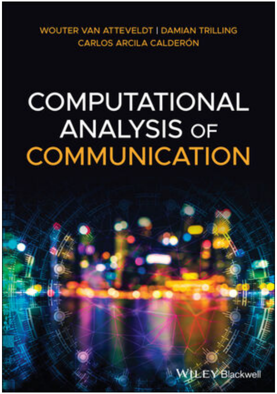
Socio-Informatics 348

Supervised Machine Learning
Modelling Overview

Dr Lisa Martin

Department of Information Science
Stellenbosch University

Today's Reading



Computational Analysis of Communication, Chapter 8

Statistical Modeling and Prediction

- Supervised Machine Learning (SML) – a form of machine learning
- SML shares many methods with classical statistical modeling.
- SML is usually applied to classification and regression problems
- Goal: Predict a variable that, for at least a part of our data, is known

Statistical Modeling and Prediction

- Example: media.csv
  - How many days per week respondents turn to different media types (radio, newspaper, tv and Internet) in order to follow the news.
  - Age (in years), gender (coded as female = 0, male = 1), and education (on a 5-point scale)
  - How far do the sociodemographic characteristics of the respondents explain their media use?
- **Statistics**: Typical aproach would be to run a regression analysis.
- **OLS**:

$$y = \beta_0 + \beta_1 \cdot \text{age} + \beta_2 \cdot \text{gender} + \beta_3 \cdot \text{education} + \epsilon$$

  where y is the number of days per week a respondent uses the Internet to follow the news.
- Also look at R^2 to see how well the model fits the data.

Statistical Modeling and Prediction

1. Fit a model using known (labeled) data (`statsmodels package`).
2. Use the fitted model to predict outcomes for new, unseen data.
3. Report and interpret: coefficients, standard errors, confidence intervals, p-values and R^2.

```
df = read.csv("https://cssbook.net/d/media.csv")
mod = lm(formula = "newspaper ~ age + gender",
         data = df)
# summary(mod) would give a lot more info,
# but we only care about the coefficients:
mod
```

```
Call:
lm(formula = "newspaper ~ age + gender", data = df)

Coefficients:
(Intercept)          age       gender
   -0.08956      0.06762      0.17666
```

Statistical Modeling and Prediction

- We can now plug in values to make predictions.

$$\hat{y} = -0.0896 + 0.0676 \cdot \text{age} + 0.1767 \cdot \text{gender}$$

```
gender = c(1,0)
age = c(20,40)
newdata = data.frame(age, gender)
predict(mod, newdata)
```

```
       1        2
1.439508 2.615248
```

Note: Prediction model can take the form of *any* function, as long as it takes some characteristics (or “features”) of the cases (in this case, people) as input and returns a prediction.

Model Limitations and Interpretation

- Predictions may be implausible (e.g., negative or >7 days/week).
- Linear models assume:
  - Linearity
  - Independence of errors
  - Homoskedasticity
- These assumptions may not hold in practice.
- Many tasks are better suited for **classification** than regression.
  - Depends on the goal of the analysis.
  - Sometimes predict outcomes, not necessarily an exact value.

Concepts and Principles of SML

- Learn from labeled data to predict outcomes for unseen data.
  - We did a simple version of this with OLS regression.
  - When do we need to use a SML approach?
- Two preconditions:
  - Large dataset
  - Random subset of data with known outcomes (labels)

Machine Learning vs. Statistical Terminology

Machine Learning	**Statistics**
Feature	Independent variable
Label	Dependent variable
Labeled dataset	Data with known outcomes
Train / fit model	Estimate model
Classifier	Model predicting nominal outcomes

Typical Supervised Learning Workflow

1. Annotate or label a subset of the data.
   - Should be *class balanced*.
2. Split the labelled data into **training** and **test** sets.
   - Common split ratios: 50:50 to 80:20.
3. Train model on training data.
4. Evaluate performance on unseen test data.

Typical Supervised Learning Workflow

Split the labelled data into training and test sets:

```
df = read.csv("https://cssbook.net/d/media.csv")
df = na.omit(df %>% mutate(
    usesinternet=recode(internet,
            .default="user", `0`="non-user")))

set.seed(42)
df$usesinternet = as.factor(df$usesinternet)
print("How many people used online news at all?")
```

```
[1] "How many people used online news at all?"
```

```
print(table(df$usesinternet))
```

```
non-user     user
     803     1262
```

Typical Supervised Learning Workflow

Split the labelled data into training and test sets:

```
split = initial_split(df, prop = .8)
traindata = training(split)
testdata  = testing(split)

X_train = select(traindata,
                 c("age", "gender", "education"))
y_train = traindata$usesinternet
X_test = select(testdata,
                c("age", "gender", "education"))
y_test = testdata$usesinternet

glue("We have {nrow(X_train)} training and {nrow(X_test)} test cases.")
```

```
We have 1652 training and 413 test cases.
```

Typical Supervised Learning Workflow

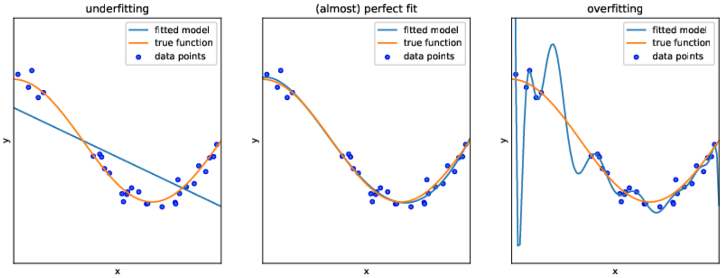
We now can train our classifier!
We estimate our model using the training dataset objects `X_train` and `y_train`:

```
myclassifier = train(x = X_train, y = y_train,
                     method = "naive_bayes")
y_pred = predict(myclassifier, newdata = X_test)
```

BUT before we can use this classifier, we need to test how capable it is to predict the correct labels, given a set of features.

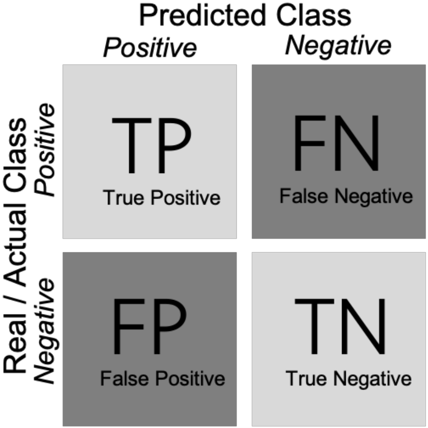
Evaluating Model Performance

- For this, we need to evaluate the model on the unseen test set.
- We could use the same input data, btu this is not strict enough.
- We don't want a model that is only good at predicting its own training data (overfitting).
- We want a model that generalizes well to unseen data.



Evaluating Model Performance

- Use a **confusion matrix** to compare predictions and true labels.



Precision and Recall Metrics

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

- **Precision**: correctness of positive predictions.
- **Recall**: completeness of positive predictions.
- Often, increasing one decreases the other (trade-off).