CS245 Homework 3

July 31, 2024

Name: Kelly Furuya

1 Problem 1: Naive Bayes and Bayesian Networks

1.1 Implement the Naive Bayes method to compute the probability that an individual does not have a dog given that they are educated, travels for work regularly, and has low levels of physical activity

Table 1 shows the probabilities of owning a dog given each individual condition based on the assumption that they are all independent of each other.

Condition	**Probability**
P(Education = "Educated" | Dog = "No")	2/4 = 0.5
P(Work Travel = "Regular" | Dog = "No")	2/4 = 0.5
P(Physical Activity = "Low" | Dog = "No")	0/4 = 0

Table 1: Individual probabilities for each condition

If we multiply each of the conditional probabilities together, we get 0.5 * 0.5 * 0 = 0. Since there are no instances of a person with "Low" physical activity, the probability of any given person with that attribute owning a dog is 0.

1.2 Assume that (1) work travel and physical activity are depedent on education and (2) having a dog is dependent on work travel and physical activity but not directly on education. Construct a Bayesian network for this dataset, specifying the Conditional Probability Table (CPT) for each node. Based on your network, compute the probability that an individual does not have a dog given that they are uneducated, never travels for work, and has low levels of exercise.

Figure 1 below is a directed acyclic graph that shows the dependencies between each status along with the conditional probabilities for each node based on the result of the previous one.

Education

Education = "Educated"	Education = "Uneducated"
5/10 = 0.5	5/10 = 0.5

Work Travel

Education	Work Travel = "Regular"	Work Travel = "Non-Regular"	Work Travel = "Never"
"Educated"	2/5 = 0.4	1/5 = 0.2	2/5 = 0.4
"Uneducated"	2/5 = 0.4	2/5 = 0.4	1/5 = 0.2

Physical Activity

Education	Physical Activity = "High"	Physical Activity = "Low
"Educated"	2/5 = 0.4	3/5 = 0.6
"Uneducated"	3/5 = 0.6	2/5 = 0.4

Dog

Work Travel, Physical Activity	Dog = "Yes"	Dog = "No"
Work Travel = "Regular", Physical Activity = "High"	1/3 = 0.333...	2/3 = 0.666...
Work Travel = "Regular", Physical Activity = "Low"	1/1 = 1	0
Work Travel = "Non-Regular", Physical Activity = "High"	0	1/1 = 1
Work Travel = "Non-Regular", Physical Activity = "Low"	2/2 = 1	0
Work Travel = "Never", Physical Activity = "High"	0	1/1 = 1
Work Travel = "Never", Physical Activity = "Low"	2/2 = 1	0

Figure 1: Conditional Probability Table for each node

We can then take the probabilities from these tables to calculate the probability that a person does not have a dog given that they are uneducated, never travels for work, and has low levels of physical activity, P(Dog = "Yes" | Educated = "Uneducated", Work Travel = "Never", Physical

Activity = "Low"), by summing up all probabilities where a person meets the mentioned conditions and divide that by the sum of all the probabilities where a person meets all the conditions while ignoring dog ownership.

$$\frac{\sum_{D,E,T,P} P(Dog = \text{“}Yes\text{”}, Educated = \text{“}Uneducated\text{”}, WorkTravel = \text{“}Never\text{”}, PhysicalActivity = \text{“}Low\text{”})}{\sum_{E,T,P} P(Educated = \text{“}Uneducated\text{”}, WorkTravel = \text{“}Never\text{”}, PhysicalActivity = \text{“}Low\text{”})}$$

$$= \frac{0.5 * 0.2 * 0.4 * 0}{(0.5 * 0.2 * 0.4 * 1) + (0.5 * 0.2 * 0.4 * 0)}$$

$$= 0$$

1.3 Compare the results obtained from Naive Bayes and Bayesian Networks. Discuss the assumptions made by each method and why it may or may not be reasonable for this dataset.

Both methods obtained 0 for the probability of not owning a dog for the given conditions. One assumption that both methods assume is that the dataset is comprehensive and representative of the broader population. This can cause problems as it can be difficult to effectively obtain enough data to truly represent the actual population. A general method around this is to add some number, usually 1, to all combinations of conditions unless it is physically impossible for a specific situation to occur. In both of our cases, this would have helped prevent a 0 probability.

Naive Bayes makes the assumption that all variables are independent of each other which rarely happens in actual datasets. For this dataset, this can cause problems as there seems to be a strong correlation between Physical Activity and whether or not the person owns a dog. Without taking into account certain dependencies, Naive Bayes can miss additional information that dependencies can probide. However, it could still be a reasonable method for this dataset as we generally will not know what variables are dependent on another. Since Naive Bayes is a simplified version of Bayesian Networks, being able to work without having to know (or have a good guess) of any of the dependencies is still useful and often reasonable and effective.

Bayesian Networks makes the assumption that the dependencies specified are likely to be true. If these dependencies are not correct or are only partially correct, this can lead the algorithm to give more weight to certain conditions over others, resulting in skewed probabilities. For example, with the data in this problem, there does not appear to be a strong correlation between any of the conditions except Physical Activity and dog ownership. The CPT in Figure 1 shows that all of the probabilities for Work Travel and Physical Activity are fairly even regardless of Education status. We can also see from the same CPT that whether or not a person owns a dog seems to be more related to their Physical Activity than their Work Travel. Although we do not necessarily end up with wildly incorrect probabilities from these assumptions, it does cause potentially unneccesary calculations.

1.4 Modify the Bayesian network by adding a new variable "Income" with levels "High" and "Low". Discuss how this change might influence the choice to buy a dog.

If we add Income to the network without any information about potential dependencies, the network would look like Figure 2. Income is separate with no arrows pointing to or from it.

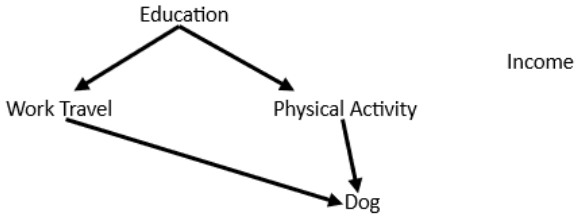


Figure 2: Bayesian Network with Income without dependencies

Some general assumptions we could make about the relation between Income and dog ownership is that those with higher income might be more likely to own a dog as they will have more disposable income to afford a pet whereas those with a low income might not be able to. Very generally speaking, people with an education tend to have a higher income than those without an education, so Income will likely be influenced by Education in the Bayesian Network. If we view these as the only new restrictions to add, the new network without the probability tables would look like Figure 3.

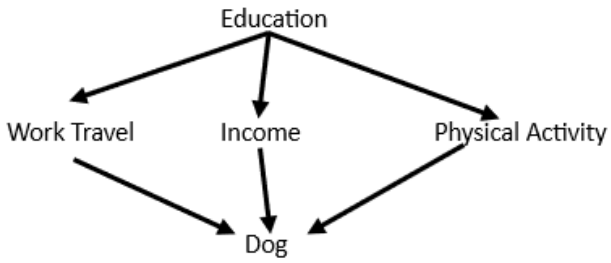


Figure 3: New Bayesian Network with Income with dependencies and no CPTs

We could then from here generate the CPTs using arbitrary assignments to the Income variable for each individual in the original data. The main difference would be that the CPT for Dog would contain twice as many entries as the original one since Income has two options. There could be other versions of the network that we could also generate if we determine that Income is dependent on or directly influences other variables based on the data, but in this instance, we will stick with assuming that only Education and Dog ownership are relevant.

2 Problem 2: Decision Trees

2.1 Construct a decision tree using the ID3 algorithm. Show your work in computing information gain for each attribute at each decision point.

The table provided has 8 entries broken down into four different variables, Forecast, Time of Showing, Traffic, and Buys Tickets. We want to know whether or not a person is going to buy a ticket, so we know that must be the final result in the decision tree.

For Forecast, there are three categories, Sunny, Overcast, and Rainy. Table 2 shows the breakdown of each weather condition and how many entries did or did not buy tickets.

Forecast	**Buys**	**Did Not Buy**
Sunny	1	2
Overcast	2	0
Rainy	1	2

Table 2: Forecast breakdown on how many people did or did not buy tickets

We then repeat this for the other two variables to determine their breakdowns as shown in Table 3.

Time of Showing	**Buys**	**Did Not Buy**		**Traffic**	**Buys**	**Did Not Buy**
Day	1	3		True	2	2
Night	3	1		False	2	2

Table 3: Time of showing and Traffic breakdown on how many people did or did not buy tickets

From the table values, we can now calculate the information gain for each variable. First, we need to calculate the Information for whether or not a ticket was bought for all eight entries.

$$\begin{aligned} I_{Bought} &= -\frac{4}{8} * log_2(\frac{4}{8}) - \frac{4}{8} * log_2(\frac{4}{8}) \\ &= -0.5 * -1 - 0.5 * -1 \\ &= 1 \end{aligned}$$

Next, we calculate the information for each category in Forecast, rounded to the fourth decimal place.

$$\begin{aligned} I_{Sunny} &= -\frac{1}{3} * log_2(\frac{1}{3}) - \frac{2}{3} * log_2(\frac{2}{3}) \\ &= -\frac{1}{3} * -1.5850 - \frac{2}{3} * -0.5850 \\ &= 0.5283 + 0.38998 \\ &= 0.9183 \end{aligned}$$

$$\begin{aligned} I_{Overcast} &= -\frac{2}{2} * log_2(\frac{2}{2}) - \frac{0}{2} * log_2(\frac{0}{2}) \\ &= 0 \end{aligned}$$

$$\begin{aligned} I_{Rainy} &= -\frac{1}{3} * log_2(\frac{1}{3}) - \frac{2}{3} * log_2(\frac{2}{3}) \\ &= -\frac{1}{3} * -1.5850 - \frac{2}{3} * -0.5850 \\ &= 0.5283 + 0.38998 \\ &= 0.9183 \end{aligned}$$

Next, we calculate the information for each category in Time of Showing.

$$I_{Day} = -\frac{1}{4} * log_2(\frac{1}{4}) - \frac{3}{4} * log_2(\frac{3}{4})$$
$$= -\frac{1}{4} * -2 - \frac{3}{4} * -0.4150$$
$$= 0.5 + 0.3113$$
$$= 0.8113$$

$$I_{Night} = -\frac{3}{4} * log_2(\frac{3}{4}) - \frac{1}{4} * log_2(\frac{1}{4})$$
$$= -\frac{3}{4} * -0.4150 - \frac{1}{4} * -2$$
$$= 0.3113 = 0.5$$
$$= 0.8113$$

Finally, we calculate the information for each category in Traffic.

$$I_{True} = -\frac{2}{4} * log_2(\frac{2}{4}) - \frac{2}{4} * log_2(\frac{2}{4})$$
$$= -\frac{2}{4} * -1 - \frac{2}{4} * -1$$
$$= 1$$

$$I_{False} = -\frac{2}{4} * log_2(\frac{2}{4}) - \frac{2}{4} * log_2(\frac{2}{4})$$
$$= -\frac{2}{4} * -1 - \frac{2}{4} * -1$$
$$= 1$$

Now, we can calculate the entropy for each variable.

$$E_{Forecast} = \frac{3}{8} * (I_{Sunny}) + \frac{2}{8} * (I_{Overcast}) + \frac{3}{8} * (I_{Rainy})$$
$$= \frac{3}{8} * (0.9183) + \frac{2}{8} * (0) + \frac{3}{8} * (0.9183)$$
$$= 0.6887$$

$$E_{TimeofShowing} = \frac{4}{8} * (I_{Day}) + \frac{4}{8} * (I_{Night})$$
$$= \frac{4}{8} * (0.8113) + \frac{4}{8} * (0.8113)$$
$$= 0.8113$$

$$E_{Traffic} = \frac{4}{8} * (I_{True}) + \frac{4}{8} * (I_{False})$$

$$= \frac{4}{8} * (1) + \frac{4}{8} * (1)$$
$$= 1$$

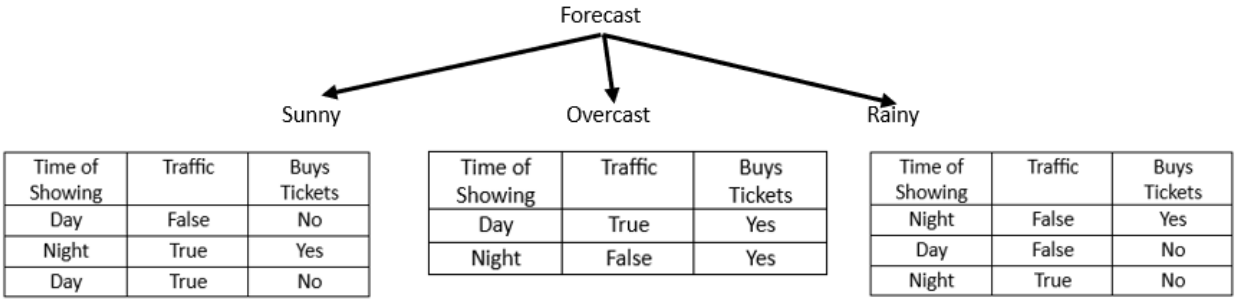
Laslty, we calculate the Information Gain for each variable.

$$Gain_{Forecast} = I_{Bought} - E_{Forecast}$$
$$= 1 - 0.6887$$
$$= 0.3113$$

$$Gain_{TimeofShowing} = I_{Bought} - E_{TimeofShowing}$$
$$= 1 - 0.8113$$
$$= 0.1887$$

$$Gain_{Traffic} = I_{Bought} - E_{Traffic}$$
$$= 1 - 1$$
$$= 0$$

Based off of the Information Gain, the first split of the tree should be based on Forecast. Figure 4 shows this split and the resutling partitioned entries.



Time of Showing	Traffic	Buys Tickets
Day	False	No
Night	True	Yes
Day	True	No

Time of Showing	Traffic	Buys Tickets
Day	True	Yes
Night	False	Yes

Time of Showing	Traffic	Buys Tickets
Night	False	Yes
Day	False	No
Night	True	No

Figure 4: Decision Tree with initial split based on the Forecast.

We can see that for both instances of Overcast, tickets were purchased regardless of the other variables. This means that we can simplify the tree in the next iteration to simply be "Yes" if the Forecast is Overcast.

The next step is to determine the Information Gain based on the new partitioned entries to determine the best split for each branch. We will start with the Information of the Sunny branch.

$$I_{Bought_{Sunny}} = -\frac{1}{3} * log_2(\frac{1}{3}) - \frac{2}{3} * log_2(\frac{2}{3})$$
$$= 0.9183$$

$$I_{Day} = -\frac{0}{2} * log_2(\frac{0}{2}) - \frac{2}{2} * log_2(\frac{2}{2})$$
$$= 0$$

$$I_{Night} = -\frac{1}{1} * log_2(\frac{1}{1}) - \frac{0}{1} * log_2(\frac{0}{1})$$
$$= 0$$

$$I_{True} = -\frac{1}{2} * log_2(\frac{1}{2}) - \frac{1}{2} * log_2(\frac{1}{2})$$
$$= 1$$

$$I_{False} = -\frac{0}{1} * log_2(\frac{0}{1}) - \frac{1}{1} * log_2(\frac{1}{1}$$
$$= 0$$

Now, we calculate Entropy.

$$E_{TimeofShowing} = \frac{2}{3} * I_{Day} + \frac{1}{3} * I_{Night}$$
$$= 0$$

$$E_{Traffic} = \frac{2}{3} * I_{True} + \frac{1}{3} * I_{False}$$
$$= 0.6667$$

Then we calculate Information Gain.

$$Gain_{TimeofShowing} = I_{Bought_{Sunny}} - E_{TimeofShowing}$$
$$= 0.9183$$

$$Gain_{Traffic} = I_{Bought_{Sunny}} - E_{Traffic}$$
$$= 0.2516$$

Time of Showing has the largest Information Gain, so it will be the next split for the Sunny branch. We can also clearly see that all instances of the Day showing had no tickets purchased while the only instance of Night had tickets purchased, regardless of Traffic. This lines up with the calculations.

Next, we will examine the Rainy branch.

$$I_{Bought_{Rainy}} = -\frac{1}{3} * log_2(\frac{1}{3}) - \frac{2}{3} * log_2(\frac{2}{3})$$
$$= 0.9183$$

$$I_{Day} = -\frac{0}{1} * log_2(\frac{0}{1}) - \frac{1}{1} * log_2(\frac{1}{1})$$
$$= 0$$

$$I_{Night} = -\frac{1}{2} * log_2(\frac{1}{2}) - \frac{1}{2} * log_2(\frac{1}{2})$$
$$= 1$$

$$I_{True} = -\frac{0}{1} * log_2(\frac{0}{1}) - \frac{1}{1} * log_2(\frac{1}{1})$$
$$= 0$$

$$I_{False} = -\frac{1}{2} * log_2(\frac{1}{2}) - \frac{1}{2} * log_2(\frac{1}{2}$$
$$= 1$$

$$E_{TimeofShowing} = \frac{1}{3} * I_{Day} + \frac{2}{3} * I_{Night}$$
$$= 0.6667$$

$$E_{Traffic} = \frac{1}{3} * I_{True} + \frac{2}{3} * I_{False}$$
$$= 0.6667$$

$$Gain_{TimeofShowing} = I_{Bought_{Rainy}} - E_{TimeofShowing}$$
$$= 0.2516$$

$$Gain_{Traffic} = I_{Bought_{Rainy}} - E_{Traffic}$$
$$= 0.2516$$

We can see that we get the same information gain for both variables which indicates that neither are better to split along. We could split the decision tree along either one and end up with the same "impurities" in either version of the tree. For this exercise, we will arbitrarily select Time of Showing as the variable to split initially for this branch as that is the first variable the algorithm encounters. Figure 5 shows the final decision tree.

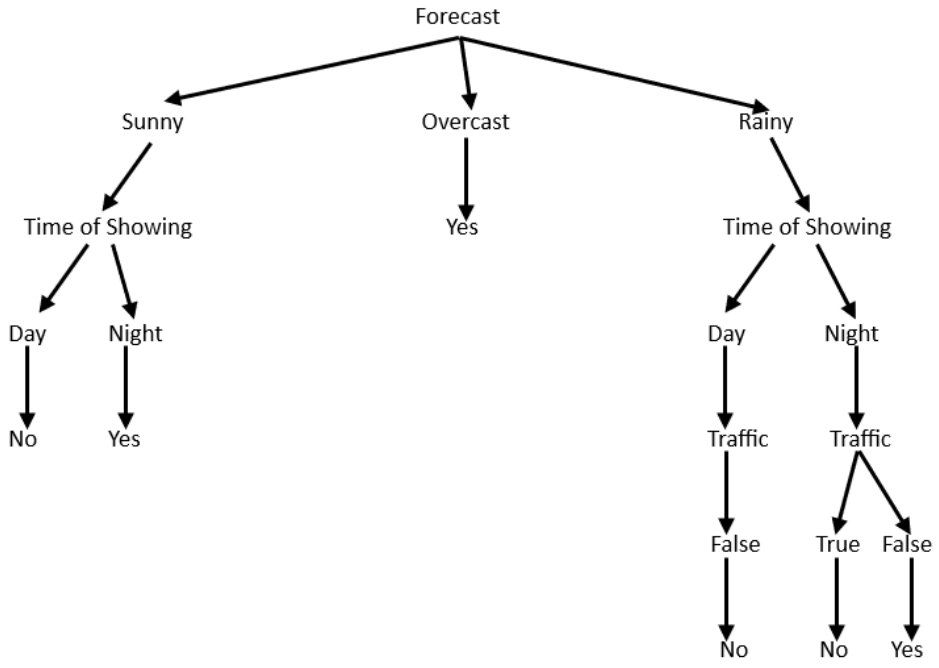


Figure 5: Final Decision Tree split based on Information Gain

2.2 Based on your decision tree, what conditions will most likely result in "Yes" for buying movie tickets?

The conditions most likely to result in "Yes" for buying movie tickets are Overcast weather and a Night showing time, independent of each other.

2.3 How does changing the decision rule (e.g., from information gain to Gini index) impact the structure of the decision tree?

We will examine how using the Gini Index might result in a different decision tree. First, we will calculate the Gini index for Forecast.

$$Gini_{Sunny} = 1 - (\frac{1}{3})^2 - (\frac{2}{3})^2$$
$$= 0.4444$$

$$Gini_{Overcast} = 1 - (\frac{2}{2})^2 - (\frac{0}{2})^2$$
$$= 0$$

$$Gini_{Rainy} = 1 - (\frac{1}{3})^2 - (\frac{2}{3})^2$$
$$= 0.4444$$

$$Gini_{Forecast} = \frac{3}{8}(Gini_{Sunny}) + \frac{2}{8}(Gini_{Overcast}) + \frac{3}{8}(Gini_{Rainy})$$

$$= 0.3333$$

Next, we will calculate the Gini index for Time of Showing.

$$Gini_{Day} = 1 - (\frac{1}{4})^2 - (\frac{3}{4})^2$$
$$= 0.375$$

$$Gini_{Night} = 1 - (\frac{3}{4})^2 - (\frac{1}{4})^2$$
$$= 0.375$$

$$Gini_{TimeofShowing} = \frac{4}{8}(Gini_{Day}) + \frac{4}{8}(Gini_{Night})$$
$$= 0.375$$

Lastly, we will look at Traffic.

$$Gini_{True} = 1 - (\frac{2}{4})^2 - (\frac{2}{4})^2$$
$$= 0.5$$

$$Gini_{False} = 1 - (\frac{2}{4})^2 - (\frac{2}{4})^2$$
$$= 0.5$$

$$Gini_{TimeofShowing} = \frac{4}{8}(Gini_{True}) + \frac{4}{8}(Gini_{False})$$
$$= 0.5$$

For the Gini Index, we are looking for the smallest number as it indicates the level of impurity. In this case, we see that Forecast is still the best variable to make the first split. This will give us the same tree as in Figure 4. As before, the next step is to repeat the calculations for each of partitions to determine the next best splits for each branch, starting with Sunny.

$$Gini_{Day} = 1 - (\frac{0}{2})^2 - (\frac{2}{2})^2$$
$$= 0$$

$$Gini_{Night} = 1 - (\frac{1}{1})^2 - (\frac{0}{1})^2$$
$$= 0$$

$$Gini_{TimeofShowing} = \frac{2}{3}(Gini_{Day}) + \frac{1}{3}(Gini_{Night})$$
$$= 0$$

$$Gini_{True} = 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2$$

$$= 0.5$$

$$Gini_{False} = 1 - (\frac{0}{1})^2 - (\frac{1}{1})^2$$
$$= 0$$

$$Gini_{Traffic} = \frac{2}{3}(Gini_{True}) + \frac{1}{3}(Gini_{False})$$
$$= 0.3333$$

Just like with Information Gain, Gini Index indicates that the next best split is along Time of Showing. Since we already know that all instances of Overcast weather end with tickets being purchased, we can again simply move on to the Rainy branch.

$$Gini_{Day} = 1 - (\frac{0}{1})^2 - (\frac{1}{1})^2$$
$$= 0$$

$$Gini_{Night} = 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2$$
$$= 0.5$$

$$Gini_{TimeofShowing} = \frac{1}{3}(Gini_{Day}) + \frac{2}{3}(Gini_{Night})$$
$$= 0.3333$$

$$Gini_{True} = 1 - (\frac{0}{1})^2 - (\frac{1}{1})^2$$
$$= 0$$

$$Gini_{False} = 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2$$
$$= 0.5$$

$$Gini_{Traffic} = \frac{1}{3}(Gini_{True}) + \frac{2}{3}(Gini_{False})$$
$$= 0.3333$$

Once again, we end up with the same problem as Information Gain where Time of Showing and Traffic end up with the same Gini Index. This means that our decision tree will end up looking the same as Figure 5.

3 Problem 3: Support Vector Machines

3.1 Draw the decision boundary and identify the support vectors

Figure 6 shows the decision boundaries for the provided dataset.

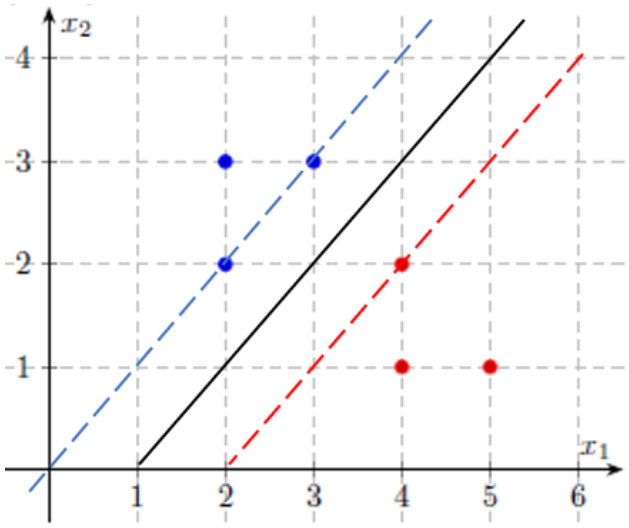


Figure 6: The blue dotted line represents the decision boundary for Class 1, and the red dotted line represents the decision boundary for Class -1

The support vectors are (2, 2) and (3, 3) for Class 1 and (4, 2) for Class -1.

We can calculate maximal margin as $\frac{2}{||W||}$ where $||W||$ is the Euclidean norm of W. In this case, it is $\sqrt{(4-3)^2+(2-3)^2} = \sqrt{2}$. This means that for the generalized equation $W * x + b = 0$, W = [1, -1] and $b = 1$.

3.2 How would the classification boundary change if we used a non-linear kernel, such as the Radial Basis Function (RBF) kernel?

When dealing with data that is not linearly separable, we first project the data to a higher dimensional space so that it can be linearly separable. Once we've done that, we can then use a non-linear kernel like RBF to find the separating hyperplane. This hyperplane also tend to appear non-linear when viewed from the original dimensions, often appearing as curved lines or even complete circles. Using a kernel can be thought of similarly to performing the dot product on the original data. RBF, in this case, uses the squared Euclidean distance and an additional parameter, γ, to determine the boundaries and how smooth they will be. These new boundaries will go through however many dimensions the new space contains instead of the two dimensions this data originally used.

For the data given in this problem, we can test to see what the resulting hyperplane from RBF would look like by using the sklearn and mlxtend libraries in python. The code below produces a rough indication of the decision boundaries. We can then easily adjust γ to see what result that has on the graph.

```
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import numpy as np
from mlxtend.plotting import plot_decision_regions
```

```
X = np.array([[3, 3], [2, 3], [2,2], [4,1], [4,2], [5,1]])
y = np.array([1,1,1,-1, -1, -1])
svm = SVC(kernel='rbf', gamma = 0.1, C=0.5)

svm.fit(X, y)
plot_decision_regions(X, y, clf=svm)
plt.show
```

Figure 7 shows the equivalent result from the code, drawn onto the originally provided graph.

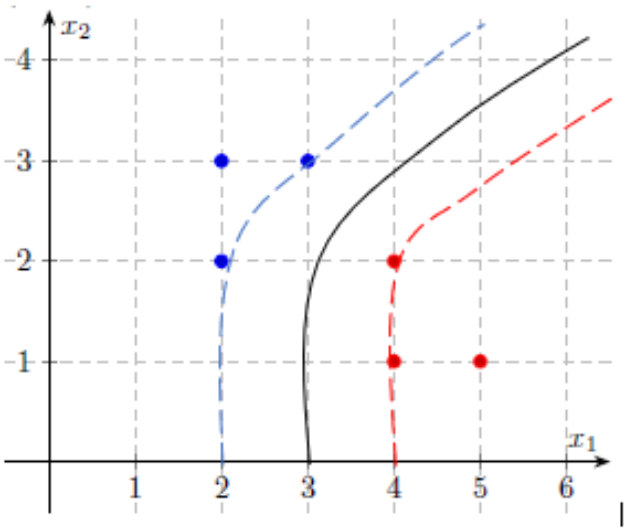


Figure 7: Possible boundaries for RBF. As before, the blue dotted line represents the decision boundary for Class 1, and the red dotted line represents the decision boundary for Class -1

We can see that now the support vectors likely includes (4, 1) as the boundaires have shifted. As γ is increased, the decision boundaries begin to shift to become less and less curved and become closer to vertical lines.

3.3 Explain how the C parameter in SVM affects the decision boundary and generalization of the model.

The C parameter controls the size of the margin used when creating the boundaries to classify data points. A smaller C creates larger margins, while a larger C creates smaller margins. Generally speaking, larger margins will result in more mis-classifications and errors than a smaller C.

4 Problem 4: Comparative Analysis and Extension

4.1 Compare the Bayesian Networks, SVM, and Decision Tree models in terms of their assumptions, strengths, and weaknesses. Provide at least two real-world application scenarios where one model might be preferred over the others.

Bayesian networks rely on the assumption that most, if not all, of the dependencies are known in order to classify an outcome or to find its probability. In the rare cases where this is true, this allows

the model to calculate the likelihood of a certain outcome given specific conditions and vice versa. The model can also be incrementally trained by adding in new training data to adjust probabilities, but tihs can run into the problem of large number of calculations that can bog down the runtime. It is also unlikelly that all dependencies will be known ahead of time in a real world situation.

Naive Bayes tries to avoid this dependency problem by assuming that ll variables are independent of one another. Despite its simplicity, this method often produces satisfactory results with the limited information and is often comparable to other, more complex algorithms. It is also more convenient to use in real world situations as we rarely know all of the dependencies within a dataset, and it has a reduced computation cost as we don't need to consider dependencies. However, ignoring dependencies also means that we inevitably lose some degree of accuracy, especially as the number of dependencies increase and become more intertwined.

Decision Trees, visuallly, are a very effective way of displaying the classification process for a dataset. Each step splits the dataset based on a single variable selected by some algorithm. Each algorithm has pros and cons and may be better for one type of data over another. For example, Informaion Gain favors variables that have several values. Decision trees, however, assume that all entries in the dataset fall under a classification rule that can be represented in the tree structure. This assumption can lead to overfitting wherein the tree splits into too many branches in an attempt to accurately portray the outcomes. This can lead to poor accuracy for new data that may contain unseen examples.

SVMs assume that the data can be linearly separated which can cause problems when that is not the case. Kernels can help circumvent this issue by projecting the data into higher dimensions and then using a selected process to find a hyperplane that separates the data. These methods are useful for data with many features as there is no upper limit to the number of dimensions we can use. SVMs also operate under the assumption that the support vectors are the most likely points to be misclassified as they are the closest to the decision boundaries. A noteable downside of SVMs is that they do not calculate probabilities as they only classify a data point.

A real world application where SVMs might be preferred over the other mentioned models is when trying to predict if a patient has a disease or is at risk of a certain disease. Medical conditions can be complex with a multitude of symptoms and causal factors. This means that the classification model must be able to handle several features. Non-linear SVMs can handle a large number of features since it can add several dimensions without too many issues. Bayesian networks would need to know most or all of the dependencies between features, which is difficult in the case of medical diagnoses. Naive Bayes would assume that there are no dependencies which, in this case, would cause an arguably unacceptable drop in accuracy. Decision trees would likely end up overclassifying the results as it would need too many branches to represent all of the symptoms and factors.

An example of when a decision tree may be preferred over the other methods is a car dealership deciding if a customer is going to buy a car or not. The dealership would only know certain features about the customer such as credit score, income, and whether or not they already own a car. This means the number of features are limited and are either binary options or can be broken down into general ranges. The decision tree would provide a quick, simple to read, yes or no indication for a given customer. Just as with the medical diagnosis example, Bayesian networks rely on knowing the dependencies ahead of time and Naive Bayes may be too inaccurate. SVMs may work in this case as well but have a higher training and computation cost than a decision tree.

4.2 Propose a variant of the Random Forest algorithm that you think could potentially outperform the original version. Justify your proposal. The number of points given will be based on the qualty and thoughtfulness provided in your answer.

The Random Forest algorithm works by randomly selecting a subset of data points to "bag" and generates several decision trees based on each bagging. These trees are then used to classify new data. This bagging method could potentially be improved by making use of some variation of DBSCAN. Assuming that the data is plottable, DBSCAN could provide a way to better decide what data points to bag based on core points. The assumption is that by using core points in the bags, the decision trees would be based off of points that are more firmly located in a cluster. This would hopefully allow the algorithms deciding the splits for each tree to deal with less noisy data. The parameters of DBSCAN could then be adjusted to try to increase the performance of the Random Forest. This would increase the startup time and needed resources to begin the training, but DBSCAN would only need to be completed once per training. It may also be possible to make use of the centroid selection method used in KMeans++ to attempt to further diversify the core points selected for each bag, but that would only further increase the amount of time and resources needed. This additional tradeoff may not be worth the increase in accuracy depending on the use case and available resources.