# MACHINE LEARNING USING R

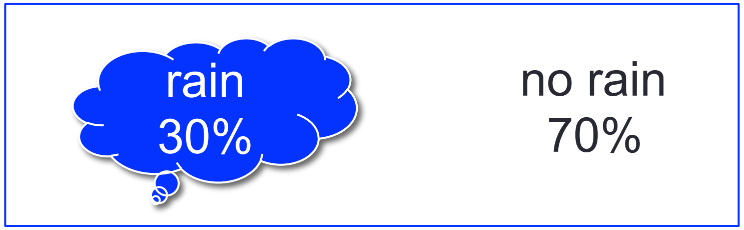
Class 3 Intensive

# naïve Bayes

- Classification using Supervised Learning
- naive Bayes uses data about prior events to estimate the probability of future events.
- Bayesian classifiers are best applied to problems in which the information from numerous attributes should be considered simultaneously in order to estimate the probability of an outcome.
- Example:
  - Using frequency of words in spam emails, determine if the incoming message is span or not
  - Using past disease data, diagnosing medical condition
  - Using past weather conditions, determine if it s a good day to play golf
  - Based on Income and Debt, classify if a person will default on payment

# naïve Bayes: Probability

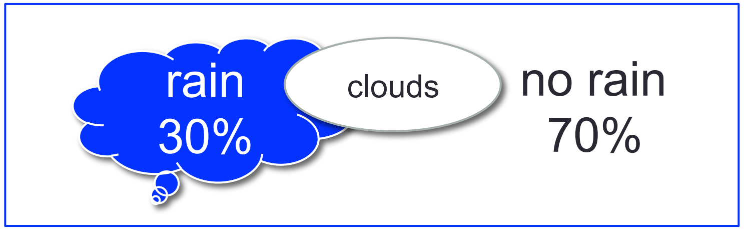
- It rained 3 days out of the last 10 days
  - Probability of rain can be estimated as (3/10)*100 or 30%.
- P(A) denotes probability of event A
  - P(rain) = 0.30
- Total probability of all possible outcomes in a trial must be 100%
  - P(rain) + P(no rain) = 1 or P(no rain) = 0.7
    - i.e. 70% probability of no rain
  - This works because rain and no rain events cannot occur at the same time i.e. they are mutually exclusive and exhaustive
    - P(¬rain) = 0.7



# naïve Bayes: Joint Probability

- In reality we are monitoring events that are non-mutually exclusive.
- If some event occurs with event of interest, we could use them to make predictions.
- Joint Probability $P(A \cap B) = P(A) * P(B)$
- Example:
  - Clouds can be present in the sky on a day when it rains but they could also be present on non-rainy days.
  - 25% of days when there were clouds, it rained.
  - Joint probability $P(rain \cap clouds) = P(rain) * P(clouds) = 0.3*0.25$
    $= 0.075$

This is case for independent events.
In reality rain and clouds are highly dependent.



# naïve Bayes: Conditional Probability

- Relationship between dependent events can be described using Bayes' Theorem:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} = \frac{P(A \cap B)}{P(B)}$$

- **Prior probability P(A):** without any additional evidence, most reasonable guess that it will rain i.e 30%
- **Likelihood P(B|A):** probability that clouds appeared on previous rainy days
- **Marginal Likelihood P(B):** probability that clouds appeared on any day
- **Posterior Probability P(A|B)**: how likely it will rain today
    - Posterior Probability > 50% : it will rain today

$$P(rain \mid clouds) = \frac{P(clouds \mid rain) * P(rain)}{P(clouds)}$$

# naïve Bayes: Conditional Probability

| | clouds | | |
|---|---|---|---|
| **Frequency** | **Yes** | **No** | **Total** |
| **rain** | 23 | 7 | 30 |
| **no rain** | 2 | 68 | 70 |
| **Total** | 25 | 75 | 100 |

| | clouds | | |
|---|---|---|---|
| **Likelihood** | **Yes** | **No** | **Total** |
| **rain** | 23 / 30 | 7 / 30 | 30 |
| **no rain** | 2 / 70 | 68 / 70 | 70 |
| **Total** | 25 /100 | 75 /100 | 100 |

Likelihood: 23 / 30

Prior Probability: 30/100

Marginal likelihood: 25 / 100

Posterior Probability = ((23/30)*(30/100))/(25/100) = 0.92

There is a 92% chance of rain if there are clouds in the sky.

# naïve Bayes classification

| | clouds (a1) | | wind (a2) | | humidity (a3) | | |
|---|---|---|---|---|---|---|---|
| **Likelihood** | **Yes** | **No** | **Yes** | **No** | **Yes** | **No** | **Total** |
| **rain** | 23 / 30 | 7 / 30 | 20 / 30 | 10 / 30 | 22 / 30 | 8 / 30 | 30 |
| **no rain** | 2 / 70 | 68 / 70 | 25 / 70 | 45 /70 | 10 / 70 | 60 / 70 | 70 |
| **Total** | 25 / 100 | 75 / 100 | 45 / 100 | 55 /100 | 32 / 100 | 68 / 100 | 100 |

Let us evaluate if it will rain on a day when there are clouds and it is humid but there is no wind:

$$P(rain \mid a1 \cap \neg a2 \cap a3) = \frac{P(a1 \cap \neg a2 \cap a3 \mid rain) * P(rain)}{P(a1 \cap \neg a2 \cap a3)}$$

This formula is computationally difficult to solve because of additional features. If we assume class-conditional independence, we can much easily compute probabilities.

Class-conditional independence means that events are independent so long as they are conditioned on the same class value.

# naïve Bayes classification

- Recall: P(A ∩ B) = P(A) * P(B)
- The previous formula simplifies to:

$$P(rain \mid a1 \ \cap \neg a2 \ \cap a3\ ) = \frac{P(a1 \mid rain)P(\neg a2 \mid rain)P(a3 \mid rain)P(rain)}{P(a1\ )P(\neg a2)P(a3)}$$

- overall likelihood of rain= ((23/30) * (10/30) * (22/30) * (30/100)) = 0.056
- overall likelihood of no rain = ((2/70) * (45/70) * (10/70) * (70/100)) = 0.002

# naïve Bayes classification

- overall likelihood of rain=
  ((23/30) * (10/30) * (22/30) * (30/100)) = 0.056
- overall likelihood of no rain =
  ((2/70) * (45/70) * (10/70) * (70/100)) = 0.002

- Probability of rain =
  likelihood of rain/(likelihood of rain + no rain)
  = 0.056/(0.056+0.002) = 0.96

- Probability of no rain =
  likelihood of no rain/(likelihood of rain + no rain)
  = 0.002/(0.056+0.002) = 0.04

# Laplace Estimator

| | clouds (a1) | | wind (a2) | | humidity (a3) | | |
|---|---|---|---|---|---|---|---|
| **Likelihood** | **Yes** | **No** | **Yes** | **No** | **Yes** | **No** | **Total** |
| **rain** | 23 / 30 | 7 / 30 | 0 / 30 | 30 / 30 | 22 / 30 | 8 / 30 | 30 |
| **no rain** | 2 / 70 | 68 / 70 | 30 / 70 | 45 /70 | 10 / 70 | 60 / 70 | 70 |
| **Total** | 25 / 100 | 75 / 100 | 45 / 100 | 55 /100 | 32 / 100 | 68 / 100 | 100 |

overall likelihood of rain=

overall likelihood of no rain =

Probability of rain = 0/(0+0.001) =

Probability of no rain = 0.001/(0+0.001) =

# Laplace Estimator

| | clouds (a1) | | wind (a2) | | humidity (a3) | | |
|---|---|---|---|---|---|---|---|
| **Likelihood** | **Yes** | **No** | **Yes** | **No** | **Yes** | **No** | **Total** |
| **rain** | 23 / 30 | 7 / 30 | 0 / 30 | 30 / 30 | 22 / 30 | 8 / 30 | 30 |
| **no rain** | 2 / 70 | 68 / 70 | 30 / 70 | 45 /70 | 10 / 70 | 60 / 70 | 70 |
| **Total** | 25 / 100 | 75 / 100 | 45 / 100 | 55 /100 | 32 / 100 | 68 / 100 | 100 |

overall likelihood of rain=
((23/30) * (0/30) * (22/30) * (30/100)) = 0

overall likelihood of no rain =
((2/70) * (30/70) * (10/70) * (70/100)) = 0.001

Probability of rain = 0/(0+0.001) = 0

Probability of no rain = 0.001/(0+0.001) = 1

# Laplace Estimator

- Laplace estimator adds a small number to each of the counts in the frequency table, which ensures that each feature has a non-zero probability of occurring within each class.

- Typically Laplace estimator is set to 1 but it can be set to any value and does not necessarily have to be same for all the features.

- overall likelihood of rain=
  ((24/33) * (1/33) * (23/33) * (30/100)) = 0.005

- overall likelihood of no rain =
  ((3/73) * (31/73) * (11/73) * (70/100)) = 0.002

- Probability of rain = 0.005/(0.005+0.002) = 0.71

- Probability of no rain = 0.002/(0.005+0.002) = 0.29

# Numeric features and naïve Bayes

- Since naïve Bayes uses frequency table for learning, each feature must be categorical.
- Numeric features do not have categories, hence naïve Bayes does not directly work directly
- Solution is to discretize data:
  - Binning
  - Cut Points
- Discretizing a numeric feature always results in a reduction of information, as the feature's original granularity is reduced to a smaller number of categories. It is important to strike a balance, since too few bins can result in important trends being obscured, while too many bins can result in small counts in the naive Bayes frequency table

# naïve Bayes: Advantages

- The advantage of Naïve Bayes is that it is relatively simple and straightforward to use.
- It is suitable when the training set is relative small, and may contain some noisy and missing data.
- Moreover, you can easily obtain the probability for a prediction.
- Requires relatively few examples for training, but also works well with very large numbers of examples

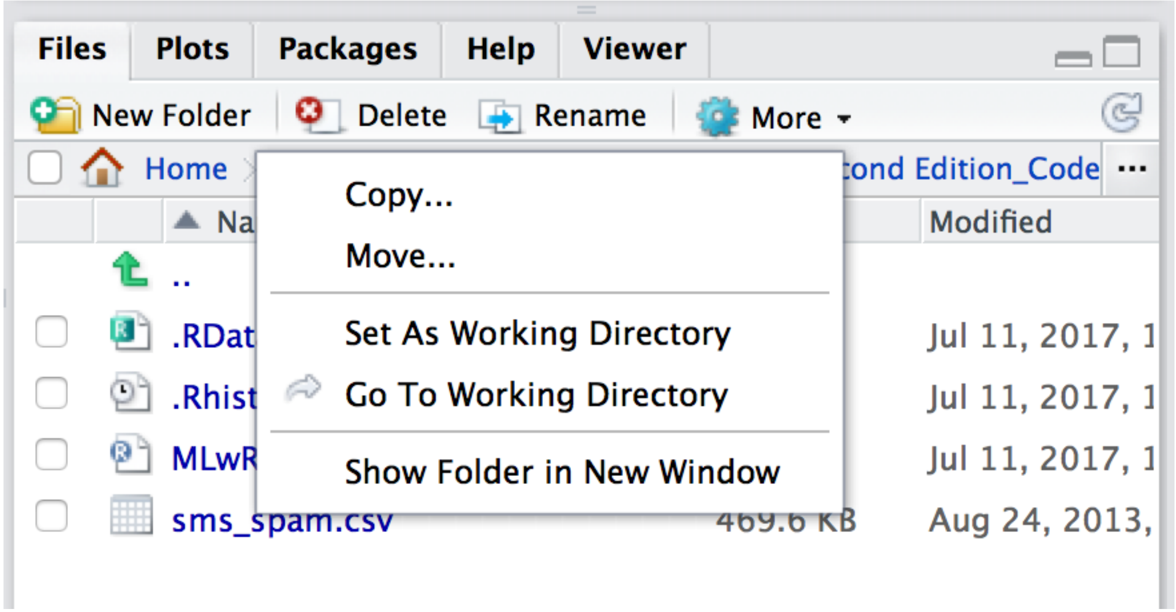
# naïve Bayes: Disadvantages

- The drawbacks of Naïve Bayes are that it assumes that all features are independent and equally important, which is very unlikely in real-world cases.
- Not ideal for datasets with large numbers of numeric features
- Estimated probabilities are less reliable than the predicted classes

# naïve Bayes: Data

- Download the `sms_spam.csv` file from the Packt Publishing's website and save it to your R working directory.

- getwd()
- setwd(dir)



# EXPLORING AND PREPARING THE DATA

# Text Mining: tm

install.packages(“tm”)
library(tm)

Corpus: A collection of text documents. Main structure of managing documents in tm

`VCorpus`

Volatile Corpus. Creates an R object to store text documents held fully in memory.

`PCorpus`

Permanent Corpus. Creates an R object to store text documents stored outside of R in a database for example.

# Text Mining: tm

`VCorpus (x, renderControl)`

| | |
|---|---|
| `x` | Source Object. Specified to abstract input location and acquire content in a uniform way.<br>`getSources()` |
| `renderControl` | List of named components reader and language<br>`getReaders()` |

# Text Mining: tm

Build a corpus using the text mining (tm) package

```
sms_corpus <-
VCorpus(VectorSource(sms_raw$text))
```

# Text Mining: tm

Examine the SMS corpus

```
print(sms_corpus)
inspect(sms_corpus[1:3])
as.character(sms_corpus[[1]]
lapply(sms_corpus[1:3], as.character)
```

# Text Mining: tm

Clean up the SMS corpus using `tm_map ()`

Get available transformation for `tm_map` function:
`getTransformations()`

| | |
|---|---|
| `"removeNumbers"` | Remove numbers from a text document |
| `"removePunctuation"` | Remove punctuation marks from a text document |
| `"removeWords"` | Remove words from a text document. eg `stopwords()` |
| `"stemDocument"` | Stem words in a text document using Porter's stemming algorithm. |
| `"stripWhitespace"` | Strip extra whitespace from a text document. Multiple whitespace characters are collapsed to a single blank. |

Content_transformer
`"tolower"` Convert text to lowercase.

*Also check out* `gsub()`

# Text Mining: tm

Tokenization: Split data into individual components (single elements of a text string). For SMS data, tokens are words.

Creating term-document matrix, also called sparse matrix:

`DocumentTermMatrix()`

| | term1 | term2 | term3 |
|---|---|---|---|
| Doc1 | 0 | 0 | 1 |
| Doc2 | 1 | 0 | 0 |

- the columns are the union of words in our corpus
- the rows correspond to each text message
- the cells are the number of times each word is seen

`TermDocumentMatrix()`

| | Doc1 | Doc2 |
|---|---|---|
| term1 | 0 | 1 |
| term2 | 0 | 0 |
| term3 | 1 | 0 |

- the rows are the union of words in our corpus
- the columns correspond to each text message
- the cells are the number of times each word is seen

# Text Mining: Data Preparation

Let us split the data 75% for training and 25% for testing.

- Split the raw data:

```
sms_train = sms_raw[1:4200, ] # about 75%
sms_test  = sms_raw[4201:5574, ] # the rest
```

- Split the document-term matrix

```
sms_dtm_train = sms_dtm[1:4200, ]
sms_dtm_test  = sms_dtm[4201:5574, ]
```

- Split the corpus

```
sms_corpus.train = sms_corpus_clean[1:4200]
sms_corpus.test  = sms_corpus_clean[4201:5574]
```

- Split raw data labels

```
sms_train_labels <- sms_raw[1:4169, ]$type
sms_test_labels  <- sms_raw[4170:5559, ]$type
```

- Subset raw data based type

```
spam <- subset(sms_raw, type == "spam")
ham  <- subset(sms_raw, type == "ham")
```

# Text Mining: Data Visualization

Word cloud is a visual way to depict frequency at which words appear in text data

```
install.packages(“worldcloud”)
library(worldcloud)

pal <-brewer.pal(12,"Paired")
```

`brewer.pal`: package `RColorBrewer.`
Creates nice looking color palettes especially for thematic maps

- Word Cloud of all data

```
wordcloud(sms_corpus_clean, min.freq = 50, random.order = FALSE, colours = pal)
```

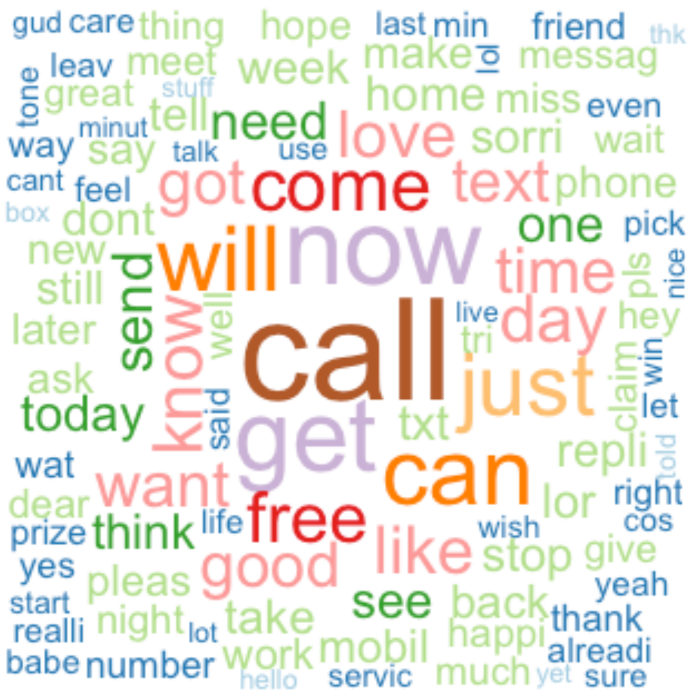
- Word Cloud of spam data

```
wordcloud(spam$text, max.words = 40, scale = c(3, 0.5), random.order = FALSE,
colors = pal)
```

- Word Cloud of ham data

```
wordcloud(ham$text, max.words = 40, scale = c(3, 0.5), random.order = FALSE,
colors = pal)
```

# Text Mining: Data Visualization




All data

Spam


Ham

# Text Mining: Frequent Terms

```
sms_dtm_freq_train <- removeSparseTerms(sms_dtm_train, 0.999)
sms_dtm_freq_train

# indicator features for frequent words
findFreqTerms(sms_dtm_train, 5)

# save frequently-appearing terms to a character vector
sms_freq_words <- findFreqTerms(sms_dtm_train, 5)
str(sms_freq_words)

# create DTMs with only the frequent terms
sms_dtm_freq_train <- sms_dtm_train[ , sms_freq_words]
sms_dtm_freq_test <- sms_dtm_test[ , sms_freq_words]

# convert counts to a factor
convert_counts <- function(x) {
  x <- ifelse(x > 0, "Yes", "No”)}

# apply() convert_counts() to columns of train/test data
sms_train <- apply(sms_dtm_freq_train, MARGIN = 2, convert_counts)
sms_test  <- apply(sms_dtm_freq_test, MARGIN = 2, convert_counts)
```

# TRAINING A MODEL ON THE DATA

# Training a model on the data

- `library(e1071)`

- `sms_classifier <- naiveBayes(sms_train, sms_train_labels)`

# EVALUATING MODEL PERFORMANCE

# Evaluating model performance

- `sms_test_pred <- predict(sms_classifier, sms_test)`

- `library(gmodels)`
- `CrossTable(sms_test_pred, sms_test_labels,`
```
prop.chisq = FALSE, prop.t = FALSE,
prop.r = FALSE, dnn = c('predicted',
'actual'))
```

# IMPROVING MODEL PERFORMANCE

# Improving model performance

- `sms_classifier2 <- naiveBayes(sms_train, sms_train_labels, laplace = 1)`
- `sms_test_pred2 <- predict(sms_classifier2, sms_test)`
- `CrossTable(sms_test_pred2, sms_test_labels,`

```
            prop.chisq = FALSE, prop.t = FALSE,
            prop.r = FALSE,
            dnn = c('predicted', 'actual'))
```

# Text Mining: Reading Assignment

Documentation on tm package:

https://cran.r-project.org/web/packages/tm/vignettes/tm.pdf

Tidy in an alternate to tm:

http://tidytextmining.com/tidytext.html

RStudio cheatsheets:

https://www.rstudio.com/resources/cheatsheets/

# Coding Challenge

Build a naïve Bayes model for Titanic dataset to predict if a passenger Survived or not.

Build a naïve Bayes model for HairEyeColor dataset to predict the sex based on hair and eye color.

Build a naïve Bayes model for HouseVote84 dataset in ‘mlbench’ package to predict the Class of the representative based on votes.