Features

Machine Learning with R

Basel R Bootcamp

October 2019

Feature issues

Too many features

- Curse of dimensionality
- Feature importance

Wrong features

- Feature scaling
- Feature correlation
- Feature quality

Create new features

- Feature engineering

from xkcd.com

Curse of dimensionality

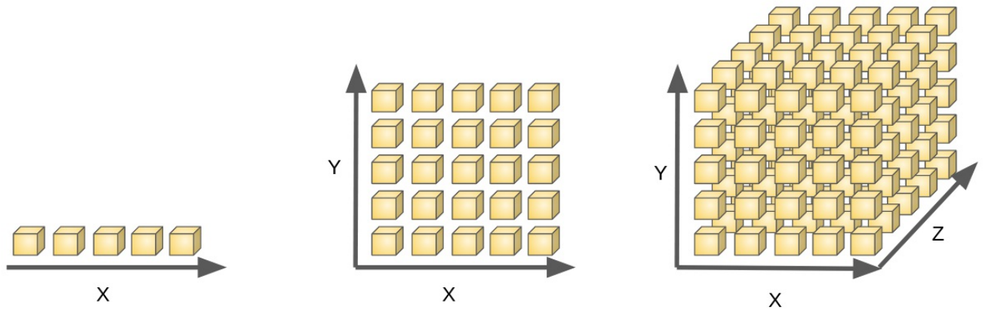
As the number of features grows...

Performance - the amount of data that needs to generalize accurately grows exponentially.

Efficiency - the amount of computations grows (how much depends on the model).

Redundancy - the amount of redundancy grows (how much depends on the model).

→ **Small set of good predictors**



from medium.freecodecamp.org

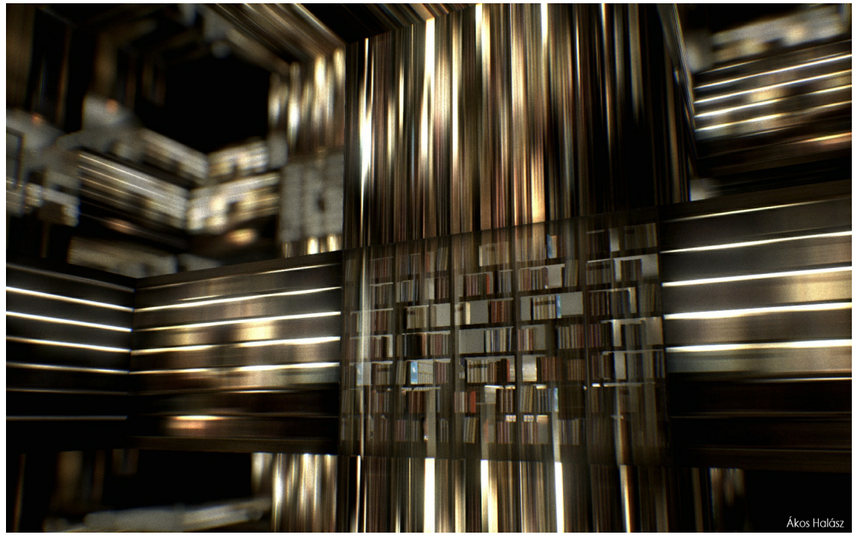
How to reduce dimensionality?

3 ways

1 - Reduce variables **manually** based on statistical or intuitive considerations.

2 - Reduce variables **automatically** using the right ML algorithms, e.g., `random forests` or `lasso regression`, or feature selection algorithms, e.g., `recursive feature selection`.

3 - Compress variables using **dimensionality reduction algorithms**, such as `principal component analysis` (PCA).



from Interstellar

Feature importance

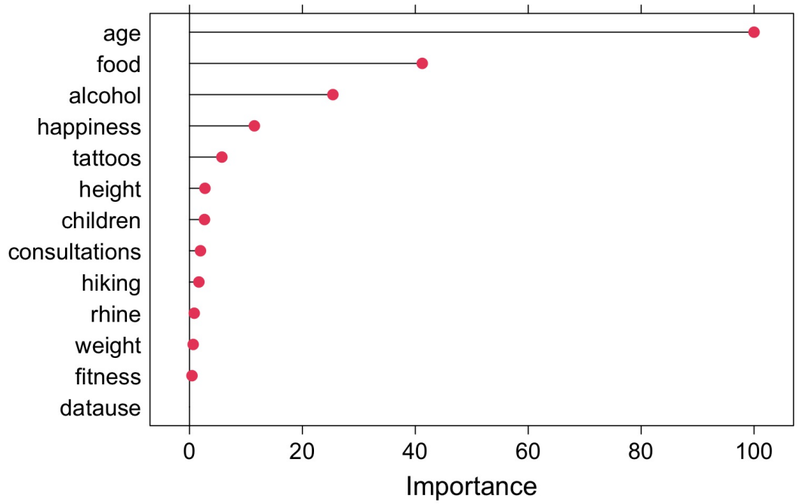
Feature importance characterizes how much a feature contributes to the fitting/prediction performance.

The metric is **model specific** , but typically **normalized** to `[0, 100]`.

Strategies

- Single variable prediction (e.g., using `LOESS`, `ROC`)
- Accuracy loss from scrambling
- `random forests` importance
- etc.

```
# plot variable importance for lm(income ~ .)
plot(varImp(income_lm))
```



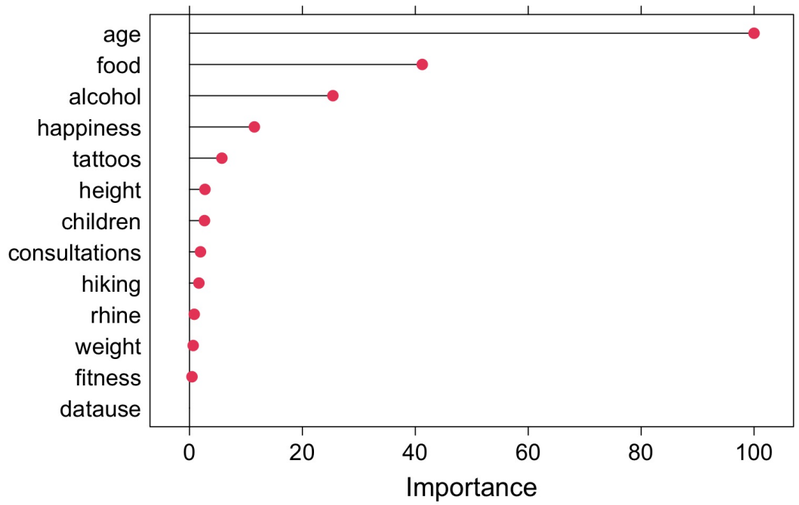
varImp()

varImp() **automatically selects appropriate measure** of variable importance for a given algorithm.

```
varImp(income_lm)
```

```
lm variable importance

           Overall
age        100.000
food        41.134
alcohol     25.247
happiness   11.582
tattoos      5.888
children     2.745
height       2.337
weight       0.983
```

```
# plot variable importance for lm(income ~ .)
plot(varImp(income_lm))
```



Recursive feature selection `rfe()`

Find the **best number of `n` predictors**, with `n` drawn from specified candidate sets `N`, e.g., `N = [2,3,5,10]`.

Algorithm

1. **Resample** and split data
2. Identify **best `n` predictors** and their prediction performance
3. **Aggregate performance** and select best `n` and the accordingly best predictors

```
# Run feature elimination
rfe(x = ..., y = ...,
    sizes = c(3,4,5,10), # feature set sizes
    rfeControl = rfeControl(functions = lmFuncs))
```

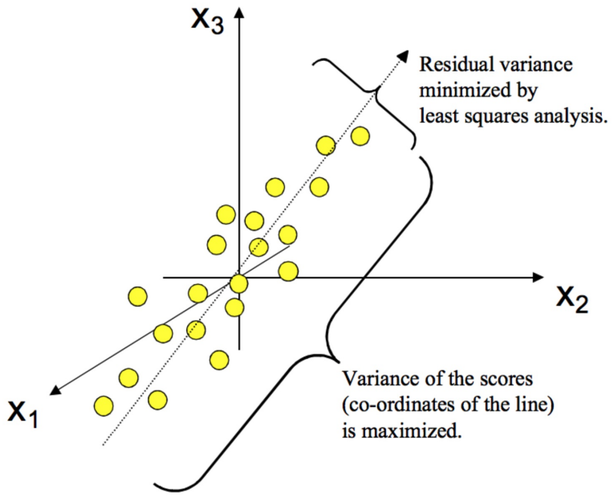
```
Recursive feature selection

Outer resampling method: Bootstrapped (25 reps)

Resampling performance over subset size:

 Variables  RMSE Rsquared   MAE RMSESD RsquaredSD   MAESD Selecte
         3 0.371    0.861 0.292 0.0133     0.0139 0.01078
         4 0.367    0.863 0.288 0.0127     0.0126 0.00992
         5 0.368    0.863 0.289 0.0122     0.0124 0.00951
        10 0.371    0.861 0.292 0.0123     0.0121 0.00922
        14 0.371    0.861 0.291 0.0122     0.0120 0.00918

The top 4 variables (out of 4):
   age, food, alcohol, happiness
```

Dimensionality reduction using PCA

The go-to algorithm for dimensionality reduction is **principal component analysis** (PCA).

PCA is an **unsupervised**, regression-based algorithm that re-represents the data in a **new feature space**.

The new features aka principal components are greedy. **Skimming off the best components** results in a small number of features that **preserve the original features** as well as possible.

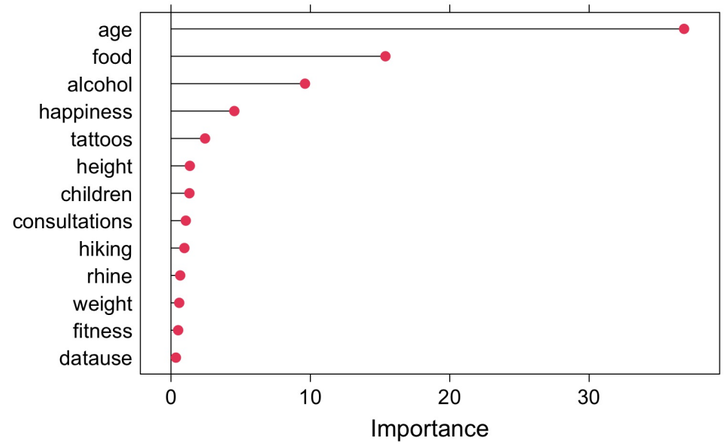


from

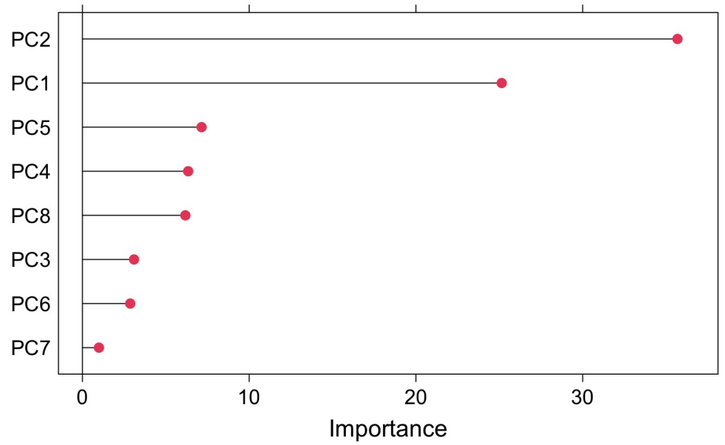
blog.umetrics.com

Using PCA

```
# train model WITHOUT PCA preprocessing
model = train(income ~ ., method = 'lm',
              data = bas_train)

plot(varImp(model))
```




```
# train model WITH PCA preprocessing
model = train(income ~ ., method = 'lm',
              data = bas_train,
              preProc = c('pca'))
plot(varImp(model))
```



Other, easy feature problems

Multi-collinearity

Multi-collinearity, **high feature correlations**, mean that there is redundancy in the data, which can lead to **less stable fits**, **uninterpretable variable importances**, and **worse predictions**.

```
# identify redundant variables
findCorrelation(cor(baselers))
```

```
[1] 5
```

```
# remove from data
remove <- findCorrelation(cor(baselers))
baselers <- baselers %>%
  select(-remove)
```

Unequal & low variance

Unequal variance **breaks regularization** (L1, L2) and renders estimates difficult to interpret.

```
# standardize and center variables
train(..., preProc("center", "scale"))
```

Low variance variables add parameters, but **can hardly contribute to prediction** and are, thus, also redundant.

```
# identify low variance variables
nearZeroVar(baselers)
```

```
integer(0)
```

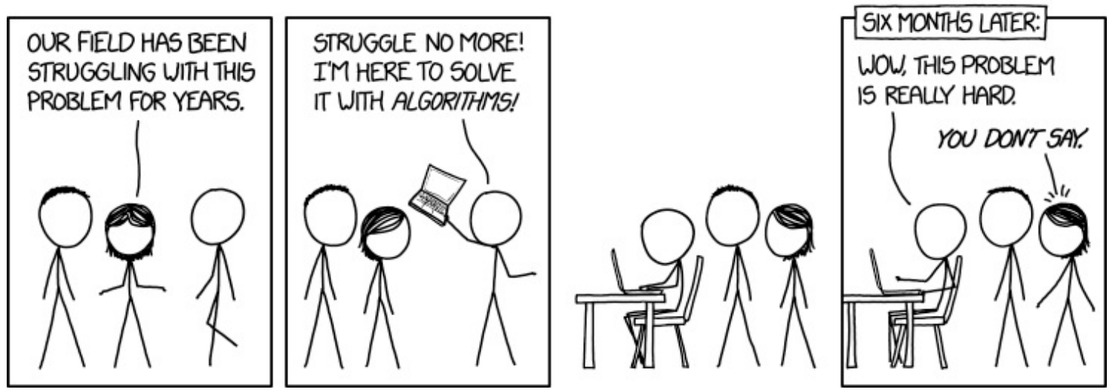
Difficult feature problems

1 - **Trivial features**

Successful prediction not necessarily implies that a meaningful pattern has been detected.

2 - **Missing features**

Some problems are hard, requiring the engineering of new features.



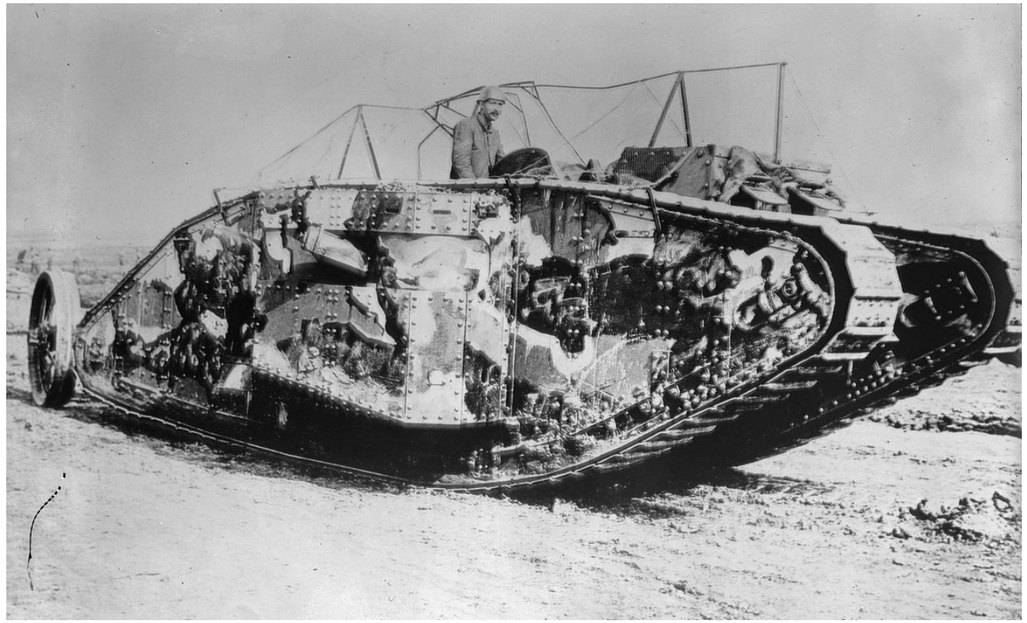
from xkcd.com

Trivial features

An urban myth?!

"The Army trained a program to differentiate American tanks from Russian tanks with 100% accuracy. Only later did analysts realize that the American tanks had been photographed on a sunny day and the Russian tanks had been photographed on a cloudy day. The computer had learned to detect brightness."

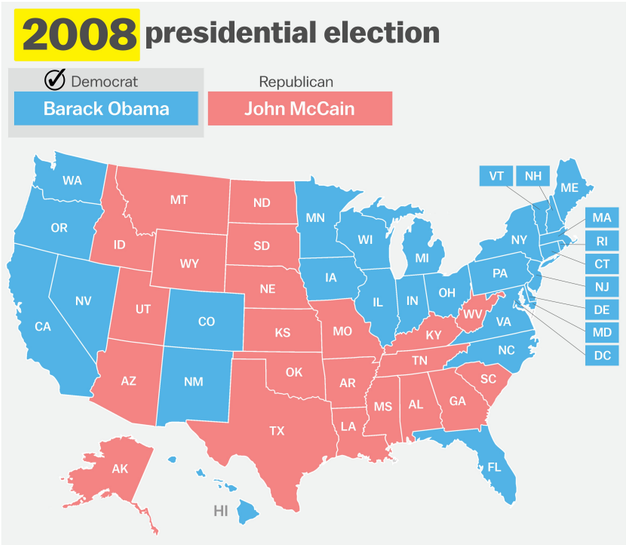
New York Times [Full text]



from wikipedia.org

Trivial features

In 2012, Nate Silver was praised to have correctly predicted the outcomes of the presidential election in 50 states after having correctly predicted 49 states in 2008. **But how much of a challenge was that?**



from vox.com

from vox.com

(Always!) missing features

"…some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used."

Pedro Domingos

"The algorithms we used are very standard for Kagglers. […] We spent most of our efforts in feature engineering. [...] We were also very careful to discard features likely to expose us to the risk of over-fitting our model."

Xavier Conort

"Coming up with features is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering."

Andrew Ng

Feature engineering

“Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.”

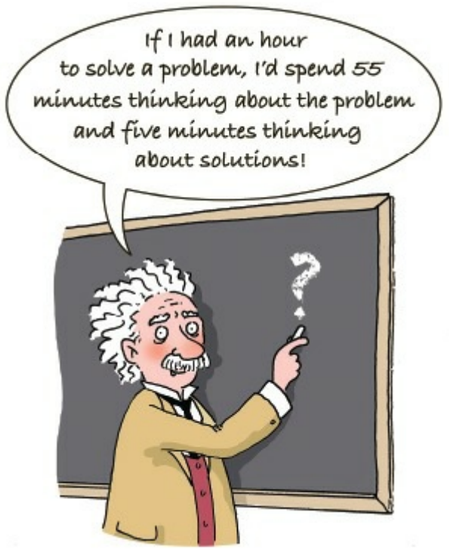
Jason Brownlee

"...while avoiding the curse of dimensionality."

duw

Feature engineering involves

- **Transformations**
- **Interactions**
- **New features**



from open.edu

Practical