Cancer Classification using Gene Expression Data

HarvardX Data Science: Capstone

Skyler Shapiro

1/8/2021

Contents

1. Introduction

1.1 Project Overview

This is the second and final project of the capstone of the Data Science Professional Certificate program offered by HarvardX. The certificate program is comprised of nine courses, with the first 8 covering topics from basic R programming to probability to machine learning techniques. The 9th and final course is the capstone, which is contains a movie recommender system and an independent project.

The goal of this project is to classify cancer type using gene expression data. To achieve this goal, I implemented techniques including dimension reduction by Principal Components Analysis, random partitioning of data, and three machine learning multinomial classification algorithms.

1.2 Data

This dataset contains gene expression data for 198 human patients, all of whom have one of 14 types of cancer: breast, prostate, lung, colorectal, lymphoma, bladder, melanoma, uterus, leukemia, renal, pancreas, ovary, meso and cns. The patients are split across two groups, training (144) and testing (54). The data is broken up into four files: "14cancer.xtrain.txt" and "14cancer.ytrain.txt", and "14cancer.xtest.txt" and "14cancer.ytest.txt". The matrices "14cancer.xtrain.txt" [16,063 x 144] and "14cancer.xtest.txt" [16,063 x 54] contain gene-expression data. The lists "14cancer.ytrain.txt" and "14cancer.ytest.txt" contain the cancer type labels of each subject. The contents of the data, are quantitative continuous with 14 categorical outcomes. All four data files were pre-cleaned, having no empty cells or NA values.

This data was created by S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C.H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J.P. Mesirov, T. Poggio, W. Gerald, M.Loda, E.S. Lander, T.R. Golub (2001) (https://www.pnas.org/content/98/26/15149). In their paper they used a multiclass classifier based on a support vector machine algorithm and achieved a prediction accuracy of 0.780.

The data was downloaded from (https://web.stanford.edu/~hastie/ElemStatLearn/) and is also available to download on the GitHub.com project repository (https://github.com/skylershapiro/Cancer_Gene_Expression).

1.3 Algorithms

The three algorithms selected for this project were K-Nearest Neighbors, Random Forest, and Multinomial Logistic Regression. These algorithms were picked mainly because of their capacity to handle multinomial classification tasks.

1.3.1 K - Nearest Neighbors, KNN

K-Nearest Neighbors, or KNN is a model that classifies data points based on the points that are most similar to it. It uses test data to predict classification for unclassified points. We can also specify the number of similar points, or "neighbors" to consider when predicting to tune our model. Some of the benefits of KNN are its ease of use and low calculation time.

1.3.2 Random Forest, RF

Random Forests consist of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. Some of the benefits of this model are that it can be used for both regression and classification tasks. Additionally, it is also easy to view the relative importance it assigns to the input features.

1.3.3 Multinomial Logistic Regression, MLR

Unlike standard logistic regression which can only handle binary classification tasks, Multinomial Logistic Regression is a form logistic regression that is well suited to handle multiclass classification tasks.

1.4 Accuracy Measurement

In order to evaluate the performance of the models, it is important to establish a standard measure of accuracy. Unlike data with continuous outcomes, take movie ratings for example, we cannot use a loss function like Root Mean Squared Error to determine accuracy. Because our data has strictly categorical outcomes, the simplest way to measure accuracy is to to divide the number of correctly predicted cancer types over the total number of predictions. Using this calculation, dividing correct guesses by the total number of guesses, we will be able to evaluate and compare the models.

2. Methods and Analysis

2.1 Data Wrangling

Although the data had already been split into testing and training files, I recombined all of the data files so I could create my own randomly generated training and testing sets. The two objects containing gene expression data, "xtrain" and "xtest" were bound together by column and then transposed to produce a [16,063 x 198] data frame. Each row of the data frame corresponds to a cancer patient, and each column corresponds to a different gene expression measurement. The cancer type labels were originally a vector of numbers, 1-14, that corresponded to a cancer type. To make my code more interpretable, I converted the numeric vector into a character vector of text labels,The raw data files can be accessed through the project repository on GitHub at (https://github.com/skylershapiro/Cancer_Gene_Expression). In the first table below, we can see the first five rows of our gene expression data, and in the next table we can view the cancer type labels.

```r
# Download data files from github repo
ytrain <- scan(
"https://raw.github.com/skylershapiro/Cancer_Gene_Expression/master/14cancer.ytrain.txt")
xtrain <- read.table(
"https://raw.github.com/skylershapiro/Cancer_Gene_Expression/master/14cancer.xtrain.txt",
header=FALSE, sep="")

ytest <- scan(
"https://raw.github.com/skylershapiro/Cancer_Gene_Expression/master/14cancer.ytest.txt")
xtest <- read.table(
"https://raw.github.com/skylershapiro/Cancer_Gene_Expression/master/14cancer.xtest.txt",
header=FALSE, sep="")

# Create cancer gene expression dataset from pre-made test and train sets
cancer_gene_expression <- as.data.frame(t(cbind(xtrain, xtest)))
rownames(cancer_gene_expression) <- 1:nrow(cancer_gene_expression)
cancer_type_nums <- c(ytrain, ytest)
```

Table 1: First 5 Rows and 10 Columns of cancer_gene_expression data set

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
1	-73	-69	-48	13	-86	-147	-65	-71	-32	100
2	-16	-63	-97	-42	-91	-164	-53	-77	-17	122
3	4	-45	-112	-25	-85	-127	56	-110	81	41
4	-31	-110	-20	-50	-115	-113	-17	-40	-17	80
5	-33	-39	-45	14	-56	-106	73	-34	18	64

Table 2: Table of Number and Text Labels of Cancer Types

number_label	text_label
1	breast
2	prostate
3	lung
4	colorectal
5	lymphoma
6	bladder
7	melanoma
8	uterus
9	leukemia
10	renal
11	pancreas
12	ovary
13	meso
14	cns

2.2 Principal Components Analysis (PCA) for Dimension Reduction

Using Principal Components Analysis, we can transform a large set of variables into a smaller one that still contains most of the information in the large set. This dataset has an extremely large number of predictors (16,063). Using PCA will allow us to reduce the dimensions of our data to shorten code runtime, scale down the size of our computation, and improve model accuracy. The function “prcomp()” was used to calculate the principal components.

```r
# Calculate principal components
pc <- prcomp(cancer_gene_expression)
X <- pc$x
```

2.3 Create Data Partition

Using the R function “createDataPartition()” the PCA matrix was split into testing and training sets. I chose the training-testing split to be 80/20 to ensure that there each cancer type would be represented in both sets.

```r
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`

# Create testing index
test_index <- createDataPartition(y = cancer_types,
                                  times = 1, p = 0.2, list = FALSE)

# Partition matrix of principal components using test index
pc_test <- X[test_index,]
pc_train <- X[-test_index,]

# Partition cancer type labels using test index
ctype_test <- cancer_types[test_index]
ctype_train <- cancer_types[-test_index]
```

2.4 Data Exploration

From the tables below, we can see the distribution of cancer type accross the full dataset as well as the testing and training sets. We can see that there are a few of each cancer type in the testing and training sets. Our 80/20 partition ended up working well, allowing for two representative sets.

Table 3: Frequency of Cancer Type from Full Dataset

	Count
leukemia	30
lymphoma	22
cns	20
prostate	14
breast	12
lung	12
colorectal	12
ovary	12
bladder	11
renal	11
pancreas	11
meso	11
melanoma	10
uterus	10

Table 4: Frequency of Cancer Type in Training Set

	Count
leukemia	24
lymphoma	17
cns	16
prostate	11
breast	9
lung	9
colorectal	9
ovary	9
bladder	8
melanoma	8
uterus	8
renal	8
pancreas	8
meso	8

Table 5: Frequency of Cancer Type in Training Set

	Count
leukemia	6
lymphoma	5
cns	4
breast	3
prostate	3
lung	3
colorectal	3
bladder	3
renal	3
pancreas	3
ovary	3
meso	3
melanoma	2
uterus	2

Now, lets inspect the results of our dimension reduction using PCA. We started off with over 16,000 variables, which makes building working models challenging on a standard computer. In the plot below, we can visualize the cumulative proportion of variation explained by the number of principal components. The Blue lines mark the 80% variation explained which corresponds to the first 15 principal components. Red lines mark the 90% variation explained which corresponds to the first 37 principal components. From the table below, we can see the cumulative variation explained by principal components.

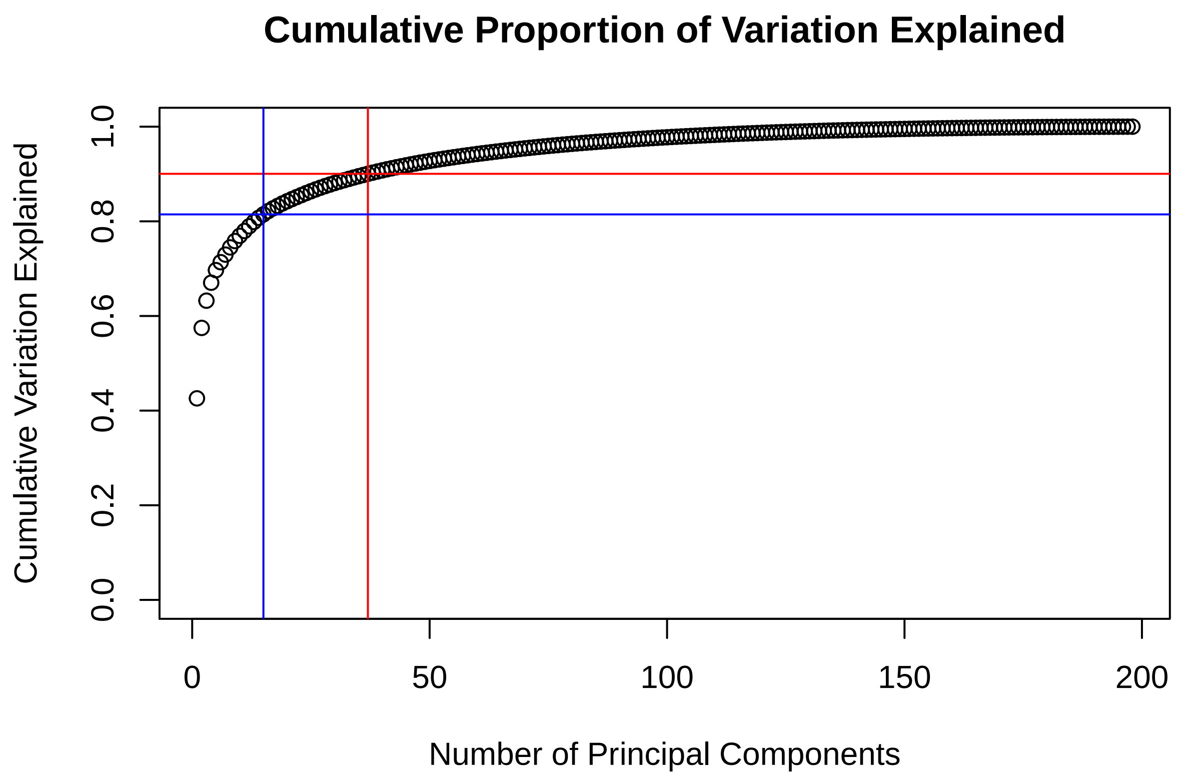
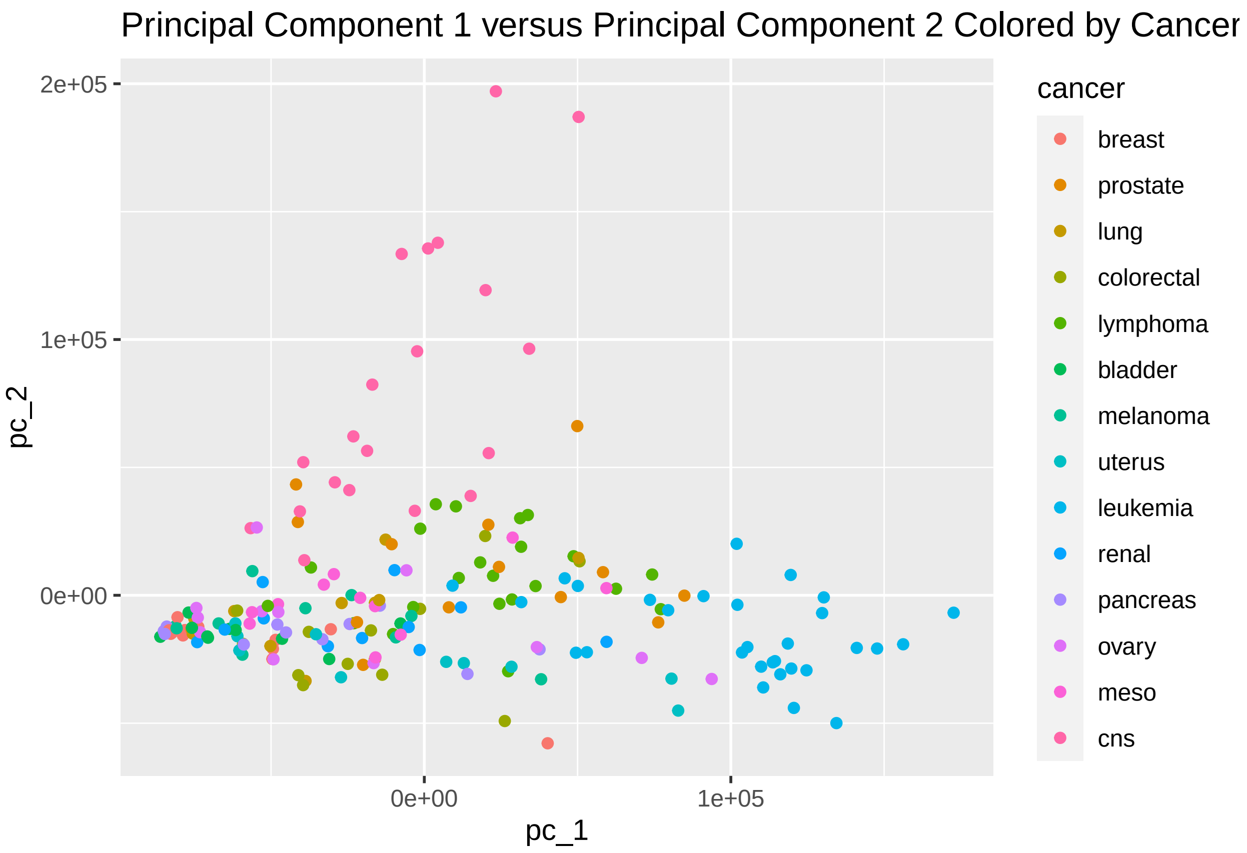
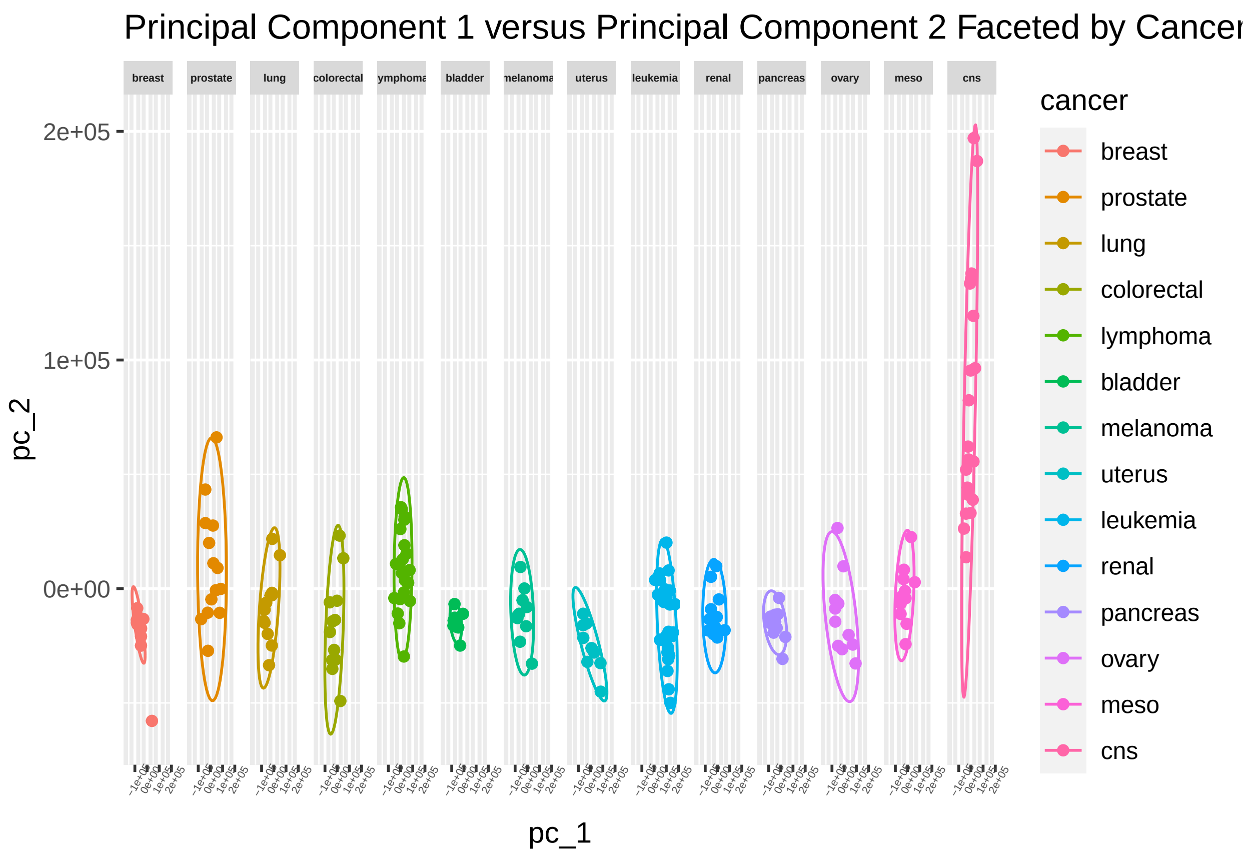


Table 6: Cumulative Variation Explained by Principal Components

number_pcs	variation_explained
First Principal Component	0.4259164
First 2 Principal Components	0.5747506
First 3 Principal Components	0.6323363
First 5 Principal Components	0.6968493
First 10 Principal Components	0.7693707
First 15 Principal Components	0.8146640
First 20 Principal Components	0.8424555
First 30 Principal Components	0.8809064
First 37 Principal Components	0.9004054

Now, we can view the distribution in the first scatterplot below of the first princial component versus the second principal component grouped by cancer type. Although we can see clusters of cns and leukemia, the distrubution doesn't seem to have a clear pattern. In the second plot, we can clearly see the distribution of the first and second principal compoments faceted by cancer type.





2.5 Model Construction

2.5.1 Data Wrangling for Model Construction

Now that the data has been explored and the dimension of our data has been reduced, we can prepare to construct our models. Earlier, we stated that we would be using the first 37 principal components to build our models as well as the first 15 components to compare later. We follow an identical procedure for the testing and training sets using the first 15 principal components as well.

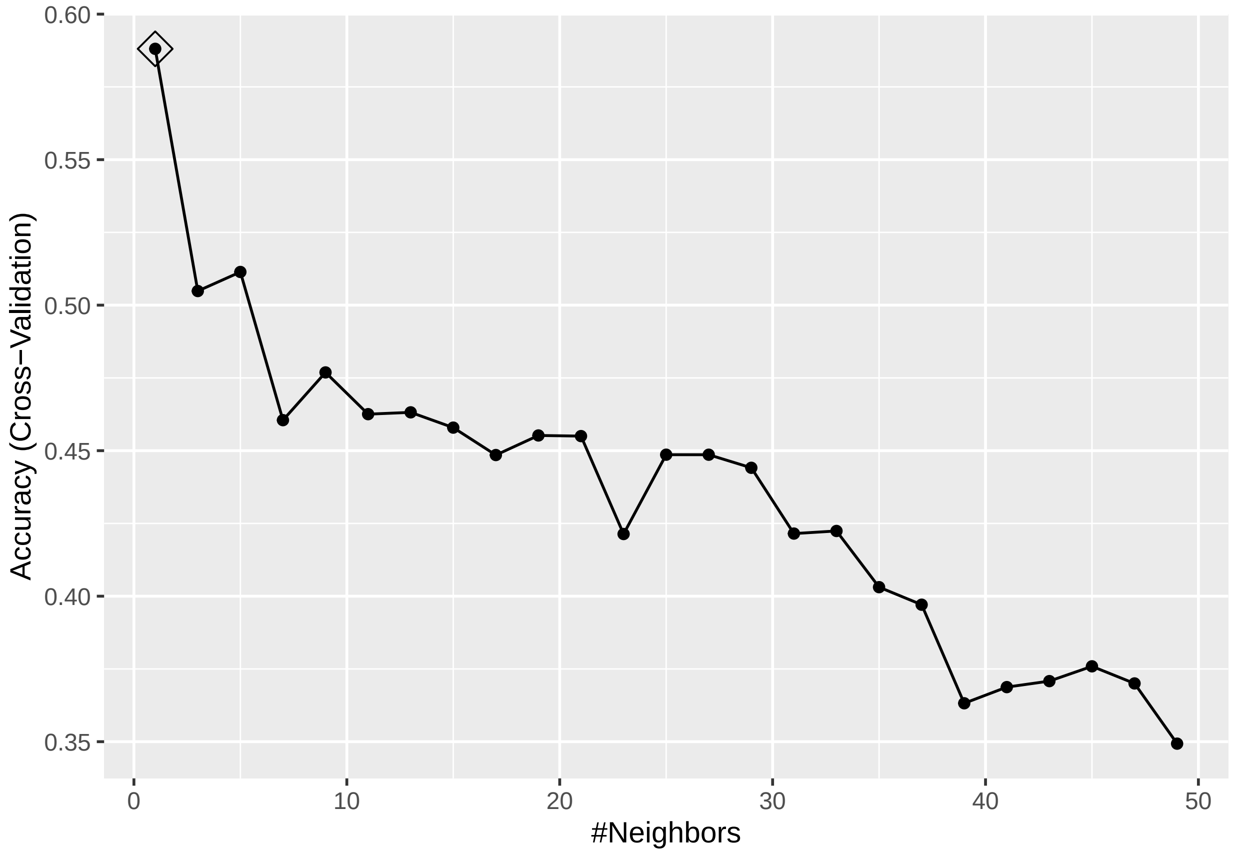
```r
# (TRAIN SET) Bind principal components matrix with cancer type labels
newdat_train <- cbind(pc_train[,1:37], ctype_train)
# Coerce class off training set to dataframe
temp <- as.data.frame(newdat_train)
newdat_train <- temp
# Re-attach cancer type labels to switch from numeric labels to text labels
# (for example changing "1" to "breast")
newdat_train$ctype_train <- ctype_train

# (TEST SET) Bind principal components matrix with cancer type labels
newdat_test <- cbind(pc_test[,1:37], ctype_test)
# Coerce class off training set to dataframe
temp <- as.data.frame(newdat_test)
newdat_test <- temp
# Re-attach cancer type labels to switch from numeric labels to text labels
# (for example changing "1" to "breast")
newdat_test$ctype_test <- ctype_test
```

2.5.2 Model 1: K - Nearest Neighbors, KNN

The first model used was K-Nearest Neighbors. The "train()" function and method argument "knn" from the Caret package was used to construct the model. One of the benefits of using the Caret package for model construction is the built-in tuning arguments. The "trControl" method was implemented for 10-fold, cross validation. The "tuneGrid" argument trains a new model for every k-value stored in the argument and stores it in the "train_knn" object making optimization very easy. From the line plot below we can see a plot of K-values versus model accuracy and that a k value of 1 yields the most accurate model in this case. We can also print out the optimal k-value using the bestTune component of our 37 principal component model.

```
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
# Build KNN model
control <- trainControl(method = "cv", number = 10, p = .9)
train_knn <- train(ctype_train ~ ., method = "knn",
                      data = newdat_train,
                      tuneGrid = data.frame(k = seq(1, 50, 2)),
                      trControl = control)
```



```
## $title
## [1] "K-values versus Model Accuracy"
##
## attr(,"class")
## [1] "labels"
```

```
# Choose optimal k
k = train_knn$bestTune
k
```

```
##   k
## 1 1
```

Next we can evaluate the model using the test set and calculate its accuracy using the "ConfusionMatrix()" function. The exact same procedure was followed for the 15 principal component model.

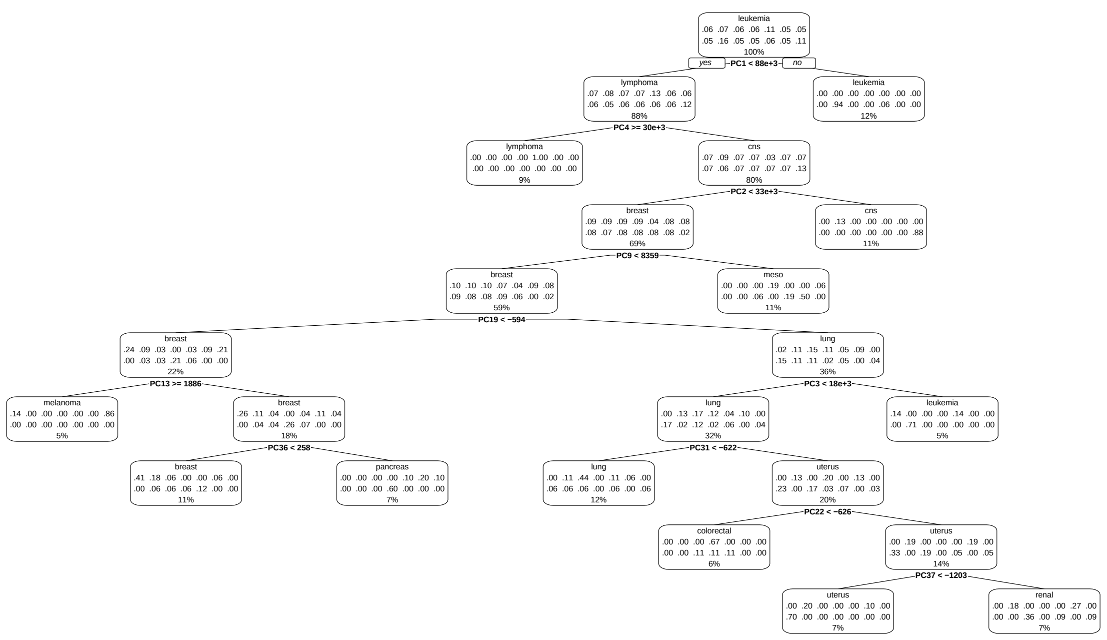
```r
# Predict on test set
y_hat_knn <- predict(train_knn, newdat_test, type = "raw")

# Check final accuracy
knn_test_acc <- confusionMatrix(data = y_hat_knn, reference = ctype_test)$overall["Accuracy"]
```

2.5.3 Model 2: Random Forest, RF

The second model was chosen was Random Forest or RF for short. The "train()" function and method argument "Rborist" from the Rborist package was used to construct the model. The Rborist package provides additional arguments for optimization such as "predFixed", which modifies the number of randomly selected predictors, and "minNode",which modifies minimal node size. Before constructing our actual model, a simpler random forest model was trained unsing the "rpart" package in order to produce an example plot, seen below for the 37 principal component model. The exact same procedure was followed for the 15 principal component model.

Decision Tree from 'rpart' Random Forest



```r
# Build rf model using Rborist package
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
control <- trainControl(method='repeatedcv',
                        number=10,
                        repeats=3)
train_rf <- train(ctype_train ~ .,
                  method = "Rborist",
                  tuneGrid = data.frame(predFixed = 2, minNode = c(3, 50)),
                  data = newdat_train,
                  trControl=control)
```

Table 7: Classification table of actual (rows) versus predicted (columns) cancer types

	breast	prostate	lung	colorectal	lymphoma	bladder	melanoma	uterus	leukemia	renal	pancreas	ovary	meso	cns
breast	0	1	1	0	0	1	0	0	0	0	0	0	0	0
prostate	0	1	0	0	0	0	1	0	0	1	0	0	0	0
lung	1	0	1	1	0	0	0	0	0	0	0	0	0	0
colorectal	0	0	0	0	1	0	0	0	0	0	0	2	0	0
lymphoma	0	0	0	0	3	0	0	0	0	1	1	0	0	0
bladder	0	1	0	0	0	2	0	0	0	0	0	0	0	0
melanoma	0	0	0	0	0	0	1	0	0	0	0	0	1	0
uterus	0	0	0	0	0	2	0	0	0	0	0	0	0	0
leukemia	0	0	0	0	0	0	0	1	3	0	1	1	0	0
renal	0	0	0	0	0	0	0	0	0	2	0	1	0	0
pancreas	0	0	0	0	1	0	0	0	0	0	2	0	0	0
ovary	0	0	0	0	0	0	0	0	0	0	0	3	0	0
meso	0	0	0	2	0	0	1	0	0	0	0	0	0	0
cns	0	0	0	1	0	0	0	0	0	0	0	0	0	3

```
# Evaluate model accuracy on test set
rf_acc <- as.numeric(confusionMatrix(predict(train_rf, newdat_test), ctype_test)$overall["Accuracy"])
```

2.5.4 Model 3: Multinomial Logistic Regression, MLR

The third and final model chosen was Multinomial Logistic Regression or MLR. MLR is a form of logistic regression that deals with multi-class categorical outcomes instead of binary outcomes (only two classes). The following code is for the 37 principal component model.

```
# Create reference
newdat_train$ctype_train <- relevel(newdat_train$ctype_train, ref = "breast")

# Build model with newly made reference
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
train_multi_log <- multinom(ctype_train ~ ., data = newdat_train)
```

After training our model, we then evaluate it on the test set. The table below displays the actual cancer type (on the rows) and the predicted cancer type (on the columns) for the 37 principal component model. Finally, the model accuracy is computed on the test set using the table. The same process was followed for the 15 principal component model.

```
# Predicting the class for test dataset
newdat_test$ctypepredicted <- predict(train_multi_log, newdata = newdat_test, "class")

# Building classification table
tab <- table(newdat_test$ctype_test, newdat_test$ctypepredicted)
tab %>%
kable(caption = "Classification table of actual (rows) versus predicted (columns) cancer types",
        format = "latex") %>%
  kable_styling(font_size = 5)
```

```
# Compute final accuracy
mlr_acc <- (round((sum(diag(tab))/sum(tab))*100,2))/100
```

3. Results

From the table below we can see that the Random Forest model using 37 principal components had the highest test accuracy at 0.782. The Random Forest model was the most accurate 15 principal component model as well with an accuracy of 0.717. The KNN 37 principal component model had the second highest accuracy at 0.586 but the lowest accuracy out of the 15 principal component models. The MLR model had the lowest 37 principal model accuracy and the lowest overall accuracy at 0.456. For unknown reasons, 15 principal component MLR model was more accurate than the 37 principal component MLR model.

Table 8: Model Accuracy using 15 and 37 Principal Components

Method	37_pcs_accuracy	15_pcs_accuracy
1: K - Nearest Neighbors (KNN)	0.5869565	0.5217391
2: Random Forest (RF)	0.7826087	0.6956522
3: Multinomial Logistic Regression (MLR)	0.4565000	0.5870000

4. Conclusion

4.1 Project Summary

The goal of this project was to classify a patients cancer type given gene expression data. Using Principal components analysis for dimension reduction, randomly partitioning data into training and testing sets, and constructing three machine learning algorithms, I was able to achieve an accuracy proportion of 0.782 using the 37 principal component Random Forest model. This was the same level of accuracy as the creators of this dataset who used a multiclass classifier based on a support vector machine algorithm. (Study reference can be found on the github project repository README file)

4.2 Limitations

A limitation of this project was the small sample size. Because there were only 198 patients for 14 types of cancer, certain types of cancer were not very prevalent. Having a larger sample could possibly improve model accuracy. Another limitation of this project was the MLR model. For an unknown reason, the 37 principal component model performed worse than the 15 principal component model even though the 37 principal component model captured more of the variation in the data. It could be that MLR is better suited to handle fewer predictors, or perhaps MLR is better suited for classification tasks with fewer classes.

4.3 Future Work

One future direction for this project would be to use an ensembling method, making use of Random Forests, K-Nearest Neighbors, and Multinomial Logistic Regression together in one model which could improve accuracy. Further, more advanced methods like Support Vector Machine (SVM) and signal processing could be used to achieve better accuracy.