Some ML Metrics

Bradley Mitchell

November 10, 2016

Purpose of this talk

- When assessing the quality of ML and predictive models to real world data there are many metrics to choose from
- There are a lot of good references on the web for these already but maybe not as much in the way of practical lessons learned
- Rather than try to survey these here, I'd like to go in depth on just a couple based on my lessons learned in "the school of hard knocks"
- Idea is not to show that a particular metric is better than others but to show a few of the ways they can be misused

What I'll cover

- Supervised learning with majority of talk on regression
- Skip formal descriptions of these metrics in favor of descriptions in code.
- No formal proofs just empirical evidence (and references at the end)

A recent experience

- Data Science competition for college students
- A regression problem was posed
- Which evaluation metric: MSE or R2?

Mean Squared Error (MSE)

- Sum the squared differences between predictions and actual values
- Then divide the result by the number of predictions
- You can take the square root to scale in the original units of the probem
- Very familiar coming from an algorithms background

Mean Squared Error (MSE)

```r
mse <- function(y,f) {mean((y-f)^2)}
```

R-squared (R2/RSQ)

- It's just the squared coefficient of correlation... (or is it?)
- Essentially unitless - falls on a range from 0 to 1
- Google sheets has a predifined function for this :-)

Actually 3 (maybe more) definitions of R-squared!

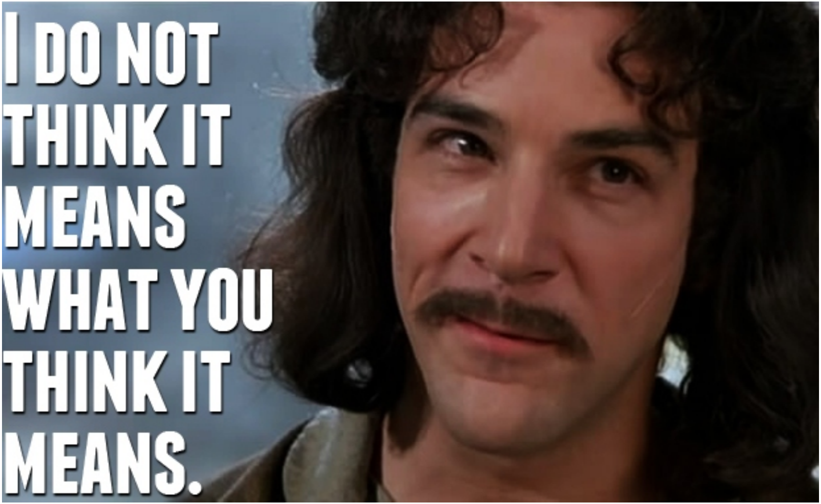


Figure 1:

The Google Sheets / MS Excel definition [1]

```r
ms_rsq <- function(y,f) { cor(y,f)^2 }
```

The Wikipedia Definition [2]

```r
# helper functions
ss_total <- function(y) {sum((y-mean(y))^2)}
ss_residual <- function(y,f) {sum((y-f)^2)}
ss_regression <- function(y,f) {sum((f-mean(y))^2)}

# the actual definition
wiki_rsq <- function(y,f) {
  1 - ss_residual(y,f) / ss_total(y)
}
```

As Fraction of Variance Explained [3]

```
frac_var_rsq <- function(y,f) {
  ss_regression(y,f) / ss_total(y)
}
```

An experiment

```
# Simulate the dependent variable
set.seed(42)
y <- rexp(100)
```

- Create a few different “models” with simulated predictions
- Create scatter plots of the predictions vs actuals
- Compare MSE and the different definitions of R squared across these models

An experiment

First, we'll get metrics for each of the models. I'll show how those models were generated later.

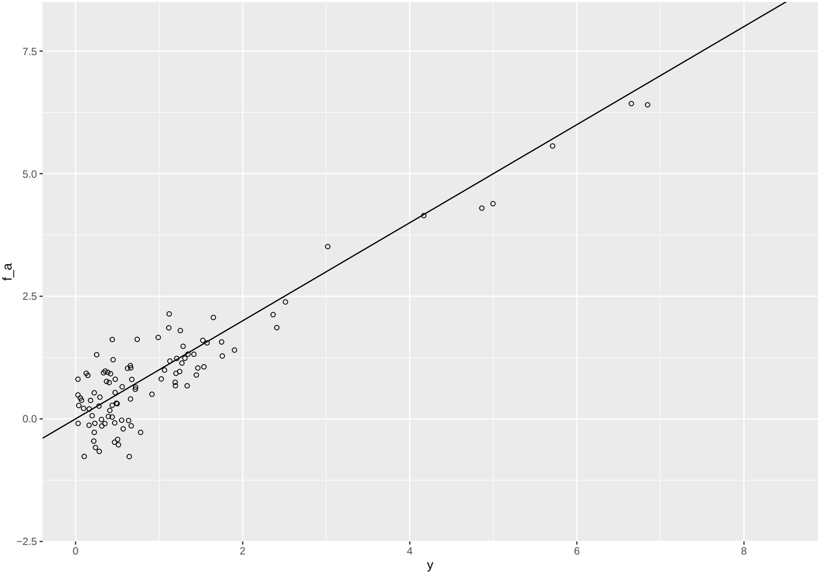
```
# Combine data from all 4 models
data <- data.frame(cbind(y, f_a, f_b, f_c, f_d))

# Get metrics for each model
ms_rsq_ <- apply(select(data, -y), 2,
                 function(x) (ms_rsq(data$y, x)))
wiki_rsq_ <- apply(select(data, -y), 2,
                   function(x) (wiki_rsq(data$y, x)))
frac_var_rsq_ <- apply(select(data, -y), 2,
                       function(x) (frac_var_rsq(data$y, x)))
mse_ <- apply(select(data, -y), 2,
              function(x) (mse(data$y, x)))

# Combine the results
results <- cbind(ms_rsq_, wiki_rsq_, frac_var_rsq_, mse_)
```

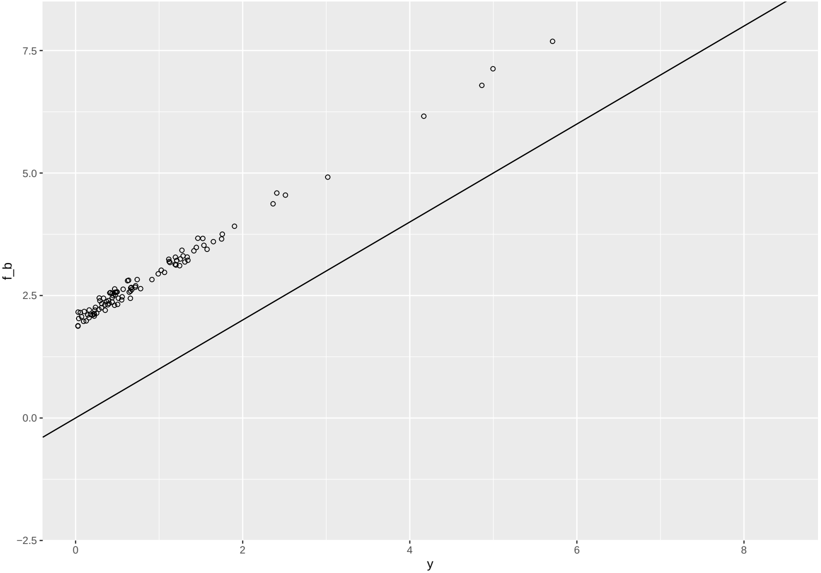
Model A

```
## Warning: Removed 1 rows containing missing values (geom_point
```



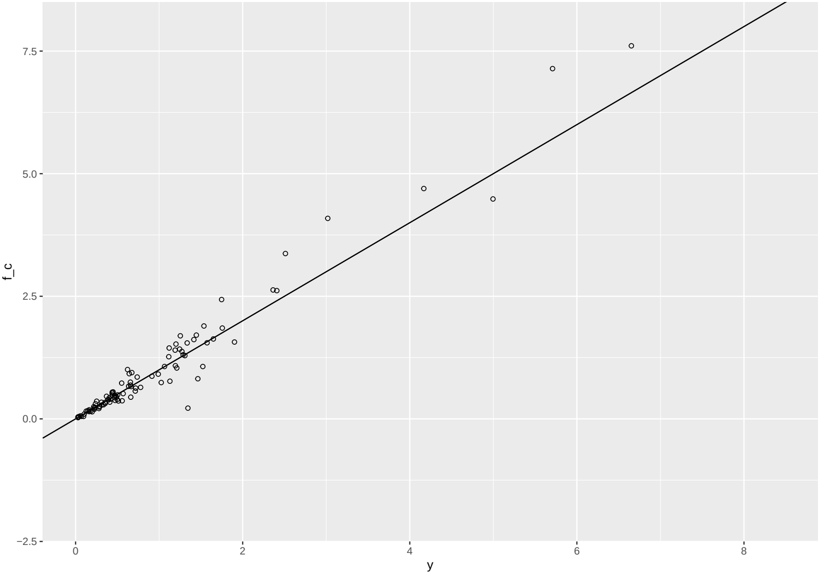
Model B

```
## Warning: Removed 3 rows containing missing values (geom_point
```



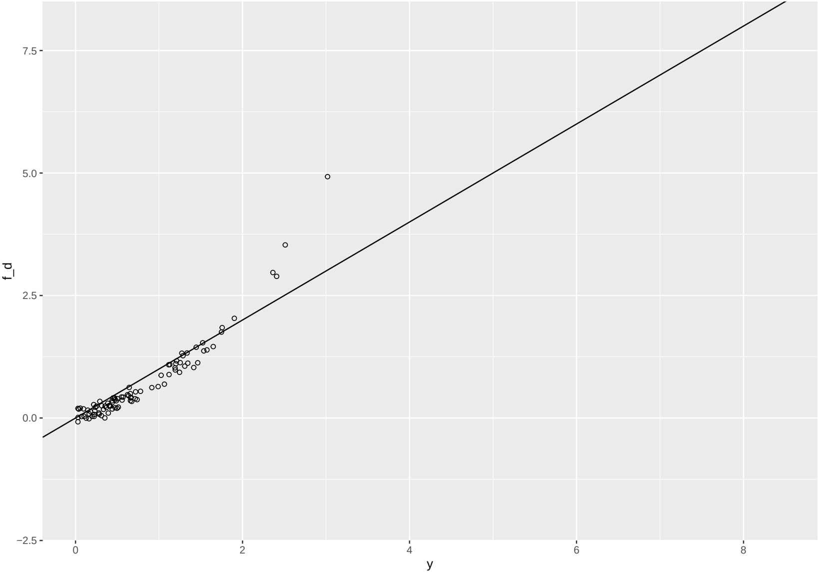
Model C

```
## Warning: Removed 3 rows containing missing values (geom_point
```



Model D

```
## Warning: Removed 7 rows containing missing values (geom_point
```



Tablulated model results

```
##        ms_rsq_    wiki_rsq_ frac_var_rsq_         mse_
## f_a 0.8860233    0.8755986      1.063970   0.2789967
## f_b 0.9954418   -0.7652786      2.757915   3.9590145
## f_c 0.9573189    0.8610941      1.637110   0.3115263
## f_d 0.6467695 -102.4575075    119.823865 232.0255775
```

Behind the scenes:

```r
f_a <- y + rnorm(100, sd=0.5)
f_b <- y + rnorm(100, sd=0.1) + 2
f_c <- y + rnorm(100, sd=0.25*y)
f_d <- 1.1^y + rnorm(100, sd=0.1)
```

Training vs Test data

It turns out that RSQ behaves differently for training vs test data.

```r
# Create a linear model for each model's predictions
lm_a <- lm(y ~ f_a)
lm_b <- lm(y ~ f_b)
lm_c <- lm(y ~ f_c)
lm_d <- lm(y ~ f_d)

# Get predictions based on training data
f_at <- predict(lm_a)
f_bt <- predict(lm_b)
f_ct <- predict(lm_c)
f_dt <- predict(lm_d)

# Combine the predictions
data_train <- data.frame(cbind(y, f_at, f_bt, f_ct, f_dt))
```

Training vs Test Data

It turns out that RSQ behaves differently for training vs test data.

```
# Evaluate the predictions
ms_rsq_t <- apply(select(data_train, -y), 2, function(x)
                              (ms_rsq(data_train$y, x)))
wiki_rsq_t <- apply(select(data_train, -y), 2, function(x)
                                (wiki_rsq(data_train$y, x)))
frac_var_rsq_t <- apply(select(data_train, -y), 2, function(x)
                                    (frac_var_rsq(data_train$y, x)))
```

Training vs Test Data

```r
cbind(ms_rsq_t, wiki_rsq_t, frac_var_rsq_t)
```

```
##       ms_rsq_t wiki_rsq_t frac_var_rsq_t
## f_at 0.8860233  0.8860233      0.8860233
## f_bt 0.9954418  0.9954418      0.9954418
## f_ct 0.9573189  0.9573189      0.9573189
## f_dt 0.6467695  0.6467695      0.6467695
```

Relationship between MSE and RSQ

```
# Calculate MSE for training data
mse_train <- apply(select(data_train, -y), 2,
                   function(x) (mse(data_train$y, x)))

# Get RSQ from MSE
rsq_from_mse <- 1 - (mse_train / var(data_train$y))
cbind(frac_var_rsq_t, rsq_from_mse)
```

```
##      frac_var_rsq_t rsq_from_mse
## f_at      0.8860233    0.8871630
## f_bt      0.9954418    0.9954873
## f_ct      0.9573189    0.9577457
## f_dt      0.6467695    0.6503018
```

Linear vs Nonlinear Models

R-squared depends on the relationship:

```
SS.Total == SS.Regression + SS.Error
```

This only holds for linear models [4]. Using R-squared to select the best non-linear model can lead to sub-optimal results!

Classification

There are many different classification metrics: e.g. accuracy, precision, recall, F1, AUC, etc.

Accuracy is the number of correctly labeled items divided by the number of all items.

Consider what happens when the classes are imbalanced.

An example ROC Curve

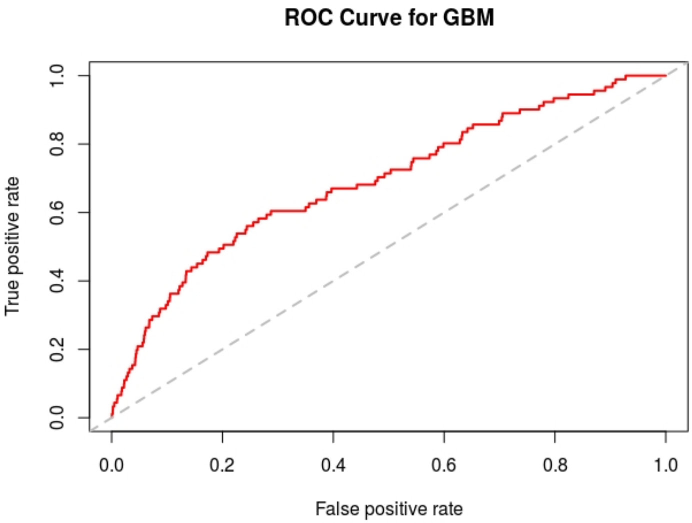


Figure 2:

Effects of Skew (imbalanced classes) on different metrics [5]

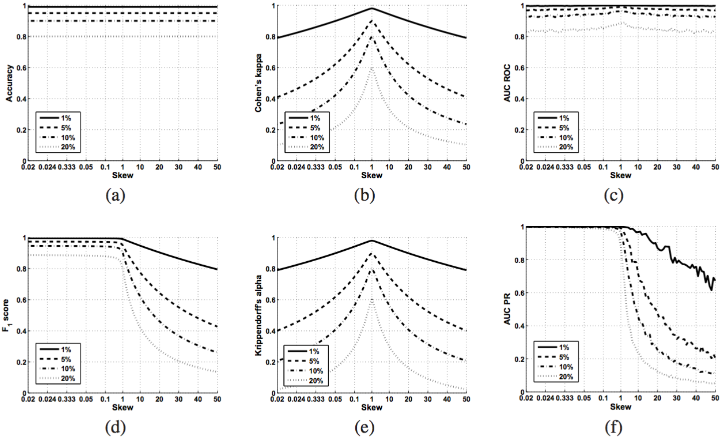


Figure 1. The behaviour of different metrics using simulated classifiers. The horizontal axis depicts the skew ratio ($Skew = \frac{Negative\ examples}{Positive\ examples}$, while the vertical axis shows the given metric score. The metrics are (a): Accuracy, (b): Cohen's kappa, (c) Area Under ROC, (d) F_1 score, (e) Krippendorff's alpha, (f) Area Under PR Curve. The different lines show the relative misclassification rates of the simulated classifiers.

Figure 3:

Conclusion

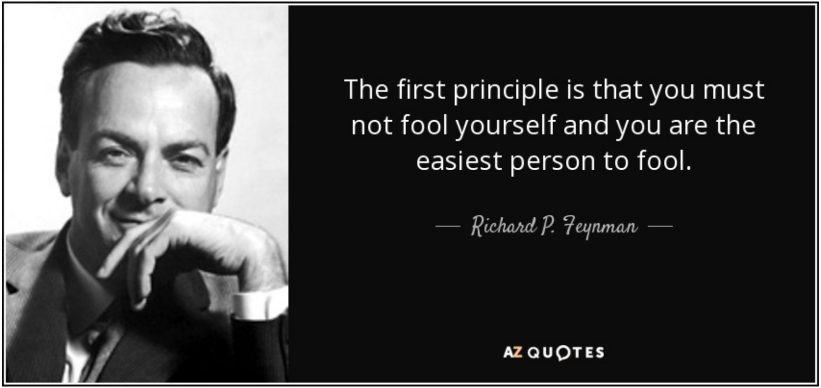


Figure 4:

References

1. Google Support Docs: "RSQ" https://goo.gl/xh3hlB
2. Wikipedia: "Coefficient of determination" https://goo.gl/aO3HkQ
3. Wikipedia: "Fraction of variance unexplained" https://goo.gl/BESswR
4. Minitab blog post: "How do I interpret R-squared?" https://goo.gl/4psDMu
5. "Facing Imbalanced Data Recommendations" Jeni, Cohn, and De La Torre https://goo.gl/H6jDEf