Ryan_Plain_HW4_Report

March 7, 2022

1 1. Optimization

$$\max_{\theta} \ell(\theta),$$

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

1. Show step-by-step mathematical derivation for the gradient of the cost function l( ) in (1)

$$\frac{\partial(\theta)}{\partial\theta} = \sum_{i=1}^{n}(y_i x_i - x_i - (\frac{e^{\theta^T x_i}}{1 + e^{\theta^T x_i}})x_i)$$

$$= \sum_{i=1}^{n}(y - \frac{1}{1 + e^{\theta^T x}})x_i$$

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

To maximize the likelihood of the parameters (θ), use gradient ascent

GD:

initialize θ

While $\theta^{new} - \theta^{old} > \epsilon$

$$\theta_i^{new} = \theta_i^{old} + \lambda \sum_{i=1}^{n}(y - \frac{1}{1 + e^{\theta^T x}})x_i$$

3. 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.
SGD:

- initialize θ
- While $\theta^{new} - \theta^{old} > \epsilon$
- sample K from the data

- for k in K do

$$\theta_k^{new} = \theta_i^{old} + \lambda \sum_{i=1}^{K} (y - \frac{1}{1 + e^{\theta^T x}}) x_i$$

Stochastic Gradient Decent is only different in that you there is a sampled selection of data points the algorthim is ran on at a time. To do the gradient on the entire dataset can be computationally expensive, and the subset allows it to converge more quickly.

2 2. Comparing Classifiers

2.0.1 2.1 Divorce classification/prediction

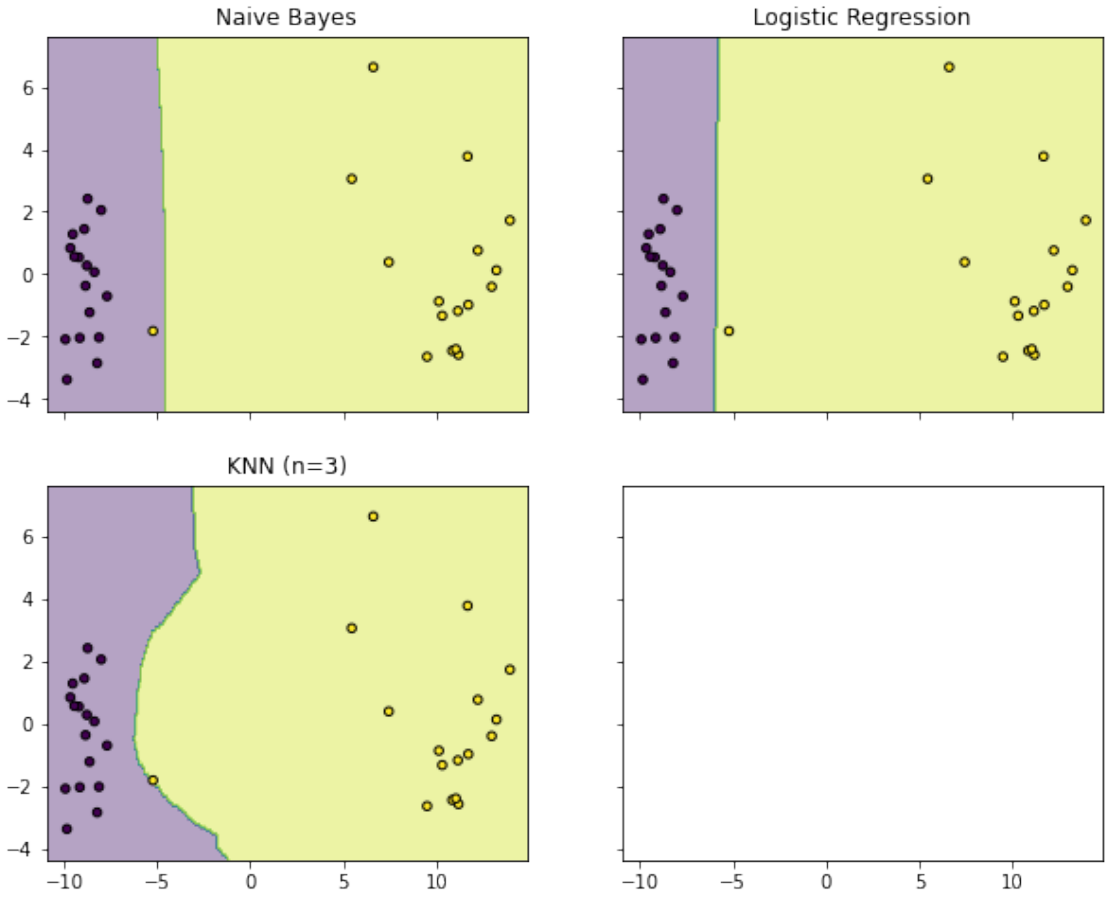
a.) Report testing accuracy for each of the three classifiers. Comment on their perfromance: which performs the best and make a guess why they perform the best in this setting. The data was broken into train/test splits with 20% of the data reserved for testing. Each of the models performed the **exact** same. The random seed selected produced 97% accuracy for Naive Bayes, Logisitic Regression, and KNN. This speaks to the dataset and one of the most important asepcts to machine learning, understanding the data.

The model used should be a function of the data analysis. Each classifier has it's own strengths and weaknesses. For this dataset, none of the weaknesses were exposed, which most likely means that the data is easily seperable and almost any solid classification algorithm will work.

b.) Plot the data points and decision boundary of each classifier in the two-dimensional space. Comment on the difference between the decision boundary for the three classifiers. Please clearly represent the data points with different labels using different colors. Using PCA, I was able to transform the training set into a 2 dimensional space. Using the fitted training data, the test data can be transformed as well, which makes it easier to visualize the data points and decision boundary.

As noted above, the classifiers all perfomed exceptionally well. The random seed would affect if it was 97 or 100% accurate. Shown below, the data is very clear that the divorce target variable has clear separation in the predictor variables. You could effectively look at making a rule based prediction provided the data. There only appears to be 1 single outlier in the dataset.

The decision boundaries for Naive Bayes and Logistic regression are mostly linear and linear respectively. The KNN model has a non-linear decision boundary.



2.0.2 Handwritten digit classification

a.) Report confusion matrix, precision, recall, and F-1 score for each of the classifiers. For precision, recall, and F-1 score of each classifier, we will need to report these for each of the digits. So you can create a table for this. The dataset contains handwritten digits 0-9, and a multiclass classification problem. Outputted below is the confusion matrix, along with metrics: percision, recall, and f-1 score. These metrics are able to define model performance better than raw accuracy scores.

The confusion matrix identifies how well each of the combinations of classifications is predicted. Prior knowledge of handwritten digits is intuitively noticed with things such as higher misclassification with 1 and 7, since they do look fairly similar. Same thing with (3, 8) and (4, 9). Ideally we want as many on the diagonal as possible, but all of the models performed well and pass the eye test with the confusion matrices.

Accuracy score doesn't represent the model fully. If there are a low amount of target variables in a certain category, the accuracy could still be relatively high even though the model outputted 0 instances of that class though observed.

Precision is defined as $\frac{TP}{TP+FP}$ which identifies the proportion that were accurately predicted.

Recall (sensitivity) is defined as $\frac{TP}{TP+FN}$ and represents the proportion of actual positives identified correctly.

Together these metrics can provide key insight on actual model performance. Both of them work in harmony to produce the F1-Score, which is a measure of the accuracy with provided weights to precision or recall.

These metrics work on binary classification variables, which is why they were assesed for each digit in the dataset. The aggregated scores are taken to give a wholistic view of the model.

Perfomance

The Logistic Regression and Linear SVM performed the worst out of the group. The data has non-linear properties that would make sense to cause those to perform worse.

The best performing model was the Kernel SVM. The kernel trick really helped classify with the non-linear properties. I think it is important to note, that there was minimal model tuning with KNN and Neural Network. Neural Network would have probably performed the best if given the resources to tune with. It is typically the best for image classification and learning highly non-linear datasets.

KNN

Confusion Matrix:

	0	1	2	3	4	5	6	7	8	9
0	971	1	1	0	0	1	4	1	1	0
1	0	1130	1	2	0	0	2	0	0	0
2	15	14	960	10	1	1	3	23	5	0
3	1	1	4	943	1	25	3	12	9	11
4	0	12	0	0	916	0	8	6	2	38
5	8	4	0	27	3	830	10	2	2	6
6	9	4	0	0	3	3	938	0	1	0
7	0	26	8	1	4	1	0	970	1	17
8	9	5	8	32	4	21	8	8	860	19
9	6	6	3	7	20	3	2	17	5	940

Scores:

	precision	recall	f1-score	support
0	0.95	0.99	0.97	980
1	0.94	1.00	0.97	1135
2	0.97	0.93	0.95	1032
3	0.92	0.93	0.93	1010
4	0.96	0.93	0.95	982
5	0.94	0.93	0.93	892
6	0.96	0.98	0.97	958
7	0.93	0.94	0.94	1028
8	0.97	0.88	0.92	974
9	0.91	0.93	0.92	1009
accuracy			0.95	10000
macro avg	0.95	0.95	0.95	10000
weighted avg	0.95	0.95	0.95	10000

LINEAR SVM
Confusion Matrix:

	0	1	2	3	4	5	6	7	8	9
0	954	0	7	1	0	6	8	2	1	1
1	0	1120	1	2	1	1	4	1	5	0
2	10	11	926	11	11	4	14	11	31	3
3	3	3	27	904	2	24	2	10	26	9
4	2	2	10	0	926	0	5	5	2	30
5	12	5	7	62	9	748	15	1	27	6
6	11	3	11	0	10	13	907	0	2	1
7	2	9	23	14	10	0	0	937	4	29
8	7	17	10	41	9	34	11	6	829	10
9	8	7	1	16	58	5	1	41	5	867

Scores:

	precision	recall	f1-score	support
0	0.95	0.97	0.96	980
1	0.95	0.99	0.97	1135
2	0.91	0.90	0.90	1032
3	0.86	0.90	0.88	1010
4	0.89	0.94	0.92	982
5	0.90	0.84	0.87	892
6	0.94	0.95	0.94	958
7	0.92	0.91	0.92	1028
8	0.89	0.85	0.87	974
9	0.91	0.86	0.88	1009
accuracy			0.91	10000
macro avg	0.91	0.91	0.91	10000
weighted avg	0.91	0.91	0.91	10000

NEURAL NETWORK
Confusion Matrix:

	0	1	2	3	4	5	6	7	8	9
0	949	0	1	1	5	4	7	5	5	3
1	0	1117	2	3	0	2	4	2	5	0
2	6	4	968	18	5	3	5	12	10	1
3	6	6	13	934	0	12	2	14	18	5
4	1	0	4	2	946	3	3	2	3	18
5	3	1	0	25	1	836	14	1	9	2
6	6	5	6	1	3	8	921	1	7	0
7	3	6	12	13	1	4	1	969	6	13
8	9	7	2	18	11	9	11	6	888	13
9	7	3	3	12	19	1	1	5	8	950

Scores:

	precision	recall	f1-score	support
0	0.96	0.97	0.96	980
1	0.97	0.98	0.98	1135
2	0.96	0.94	0.95	1032
3	0.91	0.92	0.92	1010
4	0.95	0.96	0.96	982
5	0.95	0.94	0.94	892
6	0.95	0.96	0.96	958
7	0.95	0.94	0.95	1028
8	0.93	0.91	0.92	974
9	0.95	0.94	0.94	1009
accuracy			0.95	10000
macro avg	0.95	0.95	0.95	10000
weighted avg	0.95	0.95	0.95	10000

---

3 Naive Bayes for spam filtering

1. Calculate class prior P(y = 0) and P(y = 1) from the training data, where y = 0 corresponds to spam messages, and y = 1 corresponds to non-spam messages. Note that these class prior essentially corresponds to the frequency of each class in the training sample. Write down the feature vectors for each spam and non-spam messages.

- **V = {secret, offer, low, price, valued, customer, today, dollar, million, sports, is, for, play, healthy, pizza}**

$P(y = 0) = 3/7$
$P(y = 1) = 4/7$

Spam

million dollar offer = $[0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]$
secret offer today = $[1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]$
secret is secret = $[2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]$

Not Spam

low price for valued customers = $[0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0]$
play secret sports today = $[1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0]$
sports is healthy = $[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0]$
low price pizza = $[0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]$