Machine Learning 2024 — Homework 2

Marta Vasconcelos, Gonçalo Nunes
(ist1106127, ist1107097)

All code can be found on GitHub: `https://github.com/martinha-ssv/ML-Homeworks`

I Pen and Paper

We collected four positive (P) observations, $\{x_1 = (A, 0), x_2 = (B, 1), x_3 = (A, 1), x_4 = (A, 0)\}$, and four negative (N) observations, $\{x_5 = (B, 0), x_6 = (B, 0), x_7 = (A, 1), x_8 = (B, 1)\}$. Consider the problem of classifying observations as positive or negative.

Question 1. Compute the F1-measure of a kNN with and Hamming distance using $k = 5$ leave-one-out evaluation schema. Show all calculus.

Let's begin by structuring our data on a table:

Table 1: Supplied Data

	y_1	y_2	y_{out}
x_1	A	0	P
x_2	B	1	P
x_3	A	1	P
x_4	A	0	P
x_5	B	0	N
x_6	B	0	N
x_7	A	1	N
x_8	B	1	N

Now, first, we will calculate a distance matrix, using the Hamming distance as our metric.

Table 2: Distance matrix calculated using the Hamming distance

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	0	2	1	0	1	1	1	2
x_2	2	0	1	2	1	1	1	0
x_3	1	1	0	1	2	2	0	1
x_4	0	2	1	0	1	1	1	2
x_5	1	1	2	1	0	0	2	1
x_6	1	1	2	1	0	0	2	1
x_7	1	1	0	1	2	2	0	1
x_8	2	0	1	2	1	1	1	0

For each x_i, we will calculate its $\hat{y}_{out}$ by taking the mode of its k - in this exercise, 5 - nearest neighbors, *i.e.*, vectors which are the smallest distance from it.

$$\hat{y}_{out}(x_1) = \text{mode}(y_{out}(\{x_3, x_4, x_5, x_6, x_7\}) = \text{mode}(\{P, P, N, N, N\}) = N$$

$$\hat{y}_{out}(x_2) = \text{mode}(y_{out}(\{x_3, x_5, x_6, x_7, x_8\}) = \text{mode}(\{P, N, N, N, N\}) = N$$

$$\hat{y}_{out}(x_3) = \text{mode}(y_{out}(\{x_1, x_2, x_4, x_7, x_8\}) = \text{mode}(\{P, P, P, N, N\}) = P$$

$$\hat{y}_{out}(x_4) = \text{mode}(y_{out}(\{x_1, x_3, x_5, x_6, x_7\}) = \text{mode}(\{P, P, N, N, N\}) = N$$

$$\hat{y}_{out}(x_5) = \text{mode}(y_{out}(\{x_1, x_2, x_4, x_6, x_8\}) = \text{mode}(\{P, P, P, N, N\}) = P$$

$$\hat{y}_{out}(x_6) = \text{mode}(y_{out}(\{x_1, x_2, x_4, x_5, x_8\}) = \text{mode}(\{P, P, P, N, N\}) = P$$

$$\hat{y}_{out}(x_7) = \text{mode}(y_{out}(\{x_1, x_2, x_3, x_4, x_8\}) = \text{mode}(\{P, P, P, P, N\}) = P$$

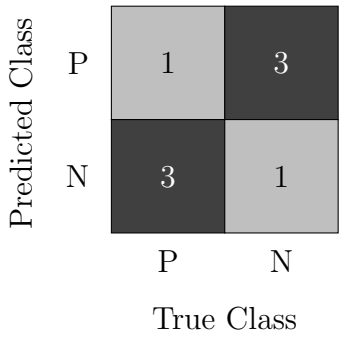
$$\hat{y}_{out}(x_8) = \text{mode}(y_{out}(\{x_2, x_3, x_5, x_6, x_7\}) = \text{mode}(\{P, P, N, N, N\}) = N$$

This yields us a set of predictions, $\{\hat{y}_{out}\}$, which we use to generate the confusion matrix for this kNN classification.

Table 3: True vs Predicted Labels

	y_{out}	$\hat{y}_{out}$
x_1	P	N
x_2	P	N
x_3	P	P
x_4	P	N
x_5	N	P
x_6	N	P
x_7	N	P
x_8	N	N

Table 4: Confusion Matrix



From these tables, we can calculate the Precision and Recall ratios:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{1}{1+3} = \frac{1}{4}$$
$$\text{Recall} = \frac{TP}{TP + FN} = \frac{1}{1+3} = \frac{1}{4}$$

And now we can calculate the F_1 score:

$$F_1 = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2}{4+4} = \frac{1}{4}$$

Question 2. Propose a new metric (distance) that improves the latter's performance (i.e., the F_1-measure) by three fold.

Upon doing some research, we attempted to learn the data using the Jaccard distance, and we found good results.

The Jaccard Distance is a metric derived from the Jaccard Similarity Coefficient. The similarity measured is calculated as the cardinality of the intersection of the elements in the 2 vectors over the cardinality of their union:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

The Jaccard Distance is complementary to it:

$$D_J(A, B) = 1 - J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

As an example of the application of the formula, let's calculate the distance between x_1 and x_5:

$$\begin{aligned} D_J(x_1, x_5) = D_J((A, 0), (B, 0)) = \\ 1 - \frac{|\{0\}|}{|\{A, B, 0\}|} = \\ 1 - \frac{1}{3} = \frac{2}{3} \approx 0.6667 \end{aligned}$$

The distance matrix we get using the Jaccard distance is present in Table 5. The confusion matrix allows for the fast calculation of the F_1 score, which yields precisely 3 times the value we obtained in the previous question.

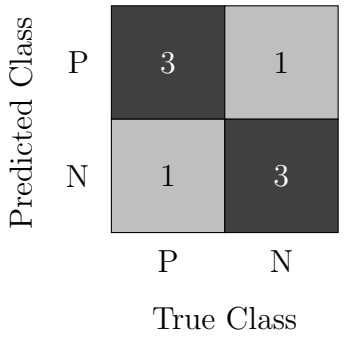
Table 5: Distance matrix calculated from the given data using the Jaccard Distance

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	0	1	0.67	0	0.67	0.67	0.67	1
x_2	1	0	0.67	1	0.67	0.67	0.67	0
x_3	0.67	0.67	0	0.67	1	1	0	0.67
x_4	0	1	0.67	0	0.67	0.67	0.67	1
x_5	0.67	0.67	1	0.67	0	0	1	0.67
x_6	0.67	0.67	1	0.67	0	0	1	0.67
x_7	0.67	0.67	0	0.67	1	1	0	0.67
x_8	1	0	0.67	1	0.67	0.67	0.67	0

Table 6: True vs Predicted Labels using Jaccard Distance

	y_{out}	$\hat{y}_{out}$
x_1	P	P
x_2	P	N
x_3	P	P
x_4	P	P
x_5	N	N
x_6	N	N
x_7	N	P
x_8	N	N

Table 7: Confusion Matrix resulting from the 2nd classification



$$\text{Precision} = \frac{TP}{TP + FP} = \frac{3}{3+1} = \frac{3}{4}$$
$$\text{Recall} = \frac{TP}{TP + FN} = \frac{3}{3+1} = \frac{3}{4}$$

And now we can calculate the F_1 score:

$$F_1 = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2}{\frac{8}{3}} = \frac{3}{4}$$

An additional positive observation was acquired, $x_9 = (B, 0)$, and a third variable was independently monitored, yielding estimates,

$$y_3|P = \{1.1, 0.8, 0.5, 0.9, 0.8\} \quad \text{and} \quad y_3|N = \{1, 0.9, 1.2, 0.9\}.$$

Question 3. Considering the nine training observations, learn a Bayesian classifier assuming: i) y_1 and y_2 are dependent; ii) y_1,y_2 and y_3 variable sets are independent and equally important; and iii) y_3 is normally distributed. Show all parameters.

Let's begin by looking at our updated data, in Table 8.

Table 8: Updated Data

	y_1	y_2	y_2	y_{out}
x_1	A	0	1.1	P
x_2	B	1	0.8	P
x_3	A	1	0.5	P
x_4	A	0	0.9	P
x_5	B	0	1	N
x_6	B	0	0.9	N
x_7	A	1	1.2	N
x_8	B	1	0.9	N
x_9	B	0	0.8	P

In order to learn a Bayesian Classifier, we must learn all of its parameters.
Bayes Theorem states:

$$P(c_i|x) = \frac{P(x|c_i)P(c_i)}{P(x)}$$

A Bayesian Classifier seeks to find the class which maximizes $P(c_i|x)$, *i.e.*, to return:

$$\text{argmax}\{P(c_i|x)\} = \text{argmax}\{\frac{P(x|c_i)P(c_i)}{P(x)}\}$$

And, therefore, to learn all parameters for this classifier, we need to learn $P(x|c_i)$ (likelihoods) for $\forall x$ and also all $P(c_i)$ (priors).

Calculating the priors is a simple matter of taking the ratio of the number of observations of that class by the total number of observations:

$$P(P) = \frac{5}{9} \approx 0.556$$
$$P(N) = \frac{4}{9} \approx 0.444$$

Calculating the likelihoods is done as follows.

$$P(x|c_i) = P(y_1, y_2, y_3|c_i) =$$

Given that $\{y_1, y_2\}$ and $\{y_3\}$ are independent from each other, we can further develop the expression:

$$= P(y_1, y_2|c_i) \cdot P(y_3|c_i)$$

The first term is, again, calculated by counting frequencies - more precisely, by calculating the number of observations of class c_i where each variable y_1 and y_2 assume a given pair of values, divided by the total number of observations in said class:

$$P(y_1 = a, y_2 = b|c_j) = \frac{\#\{x_i : y_1(x_i) = a \wedge y_2(x_i) = b \wedge y_{out} = c_j\}}{\#\{y_{out}(x_i) = c_j\}}$$

These were calculated and are organized in the Tables below.

Table 9: Likelihoods given class **P**

$y_{out} = P$

y_2 \ y_1	A	B
0	0.4	0.2
1	0.2	0.2

Table 10: Likelihoods given class **N**

$y_{out} = N$

y_2 \ y_1	A	B
0	0	0.5
1	0.25	0.25

In order to get $P(y_3|c_i)$, we need to model the distribution of y_3 by fitting it with a Gaussian curve.

Let's take the 2 given subsets of the y_3 vector and take the (sample) mean of each, as well as the (sample) standard deviation, to obtain the class-conditional maximum likelihood estimations for the parameters of the random variable's underlying distribution.

$$\mu_P = \frac{1.1 + 0.8 + 0.5 + 0.9 + 0.8}{5} = 0.82$$
$$\mu_N = \frac{1 + 0.9 + 1.2 + 0.9}{4} \approx 1.00$$
$$\sigma_P = 0.2168$$
$$\sigma_N = 0.1414$$

Learning the Evidence is quite a similar process.

Table 11: P(x), evidence, for each (y_1, y_2) value pair

y_2 \ y_1	A	B
0	2/9	1/3
1	2/9	2/9

$$\mu = \frac{1.1 + 0.8 + 0.5 + 0.9 + 1 + 0.9 + 1.2 + 0.9 + 0.8}{9} = 0.9$$

$$\sigma = 0.2$$

And, just like this, the Bayesian classifier is fully learnt. The priors, and the likelihoods were calculated and prepared to be plugged into the class probability formula.

Consider now three testing observations,

$$\{(A, 1, 0.8), (B, 1, 1), (B, 0, 0.9)\}$$

.

Question 4. Under a MAP assumption, classify each testing observation showing all your calculus.

The MAP assumption leads us to ignore the evidence, since it does not vary with variable i.

In this case, the classifier will be outputting the class provided by:

$$\text{argmax}\{P(x|c_i)P(c_i)\}$$

i) $(A, 1, 0.8)$

$$P(P|(A, 1, 0.8)) \propto P(y_1 = A, y_2 = 1|P) \cdot P(y_3 = 0.8|P) \cdot P(P) =$$
$$0.2 \cdot f_{y_3,P}(0.8) \cdot \frac{5}{9} =$$
$$0.2 \cdot 1.8324 \cdot 0.5556 = \boxed{0.2036}$$

$$P(N|(A, 1, 0.8)) \propto P(y_1 = A, y_2 = 1|N) \cdot P(y_3 = 0.8|N) \cdot P(N) =$$
$$0.25 \cdot f_{y_3,N}(0.8) \cdot \frac{4}{9} =$$
$$0.25 \cdot 1.0378 \cdot 0.4444 = \boxed{0.1153}$$

Therefore, a Bayesian Classifier under a MAP assumption would classify vector $(A, 1, 0.8)$ as **Positive**.

ii) $(B, 1, 1)$

$$P(P|(B, 1, 1)) \propto P(y_1 = B, y_2 = 1|P) \cdot P(y_3 = 1|P) \cdot P(P) =$$
$$0.2 \cdot f_{y_3,P}(1) \cdot \frac{5}{9} =$$
$$0.2 \cdot 1.3037 \cdot 0.5556 = \boxed{0.1449}$$

$$P(N|(B, 1, 1)) \propto P(y_1 = B, y_2 = 1|N) \cdot P(y_3 = 1|N) \cdot P(N) =$$
$$0.25 \cdot f_{y_3,N}(1) \cdot \frac{4}{9} =$$
$$0.25 \cdot 2.8209 \cdot 0.4444 = \boxed{0.3134}$$

Therefore, a Bayesian Classifier under a MAP assumption would classify vector $(B, 1, 1)$ as **Negative**.

iii) $(B, 0, 0.9)$

$$P(P|(B,0,0.9)) \propto P(y_1 = B, y_2 = 0|P) \cdot P(y_3 = 0.9|P) \cdot P(P) =$$
$$0.2 \cdot f_{y_3,P}(0.9) \cdot \frac{5}{9} =$$
$$0.2 \cdot 1.7191 \cdot 0.5556 = \boxed{0.1910}$$

$$P(N|(B,1,1)) \propto P(y_1 = B, y_2 = 1|N) \cdot P(y_3 = 1|N) \cdot P(N) =$$
$$0.5 \cdot f_{y_3,N}(1) \cdot \frac{4}{9} =$$
$$0.5 \cdot 2.1970 \cdot 0.4444 = \boxed{0.4882}$$

Therefore, a Bayesian Classifier under a MAP assumption would classify vector $(B, 0, 0.9)$ as **Negative**.

The final classifications based on a Bayes Classifier under a MAP assumption led us to classify the 3 given vectors with the class c_i that maximized $P(c_i|\text{vector})$ - respectively, **P**, **N** and **N**.

At last, consider only the following sentences and their respective connotations,

$$\{(\text{"Amazing run"}, P), (\text{"I like it"}, P), (\text{"Too tired"}, N), (\text{"Bad run"}, N)\}.$$

Question 5. Using a naïve Bayes under a ML assumption, classify the new sentence *"I like to run"*. For the likelihoods calculation consider the following formula,

$$p(t_i|c) = \frac{freq(t_i) + 1}{N_c + V}$$

,

where t_i represents a certain term i, V the number of unique terms in the vocabulary, and N_c the total number of terms in class c.

A Naïve Bayes model will assume that each term is independent from every other term, and, therefore, $P(\text{sentence}|c) = \prod_i p(t_i|c)$.

Bayes Theorem would tell us that we need to maximize $P(c|\text{sentence})$ such that it is equal to:

$$\frac{P(\text{sentence}|c)P(c)}{P(\text{sentence})}$$

The ML assumption further simplifies the matter at hand, by stating that all beliefs are equally likely, and as such:

$$P(c|\text{sentence}) \sim P(\text{sentence}|c) = \prod_i p(t_i|c)$$

We processed the data and placed the required counts for using the formula in Table 12

Table 12: Word frequency for each class

	P	N
"amazing"	1	0
"bad"	0	1
"I"	1	0
"it"	1	0
"like"	1	0
"run"	1	1
"tired"	0	1
"too"	0	1
N_c	5	4

For class P:

$$P(P|\text{sentence}) \sim \prod_i p(t_i|P) =$$
$$p(\text{"}I\text{"}|P) \cdot p(\text{"}like\text{"}|P) \cdot p(\text{"}to\text{"}|P) \cdot p(\text{"}run\text{"}|P) =$$
$$\frac{1+1}{5+8} \cdot \frac{1+1}{5+8} \cdot \frac{0+1}{5+8} \cdot \frac{1+1}{5+8} =$$
$$\frac{8}{13^4} = \boxed{2.801 \cdot 10^{-4}}$$

For class N:

$$P(N|\text{sentence}) \sim \prod_i p(t_i|N) =$$
$$p(\text{"}I\text{"}|N) \cdot p(\text{"}like\text{"}|N) \cdot p(\text{"}to\text{"}|N) \cdot p(\text{"}run\text{"}|N) =$$
$$\frac{0+1}{4+8} \cdot \frac{0+1}{4+8} \cdot \frac{0+1}{4+8} \cdot \frac{1+1}{4+8} =$$
$$\frac{2}{12^4} = \boxed{9.645 \cdot 10^{-5}}$$

As the probability of class P given the provided sentence is higher, that is how the Naïve Bayes model classifies it.

II Programming

Consider the `heart-disease.csv` dataset available at the course webpage's homework tab.

Using `sklearn`, apply a 5-fold stratified cross-validation with shuffling (`random_state=0`) for the assessment of predictive models along this section. This was done using the code

in Listing 1.

Question 1. Compare the performance of a k-NN with $k = 5$ and a naïve Bayes with Gaussian assumption (consider all remaining parameters as default).

a) Plot two boxplots with the fold accuracies for each classifier. Is there one more stable than the other regarding performance? Why do you think that is the case? Explain.

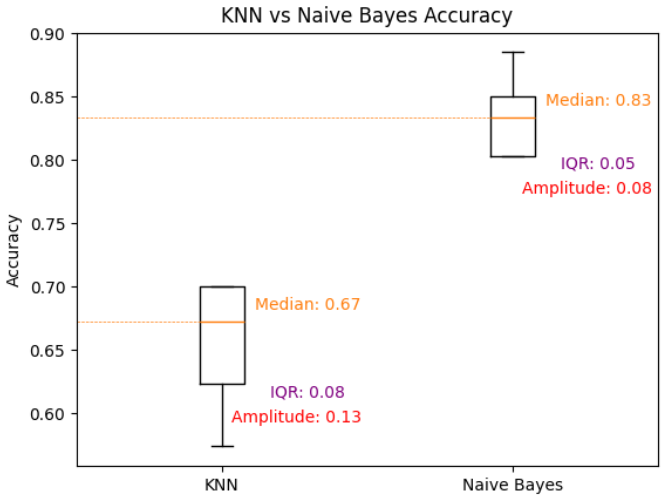


Figure 1: Boxplots displaying 5-fold cross validation accuracy for the kNN and the Naïve Bayes classifiers

kNN and Naïve Bayes classifiers were fitted to the train data and scored on the test fold, for each of the 5 folds. Their accuracy was plotted in Figure 1. Check Listing 3 for code.

The smaller IQR and amplitude for Naïve Bayes' accuracies suggest a more consistent performance across folds compared to k-NN. The smaller variation indicated by the low IQR is synonymous with bigger consistency.

This is likely the case because, while the Naïve Bayes Classifier learns global parameters from the whole train set, and then applies them to the test set, a k NN classifier classifies each sample based on a local view of the train set.

The k NN will largely be affected by local noise, while the Naïves Bayes global approach should minimize fluctuations. Although the K Fold cross validation is stratified and randomized, the train set is different at every fold - which will cause major fluctuations locally.

b) Report the accuracy of both models, this time scaling the data with a Min-Max scaler before training the models. Explain the impact that this preprocessing step has on the performance of each model, providing an explanation for the results.

A MinMax Scaler was fit on the train data. We then transformed both the train data and the test data using it. Relevant code found in Listing 3.

MinMax Scaling had a large impact on the performance of the kNN model, but no visible effect on the performance of the Naïve Bayes model, as we can see from Figure 2.

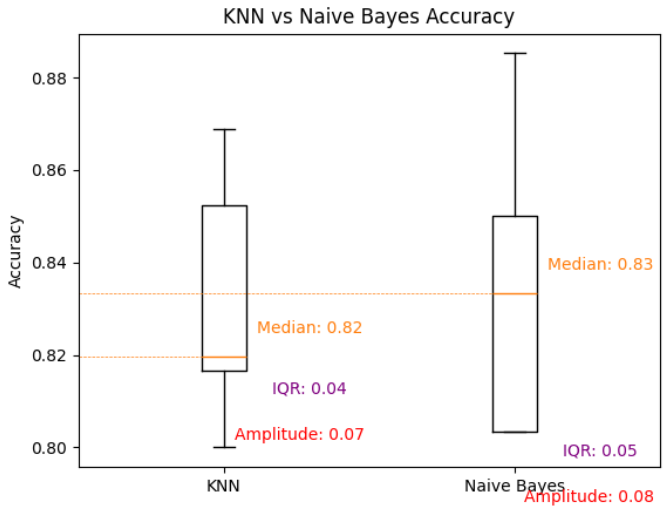


Figure 2: Boxplots displaying 5-fold cross validation accuracy for the kNN and the Naïve Bayes classifiers, applied on previously MinMax Scaled data

Naive Bayes Accuracies:

- {0.8852, 0.8033, 0.8033, 0.8500, 0.8333}
- **Mean:** 0.8350

KNN Accuracies:

- {0.8033, 0.8525, 0.8852, 0.8000, 0.8667}
- **Mean:** 0.8415

MinMax Scaling adjusts each feature in the dataset so that its values fall within the range of 0 to 1. This is done by taking each value and placing it relative to the minimum and maximum values of that feature. Since the features in the heart-disease.csv dataset have different units and ranges, scaling them this way ensures that no single feature dominates the others just because it has larger numbers. This process is important because it allows all features to contribute equally to the model's predictions. Without scaling, some features might unfairly influence the results due to their larger or smaller values.

Since the kNN model relies on distance calculation (such as Euclidean distance), MinMax Scaling heavily affects its output. On the other hand, the Naïve Bayes under a Gaussian assumption models the data's probability by fitting it with normal distributions. Since the mean and standard deviation are calculated parameters, the variation of the data is already prioritised in this processing step. Besides, the output, a probability, already comes normalised - probabilities are confined to the [0,1] range.

When scaling the data, although it might alter the values for standard deviation and the mean that are calculated, it doesn't change the output of the Gaussian curves.

c) Using `scipy`, test the hypothesis "the k-NN model is statistically superior to Naïve Bayes regarding accuracy," asserting whether it is true.

The hypothesis should be accepted if the probability of it being wrong (= H_0 being true) is below a threshold. Let's say that threshold is the usual value of 5% and so let's test for $p < 0.05$.

If we let:

$$X \sim \text{R.V. which represents the accuracy of the kNN classifier}$$
$$Y \sim \text{R.V. which represents the accuracy of the NB classifier}$$

Then the set of accuracies we got is a sample of X and Y, respectively for each classifier. We define our hypotheses below:

$$H_0 : \overline{X} - \overline{Y} \leq 0 \ (\iff \overline{\text{accuracy}}_{kNN} \leq \overline{\text{accuracy}}_{NB})$$
$$H_1 : \overline{X} - \overline{Y} 0 \ (\iff \overline{\text{accuracy}}_{kNN} \overline{\text{accuracy}}_{NB})$$

By running the code in Listing 4, we get the following ($\alpha = 0.5$):

Table 13: Hypothesis Test Results: KNN vs. Naive Bayes with MinMax Scaling

Null Hypothesis (H_0)	KNN Classifier is not better than Naive Bayes Classifier when using MinMax Scaling
T-statistic	-0.2077
P-value	0.5772
Degrees of Freedom	4
Conclusion	We cannot reject the null hypothesis, thus we cannot conclude that the KNN model is better than the Naive Bayes model when using MinMax Scaling.

Table 14: Hypothesis Test Results: KNN vs. Naive Bayes without MinMax Scaling

Null Hypothesis (H_0)	KNN Classifier is not better than Naive Bayes Classifier without using MinMax Scaling
T-statistic	-6.6903
P-value	0.9987
Degrees of Freedom	4
Conclusion	We cannot reject the null hypothesis, thus we cannot conclude that the KNN model is better than the Naive Bayes model without using MinMax Scaling.

Either scenario leads us to reject the hypothesis stated in the question (and accept H_0). This means that, statistically, Naïve Bayes is either equal or statistically superior to the k-NN model.

Question 2. Using a 80-20 train-test split, vary the number of neighbors of a kNN classifier using $k = \{1, 5, 10, 20, 30\}$. Additionally, for each k, train one classifier using uniform weights and distance weights.

a) Plot the train and test accuracy for each model.

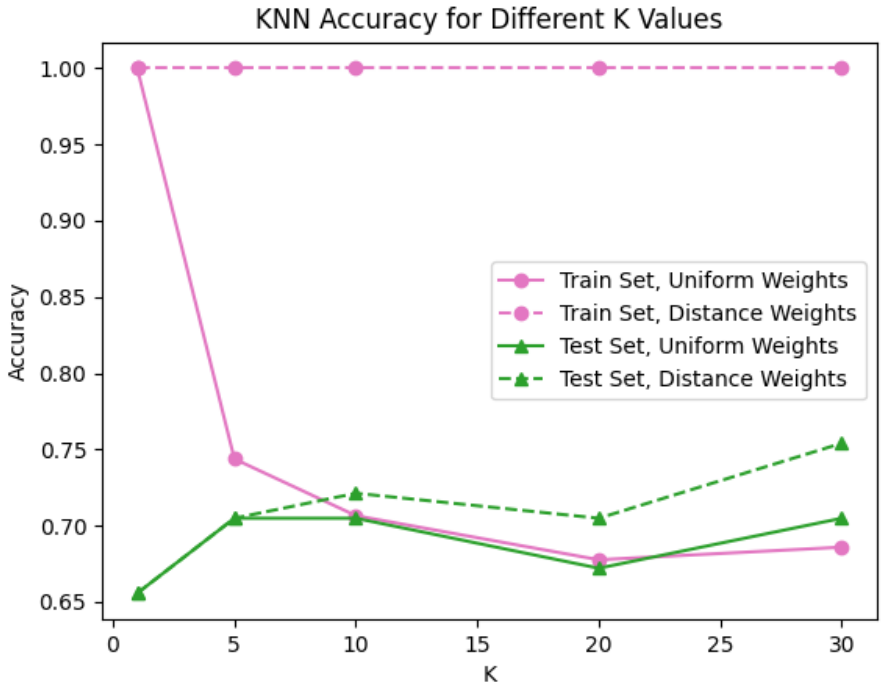


Figure 3: Variation of model accuracy with number of neighbors k, grouped by calculation weights.

a) Explain the impact of increasing the neighbors on the generalization ability of the models.

We fit and scored kNN models for each of the given ks, and both uniform and distance weights, on the train set and test set of the train-test split. This split was done using the code in Listing 5

For the models with uniform weights, the train accuracy starts out very high and then decays down to a plateau around the 67% mark. The test accuracy, on the other hand, starts out at around 60% and goes up to around 70%, at which point it also seems to plateau. If we experiment with higher k values to test this suspicion, we find that the accuracy decreases - check Figure 4.

As k increases, the model tends to generalize better. This is evident from the decrease in training accuracy alongside the increase in test accuracy, indicating improved performance on unseen data. However, after a certain point (around $k = 25$), this improvement plateaus, suggesting diminishing returns. Therefore, we can infer that there is an optimal value, or

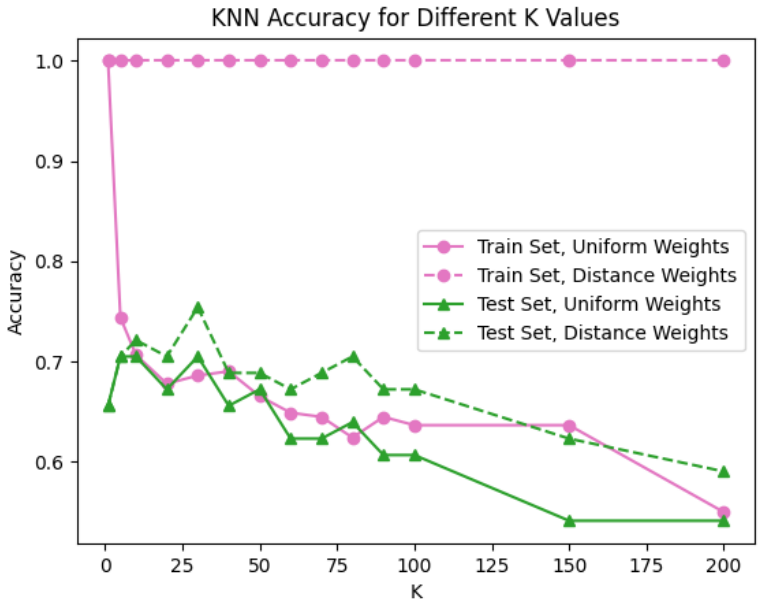


Figure 4: Variation of model accuracy with number of neighbors k, grouped by calculation weights - now with more data points.

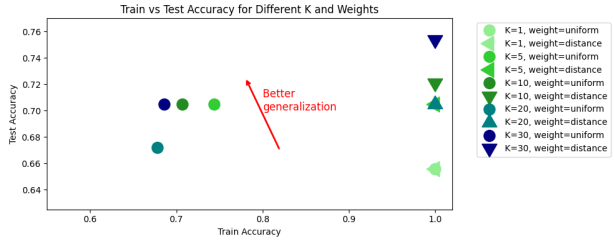


Figure 5: Each models's test accuracy plotted vs its train accuracy.

"sweet spot," for k, where the model achieves the best balance between overfitting and generalization.

One can clearly see this train/test accuracy trade off in Figure 5.

On the other hand, on models with distance weights, performance on the train dataset is stable at 100% regardless of the k value. Performance on the test set seems to increase with k, although one can likely also expect to see a plateau with a higher k, given the small variability of the rightmost points' test accuracy.

The 100% accuracy on the train set is a result of self inclusion and, since the distance of a vector to itself is 0, its calculated weight is $\frac{1}{0}$, which will be computed approximately to a very large value, overpowering all other data points.

By increasing the number of neighbors, we effectively expand the voting pool, which helps reduce random noise and makes the underlying patterns in the data more apparent. A larger voting pool enhances the reliability of the classification by focusing on broader trends. However, if k becomes too large, the model begins to underfit, as it loses the core advantage

of the algorithm: leveraging the neighborhood of a given point. With an excessively large neighborhood, the algorithm includes data points that may not be closely related to the point being classified, leading to less accurate predictions.

Question 3. Considering the unique properties of the `heart-disease.csv` dataset, identify two possible difficulties of the naïve Bayes model used in the previous exercises when learning from the given dataset.

The `heart-disease.csv` dataset's information page can be found on Kaggle via a quick search.

1 We can first consider the way that `GaussianNB()` models data. By fitting Gaussians to all variables, it disregards the possibility that their distributions may be different. Categorical variables are not well modeled by Normal distributions here, and are assumed to be continuous variables.

   Besides, even the continuous variables may not have an underlying Gaussian distribution: if we refer to the dataset's Data Card, we can see a chart like the one in Figure 6, where the variable `oldpeak` doesn't quite resemble the normal distribution (in green). We also plotted other distributions for comparison.

2 The Naïve Bayes model assumes all variables are independent. But this is not always the case. Rather, the real meaning of the dataset would intuitively tell us that this is definitely not the case. For example, `chol` (cholesterol) and `age` are most likely not independent.

   If we take χ^2 independence tests - see Figure 7, we can further confirm this intuition—the p-value for the test done between these two variables is so close to 1 that we can almost be certain they are dependent.

III Appendix

```python
def generate_splits(df, N_SPLITS = 5):
    strat_kfold = StratifiedKFold(n_splits=N_SPLITS, shuffle=True,
    random_state=0)
    split = strat_kfold.split(np.zeros(len(df)), df['target'])
    return split
```

Listing 1: Function to generate `N_SPLITS` splits

```python
models_accuracies = []

for k in ks:
    for weight in ['uniform', 'distance']:
        knn = KNeighborsClassifier(n_neighbors=k, weights=weight)
        knn.fit(X_train, y_train)
        train_acc = knn.score(X_train, y_train)
        test_acc = knn.score(X_test, y_test)
```

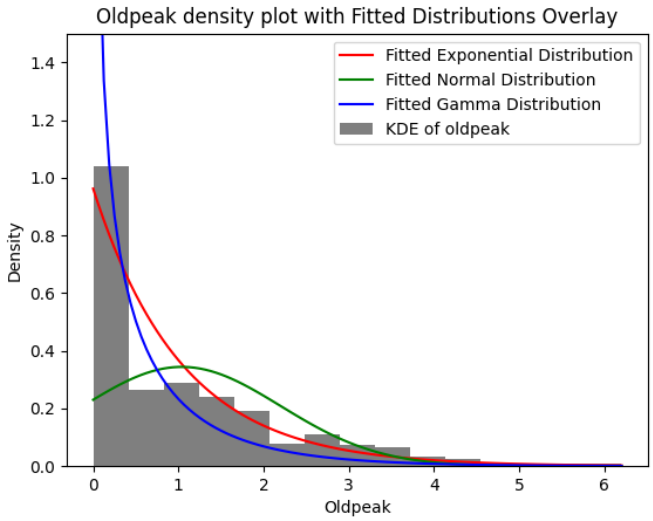


Figure 6: `oldpeak` feature density plot with Fitted Distributions Overlay

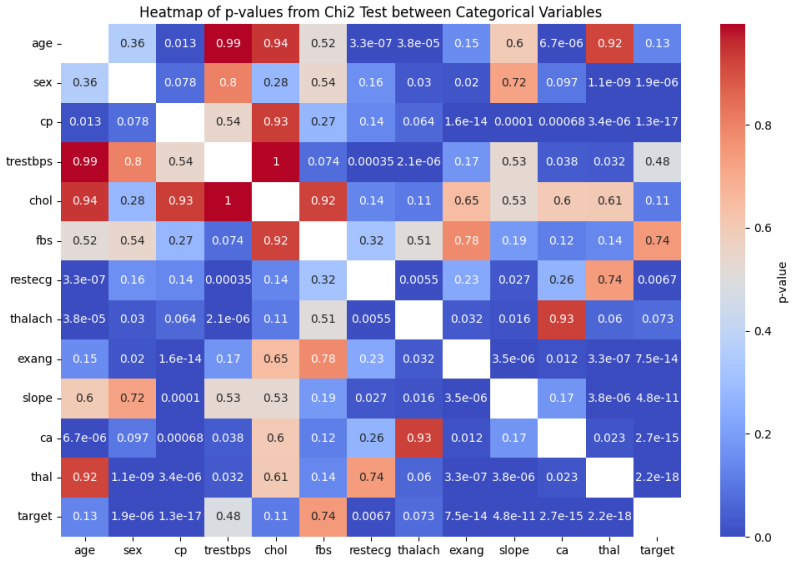


Figure 7: Heatmap of p-values from χ^2 Test between Categorical Variables

```
        models_accuracies.append({'k': k, 'weight': weight, 'train_acc':
    train_acc, 'test_acc': test_acc})

models_accuracies = pd.DataFrame(models_accuracies)
models_accuracies
```

Listing 2: Code that fits and scores k NNmodels on train-test data for each combination of k and weights

```
# Run models and store accuracies

N_NEIGHBORS = 5
knn = KNeighborsClassifier(n_neighbors=5)
nbayes = GaussianNB()

def fit_and_score(model, train, test):
    model = model.fit(train.drop('target', axis=1, inplace=False), train['
    target'])
    return model.score(test.drop('target', axis=1, inplace=False), test['
    target'])


def knn_vs_nbayes(df, scale=False):
    knn_accuracies = []
    nbayes_accuracies = []
    for train_index, test_index in generate_splits(df):
        train = df.loc[train_index]
        test = df.loc[test_index]
        if scale:
            MINMAXSCALER = MinMaxScaler()
            MINMAXSCALER.fit_transform(train[['age','trestbps','chol','
    thalach','oldpeak']])

            train[['age','trestbps','chol','thalach','oldpeak']] =
    MINMAXSCALER.transform(train[['age','trestbps','chol','thalach','
    oldpeak']].copy())
            test[['age','trestbps','chol','thalach','oldpeak']] =
    MINMAXSCALER.transform(test[['age','trestbps','chol','thalach','oldpeak
    ']].copy())            #test['target'] = target

        knn_accuracies.append(fit_and_score(knn, train, test))
        nbayes_accuracies.append(fit_and_score(nbayes, train, test))
    return knn_accuracies, nbayes_accuracies

knn_accuracies, nbayes_accuracies = knn_vs_nbayes(data)
```

Listing 3: Code that fits and scores kNN and scores using 5-fold cross validation

```
# SCALED DATA ###################################
print(colored('H0: KNN Classifier is not better than Naive Bayes
    Classifier when using MinMax Scaling', 'blue'))
test_result = stats.ttest_rel(knn_accuracies_scaled,
    nbayes_accuracies_scaled, alternative='greater')     # [alternative=]
        greater    : the mean of the distribution underlying the first sample
```

```
    is greater than the mean of the distribution underlying the second
    sample.

print('T-statistic: ', test_result.statistic)
print('P-value: ', test_result.pvalue)
print('Degrees of Freedom: ', test_result.df)

if test_result.pvalue < 0.05:
    print(colored('We can reject the null hypothesis, and conclude that
    the KNN Classifier is better than the Naive Bayes Classifier, when
    using MinMax Scaling', "green"))
else:
    print(colored('We cannot reject the null hypothesis, thus we cannot
    conclude that the KNN model is better than the Naive Bayes model, when
    using MinMax Scaling', "red"))
```

Listing 4: Hypothesis testing for comparing the performance of kNN and Naïve Bayes

```
X_train, X_test, y_train, y_test = train_test_split(data.drop('target',
    axis=1, inplace=False), data['target'], test_size=0.20, random_state=0,
    stratify=data['target'])
```

Listing 5: Train test split