Amirkabir University of Technology
(Tehran Polytechnic)

Computer Engineering Department

Machine Learning Course
CE5501

Homework 3

Student
Reza Sajedi
400131072
r.sajedi@aut.ac.ir

Teacher
Dr. Nazerfard

Fall 2022

Table of Contents

Problem 1: Why and how

Part a

The ROC curve has been widely used to adjust an appropriate threshold for binary classification. This curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. In this way, the point closest to the top left corner of the ROC curve is chosen as the threshold, and the threshold used to generate this point should be the optimal one. Another method is applying the Precision-Recall curve. It can be more informative than the ROC curve when the dataset is imbalanced.

Part b

We will treat each class as a binary classification problem. This approach is called the one vs all method. In the one vs all method, when we work with a class, that class is denoted by 1 and the rest of the classes becomes 0.

The other way is to use multinomial logistic regression. To apply this, the logistic equation is transformed to allow probability for more than two classes. The probability equations with multinomial logistic regression for a classification task with three classes can be formulated as follows[1]:

$$Pr\,(y=1) = \frac{1}{\frac{e^{f(x)}}{z}}$$

$$Pr\,(y=2) = \frac{1}{\frac{e^{f(x)}}{z}}$$

$$Pr\,(y=3) = \frac{1}{\frac{e^{f(x)}}{z}}$$

Where z is the sum of the e^f(x) for all classes in the model.

Part c

Logistic Regression has traditionally been used as a linear classifier. So, the decision boundary can be formulated as follows:

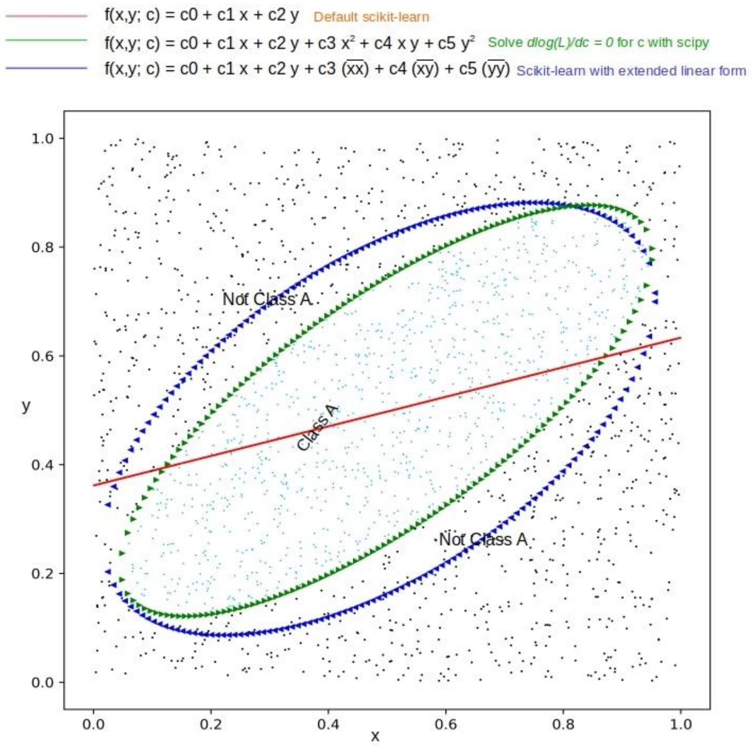
$$f(x, y, c) = \mathrm{c}_0 + \mathrm{c}_1 x + \mathrm{c}_2 y$$

We can have a non-linear decision boundary by considering a higher-order polynomial for f and transforming the dataset[2]:

$$f(x, y, c) = \mathrm{c}_0 + \mathrm{c}_1 x + \mathrm{c}_2 y + \mathrm{c}_3 \mathrm{x}^2 + \mathrm{c}_4 \mathrm{xy} + \mathrm{c}_5 \mathrm{y}^2$$

---

[1] https://pub.towardsai.net/logistic-regression-for-multi-class-classification-hands-on-with-scikit-learn-bcc0bbad1def
[2] https://xplordat.com/2019/03/13/logistic-regression-as-a-nonlinear-classifier/



Part d

$$P(Target = 1 \mid F1 = 1, F2 = 1, F3 = 0) = \frac{P(F1 = 1, F2 = 1, F3 = 0 \mid Target = 1)\ P(Target = 1)}{P(F1 = 1, F2 = 1, F3 = 0)}$$

$$= \frac{P(F1 = 1 \mid Target = 1)\ P(F2 = 1 \mid Target = 1)\ P(F3 = 0 \mid Target = 1)\ P(Target = 1)}{P(F1 = 1)\ P(F2 = 1)\ P(F3 = 0)}$$

$$= \frac{2/4 \times 1/4 \times 2/4 \times 4/8}{4/8 \times 3/8 \times 3/8} = 0.44$$

Part e

e.1)

$$P(Cheat = \ 'No' \big| MaritalStatus = \ 'Married')$$

$$= \frac{P(MaritalStatus = \ 'Married' \big| Cheat = \ 'No')P(Cheat =' \ No')}{P\big(MartialStatus = \ 'Married'\big)} = \frac{4/7 \times 7/10}{4/10}$$

$$= 1$$

e.2)

$$P(Income = 120 \mid Cheat = 'Yes') = \frac{1}{\sqrt{2\pi(25)}} exp(-\frac{(120 - 90)^2}{2(25)}) = 1.2152 \times 10^{-9}$$

e.3)

$$P(X \mid 'Yes') = P(Refund = 'No' \mid 'Yes')\ P(Martial\ Status = 'Married' \mid 'Yes')\ P(Income = 120 \mid 'Yes') = 3/3 \times 0/3 \times 1.2152 \times 10^{-9} = 0$$

$$P(X \mid 'No') = P(Refund = 'No' \mid 'No')\ P(Martial\ Status\ = \ 'Married' \mid 'No')\ P(Income = 120 \mid 'No')\ =\ 4/7\ \times\ 4/7\ \times\ 0.0072\ =\ 0.0024$$

Part f

Pr(p3) = Pr(p3 | p2) Pr(p2) + Pr(p3 | ~p2) Pr(~p2)

Pr(p2) = Pr(p2 | p1) Pr(p1) + Pr(p2 | ~p1) Pr(~p1) = 0.8 * 0.4 + 0.5 * 0.6 = 0.62

Pr(~p2) = 1 - Pr(p2) = 0.38

Pr(p3) = 0.2 * 0.62 + 0.3 * 0.38 = 0.238

Pr(~p3) = 1 - Pr(p3) = 0.762

Pr(p1 | p2, ~p3) = Pr(p2, ~p3 | p1) Pr(p1) / Pr(p2, ~p3)

= Pr(~p3 | p2) Pr(p2 | p1) Pr(p1) / Pr(~p3 | p2) Pr(p2)

= 0.8 * 0.8 * 0.4 / 0.8 * 0.62 = 0.516

Problem 2: Breast Cancer

Part a

solver: Algorithm to use in the optimization problem. For small datasets, 'liblinear' is a good choice, whereas 'sag' is faster for large ones.

max_iter: Maximum number of iterations taken for the solvers to converge.

```
param_grid={
    'solver': ['lbfgs', 'newton-cg', 'liblinear', 'sag', 'saga'],
    'max_iter': [100, 1000, 2500, 5000],
}

Best Parameters for Logistic Regression: {'max_iter': 100, 'solver': 'lbfgs'}
Best Train Accuracy Score for Logistic Regression: 0.9802
Best Test Accuracy Score for Logistic Regression: 0.9649
Confusion Matrix for Logistic Regression:
[[34  0]
 [ 4 76]]
```

Part b

The results show that Logistic Regression's accuracy is higher than Naïve Bayes (about 3%). LR and GNB are very similar. when the size of train data is small, GNB performs better because of the independency assumption between features, and LR performs better when the train data's size is large.

var_smoothing: Portion of the largest variance of all features that are added to variances for calculation stability. is a stability calculation to widen (or smooth) the curve and therefore account for more samples that are further away from the distribution mean. In this case, np.logspace returns numbers spaced evenly on a log scale, starts from 0, ends at -9, and generates 100 samples.

```
param_grid={
    'var_smoothing': np.logspace(0, -9, 100)
}
```

```
Best Parameters for GaussianNB: {'var_smoothing': 0.0533669923120631}
Best Train Accuracy Score for GaussianNB: 0.9362
Best Test Accuracy Score for GaussianNB: 0.9298
Confusion Matrix for GaussianNB:
[[32  2]
 [ 6 74]]
```

Part c

In this case, feature selection has improved the accuracy slightly for both LR and GNB (about 1%). It is worth it because when the dimension has been decreased, we have a simpler model with lower computation time.

Feature	AUC Score
mean concave points	0.9415
worst concave points	0.9290
mean concavity	0.8890
mean area	0.8765
mean radius	0.8640
worst concavity	0.8599
worst radius	0.8555
worst area	0.8555
worst perimeter	0.8515
mean perimeter	0.8430
area error	0.8239
radius error	0.8176
perimeter error	0.8176
mean compactness	0.7989
worst compactness	0.7676
concave points error	0.6982
worst smoothness	0.6813
worst symmetry	0.6768
mean smoothness	0.6415
mean texture	0.6397
compactness error	0.6309
worst fractal dimension	0.6099

concavity error	0.5886
mean symmetry	0.5868
worst texture	0.5724
mean fractal dimension	0.5000
texture error	0.5000
smoothness error	0.5000
symmetry error	0.5000
fractal dimension error	0.5000

```
Accuracy Score for Logistic Regression: 0.9737
Improvement for Logistic Regression Compared to Part a: 0.0088
Confusion Matrix for Logistic Regression:
[[34  0]
 [ 3 77]]

Accuracy Score for GaussianNB: 0.9386
Improvement for GaussianNB Compared to Part b: 0.0088
Confusion Matrix for GaussianNB:
[[32  2]
 [ 5 75]]
```

Part d

In this case, the results show a decrement in accuracy. It depends on the selected train data because we are splitting the data set randomly, sometimes feature selection has good effects but in this case most of the time it leads to poor performance .

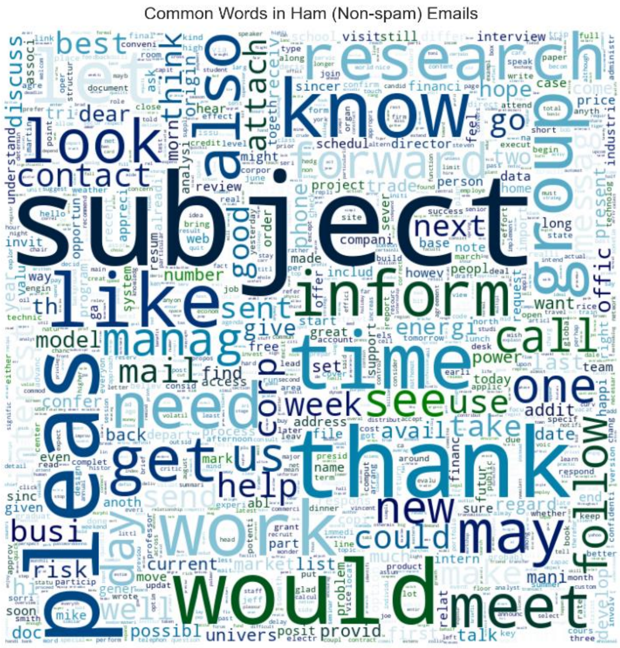
```
Accuracy Score for Logistic Regression: 0.9298
AUC Score for Logistic Regression: 0.95
Improvement for Logistic Regression Compared to Part a: -0.0351
Improvement for Logistic Regression Compared to Part c: -0.0439
Confusion Matrix for Logistic Regression:
[[34  0]
 [ 8 72]]

Accuracy Score for GaussianNB: 0.886
AUC Score for GaussianNB: 0.9103
Improvement for GaussianNB Compared to Part b: -0.0439
Improvement for GaussianNB Compared to Part c: -0.0526
Confusion Matrix for GaussianNB:
[[33  1]
 [12 68]]

Accuracy Score for MultinomialNB: 0.9123
AUC Score for MultinomialNB: 0.8868
Confusion Matrix for MultinomialNB:
[[28  6]
 [ 4 76]]
```

Problem 3: Email Spam Classifier

Part a

Preprocessing steps of textual data can be listed as follows:

- Lowercasing all of the words
- Removing punctuation characters
- Removing single character words, because usually they have no meaning.
- Removing stop words (like 'the', 'any', 'such', 'this') because they are common words in every context and don't give us information
- Removing non-English words
- Stemming (words like 'go', 'goes', 'going' will be replaced by a single 'go')



Part b

We use the Bayes theorem to find out the likelihood of an email being spam. Each email consists of multiple words. We should calculate the probability of each word being considered spam word.

$$P(S|W) = \frac{P(W|S)}{P(W|S) + P(W|H)}$$

We calculate spam probability of an email by combining the spam probabilities of all contained words in that email.

$$P(S|W) = \frac{P(S|W_1)P(S|W_2)\cdots P(S|W_N)}{P(S|W_1)P(S|W_2)\cdots P(S|W_N) + \big(1 - P(S|W_1)\big)\big(1 - P(S|W_2)\big)\cdots\big(1 - P(S|W_N)\big)}$$

It is better to use the alternative form of the above formula to avoid floating point underflow.

$$P(S|W) = \frac{1}{1 + e^{\left(\sum_{i=1}^{N}[\ln(1-p_i) - \ln p_i]\right)}}$$

Rare words can cause some problems in our calculations. We can handle that by using the corrected probability.

$$P(S|W) = \frac{s \cdot P(S) + n \cdot P(S|W)}{s + n}$$

Where s is the strength we give to background information about incoming spam and n is the number of occurrences of this word during the learning phase.

Part c

```
TP: 244
TN: 790
FP: 79
FN: 32
Accuracy: 0.9031
F1-score: 0.8147
Precision: 0.7554
TPR (Recall, Sensitivity): 0.8841
TNR (Specificity): 0.9091
```