ELL409 Assignment 1

Amogh Agrawal
2018EE10441

Shauryasikt Jena
2018EE10500

***Abstract*—The purpose of this report is to present our findings on applying various algorithms a binary classification problem, a regression problem and a multi-class classification problem. The purpose of this exercise is to understand the theory behind some basic pattern recognition algorithm and appreciate their importance by implementing them from scratch in a world dominated by neural networks and off-the-shelf libraries.**

***Index Terms*—theory, pattern recognition, classification, regression**

I. Problem 1: Binary Classification

We were given health data of 700 individuals, classified into 2 categories '0' and '1' based on their age in years, resting pulse, and cholesterol levels. In approaching this problem of binary classification, we employed several Machine Learning techniques such as Naive Bayes Classifier, Bayes Classifier, K-Nearest Neighbours, Parzen Window Density Estimates, Linear Models, and Generalized Linear Models such as Logistic Regressor and Perceptron.

A. Naive Bayes Classifier

As is the case with Naive Bayes Classifier, we assumed that all the features are independent of one another. To estimate maximum likelihood, we aggregated the class probabilities for each feature by taking their product. For MAP estimate, we also incorporated the relative frequencies of the classes in the training set.
For plotting ROC, we varied the threshold for class predictions to accumulate a series of *False Positive Rates* and corresponding *True False Positive Rates.*

```
Train Accuracy: 0.8571428571428571
Test Accuracy: 0.8571428571428571
Mean Precision: 0.83497717747114
Mean Recall: 0.8349369412515963
Mean F1: 0.8333046797745194
Mean Specificity: 0.8752475594829388
```

Fig. 1. Correctness Metrics: Naive Bayes (MLE)

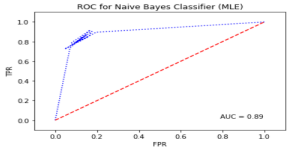


Fig. 2. ROC: Naive Bayes (MLE)

```
Train Accuracy: 0.8574999999999999
Test Accuracy: 0.8557142857142856
Mean Precision: 0.8387577911427011
Mean Recall: 0.8230072400927847
Mean F1: 0.8287714071028025
Mean Specificity: 0.8816829830752171
```

Fig. 3. Correctness Metrics: Naive Bayes (MAP)

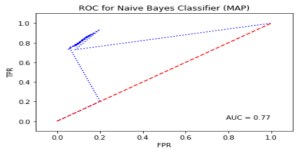


Fig. 4. ROC: Naive Bayes (MAP)

B. Bayes Classifier

For Bayes Classifier, we assumed that the features are dependent on one another in some capacity. For plotting ROC, we varied the threshold for class predictions to accumulate a series of *False Positive Rates* and corresponding *True False Positive Rates.*
Bayes Classifier works with MLE as its basis here. We employed this algorithm with 2 kinds of class conditional probabilities as follows:

```
Train Accuracy: 0.8585714285714285
Test Accuracy: 0.8514285714285714
Mean Precision: 0.8018564372983474
Mean Recall: 0.8665936101250192
Mean F1: 0.8311238073854648
Mean Specificity: 0.8417318225865685
```

Fig. 5. Correctness Metrics: Bayes Classifier (Gaussian)

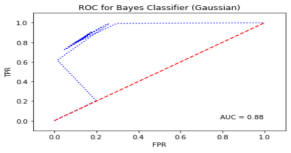


Fig. 6. ROC: Bayes Classifier (Gaussian)

C. K-Nearest Neighbours

For any given sample in the test set, we mapped its distances with the nearest k samples from the training set. The class of samples having maximum neighbours in the test sample's k

```
Train Accuracy: 0.8578571428571428
Test Accuracy: 0.8542857142857143
Mean Precision: 0.7942549731676218
Mean Recall: 0.8913685259648763
Mean F1: 0.8385275528339384
Mean Specificity: 0.8277394065439356
```

Fig. 7. Correctness Metrics: Bayes Classifier (Gaussian Mixture Models: EM Algorithm

nearest neighbours was predicted to be the class for the test sample as well.
For plotting ROC, we varied the number of nearest neighbours for any given test sample.

```
Mean Train Accuracy: 0.8667857142857143
Mean Test Accuracy: 0.8171428571428573
Mean Precision: 0.7943908144995101
Mean Recall: 0.757582286877559
Mean F1 score: 0.770847722774952
Mean Specificity: 0.8598672467648371
```

Fig. 8. Correctness Metrics: KNN

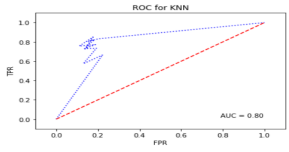


Fig. 9. ROC: KNN

D. Parzen Window Density Estimates

For Parzen estimate, we employed a hypersphere instead of a hypercube for more well-rounded dimensionality w.r.t. the test sample. In this case, we varied the radius of this hypersphere to get our ROC Curve.

```
Mean Train Accuracy: 0.8571428571428571
Mean Test Accuracy: 0.8571428571428573
Mean Precision: 0.8388199382317028
Mean Recall: 0.8518754774637127
Mean F1 score: 0.8447395016581997
Mean Specificity: 0.8630480480480479
```

Fig. 10. Correctness Metrics: Parzen Density

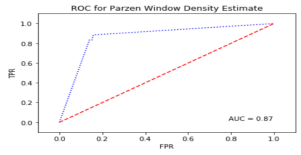
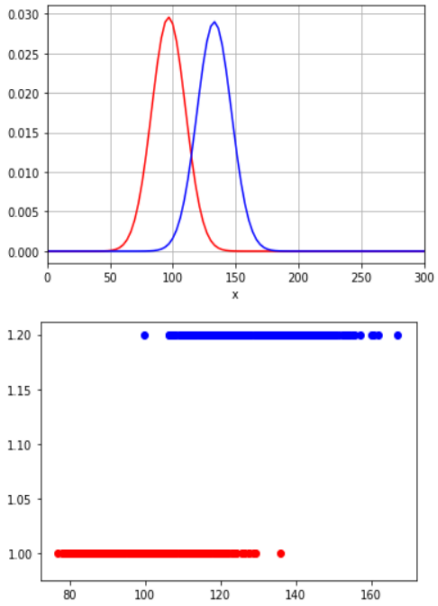


Fig. 11. ROC: Parzen Density

E. Linear Discriminant Analysis

We employed Fischer's LDA algorithm for this binary classification problem to not wholly satisfying results. However, we gained an important insight as to the limitations of LDA.




```
Threshold: 0.0
Accuracy: 0.6357142857142857
F1-score: 0
```

Fig. 12. Fischer LDA

F. Generalized Linear Models

(Discussed in more detail in the following section: Regression)

```
Running Logistic Regression with no regularization and with loss function: log
A polynomial kernel of degree  2  is used to transform feature set
Mean Train Accuracy: 0.8574999999999999
Mean Test Accuracy: 0.8542857142857143
Mean Precision: 0.7963943241842859
Mean Recall: 0.8836006683375104
Mean F1 score: 0.8368613364725866
Mean Specificity: 0.8339923329682366
Running Logistic Regression with no regularization and with loss function: mse
A polynomial kernel of degree  2  is used to transform feature set
Mean Train Accuracy: 0.8596428571428572
Mean Test Accuracy: 0.8371428571428572
Mean Precision: 0.8368633584741125
Mean Recall: 0.7807659582910773
Mean F1 score: 0.8063022554551248
Mean Specificity: 0.8812443153768544
Running Logistic Regression with no regularization and with loss function: mae
A polynomial kernel of degree  2  is used to transform feature set
Mean Train Accuracy: 0.8710714285714285
Mean Test Accuracy: 0.8200000000000001
Mean Precision: 0.7537892041514624
Mean Recall: 0.8188719133256338
Mean F1 score: 0.7805250384690853
Mean Specificity: 0.8230885473985854
```

Fig. 13. Correctness Metrics: Logistic Regressor

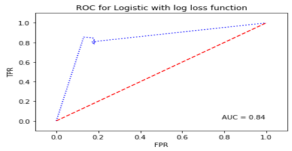


Fig. 14. ROC: Logistic Regressor

G. Bias-Variance Trade-off

We will discuss the bias variance trade-off in a little more detail once we introduce generalized linear models in the regression section. Here, we will look at the test and training MSE errors with an 80-20 train-test split on logistic regression

```
Running Perceptron Regression with no regularization and with loss function: log
A polynomial kernel of degree  1  is used to transform feature set
Mean Train Accuracy: 0.8539285714285715
Mean Test Accuracy: 0.8657142857142857
Mean Precision: 0.823502118022666
Mean Recall: 0.9180685980685983
Mean F1 score: 0.8676557248969985
Mean Specificity: 0.8186151866151867
```

Fig. 15. Correctness Metrics: Perceptron

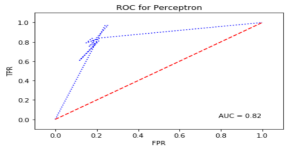


Fig. 16. ROC: Perceptron

minimizing log loss. We have also collected MAE and RMSE error data along with r2 squared data for these loss functions along with all of this for MSE loss and MAE loss. Further, we also have collected this data for the perceptron algorithm, which is quite similar to that of logistic regression under log loss. We won't have the space to present that data here.

Degree	Training Erro	Test Error
1	0.03148155	0.72002498
2	0.02957696	0.71895741
3	0.02866136	0.73057667
4	0.02720762	0.72639894
5	0.02666128	0.72012125
6	0.02619776	0.71753907
7	0.02578093	0.71668219
8	0.02538738	0.71637131
9	0.02500931	0.71584725
10	0.02462778	0.71572838

Fig. 17. Unregularized Logistic Regression

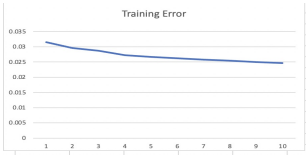


Fig. 18. Unregularized Logistic Regression Training MSE Error

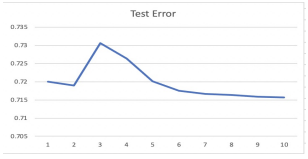


Fig. 19. Unregularized Logistic Regression Test MSE Error

Degree	Training Erro	Test Error
1	0.03599884	0.67508275
2	0.03333129	0.68352208
3	0.03288249	0.69045265
4	0.03253908	0.68846067
5	0.03254395	0.69510012
6	0.03221192	0.69252979
7	0.03217119	0.69340908
8	0.03199205	0.69087307
9	0.03192846	0.69069991
10	0.03187251	0.69035879

Fig. 20. Ridge Logistic Regression

Fig. 21. Ridge Logistic Regression Training MSE Error

Fig. 22. Ridge Logistic Regression Test MSE Error

Degree	Training Erro	Test Error
1	0.03367466	0.69447631
2	0.03293592	0.69894352
3	0.03291394	0.69837105
4	0.0329172	0.69872647
5	0.03289519	0.69794305
6	0.03288587	0.69795383
7	0.03291314	0.69777042
8	0.03290362	0.69780705
9	0.03289388	0.69780387
10	0.03289773	0.69792346

Fig. 23. Lasso Logistic Regression

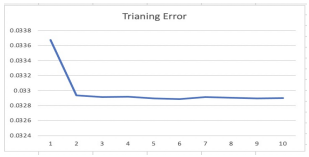


Fig. 24. Lasso Logistic Regression Training MSE Error

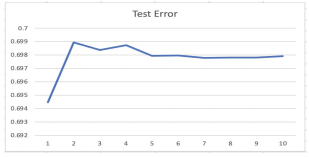


Fig. 25. Lasso Logistic Regression Test MSE Error

Degree	Training Erro	Test Error
1	0.036844	0.6694855
2	0.0346034	0.6692753
3	0.0343855	0.6750885
4	0.0342184	0.6712436
5	0.0342369	0.6721066
6	0.034188	0.6722494
7	0.0341464	0.6720165
8	0.0341116	0.6716644
9	0.0341148	0.6716631
10	0.0340734	0.6716353

Fig. 26. Elastic Net Logistic Regression

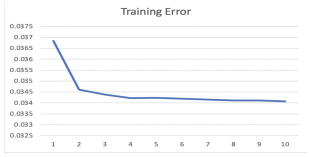


Fig. 27. Elastic Net Logistic Regression Training MSE Error

II. Problem 2: Regression

In this supervised learning problem, we were given 1000 weather data points (dew point, humidity, wind speed, Air

Fig. 28. Elastic Net Logistic Regression Test MSE Error

Pressure and the presence or absence of rain and smoke) to predict the temperature. Here, we were given the labels (the continuous temperature values in Celsius) for each data point and our task was to come up with a model to predict the labels (temperature) from the feature set (the other parameters). Since, we didn't have unknown data, to test the robustness of our model, we used train-test split and cross-validation with 5 folds interchangeably according to the situation.

The most obvious choice for predicting such continuous valued outputs is linear regression (or polynomial regression). We started off with a simple linear regressor on the MSE loss function with a constant learning rate, but realized that convergence was poor (it diverged or was too slow to converge). We realized that this was due to values being uneven compared to others and decided to have standardization and normalization functions to take care of this. We also added adaptive learning rate to ensure convergence is faster far away from the minima but slower as we reach closer to it. We then created different functions for batch gradient descent with different loss functions. After this point, we realised that too many other things were required (like stochastic gradient descent, newton's method and polynomial regression) that we decided to make a general linear model which could do all of this from a single class.

A. General Linear Model

The general linear model created took as input the degree of the polynomial (where 1 meant linear regression), the loss function, the convergence type, the learning type (adaptive or constant) and various other things like whether standardization was required or not. We were able to do this because we separated out a function to calculate gradient depending on the loss function. This further allowed us to add perceptron and logistic regressors and later regularization as the required small modifications to the gradient function. The normal equations and newton method functions weren't made as robust as they aren't very important in real life.

B. Linear Regression Examples

Thereafter, we played around with different loss functions, convergence types, polynomials, regularizers etc. in linear (polynomial) regression to get a feel for what was going on. We found that MSE and RMSE were pretty similar, which is expected since their gradients differ by a factor of the RMSE error, while the MAE offers decent performance but not as good as MSE which performed best. We also saw a general trend of decreasing training error as we increased model complexity (degree of the polynomial in this case). We weren't able to reach the region where test error started increasing, but we do see that test error is lower for regularized regressors (although their corresponding training errors increased). This will be discussed in detail in the section about bias-variance trade-offs. Results of some experiments (in the form of errors) are present in the Jupyter notebooks. Further, the data from bias variance trade-off tables and curved would cover the data explored through these experiments and thus it would be redundant to present.

C. Bias-Variance Trade-off

The Bias-Variance Trade-off comes from the fact that risk, if taken an expectation over all possible data of a given size, decomposes into two terms, one which is the bias gives a measure of how close the prediction matched the training data and another term variance which measures the sensitivity to data sets. It turns out, for finite samples, these terms have an inverse relation, that is, on reducing one, the other is bound to increase. Given how bias is defined, we can see that it is very close to the empirical risk, which is similar to the training error. Further, we can approximate the true error with the test error. Here, we go for a simple 80-20 test train split, and measure the training error and test error as we increase the degree of the polynomial for a few combinations of regularizer coefficients for all the given loss functions. We will discuss the results of stochastic gradient descent on MSE loss with four types of regularization (regularized, l1, l2, l1-l2).

In general, we can see that the test error is quite a bit greater than the training error, which is quite obvious since the empirical risk doesn't capture the variance term. We can also see that all the regularizations put a penalty on the training error, where it was able to reduce to as low as 0.05 in the unregulated case, which was a difference at least one order from all the others. Further, we do see quite a benefit to all the regularizations for smaller model size (smaller degree polynomial), which tends to reduce as the degree of the polynomial increased. This suggests that this penalty is not enough for higher order polynomials. Although, we didn't see the test error increase a whole lot in the unregularized case, meaning we haven't really over fit the model with a 10 degree polynomial which is the maximum we went with (or the maximum our laptops could handle since we were running each gradient descent for almost 100000 iterations)

Degree	Training Erro	Test Error
1	7.71367793	6.75281317
2	2.95444807	3.2381517
3	0.68895604	2.71970717
4	0.25390911	2.32703511
5	0.11473942	1.9413872
6	0.07957985	1.91394545
7	0.07011944	1.89794333
8	0.06282963	1.89765946
9	0.05926031	1.90249248
10	0.05671705	1.90298191

Fig. 29. Unregularized Linear Regression

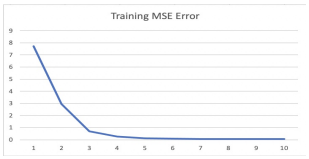


Fig. 30. Unregularized Linear Regression Training MSE Error

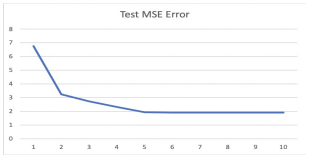


Fig. 31. Unregularized Linear Regression Test MSE Error

Degree	Training Erro	Test Error
1	7.72330174	6.80038338
2	3.04932355	3.24853214
3	1.15168716	1.78164045
4	0.97578966	1.75389991
5	0.8734226	1.73059914
6	0.83056637	1.74484628
7	0.80618577	1.74903253
8	0.79291774	1.75801428
9	0.7886199	1.75995282
10	0.78386511	1.76395818

Fig. 32. Ridge Linear Regression

Fig. 33. Ridge Linear Regression Training MSE Error

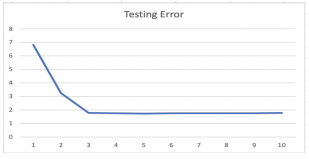


Fig. 34. Ridge Linear Regression Test MSE Error

Degree	Training Erro	Test Error
1	7.71329568	6.7496961
2	2.96490554	3.20745304
3	0.8375619	1.69184889
4	0.52469499	1.89874428
5	0.51037323	1.93643065
6	0.4982767	1.94264911
7	0.47036583	1.93688557
8	0.46969766	1.93114425
9	0.47143343	1.93571229
10	0.47326371	1.93986759

Fig. 35. Lasso Linear Regression

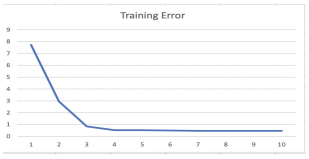


Fig. 36. Lasso Linear Regression Training MSE Error

III. Problem 3: Multi-Class Classification

We were given MNIST Medical Data of around 60,000 images from 6 categories: AbdomenCT, BreastMRI, ChestCT, CXR, Hand and HeadCT. In approaching this problem of multi-class classification, we employed several Machine Learning techniques such as Naive Bayes Classifier, Bayes Classifier, K-Nearest Neighbours, and Parzen Window Density Estimates.We refrained from linear models due to their relatively poor performance in multi-class classification.

A. Naive Bayes Classifier

B. Bayes Classifier

C. K-Nearest Neighbours

D. Parzen Window Density Estimates

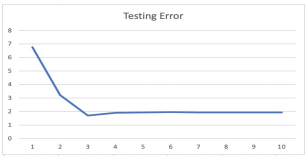


Fig. 37. Lasso Linear Regression Test MSE Error

Degree	Training Erro	Test Error
1	7.71876615	6.76501247
2	3.0658527	3.23150358
3	1.22908886	1.8012458
4	1.11154804	1.79649269
5	1.05065296	1.79329985
6	1.03645616	1.8111526
7	1.02834617	1.82466009
8	1.0246229	1.82775699
9	1.02246253	1.82974518
10	1.02143637	1.82954692

Fig. 38. Elastic Net Linear Regression

Fig. 39. Elastic Net Linear Regression Training MSE Error

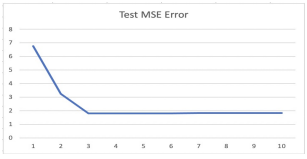


Fig. 40. Elastic Net Linear Regression Test MSE Error

IV. Conclusion

This Assignment was a true test of ML theory, where we were deriving concepts learnt in class and revising them to make the most efficient ML models. Through this assignment, we were truly able to appreciate the effort it takes to ensure well functioning algorithms. Given the time taken to implement already existing algorithms, this further intrigues us about the work involved in developing something from scratch. Although, we couldn't have a project this semester, assignments like these do compensate to some extent (although taking no less time)

```
For Class Type 0
Train Accuracy: 0.8571428571428571
Test Accuracy: 0.8571428571428571
Mean Precision: 0.8311456607620746
Mean Recall: 0.834114774114774
Mean F1: 0.8308888576676352
Mean Specificity: 0.8766988914975185
For Class Type 1
Train Accuracy: 0.8571428571428571
Test Accuracy: 0.8571428571428571
Mean Precision: 0.8327907110956818
Mean Recall: 0.8319971139551594
Mean F1: 0.831905126234998
Mean Specificity: 0.8730873678354698
For Class Type 2
Train Accuracy: 0.8585714285714285
Test Accuracy: 0.8514285714285714
Mean Precision: 0.8284048365330199
Mean Recall: 0.8278262130815512
Mean F1: 0.8271113489481763
Mean Specificity: 0.8693468676009051
For Class Type 3
Train Accuracy: 0.8574999999999999
Test Accuracy: 0.8557142857142856
Mean Precision: 0.8298896221338279
Mean Recall: 0.8318066209829584
Mean F1: 0.8297940576736078
Mean Specificity: 0.874209814144792
For Class Type 4
Train Accuracy: 0.8582142857142857
Test Accuracy: 0.8528571428571429
Mean Precision: 0.8316970997812134
Mean Recall: 0.8272927454595974
Mean F1: 0.8281484000760597
Mean Specificity: 0.8722851577524933
For Class Type 5
Train Accuracy: 0.8582142857142857
Test Accuracy: 0.8528571428571429
Mean Precision: 0.8284639767955364
Mean Recall: 0.8260608549757069
Mean F1: 0.8263269143595876
Mean Specificity: 0.873299313829919
Macro F1-score: 0.8301240950589683
```

Fig. 41. Correctness Metrics: Naive Bayes (MLE)

```
For Class Type 0
Train Accuracy: 0.8567857142857143
Test Accuracy: 0.8585714285714285
Mean Precision: 0.8455963225768766
Mean Recall: 0.8197324574850476
Mean F1: 0.8298173923921768
Mean Specificity: 0.8878279905649226
For Class Type 1
Train Accuracy: 0.8582142857142857
Test Accuracy: 0.8528571428571429
Mean Precision: 0.8332396204332854
Mean Recall: 0.820987871698201
Mean F1: 0.8261208873333921
Mean Specificity: 0.8788205360115361
For Class Type 2
Train Accuracy: 0.8578571428571429
Test Accuracy: 0.8542857142857143
Mean Precision: 0.8379753136245922
Mean Recall: 0.8174671412569146
Mean F1: 0.8260512127304768
Mean Specificity: 0.8832505190173082
For Class Type 3
Train Accuracy: 0.8564285714285713
Test Accuracy: 0.86
Mean Precision: 0.8456294214668981
Mean Recall: 0.8281205399768832
Mean F1: 0.8350420633426223
Mean Specificity: 0.885447784636901
For Class Type 4
Train Accuracy: 0.8571428571428573
Test Accuracy: 0.8571428571428571
Mean Precision: 0.8445664048707376
Mean Recall: 0.81657867729669
Mean F1: 0.8300340841189536
Mean Specificity: 0.8869678398169978
For Class Type 5
Train Accuracy: 0.8575000000000002
Test Accuracy: 0.8557142857142856
Mean Precision: 0.8396780303030305
Mean Recall: 0.8142436068610159
Mean F1: 0.8263507700940534
Mean Specificity: 0.88651499014005
Macro F1-score: 0.8301775722753378
```

Fig. 42. Correctness Metrics: Naive Bayes (MAP)

```
For Class Type 0
Train Accuracy: 0.8596428571428572
Test Accuracy: 0.8471428571428572
Mean Precision: 0.8018489918489917
Mean Recall: 0.858154055457708
Mean F1: 0.8256951895602309
Mean Specificity: 0.8409347019866995
For Class Type 1
Train Accuracy: 0.8564285714285713
Test Accuracy: 0.86
Mean Precision: 0.8143965883691913
Mean Recall: 0.8614184782608696
Mean F1: 0.8367149881748421
Mean Specificity: 0.8543773940205405
For Class Type 2
Train Accuracy: 0.8582142857142857
Test Accuracy: 0.8528571428571429
Mean Precision: 0.8166757853518417
Mean Recall: 0.8425637429424245
Mean F1: 0.8287511213826647
Mean Specificity: 0.8563049557457179
For Class Type 3
Train Accuracy: 0.8589285714285715
Test Accuracy: 0.85
Mean Precision: 0.8064716234278674
Mean Recall: 0.8528923825626311
Mean F1: 0.8278641194693274
Mean Specificity: 0.848188606461133
For Class Type 4
Train Accuracy: 0.8578571428571428
Test Accuracy: 0.8542857142857143
Mean Precision: 0.8060627045348292
Mean Recall: 0.8712171961748233
Mean F1: 0.8367207151124904
Mean Specificity: 0.8426888125443337
For Class Type 5
Train Accuracy: 0.8585714285714288
Test Accuracy: 0.8514285714285714
Mean Precision: 0.8096350692577401
Mean Recall: 0.8516161616161616
Mean F1: 0.8299001277140656
Mean Specificity: 0.8495019924431689
Macro F1-score: 0.8320792652234872
```

Fig. 43. Correctness Metrics: Bayes Classifier (Gaussian)

```
For Class Type 0
Train Accuracy: 0.8578571428571429
Test Accuracy: 0.8542857142857143
Mean Precision: 0.8099119955902758
Mean Recall: 0.8589668396120009
Mean F1: 0.8329288328571506
Mean Specificity: 0.8504243054243055
For Class Type 1
Train Accuracy: 0.8589285714285715
Test Accuracy: 0.85
Mean Precision: 0.8001486518636478
Mean Recall: 0.8614379442318206
Mean F1: 0.8293056505478047
Mean Specificity: 0.8397677631010965
For Class Type 2
Train Accuracy: 0.8589285714285714
Test Accuracy: 0.85
Mean Precision: 0.8042958870579838
Mean Recall: 0.8618044338770522
Mean F1: 0.8311050521157023
Mean Specificity: 0.8386757970434882
For Class Type 3
Train Accuracy: 0.8578571428571429
Test Accuracy: 0.8542857142857143
Mean Precision: 0.808132183908046
Mean Recall: 0.8685706117781591
Mean F1: 0.8350851410370657
Mean Specificity: 0.8461523034190377
For Class Type 4
Train Accuracy: 0.8578571428571429
Test Accuracy: 0.8542857142857143
Mean Precision: 0.8065827228327229
Mean Recall: 0.862317218960127
Mean F1: 0.8320984141035067
Mean Specificity: 0.8451946079380809
For Class Type 5
Train Accuracy: 0.8589285714285715
Test Accuracy: 0.85
Mean Precision: 0.8094751336582988
Mean Recall: 0.8500662850629398
Mean F1: 0.8287683429907716
Mean Specificity: 0.849715303837054
Macro F1-score: 0.8325978421634754
```

Fig. 44. Correctness Metrics: Bayes Classifier (Gaussian Mixture Models: EM Algorithm

```
For Class Type 0
Mean Train Accuracy: 0.8617857142857142
Mean Test Accuracy: 0.832857142857143
Mean Precision: 0.8279309968134412
Mean Recall: 0.7497721184288348
Mean F1 score: 0.7855974386523604
Mean Specificity: 0.8902497985495568
For Class Type 1
Mean Train Accuracy: 0.8598214285714286
Mean Test Accuracy: 0.8407142857142856
Mean Precision: 0.8462172004700284
Mean Recall: 0.7648084787301407
Mean F1 score: 0.8014707184232959
Mean Specificity: 0.8956741589280464
For Class Type 2
Mean Train Accuracy: 0.861904761904762
Mean Test Accuracy: 0.8323809523809523
Mean Precision: 0.8372225527495952
Mean Recall: 0.7642764987954268
Mean F1 score: 0.7965081400458401
Mean Specificity: 0.8828788689145715
For Class Type 3
Mean Train Accuracy: 0.8620535714285713
Mean Test Accuracy: 0.8325000000000001
Mean Precision: 0.8289414455867273
Mean Recall: 0.7720686463604793
Mean F1 score: 0.796929080985145
Mean Specificity: 0.8765978468710612
For Class Type 4
Mean Train Accuracy: 0.8642142857142858
Mean Test Accuracy: 0.824857142857143
Mean Precision: 0.8190017208629935
Mean Recall: 0.7643461909442282
Mean F1 score: 0.7885702656719935
Mean Specificity: 0.8694898695358918
For Class Type 5
Mean Train Accuracy: 0.8653571428571428
Mean Test Accuracy: 0.8207142857142858
Mean Precision: 0.8163639863879534
Mean Recall: 0.7603664731044458
Mean F1 score: 0.7846470825024876
Mean Specificity: 0.8664406074286185
Macro F1-score: 0.7945467819291947
```

Fig. 45. Correctness Metrics: KNN

```
For Class Type 0
Mean Train Accuracy: 0.8578571428571429
Mean Test Accuracy: 0.8542857142857143
Mean Precision: 0.8353541251874075
Mean Recall: 0.8359336004497294
Mean F1 score: 0.8346152615305842
Mean Specificity: 0.8702176255117433
For Class Type 1
Mean Train Accuracy: 0.8567857142857142
Mean Test Accuracy: 0.8585714285714285
Mean Precision: 0.8285446777693066
Mean Recall: 0.842036083602801
Mean F1 score: 0.8345629221569231
Mean Specificity: 0.8703856204014567
For Class Type 2
Mean Train Accuracy: 0.8572619047619047
Mean Test Accuracy: 0.8566666666666667
Mean Precision: 0.8304936738841115
Mean Recall: 0.8409525507127203
Mean F1 score: 0.835075536641009
Mean Specificity: 0.868522949419583
For Class Type 3
Mean Train Accuracy: 0.8563392857142856
Mean Test Accuracy: 0.8603571428571429
Mean Precision: 0.8379047981459173
Mean Recall: 0.8434299464207207
Mean F1 score: 0.8399151900352138
Mean Specificity: 0.8729218713042066
For Class Type 4
Mean Train Accuracy: 0.8561428571428571
Mean Test Accuracy: 0.8611428571428571
Mean Precision: 0.8363287890108525
Mean Recall: 0.8468134582197929
Mean F1 score: 0.8408377708466687
Mean Specificity: 0.871830415579615
For Class Type 5
Mean Train Accuracy: 0.8564880952380952
Mean Test Accuracy: 0.8597619047619047
Mean Precision: 0.8361425419871882
Mean Recall: 0.8430323528284259
Mean F1 score: 0.8389014888139699
Mean Specificity: 0.8725446167865207
Macro F1-score: 0.8380619098151944
```

Fig. 46. Correctness Metrics: Parzen Density