Logistic Regression

Delta Analytics builds technical capacity around the world.

This course content is being actively developed by Delta Analytics, a 501(c)3 Bay Area nonprofit that aims to empower communities to leverage their data for good.

Please reach out with any questions or feedback to inquiry@deltanalytics.org.

Find out more about our mission here.

Quick Review

Regression & Classification

- ML studies how to **automatically learn** to make accurate predictions based on **past observations.**

- Two types of supervised tasks, regression and classification.

Regression
Continuous variable

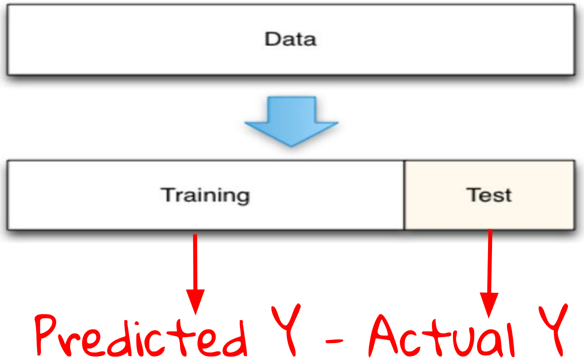
Classification
Categorical variable

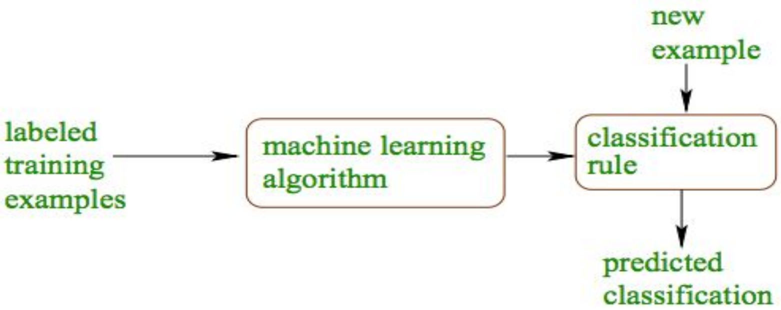
Ordinary Least Squares (OLS) regression

Logistic regression

Model's Performance and Evaluation

- Ability to generalize to unseen data:





Source: Machine Learning Algorithms for Classification, Schapire (2016)

- General Steps:
  - Split data into "training" and "test" sets.
  - Use regression/classification results from "training" set to predict"test" set
  - Compare Predicted Y to Actual Y

- Validation metrics (OLS):

** R2 ** Adjusted R2 ** MSE

Module 3: Logistic Regression

Overview of Logistic Regression:

Task	Model the probability that *Y* belongs to a particular category
Learning Experience	Supervised
Performance Measure	Accuracy, misclassification rate, precision, recall. ROC curve, confusion matrix, etc.

Source: Deep Learning Book - [Chapter 5: Introduction to Machine Learning]

Task

Module Checklist

- ❑ Logistic Regression
  - ❑ Task
    - ❑ Dilemma using OLS
    - ❑ Odds ratio
    - ❑ Logit link function
    - ❑ Probability thresholds
  - ❑ Learning Experience
    - ❑ Cost function
    - ❑ Optimization process
  - ❑ Performance
    - ❑ Confusion Matrix
    - ❑ ROC and AUC

Task

Dilemma using OLS

- A linear regression with variable(s) matrix ***X*** predicting target ***y*** is formulated as:

$$E(y|X) = \beta_0 + \sum_j^p \beta_j x_j$$

Expected value (mean) of y given variable matrix X

Beta (zero) intercept

SUM (predictors j thru p (columns) of the matrix X)

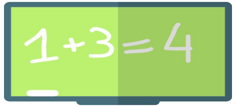
Beta (j), coefficient for the predictor x_j, the j^{th} column in variable matrix X

- With linear regression, it is difficult to assign an observation to a category
- Let's use a simple example: predict college admissions using GRE, GPA, and college prestige

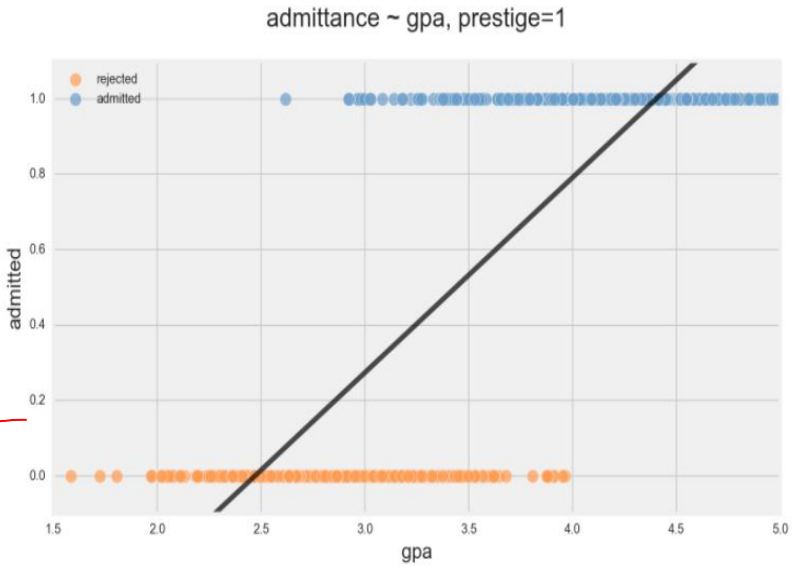
Task

Dilemma using OLS

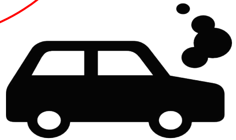
Predicting College admission from gpa, gre and school prestige



	admit	gre	gpa	prestige
0	0	380.0	3.61	3.0
1	1	660.0	3.67	3.0
2	1	800.0	4.00	1.0



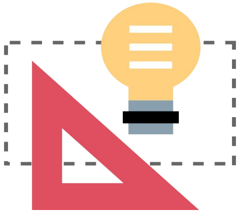
Houston we have a problema!!



Task

Dilemma using OLS

Framing the idea in classification terms



- We have a basic "binary" classification problem
  - *1 = admitted and 0 = rejected*

- Keep in mind that the logistic regression is still solving for an expected value. In the binary classification case this expected value is the probability of one class:

$$E[y \in 0, 1] = P(y = 1)$$

- In regression syntax we would have:

$$P(y = 1) = \beta_0 + \sum_j^p \beta_j x_j$$

Task

Dilemma using OLS

Estimate the probability instead of a real number!!!

- There is a problem with this new equation: we want to estimate a probability instead of a real number.
    - *We need y to fall in the range [-infinity, infinity] for the regression to be valid!*

$$P(y=1) = \beta_0 + \sum_j^p \beta_j x_j$$

y to fall in the range [-infinity, infinity]

- Here is where the logit "link function" comes to our rescue!!

Probability of class membership....

- Logistic regression is a twist on regression for categorical/class target variables, where instead of solving for the *mean* of ***y***, logistic regression solves for the ***probability of class membership of y***.
- How does it do this? It uses a **link function** to describe a linear relationship between the probability and our independent variables

The link function is a function of the expected value of the target variable

$$logit(E(y|X)) = \beta_0 + \sum_j^p \beta_j x_j$$

Task

Odds ratio

Modify regression equation...

- What is our link function in the case of logistic regression?
- Our link function will use something called the **odds ratio**

The odds ratio of a probability is a measure of how many times more likely it is than the inverse case.

$$\text{odds ratio}(p) = \frac{p}{1-p}$$

- When `p = 0.5`: `odds ratio = 1`
  - it is equally likely to happen as it is to not happen.
- When `p = 0.75`: `odds ratio = 3`
  - it is 3 times more likely to happen than not happen.
- When `p = 0.40`: `odds ratio = 0.666..`
  - it is 2/3rds as likely to happen than not happen.

Task

Odds ratio

In our example...

Predicting college admission

	admit	gre	gpa	prestige
0	0	380.0	3.61	3.0
1	1	660.0	3.67	3.0
2	1	800.0	4.00	1.0

Probabilities of admittance by college prestige

```
admissions.prestige.unique()
```

```
array([ 3.,  1.,  4.,  2.])
```

```
y_p1 = admissions[admissions.prestige == 1].admit.values
y_p2 = admissions[admissions.prestige == 2].admit.values
y_p3 = admissions[admissions.prestige == 3].admit.values
y_p4 = admissions[admissions.prestige == 4].admit.values
```

```
print 'P(admit | prestige = 1):', np.mean(y_p1)
print 'P(admit | prestige = 2):', np.mean(y_p2)
print 'P(admit | prestige = 3):', np.mean(y_p3)
print 'P(admit | prestige = 4):', np.mean(y_p4)
```

```
P(admit | prestige = 1): 0.540983606557
P(admit | prestige = 2): 0.358108108108
P(admit | prestige = 3): 0.231404958678
P(admit | prestige = 4): 0.179104477612
```

Odds ratios of admittance by college prestige

```
def odds_ratio(p):
    return (float(p) / (1 - p))
```

```
print 'odds(admit | prestige = 1):', odds_ratio(np.mean(y_p1))
print 'odds(admit | prestige = 2):', odds_ratio(np.mean(y_p2))
print 'odds(admit | prestige = 3):', odds_ratio(np.mean(y_p3))
print 'odds(admit | prestige = 4):', odds_ratio(np.mean(y_p4))
```

```
odds(admit | prestige = 1): 1.17857142857
odds(admit | prestige = 2): 0.557894736842
odds(admit | prestige = 3): 0.301075268817
odds(admit | prestige = 4): 0.218181818182
```

Task

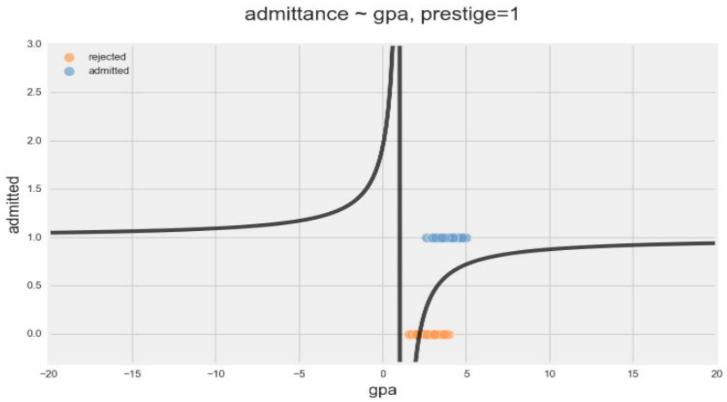
Odds ratio

Modify regression equation...

- If we put the odds ratio in place of the probability in the regression equation, ***the range of odds ratio***, our predicted value, is now restricted to be in the range **[0, infinity]**

$$\frac{P(y=1)}{1-P(y=1)} = \beta_0 + \sum_j^p \beta_j x_j$$

- And graphically will look like this:



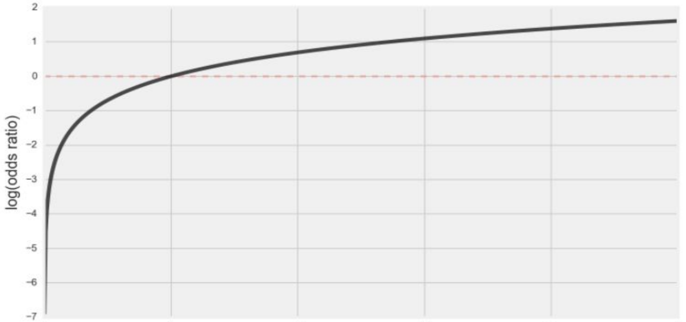
..... hmm something is missing ....

Task

Logit link function

Modify regression equation...

- If we take the natural logarithm of a variable that falls between 0 and infinity, we can actually transform it into a variable that falls between the range negative infinity and infinity.
  - Why? Because taking the logarithm of fractions results in negative numbers.
- And now our graph looks like:



... this is how it needs to look..

Task

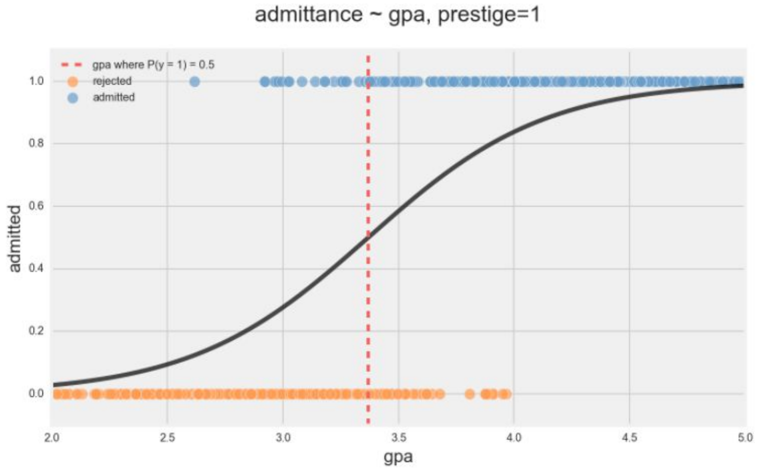
Logit link function

Modify regression equation...

- The combination of converting the probability to an odds ratio and taking the logarithm of that is called the **logit link function**, and is what regression uses to estimate probability:

$$\mathrm{logit}\big(E[y]\big) = \mathrm{logit}\big(P(y=1)\big) = log\left(\frac{P(y=1)}{1-P(y=1)}\right) = \beta_0 + \sum_j^p \beta_j x_j$$

- Graphically looks like this:



Houston we solved the problema!

Task

Probability thresholds

- Now that we have a probability, how do we actually classify the data?
- Choose a probability depending on the type of the classification problem we're solving for:

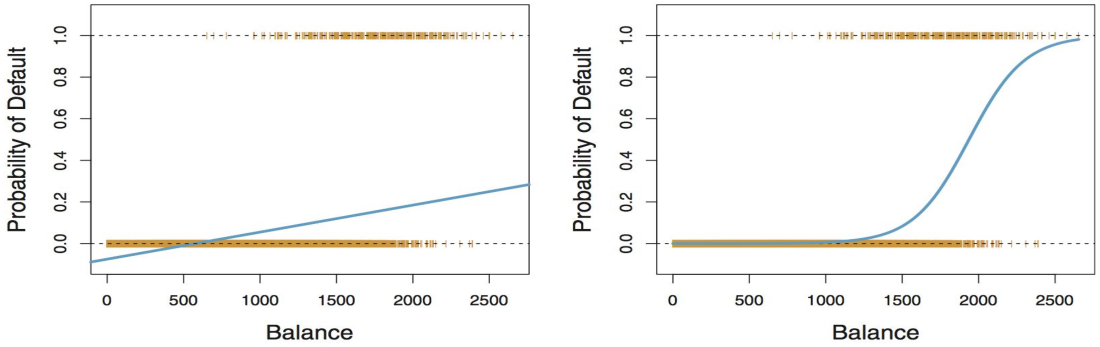
$$y = \begin{cases} 0 & if\ p < 0.5 \\ 1 & if\ p \geq 0.5 \end{cases}$$

In this case, 0.5 is the threshold probability. Threshold can be adjusted by model.

Task

Let's check our understanding of logistic regression

- Here is a classification example, where account balance is used to predict the probability of default.
- Can you guess what is the correct classification method linear regression (left) and logistic regression (right)?



Learning Methodology

- Logistic regression, like OLS, is solved by minimizing a loss function, also called a **cost function**
- The cost function for OLS was the RSS (residual sum of squares), but the cost function for logistic regression is:

$$
\begin{aligned}
J(\theta) &= \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)}) \\
&= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log (1-h_\theta(x^{(i)}))]
\end{aligned}
$$

Learning Experience

Cost function

- Let's break this down. We want to minimize the cost function, J

1. For each classifier we've predicted...

2. Add up the "cost" of the prediction, where h(x) is the prediction and y is the actual classification

$$\boxed{J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})}$$

$$= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))\right]$$

Learning Experience

Cost function

- The cost function should be higher when our predictions are wrong and lower when they are right

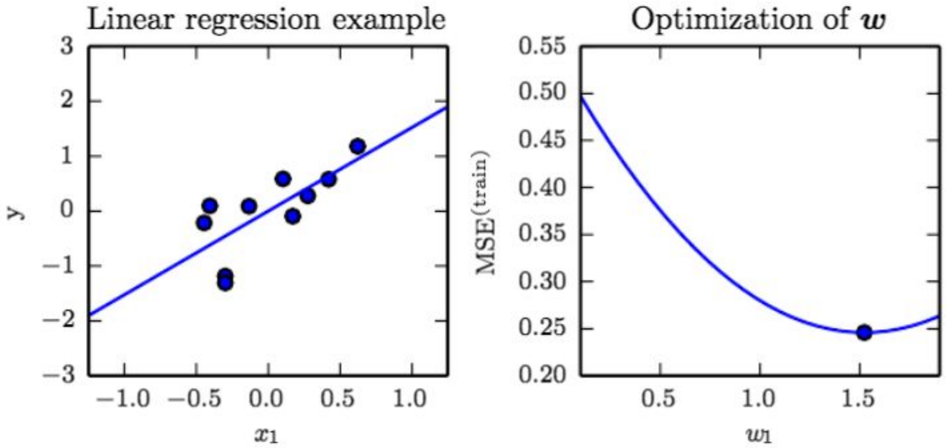
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$
$$= -\frac{1}{m}[\sum_{i=1}^{m} \boxed{y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]}$$

- The cost function meets our need! When h(x) = 1 and y(0) , then the cost function is infinite

Learning Methodology

Optimization process

Optimization of MSE:



- Like OLS, logistic regression **learns by gradient descent** to minimize the cost function
- Reminder of gradient descent from OLS

Source: Seeing Theory - Regression

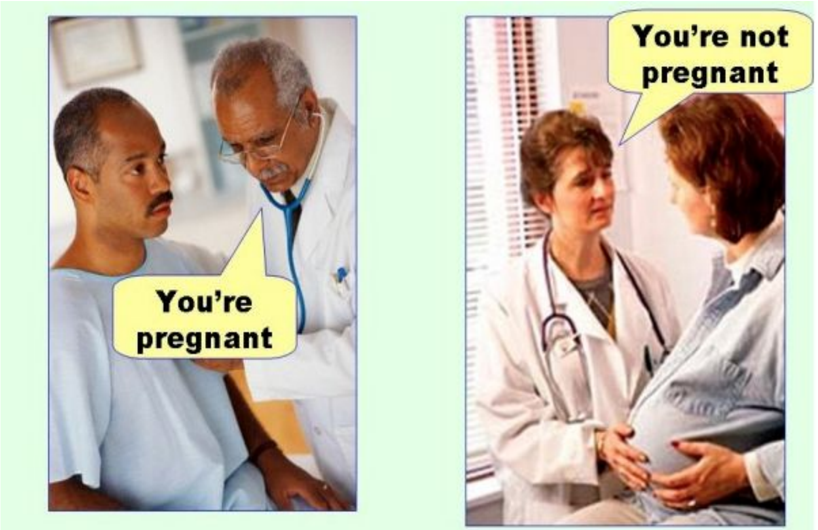
Performance Metrics

Performance

Model evaluation

Confusion Matrix

Imagine you are going to see your doctor and you get an incorrect diagnosis



Source: [Computing for the Social Sciences, University of Chicago]

		predicted	
		positive	negative
truth	positive	tp	fn
	negative	fp	tn

True Positive (tp): The cases in which the model predicted "yes/positive", and the truth is also "yes/positive."

True Negatives (tn): The cases in which the model predicted "no/negative", and the truth is also "no/negative."

False Positives (fp): The cases in which the model predicted "yes/positive", and the truth is "no/negative".

False Negatives (fn): The cases in which the model predicted "no/negative", and the truth is "yes/positive".

Using information from the confusion matrix

		predicted	
		positive	negative
truth	positive	tp	fn
	negative	fp	tn

Number samples:

n = tp + tn + fp + fn

Accuracy:

In general how often is the classifier correct? => (tp + tn) / n

Misclassification Rate (Error Rate):

How often is the model wrong => (fp + fn) / n

Precision:

When the model predicts "yes", how often is it correct? => tp / (tp + fp)

Recall (True Positive Rate):

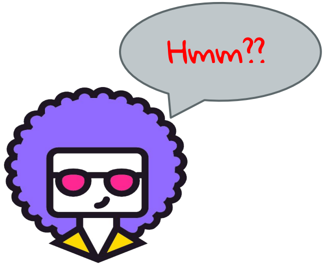
How often the model predicts yes, when it's actually yes => tp / (tp + fn)

Performance

Model evaluation Example

Can we predict if a congressmen/women is a republican or democrat? Lets use the [1984 United States Congressional Voting Records Database]

Assume that we have set and run a logistic regression (gridsearch for hyperamaters), etc and we got the following output:




```
Best estimators on the left out data:
LogisticRegression(C=6.1584821106602643, class_weight=None, dual=False,
          fit_intercept=False, intercept_scaling=2, max_iter=100,
          multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
          solver='liblinear', tol=0.0001, verbose=0, warm_start=False)

Best C / Regularization Param on the left out data:
6.15848211066

Best Params on hold out data (train):
{'C': 6.1584821106602643, 'intercept_scaling': 2, 'fit_intercept': False, 'solver': 'liblinear', 'penalty': 'l2', 'cl
ass_weight': None}

Best Score on left out data:0.964
```

Now, evaluate the model => knowing that if we randomly choose from our dataset, 61 % of the time you will guess /choose democrat ***(there are 267 democrats and 168 republicans in the dataset)***

Note: ***In this lecture, we are omitting other factors (class imbalance, how to gridsearch hyperparameters, etc).***

Performance

Model evaluation Example

Here is the confusion matrix, let's calculate some model performance indicators

	Predict_Label_0 Republican	Predict_Label_1 Democrat
True_Label_0 Republican	49	2
True_Label_1 Democrat	2	78

Number of samples:

n = tp + tn + fp + fn => 49 +78 + 2 + 2 = 131

Accuracy:

In general how often is the classifier correct? => (tp + tn) / n

=> (49+78) 131 => 0.9694 or 96.94%

Misclassification Rate (Error Rate):

How often is the model wrong => (fp + fn) / n => 4 / 131 => 0.03053 or 3.053%

Precision:

When the model predicts "yes", how often is it correct? => tp / (tp + fp)

=> 49 / (49 + 2) => 97.5%

Recall (True Positive Rate):

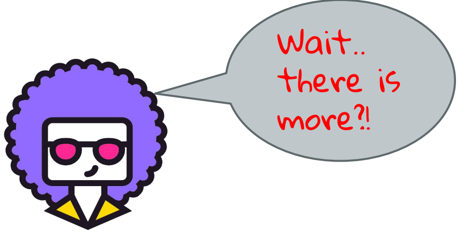
How often the model predicts yes, when it's actually yes => tp / (tp + fn)

=> 49 / (49 + 2) => 97.5%

Performance

Model evaluation

Other thresholds



What if instead of boosting the overall model *accuracy*, we want to improve a “class-specific” accuracy?

- This can be the case when we want to increase *sensitivity/recall* => increase of the true positive rate (TPR)
    - True Positive Rate = tp / (tp + fn) => 49 / (49 + 2) => 97.5%
- On the other hand, if we want to increase *specificity* we will need to increase the true negative rate (TNR)
    - False Positive Rate = fp / (fp + tn) => 2 / (2 + 78) => 2.5%

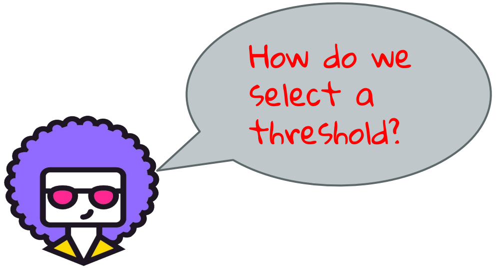
How do we accomplish this?

- Estimate a better model (achieve higher sensitivity and specificity)
- Use our existing model to meet one of these goals
    - Adjusting a threshold, or the *cut-off point* for classifying individuals as “democrats or republicans”

Performance

Model evaluation

ROC Curve



- We can graph across many combinations of thresholds, and then select a threshold level at a point on which we are comfortable.
- The best approach is having *domain knowledge* on the benefits and costs of making/considering a threshold (*trade off*).

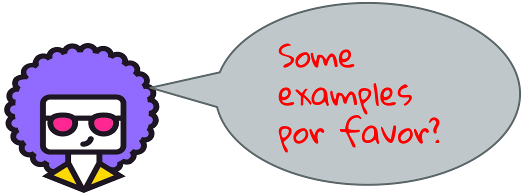
- Receiving Operating Characteristic (ROC) visual way to inspect the performance of a *binary classifier*

  - *In a nutshell with a ROC curve we're measuring the "trade off" between the rate at which the model correctly predicts something, with the rate at which the model predicts something incorrectly.*
  - *As the class assignment threshold increases for the positive class, the false positive rate and true positive rate necessarily increase.*

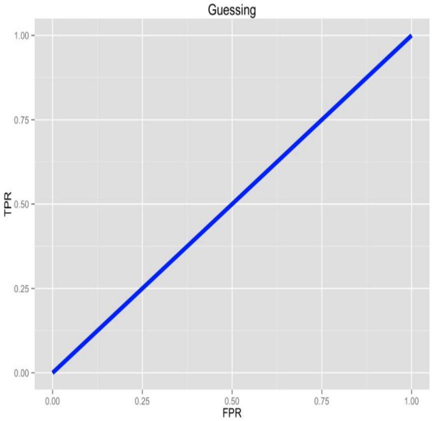
Performance

Model evaluation

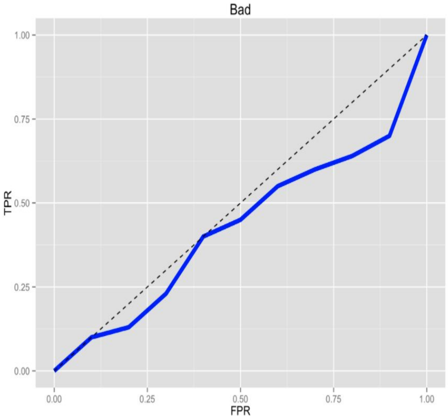
ROC Curve



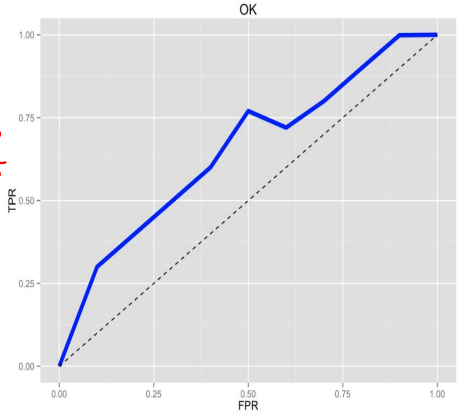
Classifier is making completely random guesses (50/50 chance)



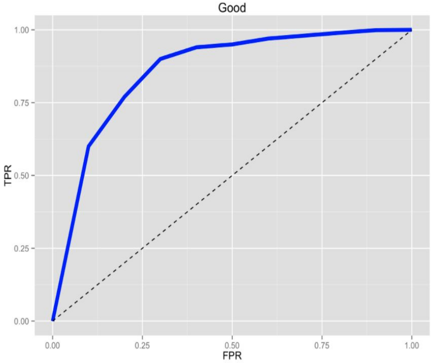
Worse than guessing, the blue line is below the dotted line



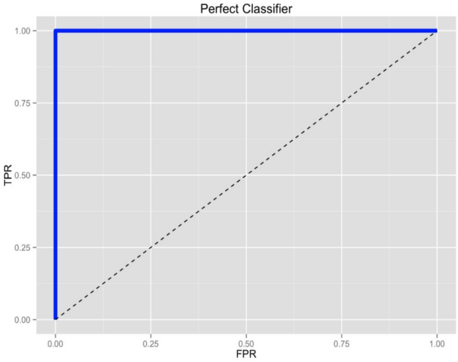
Mediocre classifier, lines that show dips



Good Classifier, the ideal scenario where there is a 'hump shaped' curve that is continually increasing



A perfect classifier is the one that shows a perfect trade-off between TPR and FPR => TPR of one and FPR of zero

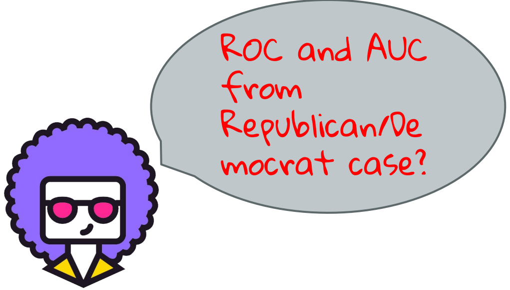


Source: yhat

Performance

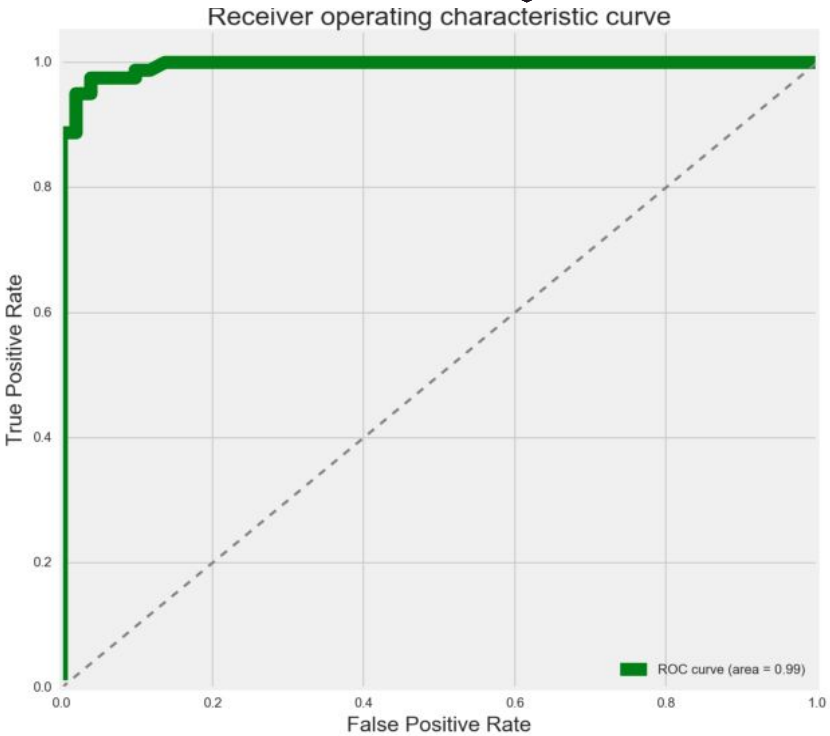
Model evaluation

ROC Curve and AUC



There is one more concept we should know:

- Area under the curve or AUC, is the amount of space underneath the ROC curve.
- AUC shows how well the TPR and FPR is looking in the aggregate.
- The greater the area under the curve, shows the higher quality of the model.
- The greater the area under the curve, the higher the ratio of true positives to false positives as the threshold becomes more lenient
    - AUC = 0 => BAD
    - AUC = 1 => GOOD



Module Checklist

- ✓ Logistic Regression
  - ✓ Task
    - ✓ Dilemma using OLS
    - ✓ Odds ratio
    - ✓ Logit link function
    - ✓ Probability thresholds
  - ✓ Learning Experience
    - ✓ Cost function
    - ✓ Optimization process
  - ✓ Performance
    - ✓ Confusion Matrix
    - ✓ ROC and AUC

Advanced resources

Want to take this further? Here are some resources we recommend:

- Textbooks
  - An Introduction to Statistical Learning with Applications in R (James, Witten, Hastie and Tibshirani): Chapters 4.1, 4.2 4.3

- Online resources
  - Statistical learning: logistic regression - *MACS 30100 - Perspectives on Computational Modeling*
  - Simple guide to confusion matrix terminology
  - A Simple Logistic Regression Implementation

- If you are interested in gridsearch of hyperparameters:
  - Tuning the hyper-parameters of an estimator
  - LogisticRegression (sklearn.linear_model)

Congrats! You finished the module!

Find out more about Delta's machine learning for good mission here.