Excercise 1.a

The following results were obtained after running my script (for $k = 1, ..., 9$) :

Value of k	Accuracy	Precision	Recall
1	0.81	0.81	0.93
2	0.81	0.81	0.93
3	0.71	0.79	0.79
4	0.71	0.79	0.79
5	0.76	0.85	0.79
6	0.76	0.85	0.79
7	0.76	0.8	0.86
8	0.76	0.8	0.86
9	0.71	0.79	0.79

Excercise 1.b

Assume that your program is executed in a way that explores many different values of k. An expert will argue that by looking at the output file, we cannot determine what the best value of k is. How do we need to structure our data and what process do we need to follow in order to be able to determine the best k?

- picking the k which performs best on the test set will lead to overfitting: we are picking that k which performs the best on this *particular* test set. On a new training and test set it is very much possible that we would get a different k.
- in order to avoid this we have to do a *nested* cross validation:
  - nested because in the inner loop we determine the ideal k (model selection) and in the outer loop we test the selected model (model assessment).
  - it goes like this:

Algorithm 1 Nested Cross Validation

Split data into m equal sized folds $\{f_1, ..., f_m\}$
for i in 1:m **do**
  1. $X = \{f_1, ..., f_m\} \setminus \{f_i\}$ #data for inner loop
  2. $t = \{f_i\}$ # test fold outer loop

  for k in 1:K **do**
    for q in 1:$(m-1)$ **do**
      1. $X_{inner} = X \setminus \{f'_q\}$
      2. $t_{inner} = \{f'_q\}$
      3. fit KNN_{inner} with hyperparameter k on X_{inner}
      4. use model to predict on t_{inner} & record $Acc_{k,q,inner}$
    end for
    record $Acc_k = sum(Acc_{k,q,inner})/(m-1)$
  end for

  3. Fit KNN_{outer} with $k = \text{argmax}_k\, Acc_k$ on X
  4. predict on t & record $Acc_{i,outer}$
end for
compute $Acc_{final} = sum(Acc_{i,outer})/(m)$ for model assessment

Note that *Acc* stands for *accuracy* and can be replaced by any other (classification) performance metric. Further, note that in reality there is no real fitting of kNN on X and X_{inner} since kNN is a model-free/non-parametric approach. I just wrote it there to better illustrate that different values of k are used in the for-loop.

Now the interesting thing is that Acc_{final} is an estimate of the *expected test* performance of our KNN classifier but we don't have a specific value for hyperparameter k yet, so how do we obtain that? There are two options:

- we determine k via a regular cross validation on *all* of the data (so no inner loop this time), which would yield the final model (but its expected test performance is still assumed to be given by Acc_{final}). This was thaught in Prof. Maathuis class *Computational Statistics.*
- we pick that k which appeared most of frequent in the inner loop of the nested cross validation (Prof. Borgwardt explained this in the lecture - I asked that question).

Excercise 1.c

Using 'big O' notation (i.e. Landau symbols) and assuming all data have been preprocessed, what is the time complexity of the training step in k-NN? What is the space complexity?

This is a verbatim quote from Prof. Borgwardt's lecture from 21st of October: "NN-classification is an instance of instanced-based learning and lazy learning, so [...] our prediction works directly on the training data [...] [and so] no model is derived [...] [and thus,] no training happens here. So all computations that happen, happen at **test** time [...] [so] there is no training process."

Let n denote the number of training points and d the dimension of the feature space.

Since training doesn't happen, I conclude that its time complexity doesn't exist. However, I would argue that a space complexity of $\mathcal{O}(n \cdot (d+1))$ exists since we have to store the training data and the labels.

Excercise 1.d

Now focusing on the prediction step, is its complexity different in a problem with c classes, where $c > 2$?

No, the (space and time) complexity is not different because:

- In case of $c = 2$, the complexity of the prediction is a constant, i.e. $\mathcal{O}(1)$, since majority voting does not depend on n or d.
- For $c > 2$, where majority voting is still applied, just over more classes, it is also $\mathcal{O}(1)$ for the same reason as in the case of binary classification.

Excercise 1.e

In your implementation, you used the Euclidean distance, which is a metric. Does k-NN work with other metrics such as the Manhattan distance as well (1)? Moreover, does it also work for semimetrics, i.e. functions that do not satisfy the triangle inequality, such as DTW (2)? Briefly justify your answer.

(1) Yes, kNN also works with other metrics, because the principle of the algorithm stays the same, just the "measurment unit" is a different one. Prasath et al. (2017, p. 2) confirm this:

> "Chomboon et al analyzed the performance of KNN classifier using 11 distance measures. These include Euclidean, Mahalanobis, Manhattan, Minkowski, Chebychev, Cosine, Correlation, Hamming, Jaccard, Standardized Euclidean and Spearman distances. [...] The results showed that the Manhattan, Minkowski, Chebychev, Euclidean, Mahalanobis, and Standardized Euclidean distance measures achieved similar accuracy results and outperformed other tested distances."

(2) Yes, semimetrics such as DTW can be used and seem to work well, as Geler, Kurbalija, Radovanović, and Ivanović (2014, p. 2f.) state in their paper "Impact of the Sakoe-Chiba band on the DTW time series distance measure for kNN classification":

> "The advantages of Euclidean distance [...] have made it, over time, probably one of the most commonly used similarity measure for time series. However, due to the linear aligning of the points of the time series it is sensitive to distortions and shifting along the time axis. To address this shortcoming, many different elastic similarity measures were proposed. Among them, some of the most widely used and studied are Dynamic Time Warping (DTW) [...]. It is reported that the elastic measures can have better classification accuracy than Euclidean distance and that constraining the warping window can further improve the accuracy of these measures.

Excercise 1.f

So far, you have used k-NN for classification. Is it possible to use k-NN for regression as well?

Yes, kNN-regression is very well possible (I've coded it myself in computational statistics). The way it differs from kNN-classification is that instead of taking the majority vote (for classification prediction for x'), knn-regression averages the y values of the k nearest neighbours and then that average serves as the prediction (i.e. the prediction is now a real number and not a label).

Excercise 2.a

Here is the output you obtain when you run my script:

`output_summary_class_2.txt:`

Value	clump	uniformity	marginal	mitoses
1	0.315	0.836	0.821	0.97
2	0.103	0.081	0.08	0.019
3	0.209	0.053	0.067	0.005
4	0.145	0.019	0.011	0
5	0.182	0	0.007	0.002
6	0.034	0.005	0.009	0
7	0.002	0.002	0	0.002
8	0.009	0.002	0	0.002
9	0	0.002	0.002	0
10	0	0	0.002	0

`output_summary_class_4.txt:`

Value	clump	uniformity	marginal	mitoses
1	0.013	0.018	0.137	0.574
2	0.013	0.037	0.091	0.112
3	0.054	0.115	0.105	0.126
4	0.049	0.138	0.119	0.049
5	0.188	0.124	0.087	0.018
6	0.067	0.106	0.078	0.013
7	0.09	0.069	0.05	0.027
8	0.166	0.115	0.105	0.031
9	0.063	0.018	0.014	0
10	0.296	0.258	0.215	0.049

Use the probabilities reported to predict the class label of the following data point:

$$\boldsymbol{x}_i = [\ \text{clump} = 6, \text{uniformity} = 2, \text{marginal} = 2, \text{mitoses} = 1\ ]^T$$

So the classfication rule is:

$$\hat{y_i} = \underset{y_i}{\text{argmax}}\ \mathbb{P}(Y = y_i | X = \boldsymbol{x}_i) \quad \text{where } y_i \in \{2, 4\}$$

Further, from the "naive" assumption it follows that,

$$\mathbb{P}(Y = y_i | X = \boldsymbol{x}_i) \propto \mathbb{P}(Y = y_i)\mathbb{P}(X = \boldsymbol{x}_i | Y = y_i) = \mathbb{P}(Y = y_i) \prod_{j=1}^{4} \mathbb{P}(X = x_i^{(j)} | Y = y_i)$$

Calculation for $\hat{\mathbb{P}}(Y = 2|X = \boldsymbol{x}_i)$:

- prior: $\hat{\mathbb{P}}(Y = 2) = \frac{\text{count}(Y=2)}{n} = \frac{440}{664} = 0.66$
- $\hat{\mathbb{P}}(Y = 2|X = \boldsymbol{x}_i) \propto \hat{\mathbb{P}}(Y = 2)\hat{\mathbb{P}}(X = \boldsymbol{x}_i|Y = 2) = 0.66 \times \prod_{j=1}^{4} \hat{\mathbb{P}}(X = x_i^{(j)}|Y = 2)$
  $= 0.66 \times 0.034 \times 0.081 \times 0.08 \times 0.97 = 0.000141048864$

Calculation for $\hat{\mathbb{P}}(Y = 4|X = \boldsymbol{x}_i)$:

- prior: $\hat{\mathbb{P}}(Y = 4) = \frac{\text{count}(Y=4)}{n} = \frac{224}{664} = 0.34$
- $\hat{\mathbb{P}}(Y = 4|X = \boldsymbol{x}_i) \propto \hat{\mathbb{P}}(Y = 4)\hat{\mathbb{P}}(X = \boldsymbol{x}_i|Y = 4) = 0.34 \times \prod_{j=1}^{4} \hat{\mathbb{P}}(X = x_i^{(j)}|Y = 4)$
  $= 0.34 \times 0.067 \times 0.037 \times 0.091 \times 0.574 = 4.402594924 \times 10^{-05}$

Prediction for $\boldsymbol{x}_i$ is then $\hat{y}_i = 2$ because $\hat{\mathbb{P}}(Y = 2|X = \boldsymbol{x}_i) > \hat{\mathbb{P}}(Y = 4|X = \boldsymbol{x}_i)$

Excercise 2.b

The data contain missing values (NaN's). How do these affect the computation of the probabilities in Figure 4?

When computing the various probabilities for each value/feature in figure 4, we don't include the NaN's in the count of the denominator. To illustrate this with an example: Denote "clump" with x_c then the probabiliy of $x_c = 5$ in group $y = 2$ is calculated as

$$\hat{P}(x_c = 5|y = 2) = \frac{\text{count}(x_c = 5 \cap y = 2)}{\text{count}(x_c \neq NaN \cap y = 2)}$$

Therefore, the NaN's don't have any impact on the probabilities in figure 4.

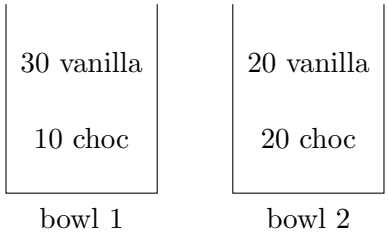
Excercise 2.c

What strategy can you suggest to overcome the problem known as 'zero-frequency'? This occurs when you need to compute the probability of a feature/value and class but there are zero instances of it in the training data, i.e. $P(X_j = x_j|Y = y_i) = 0$ **for a given** i **and** j**.**

An intuitive and pragmatic solution to the zero-frequency problem would be to to add a 1 to each feature/value count of each class. Therefore, any zero instances are eliminated, while their respective class prior probabilities remain unchanged.

Excercise 3.a

You pick a bowl at random and draw a vanilla brownie. Use Bayes' theorem to calculate the probability that the bowl you selected is bowl 1.



Let random variable $X \in \{1, 2\}$ denote which bowl we pick and let random variable $Y \in \{vanilla, choc\}$ denote what type of brownie we interested.

So what we are interested in is:

$$\mathbb{P}(X = 1|Y = vanilla)$$

Using Bayes' theorem we know that:

$$\mathbb{P}(X = 1|Y = vanilla) = \frac{\mathbb{P}(X = 1) \cdot \mathbb{P}(Y = vanilla|X = 1)}{\mathbb{P}(Y = vanilla)} \tag{1}$$

In our problem above we have:

- $\mathbb{P}(X = 1) = 0.5$ as prior since we have two bowls with equally likely chance of getting picked.
- $\mathbb{P}(Y = vanilla|X = 1) = 0.75$ since 30 brownies out of 40 are vanilla brownies.
- we can derive $\mathbb{P}(Y = vanilla)$ by marginalizing:

$$\begin{aligned}\mathbb{P}(Y = vanilla) &= \mathbb{P}(X = 1)\mathbb{P}(Y = vanilla|X = 1) + \mathbb{P}(X = 2)\mathbb{P}(Y = vanilla|X = 2) \\ &= 0.5 \cdot 0.75 + 0.5 \cdot 0.5 = 0.626\end{aligned}$$

Pluggin these into (1) yields:

$$\mathbb{P}(X = 1|Y = vanilla) = \frac{0.5 \cdot 0.75}{0.625} = 0.6$$

Excercise 3.b

Calculate the posterior probability $\mathbb{P}(N|D)$ for all valid values of N.

Run my script `nbayes_uber.py` via console:

```
>python nbayes_uber.py --outdir output
```

You can see the output directly in the console. Additionally, it will generate `output_nbayes_uber.txt` and `output_nbayes_uber_max.txt` where you can see all the posterior probabilities for different N and the maximum posterior probability with its respective N.

References

Geler, Z., Kurbalija, V., Radovanović, M., & Ivanović, M. (2014). Impact of the sakoe-chiba band on the dtw time series distance measure for knn classification. In *International conference on knowledge science, engineering and management* (pp. 105–114). Retrieved from `https://perun.pmf.uns.ac.rs/radovanovic/publications/2014-ksem-dtw.pdf`

Prasath, V., Alfeilat, H. A. A., Hassanat, A., Lasassmeh, O., Tarawneh, A. S., Alhasanat, M. B., & Salman, H. S. E. (2017). Effects of distance measure choice on knn classifier performance-a review. *arXiv preprint arXiv:1708.04321*. Retrieved from `https://arxiv.org/pdf/1708.04321.pdf`