Exercise Set 2

- Submit the answer via Moodle *before* 1 December 2021 at 23:59.
- You can answer anonymously or write your name to the answer sheet, at your choice.
- The assignment should be completed by one person, but discussions and joint solving sessions with others are encouraged. Your final solution must however be your own. You are not not allowed to copy ready-made solutions or solutions made by other students. You are allowed to use external sources, web searches included.
- The problems can be discussed in the 29 November exercise session.
- Your answer will be peer-reviewed by you and randomly selected other students.
- The language of the assignments is English.
- The submitted report should be in a single Portable Document Format (pdf) file.
- Answer to the problems in the correct order.
- Read the general instructions and grading criteria in Moodle before starting with the problems.
- Main source material: James et al., Chapters 4-5 and 8-9. Please feel to use other resources as well. "James et al." refers to: James, Witten, Hastie, Tibshirani, 2021. An Introduction to Statistical Learning with applications in R, 2nd edition. Springer.
- Notice that you can submit your answer to the Moodle well before deadline and revise it until the deadline. Therefore: please submit your answer well in advance, already after you have solved some problems! Due to peer review process we cannot grant extensions to the deadline. Even though the Moodle submission may occasionally remain open for a while after 23:59, the submission system will eventually close. If you try to submit your answers late you will not get any points (including peer-review points) from this Exercise Set. You have been warned.
- Please double-check that the submitted pdf is formatted properly and, e.g., contains all figures. It seems to be especially difficult to produce properly formatted pdf files with Jupyter Notebooks: remember to check the produced pdf or consider using R Markdown.

Problem 8

[6 points]

Objective: naive Bayes classifier

In this problem you will study the Palmer penguins. Please first read the dataset description at https://cran.r-project.org/web/packages/palmerpenguins/readme/README.html You should use the data in `penguins_train.csv` as your training data and `penguins_test.csv` as your testing data. You task is to build a Naive Bayes (NB) classifier for a binary classification task of classifying the penguin species as $y \in \{\text{Adelie}, \text{notAdelie}\}$ (`notAdelie` combining Gentoo and Chinstrap species) based on a total of 4 morphological and weight measurements of the individual penguins, denoted by $\mathbf{x} = (x_1, x_2, x_3, x_4)^T \in \mathbb{R}^4$. You do not need to build a generic classifier (such as `naiveBayes` in the R `e1071` library; it is enough that your classifiers works for this particular task). You classifier should use normal distributions to model the covariates.

Task a

Compute and report the means and standard deviations of each of the attributes in the training set for the both classes separately. Estimate and report the class probabilities, using Laplace smoothing *with pseudocount* of 1 on the training set. I.e., you should produce a total of 18 numbers from this task.

Task b

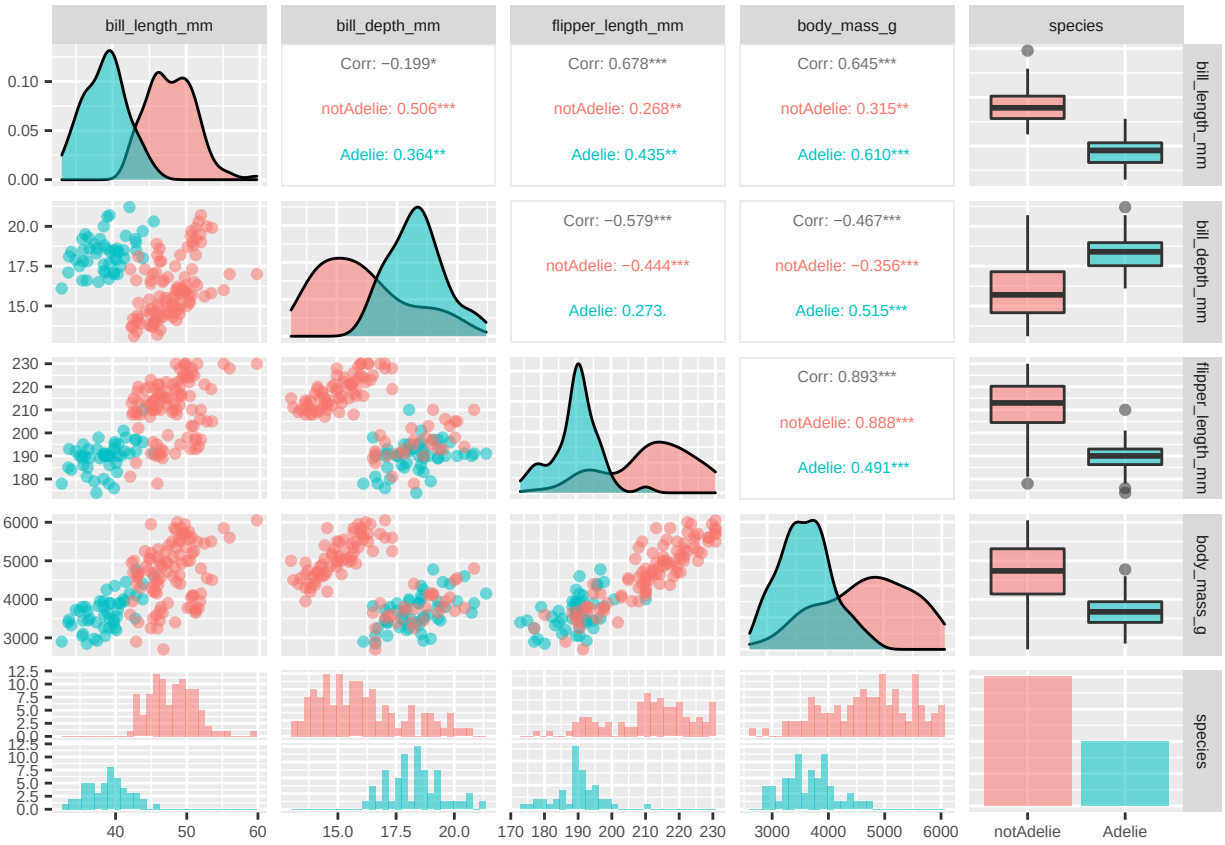
Now you can find the class-specific expressions for $p(\mathbf{x} \mid y)$ needed by the NB classifier. Remember that according to NB assumption the dimensions are independent, and hence, you can represent the class-specific $p(\mathbf{x} \mid y)$ likelihoods as products of 4 1-dimensional normal distributions. Write down the formula needed to compute the posterior probability of the class being `Adelie` $\hat{p}(y = \text{Adelie} \mid \mathbf{x})$ as a function of the four measurements in $\mathbf{x}$ and the statistics (means, standard deviations, class probabilities) you computed in the task a above.

Task c

Compute and report the classification accuracy for you classifier on the test set, by using the formula you derived in the task b above.

Hint: When computing classification accuracy you can use the following rule to obtain "hard" classes:

$$\hat{y} = \begin{cases} \text{Adelie} & , \quad \hat{p}(y = \text{Adelie} \mid \mathbf{x}) \geq 0.5 \\ \text{notAdelie} & , \quad \hat{p}(y = \text{Adelie} \mid \mathbf{x}) < 0.5 \end{cases}$$



Problem 9

[5 points]

Topic: logistic regression, discriminative vs. generative classifiers [Ch. 4]

In this problem you should apply logistic regression (with an intercept term) to the same dataset as in the previous problem. In R you can use `glm` to do the logistic regression; first read through Sect. 4.7.2 of James et al.

Task a

Train the logistic regression model on the training data and report the model coefficients as well as the accuracy on the training and testing data (you can use the same rule to obtain hard classes from probabilities as in the previous problem). Make a plot that shows the dataset where the x-axis is the linear response t (where $t = \beta^T \mathbf{x}$) and y-axis the probability $Pr(y = \text{Adelie} \mid \mathbf{x})$ estimated by logistic regression for this particular linear response.

Indicate the values of the linear response of your datapoints in the plot, e.g., by plotting a data point corresponding to a individual penguin so that the x-coordinate is given by $\beta^T \mathbf{x}$ and the y-coordinate is ~ 0 for class `notAdelie` and ~ 1 for class `Adelie`. Plot the penguins in the training and testing data sets differently, e.g., with a different glyph and/or colour (remember to explain which is which!).

Task b

The logistic regression may issue warnings (in R: `Warning: glm.fit: algorithm did not converge` or `Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred`, in Python the warnings may look different or, depending on implementation, the model fitting may even crash). Why does this happen? Do you need worry and how could you fix it so that you would not get the warnings and/or crashes?

Task c

Explain the difference and pros and cons between generative classifier (such as NB) and discriminative classifier (such as logistic regression). For this dataset, which of the approaches would be better and why?

Problem 10

[5 points]

Objective: Understanding discriminative vs. generative learning.

Download Ng et al. (2001). You **do not need to read the full paper** in detail or understand all of the details! Rather try to find the answers to the following questions.

Reference: Ng, Jordan (2001) On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. NIPS. http://papers.nips.cc/paper/2020-on-discriminative-vs-generative-classifiers-a-comparison-of-logistic-regression-and-naive-bayes.pdf

Task a

Read the *abstract* and the *Introduction* (Sect. 1). Is discriminative learning better than generative learning, according to the authors? Justify your answer.

Task b

By a "parametric family of probabilistic models", the authors mean a set of distributions, where each distribution is defined by a set of parameters. An example of such a family is our friend, the family of normal distributions where the parameters are μ and Σ. Ng and Jordan denote by h_{Gen} and h_{Dis} two models chosen by optimizing different things. Find an explanation of what these "things" are that are being optimized in the paper, i.e., what characterizes these two models. Which two families do the authors discuss, and what are the (h_{Gen}, h_{Dis}) pairs for those models?

Task c

Study Figure 1 in the paper. Explain what it suggests (see the last paragraph of the Introduction). Reflect this on the previous item.

Problem 11

[5 points]

Objective: comparing classifiers on synthetic data, application of different classifiers

Consider a toy data with a binary class variable $y \in \{0, 1\}$ and with two discrete features $x_1 \in \{0, 1\}$ and $x_2 \in \{0, 1, 2\}$. The true distribution from which the data is sampled is such that $P(y = 1) = 1 - P(y = 0) = 0.55$ and the class-conditioned distributions for (x_1, x_2) are specified as follows:

$$P(x_1, x_2 \mid y = 0) = \begin{cases} 0.2 & , \quad x_1 = 0 \wedge x_2 = 0 \\ 0.4 & , \quad x_1 = 0 \wedge x_2 = 1 \\ 0 & , \quad x_1 = 0 \wedge x_2 = 2 \\ 0.1 & , \quad x_1 = 1 \wedge x_2 = 0 \\ 0.2 & , \quad x_1 = 1 \wedge x_2 = 1 \\ 0.1 & , \quad x_1 = 1 \wedge x_2 = 2 \end{cases}$$

and

$$P(x_1, x_2 \mid y = 1) = \begin{cases} 0.6 & , \quad x_1 = 0 \wedge x_2 = 0 \\ 0.1 & , \quad x_1 = 0 \wedge x_2 = 1 \\ 0.1 & , \quad x_1 = 0 \wedge x_2 = 2 \\ 0.1 & , \quad x_1 = 1 \wedge x_2 = 0 \\ 0 & , \quad x_1 = 1 \wedge x_2 = 1 \\ 0.1 & , \quad x_1 = 1 \wedge x_2 = 2 \end{cases}$$

Task a

Is the naive Bayes (NB) assumption valid for data generated by this procedure? Explain why or why not.

Task b

Generate a test data set of 10000 points and 10 training data sets of sizes $n \in \{2^4, 2^5, \ldots, 2^{13}\}$, respectively.

For each of the training set, train the following classifiers that output probabilities.

- Naive Bayes (NB) (e.g., `naiveBayes` from library `e1071`)
- Logistic regression without interaction term (e.g., `glm`)
- Logistic regression with interaction term (e.g., `glm`)
- Support Vector Machine (SVM) with radial basis function (e.g., `svm` from library `e1071`)
- Bayes classifier that uses the true class conditional probabilities (that you know in this case!) to compute $p(y \mid x_1, x_2)$ for a given (x_1, x_2) - no probabilistic classifier can do better than this
- "Dummy classifier" that does not depend of x and that always outputs probability $\hat{p}(y = 1 \mid x_1, x_2) = 0.55$ ("dummy" means here that the classifier output does not depend on the covariates, it is always a good idea to include dummy classifier in your comparison!)

Make a plot - or table - where you show for different models the classification accuracy and perplexity on the test set as a function of training data set size.

Hint: Perplexity is another and perhaps easier-to-interpret way to express log-likelihood. Perplexity on the test set of size $n = 10000$ is $\exp(-L/n)$, where L is the log-likelihood given by $L = \sum_{i=1}^{n} \log p(y_i \mid x_i)$ and (x_i, y_i) are the data items in the test set. The perplexity is 2 for coin flipping for which $p(y_i \mid x_i) = 0.5$ and 1 for a "perfect classifier" that always gives $p(y_i \mid x_i) = 1$.

Hint: Most of the code needed is given below in the instructions.

Task c

Discuss your observations and what you can conclude. Which of the models above are probabilistic, discriminative, and generative? How do accuracy and perplexity (log-likelihood) compare? Is there a relation to the insights from Problem 11 above? Why does logistic regression with the interaction term perform so well

for larger datasets? Does your dummy classifier ever outperform other classifiers, or do other classifiers ever outperform the Bayes classifier?

Instructions

The following R function should provide you the needed datasets:

```
set.seed(42)
makedata <- function(n) {
  a <- data.frame(y =factor(c(0,0,0,0,0,0,1,1,1,1,1,1)),
                  x1=factor(c(0,0,0,1,1,1,0,0,0,1,1,1)),
                  x2=factor(c(0,1,2,0,1,2,0,1,2,0,1,2)),
                  p=c(0.45*c(0.2,0.4,0.0,0.1,0.2,0.1),
                      0.55*c(0.6,0.1,0.1,0.1,0.0,0.1)))
  a[sample.int(12,n,replace=TRUE,prob=a$p),,drop=FALSE]
}
data_test <- makedata(10000) # test data set
data <- lapply(2^(4:13),makedata) # training data sets
```

You can include the interaction term in logistic regression by writing, e.g.,:

```
model <- glm(y ~ x1*x2,data[[4]],family=binomial)
```

"*" implies that an interaction should be included in the model, instead of "+" which assumes only additive effects.

Here, it is useful to make a function `phat` that takes the training and testing data as input and outputs the the probabilities $p(y = 1 \mid x_1, x_2)$ and then computes accuracy and perplexity by using theses probabilities. Below, you can find one way to accomplish this, if you use R. You can of course do the problem in some other way or language as well.

```
library(e1071)
## estimate phat. The idea is to make a function that outputs the predicted
## probabilities and the parameters can be modified "easily" for different
## models.
phat <- function(data.tr, # training data
                 data.te=data_test, # test data - here data_test
                 form=y ~ x1+x2, # formula
                 model=function(f,d) naiveBayes(f,d,laplace=1), # model family
                 m=model(form,data.tr), # model trained on data.tr
                 pred=function(m,x) predict(m,x,type="raw")[,2]) { # function to make the predictions b
  pred(m,data.te)
}

## accuracy
acc <- function(p,y=data_test$y) mean(ifelse(p>=0.5,1,0)==y)

## perplexity
perp <- function(p,y=data_test$y) exp(-mean(log(ifelse(y==1,p,1-p))))
```

Then you can for example collect the requested numbers for NB into dataframe `res`:

```
# collect results to data frames
res <-  data.frame(n=sapply(data,nrow))

phat_nb <- lapply(data,phat)
res$nb_acc  <- sapply(phat_nb,acc)
```

```
res$nb_perp <- sapply(phat_nb,perp)
```

Now you have your results for the NB. Then it remains only to do the same for all other classifiers and add them to your table in `res`, which is easy because you just use different libraries (but unfortunately, there are slight differences in model options) - the SciPy sklearn is a bit nicer in this respect.

Probabilities for logistic regression:

```
phat_glm <- lapply(data,function(d) {
  phat(d,
       model=function(f,d) glm(f,d,family=binomial),
       pred=function(m,x) predict(m,x,type="response"))
})
```

You can compute the accuracies and perplexities the same way you did for the NB above by using `phat_glm` instead of `phat_nb`. Logistic regression with interaction, i.e., `x1*x2` instead of `x1+x2` in the formula:

```
phat_glmx <- lapply(data,function(d) {
  phat(d,
       form=y ~ x1*x2,
       model=function(f,d) glm(f,d,family=binomial),
       pred=function(m,x) predict(m,x,type="response"))
})
```

SVM:

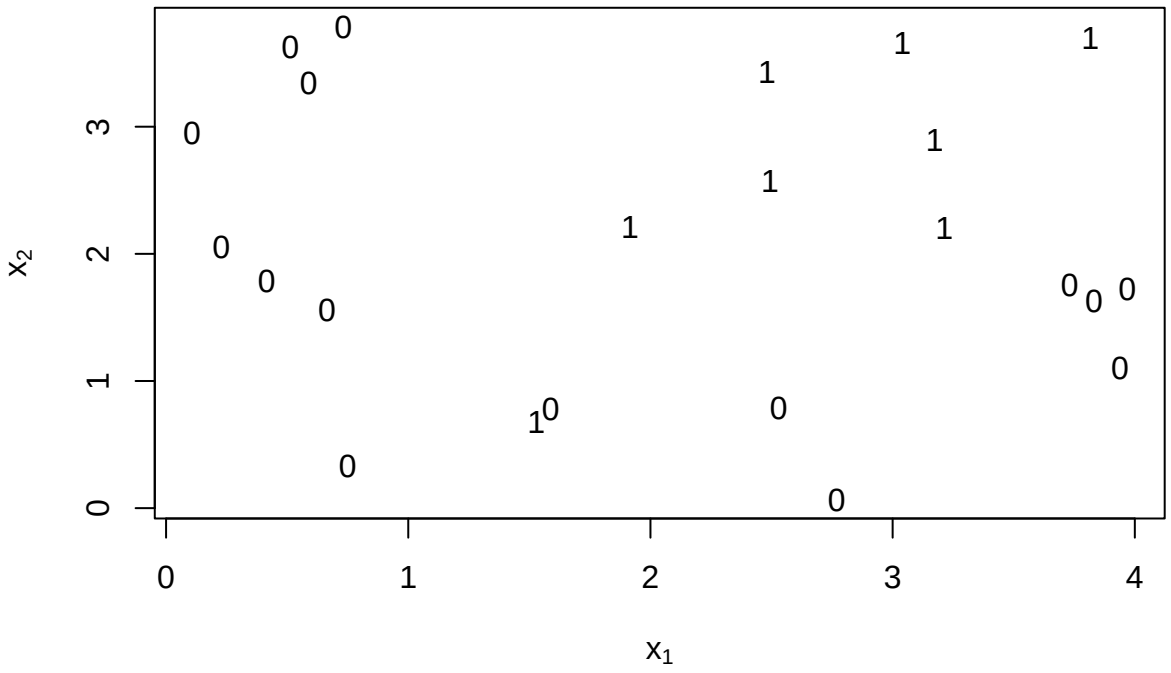
```
phat_svm <- lapply(data,function(d) {
  phat(d,
       model=function(...) svm(...,probability=TRUE),
       pred=function(m,x) attr(predict(m,x,probability=TRUE),"probabilities")[,"1"])
})
```

Problem 12

[6 points]

Objectives: basic principles of decision trees

Familiarise yourself with Section 8.1 of James et al. Pick one of the impurity measures presented in Equations (8.5), (8.6), or (8.7). Then simulate the tree building algorithm by hand.



Task a

Sketch a run of the classification tree algorithm with the toy data set in the figure above(binary classification task in $\mathbb{R}^2$) and draw the resulting classification tree. Report the values of the chosen impurity measure for each split and try to choose the splits that obtain the best impurity measure. You do not need to worry about over-fitting here: the resulting classification tree should fit the training data with no error. Don't worry that you don't count the classes exactly right or that your results are not super-accurate as long as they are "in the ballpark".

Problem 13

[5 points]

Learning objectives: basics of k-NN method.

Consider the problem of applying *k-nearest neighbour* (k-NN) classifier to the training dataset $D = \{(x_i, c_i)\}_{i=1}^{14}$, where the covariates $x_i \in \mathbb{R}$ and the classes $c_i \in \{-1, +1\}$ are given in below. You should be able to do this by pen and paper.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
x_i	0.0	2.0	3.0	5.0	6.0	8.0	9.0	12.0	13.0	15.0	16.0	18.0	19.0	21.0
c_i	+1	+1	+1	−1	+1	+1	+1	−1	−1	−1	+1	−1	−1	−1

Task a

Where are the classification boundaries for the 1-NN and 3-NN classifiers? What are the respective classification errors on the training dataset?

Task b

How does the choice of k in k-NN affect the classification boundary (not in the above example but in general)? Give examples of the behaviour for extreme choices.

Problem 14

[6 points]

Topic: SVM [Ch. 9]

To prepare for this exercise, you may want to follow the steps in the Lab in Sect. 9.6 of James et al. You don't need to do the part on ROC curves. In this task you can use the data in file `data_svm_train.csv` to train the models and the data in `data_svm_test.csv` to compute the accuracies. Both datasets contain 100 points in (x_1, y) having class $y = -1$ and covariate x_1 sampled from normal distribution with zero mean and unit variance, and 100 points having class $y = +1$ with x_1 sampled from normal distribution with zero mean and standard deviation of 5.

Task a

Apply a support vector classifier - an SVM with a linear kernel (`kernel="linear"`) - with a couple of different tuning parameters (argument `cost`). Confirm that a linear kernel has no hope in performing well in this task. Try an RBF kernel (`kernel="radial"`) to see that performance improves. Experiment with different tuning parameters (see the documentation of function `svm` and the Lab in the textbook for information about the parameters). Report the accuracies and the respective tuning parameters in a table.

Hint: With `scikit-learn` you can use, e.g., `sklearn.svm.SVC` which implements all of the above mentioned kernels. You may find the documentation at https://scikit-learn.org/stable/modules/svm.html useful.

Task b

Add a new covariate $x_2 = x_1^2$ (squared feature) to your data and try again with SVM with linear kernel and default parameters and report the classification accuracy. Does the classification accuracy improve? Visualize the test data in the (x_1, x_2) coordinates (with different classes in different glyphs and/or colors) and indicate the decision boundary of the linear SVM model. Briefly explain how this all is related to the kernel approach in SVMs.

Hint: In R you can make plots as described in James et al. section mentioned above. With Python this may require a little bit more code, see, e.g., https://scikit-learn.org/stable/auto_examples/svm/plot_iris_svc.html

Problem 15

[2 points]

Objectives: self-reflection, giving feedback of the course

Tasks

- Write a learning diary of the topics of lectures 5-8 and this exercise set.

Instructions

The length of your reply should be 1-3 paragraphs of text. Guiding questions: What did I learn? What did I not understand? Was there something relevant for other studies or (future) work?