Machine Learning Exercise 2

Michael Mitsios cs2200011

April 22, 2021

2 Naive Bayes

In this exercise our data is classified between 2 different classes: salary<=50K and salary>50K. Also our dataset consists of 2 different types of features: **continue** and **discrete.**
continue:['age','fnlwgt','education-num','capital-gain','capital-loss','hours-per-week']
discrete:['workclass',education,'marital-status','occupation','relationship','race','sex','native-country']

In a few words the continue features are the numbers and the discrete features are the strings.
Prior Class Probabilities: Denotes the probability that a random chosen observation y is associated with the k^{th} class. To compute this probability we follow this:

$$\pi_\kappa = p(Y = k) = \frac{N_k}{N}$$

where $N_k$is the population of the k^{th}class and N is all the rows. Each class eventually will have its own prior probability.
Parameter Estimation: In this section we have to compute the parameters for the continue features. We assume that the features are following the **Gauss distribution**. So the parameter that we have to define for each continue feature are: **μ, σ** (and for each class). In others words, we want the probability of the k^{th}class for the item x to **has the maximum value** when the item belongs to this class. Having said that, we will use the Maximum Likelihood Estimation (MLE) technique. In other words we are interested in obtaining the parameters of our model that maximize the Likelihood function given of a given set of observations. We can express the above mathematically as:

$$\mu, \sigma = argmax_{\mu,\sigma} p(x|\mu, \sigma)$$

Because our values are independent between them we can rewrite the above expression as the product of all the individual feature_values.

$$\mu, \sigma = argmax_{\mu,\sigma} \prod_{i=1}^{\mathrm{N}} p(x_i|\mu, \sigma)$$

Now we have to follow a specific approach. Because the ln function is concave this means that the point where the above equation will be reaching the maximum value, will be the same with the point which maximize the value of:

$$\mu, \sigma = argmax_{\mu,\sigma} ln \left\{ \prod_{i=1}^{\mathrm{N}} p(x_i|\mu, \sigma) \right\}$$

And in this form thanks to the Logarithm properties we can convert the product to sum of ln

$$\mu, \sigma = argmax_{\mu,\sigma} \sum_{i=1}^{\mathrm{N}} ln\left(p(x_i|\mu, \sigma)\right)$$

But as we have mentioned we are assuming that our data are following the Gauss distribution:$p(x_1|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} exp\left\{-\frac{(x_1-\mu)^2}{2\sigma^2}\right\}$and if we replace it on the above term:

$$\mu, \sigma = argmax_{\mu,\sigma} \sum_{i=1}^{\mathrm{N}} ln\left(\frac{1}{\sqrt{2\pi}\sigma} exp\left\{-\frac{(x_i-\mu)^2}{2\sigma^2}\right\}\right)$$

By using the log properties: $ln(ab) = ln(a) + ln(b)$ and $ln(\frac{a}{b}) = ln(a) - ln(b)$ we take:

$$\mu, \sigma = argmax_{\mu,\sigma} \sum_{i=1}^{\mathrm{N}} ln\left(1\right) - ln\left(\sqrt{2\pi}\right) - ln\left(\sigma\right) - \frac{(x_i-\mu)^2}{2\sigma^2}$$

Because the only thing we care about is the max we can remove the constant terms

$$\mu, \sigma = argmax_{\mu,\sigma} \sum_{i=1}^{\mathrm{N}} -ln\left(\sigma\right) - \frac{(x_i-\mu)^2}{2\sigma^2}$$

Finally by merging the max with the - we end up with a minimization problem

$$\mu, \sigma = argmin \sum_{i=1}^{\mathrm{N}} ln\left(\sigma\right) + \frac{(x_i-\mu)^2}{2\sigma^2}$$

Knowing that the slope at the minimum is zero, we can determine μ, σ analytically by calculating the partial derivatives and setting them to zero:
To define μ:

$$\frac{\partial}{\partial\mu} \sum_{i=1}^{\mathrm{N}} ln\left(\sigma\right) + \frac{(x_i-\mu)^2}{2\sigma^2} = 0$$

$$\sum_{i=1}^{\mathrm{N}} \frac{2(x_i-\mu)(-1)}{2\sigma^2} = 0$$

$$2(\sum_{i=1}^{\mathrm{N}} x_i - \sum_{i=1}^{\mathrm{N}} \mu) = 0$$

$$\sum_{i=1}^{\mathrm{N}} x_i = \sum_{i=1}^{\mathrm{N}} \mu$$

$$\sum_{i=1}^{\mathrm{N}} x_i = N\mu$$

$$\mu = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} x_i$$

To define σ:

$$\frac{\partial}{\partial \sigma} \sum_{i=1}^{\mathrm{N}} ln\left(\sigma\right) + \frac{(x_i - \mu)^2}{2\sigma^2} = 0$$

Following the same pattern and knowing that $\frac{\partial}{\partial \sigma} ln(\sigma) = \frac{1}{\sigma}$ and $\frac{\partial}{\partial \sigma} \frac{(x_i-\mu)^2}{2\sigma^2} = -\frac{(x_i-\mu)^2}{\sigma^3}$ we have:

$$\mathrm{N}\frac{1}{\sigma} - \sum_{i=1}^{\mathrm{N}} \frac{(x_i - \mu)^2}{\sigma^3} = 0$$

$$\mathrm{N}\frac{1}{\sigma} = \sum_{i=1}^{\mathrm{N}} \frac{(x_i - \mu)^2}{\sigma^3}$$

$$\mathrm{N}\frac{1}{\sigma} = \frac{1}{\sigma^3} \sum_{i=1}^{\mathrm{N}} (x_i - \mu)^2$$

$$\sigma^2 = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} (x_i - \mu)^2$$

$$\sigma = \sqrt{\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} (x_i - \mu)^2}$$

So for every class k and for every continue feature we have to determine the μ,σ that Maximize the Likelihood each time. **Note:**These calculations are refereed for the one column for one class.

Log-Posterior: The metric that will decide the class that a row belongs is the $logf_k(x)\pi_k$. But until now we have an estimation only for the continue values. By using these estimations we can calculate the density based on the Gauss distribution. In order to have a density value from the discrete features we follow the slides: fk(x) = p(X = x|Y = k): the density for X. So for every value $x_i$of every discrete feature for every class we are going to find the probability of

$$P(X = x|Y = k) = \frac{P(X = x \cap Y = k)}{P(Y = k)}$$

The $P(X = x \cap Y = k)$ describes in a few words the number of rows which has in the feature X the specific value x and at the same time belong to class k. This term divided by the number of rows that belong on the class k.
Having found these values then we can calculate the Log-posterior as the product of all the above values and the priors. Eventually we are going to have as many metrics as the number_of_row*number_of_classes.

Note: In order to avoid the 0 probabilities in the final product (in other word to have all the product 0) I used the m-estimation technique where I assume to have exactly one extra element that will satisfy the requested combination of values. In this way we are going to have a very small number but not 0 and we can compare the results even if both classes contain a P=0.

Prediction: Last but not least in order to make our prediction and classify our data, we have to scan all the rows and calculate the log-posterior for every class. The class with the bigger representative will be and our final prediction. The higher the metric the more suitable is the row for this class(the features fit the best with the specific class).

Evaluation

As for the evaluation I passed on the requested file the information needed. For the classification error I also included 2 more files that have 2 confusion matrices for a clearer view of the error. Our model seems to have very good results with accuracy around 82% both on train and test set.

Experimentation

In this section I made an extra experiment on our data, in order to figure out whether the mean=0 and s=1 help us or not. For this part the **Read of adult.data and adult.test** section has some code in comments. This code is used to standarise only the continue features and make each column have a **mean VERY close to 0** and a **variance VERY close to 1** (as we can see from the mean and variance of the column age). In this way all the columns are sharing the same mean and variance(or at least they have **VERY SIMILAR** mean and variance). Having done that we can calculate the density value of our continue features with a simpler form of the Gauss function. The first form of the Gauss function is:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$$

By setting the mean(μ)=0 and the variance(σ)=1 the new form we have is:

$$f(x) = \frac{1}{\sqrt{2\pi}} exp\left\{-\frac{x^2}{2}\right\}$$

But Naive Bayes doesn't really care about this change. He only **cares** about the populations and **how they are distributed**. For example if

we take age column, in order to distribute the age around 0 (in order to have mean=0), we have to make a Normalization to our data. This normalization will just take age and make the values smaller, in order to achieve the mean=0 ,var=1. The general distribution of data will be exactly the same as the old values. The new calculated estimators μ,σ from the Maximum Likelihood would be smaller than the ones that are not Normalized(in our case of age). But these new estimators will be adapt to the data the exact same way the previous ones did. And because the Normalization won't affect the original classification, it won't affect the Bayes either. The Gauss function will give results from the normalized values which are analogous to the previous results. In this way the same rows that belonged to the k^{th} class will continue to belong to the k^{th}after the standardization. This is why if we use StandardScaler to make the continues features have mean=0 and var=1 won't affect a bit the final results (remove the comments to testify).

Train set correct predictions: 24959 / 30162

Test set correct predictions: 12424 / 15060

Ridge Regression

In the same way with the previous exercise (Linear Regression). We have to minimize the least Squares error. Given the equation to be optimized:

$$argmin_\beta ||Y_n - X_n\beta||_2^2 + \lambda||\beta||_2^2$$

where λ>0 is a non-negative parameter and β are the coefficients of the model. From the previous term we can see the use of 2 euclidean norms raised to the square:

$$||Y_n - X_n\beta||_2^2$$

and

$$\lambda||\beta||_2^2$$

For an matrix X we know that:

$$||X||_2^2 = x^T x$$

$$argmin_\beta (Y - X\beta)^T (Y - X\beta) + \lambda\beta^T\beta$$

In order to find the β that minimize the above term we have to take the first derivative and set it to 0.

$$\frac{\partial (Y - X\beta)^T (Y - X\beta) + \lambda\beta^T\beta}{\partial\beta}$$

After that we can split the derivative into 2 terms in order to deal with them individually.

$$\frac{\partial (Y - X\beta)^T (Y - X\beta)}{\partial\beta}$$

The above term is the same one with the linear regression from the first exercise. We found out that the derivative will have this form:

$$-2X^TY + 2X^TXb$$

and

$$\frac{\partial \lambda \beta^T \beta}{\partial \beta}$$

From the Matrix calculus we know for fact that for a matrix x:

$$\frac{\partial x^T x}{\partial x} = 2x$$

and λ is just a number, so:

$$\frac{\partial \lambda \beta^T \beta}{\partial \beta} = 2\lambda\beta$$

Now that we have found the parts of the first derivative we have to put them together and equate then with 0.

$$-2X^TY + 2X^TX\beta + 2\lambda\beta = 0 =>$$

$$X^TX\beta + \lambda\beta = X^TY$$

Now we can have as common factor the matrix β. By doing that in order to make sense we have to leave behind the I identity matrix. $I \cdot \beta = \beta$ (1)

$$(X^TXI + \lambda I)\beta = X^TY$$

and due to (1) we can omit the first identity matrix:

$$(X^TX + \lambda I)\beta = X^TY$$

Last step is to solve the equation based on β. To do so we have to multiply both sides with the inverse array of $(X^TX + \lambda I)$——>$(X^TX + \lambda I)^{-1}$. The final equation has the form below:

$$\beta = (X^TX + \lambda I)^{-1}X^TY$$

This is the equation that will give us the best possible coefficients that minimize our RR (and make our predictions the best possible).

Ridge Regression vs Linear Regression

As we have seen from the purpose of the linear regression is to fit a model to a dataset and based on the produced "line" make some predictions. Sometimes we may want not to perfectly fit the model to our dataset but to have some small difference in order to avoid later on not as good predictions on our test set. There are occasions where our model will fit perfectly on our train set

(overfitting),which is something we don't want. We want our model to be able to generalize(not very much), because later on when we have to deal with points that we don't have feedback our model's predictions may have a big error. This is why we don't want to fit our model perfectly to the train data. This is why we use the Ridge Regression because it is a way to "regularize" a linear model. In other words, we prefer he coefficients that are smaller than those in the linear regression. The goal here as i said earlier is to add to our model a small error in order to make it more general. Also, our coefficients will become smaller as the λ grows bigger, due to the shrinkage penalty. The degree of the regularization is controlled by the λ value.

Bias vs Variance

In our case the more the λ the more we affect-generalize our model from its original form. If λ=0 then we are making the same process as Linear Regression.. Following this logic the lower the λ the more fitted our model will be to the train set and less to the test set, so we expect to have low bias and high variance. On the other side if we have a very high λ we expect to have a very general model, which eventually it will have low variance but high bias. Note: The closer to ∞ λ gets,more the model will taking the form of a horizontal line(the most general form our model can get).

Notes for the Ridge Regression Implementation

I tried to apply the standardization scaler on my datasets in order to have mean=0 and var=1. It seems that in this way the data will have different behavior. The line of the plot RSE_test/λ will have a monotonically decreasing form. This is reasonable because the Ridge Regression now needs the values of our features in order to produce the coefficients. So I avoided this approach.

The Ridge Regression code is following the logic we had for the Linear regression . First we have to add an extra column on our β in order to include the $\beta_0$parameter. When it comes to the coefficients we have to follow the logic of the equation we found earlier $\beta = (X^T X + \lambda I)^{-1} X^T Y$. First I want to clarify that the I is the identity matrix of β and it has the same length as β(without the β_0). In the LR implementation we included the β_0 in our β values as an extra column of 1 in the end, for our convenience. As we can see from the book Introduction to Statistical Learning, the $\beta_0$has been left out of the penalty term.

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

Ridge regression is very similar to least squares, except that the coefficients are estimated by minimizing a slightly different quantity. In particular, the ridge regression coefficient estimates $\hat{\beta}^R$ are the values that minimize

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2, \qquad (6.5)$$

So in our RR implementation having a β matrix that has en extra column of $\beta_0$will affect our identity matrix I. The I matrix will be forced to have the size of out new custom matrix β. But as the book said we have to left out the $\beta_0$from our penalty so for the column on the β matrix that represents the $\beta_0$we are going to the identity matrix and zero this specific one in the diagonal. In this way we can keep the LR implementation for the RSS part and penalty the coefficients based on their values(besides β_0).

Further Investigation

1

As we can see from our results as the coefficients are going to 0 the λ grows. This is reasonable because of this term:

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

We want the term above to have the **most possible minimum value** (closer to 0 we do not have negative terms). Here we have 2 terms that they need to have their min value:

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

and

$$\lambda \sum_{j=1}^{p} \beta_j^2$$

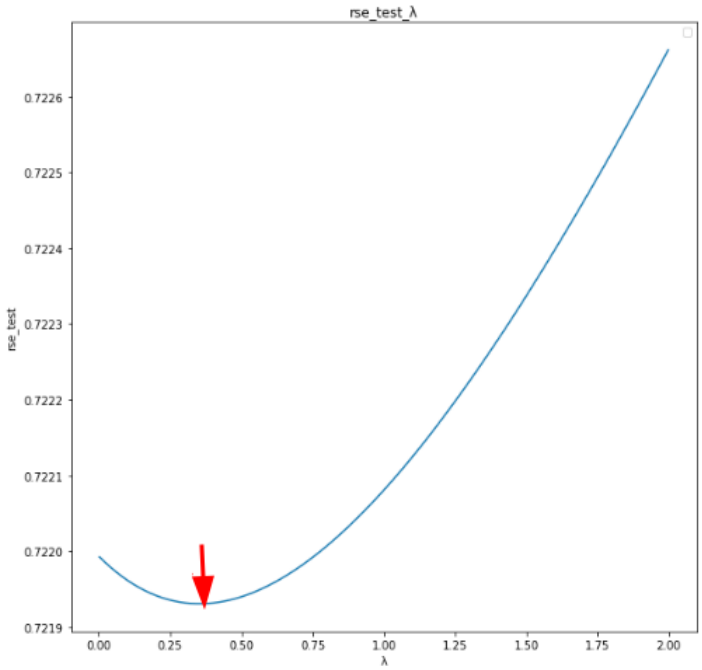
On the second term when we are making the **λ** parameter **bigger** the $\sum_{j=1}^{p} \boldsymbol{\beta_j^2}$ needs to become **smaller**. This is why the values of the coefficients on the sheets tend to 0. This becomes more obvious when we are using a very big λ. For example if we have λ=10000 then the produced coefficients will be [0.00532

0.001683 0.005339 0.003041 0.001238 0.002592 0.000265 0.015206] (very close to 0).

On the other hand when the $\sum_{j=1}^{p} \boldsymbol{\beta_j^2}$ becomes **bigger** ,the first term $\sum_{i=1}^{n} \left(\boldsymbol{y_i} - \boldsymbol{\beta_0} - \sum_{j=1}^{p} \boldsymbol{\beta_j x_{ij}}\right)^{\mathbf{2}}$ tends to become **smaller**. The only parameter that is under our control is $\beta_0$and it needs to become bigger in order to have a prediction that will remain very close to our target values(y_i). This is why eventually the β_0 parameter will grow bigger, to keep the balance.

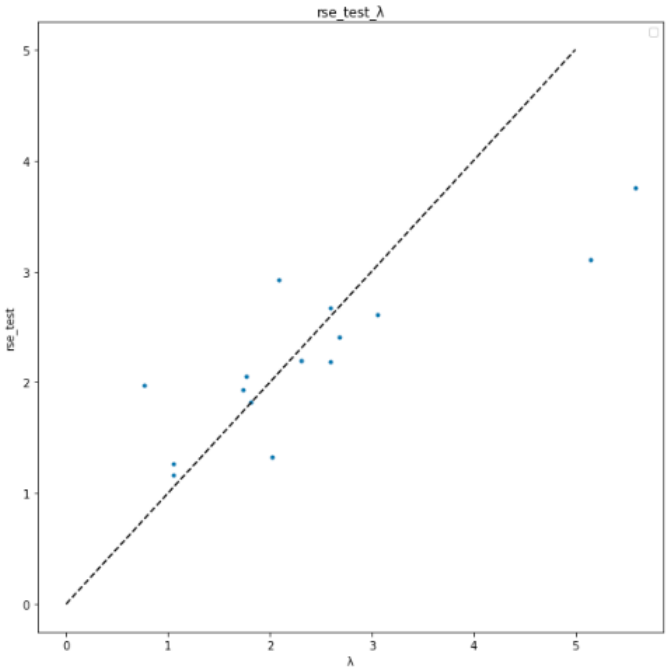
2

We plot the Sheet of Root Mean Squared Error to λ and we see that it has a curved form. There is a magic spot for λ≈0.35 where the model has its minimum error. To be more specific the minimum RSE is when the λ=0.354. This is the optimal value of λ that we are going to use. **Note:** The actual optimal value would be given from the implementation of a cross-validation technique to our model.



3

The below sheets are the 15 random points. These point have as x-axis the target-real value and on y-axis they have the predict value**(target,prediction)**. So the closer these points are on the line x=y the closer the target and predict values are. In other words **a good prediction is when the** $\frac{target}{prediction}$ **is close to 1**.



4 Linear vs Ridge Regression

Scores

Type	TRAIN_RSE	TEST_RSE	TEST_R2	TRAIN_R2
LR	0.662721486039448	0.7219930785731833	0.6943711796768238	0.5033798502381974
RR	0.6628706925832482	0.7219306835248362	0.6942335443220847	0.5034656829150823

The final results were the expected ones. The LR is a model where it fits to its train set as good as possible. The RR is a variation of LR where we add a small penalty, in order to avoid to much similarity to our train set and make our model more general. We add some bias to our model in order to lower our variance. In this way we expect the RR model to go better than the LR model when it comes to test set(increases the flexibility as fit decreases). On the other hand, the Linear Regression fit the train set in the best possible way rather than the Ridge model where we have a purposeful small penalty. That's why the LR has lower RSE on the train set than the RSE of the RR. The R2 error is inversely proportional to RSE.

Coefficients

Type	lcavol	lweight	age	lbph	svi	lcp	gleason
LR	0.57654319	0.61402	-0.01900102	0.14484808	0.73720864	-0.20632423	-0.02950288
RR	0.576747	0.59932053	-0.01867438	0.14538204	0.69836742	-0.19718781	-0.0329361

As we can see from the above results the coefficients of the RR model has the most coefficients closer to 0 than the Coefficients of the LR. We had predicted this behavior, and it happens because we want the shrinkage penalty on our β. Expect on our $\beta_0$where we expect its value to go up.

Bias

Type	Bias
LR	0.42917013284905003
RR	0.4939205964603519

We verified the above assumption.

Finally on our predictions we expect to have a smaller value on the RR model than on the LR model. This hypothesis is motivated from the fact that we added a shrinkage penalty to our function and in other words in a way we "decreased the slope" of the LR in order to take a more flexible model. I not sure 100% for the above, because on the extreme situation where the RR is a horizontal line all the predictions will be on this line. So the predictions will go closer to this line and won't get smaller necessarily. The predictions were not transferred here, but they are printed.