Seto | Vers | Virg
Seto | 10   | 0    | 0
Vers | 0    | 17   | 0
Virg | 0    | 1    | 17
----      Testing Confusion Matrix      ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 40   | 0    | 0
Vers | 0    | 30   | 3
Virg | 0    | 1    | 31
============================================
======= iris - PCA - 2 - naive_bayes =======
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 10   | 0    | 0
Vers | 0    | 15   | 2
Virg | 0    | 2    | 16
----      Testing Confusion Matrix      ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 40   | 0    | 0
Vers | 0    | 28   | 5
Virg | 0    | 4    | 28
============================================
========= iris - LDA - 2 - svm_rbf =========
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 10   | 0    | 0
Vers | 0    | 17   | 0
Virg | 0    | 0    | 18
----      Testing Confusion Matrix      ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 40   | 0    | 0
Vers | 0    | 29   | 4
Virg | 0    | 0    | 32
============================================
========= iris - LDA - 2 - svm_poly ========
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
```

```
     | Seto | Vers | Virg
Seto | 10   | 0    | 0
Vers | 0    | 17   | 0
Virg | 0    | 0    | 18
----      Testing Confusion Matrix      ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 40   | 0    | 0
Vers | 0    | 29   | 4
Virg | 0    | 0    | 32
============================================
======== iris - LDA - 2 - svm_linear =======
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 10   | 0    | 0
Vers | 0    | 17   | 0
Virg | 0    | 0    | 18
----      Testing Confusion Matrix      ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 40   | 0    | 0
Vers | 0    | 30   | 3
Virg | 0    | 0    | 32
============================================
=========== iris - LDA - 2 - knn ===========
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 10   | 0    | 0
Vers | 0    | 17   | 0
Virg | 0    | 0    | 18
----      Testing Confusion Matrix      ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 40   | 0    | 0
Vers | 0    | 31   | 2
Virg | 0    | 0    | 32
============================================
======= iris - LDA - 2 - naive_bayes =======
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
```

```
     | Seto | Vers | Virg
Seto | 10   | 0    | 0
Vers | 0    | 17   | 0
Virg | 0    | 0    | 18
----      Testing Confusion Matrix      ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 40   | 0    | 0
Vers | 0    | 30   | 3
Virg | 0    | 1    | 31
============================================
======= iris - None - None - svm_rbf =======
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 10   | 0    | 0
Vers | 0    | 17   | 0
Virg | 0    | 1    | 17
----      Testing Confusion Matrix      ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 40   | 0    | 0
Vers | 0    | 30   | 3
Virg | 0    | 0    | 32
============================================
======= iris - None - None - svm_poly ======
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 10   | 0    | 0
Vers | 0    | 17   | 0
Virg | 0    | 0    | 18
----      Testing Confusion Matrix      ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 40   | 0    | 0
Vers | 0    | 29   | 4
Virg | 0    | 0    | 32
============================================
====== iris - None - None - svm_linear =====
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
```

```
     | Seto | Vers | Virg
Seto | 10   | 0    | 0
Vers | 0    | 17   | 0
Virg | 0    | 0    | 18
----      Testing Confusion Matrix      ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 40   | 0    | 0
Vers | 0    | 31   | 2
Virg | 0    | 0    | 32
============================================
========= iris - None - None - knn =========
============================================
======       Train Size: 30%        =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 10   | 0    | 0
Vers | 0    | 17   | 0
Virg | 0    | 2    | 16
----      Testing Confusion Matrix      ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 40   | 0    | 0
Vers | 0    | 31   | 2
Virg | 0    | 1    | 31
============================================
===== iris - None - None - naive_bayes =====
============================================
======       Train Size: 30%        =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 10   | 0    | 0
Vers | 0    | 16   | 1
Virg | 0    | 1    | 17
----      Testing Confusion Matrix      ----
Act /    Predictions
     | Seto | Vers | Virg
Seto | 40   | 0    | 0
Vers | 0    | 29   | 4
Virg | 0    | 2    | 30
============================================
======== indian - PCA - 4 - svm_rbf ========
============================================
======       Train Size: 30%        =======
----      Training Confusion Matrix     ----
Act /    Predictions
```

```
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 14   | 0    | 0    | 1
| 0    | 0    | 0    | 0    | 0
2    | 0    | 187  | 11   | 0    | 1    | 1    | 0    | 1    | 0    | 16   | 195
| 23   | 0    | 0    | 0    | 0
3    | 0    | 10   | 151  | 1    | 0    | 0    | 0    | 0    | 0    | 1    | 70
| 9    | 0    | 0    | 0    | 0
4    | 0    | 0    | 13   | 42   | 0    | 2    | 0    | 0    | 0    | 0    | 1
| 1    | 0    | 0    | 0    | 0
5    | 0    | 0    | 0    | 2    | 140  | 7    | 0    | 4    | 0    | 0    | 0
| 0    | 0    | 0    | 1    | 0
6    | 0    | 0    | 0    | 0    | 3    | 209  | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 1    | 5    | 0
7    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 8    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 154  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 1    | 3    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 6    | 0    | 0    | 0    | 1    | 0    | 1    | 0    | 212  | 70
| 15   | 0    | 0    | 0    | 0
11   | 0    | 22   | 10   | 0    | 2    | 4    | 0    | 0    | 0    | 26   | 673
| 6    | 0    | 0    | 0    | 0
12   | 0    | 47   | 7    | 0    | 0    | 1    | 0    | 0    | 0    | 9    | 40
| 58   | 0    | 0    | 0    | 0
13   | 0    | 0    | 0    | 0    | 0    | 3    | 0    | 0    | 0    | 0    | 0
| 0    | 54   | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 1    | 1    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 370  | 2    | 0
15   | 0    | 0    | 0    | 0    | 1    | 46   | 0    | 0    | 0    | 0    | 0
| 0    | 9    | 31   | 35   | 0
16   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 2    | 0
| 0    | 0    | 0    | 0    | 19
----      Testing Confusion Matrix      ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 31   | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 391  | 20   | 0    | 1    | 1    | 0    | 2    | 0    | 36   | 454
| 88   | 0    | 0    | 0    | 0
3    | 0    | 20   | 362  | 16   | 0    | 0    | 0    | 0    | 0    | 1    | 171
| 18   | 0    | 0    | 0    | 0
4    | 1    | 0    | 48   | 118  | 0    | 7    | 0    | 0    | 0    | 1    | 1
| 2    | 0    | 0    | 0    | 0
5    | 0    | 0    | 0    | 4    | 288  | 13   | 0    | 10   | 0    | 6    | 6
| 0    | 0    | 0    | 2    | 0
```

```
6    | 0    | 0    | 0    | 0    | 5    | 484  | 0    | 0    | 0    | 0    | 4
| 0    | 1    | 0    | 18   | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 19   | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 324  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 1    | 15   | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 8    | 0    | 0    | 1    | 0    | 0    | 2    | 0    | 471  | 140
| 45   | 0    | 0    | 0    | 0
11   | 0    | 55   | 21   | 1    | 7    | 11   | 0    | 0    | 0    | 68   |
1529 | 20   | 0    | 0    | 0    | 0
12   | 0    | 144  | 14   | 0    | 0    | 0    | 0    | 0    | 0    | 19   | 135
| 119  | 0    | 0    | 0    | 0
13   | 0    | 0    | 0    | 0    | 0    | 12   | 0    | 0    | 0    | 0    | 1
| 0    | 135  | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 0    | 10   | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 874  | 7    | 0
15   | 0    | 0    | 0    | 0    | 8    | 116  | 0    | 0    | 0    | 3    | 2
| 2    | 10   | 61   | 62   | 0
16   | 0    | 6    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 4
| 2    | 0    | 0    | 0    | 59
============================================
======== indian - PCA - 4 - svm_poly =======
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 14   | 0    | 0    | 1
| 0    | 0    | 0    | 0    | 0
2    | 0    | 159  | 15   | 0    | 0    | 1    | 0    | 0    | 0    | 10   | 241
| 9    | 0    | 0    | 0    | 0
3    | 0    | 9    | 122  | 2    | 0    | 0    | 0    | 0    | 0    | 0    | 108
| 1    | 0    | 0    | 0    | 0
4    | 0    | 1    | 8    | 18   | 0    | 2    | 0    | 0    | 0    | 1    | 29
| 0    | 0    | 0    | 0    | 0
5    | 0    | 0    | 0    | 0    | 124  | 19   | 0    | 4    | 0    | 1    | 6
| 0    | 0    | 0    | 0    | 0
6    | 0    | 0    | 0    | 0    | 3    | 196  | 0    | 0    | 0    | 0    | 9
| 0    | 0    | 0    | 10   | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 8
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 144  | 0    | 0    | 10
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 2    | 2    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
```

```
10   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 181  | 114
| 10   | 0    | 0    | 0    | 0
11   | 0    | 19   | 4    | 0    | 0    | 2    | 0    | 0    | 0    | 26   | 692
| 0    | 0    | 0    | 0    | 0
12   | 0    | 19   | 17   | 0    | 0    | 5    | 0    | 0    | 0    | 3    | 79
| 39   | 0    | 0    | 0    | 0
13   | 0    | 0    | 0    | 0    | 0    | 3    | 0    | 0    | 0    | 0    | 0
| 0    | 54   | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 366  | 7    | 0
15   | 0    | 0    | 0    | 0    | 1    | 43   | 0    | 0    | 0    | 0    | 0
| 0    | 6    | 12   | 60   | 0
16   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 21
----      Testing Confusion Matrix      ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 29   | 0    | 0    | 2
| 0    | 0    | 0    | 0    | 0
2    | 0    | 334  | 32   | 0    | 1    | 2    | 0    | 0    | 0    | 14   | 583
| 27   | 0    | 0    | 0    | 0
3    | 0    | 15   | 317  | 20   | 0    | 0    | 0    | 0    | 0    | 1    | 231
| 4    | 0    | 0    | 0    | 0
4    | 0    | 7    | 23   | 83   | 0    | 11   | 0    | 1    | 0    | 2    | 51
| 0    | 0    | 0    | 0    | 0
5    | 0    | 0    | 1    | 1    | 246  | 43   | 0    | 8    | 0    | 2    | 28
| 0    | 0    | 0    | 0    | 0
6    | 0    | 0    | 0    | 0    | 11   | 455  | 0    | 0    | 0    | 0    | 17
| 0    | 1    | 0    | 28   | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 18
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 293  | 0    | 0    | 31
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 6    | 10   | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 415  | 222
| 29   | 0    | 0    | 0    | 0
11   | 0    | 29   | 10   | 0    | 0    | 11   | 0    | 0    | 0    | 75   |
1585 | 2    | 0    | 0    | 0    | 0
12   | 0    | 39   | 47   | 0    | 0    | 8    | 0    | 0    | 0    | 5    | 193
| 139  | 0    | 0    | 0    | 0
13   | 0    | 0    | 0    | 0    | 0    | 11   | 0    | 0    | 0    | 0    | 1
| 0    | 136  | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 0    | 3    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 863  | 25   | 0
15   | 0    | 0    | 0    | 0    | 2    | 113  | 0    | 1    | 0    | 0    | 7
| 0    | 8    | 31   | 102  | 0
```

```
16   | 0    | 4    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 2    | 4
| 2    | 0    | 0    | 0    | 60
============================================
======= indian - PCA - 4 - svm_linear ======
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 0    | 1    | 0    | 0    | 1    | 0    | 0    | 13   | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 166  | 25   | 0    | 1    | 1    | 0    | 1    | 0    | 17   | 210
| 14   | 0    | 0    | 0    | 0
3    | 0    | 2    | 160  | 4    | 0    | 0    | 0    | 0    | 0    | 0    | 76
| 0    | 0    | 0    | 0    | 0
4    | 0    | 0    | 13   | 45   | 0    | 1    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
5    | 3    | 0    | 0    | 1    | 134  | 15   | 0    | 0    | 0    | 0    | 1
| 0    | 0    | 0    | 0    | 0
6    | 0    | 0    | 0    | 0    | 15   | 192  | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 11   | 0
7    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 8    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 154  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 2    | 2    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 13   | 0    | 2    | 0    | 1    | 0    | 1    | 0    | 211  | 74
| 3    | 0    | 0    | 0    | 0
11   | 0    | 43   | 10   | 6    | 2    | 4    | 0    | 0    | 0    | 67   | 610
| 1    | 0    | 0    | 0    | 0
12   | 0    | 26   | 31   | 0    | 0    | 5    | 0    | 0    | 0    | 12   | 75
| 13   | 0    | 0    | 0    | 0
13   | 0    | 0    | 0    | 0    | 0    | 3    | 0    | 0    | 0    | 0    | 0
| 0    | 54   | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 370  | 3    | 0
15   | 0    | 0    | 0    | 0    | 9    | 38   | 0    | 0    | 0    | 0    | 0
| 0    | 9    | 28   | 38   | 0
16   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0
| 0    | 0    | 0    | 0    | 20
----      Testing Confusion Matrix      ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 31   | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
```

```
2    | 0    | 331  | 87   | 0    | 3    | 1    | 0    | 0    | 0    | 33   | 524
| 14   | 0    | 0    | 0    | 0
3    | 0    | 5    | 398  | 18   | 0    | 0    | 0    | 0    | 0    | 0    | 164
| 3    | 0    | 0    | 0    | 0
4    | 0    | 6    | 48   | 114  | 1    | 4    | 0    | 0    | 0    | 1    | 0
| 4    | 0    | 0    | 0    | 0
5    | 4    | 1    | 0    | 3    | 264  | 42   | 0    | 5    | 0    | 5    | 2
| 3    | 0    | 0    | 0    | 0
6    | 0    | 0    | 0    | 0    | 54   | 418  | 0    | 0    | 0    | 2    | 1
| 0    | 2    | 0    | 35   | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 19   | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 324  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 8    | 8    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 30   | 0    | 0    | 0    | 1    | 1    | 1    | 0    | 444  | 186
| 4    | 0    | 0    | 0    | 0
11   | 0    | 62   | 11   | 7    | 10   | 8    | 0    | 0    | 0    | 175  |
1425 | 14   | 0    | 0    | 0    | 0
12   | 0    | 63   | 111  | 0    | 0    | 7    | 0    | 0    | 0    | 31   | 174
| 45   | 0    | 0    | 0    | 0
13   | 0    | 0    | 0    | 1    | 0    | 9    | 0    | 0    | 0    | 0    | 0
| 0    | 138  | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 2    | 3    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 880  | 6    | 0
15   | 0    | 0    | 0    | 0    | 26   | 97   | 0    | 0    | 0    | 1    | 0
| 3    | 17   | 59   | 61   | 0
16   | 0    | 7    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 5    | 1
| 1    | 0    | 0    | 0    | 58
============================================
========== indian - PCA - 4 - knn ==========
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 14   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 361  | 10   | 0    | 1    | 2    | 0    | 1    | 0    | 8    | 42
| 10   | 0    | 0    | 0    | 0
3    | 0    | 11   | 196  | 4    | 0    | 0    | 0    | 0    | 0    | 1    | 24
| 6    | 0    | 0    | 0    | 0
4    | 0    | 0    | 8    | 47   | 0    | 2    | 0    | 0    | 0    | 1    | 0
| 1    | 0    | 0    | 0    | 0
5    | 1    | 0    | 0    | 2    | 148  | 0    | 1    | 2    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
```

```
6    | 0    | 0    | 0    | 0    | 3    | 211  | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 1    | 3    | 0
7    | 0    | 0    | 0    | 0    | 1    | 0    | 6    | 2    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 154  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 1    | 3    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 6    | 2    | 1    | 0    | 2    | 0    | 1    | 0    | 280  | 10
| 3    | 0    | 0    | 0    | 0
11   | 0    | 19   | 11   | 1    | 2    | 3    | 0    | 0    | 0    | 28   | 673
| 6    | 0    | 0    | 0    | 0
12   | 0    | 30   | 4    | 0    | 0    | 0    | 0    | 0    | 0    | 8    | 18
| 101  | 0    | 0    | 1    | 0
13   | 0    | 0    | 0    | 0    | 0    | 2    | 0    | 0    | 0    | 0    | 0
| 0    | 55   | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 1    | 1    | 0    | 0    | 0    | 0    | 0
| 0    | 1    | 366  | 5    | 0
15   | 0    | 0    | 0    | 0    | 1    | 37   | 0    | 0    | 0    | 0    | 0
| 0    | 7    | 11   | 66   | 0
16   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1
| 1    | 0    | 0    | 0    | 19
----      Testing Confusion Matrix      ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 27   | 0    | 0    | 0    | 0    | 0    | 0    | 4    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 731  | 35   | 1    | 3    | 1    | 1    | 0    | 0    | 25   | 148
| 48   | 0    | 0    | 0    | 0
3    | 0    | 35   | 431  | 26   | 1    | 0    | 0    | 0    | 0    | 8    | 69
| 18   | 0    | 0    | 0    | 0
4    | 1    | 2    | 20   | 129  | 2    | 7    | 0    | 0    | 0    | 13   | 4
| 0    | 0    | 0    | 0    | 0
5    | 5    | 0    | 0    | 6    | 298  | 5    | 0    | 3    | 0    | 9    | 3
| 0    | 0    | 0    | 0    | 0
6    | 0    | 0    | 0    | 0    | 12   | 476  | 0    | 0    | 0    | 0    | 0
| 1    | 3    | 3    | 17   | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 14   | 5    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 324  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 2    | 11   | 0    | 0    | 2    | 0    | 1
| 0    | 0    | 0    | 0    | 0
10   | 0    | 12   | 2    | 0    | 1    | 2    | 5    | 0    | 0    | 573  | 49
| 23   | 0    | 0    | 0    | 0
11   | 0    | 79   | 32   | 2    | 8    | 8    | 0    | 0    | 0    | 118  |
1443 | 22   | 0    | 0    | 0    | 0
```

```
12   | 0    | 103  | 15   | 0    | 3    | 1    | 0    | 0    | 0    | 28   | 64
| 217  | 0    | 0    | 0    | 0
13   | 0    | 0    | 0    | 0    | 0    | 6    | 0    | 0    | 0    | 0    | 1
| 0    | 141  | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 2    | 6    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 870  | 13   | 0
15   | 0    | 0    | 0    | 0    | 6    | 103  | 0    | 0    | 0    | 4    | 0
| 2    | 11   | 37   | 101  | 0
16   | 0    | 5    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 5
| 1    | 0    | 0    | 0    | 60
============================================
====== indian - PCA - 4 - naive_bayes ======
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 14   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1
| 0    | 0    | 0    | 0    | 0
2    | 0    | 179  | 19   | 2    | 0    | 1    | 0    | 0    | 0    | 26   | 199
| 9    | 0    | 0    | 0    | 0
3    | 0    | 22   | 132  | 16   | 0    | 0    | 0    | 0    | 0    | 0    | 68
| 4    | 0    | 0    | 0    | 0
4    | 0    | 8    | 13   | 29   | 0    | 1    | 0    | 0    | 0    | 0    | 7
| 1    | 0    | 0    | 0    | 0
5    | 1    | 2    | 0    | 1    | 107  | 26   | 1    | 0    | 0    | 1    | 0
| 5    | 0    | 0    | 10   | 0
6    | 0    | 0    | 0    | 0    | 12   | 196  | 0    | 0    | 0    | 4    | 0
| 0    | 0    | 0    | 6    | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 9    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 1    | 0    | 0    | 0    | 0    | 0    | 3    | 150  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 3    | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 36   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 172  | 82
| 15   | 0    | 0    | 0    | 0
11   | 0    | 67   | 26   | 0    | 0    | 6    | 0    | 0    | 0    | 32   | 588
| 24   | 0    | 0    | 0    | 0
12   | 0    | 37   | 33   | 0    | 0    | 3    | 0    | 0    | 0    | 2    | 66
| 20   | 0    | 0    | 1    | 0
13   | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 0
| 0    | 55   | 0    | 1    | 0
14   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 359  | 15   | 0
15   | 0    | 0    | 0    | 0    | 5    | 35   | 0    | 0    | 0    | 0    | 0
| 0    | 4    | 38   | 40   | 0
```

```
16   | 0    | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 20
----      Testing Confusion Matrix      ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 31   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 382  | 54   | 0    | 1    | 2    | 0    | 0    | 0    | 66   | 453
| 35   | 0    | 0    | 0    | 0
3    | 0    | 53   | 341  | 30   | 0    | 0    | 0    | 0    | 0    | 0    | 154
| 10   | 0    | 0    | 0    | 0
4    | 0    | 34   | 46   | 78   | 0    | 4    | 0    | 0    | 0    | 0    | 9
| 7    | 0    | 0    | 0    | 0
5    | 4    | 8    | 3    | 2    | 214  | 57   | 4    | 0    | 0    | 3    | 0
| 9    | 0    | 0    | 24   | 1
6    | 0    | 0    | 0    | 0    | 38   | 442  | 0    | 0    | 0    | 7    | 1
| 3    | 1    | 1    | 19   | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 18   | 1    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 8    | 316  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 1    | 3    | 0    | 0    | 12   | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 83   | 4    | 0    | 0    | 1    | 2    | 0    | 0    | 369  | 179
| 29   | 0    | 0    | 0    | 0
11   | 0    | 144  | 60   | 1    | 0    | 15   | 0    | 0    | 0    | 84   |
1342 | 66   | 0    | 0    | 0    | 0
12   | 0    | 126  | 62   | 0    | 0    | 4    | 0    | 0    | 0    | 3    | 163
| 71   | 0    | 0    | 1    | 1
13   | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 0
| 1    | 141  | 0    | 5    | 0
14   | 0    | 0    | 0    | 0    | 1    | 1    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 857  | 32   | 0
15   | 0    | 0    | 0    | 0    | 14   | 88   | 0    | 0    | 0    | 2    | 0
| 4    | 5    | 69   | 80   | 2
16   | 0    | 10   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 1    | 0    | 0    | 0    | 61
============================================
======== indian - LDA - 15 - svm_rbf =======
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 15   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
```

```
2     | 0     | 410   | 5     | 0     | 0     | 0     | 0     | 0     | 0     | 7     | 12
| 1     | 0     | 0     | 0     | 0
3     | 0     | 9     | 219   | 2     | 0     | 0     | 0     | 0     | 0     | 0     | 11
| 1     | 0     | 0     | 0     | 0
4     | 0     | 0     | 1     | 58    | 0     | 0     | 0     | 0     | 0     | 0     | 0
| 0     | 0     | 0     | 0     | 0
5     | 0     | 0     | 0     | 0     | 152   | 1     | 0     | 0     | 0     | 0     | 0
| 1     | 0     | 0     | 0     | 0
6     | 0     | 0     | 0     | 0     | 0     | 218   | 0     | 0     | 0     | 0     | 0
| 0     | 0     | 0     | 0     | 0
7     | 0     | 0     | 0     | 0     | 0     | 0     | 9     | 0     | 0     | 0     | 0
| 0     | 0     | 0     | 0     | 0
8     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 154   | 0     | 0     | 0
| 0     | 0     | 0     | 0     | 0
9     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 4     | 0     | 0
| 0     | 0     | 0     | 0     | 0
10    | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 289   | 15
| 0     | 0     | 0     | 0     | 0
11    | 0     | 15    | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 12    | 715
| 0     | 0     | 0     | 0     | 0
12    | 0     | 1     | 4     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0
| 157   | 0     | 0     | 0     | 0
13    | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0
| 0     | 57    | 0     | 0     | 0
14    | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0
| 0     | 0     | 374   | 0     | 0
15    | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0
| 0     | 0     | 5     | 117   | 0
16    | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0
| 0     | 0     | 0     | 0     | 21
----      Testing Confusion Matrix      ----
Act /    Predictions
      | 1     | 2     | 3     | 4     | 5     | 6     | 7     | 8     | 9     | 10    | 11
| 12    | 13    | 14    | 15    | 16
1     | 31    | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0
| 0     | 0     | 0     | 0     | 0
2     | 0     | 889   | 11    | 5     | 0     | 0     | 0     | 0     | 0     | 24    | 56
| 7     | 0     | 0     | 1     | 0
3     | 0     | 34    | 488   | 15    | 0     | 0     | 0     | 0     | 0     | 1     | 46
| 4     | 0     | 0     | 0     | 0
4     | 0     | 5     | 24    | 141   | 4     | 0     | 0     | 0     | 0     | 0     | 2
| 0     | 0     | 0     | 2     | 0
5     | 0     | 0     | 0     | 0     | 316   | 3     | 0     | 0     | 0     | 0     | 4
| 3     | 0     | 0     | 3     | 0
6     | 0     | 0     | 0     | 0     | 0     | 504   | 0     | 0     | 0     | 3     | 0
| 0     | 0     | 1     | 4     | 0
7     | 0     | 0     | 0     | 0     | 0     | 0     | 18    | 0     | 0     | 0     | 0
| 0     | 0     | 0     | 1     | 0
```

```
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 324  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 16   | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 8    | 0    | 0    | 0    | 2    | 0    | 0    | 0    | 587  | 68
| 2    | 0    | 0    | 0    | 0
11   | 0    | 45   | 13   | 0    | 12   | 0    | 0    | 0    | 0    | 55   |
1579 | 6    | 0    | 0    | 2    | 0
12   | 0    | 9    | 38   | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 5
| 377  | 0    | 0    | 1    | 0
13   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1
| 0    | 147  | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 882  | 8    | 0
15   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 10   | 254  | 0
16   | 0    | 2    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 3
| 0    | 0    | 0    | 20   | 47
============================================
======= indian - LDA - 15 - svm_poly =======
============================================
======        Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 15   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 418  | 2    | 0    | 0    | 0    | 0    | 0    | 0    | 5    | 9
| 1    | 0    | 0    | 0    | 0
3    | 0    | 4    | 231  | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 7
| 0    | 0    | 0    | 0    | 0
4    | 0    | 0    | 0    | 59   | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
5    | 0    | 0    | 0    | 0    | 153  | 0    | 0    | 0    | 0    | 0    | 0
| 1    | 0    | 0    | 0    | 0
6    | 0    | 0    | 0    | 0    | 0    | 218  | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 9    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 154  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 4    | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 296  | 8
| 0    | 0    | 0    | 0    | 0
11   | 0    | 9    | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 10   | 723
| 0    | 0    | 0    | 0    | 0
```

```
12   | 0    | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 161  | 0    | 0    | 0    | 0
13   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 57   | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 374  | 0    | 0
15   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 122  | 0
16   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 21
----      Testing Confusion Matrix      ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 31   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 887  | 18   | 2    | 0    | 0    | 0    | 0    | 0    | 30   | 51
| 4    | 0    | 0    | 1    | 0
3    | 0    | 40   | 492  | 13   | 0    | 0    | 0    | 0    | 0    | 3    | 34
| 6    | 0    | 0    | 0    | 0
4    | 0    | 7    | 35   | 129  | 4    | 0    | 0    | 0    | 0    | 2    | 0
| 1    | 0    | 0    | 0    | 0
5    | 0    | 1    | 0    | 0    | 303  | 4    | 0    | 0    | 0    | 13   | 3
| 4    | 0    | 0    | 1    | 0
6    | 0    | 0    | 0    | 0    | 6    | 501  | 0    | 0    | 0    | 4    | 0
| 0    | 0    | 0    | 1    | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 19   | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 323  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 1    | 2    | 0    | 0    | 0    | 13   | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 14   | 1    | 0    | 0    | 3    | 0    | 0    | 0    | 585  | 63
| 1    | 0    | 0    | 0    | 0
11   | 0    | 50   | 22   | 0    | 6    | 0    | 0    | 0    | 0    | 91   |
1536 | 6    | 0    | 0    | 1    | 0
12   | 0    | 13   | 24   | 0    | 0    | 0    | 0    | 0    | 0    | 5    | 3
| 385  | 0    | 0    | 0    | 1
13   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 2
| 0    | 146  | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 869  | 21   | 0
15   | 0    | 1    | 0    | 0    | 1    | 7    | 0    | 0    | 0    | 2    | 0
| 4    | 0    | 13   | 236  | 0
16   | 0    | 4    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1
| 4    | 0    | 0    | 0    | 63
============================================
====== indian - LDA - 15 - svm_linear ======
```

```
=============================================
======       Train Size: 30%        =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 15   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 382  | 12   | 1    | 0    | 0    | 0    | 0    | 0    | 17   | 21
| 2    | 0    | 0    | 0    | 0
3    | 0    | 12   | 210  | 5    | 0    | 0    | 0    | 0    | 0    | 0    | 11
| 4    | 0    | 0    | 0    | 0
4    | 0    | 0    | 1    | 58   | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
5    | 0    | 0    | 0    | 0    | 152  | 1    | 0    | 0    | 0    | 0    | 0
| 1    | 0    | 0    | 0    | 0
6    | 0    | 0    | 0    | 0    | 0    | 218  | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 9    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 154  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 4    | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 12   | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 246  | 46
| 0    | 0    | 0    | 0    | 0
11   | 0    | 51   | 5    | 0    | 3    | 0    | 0    | 0    | 0    | 30   | 654
| 0    | 0    | 0    | 0    | 0
12   | 0    | 1    | 6    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 3
| 152  | 0    | 0    | 0    | 0
13   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 57   | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 371  | 3    | 0
15   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 3    | 119  | 0
16   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 21
----      Testing Confusion Matrix      ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 31   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 862  | 29   | 6    | 0    | 0    | 0    | 0    | 0    | 35   | 53
| 7    | 0    | 0    | 1    | 0
3    | 0    | 40   | 493  | 17   | 0    | 0    | 0    | 0    | 0    | 0    | 31
| 7    | 0    | 0    | 0    | 0
```

```
4    | 0    | 4    | 15   | 154  | 3    | 0    | 0    | 1    | 0    | 0    | 0
| 1    | 0    | 0    | 0    | 0
5    | 0    | 1    | 0    | 2    | 317  | 3    | 0    | 0    | 0    | 3    | 1
| 1    | 0    | 0    | 1    | 0
6    | 0    | 0    | 0    | 0    | 12   | 498  | 0    | 0    | 0    | 1    | 0
| 0    | 0    | 0    | 1    | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 19   | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 324  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 16   | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 24   | 1    | 0    | 0    | 2    | 0    | 0    | 0    | 517  | 122
| 1    | 0    | 0    | 0    | 0
11   | 0    | 86   | 36   | 1    | 14   | 0    | 0    | 0    | 0    | 76   |
1496 | 3    | 0    | 0    | 0    | 0
12   | 0    | 6    | 45   | 1    | 0    | 0    | 0    | 0    | 0    | 5    | 10
| 363  | 0    | 0    | 0    | 1
13   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1
| 0    | 147  | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 876  | 14   | 0
15   | 0    | 0    | 0    | 0    | 4    | 6    | 0    | 0    | 0    | 1    | 1
| 3    | 0    | 13   | 236  | 0
16   | 0    | 3    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 4    | 0    | 0    | 0    | 65
============================================
========== indian - LDA - 15 - knn =========
============================================
======        Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 12   | 0    | 0    | 0    | 1    | 0    | 0    | 2    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 399  | 4    | 0    | 0    | 0    | 0    | 0    | 0    | 14   | 15
| 3    | 0    | 0    | 0    | 0
3    | 0    | 13   | 201  | 5    | 0    | 0    | 0    | 0    | 0    | 1    | 19
| 3    | 0    | 0    | 0    | 0
4    | 0    | 1    | 7    | 51   | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
5    | 0    | 0    | 0    | 0    | 150  | 3    | 0    | 0    | 0    | 0    | 0
| 1    | 0    | 0    | 0    | 0
6    | 0    | 0    | 0    | 0    | 0    | 218  | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 9    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
```

```
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 154  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 4    | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 4    | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 274  | 26
| 0    | 0    | 0    | 0    | 0
11   | 0    | 18   | 2    | 0    | 3    | 1    | 0    | 0    | 0    | 17   | 702
| 0    | 0    | 0    | 0    | 0
12   | 0    | 4    | 5    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 7
| 145  | 0    | 0    | 1    | 0
13   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 57   | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 1    | 372  | 1    | 0
15   | 0    | 0    | 0    | 0    | 0    | 2    | 0    | 0    | 0    | 0    | 0
| 0    | 1    | 8    | 111  | 0
16   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 21
----      Testing Confusion Matrix      ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 30   | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 856  | 18   | 4    | 0    | 1    | 0    | 0    | 0    | 24   | 87
| 3    | 0    | 0    | 0    | 0
3    | 0    | 37   | 467  | 13   | 0    | 0    | 0    | 0    | 0    | 1    | 65
| 5    | 0    | 0    | 0    | 0
4    | 0    | 11   | 38   | 123  | 3    | 0    | 0    | 1    | 0    | 0    | 2
| 0    | 0    | 0    | 0    | 0
5    | 0    | 1    | 1    | 0    | 311  | 6    | 0    | 0    | 0    | 5    | 1
| 3    | 0    | 0    | 1    | 0
6    | 0    | 0    | 0    | 0    | 1    | 511  | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 19   | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 324  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 16   | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 16   | 1    | 0    | 0    | 4    | 0    | 0    | 0    | 567  | 79
| 0    | 0    | 0    | 0    | 0
11   | 0    | 64   | 12   | 0    | 13   | 0    | 0    | 0    | 0    | 79   |
1537 | 7    | 0    | 0    | 0    | 0
12   | 0    | 11   | 48   | 0    | 0    | 0    | 0    | 0    | 0    | 2    | 15
| 354  | 0    | 0    | 0    | 1
13   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1
| 0    | 147  | 0    | 0    | 0
```

```
14   | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 878  | 12   | 0
15   | 0    | 0    | 0    | 0    | 5    | 9    | 0    | 0    | 0    | 0    | 0
| 1    | 1    | 17   | 231  | 0
16   | 0    | 5    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 2
| 1    | 0    | 0    | 0    | 64
============================================
====== indian - LDA - 15 - naive_bayes =====
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 13   | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0
| 0    | 0    | 0    | 1    | 0
2    | 0    | 370  | 16   | 1    | 0    | 0    | 0    | 0    | 0    | 20   | 23
| 3    | 0    | 0    | 1    | 1
3    | 0    | 17   | 196  | 6    | 0    | 0    | 0    | 0    | 0    | 0    | 20
| 3    | 0    | 0    | 0    | 0
4    | 0    | 0    | 4    | 55   | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
5    | 0    | 0    | 0    | 2    | 142  | 4    | 0    | 0    | 0    | 0    | 1
| 2    | 0    | 0    | 3    | 0
6    | 0    | 0    | 0    | 0    | 0    | 215  | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 3    | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 9    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 154  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 4    | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 8    | 1    | 0    | 0    | 1    | 0    | 0    | 0    | 251  | 38
| 3    | 0    | 0    | 1    | 2
11   | 0    | 61   | 4    | 0    | 4    | 1    | 0    | 0    | 0    | 42   | 624
| 1    | 0    | 0    | 6    | 0
12   | 0    | 3    | 17   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 4
| 135  | 0    | 0    | 1    | 2
13   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 57   | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 363  | 11   | 0
15   | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 21   | 100  | 0
16   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 21
----      Testing Confusion Matrix      ----
Act /    Predictions
```

```
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 31   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 852  | 37   | 11   | 0    | 1    | 0    | 0    | 0    | 39   | 44
| 6    | 0    | 0    | 1    | 2
3    | 0    | 44   | 471  | 16   | 0    | 0    | 0    | 0    | 0    | 0    | 55
| 2    | 0    | 0    | 0    | 0
4    | 0    | 4    | 34   | 132  | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 2    | 0    | 0    | 6    | 0
5    | 0    | 0    | 0    | 2    | 277  | 21   | 0    | 0    | 0    | 1    | 0
| 6    | 0    | 0    | 22   | 0
6    | 0    | 0    | 0    | 0    | 2    | 500  | 0    | 0    | 0    | 3    | 0
| 0    | 0    | 0    | 7    | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 19   | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 323  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 4    | 0    | 1    | 0    | 0    | 8    | 0    | 0
| 0    | 0    | 0    | 3    | 0
10   | 0    | 15   | 2    | 0    | 0    | 2    | 0    | 0    | 0    | 518  | 126
| 3    | 0    | 0    | 0    | 1
11   | 0    | 116  | 38   | 0    | 11   | 1    | 0    | 0    | 0    | 96   |
1430 | 15   | 0    | 0    | 4    | 1
12   | 0    | 5    | 79   | 1    | 0    | 0    | 0    | 0    | 0    | 0    | 7
| 338  | 0    | 0    | 0    | 1
13   | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 147  | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 3    | 1    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 863  | 24   | 0
15   | 0    | 0    | 0    | 0    | 0    | 6    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 28   | 229  | 1
16   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 8    | 0    | 0    | 0    | 64
============================================
====== indian - None - None - svm_rbf ======
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 15   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 435  | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
3    | 0    | 0    | 242  | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
```

```
4    | 0    | 0    | 0    | 59   | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
5    | 0    | 0    | 0    | 0    | 154  | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
6    | 0    | 0    | 0    | 0    | 0    | 218  | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 9    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 154  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 4    | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 305  | 0
| 0    | 0    | 0    | 0    | 0
11   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 743
| 0    | 0    | 0    | 0    | 0
12   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 162  | 0    | 0    | 0    | 0
13   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 57   | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 374  | 0    | 0
15   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 122  | 0
16   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 21
----      Testing Confusion Matrix      ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 31
| 0    | 0    | 0    | 0    | 0
2    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 993
| 0    | 0    | 0    | 0    | 0
3    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 588
| 0    | 0    | 0    | 0    | 0
4    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 178
| 0    | 0    | 0    | 0    | 0
5    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 329
| 0    | 0    | 0    | 0    | 0
6    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 512
| 0    | 0    | 0    | 0    | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 19
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 324
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 16
| 0    | 0    | 0    | 0    | 0
```

```
10   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 667
| 0    | 0    | 0    | 0    | 0
11   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |
1712 | 0    | 0    | 0    | 0    | 0
12   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 431
| 0    | 0    | 0    | 0    | 0
13   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 148
| 0    | 0    | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 891
| 0    | 0    | 0    | 0    | 0
15   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 264
| 0    | 0    | 0    | 0    | 0
16   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 72
| 0    | 0    | 0    | 0    | 0
============================================
====== indian - None - None - svm_poly =====
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 15   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 435  | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
3    | 0    | 0    | 242  | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
4    | 0    | 0    | 0    | 59   | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
5    | 0    | 0    | 0    | 0    | 154  | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
6    | 0    | 0    | 0    | 0    | 0    | 218  | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 9    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 154  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 4    | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 305  | 0
| 0    | 0    | 0    | 0    | 0
11   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 743
| 0    | 0    | 0    | 0    | 0
12   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 162  | 0    | 0    | 0    | 0
13   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 57   | 0    | 0    | 0
```

```
14   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 374  | 0    | 0
15   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 122  | 0
16   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 21
----      Testing Confusion Matrix      ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 31   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 937  | 10   | 2    | 2    | 0    | 0    | 0    | 0    | 14   | 23
| 4    | 0    | 0    | 1    | 0
3    | 0    | 28   | 524  | 13   | 0    | 0    | 0    | 0    | 0    | 0    | 15
| 8    | 0    | 0    | 0    | 0
4    | 0    | 1    | 5    | 171  | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 1    | 0    | 0    | 0    | 0
5    | 0    | 0    | 0    | 0    | 312  | 2    | 5    | 0    | 0    | 6    | 1
| 3    | 0    | 0    | 0    | 0
6    | 0    | 0    | 0    | 0    | 6    | 502  | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 4    | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 19   | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 324  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 16   | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 19   | 1    | 0    | 0    | 1    | 0    | 0    | 0    | 600  | 44
| 2    | 0    | 0    | 0    | 0
11   | 0    | 58   | 19   | 0    | 3    | 0    | 0    | 0    | 0    | 56   |
1570 | 2    | 0    | 0    | 4    | 0
12   | 0    | 4    | 21   | 0    | 0    | 0    | 0    | 0    | 0    | 2    | 2
| 402  | 0    | 0    | 0    | 0
13   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0
| 0    | 147  | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 881  | 9    | 0
15   | 0    | 0    | 0    | 0    | 0    | 8    | 0    | 0    | 2    | 3    | 0
| 10   | 1    | 4    | 236  | 0
16   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0
| 7    | 0    | 0    | 0    | 64
============================================
===== indian - None - None - svm_linear ====
============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix     ----
Act /    Predictions
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	417	0	0	0	0	0	0	0	0	18	0	0	0	0	0
3	0	0	242	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	59	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	154	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	218	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	154	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0
10	0	2	0	0	1	0	0	0	0	280	22	0	0	0	0	0
11	0	37	2	0	2	0	0	0	0	32	670	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	162	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	57	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	374	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	122	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21

---- Testing Confusion Matrix ----

Act / Predictions

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	848	22	2	3	0	0	0	0	25	82	11	0	0	0	0
3	0	29	520	16	0	0	0	0	0	0	13	10	0	0	0	0
4	0	1	6	169	0	0	0	0	0	0	0	2	0	0	0	0
5	0	1	0	0	312	5	4	0	0	3	1	3	0	0	0	0

```
6    | 0    | 0    | 0    | 0    | 4    | 499  | 0    | 0    | 0    | 4    | 0
| 0    | 0    | 0    | 5    | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 19   | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 324  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 16   | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 47   | 2    | 0    | 1    | 0    | 0    | 0    | 0    | 497  | 118
| 2    | 0    | 0    | 0    | 0
11   | 0    | 179  | 29   | 0    | 13   | 0    | 0    | 1    | 0    | 132  |
1347 | 8    | 0    | 0    | 3    | 0
12   | 0    | 5    | 21   | 0    | 1    | 5    | 0    | 0    | 0    | 5    | 3
| 391  | 0    | 0    | 0    | 0
13   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0
| 0    | 147  | 0    | 0    | 0
14   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 882  | 9    | 0
15   | 0    | 2    | 0    | 0    | 4    | 12   | 0    | 0    | 1    | 4    | 0
| 6    | 0    | 6    | 229  | 0
16   | 0    | 2    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 7    | 0    | 0    | 0    | 63
============================================
======== indian - None - None - knn ========
============================================
======        Train Size: 30%         =======
----       Training Confusion Matrix     ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 14   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0
| 0    | 0    | 0    | 0    | 0
2    | 0    | 387  | 13   | 0    | 0    | 1    | 0    | 0    | 0    | 3    | 25
| 6    | 0    | 0    | 0    | 0
3    | 0    | 7    | 222  | 6    | 0    | 0    | 0    | 0    | 0    | 2    | 4
| 1    | 0    | 0    | 0    | 0
4    | 0    | 0    | 3    | 56   | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
5    | 0    | 0    | 0    | 0    | 153  | 0    | 0    | 0    | 0    | 1    | 0
| 0    | 0    | 0    | 0    | 0
6    | 0    | 0    | 0    | 0    | 1    | 215  | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 2    | 0
7    | 0    | 0    | 0    | 0    | 1    | 0    | 8    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 154  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 0    | 3    | 0    | 0    | 1    | 0    | 0
| 0    | 0    | 0    | 0    | 0
```

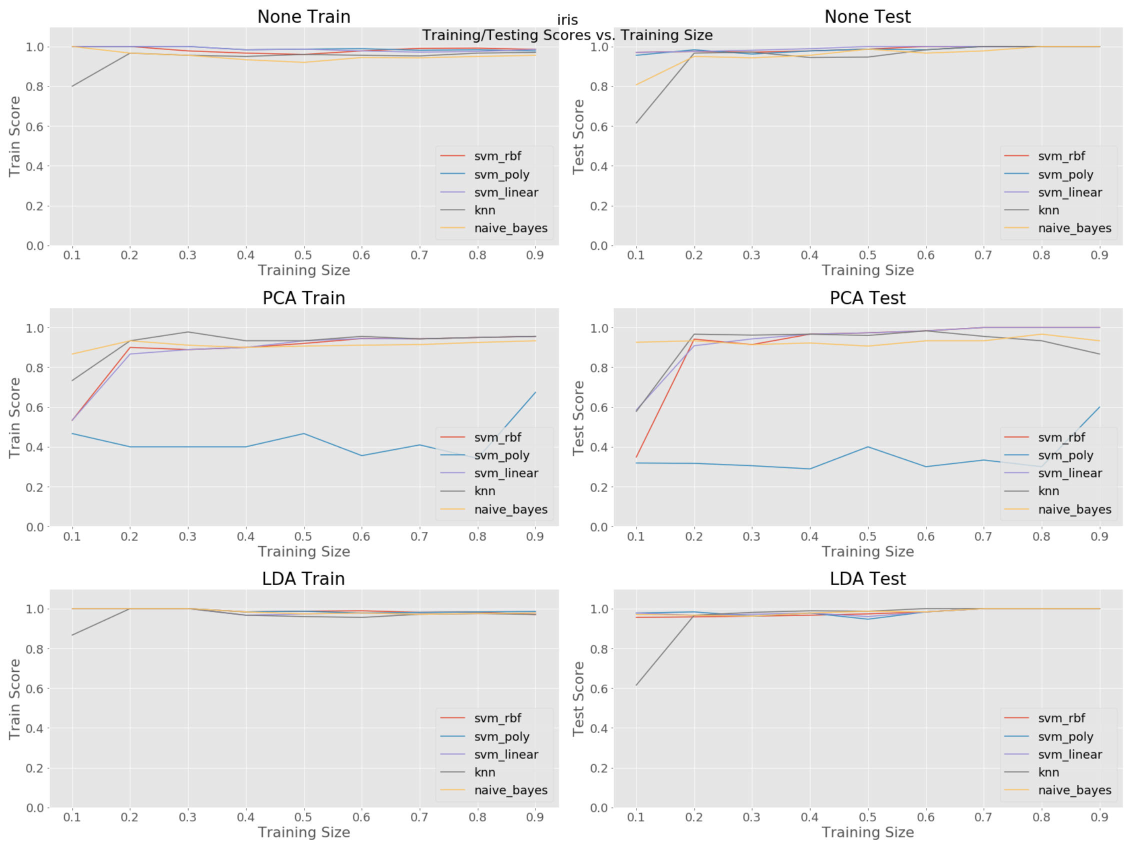
10	0	5	4	0	1	0	0	0	0	286	7	2	0	0	0	0
11	0	17	8	0	1	3	0	0	0	8	706	0	0	0	0	0
12	0	10	2	0	0	1	0	0	0	2	14	133	0	0	0	0
13	0	0	0	0	0	1	0	0	0	0	0	0	56	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	369	5	0
15	0	0	0	0	0	20	0	0	0	0	2	0	5	4	91	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21

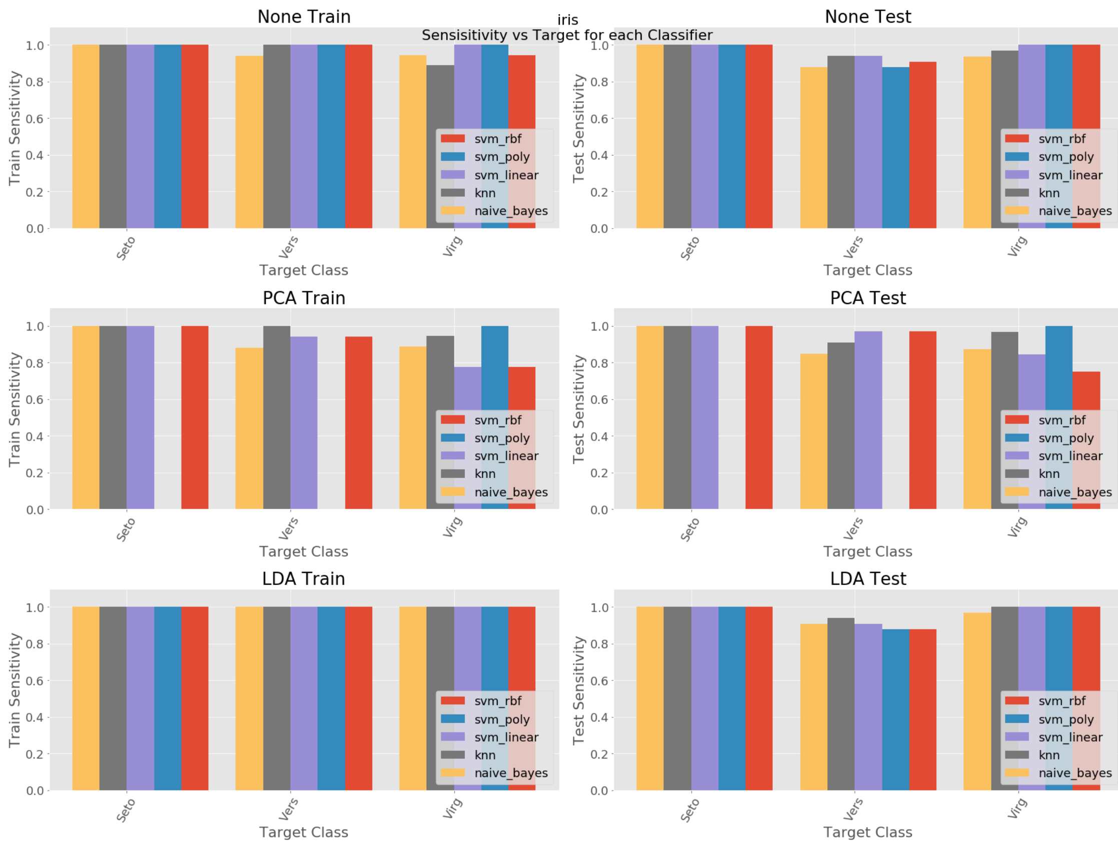
---- Testing Confusion Matrix ----

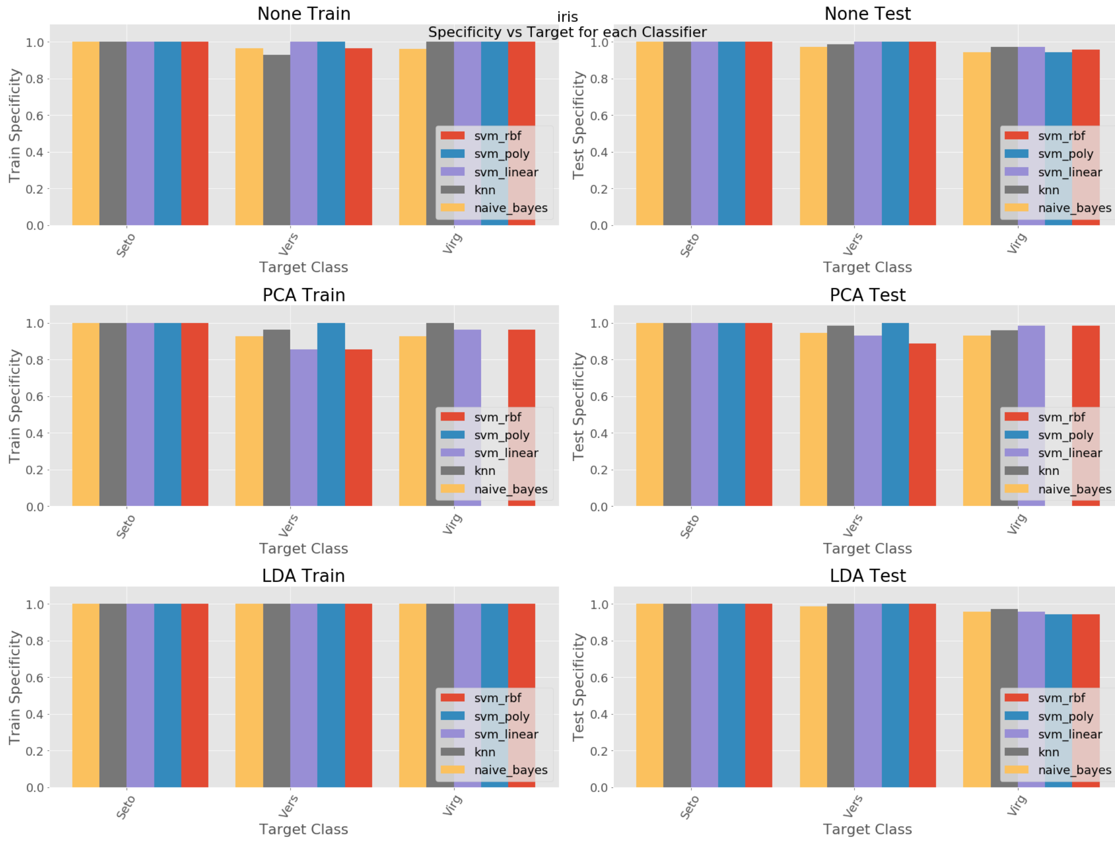
Act / Predictions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	822	55	0	3	0	0	0	0	1	81	31	0	0	0	0
3	0	30	485	22	0	0	0	0	0	6	35	10	0	0	0	0
4	0	0	16	162	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	2	307	3	1	0	0	13	3	0	0	0	0	0
6	0	0	0	0	5	495	0	0	0	0	0	0	0	0	12	0
7	1	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	324	0	0	0	0	0	0	0	0
9	0	0	0	0	0	10	0	0	5	0	0	0	1	0	0	0
10	0	7	8	0	0	3	5	0	0	618	23	3	0	0	0	0
11	0	42	28	1	8	9	0	0	0	25	1590	9	0	0	0	0
12	0	58	12	0	0	1	0	0	0	2	50	308	0	0	0	0
13	0	0	0	0	0	2	0	0	0	1	1	0	144	0	0	0
14	0	0	0	0	0	1	0	0	0	0	0	0	0	873	17	0
15	0	5	2	0	0	65	0	0	0	2	3	3	3	10	171	0

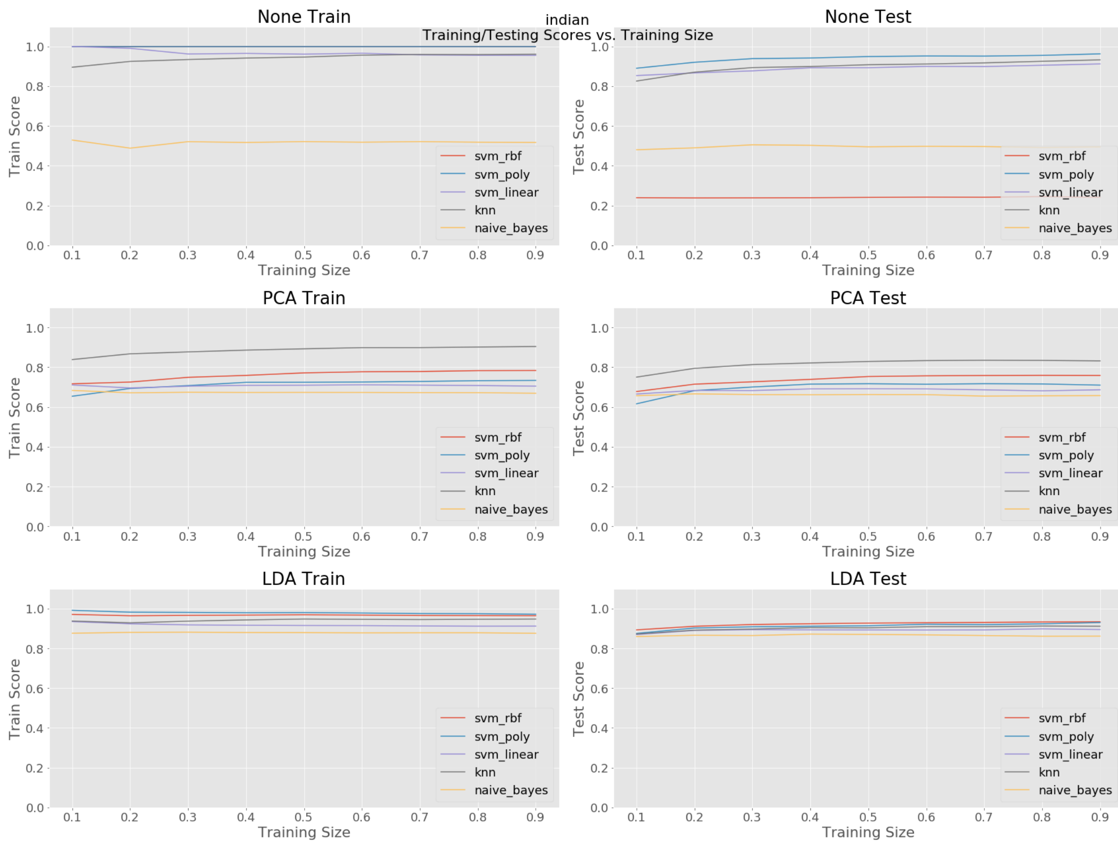
```
16   | 0    | 9    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 3
| 0    | 0    | 0    | 0    | 60
=============================================
==== indian - None - None - naive_bayes ====
=============================================
======       Train Size: 30%         =======
----      Training Confusion Matrix      ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 13   | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0
| 1    | 0    | 0    | 0    | 0
2    | 0    | 226  | 0    | 0    | 0    | 1    | 0    | 2    | 0    | 107  | 75
| 24   | 0    | 0    | 0    | 0
3    | 0    | 46   | 3    | 0    | 0    | 0    | 0    | 0    | 0    | 104  | 58
| 31   | 0    | 0    | 0    | 0
4    | 0    | 20   | 6    | 12   | 0    | 3    | 0    | 0    | 0    | 5    | 3
| 10   | 0    | 0    | 0    | 0
5    | 0    | 0    | 0    | 6    | 4    | 33   | 0    | 3    | 0    | 0    | 0
| 1    | 0    | 106  | 1    | 0
6    | 0    | 0    | 0    | 1    | 9    | 176  | 0    | 0    | 3    | 0    | 0
| 6    | 1    | 0    | 22   | 0
7    | 0    | 0    | 0    | 0    | 0    | 0    | 8    | 1    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
8    | 0    | 0    | 0    | 0    | 0    | 1    | 1    | 152  | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
9    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 4    | 0    | 0
| 0    | 0    | 0    | 0    | 0
10   | 0    | 27   | 0    | 2    | 0    | 0    | 0    | 1    | 0    | 174  | 67
| 34   | 0    | 0    | 0    | 0
11   | 0    | 150  | 4    | 8    | 0    | 2    | 0    | 0    | 0    | 211  | 312
| 55   | 0    | 0    | 1    | 0
12   | 0    | 54   | 3    | 1    | 0    | 4    | 0    | 0    | 0    | 39   | 10
| 51   | 0    | 0    | 0    | 0
13   | 0    | 0    | 0    | 0    | 0    | 3    | 0    | 0    | 0    | 0    | 0
| 0    | 53   | 0    | 1    | 0
14   | 0    | 0    | 0    | 0    | 6    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 362  | 6    | 0
15   | 0    | 0    | 0    | 0    | 11   | 32   | 0    | 0    | 0    | 0    | 0
| 0    | 9    | 37   | 33   | 0
16   | 0    | 2    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 19
----      Testing Confusion Matrix      ----
Act /    Predictions
     | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   | 11
| 12   | 13   | 14   | 15   | 16
1    | 30   | 0    | 0    | 0    | 0    | 0    | 0    | 1    | 0    | 0    | 0
| 0    | 0    | 0    | 0    | 0
```

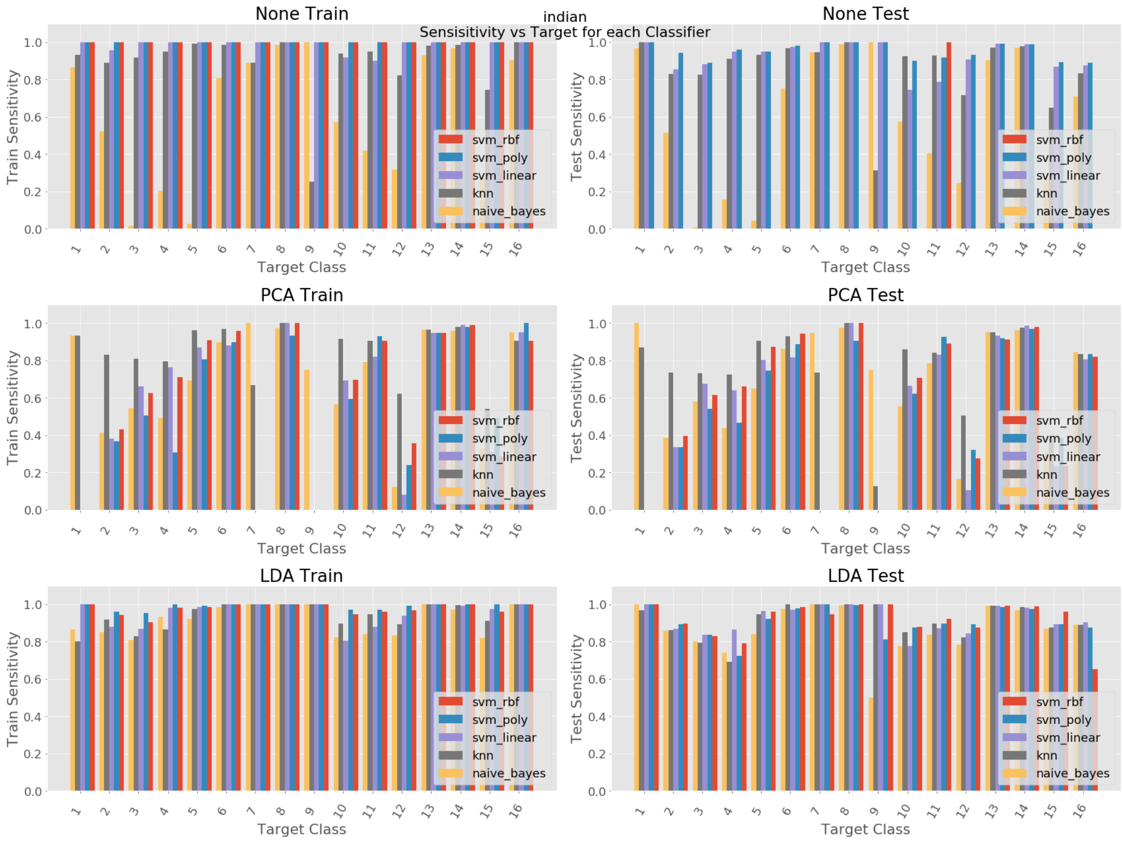
2	0	509	0	2	0	2	0	2	0	234	161	83	0	0	0	0
3	0	131	4	5	0	1	0	0	0	269	126	52	0	0	0	0
4	0	91	4	28	0	12	0	0	0	21	12	10	0	0	0	0
5	1	1	0	24	14	62	1	10	0	1	0	4	0	210	1	0
6	0	0	0	1	31	385	0	0	1	0	0	14	6	1	73	0
7	1	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0
8	1	0	0	0	0	0	2	320	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0
10	0	65	0	1	0	1	2	4	0	382	144	68	0	0	0	0
11	0	390	8	17	0	10	0	2	0	466	694	123	0	0	2	0
12	0	168	1	0	0	5	0	0	0	121	29	106	0	0	1	0
13	0	0	0	1	0	10	0	0	1	0	0	0	134	0	2	0
14	0	0	0	0	11	2	0	0	0	0	0	0	0	864	14	0
15	0	0	0	3	20	82	0	0	3	0	0	8	11	66	71	0
16	0	12	0	4	0	0	0	0	0	0	0	5	0	0	0	51

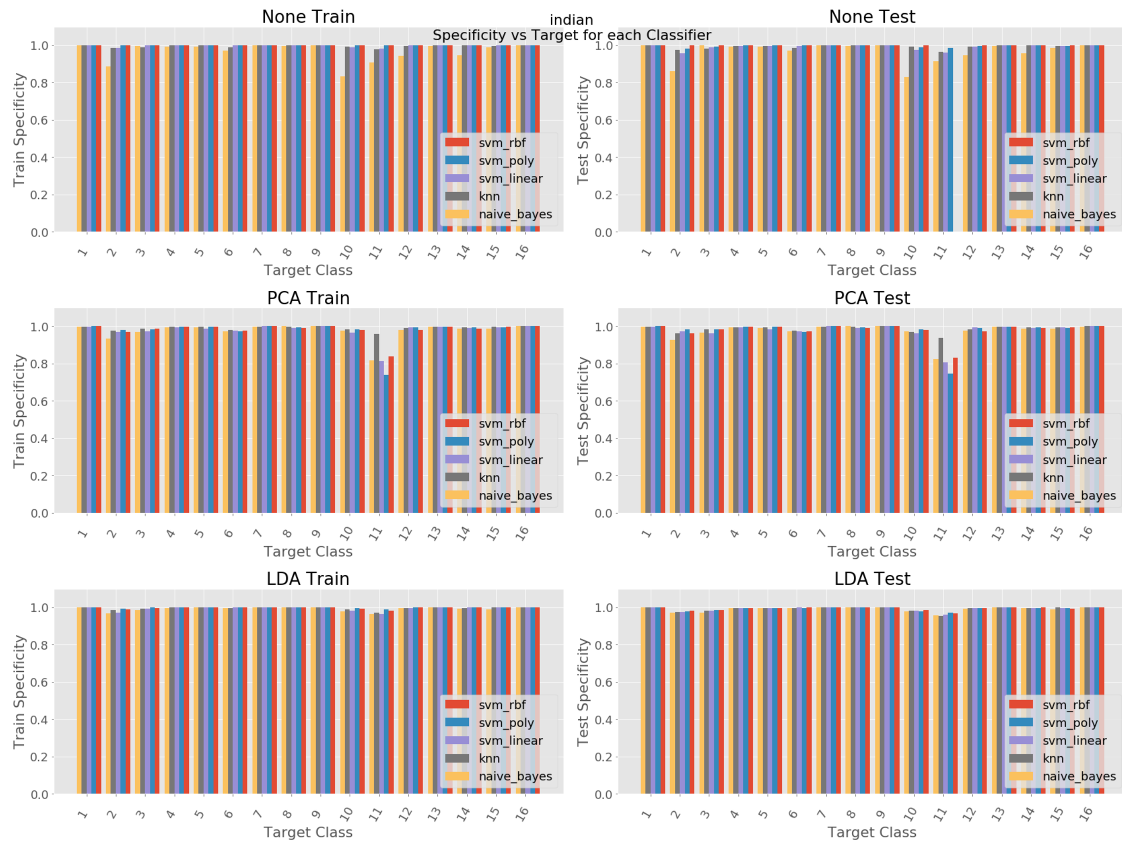

iris
Training/Testing Scores vs. Training Size
None Train
None Test
PCA Train
PCA Test
LDA Train
LDA Test
Train Score
Test Score
Training Size
svm_rbf
svm_poly
svm_linear
knn
naive_bayes


iris
Sensisitivity vs Target for each Classifier
None Train
None Test
PCA Train
PCA Test
LDA Train
LDA Test
Train Sensitivity
Test Sensitivity
Target Class
Seto
Vers
Virg
svm_rbf
svm_poly
svm_linear
knn
naive_bayes
0.0
0.2
0.4
0.6
0.8
1.0


iris
Specificity vs Target for each Classifier
None Train
None Test
PCA Train
PCA Test
LDA Train
LDA Test
Train Specificity
Test Specificity
Target Class
Seto
Vers
Virg
svm_rbf
svm_poly
svm_linear
knn
naive_bayes
0.0
0.2
0.4
0.6
0.8
1.0


indian
Training/Testing Scores vs. Training Size
None Train
None Test
PCA Train
PCA Test
LDA Train
LDA Test
Train Score
Test Score
Training Size
svm_rbf
svm_poly
svm_linear
knn
naive_bayes


indian
Sensisitivity vs Target for each Classifier
None Train
None Test
PCA Train
PCA Test
LDA Train
LDA Test
Train Sensitivity
Test Sensitivity
Target Class
svm_rbf
svm_poly
svm_linear
knn
naive_bayes



b) iris) A PCA of 2 components and LDA of 2 components were chosen based on the information from question 1.

With the iris dataset, there is not a huge difference in performances and runtimes with and without data reduction (apart from the svm-poly on pca data). As long as the train size is over 20%, the performance reaches above 95%. In the end, it is up to personal preference on which learning method to use. Personally, I would choose LDA with 2 dimensions and use a simple knn model to predict. It is simple, easy to understand, fast and highly accurate for the iris dataset. The svm-poly dataset is completely failing with pca and I cannot determine quite determine why. It appears to select everything as Virginica (based on the confusion matrix). All the others perform adequately enough that I do not feel that it is necessary to alter the methods much to improve performance.

indian) Initially I used a PCA and LDA of only 4 components, but the results were not impressive so I upped the number of components to 10, then 15. The PCA has minimal improvement with increased components, so I stuck with 4 (which is what would be expected based on the results from question 1). The LDA saw about 5-10% improvement with increased components from 4 to 15. I ended up choosing 15 components as that uses all of the classes for LDA, and its runtime is still quite fast.

With the indian dataset, we see a much more noticable difference in performance and runtime based on dimensionality reduction. Without dimensionality reduction, the runtimes for the svm learners on the dataset are quite intensive. I would not use any of the svms on the unreduced data. The learners sensitivity versus target appear much improved with LDA vs PCA or no dimensionality reduction. The specificity seems to excel in most cases, except for the svm rbf and naive bayes with no dimensionality reduction in certain classes (class 11 being a bad instance) and svm poly appears to struggle in specificity with the PCA data. Based on the plots, PCA is not a good approach for the indian dataset for most learners (though it does seem to do quite well with knn). Most likely, there is significant overlap in the data (which is shown in the first question), and merely using variances to determine principal components is not enough. LDA is able to perform better because it takes into account target classes to try and maximize the inter- and intra- class means. LDA appears to be the right approach as it has drastically faster runtimes and comparable performance with using the whole dataset. The best classifier in terms of classification accuracy is svm_rbf, but they all perform about the same. Looking at specificicity and sensitivity, they all seem to do similar as well. Each learner seems to have one or two classes it particularly struggles with (ie. naive bayes struggles with class 9). An ensemble majority vote learner using knn, naive bayes and svm_rbf might me an interesting and more fruitful approach to combat this dilemma.