A Practical Introduction to Machine Learning in Python

Day 4 - Thursday

»Supervised Machine Learning«

Damian Trilling
Anne Kroon

d.c.trilling@uva.nl, @damian0604
a.c.kroon@uva.nl, @annekroon

Gesis

March 12, 2020

Today

1. Recap: Types of Automated Content Analysis
2. Supervised Machine Learning
   - You have done it before!
   - Applications
   - An implementation
3. Vectorizers
4. Different models
5. Alternatives to train/test split
   - Train/validation/test split
   - Cross-validation
6. Finding the optimal (hyper-)parameters
   - Hyperparameter optimization with grid search
   - Tuning decision thresholds with ROC curves
7. From feature set to final classification
   - Putting stuff together with pipelines
   - Visualizing feature weights with ELI5

Recap: Types of Automated Content Analysis

	Methodological approach		
	Counting and Dictionary	*Supervised Machine Learning*	*Unsupervised Machine Learning*
Typical research interests and content features	visibility analysis sentiment analysis subjectivity analysis	frames topics gender bias	frames topics
Common statistical procedures	string comparisons counting	support vector machines naive Bayes	principal component analysis cluster analysis latent dirichlet allocation semantic network analysis

Some terminology

Supervised machine learning

You have a dataset with both predictor and outcome (independent and dependent variables; features and labels) — a *labeled* dataset. Think of regression: You measured `x1, x2, x3` and you want to predict `y`, which you also measured

Unsupervised machine learning

You have no labels. (You did not measure `y`)

Again, you already know some techniques to find out how `x1, x2,...x_i` co-occur from other courses:

- Principal Component Analysis (PCA)
- Cluster analysis
- ...

Some terminology

Supervised machine learning

You have a dataset with both predictor and outcome (independent and dependent variables; features and labels) — a *labeled* dataset. Think of regression: You measured `x1`, `x2`, `x3` and you want to predict `y`, which you also measured

Unsupervised machine learning

You have no labels. (You did not measure `y`)

Again, you already know some techniques to find out how `x1`, `x2`,... `x_i` co-occur from other courses:

- Principal Component Analysis (PCA)
- Cluster analysis
- ...

Some terminology

Supervised machine learning

You have a dataset with both predictor and outcome (independent and dependent variables; features and labels) — a *labeled* dataset. Think of regression: You measured x1, x2, x3 and you want to predict y, which you also measured

Unsupervised machine learning

You have no labels. (You did not measure y)
Again, you already know some techniques to find out how x1, x2,...x_i co-occur from other courses:

- Principal Component Analysis (PCA)
- Cluster analysis
- ...

Some terminology

Supervised machine learning

You have a dataset with both predictor and outcome (independent and dependent variables; features and labels) — a *labeled* dataset. Think of regression: You measured x1, x2, x3 and you want to predict y, which you also measured

Unsupervised machine learning

You have no labels. (You did not measure y)

Again, you already know some techniques to find out how x1, x2,...x_i co-occur from other courses:

- Principal Component Analysis (PCA)
- Cluster analysis
- ...

Some terminology

Supervised machine learning

You have a dataset with both predictor and outcome (independent and dependent variables; features and labels) — a *labeled* dataset. Think of regression: You measured x1, x2, x3 and you want to predict y, which you also measured

Unsupervised machine learning

You have no labels. (You did not measure y)

Again, you already know some techniques to find out how `x1`, `x2,...x_i` co-occur from other courses:

- Principal Component Analysis (PCA)
- Cluster analysis
- ...

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
   $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset:*
   $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$
   $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week: $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*: $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$ $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week: $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*: $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$ $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week: $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*: $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$ $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
$y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
$\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$
$\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week: $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
   $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$
   $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

This is
Supervised Machine Learning!

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding ("labeled data","annotated data" in SML-lingo)
  - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables ("features")
- Typically, IVs are word frequencies (often weighted, e.g. tf×idf) (⇒BOW-representation)

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding ("labeled data","annotated data" in SML-lingo)
  - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables ("features")
- Typically, IVs are word frequencies (often weighted, e.g. tf×idf) (⇒BOW-representation)

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding (“labeled data”,“annotated data” in SML-lingo)
  - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables (“features”)
- Typically, IVs are word frequencies (often weighted, e.g. tf×idf) (⇒BOW-representation)

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding ("labeled data","annotated data" in SML-lingo)
  - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables ("features")
- Typically, IVs are word frequencies (often weighted, e.g. tf×idf) (⇒BOW-representation)

Applications

In other fields

A lot of different applications

- from recognizing hand-written characters to recommendation systems

In our field

It starts to get popular to measure latent variables

- frames
- topics

Applications

In other fields

A lot of different applications

- from recognizing hand-written characters to recommendation systems

In our field

It starts to get popular to measure latent variables

- frames
- topics

Applications

In other fields

A lot of different applications

- from recognizing hand-written characters to recommendation systems

In our field

It starts to get popular to measure latent variables

- frames
- topics

SML to code frames and topics

Some work by Burscher and colleagues

- Humans can code generic frames (human-interest, economic, . . .)
- Humans can code topics from a pre-defined list
- **But it is very hard to formulate an explicit rule**
  (as in: code as 'Human Interest' if regular expression R is matched)

⇒ This is where you need supervised machine learning!

Burscher, B., Odijk, D., Vliegenthart, R., De Rijke, M., & De Vreese, C. H. (2014). Teaching the computer to code frames in news: Comparing two supervised machine learning approaches to frame analysis. *Communication Methods and Measures, 8*(3), 190–206. doi:10.1080/19312458.2014.937527
Burscher, B., Vliegenthart, R., & De Vreese, C. H. (2015). Using supervised machine learning to code policy issues: Can classifiers generalize across contexts? *Annals of the American Academy of Political and Social Science, 659*(1), 122–131.

SML to code frames and topics

Some work by Burscher and colleagues

- Humans can code generic frames (human-interest, economic, . . .)
- Humans can code topics from a pre-defined list
- **But it is very hard to formulate an explicit rule**
  (as in: code as 'Human Interest' if regular expression R is matched)

⇒ This is where you need supervised machine learning!

Burscher, B., Odijk, D., Vliegenthart, R., De Rijke, M., & De Vreese, C. H. (2014). Teaching the computer to code frames in news: Comparing two supervised machine learning approaches to frame analysis. *Communication Methods and Measures, 8*(3), 190–206. doi:10.1080/19312458.2014.937527

Burscher, B., Vliegenthart, R., & De Vreese, C. H. (2015). Using supervised machine learning to code policy issues: Can classifiers generalize across contexts? *Annals of the American Academy of Political and Social Science, 659*(1), 122–131.

SML to code frames and topics

Some work by Burscher and colleagues

- Humans can code generic frames (human-interest, economic, ...)
- Humans can code topics from a pre-defined list
- **But it is very hard to formulate an explicit rule** (as in: code as 'Human Interest' if regular expression R is matched)

⇒ This is where you need supervised machine learning!

Burscher, B., Odijk, D., Vliegenthart, R., De Rijke, M., & De Vreese, C. H. (2014). Teaching the computer to code frames in news: Comparing two supervised machine learning approaches to frame analysis. *Communication Methods and Measures, 8*(3), 190–206. doi:10.1080/19312458.2014.937527
Burscher, B., Vliegenthart, R., & De Vreese, C. H. (2015). Using supervised machine learning to code policy issues: Can classifiers generalize across contexts? *Annals of the American Academy of Political and Social Science, 659*(1), 122–131.

TABLE 4
Classification Accuracy of Frames in Sources Outside the Training Set

	VK/NRC →*Tel*	*VK/TEL* →*NRC*	*NRC/TEL* →*VK*
Conflict	.69	.74	.75
Economic Cons.	.88	.86	.86
Human Interest	.69	.71	.67
Morality	.97	.90	.89

Note. VK = Volkskrant, NRC = NRC/Handelsblad, TEL = Telegraaf

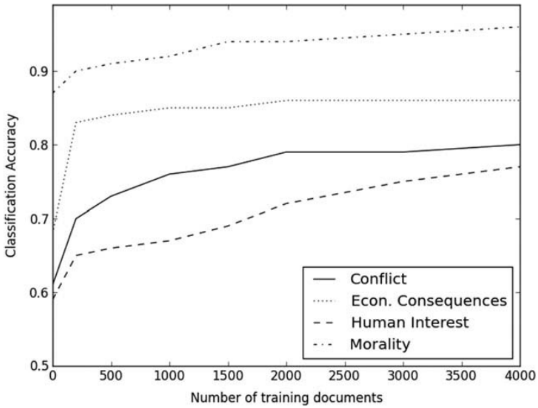


FIGURE 1 Relationship between classification accuracy and number of training documents.

FIGURE 1
Learning Curves for the Classification of News Articles and PQs

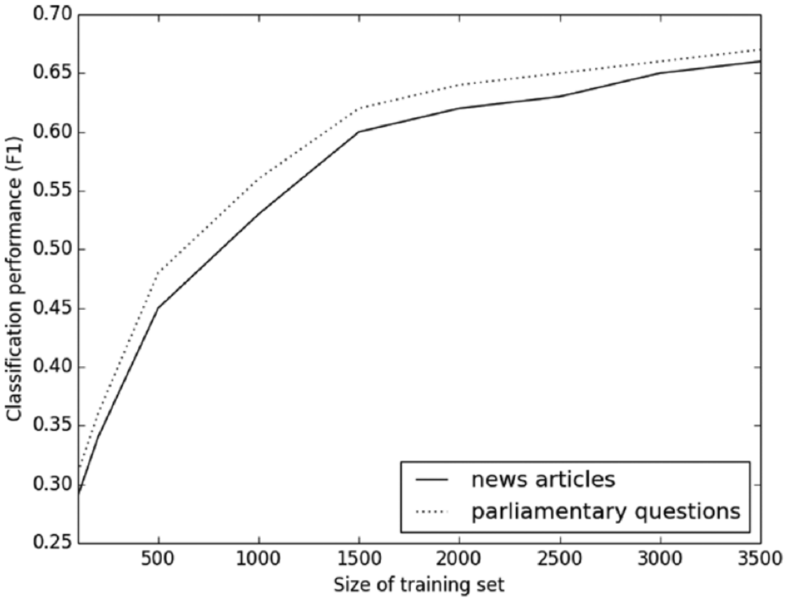


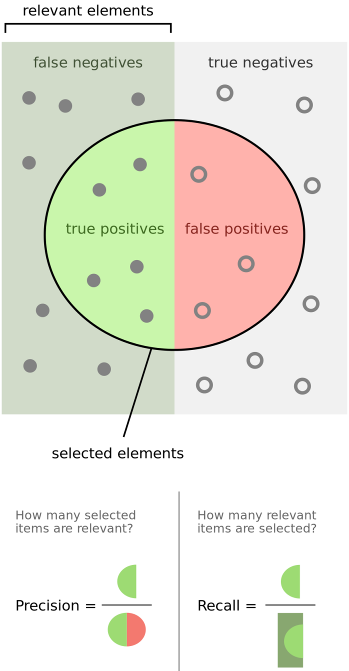
TABLE 1

F1 Scores for SML-Based Issue Coding in News Articles and PQs

Issue	News Articles			PQs	
		All Words	Lead Only		All Words
Features	*N*	F1	F1	*N*	F1
Macroeconomics	413	.54	.63	172	.46
Civil rights and minority issues	327	.34	.28	192	.53
Health	444	.70	.71	520	.81
Agriculture	114	.72	.76	159	.66
Labor and employment	217	.43	.49	174	.58
Education	188	.79	.71	229	.78
Environment	152	.34	.44	237	.59
Energy	81	.35	.59	67	.66
Immigration and integration	150	.50	.57	239	.78
Transportation	416	.58	.67	306	.81
Law and crime	1198	.70	.69	685	.77
Social welfare	115	.33	.34	214	.54
Community development and housing	113	.45	.44	136	.72
Banking, finance, and commerce	622	.62	.67	188	.58
Defense	393	.59	.55	196	.71
Science, technology, and communication	426	.64	.59	57	.53
International affairs and foreign aid	1,106	.70	.64	352	..65
Government operations	1,301	.71	.72	276	.48
Other issue	3,322	.84	.80	360	.51
Total	11,089	.71	.68	4,759	.69

NOTE: The F1 score is equal to the harmonic mean of recall and precision. Recall is the fraction of relevant documents that are retrieved, and precision is the fraction of retrieved documents that are relevant.

http://commons.wikimedia.org/wiki/File:Precisionrecall.svg



Some measures of accuracy

- Recall
- Precision
- $\mathrm{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$
- AUC (Area under curve) $[0, 1]$, $0.5 =$ random guessing

What does this mean for our research?

It we have 2,000 documents with manually coded frames and topics. . .

- we can use them to train a SML classifier
- which can code an unlimited number of new documents
- with an acceptable accuracy

Some easier tasks even need only 500 training documents, see Hopkins, D. J., & King, G. (2010). A method of automated nonparametric content analysis for social science. *American Journal of Political Science, 54*(1), 229–247.

What does this mean for our research?

It we have 2,000 documents with manually coded frames and topics...

- we can use them to train a SML classifier
- which can code an unlimited number of new documents
- with an acceptable accuracy

Some easier tasks even need only 500 training documents, see Hopkins, D. J., & King, G. (2010). A method of automated nonparametric content analysis for social science. *American Journal of Political Science, 54*(1), 229–247.

An implementation

Let's say we have a list of tuples with movie reviews and their rating:

```
reviews=[("This is a great movie",1),("Bad movie",-1), ... ...]
```

And a second list with an identical structure:

```
test=[("Not that good",-1),("Nice film",1), ... ...]
```

Both are drawn from the same population, it is pure chance whether a specific review is on the one list or the other.

Based on an example from http://blog.dataquest.io/blog/naive-bayes-movies/

Training a A Naïve Bayes Classifier

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn import metrics

# This is just an efficient way of computing word counts
vectorizer = CountVectorizer(stop_words='english')
train_features = vectorizer.fit_transform([r[0] for r in reviews])
test_features = vectorizer.transform([r[0] for r in test])

# Fit a naive bayes model to the training data.
nb = MultinomialNB()
nb.fit(train_features, [r[1] for r in reviews])

# Now we can use the model to predict classifications for our test
    features.
predictions = nb.predict(test_features)
actual=[r[1] for r in test]

print("Precision: {0}".format(metrics.precision_score(actual,
    predictions, pos_label=1, labels = [-1,1])))
print("Recall: {0}".format(metrics.recall_score(actual, predictions,
    pos_label=1, labels = [-1,1])))
```

And it works!

Using 50,000 IMDB movies that are classified as either negative or positive,

- I created a list with 25,000 training tuples and another one with 25,000 test tuples and
- trained a classifier
- that achieved an AUC of .82.

Dataset obtained from `http://ai.stanford.edu/~amaas/data/sentiment`, Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. *49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*

Playing around with new data

```
newdata=vectorizer.transform(["What a crappy movie! It sucks!", "This is
     awsome. I liked this movie a lot, fantastic actors","I would not
    recomment it to anyone.", "Enjoyed it a lot"])
predictions = nb.predict(newdata)
print(predictions)
```

This returns, as you would expect and hope:

```
[-1  1 -1  1]
```

But we can do even better

We can use different vectorizers and different classifiers.

Vectorizers

Different vectorizers

1. CountVectorizer (=simple word counts)
2. TfidfVectorizer (word counts ("term frequency") weighted by number of documents in which the word occurs at all ("inverse document frequency"))

$$tfidf_{t,d} = tf_{t,d} \cdot idf_t$$

There are different ways to weigh the idf score. A common one is taking the logarithm:

$$idf_t = \log \frac{N}{n_t}$$

where N is the total number of documents and n_t is the number of documents containing term t

Different vectorizers

1. CountVectorizer (=simple word counts)
2. TfidfVectorizer (word counts ("term frequency") weighted by number of documents in which the word occurs at all ("inverse document frequency"))

$$tfidf_{t,d} = tf_{t,d} \cdot idf_t$$

There are different ways to weigh the idf score. A common one is taking the logarithm:

$$idf_t = \log \frac{N}{n_t}$$

where N is the total number of documents and n_t is the number of documents containing term t

Different vectorizers

1. CountVectorizer (=simple word counts)
2. TfidfVectorizer (word counts ("term frequency") weighted by number of documents in which the word occurs at all ("inverse document frequency"))

$$tfidf_{t,d} = tf_{t,d} \cdot idf_t$$

There are different ways to weigh the idf score. A common one is taking the logarithm:

$$idf_t = \log \frac{N}{n_t}$$

where N is the total number of documents and n_t is the number of documents containing term t

Different vectorizers

1. CountVectorizer (=simple word counts)
2. TfidfVectorizer (word counts ("term frequency") weighted by number of documents in which the word occurs at all ("inverse document frequency"))

$$tfidf_{t,d} = tf_{t,d} \cdot idf_t$$

There are different ways to weigh the idf score. A common one is taking the logarithm:

$$idf_t = \log \frac{N}{n_t}$$

where N is the total number of documents and n_t is the number of documents containing term t

Different vectorizer options

- Preprocessing (e.g., stopword removal)
- Remove words below a specific threshold ("occurring in less than $n = 5$ documents") $\Rightarrow$ spelling mistakes etc.
- Remove words above a specific threshold ("occuring in more than 50% of all documents) $\Rightarrow$ de-facto stopwords
- Not only to improve prediction, but also performance (can reduce number of features by a huge amount)

Models (Classifiers)

(When we want to predict a binary outcome, we often refer to this as a *classification problem*, while we often call predicting a continous outcome a *regression problem*.)

Different classifiers

- Naïve Bayes
- Logistic Regression
- Support Vector Machine (SVM)
- ...

Typical approach: Find out which setup performs best (see example source code in the book).

Naïve Bayes

Bayes' theorem

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

A = Text is about sports
B = Text contains "a very good game" Furthermore, we simply multiply the propabilities for the features:

$$P(B) = P(a\ very\ close\ game) = P(a) \times P(very) \times P(close) \times P(game)$$

We can fill in all values by counting how many articles are about sports, and how often the words occur in these texts.

(Fully elaborated example on `https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/`)

Naïve Bayes

Bayes' theorem

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

A = Text is about sports
B = Text contains "a very good game" Furthermore, we simply multiply the propabilities for the features:

$$P(B) = P(a\ very\ close\ game) = P(a) \times P(very) \times P(close) \times P(game)$$

We can fill in all values by counting how many articles are about sports, and how often the words occur in these texts.

(Fully elaborated example on `https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/`)

Naïve Bayes

Bayes' theorem

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

A = Text is about sports
B = Text contains "a very good game" Furthermore, we simply multiply the propabilities for the features:

$$P(B) = P(a\,very\,close\,game) = P(a) \times P(very) \times P(close) \times P(game)$$

We can fill in all values by counting how many articles are about sports, and how often the words occur in these texts.

(Fully elaborated example on `https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/`)

Naïve Bayes

- It's "naïve" because the features are treated as completely independent ($\neq$ "controlling" in regression analysis)
- It's fast and easy
- It's a good *baseline* for binary classification problems

Naïve Bayes

- It's "naïve" because the features are treated as completely independent ($\neq$ "controlling" in regression analysis)
- It's fast and easy
- It's a good *baseline* for binary classification problems

Naïve Bayes

- It's "naïve" because the features are treated as completely independent ($\neq$ "controlling" in regression analysis)
- It's fast and easy
- It's a good *baseline* for binary classification problems

Logistic Regression

Probability of a binary outcome in a regression model

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n)}}$$

Just like in OLS regression, we have an intercept and regression coefficients.
We use a threshold (default: 0.5) and above, we assign the positive label ('good movie'), below, the negative label ('bad movie').

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- ...or to change the threshold to change our precision/recall-tradeoff

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- . . . or to change the threshold to change our precision/recall-tradeoff

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- . . . or to change the threshold to change our precision/recall-tradeoff

Logistic Regression

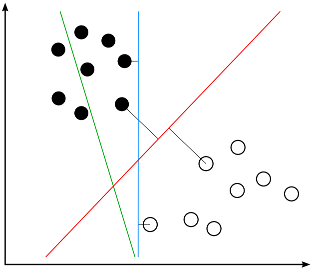
- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- ...or to change the threshold to change our precision/recall-tradeoff

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- . . . or to change the threshold to change our precision/recall-tradeoff

Support Vector Machines

- Idea: Find a hyperplane that best seperates your cases
- Can be linear, but does not have to be (depends on the so-called kernel you choose)
- Very popular

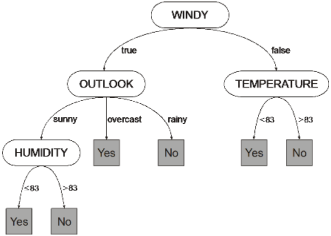


https://upload.wikimedia.org/wikipedia/commons/b/b5/Svm_separating_hyperplanes_%28SVG%29.svg

(Further reading: https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/)

Decision Trees and Random Forests

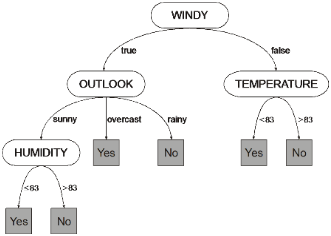
- Model problem as a series of decisions (e.g., if cloudy then ... if temperature > 30 degrees then ...)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) ("white box")



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

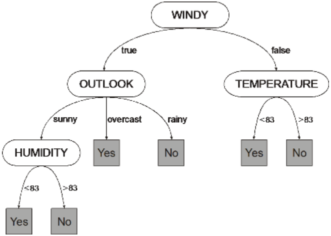
- Model problem as a series of decisions (e.g., if cloudy then ... if temperature $>$ 30 degrees then ...)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) ("white box")



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

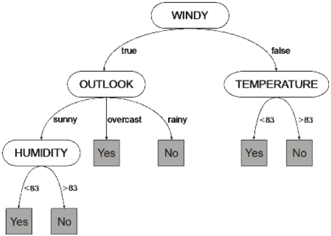
- Model problem as a series of decisions (e.g., if cloudy then . . . if temperature > 30 degrees then . . .)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) (“white box”)



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

- Model problem as a series of decisions (e.g., if cloudy then . . . if temperature > 30 degrees then . . .)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) (“white box”)



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

Disadvantages of decision trees

- comparatively inaccurate
- once you are in the wrong branch, you cannot go 'back up'
- prone to overfitting (e.g., outlier in training data may lead to completely different outcome)

Therfore, nowadays people use *random forests*: Random forests *combine* the predictions of *multiple* trees
⇒ might be a good choice for your non-linear classification problem

Decision Trees and Random Forests

Disadvantages of decision trees

- comparatively inaccurate
- once you are in the wrong branch, you cannot go 'back up'
- prone to overfitting (e.g., outlier in training data may lead to completely different outcome)

Therfore, nowadays people use *random forests*: Random forests *combine* the predictions of *multiple* trees
$\Rightarrow$ might be a good choice for your non-linear classification problem

https://scikit-learn.org/stable/supervised_learning.html