# Out of Sample Performance

Recall in lecture 1, we said that we need to split the data 3 times:

- A **training** set, which is used to fit the model
- A **validation** set, which is used to assess model performance
- A **test** set, which is used to… assess model performance

# Out of Sample Performance

The difference between the **validation** and **test** sets is that the validation set is used *during* the model-building procedure, to compare models.

The **test** set is kept in a vault, and only brought out at the very end. It is used to *report* performance.

If you were reporting to your boss, or publishing a paper, you would quote statsitics from the **test** set.

You would not use the **test** set for any other purpose- doing so would cause you to report overly optimistic out of sample performance.

The **validation** performance, though, is an unbiased estimate of the **test** performance.

# Continuous Targets

In the case of continuous targets, it is fairly simple to evaluate a model's test-set performance; you use the same metrics (squared error, correlation) as you would have on the training set.

We will focus our discussion on evaluating *binary classification models*, because

- This is a very common use case and
- It is actually more difficult than it sounds

# Binary Classification Models

confusion matrix: row — predicted class
col — actual class

Why is it more difficult than it sounds?

T: true
P: positive
N: negative

Define the **confusion matrix**:

| | True 1 | True 0 | |
|---|---|---|---|
| Pred 1 | TP | FP | type 1 |
| Pred 0 | FN | TN | |
| | type 2 | | |

which compares the targets to the predicted values. For the stats-oriented folks, this is just a $2 \times 2$ contingency table.

# Confusion Matrix

The acronyms are

- TP: True Positive
- TN: True Negative
- FP: False Positive
- FN: False Negative

and we have

$$N = TP + TN + FP + FN$$

$$\sum_{n=1}^{N} t_n = TP + FN$$

$$\sum_{n=1}^{N} (1 - t_n) = TN + FP$$

# Confusion Matrix

The *classification error* is then

$$Err = \frac{FP + FN}{N}$$

The *accuracy* is 1 minus the error.

# Confusion Matrix

power

The *true positive rate* is

$$TPR = \frac{TP}{TP + FN}$$

all items true

type i error

The *false positive rate* is

$$FPR = \frac{FP}{FP + TN}$$

$TPR$ estimates $P$(classifier predicts positive|true positive), and
$FPR$ estimates $P$(classifier predicts positive|true negative)

**Important**: $TPR \neq 1 - FPR$

## Confusion Matrix

Let's do an actual example. Suppose we are building a model to predict whether it will be rainy tomorrow or not. We get a year's worth of weather data for our region and code up $t_n = 0$ for sunny and $t_n = 1$ for rainy, for the $n^{th}$ day. We get the confusion matrix

| | True 1 | True 0 |
|---|---|---|
| Pred 1 | $200$ | $30$ |
| Pred 0 | $80$ | $55$ |

We can tell from this table that

$$TP =$$

$$FP =$$

$$TN =$$

$$FN =$$

# Confusion Matrix

We can tell from this table that

$$TPR = \frac{TP}{TP + FN} = \frac{200}{280} = 71\%$$
$$FPR = \frac{FP}{TN + FP} = \frac{30}{85} = 35\%$$

From the table, we can make the following statements:

- It rained on $200 + 80 = 280$ days in the observed data (maybe this study was done in Vancouver)
- We predicted $200$ of these rainy days correctly, and $80$ incorrectly; our model captured $200/280 = 71\%$ of the rainy days
- We predicted $30$ out of $85$ sunny days to be rainy, or $30/85 = 35\%$

## Confusion Matrix

Our model had a misclassification rate of $(80 + 30)/365 = 42\%$, so we are pretty confident it's not a good model. Why bother with those other statistics?

Consider now a similar binary classification problem: we want to predict default in the next 12 months for the customers in our credit card portfolio at the bank. We model $N = 10,000$ customers, and observe the following confusion matrix:

|  | True 1 | True 0 |
|---|---|---|
| Pred 1 | $3$ | $150$ |
| Pred 0 | $7$ | $9,840$ |

t=1 means default

The misclassifcation rate is only $157/10,000 = 1.57\%$. The model is awesome!

## Unbalanced Data

The model is not awesome. The default rate in this portfolio is only $10/10,000 = 0.1\%$.

If we just predicted $\hat{t}_n = 0$ for every customer, we'd get the following contingency table:

| | True 1 | True 0 |
|---|---|---|
| Pred 1 | $0$ | $0$ |
| Pred 0 | $10$ | $9,990$ |

0.001 = 10 / 10000

The model is predicting $1.57/0.1 = 15.7$ times worse than what we would get if we just predicted $\hat{t}_n = 0$ for every observation, *in terms of misclassification rate.*

## Unbalanced Data

The TPR and FPR for these tables tell a different story. For model 1,

$$TPR = \frac{3}{3+7} = 30\%$$
$$FPR = \frac{150}{150+9,840} = 1.5\%$$

where for model 2,

$$TPR = \frac{0}{10} = 0\%$$
$$FPR = \frac{0}{9,990} = 0\%$$

# Unbalanced Data

This is why TPR and FPR are important- they allow for the tuning of the model to produce the right *balance* of misclassifications.

In this example, since the cost of the lost business ($150$ good customers predicted as bad) is probably less than the cost of the true predicted defaults, the model might not be that bad. In fact for business purposes, you'd probably have actual estimates of each dollar cost, which you would use directly.

Confusion matrices have another major use in model validation.

## Soft Classification

Most of the models we learn about don't directly give hard 0/1 classifications- they give predicted probabilities, called **soft** classifications.

Before, we translated these into hard classifications by assigning

$$\hat{t}_{hard} = \begin{cases} 1 \text{ if } \hat{t} > 0.5 \\ 0 \text{ if } \hat{t} < 0.5 \end{cases}$$

There is no real reason why using $0.5$ would be optimal in general, though it is intuitive.

We turn to the question: for a given classification model, is it possible to pick a cutoff that gives us good hard classifications?

# Thresholds

The key observation is that each choice of the threshold, $0 < h < 1$, generates a contingency table.

And therefore, gives distinct $Err, TPR, FPR$.

We can choose the threshold to give the balance that we want.

We can also use this idea to validate the model itself, by asking "is it even possible to pick a good $h$?"

# The KS Statistic

The first way we can use this idea is to look at the **KS** statistic.

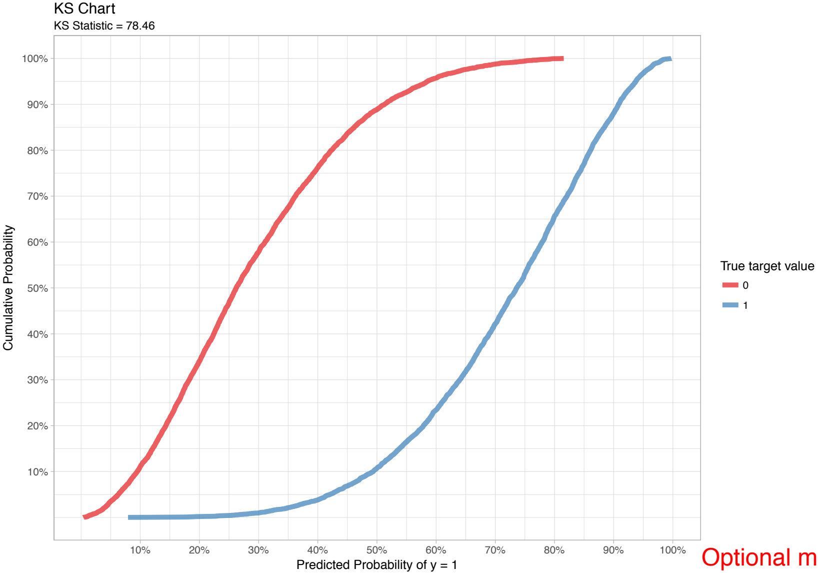
KS stands for Kolmogorov-Smirnov; this statistic is the same metric that they used in their development for a statistical test for equality between two probability distributions.

In ML, we use this idea to compare the cumulative distribution of $\hat{t}$, the soft classifications, for the true positives $t = 1$ and true negatives $t = 0$, as a function of $h$.

The maximum distance between these curves is the KS Statistic.

Optional material

# The KS Chart



Optional material

# The KS Statistic

The KS Statistic is a measure of how well the classifier *discriminates* between the two classes.

If all of the predicted probabilities are very close together, *regardless of their magnitude*, then these two cumulative distributions will be close together, and KS will be small.

If the predicted probabilities are very high for the $t = 1$ group and very low for the $t = 0$ group, then the curves will be well separated and KS will be high.

KS is invariant to strictly monotone transformations of the predicted probabilities, which is convenient.

Optional material

# ROC Curve

How does the KS statistic relate to the notion of picking a cutoff for assigning hard classifications?

For models where a good cutoff (in terms of separation of the true 1's and 0's) exists, the KS will be high.

We can look at another commonly used statistic for assessing the power of the model to discriminate between classes.

# ROC Curve

Another commonly used method to assess the power of a model to discriminate between classes is the ROC (Receiver Operating Characteristic) Curve.

Core concept: Each value of predicted probability, $h \in (0, 1)$, that we use as a cutoff generates a distinct $2 \times 2$ confusion matrix.

Sometimes the subset of $\mathbb{R}^{2\times 2}$ containing all possible contingency tables for all possible cutoffs is called the *ROC Space*.

# ROC Curve

Suppose, again, we have a collection of soft classifications $\hat{t}_n \in (0, 1)$.

We choose a cutoff $h$ such that we classify

$$\hat{t}_{n,hard} = \begin{cases} 1 \text{ if } \hat{t}_n > h \\ 0 \text{ else} \end{cases}$$

This gives us a contingency table.

This, in turn, gives us a distinct $TPR_h$ and $FPR_h$.

# ROC Curve

cutoff h -> compute confusion matrix -> plot (FPR,TPR)

The *parametric curve* $(x(h), y(h)) = (FPR_h, TPR_h)$ is called the **ROC Curve**.

want FPR (type i error) be low and TPR (power) be high

A model with a wider ROC curve will allow us to choose a threshold that gives a high TPR with a low FPR, which is desirable.

The area under the ROC curve, or **AUC**, is a measure of the ability of the model to rank order observations by their probability of $t = 1$.

In fact, $AUC = P(\hat{t}_1 > \hat{t}_2 | t_1 = 1, t_2 = 0)$, i.e. it is the probability of the model correctly rank-ordering a true 1 vs a true 0.

# Plotting the ROC Curve

To actually plot the ROC curve for a particular application:

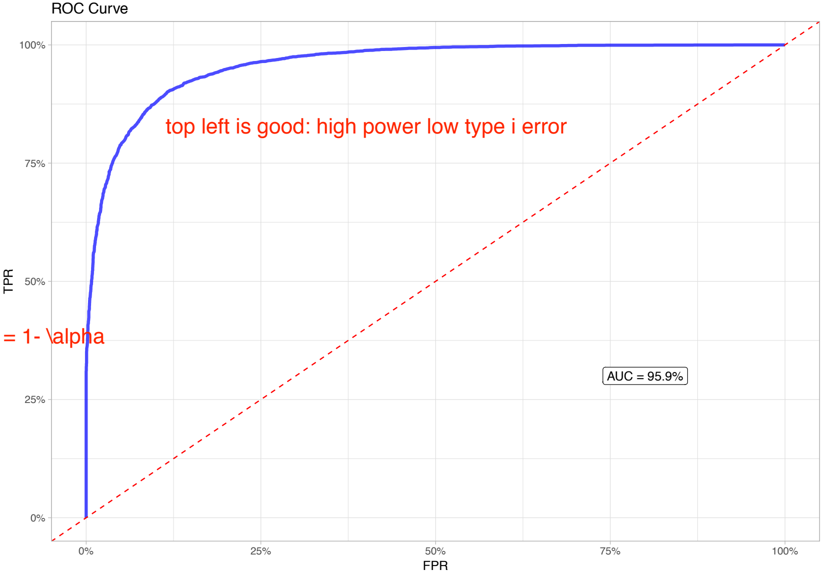
- Take all the distinct predicted probabilities that your model yields
- Compute the TPR and FPR that result from using each as a cutoff
- Plot the curve (FPR,TPR)

Compute the AUC using numerical integration (e.g. trapizoid rule):

$$A\hat{U}C = \sum_i \frac{TPR(h_i) + TPR(h_{i-1})}{2} \times (FPR(h_i) - FPR(h_{i-1}))$$

# ROC Curve

AUC <= 1 since bounded by a square



top left is good: high power low type i error

\beta = 1- \alpha