Sims_Kelly_HW4_report

October 12, 2020

Q1

Basic optimization

Consider a simplified logistic regression problem. Given m training samples (x i , y i), i = 1, . . . , m. The data x i R (note that we only have one feature for each sample), and y i {0, 1}. To fit a logistic regression model for classification, we solve the following optimization problem, where R is a parameter we aim to find:

$$max_\theta \ell(\theta)$$

where the log-likelihood function

$$\ell(\theta) = \sum_{i=1}^{m} \{-log(1 + exp\{-\theta x^i\}) + (y^i - 1)\theta x^i\} \qquad (1)$$

(a) Show step-by-step mathematical derivation for the gradient of the cost function $\ell(\theta)$ Let **X** be a vector in $\mathbb{R}^m$ that represents x^i for i = 1, ..., m. Substituting **X** in (1) and removing the summation yields:

$$-log(1 + exp\{-\theta X\}) + (y^i - 1)\theta X \qquad (2)$$

Taking the derivative of (2) w.r.t θ

$$\begin{aligned}\frac{\partial \ell}{\partial \theta} &= \frac{\partial \ell}{\partial \theta}\theta \cdot X(y-1) - \frac{\partial \ell}{\partial \theta} log(exp\{-\theta X\} + 1) \\ &= X(y-1) - \frac{\frac{\partial \ell}{\partial \theta} exp\{-\theta X\} + \frac{\partial \ell}{\partial \theta} 1}{exp\{-\theta X\} + 1} \\ &= X(y-1) - \frac{(-X \cdot \frac{\partial \ell}{\partial \theta}\theta) \cdot exp\{-\theta X\}}{exp\{-\theta X\} + 1}\end{aligned}$$

Which yields

$$\boxed{X(y-1) + \frac{exp\{-\theta X\}X}{exp\{-\theta X\} + 1}}$$

(b) Write a pseudo-code for performing gradient descent to find the optimizer *. This is essentially what the training procedure does.

```
• Given m examples of feature vector $\mathbf{X}$ and label vector $\mathbf{Y}$, learning rate $\alpha$
• Randomly initialize $\theta$ , bias $\mathbf{b}$
• Repeat until convergence:
    – Predict $\hat{y}$ = h(X) = z($\theta X + b$)
    – Calculate cost $\ell(\hat{y}, y)$
    – Calculate $\nabla\ell(\theta) = = X(Y-1) + \frac{exp(-\theta X)X}{exp(-\theta X)+1}$
    – Update $\theta \leftarrow \theta - \alpha\nabla\ell(\theta)$
```

(c) Write the pseudo-code for performing the stochastic gradient descent algorithm to solve the training of logistic regression problem (1). Please explain the difference between gradient descent and stochastic gradient descent for training logistic regression.

```
• Given m examples of $(x^{(i)}, y^{(i)})$, learning rate $\alpha$
• Randomly initialize $\theta$ , bias $\mathbf{b}$
• Repeat until convergence:
    – For i = 1, 2, ..., m
        * Predict $\hat{y}^{(i)} = h(x^{(i)})$ = z($\theta x^{(i)} + b$)
        * Calculate cost $\ell(\hat{y}^{(i)}, y^{(i)})$
        * Calculate $\nabla\ell(\theta) = x^{(i)}(y^{(i)}-1) + \frac{exp(-\theta x^{(i)})x^{(i)}}{exp(-\theta x^{(i)})+1}$
        * Update $\theta \leftarrow \theta - \alpha\nabla\ell(\theta)$
```

Gradient descent computes the gradients after iterating through all m training observations. With large training sets, this can become computationally expensive to do in the event of a vectorized implementation. Run time for an iterative approach can take a long time to loop through all observations as well. Stochastic gradient descent, on the other hand, either samples a small subset of traiing observations to compute the gradient. Or in the case of "online" learning, the gradient is calculated and the weights are updated after each observation seen (as shown above in the pseudo code).

(d) We will show that the training problem in basic logistic regression problemis concave. Derive the Hessian matrix of ( ) and based on this, show the training problem (1) is concave (note that in this case, since we only have one feature, the Hessian matrixis just ascalar). Explain why the problem can be solved efficiently and gradient descent will achieve a unique global optimizer, as we discussed in class. Taking the derivative of the result from part **(a)** yields:

$$\frac{\partial''\ell}{\partial\theta''} = \frac{\partial'\ell}{\partial\theta'}X(y-1) + \frac{exp\{-\theta X\}X}{exp\{-\theta X\}+1}$$

$$\boxed{= -\frac{exp(-\theta X)X^2}{(1+exp(-\theta X))^2} \qquad (1)}$$

Since X $\in \mathbb{R}^{mx1}$ then there is a single theta in the Logistic Regression problem. Because of this, the hessian turns out to be a single value. In this instance, that value will be negative as seen above. Any real value entered in (1) above will be negative. The hessian of a function of a single variable is concave if its second derivative is negative everywhere. A concave optimization problem and conversely a convex optimization problem when minimzing an optjective function, will have a unique global solution and can be solved efficiently without fear of being stuck in a local minimum (or maximum). This is in contrast to a non-convex (non-concave) optimization problem, such as LP or integer optimization. These types of problems can get stuck in a local minimum and/or maximum.

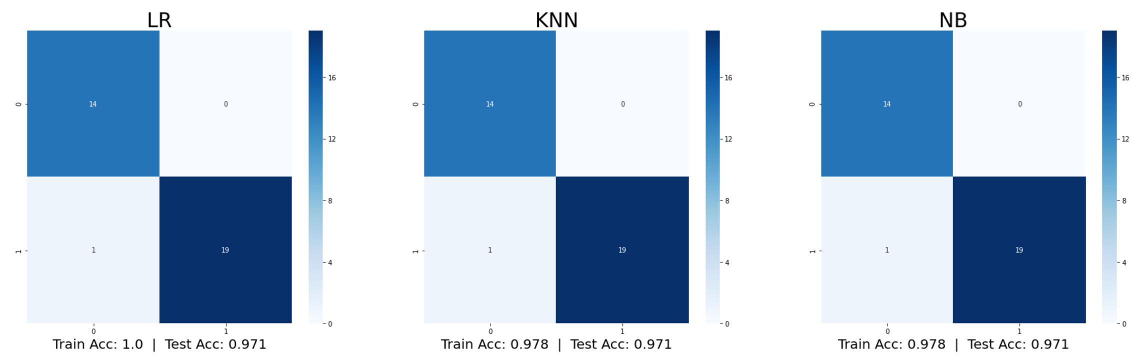
Q2

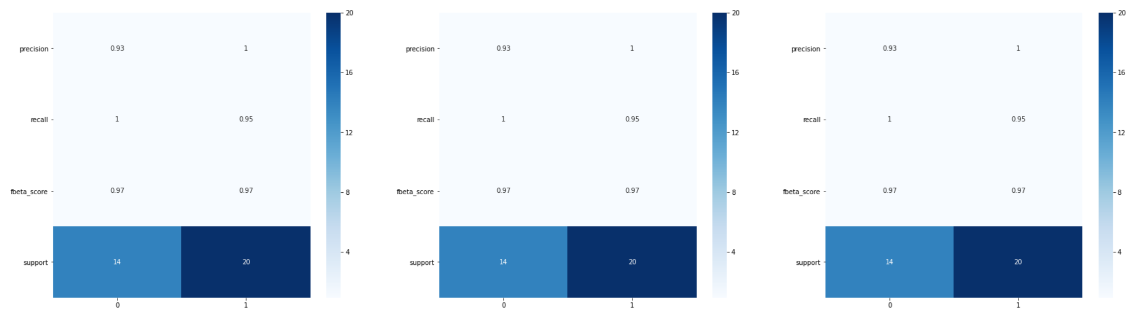
Comparing Bayes, logistic, and KNN classifiers

In lectures, we learn three different classifiers. This question is to implement and compare them. Python users, please feel free to use Scikit-learn, which is a commonly-used and powerful Python library with various machine learning tools. But you can also use other similar libraries in other languages of your choice to perform the tasks.

Part One (Divorce classification/prediction). This dataset is about participants who completed the personal information form and a divorce predic- tors scale. The data is a modified version of the publicly available at https://archive.ics.uci.edu/ml/datasets/Divorce+Predictors+data+set (by injecting noise so you will not get the exactly same results as on UCI website). The dataset marriage.csv is contained in the homework folder. There are 170 participants and 54 attributes (or predictor variables) that are all real-valued. The last column of the CSV file is label y (1 means "divorce", 0 means "no divorce"). Each column is for one feature (predictor variable), and each row is a sample (participant). A detailed 1explanation for each feature (predictor variable) can be found at the website link above. Our goal is to build a classifier using training data, such that given a test sample, we can classify (or essentially predict) whether its label is 0 ("no divorce") or 1 ("divorce"). Build three classifiers using (Naive Bayes, Logistic Regression, KNN). Use the first 80% data for training and the remaining 20% for testing. If you use scikit-learn you can use train test split to split the dataset. Remark: Please note that, here, for Naive Bayes, this means that we have to estimate the variance for each individual feature from training data. When estimating the variance, if the variance is zero to close to zero (meaning that there is very little variability in the feature), you can set the variance to be a small number, e.g., $\epsilon = 10^{-3}$. We do not want to have include zero or nearly variance in Naive Bayes. This tip holds for both Part One and Part Two of this question.

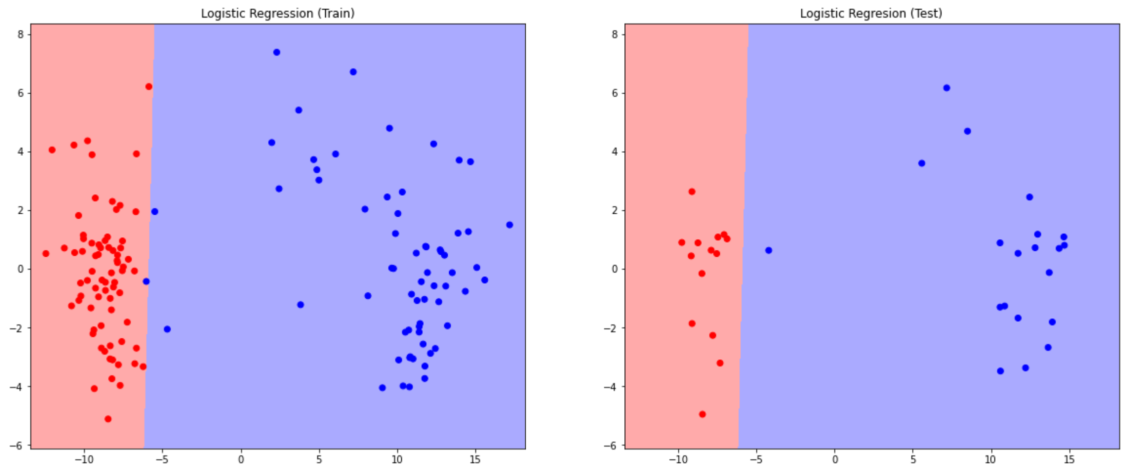
(a) Report testing accuracy for each of the three classifiers. Comment on their performance: which performs the best and make a guess why they perform the best in this setting.

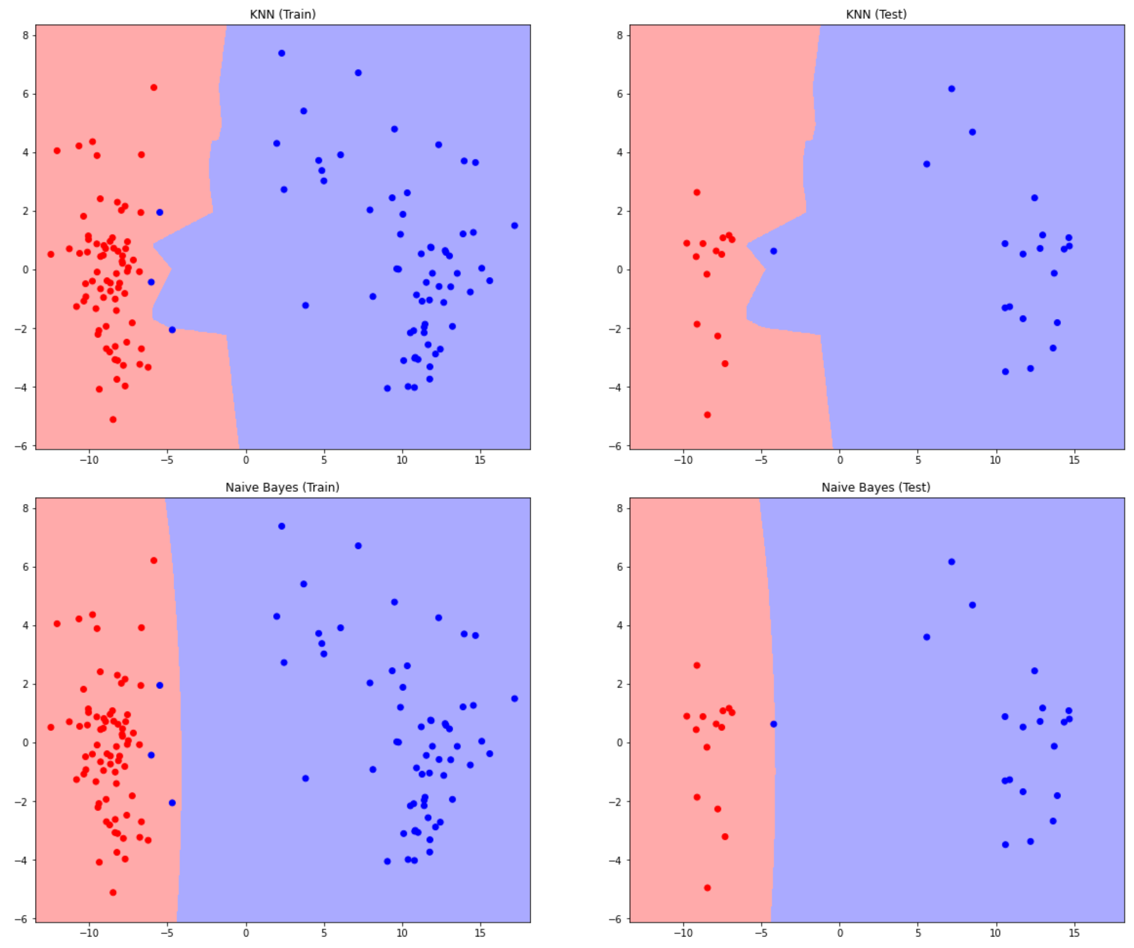




Answer All 3 classifier perform extremely well on this dataset. They each got the same accuracy on the test set of 0.971. They each failed to classify only one observation incorrectly. As we will see in the next part below, this is due to how well the data itself is linearly seperable.

(b) Now perform PCA to project the data into two-dimensional space. Plot the data points and decision boundary of each classifier. Comment on the difference between the decision boundary for the three classifiers. Please clearly represent the data points with different labels using different colors.



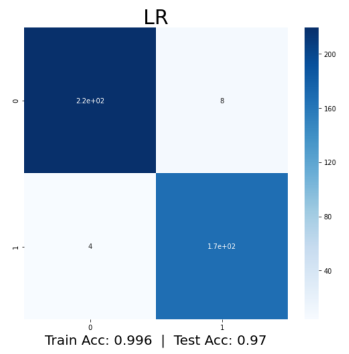


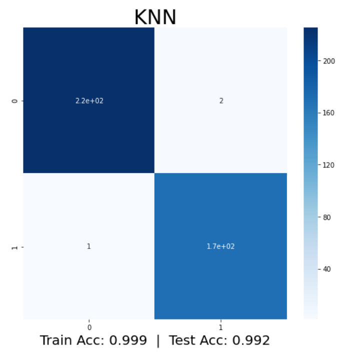
with the data projected down to the first 2 principal components, the strong decision boundary can be observed. With respect to the training set, there are 3 observations that are extremely close to the other class. This brings the decision boundary over to the left drastically. In doing so however, we can see it allows the models to correctly classify the lone isolated observation in the test sets for logistic regression and KNN Naive Bayes would be the sole model to fail to properly classify the single point at the boundary

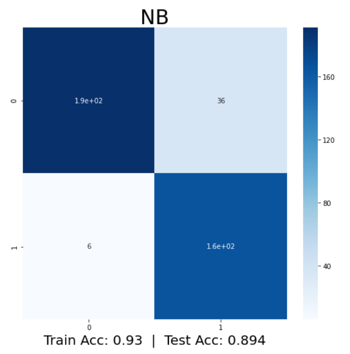
---

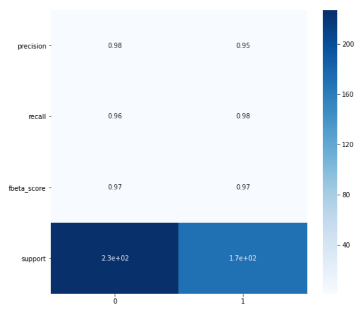
Part Two (Handwritten digits classification). Repeat the above using the MNIST Data in our previous homework. Here, give "digit" 6 label y = 1, and give "digit" 2 label y = 0. All the pixels in each image will be the feature (predictor variables) for that sample (i.e., image). Our goal is to build classifier to such that given a new test sample, we can tell is it a 2 or a 6. Using the first 80% of the samples for training and remaining 20% for testing.

(a) Report testing accuracy for each of the three classifiers. Comment on their performance: which performs the best and make a guess why they perform the best in this setting.

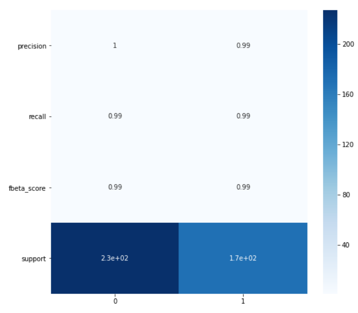




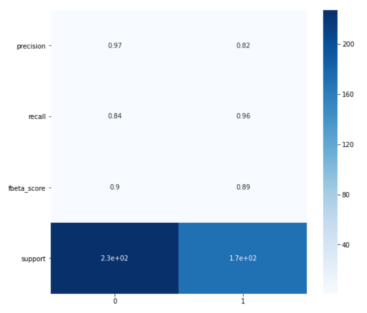




	0	1
precision	0.98	0.95
recall	0.96	0.98
fbeta_score	0.97	0.97
support	2.3e+02	1.7e+02



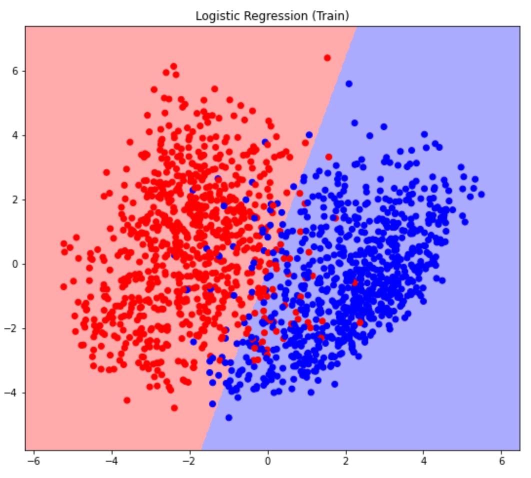
	0	1
precision	1	0.99
recall	0.99	0.99
fbeta_score	0.99	0.99
support	2.3e+02	1.7e+02

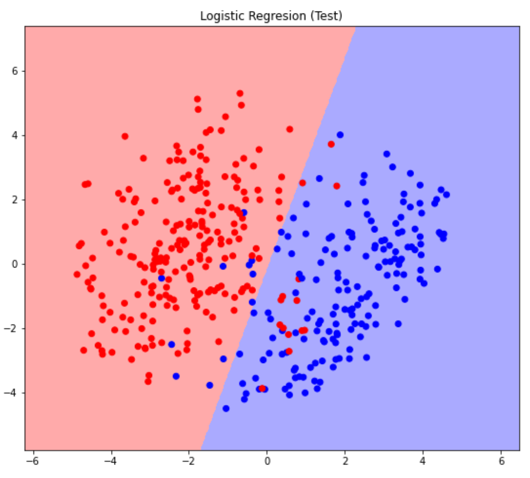


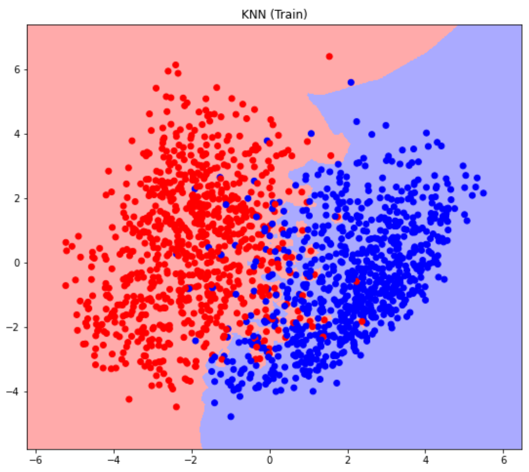
	0	1
precision	0.97	0.82
recall	0.84	0.96
fbeta_score	0.9	0.89
support	2.3e+02	1.7e+02

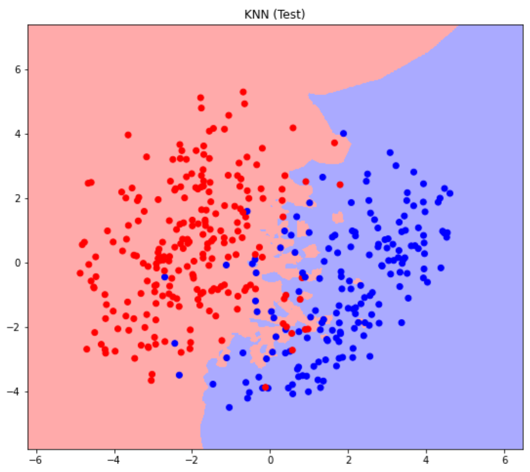
Answer Again, all 3 models perform well in classifying each observation. However KNN performs the best of the group. As we will see in the next section, this is due to KNN's ability to create a decision boundary that is non linear. Because there is observation mixing in the high dimensional space, it is not linearly separable. This is why LR doesn't perform as well. Also because of this mixing, this effects the conjugate priors utilized in the Naive Bayes method. KNN creates a decision boundary, and potentially decision islands, and this is why it performs better in this scenario

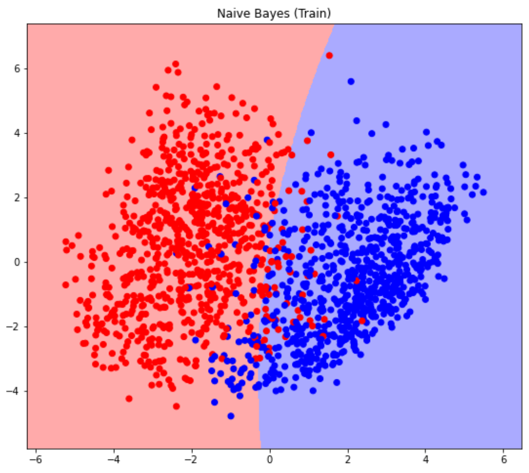
(b) Now perform PCA to project the data into two-dimensional space. Plot the data points and decision boundary of each classifier. Comment on the difference between the decision boundary for the three classifiers. Please clearly represent the data points with different labels using different colors.

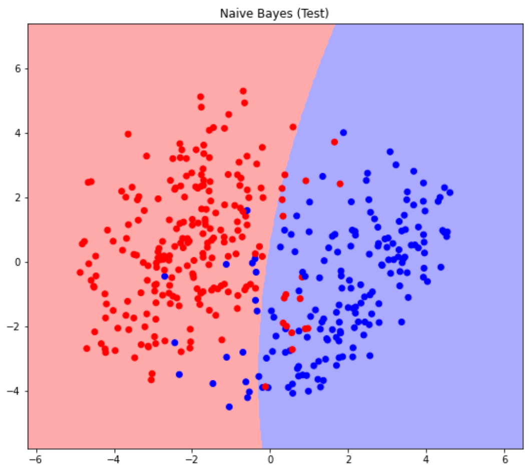












Answer After reducing the data to the first two principal componenets and plotting the decision boundary, we can see these "decision islands" as discussed in the prior section. Both Naive Bayes and Logistic regression attempt to create a decision line, whereas KNN creates a highly stochastic decision boundary with respect to where the data points lie