Discriminating Between Healthy Individuals and Patients With Pulmonary Arterial Hypertension

Author: Gabriel Berardi

University of Glasgow
School of Mathematics - Data Analytics (ODL)

Submission Date: 16 July 2021

Contents

1. Introduction

In this project, we will look at a dataset of the gene expression profiles from peripheral blood mononuclear cells for 6 healthy and 14 hypertensive patients. More specifically, we will look at a subset of 1,000 genes and study their influence on each patient's status.

We will try to answer the following three questions:

- Do certain genes have an impact on hypertension in patients?
- If yes, which genes show the largest impact on the patient's status?
- Can we find a model to predict whether a patient is suffering from hypertension, using certain statistics of these genes?

Section 2 discusses certain steps that were performed to prepare the data. Section 3 briefly explores and visualizes the dataset. In Section 4 we fit three different models, namely a k-Nearest-Neighbours model, a Classification Tree and a Support Vector Machine to predict hypertension in patients. Finally, section 5 compares the results of these models and discusses present limitations, as well as suggestions for future improvement.

2. Data Preparation

Before we start exploring the data, we need to fix some issues and split the data into a training, validation and test set. Using the `str()` function, we can see that the `GENE_IDENTIFIER` column is a factor with only 967 levels. This is an indication that we have certain genes with several measurements. Upon further investigating the duplicate entries, we can see that there are up to 3 measurements for some genes, and there seem to be some genes with faulty measurements, where the statistic for each gene in all patients is exactly 1.0 or 0.8 respectively. These faulty measurements can be dropped from the dataframe. Additionally, there is one entry where the `GENE_IDENTIFIER` is `#NAME?`, which will not be useful when we want to find the genes that influence hypertension. We will also drop this gene. However, after removing the wrong measurements, we still have 26 genes with multiple measurements, with no further indication of faulty entries. Let's therefore aggregate the statistics for these genes by taking the mean of all measurements. We are left with 923 uniquely measured genes. Other than that, the data seems valid and clean.

Before we split the data into a training, validation and test set, we should reshape the dataframe. This will facilitate any plotting and model building we do later on. Therefore, we can transpose the dataframe into tidy format, so that we have our features (genes) as columns and our observations (patients) as rows. Last but not least, we can add a further column `PAH` to indicate whether a patient is healthy `(PAH = 0)` or hypertensive `(PAH = 1)`.

Finally, we can split our data into a training, validation and test set. Since we only have 20 observations, we must make sure that each of the three sets contains observations that are healthy and hypertensive. This can be done by dividing the data into healthy and hypertensive patients, apply random sampling and then merging the data back together. In this way, we end up with a training set that contains 10 observations with 3 healthy and 7 hypertensive patients, a validation set that contains 4 observations with 1 healthy and 3 hypertensive patients and a test set that contains 6 observations with 2 healthy and 4 hypertensive patients.

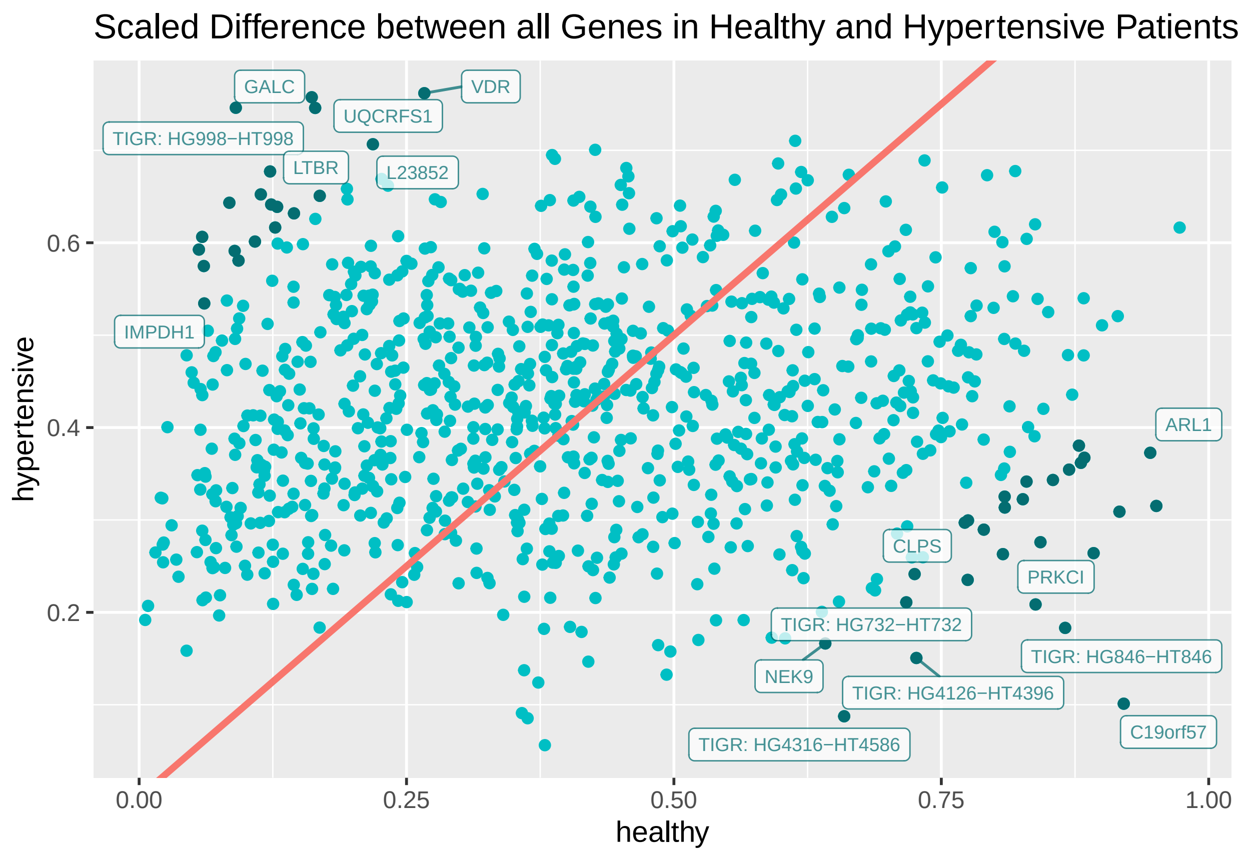
3. Exploratory Analysis

Let's start the EDA with a look at how all of the genes in our dataset differ among healthy and hypertensive patients. We can first find the average value for each gene in the training set with respect to healthy and hypertensive patients. Next, we can draw a scatterplot with 923 dots each representing a gene and its average value in healthy and hypertensive patients. However, since the order of magnitude of each gene's statistic differs quite a lot, we should first scale each gene's measurement between 0 and 1, for example by using MinMax-Scaling, which allows us to scale the features between 0 and 1 using the following formula:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

where x' is the scaled value and x is an original value.

We end up with the following plot:



Points that lie on or near the red line of equality represent genes that do not differ much between healthy and hypertensive patients. The further away from the red line a gene is plotted, the higher is its difference for the two groups of patients. As we can see, there are quite many genes that take distinct values for the healthy and hypertensive patients. In the plot above, genes with a scaled difference larger than the 95% quantile are plotted in a darker blue and some of them are annotated as well. The gene with the largest distance is `C19orf57`.

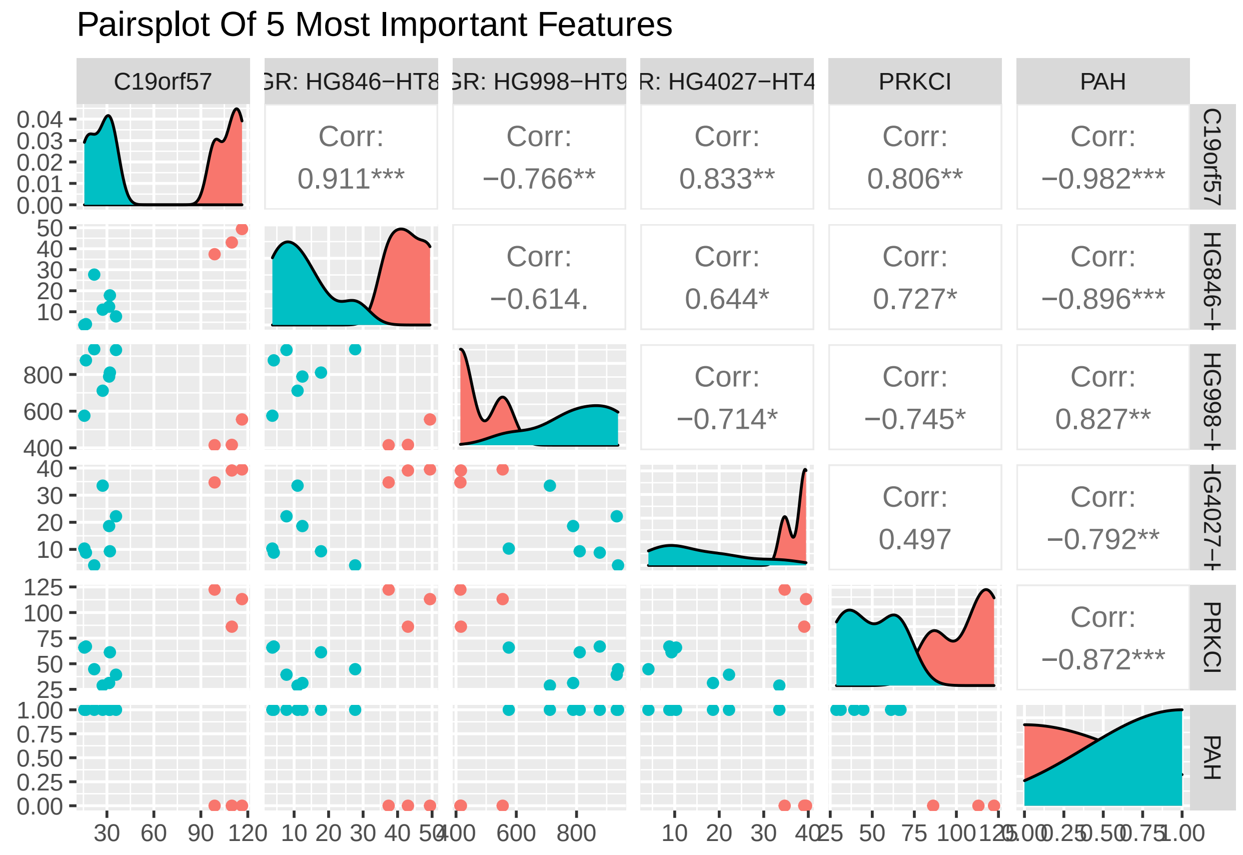
Next, let's take a look at the point-biserial correlation between the different genes and the binary target feature `PAH`. First, we can extract the genes that have a correlation with the target feature larger than the 0.99 quantile. The reason why we choose the 0.99 quantile is that a 0.95 quantile would include some weak correlations around 0.56.

Let's look at the first 5 rows:

Gene	Target	Correlation	p-value
C19orf57	PAH	-0.9819259	0.0000005
TIGR: HG846-HT846	PAH	-0.8959364	0.0004517
GALC	PAH	0.8807452	0.0007644
PRKCI	PAH	-0.8724807	0.0009891
TIGR: HG4126-HT4396	PAH	-0.8638479	0.0012714

As we can see, there are some very strong correlations between the genes and the target feature. Furthermore, 28 of the 32 genes that correlate most with the target feature are also among the genes whose average scaled difference is the largest for the healthy and hypertensive patients.

Let's draw a pairsplot for the 5 genes that correlate most with the target feature and whose average scaled difference is also the largest for the healthy and hypertensive patients, with red dots representing healthy and blue dots representing hypertensive individuals in the training set:

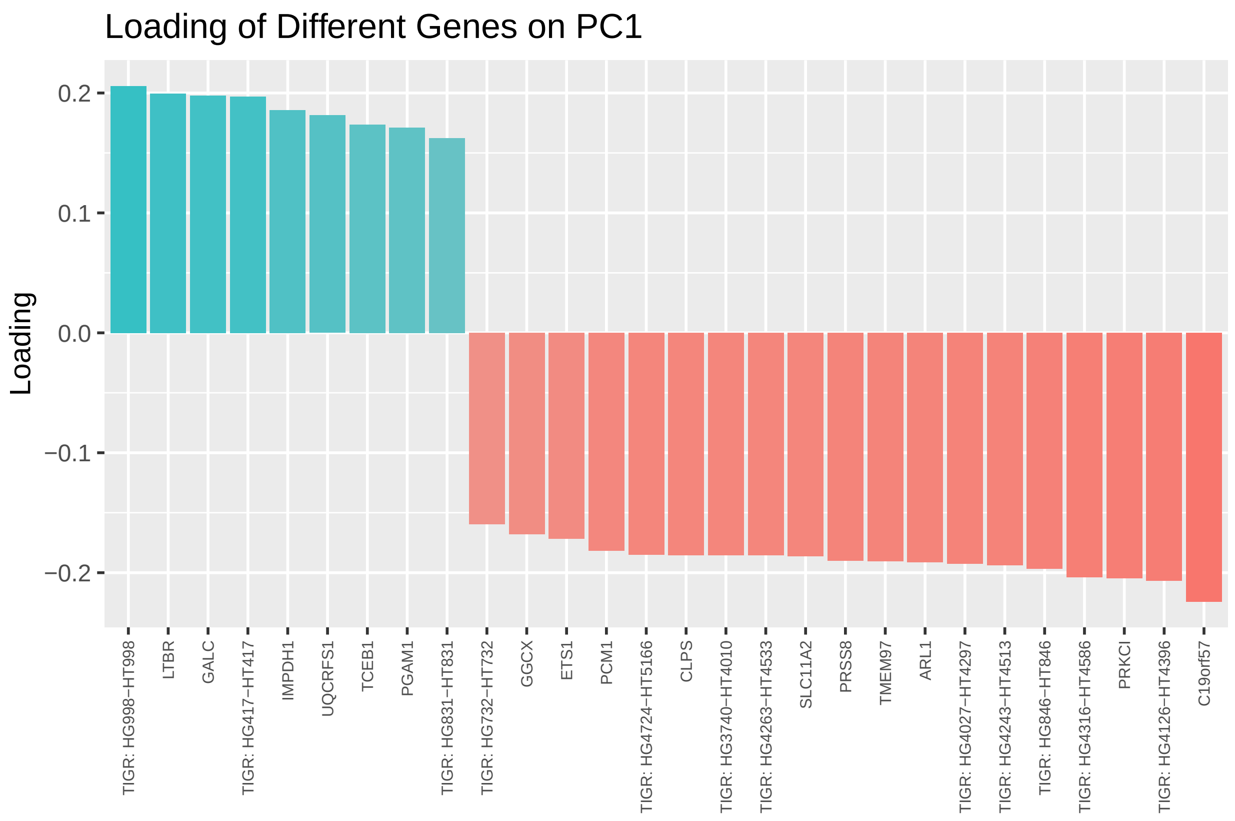


This plot shows how clear the distinction between healthy and hypertensive patients is when looking at these 5 genes. In fact, we can easily see that a linear separator would be able to divide patients into healthy and hypertensive on all of these 5 genes. Besides, the pairsplot also shows that there are very high correlations between the genes.

Last but not least, we can also perform Principal Component Analysis on the training data to gain further insights. For this, we will first select the 28 genes that correlate most with the target feature and whose average scaled difference is also the largest for the healthy and hypertensive patients. We will use the `prcomp` function to perform the dimensionality reduction and set the parameter `scale.` to `TRUE`, since the standard deviation among the 28 variables differs between 0.48 and 1,314.28. The summary of the resulting PCA is:

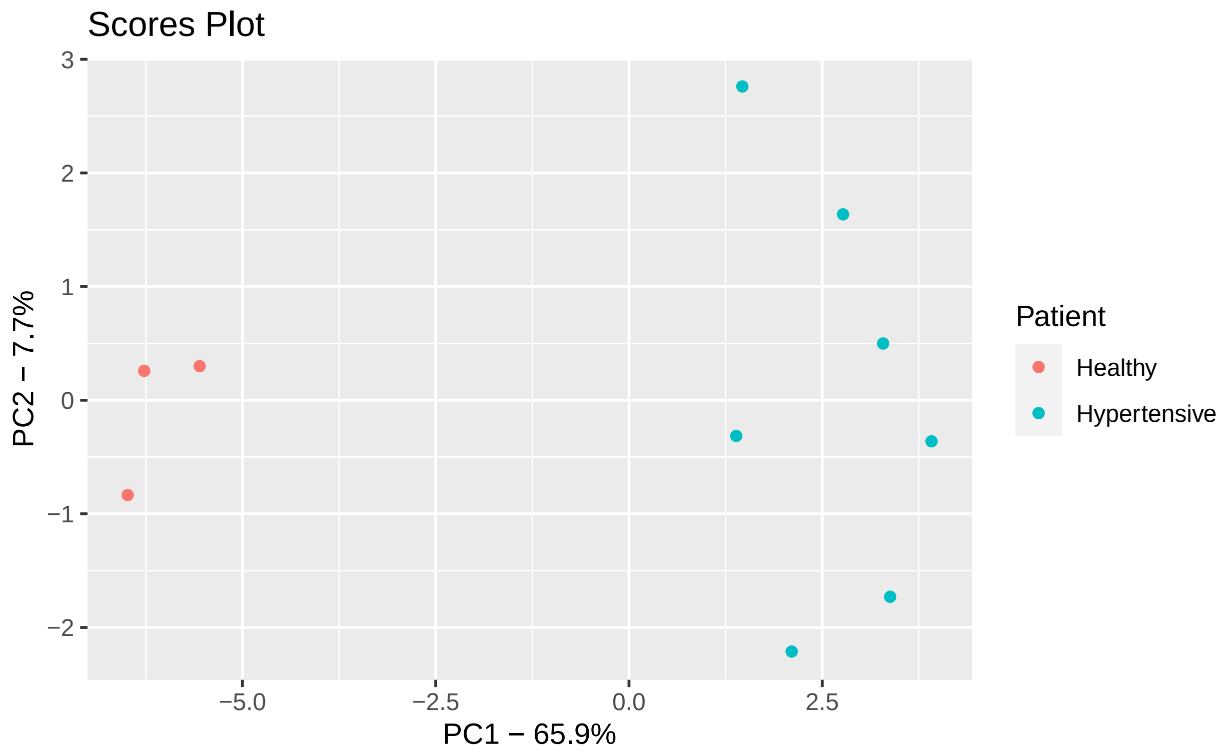
```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     4.2942 1.47258 1.37181 1.28558 1.06769 1.05530 0.92011
## Proportion of Variance 0.6586 0.07745 0.06721 0.05903 0.04071 0.03977 0.03024
## Cumulative Proportion  0.6586 0.73602 0.80323 0.86225 0.90297 0.94274 0.97298
##                            PC8     PC9      PC10
## Standard deviation     0.70808 0.50522 7.548e-16
## Proportion of Variance 0.01791 0.00912 0.000e+00
## Cumulative Proportion  0.99088 1.00000 1.000e+00
```

As we can see, three PCs constructed from the 28 variables would be enough to retain more than 80% of the variation. The first PC alone already accounts for almost 66% of the variation. Let's look at its loadings:



As we can see, all loadings in PC1 roughly lie between ± 0.2, without too much of a deviance. PC1 can therefore almost be interpreted as an average of these 28 genes.

Finally, let's draw a scores plot for the 10 patients in our training set:



As we can see, there is a very clear distinction between healthy and hypertensive patients on the first principal component. However, since we are interested in determining which genes influence a patient's status most, we will fit out models using the 28 genes from the original dimensionality space, which allows for easier interpretation.

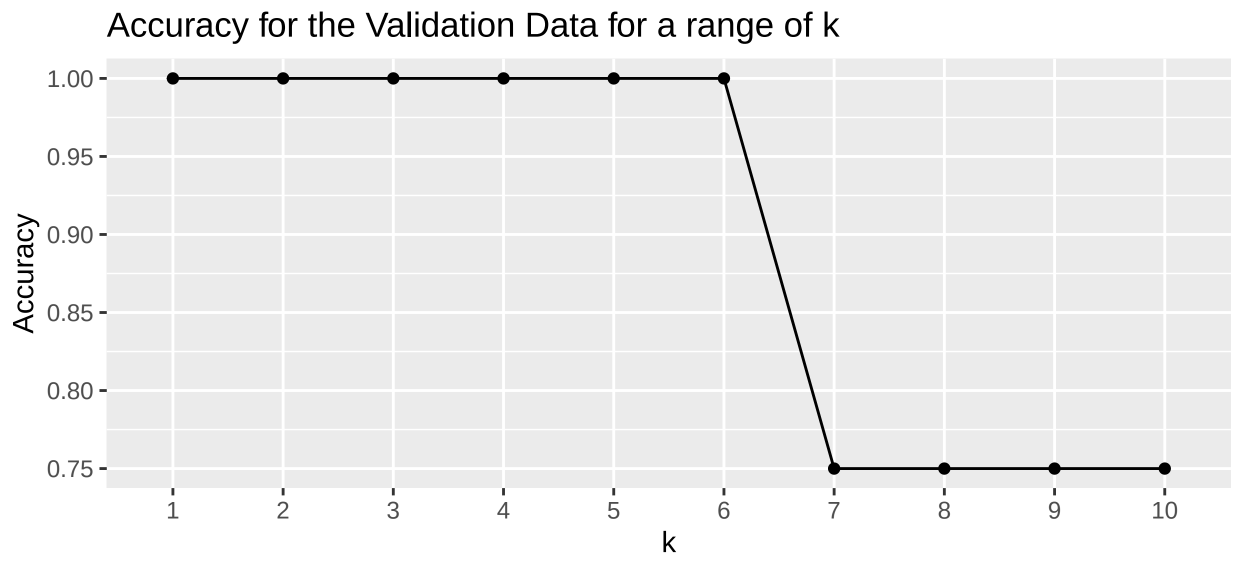
4. Results

The fact that we have some genes that correlate very strongly with the target feature indicates that we can drastically reduce the number of features we use in all of our models. We will first take the intersection of the genes with a large scaled difference between healthy and hypertensive patients and the genes that correlate most with the target feature. As discussed above, this results in a list of only 28 variables to further work with.

4.1 k-Nearest-Neighbours (kNN)

The first model we can apply to our data is the kNN algorithm. Since kNN is an algorithm that uses the distance among observations, we need to scale our data to ensure that different orders of magnitude with respect to different variables do not influence the model. There are different scaling methods we could choose. Arguably the easiest is MinMax-Scaling, which has been explained above, and which we will also apply here. Furthermore, as we have seen that the `C19orf57` gene has a very high correlation with the target variable `PAH`, we will first try to only use this gene in our kNN model.

Before we can fit a kNN model, we need to determine k, the number of neighbours that should be considered when predicting a class. We can try different values for k and check the accuracy on the validation set, which yields the following result:



The accuracy on the validation set is 100% for values of k between 1 and 6 and then drops to 75% for values greater than 5. Of course, the size of our validation set is very small and hence the significance of this test is limited. Nevertheless, we will use this information to set a value for k. Looking at the plot above, we could set k equal to 1, 3 or 5. Note that an odd number should be chosen in order to avoid incertitude through equal voting. Lower values for k tend to lead to an overfitting model, while too large values can lead to underfitting models.

Here, we will set k equal to 5, fit the kNN model on the training set and evaluate its performance on the test set. We end up with the following confusion matrix:

	Predicted: Healthy	**Predicted: Hypertensive**
Actual: Healthy	2	0
Actual: Hypertensive	0	4

From this, we can extract the following performance indicators:

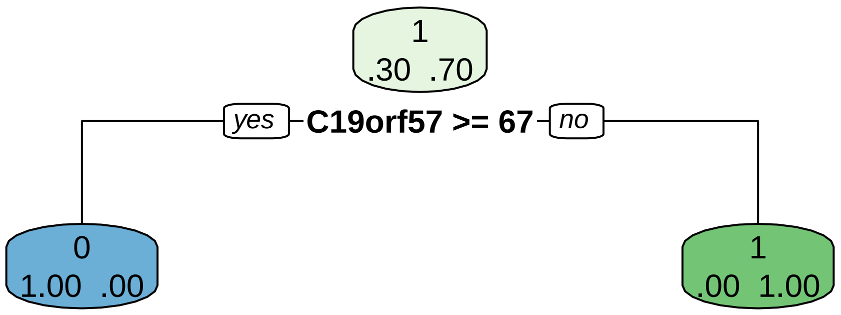
Accuracy	Class-Specific Accuracy	True Positive Rate	True Negative Rate
1	1	1	1
	1		

As we can see, the kNN model was able to perfectly predict whether or not a patient was healthy or hypertensive, using only one gene. Of course, we only have 6 samples in our test dataset and should therefore be careful about the long-term performance of our model. This is true for all of the models and will be discussed later.

4.2 Classification Tree

Next, we will look at a Classification Tree. Similarly as before, we will only use `C19orf57` as a predictor. Unlike the kNN algorithm, Classification Trees are invariant to monotonic transformations of any feature variable, meaning that scaling the data is not required.

After fitting a Classification Tree using the information as the splitting index, the resulting model looks like this:



As we only used one variable in the model, there is only one node in this tree. If the statistic for `C19orf57` is above or equal to 67, the tree predicts that the patient is healthy and otherwise hypertensive. If we look at the training data used in this tree, we can easily see how this decision rule came into place:

C19orf57	98.8	109.8	116.3	16.6	15.6	31.9	21.9	31.4	27.3	35.8
PAH	0	0	0	1	1	1	1	1	1	1

The highest value for hypertensive patients is 35.8 and the lowest value for healthy patients is 98.8. The average of these two numbers is $(35.8 + 98.8)/2 = 67.3$. While this rule works perfectly on the validation set, it does not yield good results on the test set:

	Predicted: Healthy	**Predicted: Hypertensive**
Actual: Healthy	2	0
Actual: Hypertensive	2	2

From which we can extract:

Accuracy	**Class-Specific Accuracy**	**True Positive Rate**	**True Negative Rate**
0.67	1.0 0.5	0.5	1

The interesting thing is, that this poor performance has nothing to do with the fact that we only used one variable for prediction. If we include all of our 28 genes that we found to distinguish well among healthy and hypertensive patients, the resulting Classification Tree is identical, i.e. it only uses `C19orf57` to split the data, even if we try to enforce more splits by adjusting the `minsplit`, `minbucket` or `cp` parameters of the `rpart()` function.

The reason why this happens is that no further split beyond `C19orf57` would decrease the overall lack of fit of the model. In other words, we can achieve a perfectly accurate split on our training (and validation set), by only using `C19orf57`. This is partly also influenced by the fact that we simply don't have enough samples to fit a better Classification Tree.

4.3 Support Vector Machine (SVM)

Last but not least, we will use an SVM model for the prediction. By fitting different combinations out of the most important genes, we can see that the model performs best on the validation set when using all of the 28 genes we have determined above. And similarly to the kNN model, because SVM optimization works through minimizing the decision vector, the optimal hyperplane is influenced by the scale of the input features and we should therefore also apply MinMax-Scaling to our data.

Since there are such high correlations between the predictors and the target feature and we have seen that we could easily find linear separators among the two classes, we will use a linear kernel as a starting point. Therefore, we only have one parameter that we need to determine before fitting our model, which is the cost parameter. For this, we can use a vector containing a range of different options for the cost parameter, for example by using `seq(0.0001, 10, length=1000)`. Then, we can test the parameter of 1,000 different models with each of the different cost parameters and select the cost parameter that yields the best-performing model, i.e. that yields the lowest classification error. In this way, we determine the best cost parameter to be 0.0101. We then go ahead and fit the SVM model with a linear kernel and a cost parameter equal to 0.0101. Since we have an imbalanced training set with respect to the target classes, we can also specify the `class.weights` parameter to reflect this.

The resulting SVM model has the following confusion matrix:

	Predicted: Healthy	**Predicted: Hypertensive**
Actual: Healthy	2	0
Actual: Hypertensive	0	4

From this confusion matrix, we can again extract the performance indicators:

Accuracy	**Class-Specific Accuracy**	**True Positive Rate**	**True Negative Rate**
1	1	1	1
	1		

As we can see, the SVM model is also able to perfectly predict which patients in the test set are healthy or hypertensive However, it uses 28 genes as predictors, compared to only 1 gene in the case of the kNN model. Let's now move on to further discuss these results.

5. Discussion

For the last part of this paper, we will review and comment on the performance of the models described above, discuss existing limitations and potential improvements for further analyses.

5.1 Review

Let's start with the worst out of the three models we have looked at, which is the Classification Tree. As we have seen, a Classification Tree seems unsuitable to perform the prediction task on this specific dataset, as it only uses one gene to perform the classification, while choosing an unideal cut off value. Classification Trees in general have a tendency to overfit on the training data. In a situation where we only have 10 samples to train our model with, this becomes an even larger problem, as has been demonstrated above. Even worse, due to the general disparity of the dataset with 14 hypertensive and only 6 healthy patients, we are also facing the problem of an imbalanced training set, and due to the small number of samples, downsampling is no option either. Hence, the Classification Tree model is the least preferable here. The only positive thing one could mention, is that it would allow us to spot gene `C19orf57` as the most influential gene with regard to Pulmonary Arterial Hypertension.

For the kNN and SVM model, we have seen that both of them are able to perfectly predict whether or not an individual suffers from Hypertension. However, the kNN model achieves this by using only one gene as a predictor, while the SVM uses 28 genes. From this point of view, the kNN model would be preferred over the SVM in a practical setting, as we would only need the information about this single gene in order to predict a diagnosis. Moreover, the kNN algorithm is much simpler and hence easier to interpret and explain to doctors and/or patients, which is a factor that should always be taken into consideration when developing predictive models for a clinical setting.

In general, kNN models have the downside of being *lazy learning* algorithms, which means that the generalization of the training data is deferred until a new observation needs to be classified, as opposed to *eager learning*, where the algorithm tries to generalize the training data before that. This also implies, that the kNN model needs to store all data points in the training set, which makes it computationally more expensive and potentially slower. However, with a dataset as small as the one we are dealing with here, this remains a theoretical issue only, but it is worth mentioning anyway.

Coming back to the first two questions mentioned in the introduction, we can conclude that - given the dataset we have - certain genes do seem to have an impact on Pulmonary Arterial Hypertension, with `C19orf57` having the largest influence.

5.2 Limitations

The perfect accuracy of the kNN and the SVM model seem very impressive at first. However, it is crucial to note that we might also simply be lucky with the specific constellation of our training, validation and test set. A test set with only 6 samples is just too small to make a statement about the overall, general quality of each model. The third question mentioned in the introduction can therefore only be partially answered. Yes, we are able to fit accurate models on this dataset, to predict whether a patient is suffering from hypertension. No, we cannot guarantee that the performance of these models will hold for a larger amount of samples.

Aside from this, it is also worth mentioning that we cannot make any statement about any causal effects. We see evidence of a strong correlation and potential infleunce between certain genes and PAH, but we cannot say whether these genes *cause* this illness or not. There might be other root causes outside of the dataset and our models, which both lead to certain statistics in these genes, as well as PAH in patients.

5.3 Improvements

There are many ways how we could improve the validity of this analysis. First of all, we could do the following with respect to our dataset:

- Collect data for more healthy and hypertensive patients, ideally accumulating a much larger and more balanced dataset.
- Analyze more than the 1,000 genes we started out with.
- Collect and use other potentially influential variables such as the gender or age of each person.

Aside from this, there are also things we could do with regard to the modelling aspect:

- There are many ways we could further tweak the models we have used above. For example, we could try using different transformations before fitting the models, applying other scaling methods, or using different optimization parameters (e.g. different distance measures for the kNN, different splitting indices for the Classification Tree or different kernel methods for the SVM).
- Of course, we could also try other models, such as a Logistic Regression or an Artificial Neural Network.

This concludes this project summary. As we have seen, we are able to perform accurate predictions with only very few of the original 1,000 genes we looked at. However, certain limitations exist that are worth dealing with to further improve the validity of these findings.