**Sarp Solakoglu 234308005 MidTerm**

(github: <https://github.com/memedsarp/Advanced_ML_MidTerm>)

**1. Preprocessing**

Firstly, I’ve created some arbitrary null values for two columns at 5%, and one column at 35%. Then, in order to fill these null values, I checked for which features (columns) to be used by using correlation between features. For the columns “Area”, “ConvexArea” column seems like a perfect match with correlation value of 0.99939. For “Perimeter” column, on the other hand, “EquivDiameter” would serve for the similar purpose. In case there was no perfect correlation, as was in the case for the “Eccentricity” column, we could always resort for regression whenever we need to approximate a numerical value, and for frequency analysis or Bayes for classification. For the case we have falls for the former, I applied multilinear regression with the following columns “ShapeFactor2", "ShapeFactor3", "AspectRation", "Compactness". These features have the following coefficiency values: 0.3643494 , -7.55600922, 0.44880671, 12.42836078, and the model has the intercept value of -5.038314069424181, all of which can be seen on the outputs of the codes. This model has the score of .998592, thus, we can tell it is a very accurate approximation, thus, there is no need to drop the column.

Secondly, I applied z-score normalization in order to keep the data within 1.96 spread (thus re-evaluating roughly the .05 of the data). I, then applied StandardScaler to each numeric feature, and encoding for the only non-numeric feature in the dataset, namely, the column “Class”.

**2. Feature Scaling**

After preprocessing the data, I applied two transformation to the data, Principal Component Analysis, and Linear Discriminant Analysis, and in the end, had three usable data for the machine learning phase. For the PCA, I included two dimensions, and for the LDA, 3.

**3. Modeling and Evaluation Metrics**

The models used for these three datasets are Logistic Regression, Gaussian Naive Bayes, Decision Trees, Random Forest, and Gradient Boosting. Each model is applied to the same train/test split. Below are the metrics for each. The order of the models appear as given above:

*Accuracy:*

{0: [0.9294300979519146, 0.8593054318788959, 0.890026714158504],

1: [0.8853517364203027, 0.8486197684772929, 0.8840160284951024],

2: [0.9590382902938558, 0.828806767586821, 0.8909171861086376],

3: [0.9710596616206589, 0.846393588601959, 0.9203027604630454],

4: [0.9746215494211933, 0.8515138023152271, 0.9180765805877115]}

*Precision (Each key is different model and and each sub array is different dataset):*

{0: [array([0.97090352, 0.96478873, 1. , 0.90070922, 0.975 ,

0.85394456, 0.93333333]),

array([0.95807453, 0.68965517, 1. , 0.69618056, 0.96656051,

0.83765502, 0.87687188]),

array([0.81068702, 0.93607306, 1. , 0.91917293, 0.9734375 ,

0.87486157, 0.85253054])],

1: [array([0.95129376, 0.80128205, 1. , 0.85657371, 0.95962733,

0.82896237, 0.87821612]),

array([0.9591195 , 0.69255663, 1. , 0.67475728, 0.94907407,

0.82071269, 0.86390041]),

array([0.80267062, 0.9138322 , 1. , 0.89279113, 0.97464342,

0.83586957, 0.8870822 ])],

2: [array([0.98355755, 0.91360691, 1. , 0.94827586, 0.99843505,

0.94131185, 0.953125 ]),

array([0.94736842, 0.56823821, 1. , 0.62852113, 0.94992175,

0.83036773, 0.85596708]),

array([0.87284144, 0.91051454, 1. , 0.89757914, 0.97622821,

0.85454545, 0.85532995])],

3: [array([0.99088146, 0.98148148, 1. , 0.9558011 , 0.996875 ,

0.95454545, 0.95693368]),

array([0.95524691, 0.63072776, 1. , 0.66101695, 0.9514867 ,

0.84173505, 0.86974444]),

array([0.87201125, 0.93378995, 1. , 0.9249531 , 0.97356143,

0.90983607, 0.90837004])],

4: [array([1. , 0.98383372, 1. , 0.95421245, 1. ,

0.95469989, 0.96369922]),

array([0.95503876, 0.66850829, 1. , 0.67567568, 0.953125 ,

0.84551887, 0.86552567]),

array([0.86789773, 0.925 , 1. , 0.91992551, 0.975039 ,

0.90034762, 0.91408326])]}

*Recall:*

{0: [array([0.95481928, 0.91333333, 0.90449438, 0.96212121, 0.97805643,

0.9112628 , 0.8969697 ]),

array([0.92921687, 0.57777778, 1. , 0.7594697 , 0.95141066,

0.84527873, 0.91255411]),

array([0.7996988 , 0.91111111, 1. , 0.92613636, 0.97648903,

0.89874858, 0.84588745])],

1: [array([0.94126506, 0.83333333, 1. , 0.81439394, 0.96865204,

0.82707622, 0.88658009]),

array([0.9186747 , 0.47555556, 1. , 0.78977273, 0.96394984,

0.83845279, 0.9012987 ]),

array([0.81475904, 0.89555556, 1. , 0.91477273, 0.96394984,

0.87485779, 0.85021645])],

2: [array([0.99096386, 0.94 , 1. , 0.9375 , 1. ,

0.93060296, 0.95064935]),

array([0.92168675, 0.50888889, 1. , 0.67613636, 0.95141066,

0.7963595 , 0.9004329 ]),

array([0.8373494 , 0.90444444, 1. , 0.91287879, 0.96551724,

0.85551763, 0.87532468])],

3: [array([0.98192771, 0.94222222, 1. , 0.98295455, 1. ,

0.9556314 , 0.96190476]),

array([0.93222892, 0.52 , 1. , 0.73863636, 0.95297806,

0.81683732, 0.91341991]),

array([0.93373494, 0.90888889, 1. , 0.93371212, 0.98119122,

0.88395904, 0.89264069])],

4: [array([0.99096386, 0.94666667, 1. , 0.98674242, 0.9984326 ,

0.95904437, 0.96536797]),

array([0.92771084, 0.53777778, 1. , 0.75757576, 0.95611285,

0.81569966, 0.91948052]),

array([0.92018072, 0.90444444, 1. , 0.93560606, 0.97962382,

0.88395904, 0.89350649])]}

*F1:*

{0: [array([0.96279423, 0.93835616, 0.94985251, 0.93040293, 0.97652582,

0.88167309, 0.91479029]),

array([0.94342508, 0.62877872, 1. , 0.72644928, 0.95892575,

0.8414496 , 0.89435723]),

array([0.80515542, 0.92342342, 1. , 0.92264151, 0.97496088,

0.88664422, 0.849196 ])],

1: [array([0.94625284, 0.81699346, 1. , 0.83495146, 0.96411856,

0.82801822, 0.88237829]),

array([0.93846154, 0.56389987, 1. , 0.72774869, 0.95645412,

0.8294879 , 0.88220339]),

array([0.80866966, 0.90460157, 1. , 0.90364827, 0.96926714,

0.8549194 , 0.86825818])],

2: [array([0.98724681, 0.92661555, 1. , 0.94285714, 0.99921691,

0.93592677, 0.95188557]),

array([0.93435115, 0.53692849, 1. , 0.65145985, 0.95066562,

0.81300813, 0.87763713]),

array([0.85472713, 0.90746934, 1. , 0.90516432, 0.97084318,

0.85503127, 0.86521181])],

3: [array([0.98638427, 0.96145125, 1. , 0.96918768, 0.99843505,

0.95508812, 0.95941278]),

array([0.94359756, 0.57003654, 1. , 0.69767442, 0.95223179,

0.82909931, 0.8910473 ]),

array([0.90181818, 0.92117117, 1. , 0.92931197, 0.97736144,

0.89671091, 0.90043668])],

4: [array([0.99546142, 0.96489241, 1. , 0.97020484, 0.99921569,

0.9568672 , 0.96453287]),

array([0.94117647, 0.59605911, 1. , 0.71428571, 0.95461659,

0.83034163, 0.89168766]),

array([0.89327485, 0.91460674, 1. , 0.92769953, 0.97732604,

0.89207807, 0.90367776])]}

*Mean and Standard Deviation:*

First Dataset (Raw) has .943 mean for accuracy with the standard deviation of .003. It has the highest accuracy with the least variation, making this dataset much better than the PCA and LDA. The same pattern can be observed when precision, recall, and F1 scores are calculated. We have the mean of .951, .948, and .950, and the standard deviation of .03. .031, .030, respectively. In each case, the metrics are for about .05 more than the second best dataset.

**4. ROC-AUC**

Establishing that the raw dataset is the best for modeling, the second task is to determine the best estimating model, which is, in this case GradientBoosting with the accuracy of .974.

Below are the ROC-AUC curves for each class versus all.













