Machine Learning

Feature Engineering & Model Evaluation

Kien C Nguyen

October 5, 2024

Introduction

- In the Machine Learning models we have seen and implemented so far, features are represented by fixed-size vectors.
- However, in practical Machine Learning problems, data are not always ready in this form.
- For example, in Computer Vision, images are in different sizes and therefore, without further processing, will be represented as matrices with different dimensions.
- For a face recognition problem, we will have to do object detection to extract the face in a photo and a lot more processing before the data can be used as features.
- In NLP (Natural Language Processing), texts can have different lengths, and we need to employ various techniques (E.g bag-of-words, TF-IDF, Word2Vec etc.) to extract features.

Structured, Semi-structured, and Unstructured Data

Structured Data

- Highly organized and easily searchable
- Typically stored in relational databases (e.g., SQL)
- Example: Tables with rows and columns

Semi-Structured Data

- Contains tags or markers to separate elements
- Not as rigidly structured as relational data
- Example: JSON, XML files

Unstructured Data

- Lacks a predefined data model
- Not easily searchable or organized
- Example: Text files, images, videos

Feature Engineering Overview

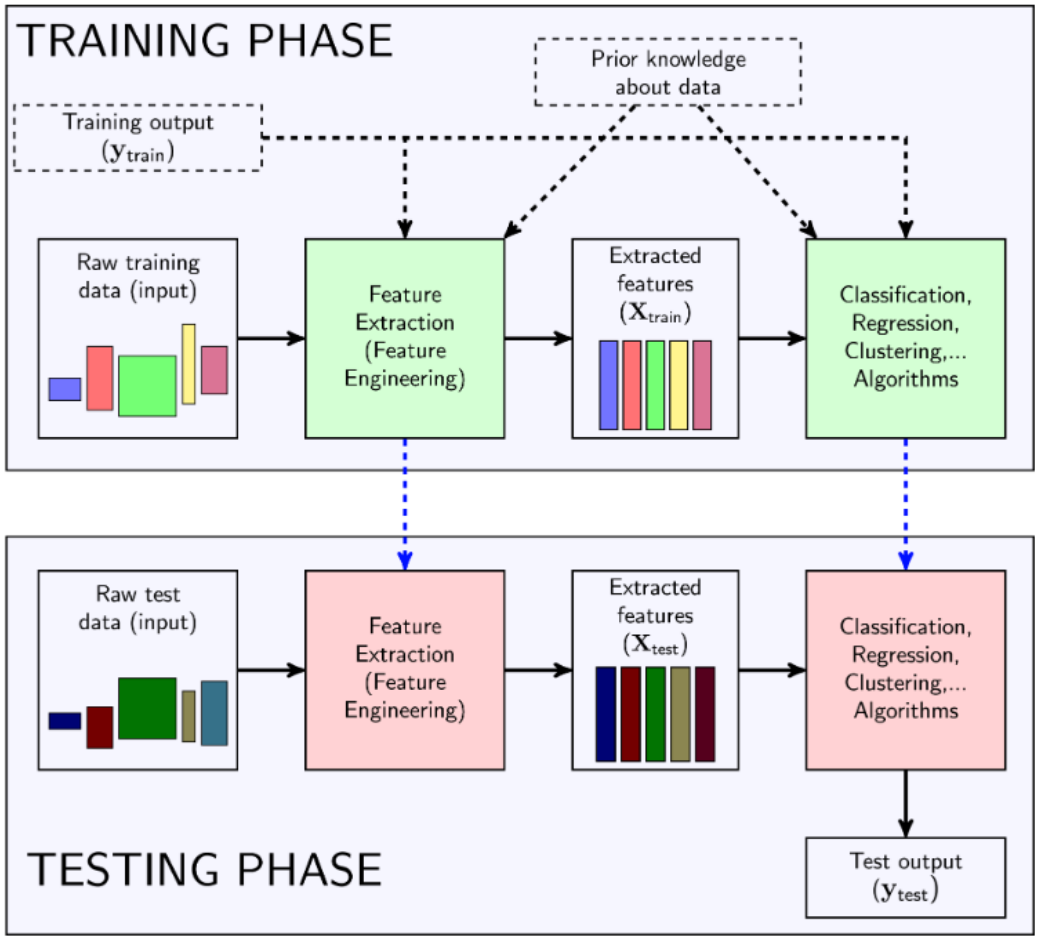


Figure: A standard machine learning pipeline (source: Machine Learning Co Ban, Vu Huu Tiep)

An example problem

Problem Statement: Based on daily Grab usage data, can we classify the income of a customer into two classes: less than 10 million VND / month and more than or equal to 10 million VND / month ?

	user_id	birthday	from_district	to_district	service	promo	data_time	gender	status	label
0	A	10-25-1995	1	1	car	1	2018-09-21 00:00:15	female	plantium	1
1	A	10-25-1995	1	1	car	0	2018-10-09 17:55:52	female	plantitum	1
2	B	20-04-1989	2	3	bike	0	2018-10-28 21:54:15	male	silver	1
3	C	11-18-1990	11	11	car	0	2018-10-28 01:52:27	female	gold	0
4	B	20-04-1989	2	1	car	1	2018-10-28 02:10:50	male	silver	1
5	D	15-03-1980	8	10	bike	1	2018-10-11 20:25:26	female	silver	0
6	E	27-08-1970	go_vap	go_vap	bike	0	2018-11-08 17:51:18	male	silver	0
7	F	11-12-2000	phu_nhuan	1	bike	1	2018-11-08 10:48:43	male	silver	0
8	C	11-18-1990	2	12	bike	1	2018-11-02 14:55:33	female	gold	0
9	A	10-25-1995	1	thu_duc	delivery	1	2018-11-02 22:29:04	female	plantium	1
10	D	15-03-1980	8	2	food	0	2018-09-01 10:39:23	female	silver	0
11	A	10-25-1995	1	1	food	0	2018-09-01 08:29:46	female	plantium	1
12	B	20-04-1989	2	1	car	1	2018-09-01 15:05:59	male	silver	1

Figure: Grab daily usage data (Synthetic)

What is feature engineering?

- "Coming up with features is difficult, time-consuming, requires expert knowledge. 'Applied machine learning' is basically feature engineering." — Prof. Andrew Ng.
- "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." — Dr. Jason Brownlee

Feature Engineering: Demo

- We **cannot** throw raw data directly to the model as input
- Our goal is to find features that are highly relevant to income

Feature Engineering: Demo

- We **cannot** throw raw data directly to the model as input
- Our goal is to find features that are highly relevant to income
  1. User's age
  2. Average amount of time using a service (car/ bike) per month/ week/ day
  3. Average number of rides per service per month/ week/ day
  4. Total payment per service per month/ week/ day
  5. Most visited district during working hours
  6. Most visited district after working hours
  7. ...

Feature Engineering

- Features can come from two major types based on the raw dataset
  1. Inherent **raw features** are features that can be obtained directly from raw data without any transformation or engineering
  2. **Derived features** are features that can be obtained from feature engineering, i.e through manipulating and transforming raw data. For example, feature *age* is derived from *birthday* column of raw data

Different types of variable in statistics

Numerical (quantitative)

- **Discrete:** integer values, typically counts. E.g. age, sick days per year.
- **Continuous:** takes any value in a range of values. E.g. weight, height.

Categorical (qualititive)

- **Nominal:** mutually exclusive and unordered categories. E.g. sex (male/female), blood types (A/B/AB/O).
- **Ordinal:** mutually exclusive and ordered categories . E.g. disease stage (mild/moderate/severe).

Feature Engineering on Numeric Data

- Let's try out some examples of feature engineering on our numeric data

	user_id	num_usage	age
0	A	4	23
1	B	3	29
2	C	2	28
3	D	2	38
4	E	1	48
5	F	1	18

Figure: Two new features: num_usage and age derived from numeric columns

Feature Engineering on Numeric Data

Some types of feature you might figure out

Indicator features

- thresholds: You can create an indicator variable for $age \geq 21$.
- special events: Tet, Black Friday, Christmas.

Statistics features

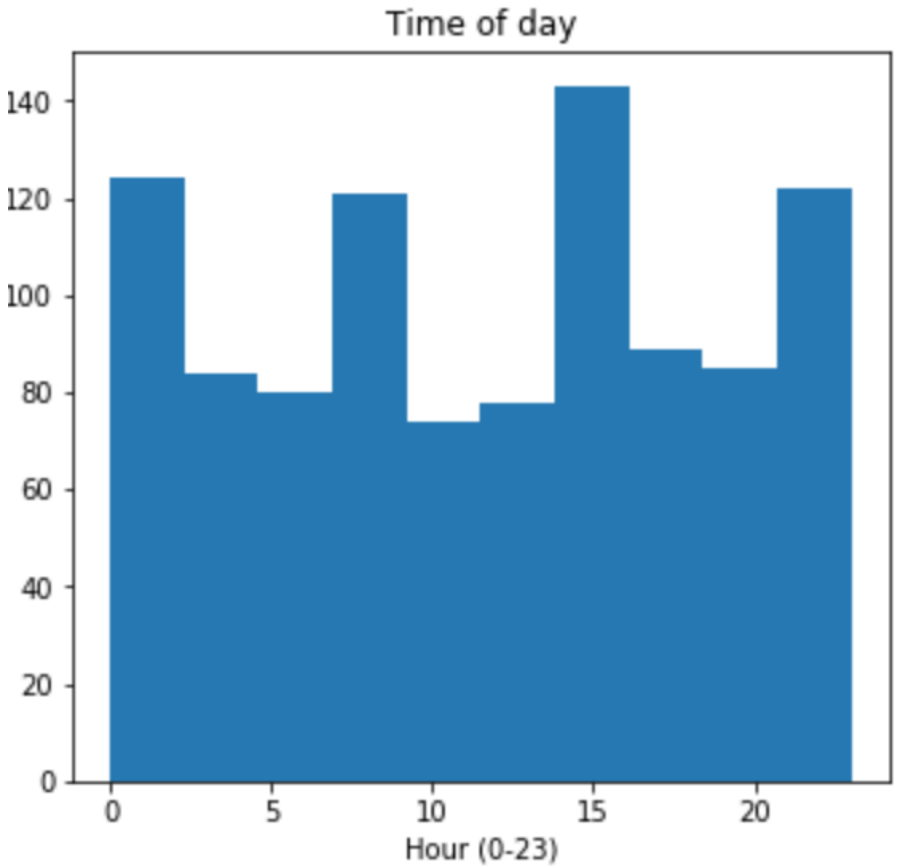
- the count or number of values
- mean
- standard deviation
- minimum, maximum
- 25, 50, and 75 percentiles

Numerical to Categorical Variable

We can transform numerical variable to categorical variable by using the following techniques: binning, ranging, percentile, threshold, etc.
Example time slot mappings. Define time slot:

- $t \in [0, 23]$: time of day.
- night: $6 \leq t$.
- working: $9 \leq t \leq 12$ and $13 \leq t \leq 17$.
- evening: $t \geq 19$.

Time slot mappings



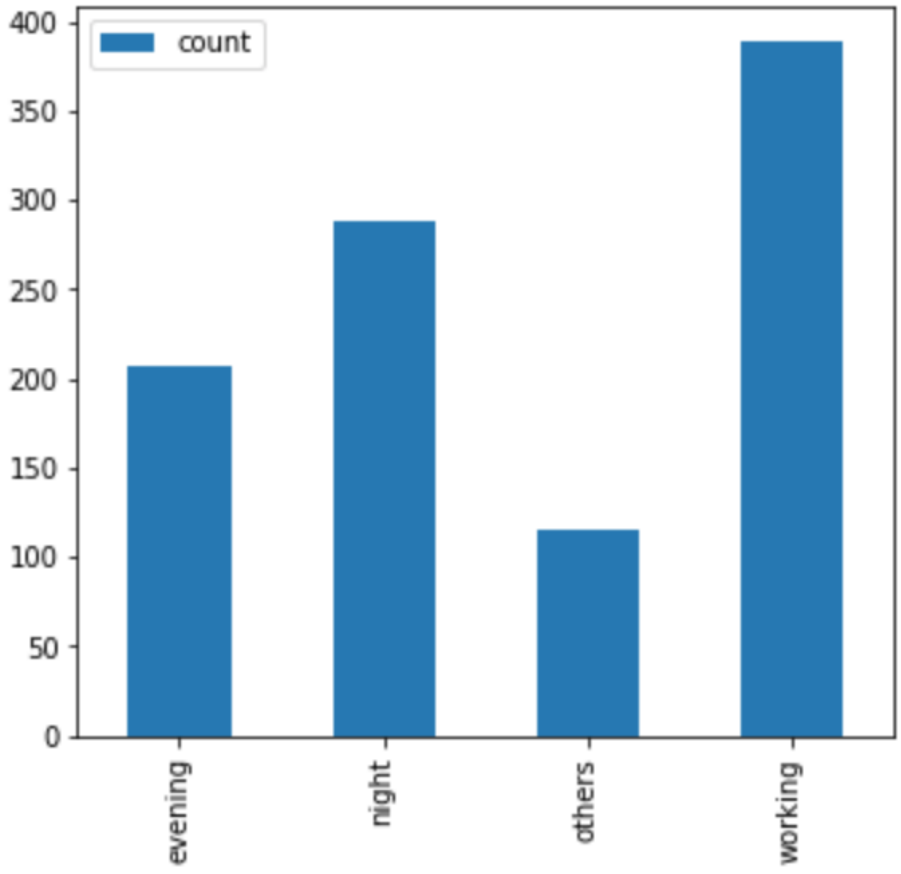


Figure: Time slot mappings from time of day

Standardizing Numerical Variables – Standard scaling

Z-score normalization are features that are re-scaled to have a mean of zero and a standard deviation of one. By doing this, we allow models such as Gradient Descent to learn optimally and not skew towards larger scaled features (e.g. age vs income).

$$z = \frac{x - \mu}{\sigma}$$

Where:

- z: z-score.
- x: previous feature value.
- μ: mean of feature value.
- σ: standard deviation of feature value.

Standard scaling example

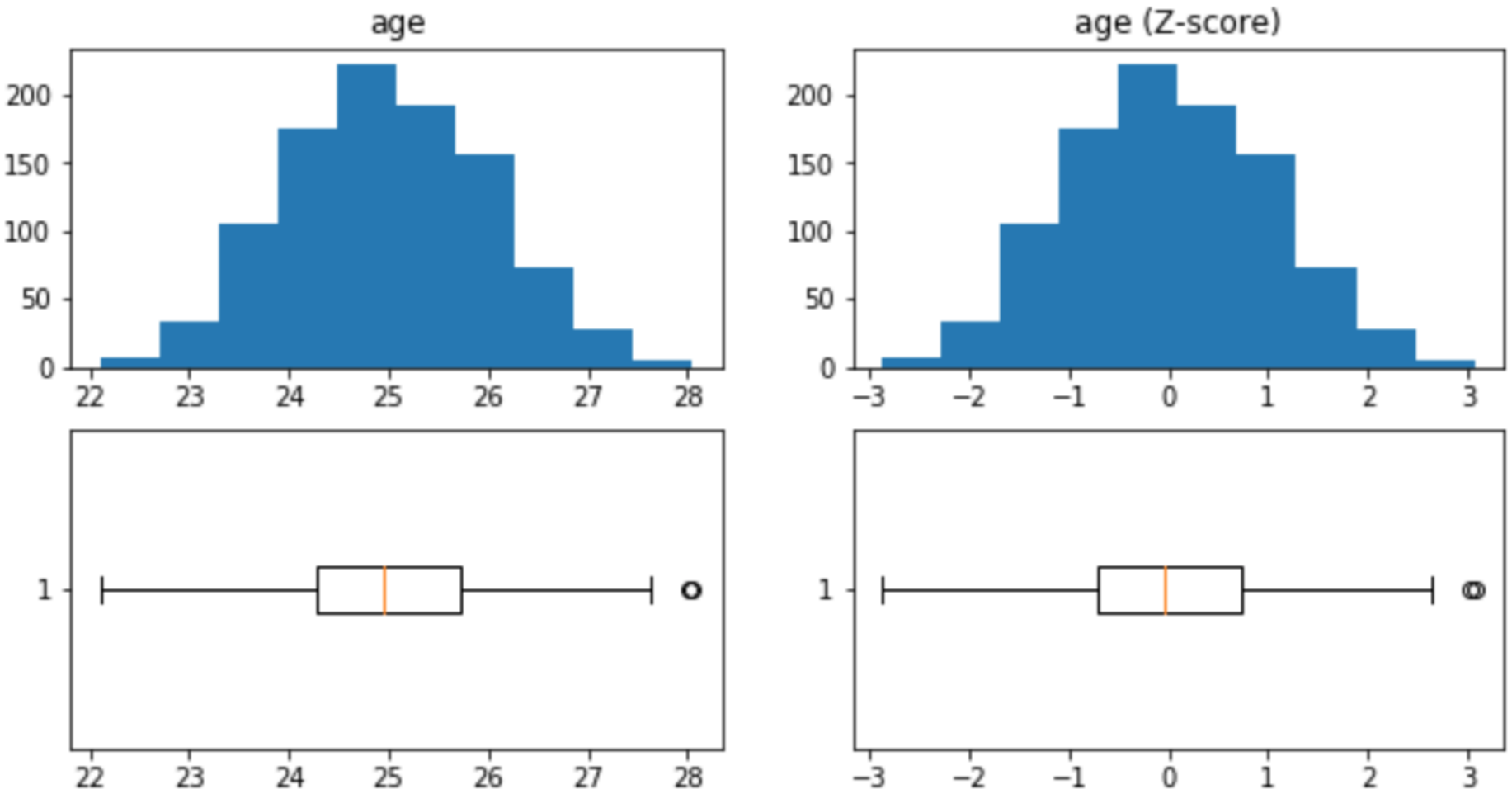


Figure: Z-score transformation on age feature

Standardizing Numerical Variables – min-max scaling

The idea is to get every input feature into approximately a $[0, 1]$ range. The name comes from the use of min and max functions, namely the smallest and greatest values in your dataset. It requires dividing the input values by the range (i.e. the maximum value minus the minimum value) of the input variable:

$$x_i' = \frac{x_i - min(x_i)}{max(x_i) - min(x_i)}$$

Where:

- x_i: is the original i-th input value.
- x_i': normalize feature.

Min-max scaling

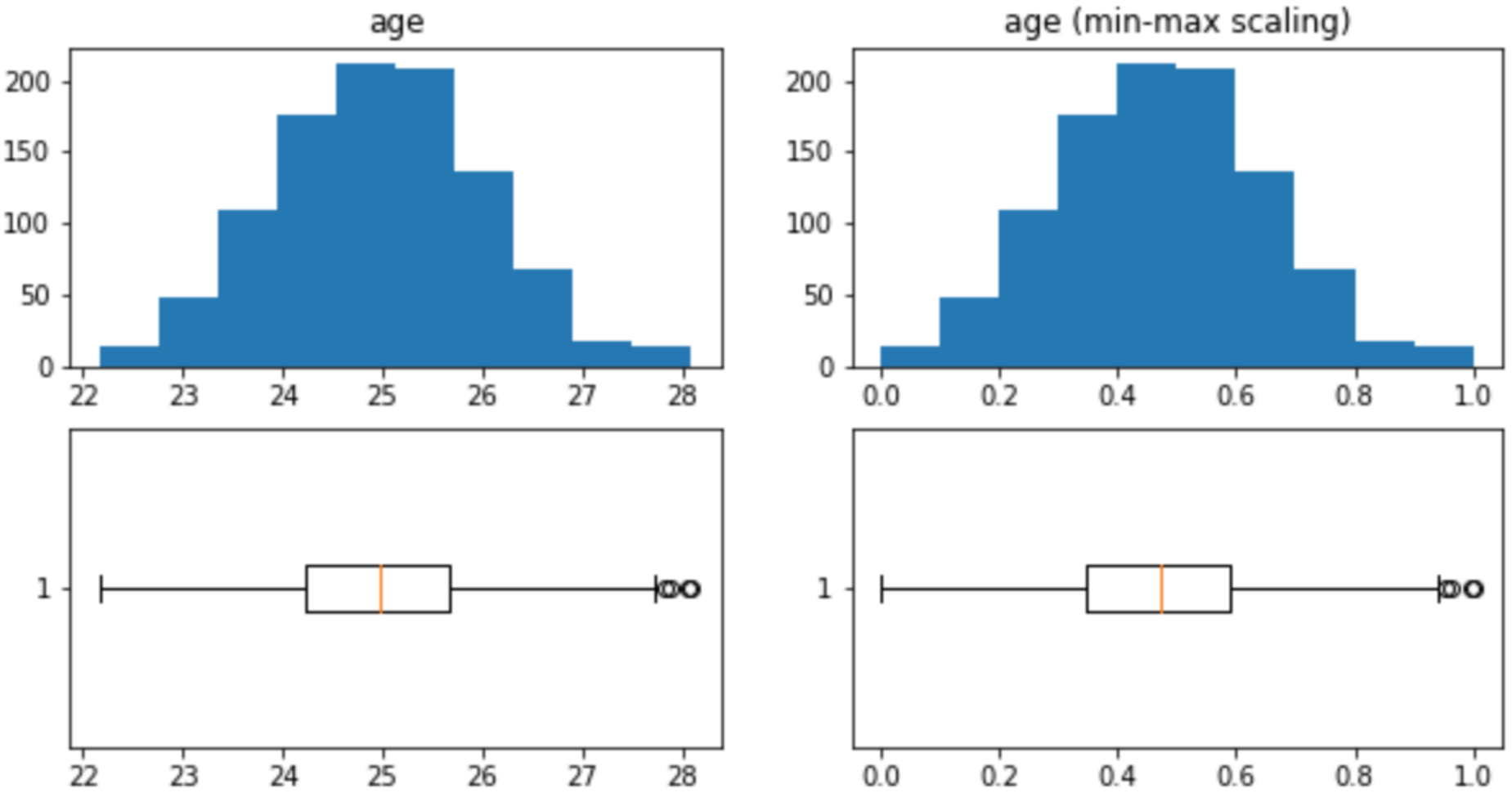


Figure: Min-Max scaling on age feature

Feature Engineering on Categorical Data

- What is the difference between nominal and ordinal features and how we can generate new features from them?
- A nominal variable is one that has two or more categories, but there is no intrinsic ordering to the categories. For example, gender (female and male) has no ordering meaning

Feature Engineering on Categorical Data

- What is the difference between nominal and ordinal features and how we can generate new features from them?
- A nominal variable is one that has two or more categories, but there is no intrinsic ordering to the categories. For example, gender (female and male) has no ordering meaning
- An ordinal variable is similar to a categorical variable. The difference between the two is that there is a clear ordering of the variables. For example, status (platinum, gold, silver) has meaning order
- Why does it matter whether a variable is categorical, or ordinal?
- How do we present information of categorical variables?

Feature Engineering on Categorical Data

- Let consider categorical columns in our dataset

	service	gender	status
0	car	female	plantium
1	car	female	plantitum
2	bike	male	silver
3	car	female	gold
4	car	male	silver
5	bike	female	silver
6	bike	male	silver
7	bike	male	silver
8	bike	female	gold
9	delivery	female	plantium
10	food	female	silver
11	food	female	plantium
12	car	male	silver

Figure: Categorical columns from grab dataset

- How do we derive meaning features from this subset of data?

Feature Engineering on Nominal Data

- Using **one-hot-encoding** for categorical features
- "One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction" -onlineSource
- Some intuitive features: number of car/bike/food/delivery usage, status
- Basically, we will count how many times user used car, bike, food, or delivery service and represent those information in a meaningful way

Feature Engineering on Ordinal Data

- Ordinal data can be converted to numerical data and then further processed using techniques for numerical data
- How ordinal data can be converted numerical data (eg., linearly, nonlinearly) depends on the meaning of ordinal data
- For example, the letter grade (in US schools) $A, A^-, B^+, B, B^-, \ldots$ become $4.00, 3.67, 3.33, 3, 2.67, \ldots$ (linear scale)
- The status of a Grab rider can be converted to the number of points needed for that status. E.g., *Platinum* $\rightarrow$ 4500, *Gold* $\rightarrow$ 1200, *Silver* $\rightarrow$ 300, or to the median points of riders with each status.

Feature Engineering on Categorical Data

	user_id	gender_female	gender_male	bike	car	delivery	food
0	A	1	0	0.0	2.0	1.0	1.0
1	A	1	0	0.0	2.0	1.0	1.0
2	A	1	0	0.0	2.0	1.0	1.0
3	A	1	0	0.0	2.0	1.0	1.0
4	B	0	1	1.0	2.0	0.0	0.0
5	B	0	1	1.0	2.0	0.0	0.0
6	B	0	1	1.0	2.0	0.0	0.0
7	C	1	0	1.0	1.0	0.0	0.0
8	C	1	0	1.0	1.0	0.0	0.0
9	D	1	0	1.0	0.0	0.0	1.0
10	D	1	0	1.0	0.0	0.0	1.0
11	E	0	1	1.0	0.0	0.0	0.0
12	F	0	1	1.0	0.0	0.0	0.0

Figure: New features from categorical columns

Roadmap Review

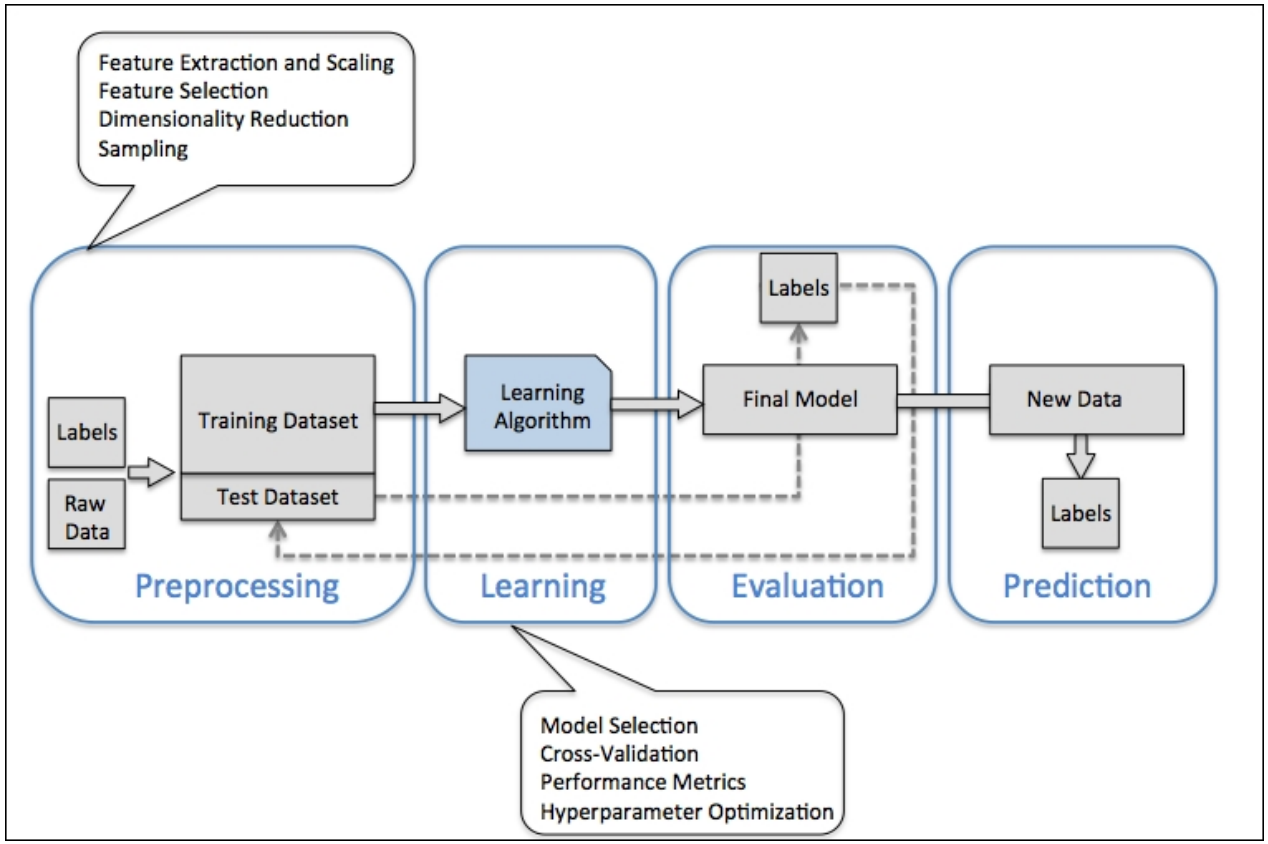


Figure: A roadmap for building machine learning systems (O'Reilly)

Type of Performance Metrics

1. Regression
   - Correlation coefficients
   - Mean Square Error / Root Mean Square Error
   - Mean Absolute Error
   - Residuals
2. Classification
   - Accuracy
   - Precision
   - Recall/Sensitivity
   - Specificity
   - F1-Score
   - ROC/AUC

Review of Linear Regression I

Linear model for regression is a linear combination of the input variables.

Formula

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \cdots + w_D x_D = w_0 + \sum_{j=1}^{D} w_j x_j$$

Loss function

$$L(w) = \frac{1}{2} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Ordinary Least Squares

Our goal is to find $\hat{\mathbf{w}}$:

$$\hat{\mathbf{w}} = argmin_{\mathbf{w}} L(\mathbf{w}) = argmin_{\mathbf{w}} \left(\frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \right)$$

Review of Linear Regression II

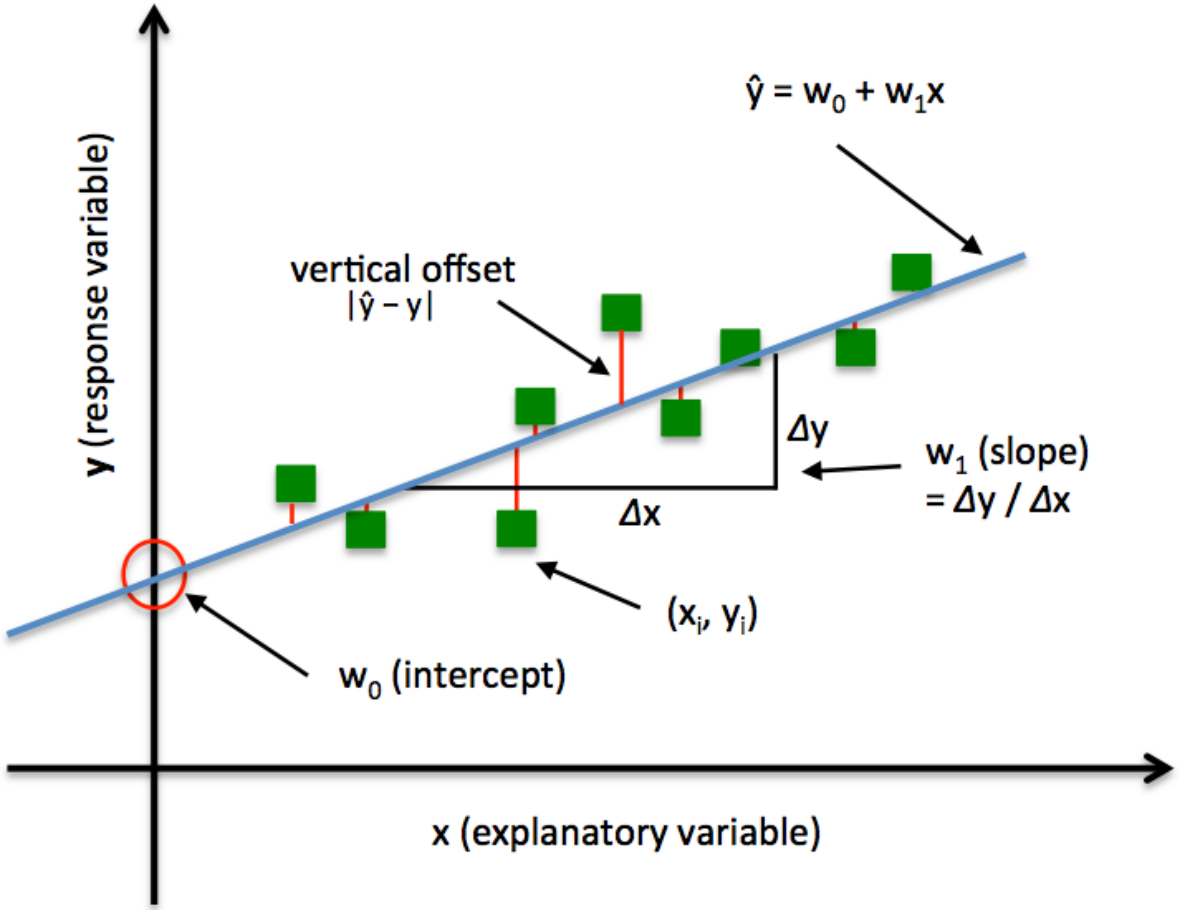


Figure: Linear Regression (Credit: mlxtend)

Performance Metrics for Regression

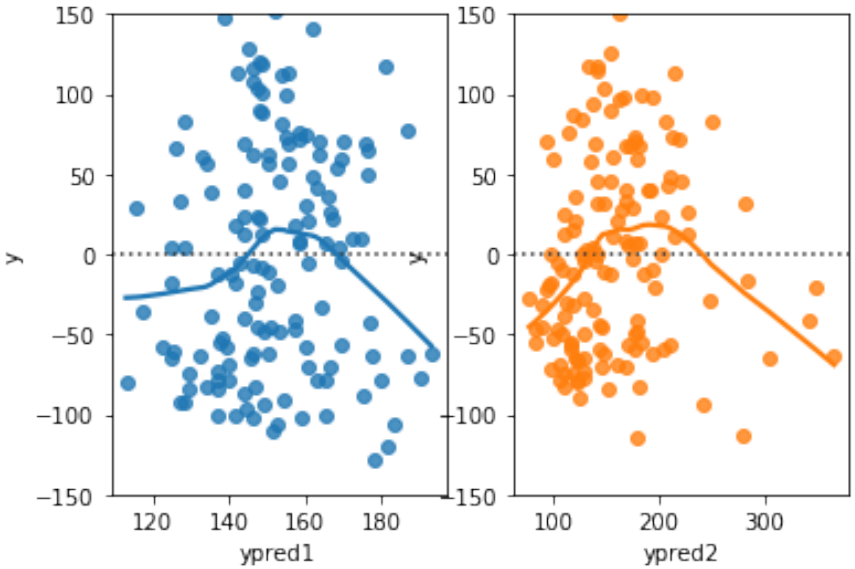
- Mean Square Error:
- Correlation coefficients
- Mean Absolute Error
- Residuals

Residuals

Residual is the difference between true and predicted values:

$$r_i = y_i - \hat{y}_i$$

Residual plot is the scatter-plot of fitted values ($\hat{y}_i$) against residuals (r_i).



Residuals

- Assess visually regression models.
- Diagnostics of regression models:
  - The regression function is nonlinear?
  - Unbalanced data?
  - Outliers?
  - ...

More info: Interpreting residual plots to improve your regression

Some metrics for regression

Mean squared error (MSE) is the average of the squares of the errors:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- MSE is a **loss function**,
- MSE is a **measure of the quality** of an estimator—it is always non-negative, and values closer to zero are better,
- MSE will be small if the predicted responses are very close to the true responses, and will be large if for some of the observations, the predicted and true responses differ substantially.
- To compare among models, MSE on the testing dataset should be considered.

Mean Square Error and the **Brothers**

- Root Mean Square Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

The Bias-Variance Trade-off

It can be proved that for a given point x_0:

$$E\left[(y_0 - \hat{y_0})^2\right] = \underbrace{Var\left[\hat{y_0}\right]}_{Variance} + \underbrace{\left(Bias\left[\hat{y_0}\right]\right)^2}_{(Bias)^2} + \underbrace{Var\left(\epsilon\right)}_{Irreducible\ \ error}$$

- **Variance** refers to the amount by which predicted value would change if we estimated it using a different training data set.
- **Bias** refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model.
- As a general rule, as we use **more flexible** $\sim$ methods, **the variance will increase** $\uparrow$ and **the bias will decrease** $\downarrow$.

The Bias-Variance Trade-off

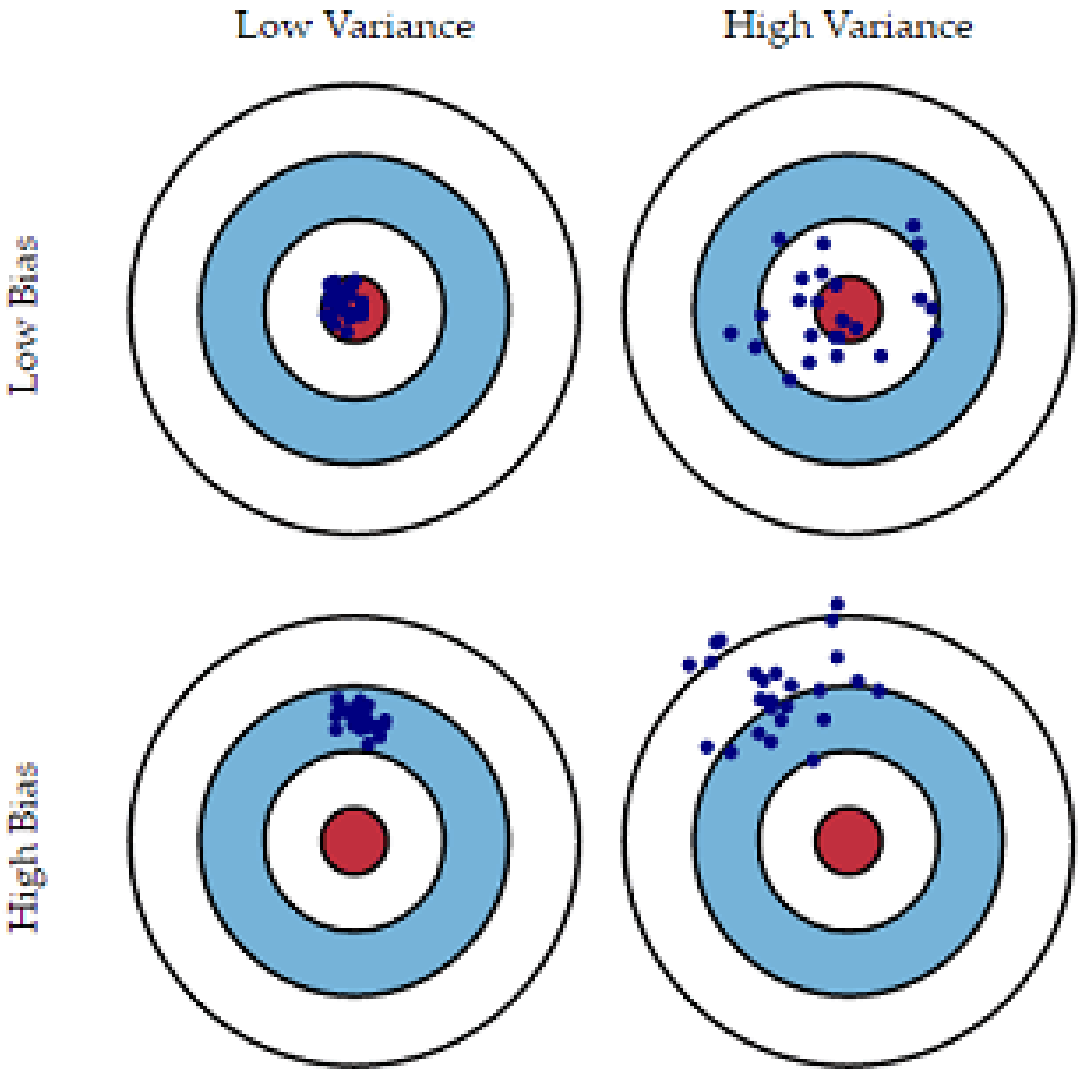


Figure: Illustation of Bias and Variance

Correlation coefficient

1. The Pearson product-moment correlation coefficient
2. Rank correlation

The Pearson product-moment correlation coefficient

The Pearson product-moment correlation coefficient measures linear association:

$$r = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$
$$= \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sqrt{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}\sqrt{\sum_{i=1}^{n} y_i^2 - n\bar{y}^2}}$$

How **Person's r** is interpreted:

- Ranges from -1 to $+1$
- -1 means Perfectly negative linear correlation
- $+1$ means Perfectly positive linear correlation
- 0 means No linear correlation

The Pearson product-moment correlation coefficient

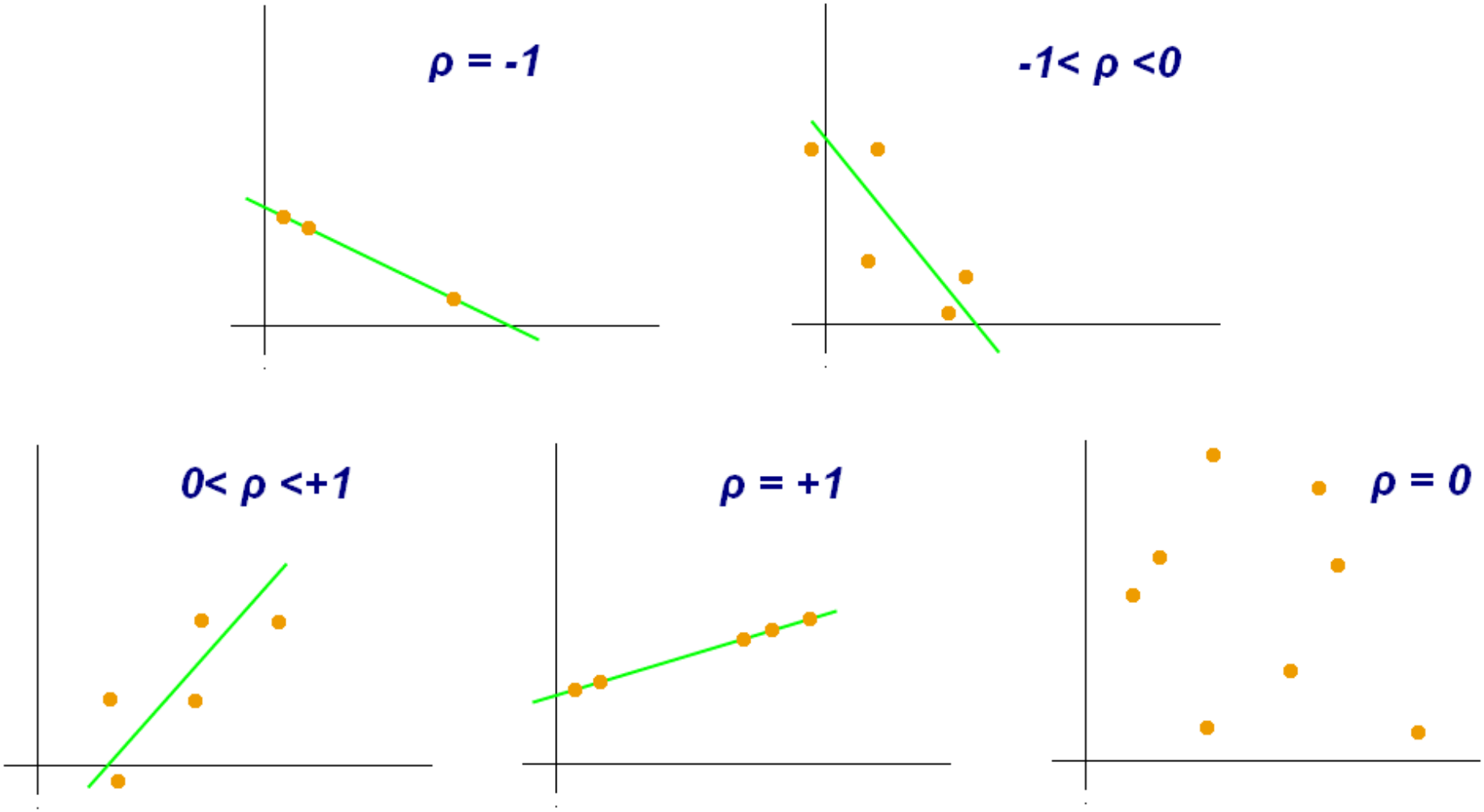


Figure: Illustration of some cases of the Pearson's *r*. (Credit: Wikipedia)

The Pearson product-moment correlation coefficient

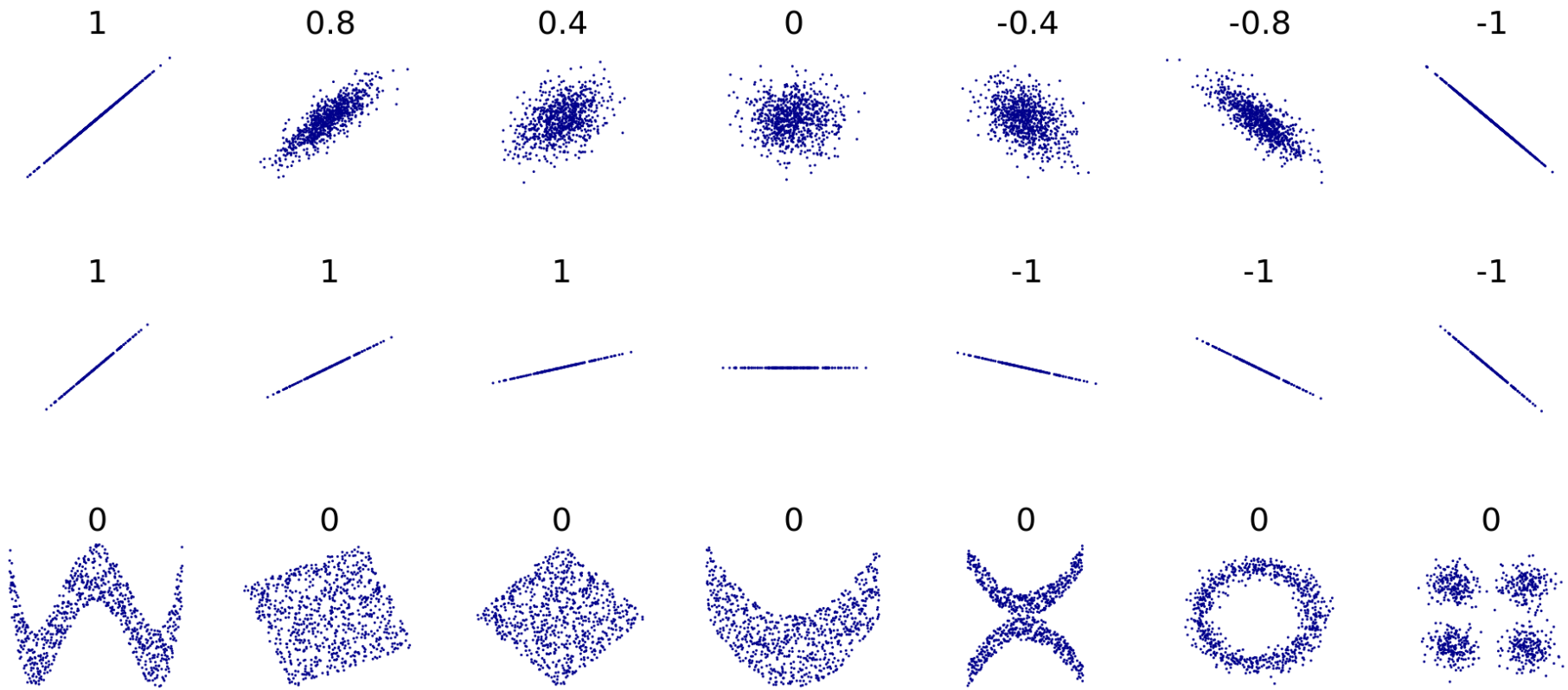


Figure: The Pearson's r is a measure of correlation, not accuracy. (Credit: Wikipedia)

The Pearson product-moment correlation coefficient

The measure r^2 or R^2 is called **coefficient of determination**.

$$R^2 = \frac{TSS - RSS}{TTS}$$

where

- $TTS = \sum(y_i - \bar{y})^2$ is *total sum of squares*, measures the total variance in the response Y.
- $RSS = \sum(y_i - \hat{y}_i)^2$ is *residual sum of squares*, measures the amount of variability that is left unexplained after performing the regression.
- $TSS - RSS$ measures the amount of variability in the response that is explained by regression model.

R^2 measures **the proportion of variability** in response that can be *explained using predictors*.

Rank

X_i	Y_i	rank x_i	rank y_i	d_i	d_i^2
86	0	1	1	0	0
97	20	2	6	-4	16
99	28	3	8	-5	25
100	27	4	7	-3	9
101	50	5	10	-5	25
103	29	6	9	-3	9
106	7	7	3	4	16
110	17	8	5	3	9
112	6	9	2	7	49
113	12	10	4	6	36

Spearman's rank correlation coefficient

Spearman's rank correlation coefficient between two variables X and Y is given by

$$r_s = 1 - \frac{6}{n(n^2-1)} \sum_{i=1}^{n} d_i^2$$

- Ranges from -1 to +1
- +1 or -1 indicates perfect association between X and Y
- When r_s is close to zero, the variables are uncorrelated.

Pearson vs Spearman

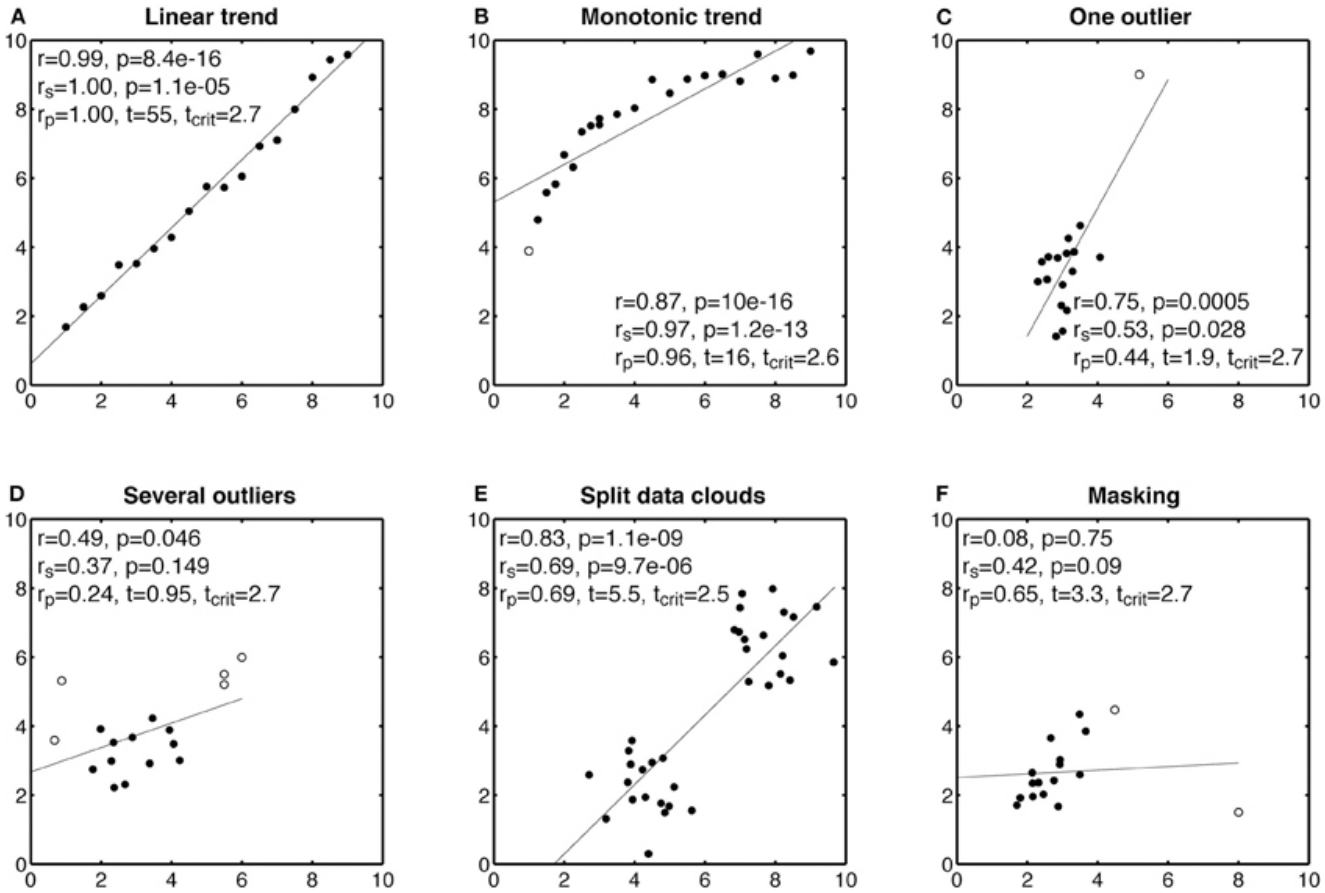


Figure: Examples of Pearson and Spearman correlations. Credit:Guillaume A Rousselet & Cyril Pernet

Classification Algorithms

- Logistics Regression
- Support Vector Machine
- Decision Tress / Random Forest
- Neural Networks
- Nearest Neighbor

Type of Classification Problems

- Binary Classification: Distinguish between 2 classes
- Multiclass Classification: More than 2 classes
- Multilabel Classification: Output muliple classes for each instance
- Multioutput-multiclass Classification (aka Multi-task classification): Generalization of multilabel classification where each label can be multiclass.

Confusion matrix

		Predicted class: P	Predicted class: N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Figure: Confusion matrix.(Credit: mlxtend)

TP = True Positive = Hit.

TN = True Negative = Correct Rejection.

FP = False Positive = False alarm = Type I error.

FN = False Negative = Miss = Type II error

Derivations from a confusion matrix

- **Precision** or Positive Predictive Value (PPV):

$$Precision = \frac{TP}{TP + FP}$$

- **Recall**, Sensitivity, Hit rate, or True Positive rate (TPR) :

$$Recall = \frac{TP}{P} = \frac{TP}{TP + FN}$$

- **Specificity**, Selectivity or True Negative rate (TNR):

$$Specificity = \frac{TN}{N} = \frac{TN}{TN + FP}$$

- **Accuracy**:

$$Accuracy = \frac{TP + TN}{P + N} = \frac{TP + TN}{TN + TN + FP + FN}$$

Why not Accuracy? Accuracy paradox

The **accuracy paradox** *is the paradoxical finding that accuracy is not a good metric for predictive models when classifying in predictive analytics. This is because a simple model may have a high level of accuracy but be too crude to be useful.*

–Wikipedia

Precision & Recall

- **Recall** tells us how confident we can be that all the instances with the positive target level have been found by the model.
- **Precision** captures how often, when a model makes a positive prediction, this prediction turns out to be correct. Precision tells us how confident we can be that an instance predicted to have the positive target level actually has the positive target level.
- Both precision and recall can assume values in the range [0,1], and **higher values** in both cases indicate **better model performance**.

F1 Score

Definition

The **F1 Score** is the harmonic mean of precision and recall and is defined as

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- Why F1 Score? Less sensitive to large outliers $\rightarrow$ highlight shortcomings rather than hide them
- Ranges (0, 1] and higher values indicate better performance.

Precision-Recall Tradeoff

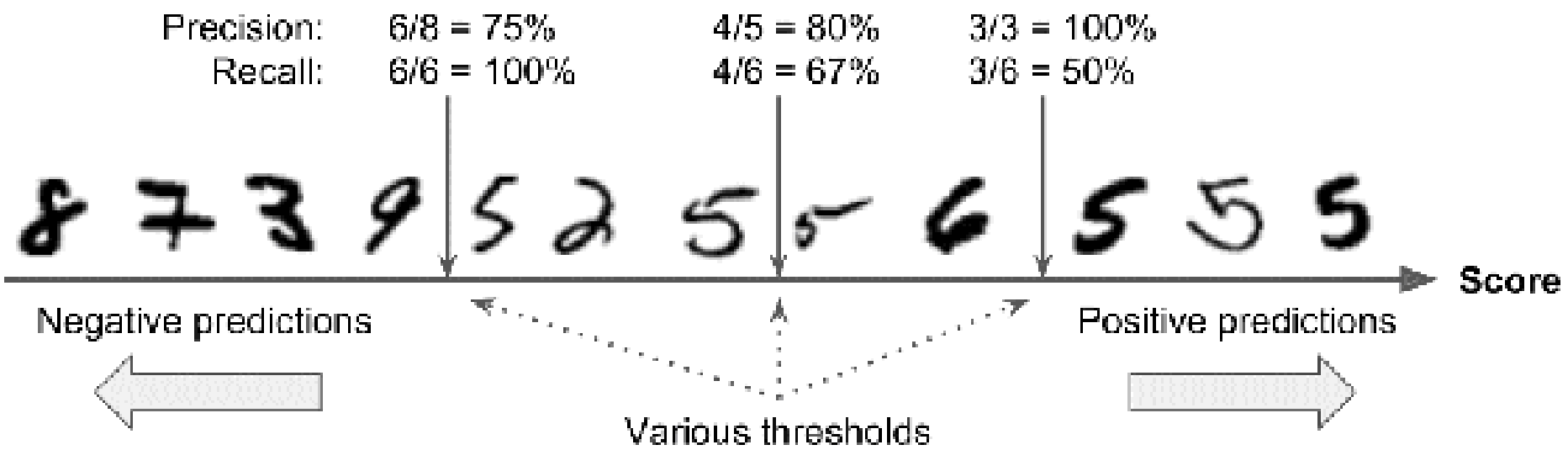


Figure: Threshold and Precision-Recall Tradeoff

Precision-Recall Tradeoff

If someone says "let's reach 99% precision,"
You should ask, "at what recall?"

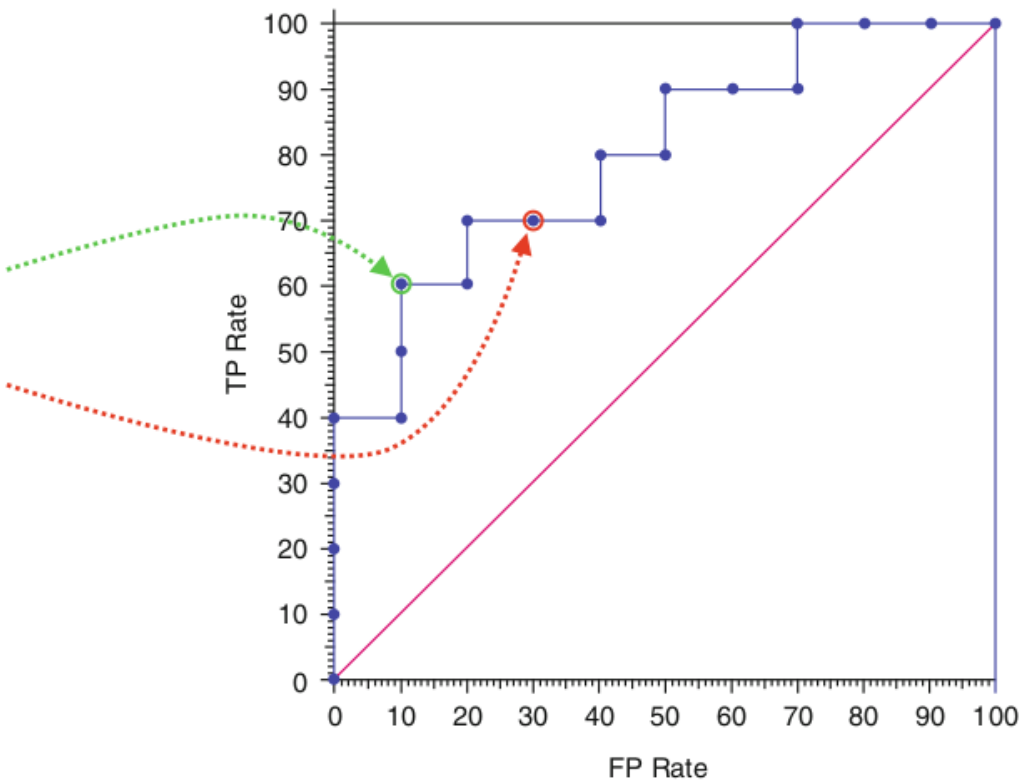
–Aurélien Géron

ROC curve = Receiver operating characteristic curve

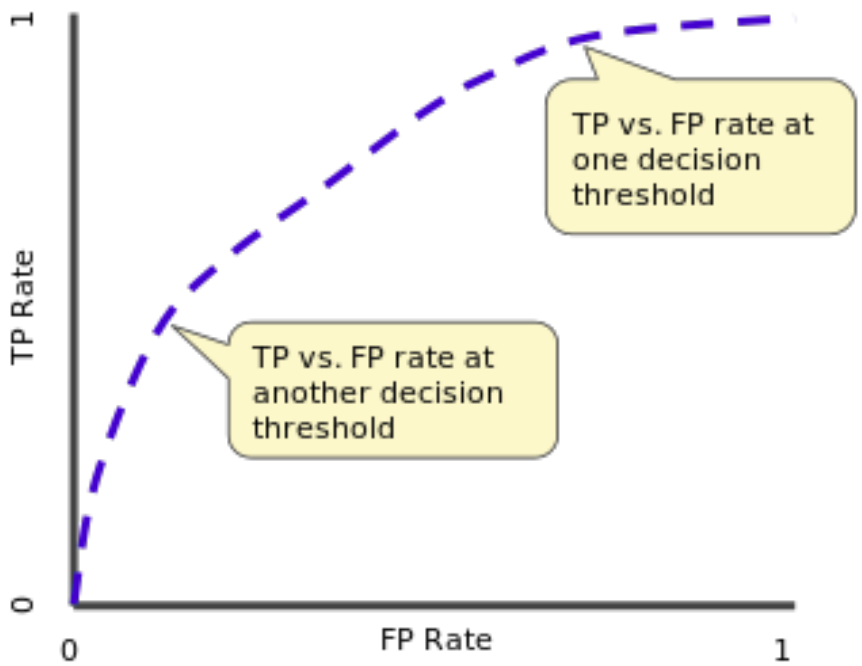
Definition

By joining all possible operating points of a scoring classifier on the ROC plane with line segments, we receive a visual representation of its performance independent of the cutoff value. This is called **the ROC curve**.

Class	Score
+	0.98
+	0.93
+	0.87
+	0.84
–	0.79
+	0.73
+	0.67
–	0.62
+	0.57
–	0.54
–	0.48
+	0.43
–	0.37
+	0.34
–	0.28
–	0.24
+	0.18
–	0.12
–	0.09
–	0.03



ROC curve = Receiver operating characteristic curve



TP rate = Sensitivity
FP rate = 1 - Specificity
(0,1): The perfect model with all **correct** classification.
(1,0): The worst model with all **incorrect** classification.
(0,0): Always predicts class 0.
(1,1): Always predicts class 1.

AUC = Area Under the ROC Curve

Definition

Area Under the ROC Curve (AUC) measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).

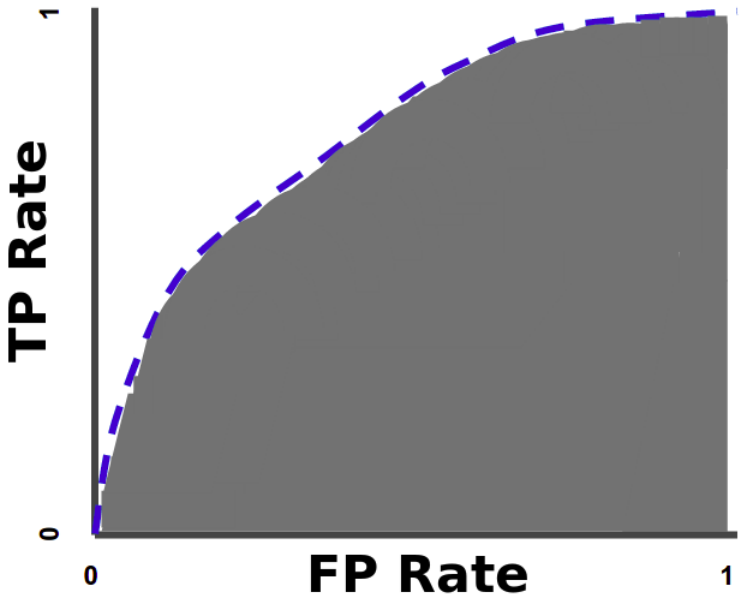


Figure: Area Under the ROC Curve. Credit: Google Developer

AUC = Area Under the ROC Curve

Characteristics of AUC:

- AUC is **scale-invariant**. It measures how well predictions are ranked, rather than their absolute values.
- AUC is **classification-threshold-invariant**. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.

How AUC is interpreted:

- AUC is as the probability that the model ranks a random positive example more highly than a random negative example.
- AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

ROC & AUC

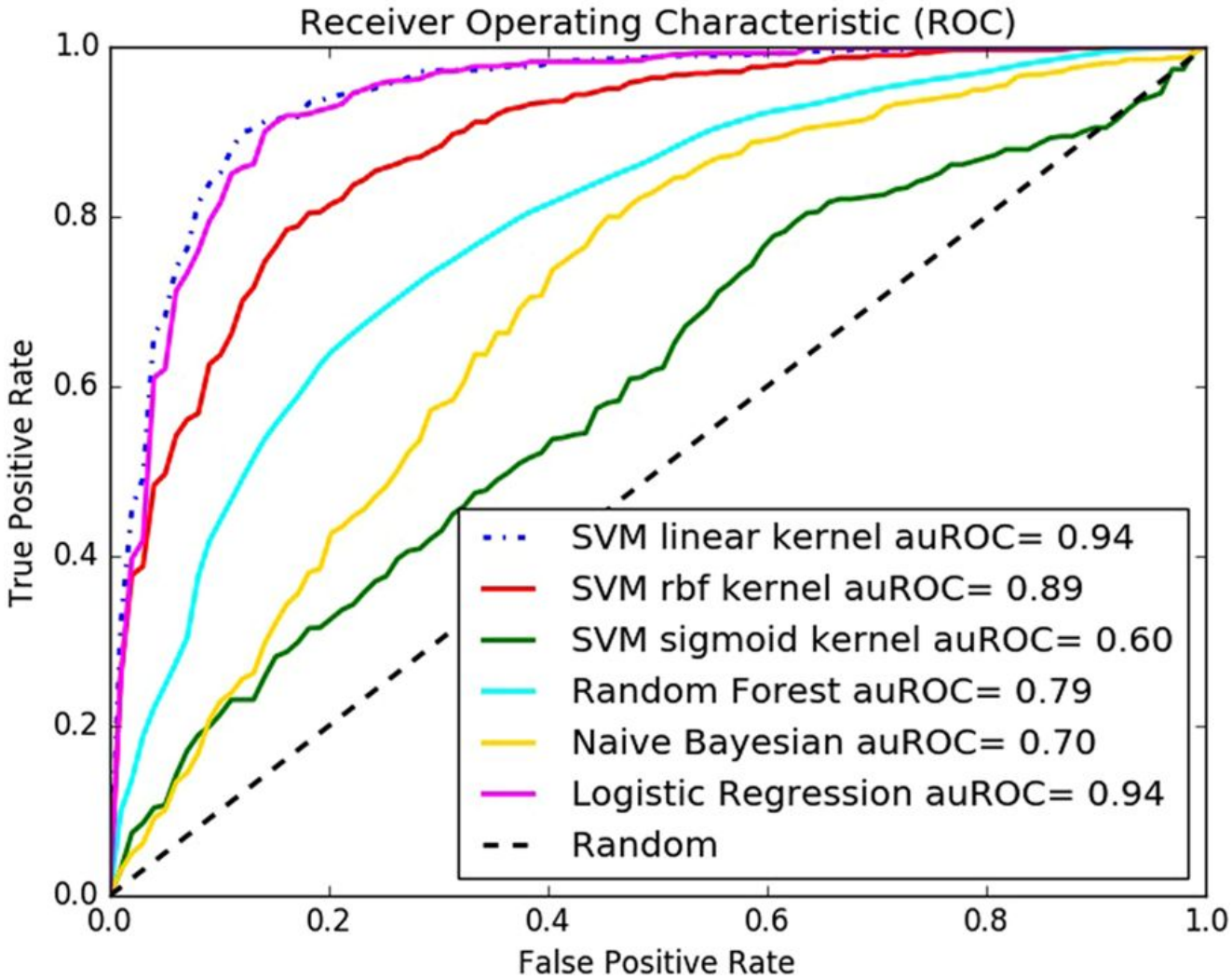
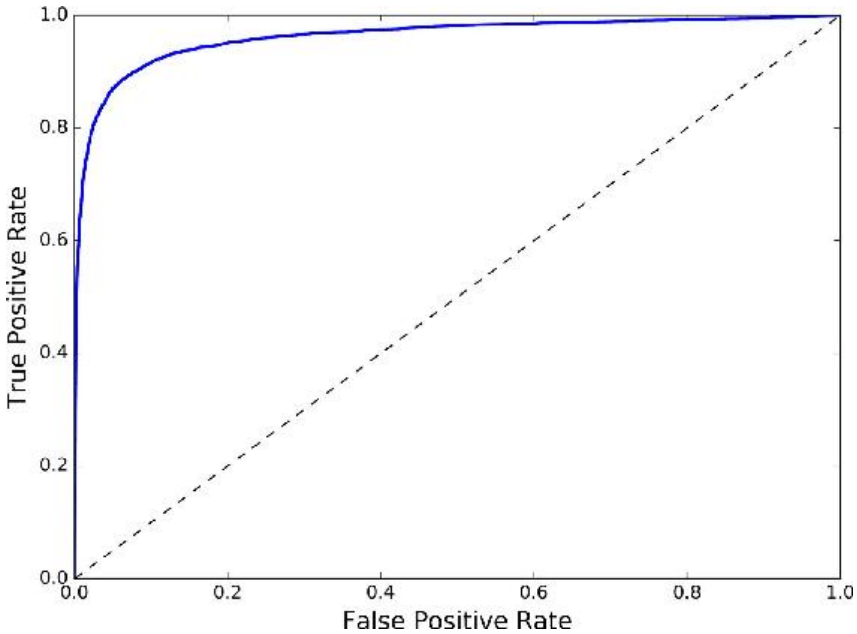
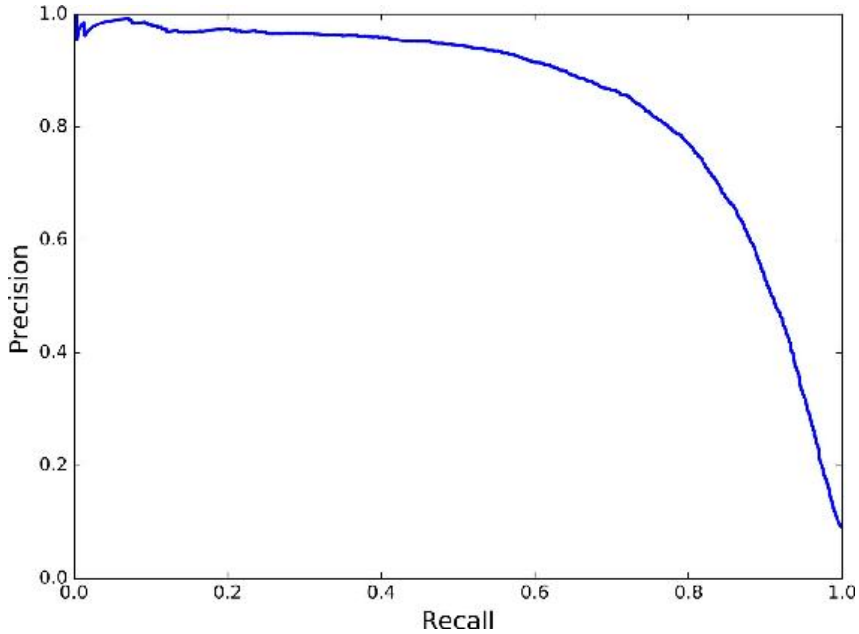


Figure: Model Benchmarks. Credit: Nature

ROC & Precision-Recall Curve



(a) ROC Curve (AUC ≈ 0.97)



(b) PR Curve

Figure: (a) or (b) or both?

Gini Index

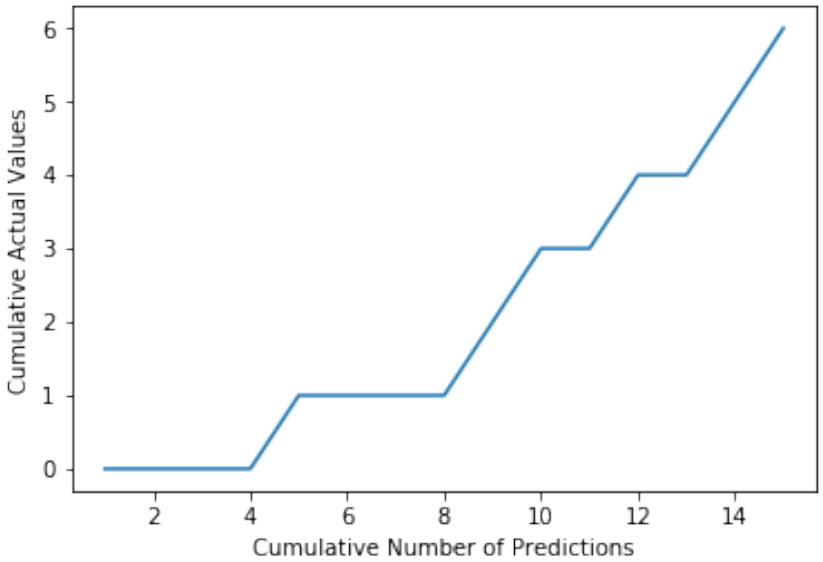
Definition

The Gini coefficient is an empirical measure of classification performance based on the area under an ROC curve (AUC).

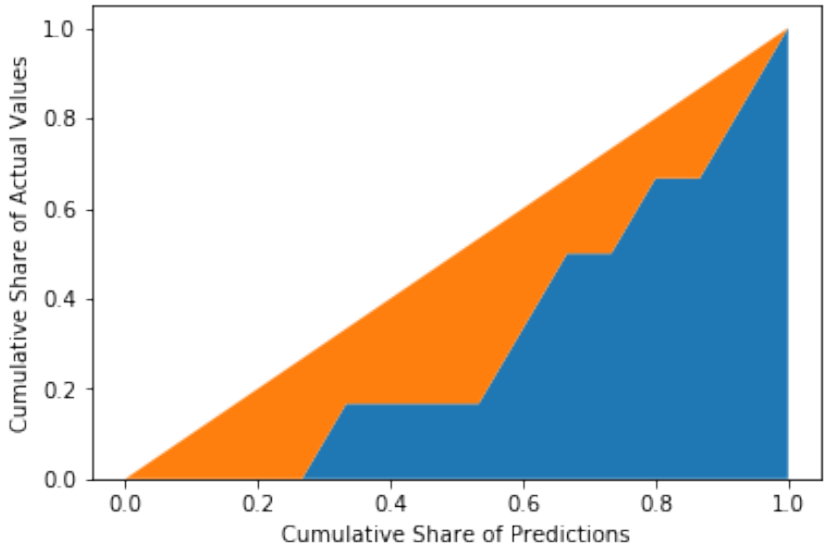
$$\text{Gini Index} = 2 \times \text{AUC} - 1$$

- Ranges [-1, 1], and higher values indicate better model.
- Used in financial modeling scenarios such as credit scoring

Gini Index



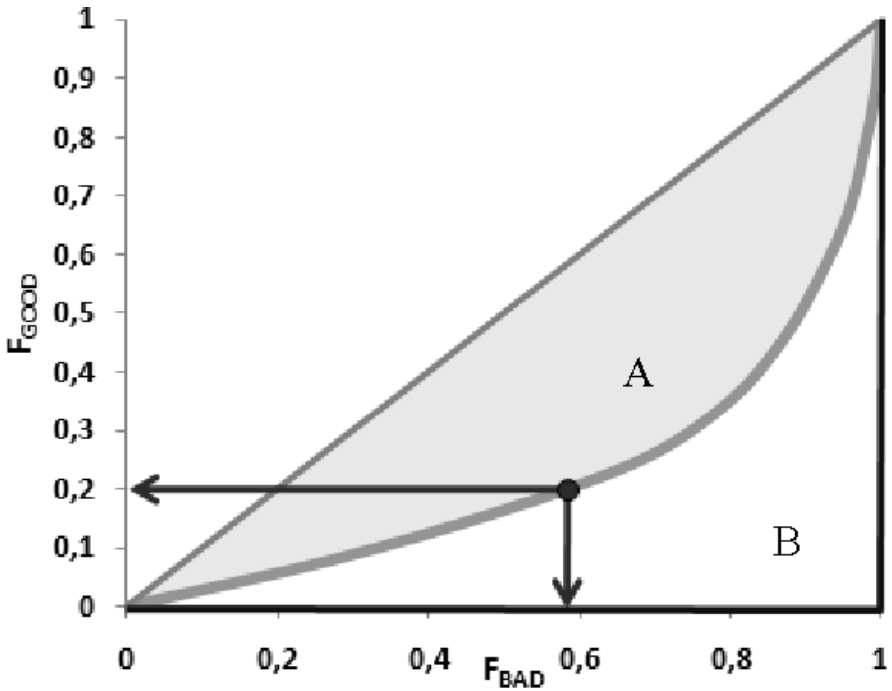
(a) Lorenz Curve



(b) Random guessing and Model

Figure: Gini Index $= \frac{\text{Orange Area}}{\text{Blue Area}}$. Credit: batzner

Gini Index in Credit Scoring



$\text{GI} = \frac{A}{B}$

$\text{GI} = 1 \rightarrow$ Scoring function perfectly separates

$\text{GI} = 0 \rightarrow$ Scoring function assigns ramdomly

Kolmogorov-Smirnov statistic

Definition

The **Kolmogorov-Smirnov statistic** (K-S statistic) is the performance measure that captures the separation between the distribution of prediction scores for the different target levels in a classification problem.

How to calculate K-S statistic:

- Determine the Cumulative probability distributions of the prediction scores for each classes.
- Plot the CP on the Kolmogorov-Smirnov chart (K-S chart)
- The K-S statistic is calculated by determining the maximum difference between CP.

Kolmogorov-Smirnov statistic

- Cumulative probability distribution:

$$CP(positive, ps) = \frac{\text{no. positive test instances with score } \leq ps}{\text{no. positive test instances}}$$

$$CP(negative, ps) = \frac{\text{no. negative test instances with score } \leq ps}{\text{no. negative test instances}}$$

- K-S statistic:

$$\text{K-S} = \max_{ps}(CP(positive, ps) - CP(negative, ps))$$

Kolmogorov-Smirnov statistic

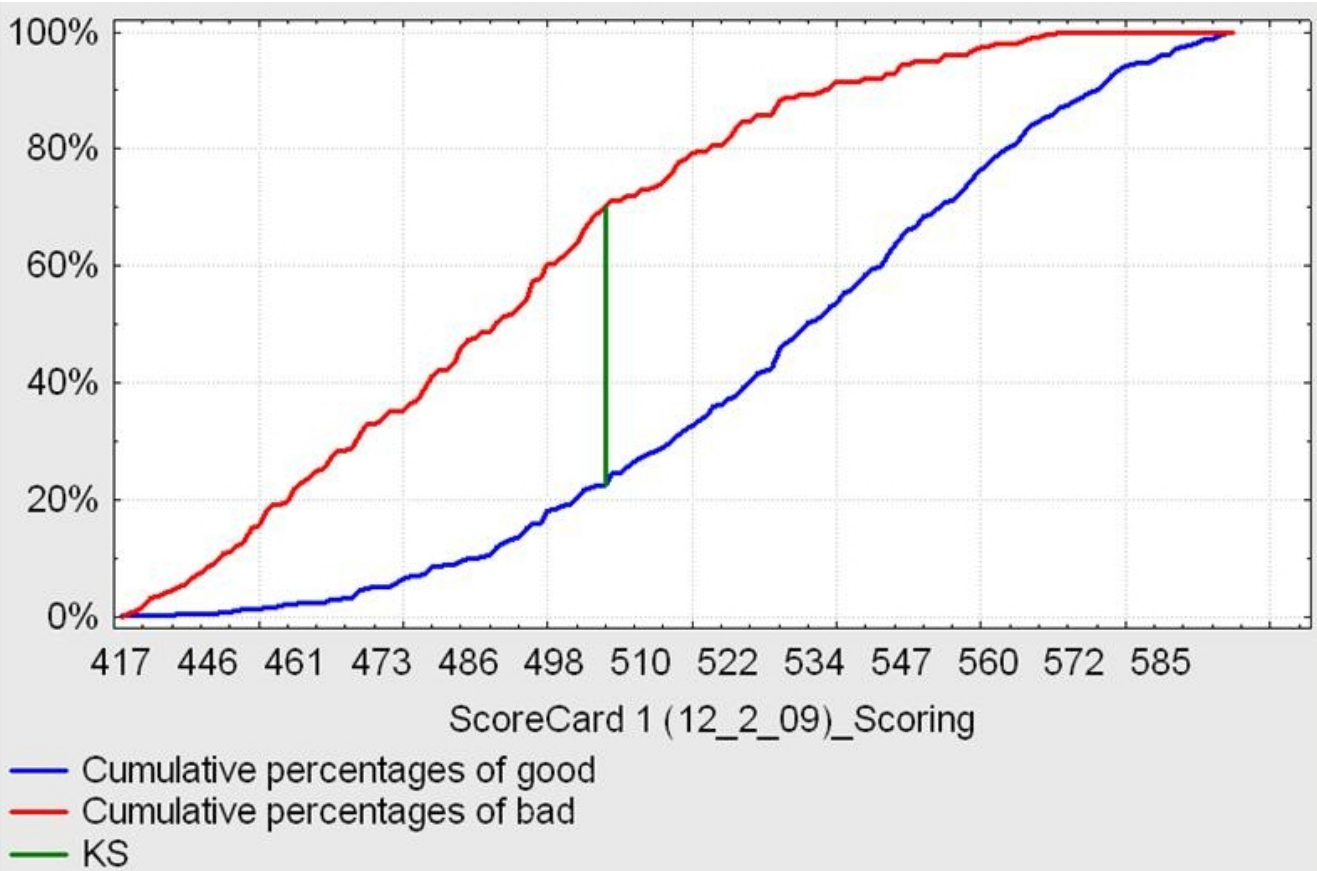


Figure: Kolmogorow-Smirnow chart. Credit: Ong Xuan Hong

Multiclass Classification

- Cohen's kappa
- Confusion matrix
- Hinge loss
- Matthews correlation coefficient (MCC)

Cohen's kappa

Cohen's kappa is a statistic that measures inter-annotator agreement, expresses the level of agreement between two annotators on a classification problem.

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

where:

- p_o: the observed accuracy
- p_e: the expected accuracy based on the marginal totals of the confusion matrix.

Cohen's kappa

- Ranges between -1 and 1
- $\kappa = 0$ means means there is no agreement between the observed and predicted classes
- $\kappa = 0$ means perfect concordance of the model prediction and the observed classes.
- Negative values indicate that the prediction is in the opposite direction of the truth
- When the class distributions are equivalent, overall accuracy and Kappa are proportional.
- Depending on the context, Kappa values within 0.30 to 0.50 indicate reasonable agreement.

Confusion Matrix

Confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group but predicted to be in group.

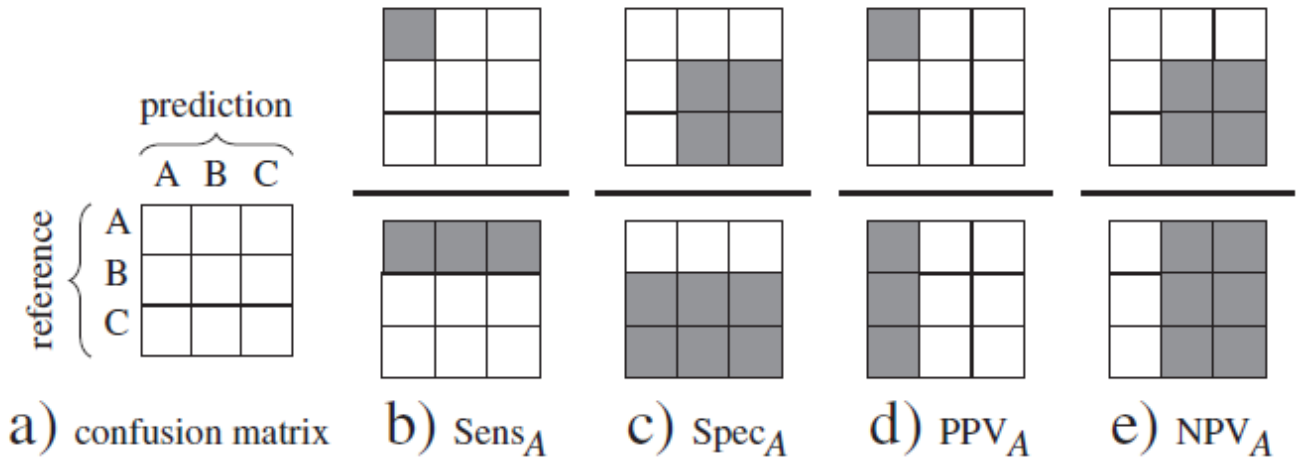


Figure: Confusion Matrix for Multiclass classification. Credit: softclassval

Confusion Matrix

	Predicted			
Actual		A	B	C
	A	TP_A	E_{AB}	E_{AC}
	B	E_{BA}	TP_B	E_{BC}
	C	E_{CA}	E_{CB}	TP_C

Predicted class

Actual class	P	N
P	*TP*	*FN*
N	*FP*	*TN*

	Predicted		
Actual		A	Not A
	A	TP_A	$E_{AB} + E_{AC}$
	Not A	$E_{BA} + E_{CA}$	$TP_B + E_{BC}$ $E_{CB} + TP_C$

	Predicted		
Actual		B	Not B
	B	TP_B	$E_{BA} + E_{BC}$
	Not B	$E_{AB} + E_{CB}$	$TP_A + E_{AC}$ $E_{CA} + TP_C$

	Predicted		
Actual		C	Not C
	C	TP_C	$E_{CA} + E_{CB}$
	Not C	$E_{AC} + E_{BC}$	$TP_A + E_{AB}$ $E_{BA} + TP_B$

Figure: Confusion Matrix for Multiclass classification. Credit: Tilani Gunawardena

Confusion Matrix

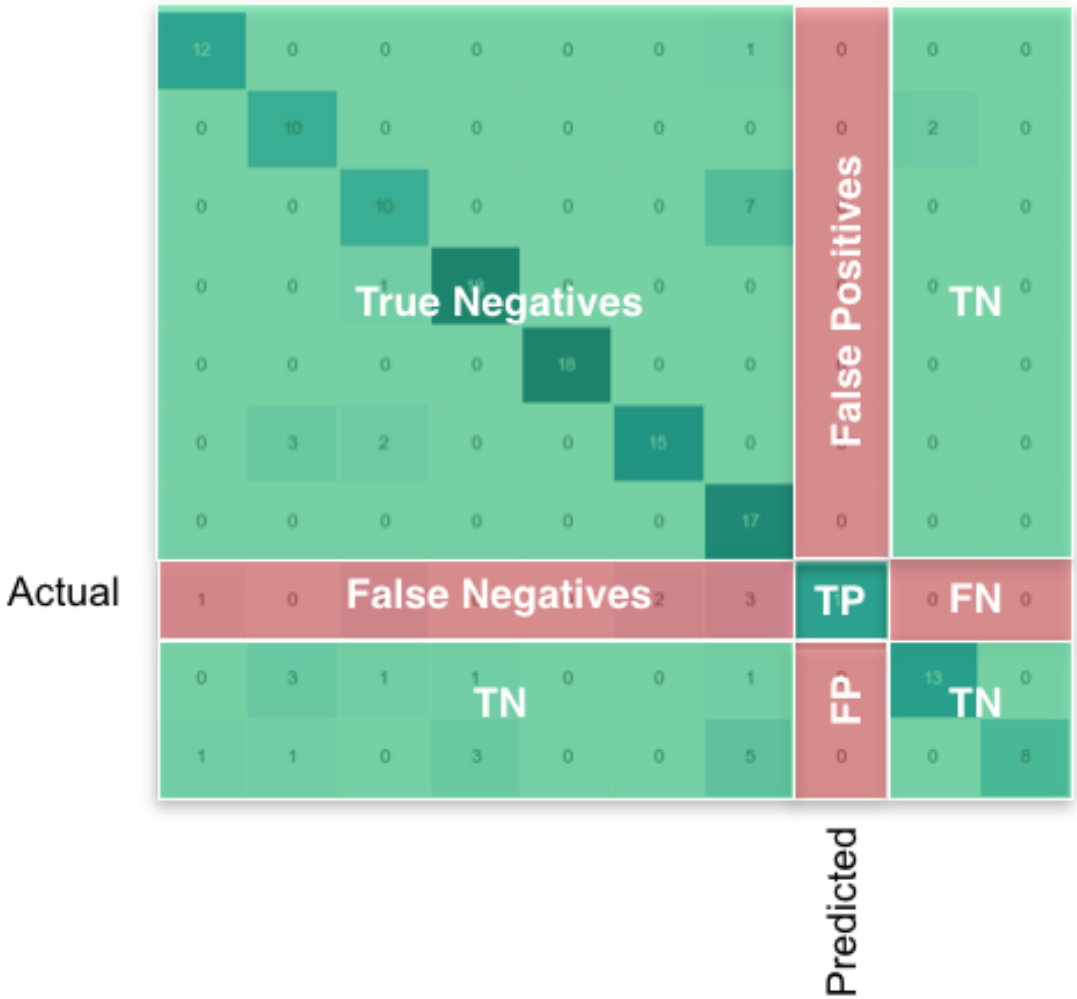


Figure: Confusion Matrix for Multiclass classification.

Confusion Matrix

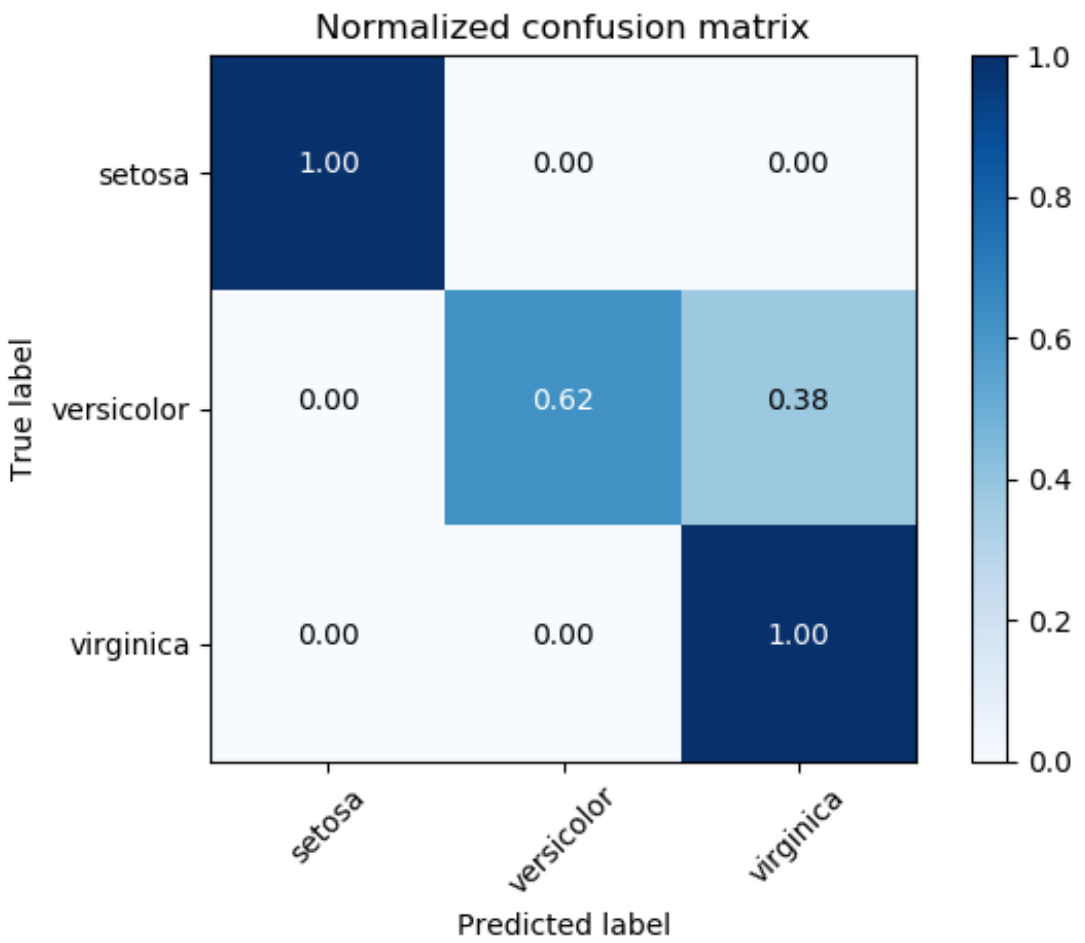


Figure: Confusion Matrix for Multiclass classification.

Multi-label Classification

- Example-based measures
  - Subset Accuracy
  - Hamming Loss
- Label-based measures
  - Macro-B
  - Micro-B

Multi-label Example-based measures

- **Subset Accuracy** evaluates the proportion of test examples whose predicted label set coincides with the ground-truth label set.

$$\frac{1}{p}\sum_{i=1}^{p}[[h(x_i) = Y_i]]$$

with $[[\pi]]$ returns 1 if predicate π holds and 0 otherwise.

- **Hamming Loss** evaluates the proportion of misclassified instance-label pairs.

$$\frac{1}{p}\sum_{i=1}^{p}\frac{1}{q}|h(x_i)\Delta Y_i|$$

with Δ is symmetric difference; $|.|$ is the cardinaltily of a set.

Multi-label Example-based measures

```
>>> import numpy as np
>>> from sklearn.metrics import accuracy_score
>>> y_pred = [0,2,1,3]
>>> y_true = [0,1,2,3]
>>> accuracy_score(y_true, y_pred)
0.5
>>> accuracy_score(y_true, y_pred, normalize = False)
2
>>> accuracy_score(np.array([[0,1],[1,1]]),
                   np.array([[1,1],[1,1]]),
                   normalize = False)
1
>>> from sklearn.metrics import hamming_loss
>>> hamming_loss(y_true, y_pred)
0.5
```

Multilabel Example-based measures

- For Hamming loss, the smaller the value, the better the generalization performance.
- For Subset Accuracy, the larger the value, the better the performance.

Multilabel Classification - Label-based measures

On each label y_i, four basic quantitiesregarding the test examples are commonly used: TP_j (True Positive), FP_j (False Positive), TN_j (True Negative), and FN_j (False Negative).
Let $B(TP_j; FP_j; TN_j; FN_j)$ denote a certain binary classification measure:

- $Macro - B = \sum_{j=1}^{q} \frac{1}{q} B(TP_j; FP_j; TN_j; FN_j)$ (assuming equal importance for each label)
- $Micro - B = B(\sum_{j=1}^{q} TP_j; \sum_{j=1}^{q} FP_j; \sum_{j=1}^{q} TN_j; \sum_{j=1}^{q} FN_j)$ (assuming equal importance for each example)

Among popular choices of $B \in \{accuracy, precision, recall, F\}$, the larger the macro/micro-B value, the better the performance.

Workflow

- Training phase: selecting algorithms
- Validation phase: model selection
  - Selecting between multiple methods
  - Fine-tuning parameters (Model complexity and Regularization)
  - Feature selection: No. of input variables and the Correct input variable
- Test phase: model assessment

Selecting algorithms

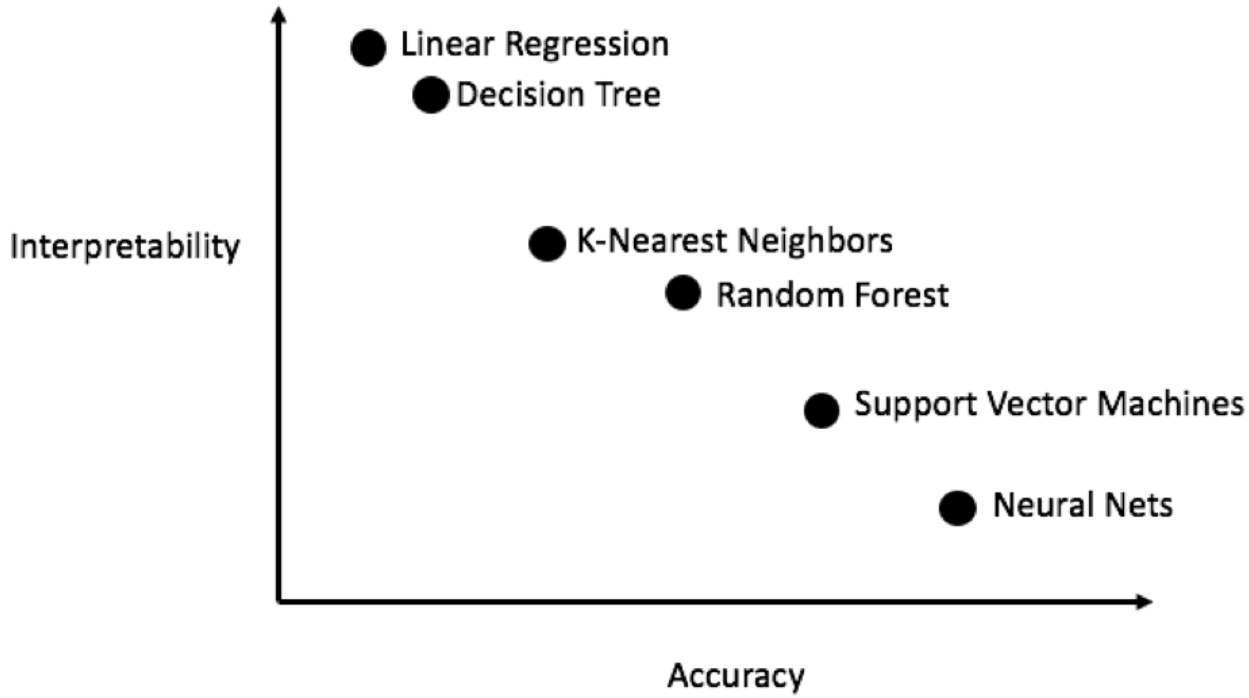


Figure: Accuracy-Interpretability trade-off. Credit: Ansaro

Goals

- **Model selection**: estimating the performance of different models to choose the best model.
- **Model assessment**: having chosen a final model, estimating its prediction error (generalization error) on new data.

Data-rich situation

Randomly divide the dataset into three parts:

- Training set: fit the model
- Validation set: estimate prediction error for *model selection*
- Test set: assessment of the generalization error of the final chosen model

Training/Validation/Test sets splitting

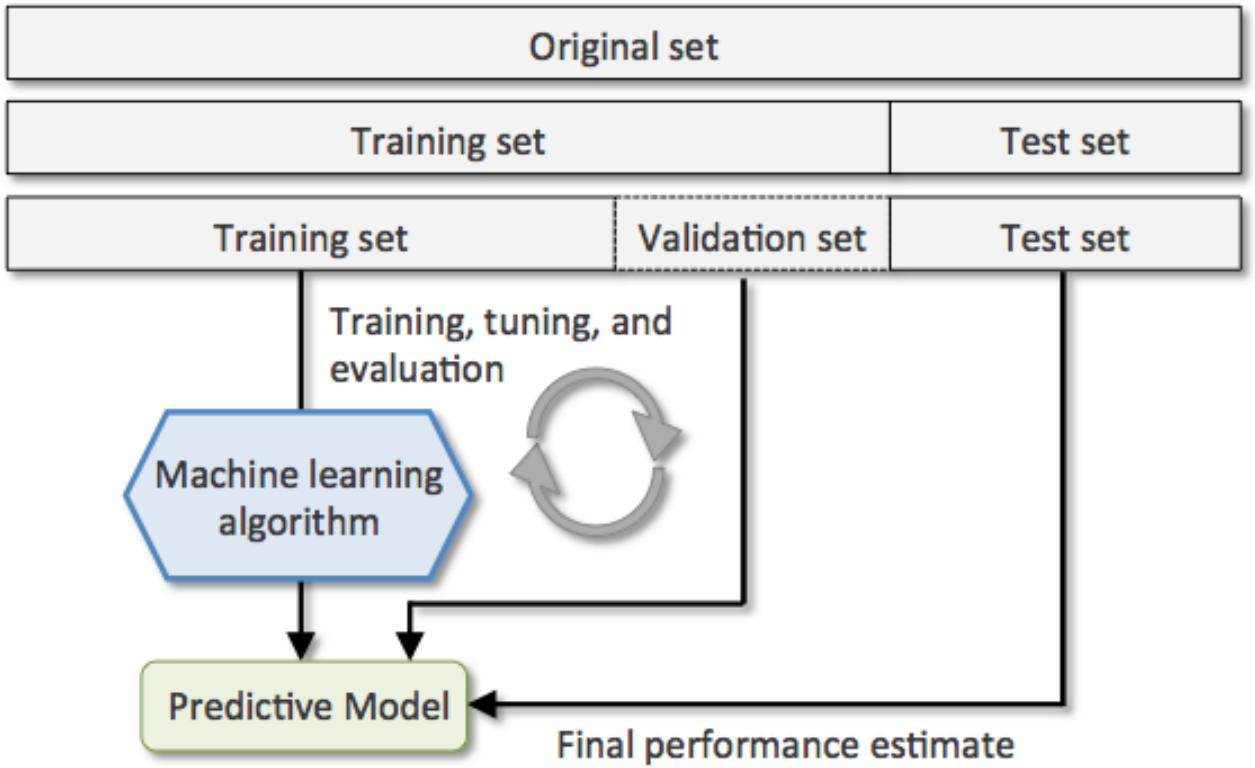


Figure: Training/Validation/Test sets splitting (Credit: Shan-Hung Wu & DataLab)

Training/Validation/Test sets splitting

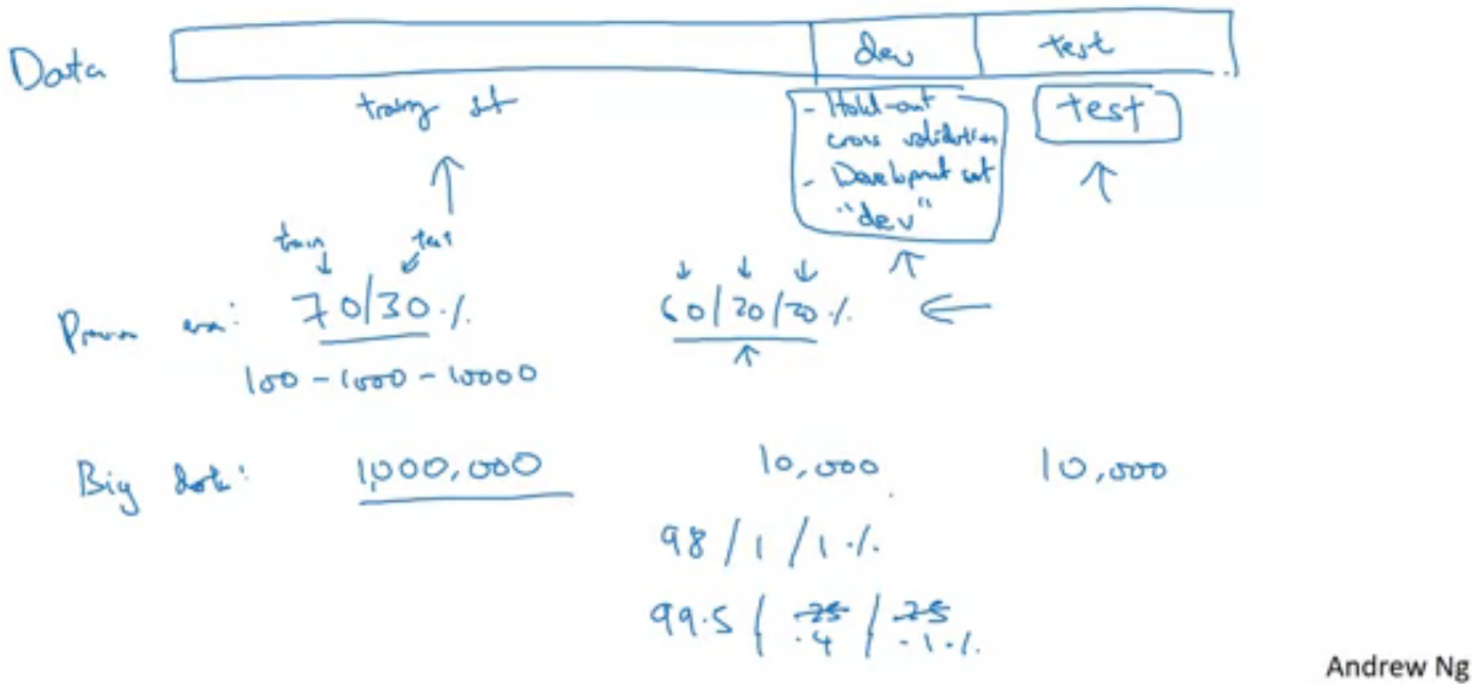


Figure: Training/Dev/Test sets splitting (Credit: Corner)

Stratified sampling

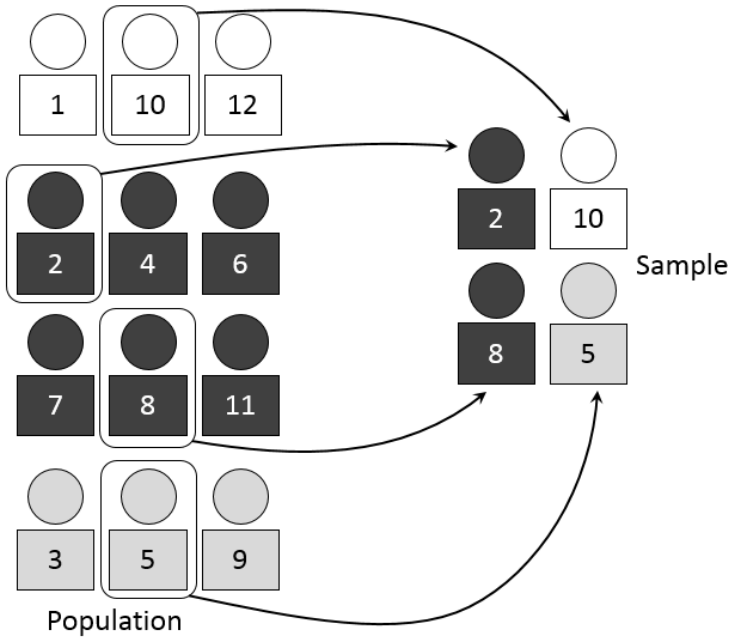


Figure: Stratified sampling. Credit: Wikipedia

Insufficient data

- Too difficult to give a general rule on how much training data
- Depends on the signal-to-noise ratio of the underlying function
- Depends on the complexity of the models

Insufficient data

In order to select the best model with respect to test error, we need to estimate this test error by:

- *Indirectly* estimate test error by making an *adjustment* to the training error to account for the bias due to overfitting
- *Directly* estimate the test error using efficient sample re-use (cross-validation and the bootstrap)

C_p statistic

The C_p estimate of the test MSE is computed using the equation

$$C_p = \frac{1}{n}(RSS + 2d\hat{\sigma}^2)$$

where $\hat{\sigma}^2$ is an estimate of $Var[\epsilon]$ [1]

- C_p statistics adds a penalty of $2d\hat{\sigma}^2$ to the training RSS
- C_p tends to take a small value of models with a low test error. The model with the lowest C_p value should be chosen.

[1] $Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon]$

Indirect estimate

For linear regression model with d predictors:

Akaike information criterion

Akaike information criterion or AIC is given by

$$AIC = \frac{1}{n\sigma^2}(RSS + 2d\hat{\sigma}^2)$$

Bayesian information criterion

Bayesian information criterion or BIC is given by

$$BIC = \frac{1}{n\sigma^2}(RSS + \log(n)d\hat{\sigma}^2)$$

Adjusted R^2

For a least squares model with d variables, the adjusted R^2 is calculated as

$$Adjusted\ R^2 = 1 - \frac{RSS/(n-d-1)}{TSS(n-1)}$$

Unlike C_p , AIC, and BIC, for which a small value indicates a model with a low test error, a large value of adjusted R^2 indicates a model with a small test error.

Indirect estimate

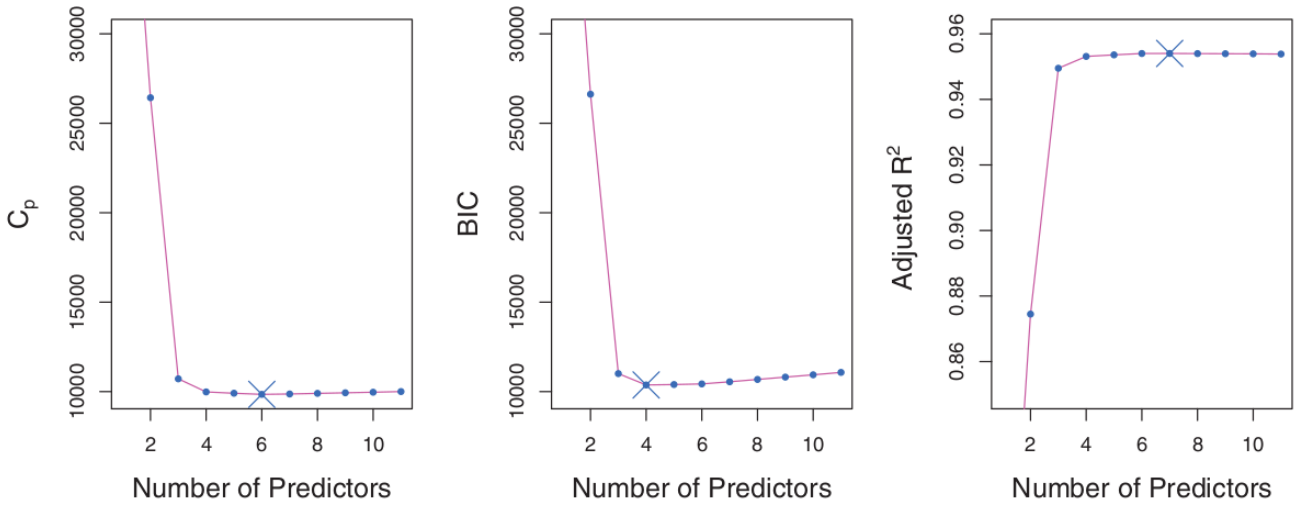


Figure: C_p, BIC and adjusted R^2. Credit: ISL book

K-fold Cross-validation

- The simplest and most popular way to estimate the test error.
- How to do:
  1. Randomly split the data into K roughly equal parts.
  2. For each k:
     - Leave the kth part out, fit the model using the other (k-1) parts, called $\hat{f}^{(-k)}(x)$
     - Calculate the performance or prediction error of the $\hat{f}^{(-k)}(x)$ on the kth part.
  3. Average the errors
- Leave-one-out Cross-validation or LOOCV is a special case of k-fold cross-validation with k = N

K-fold Cross-validation

Figure: 10-fold Cross Validation. Credit: Shan-Hung Wu & DataLab

K-fold Cross-validation

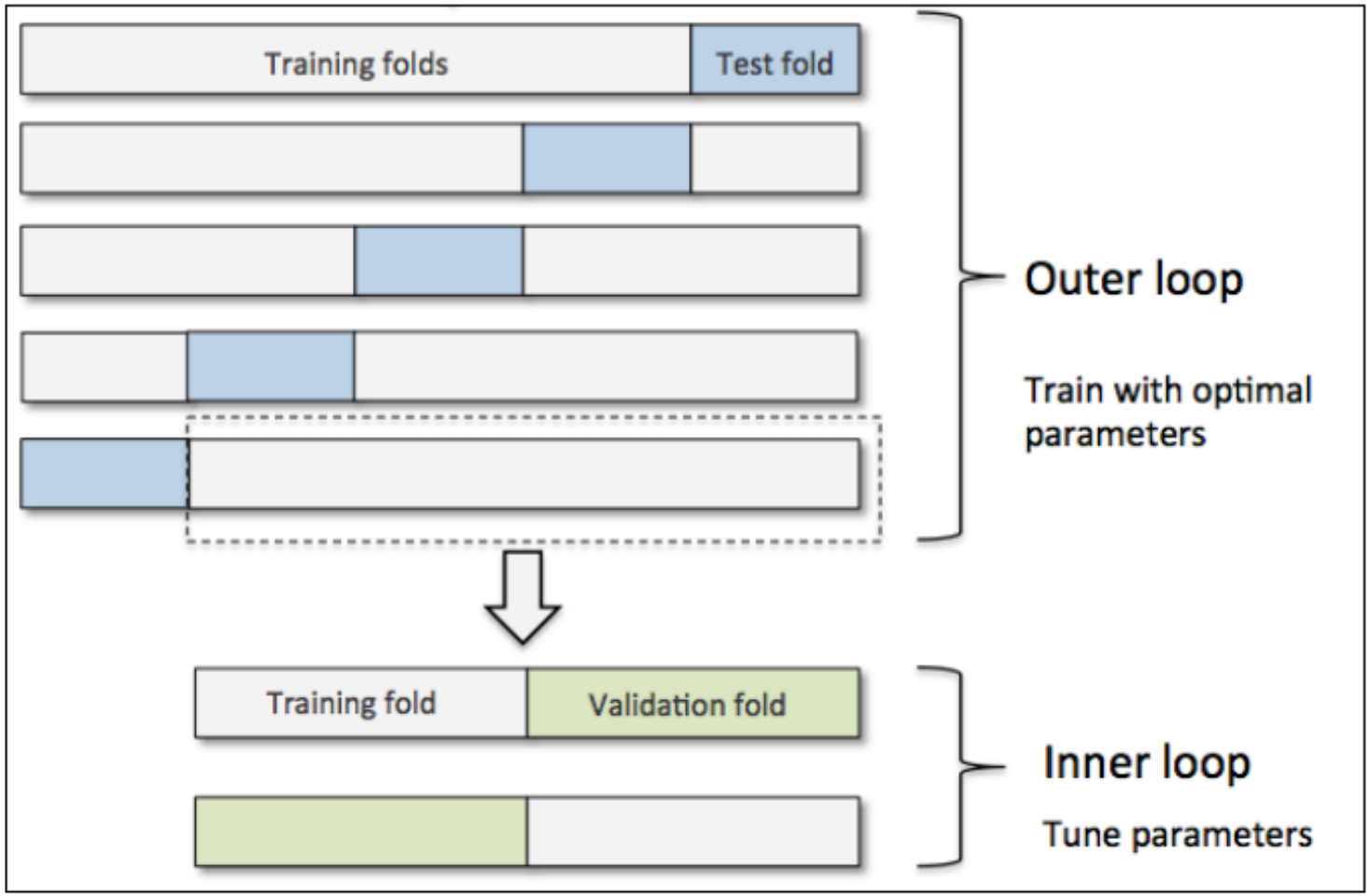


Figure: 5 by 2-fold Cross Validation. Credit: Shan-Hung Wu & DataLab

K-fold Cross-validation

The correct way to carry out cross-validation:

1. Divide the samples into K cross-validation folds (groups) at random.
2. For each fold:
   - Find a subset of "good" predictors that show fairly strong (univariate) correlation with the class labels, using all of the samples except those in fold k.
   - sing just this subset of predictors, build a multivariate classifier, using all of the samples except those in fold k.
   - Use the classifier to predict the class labels for the samples in fold k.

K-fold Cross-validation

- In practice, $k = 5$ or $k = 10$ are recommended.
- "One-standard error" rule could be used with CV.

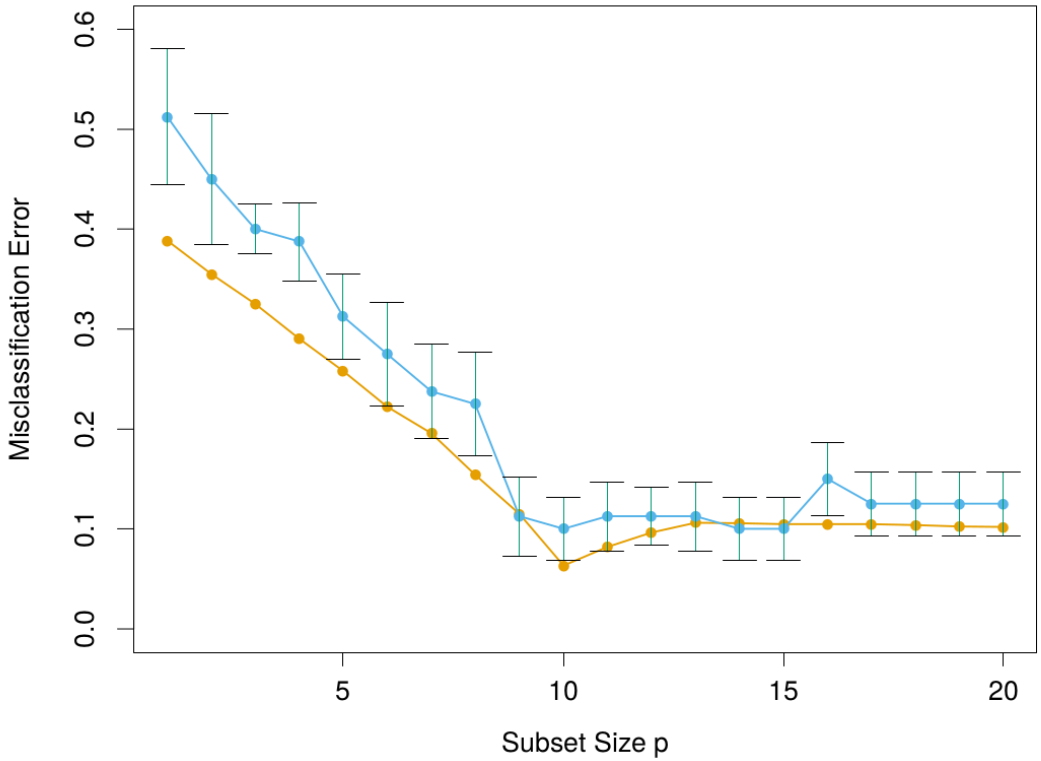
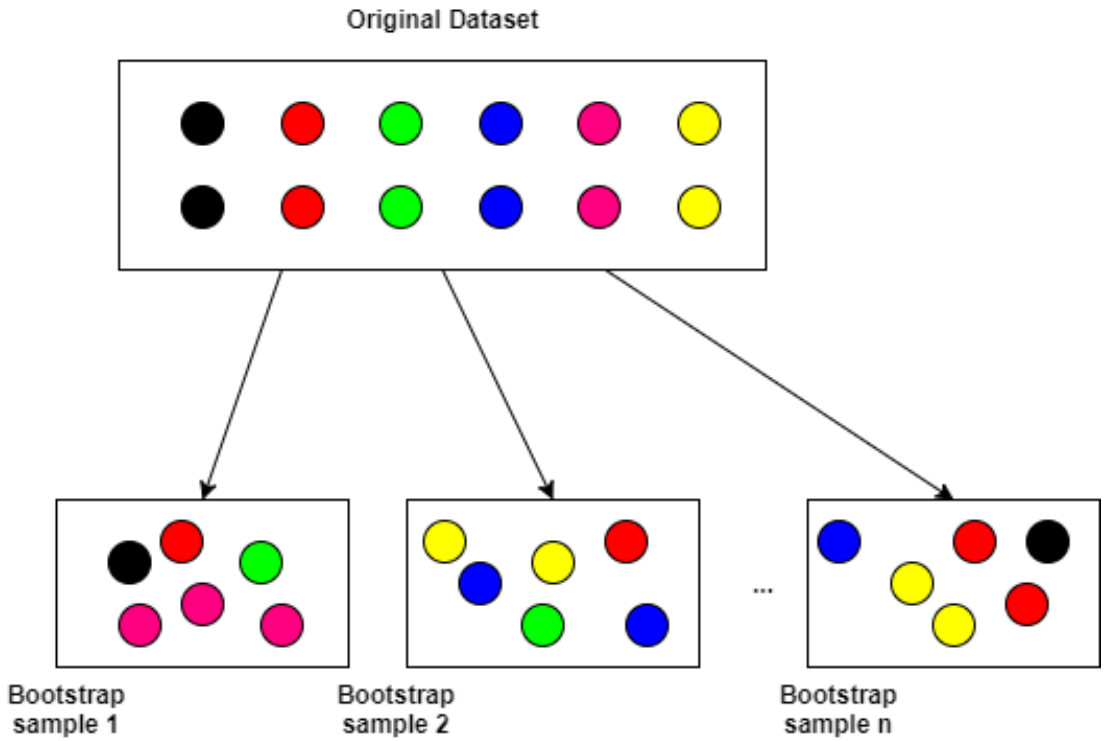


Figure: Prediction error (orange) and tenfold cross-validation curve (blue) estimated from a single training set.A model with $p = 9$ would be chosen. Credit: ESL

Bootstrap

A **bootstrap sample** is a random sample of the data taken *with replacement*. This means that, after a data point is selected for the subset, it is still available for further selection.
The unselected samples is called Out-of-bag samples.



How Bootstrap estimate Prediction error?

Predict on the original dataset

$$\hat{Err}_{boot} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{i=1}^{N} L(y_i, \hat{f}_{*b}(x_i))$$

Overlap between bootstrap set and original set can make overfit predictions look unrealistically good

Leave-one-out Bootstrap

$$\hat{Err}_{LOOB} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} L(y_i, \hat{f}_{*b}(x_i))$$

where:

- C^{-i} is the set of indices of the bootstrap samples b that do not contain observation i.
- Bootstrap sample need to be large enough to ensure $|C^{-i}| > 0 \forall i$

How Bootstrap estimate Prediction error?

".653 estimator"

$$\hat{Err}_{.632} = 0.368 \times \bar{err} + 0.632 \times \hat{Err}_{LOOB}$$

The .632 estimator works well in "light fitting" situations, but can break down in overfit ones.

How Bootstrap estimate Prediction error?

".653+ estimator"

$$\hat{Err}_{.632} = (1 - \hat{w})\bar{err} + \hat{w} \times \hat{Err}_{LOOB}$$

with

$$\hat{w} = \frac{0.632}{1 - 0.368\hat{R}}$$

$$\hat{R} = \frac{\hat{Err}_{LOOB} - \bar{err}}{\hat{\gamma} - \bar{err}}$$

$$\hat{\gamma} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} L(y_i, \hat{f}(x_{i'}))$$

γ: no-information error rate; $\hat{R}$: relative overfitting rate

Feature Importance

- Measure the strength or relevance of the predictors is to filter which should be used as inputs in a model.
- Many variable importance scores are specific to the type of data.
  - Numeric outcomes
  - Categorical outcomes

OLS

```
In [7]: print(res.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.416
Model:                            OLS   Adj. R-squared:                  0.353
Method:                 Least Squares   F-statistic:                     6.646
Date:                Mon, 14 May 2018   Prob (F-statistic):           0.00157
Time:                        21:48:12   Log-Likelihood:                -12.978
No. Observations:                  32   AIC:                             33.96
Df Residuals:                      28   BIC:                             39.82
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             0.4639      0.162      2.864      0.008       0.132       0.796
x2             0.0105      0.019      0.539      0.594      -0.029       0.050
x3             0.3786      0.139      2.720      0.011       0.093       0.664
const         -1.4980      0.524     -2.859      0.008      -2.571      -0.425
==============================================================================
Omnibus:                        0.176   Durbin-Watson:                   2.346
Prob(Omnibus):                  0.916   Jarque-Bera (JB):                0.167
Skew:                           0.141   Prob(JB):                        0.920
Kurtosis:                       2.786   Cond. No.                         176.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Figure: Sample output of a LR model by statsmodels package.
Credit: statsmodels

Random Forest

```
>>> from sklearn.datasets import load_boston
>>> from sklearn.ensemble
    import RandomForestRegressor
>>> import numpy as np
>>> boston = load_boston()
>>> X = boston["data"]
>>> Y = boston["target"]
>>> names = boston["feature_names"]
>>> rf = RandomForestRegressor()
>>> rf.fit(X, Y)
>>> print("Features sorted by their score:")
>>> print(sorted(zip(map(lambda x: round(x, 4),
    rf.feature_importances_), names), reverse=True))
Features sorted by their score:
[(0.4554, 'RM'), (0.3701, 'LSTAT'), (0.0459, 'DIS'),
 (0.0308, 'NOX'), (0.0292, 'CRIM'), (0.0233, 'TAX'), ...
```

Numerical Outcomes

- Correlation statistic
- Using flexible methods to model general nonlinear relationships (eq: LOESS)
- The maximal information coefficient - MIC
- When the predictors are categorical, evaluate whether the average outcome in each category is different (t-statistics or ANOVA).

Correlation statistic

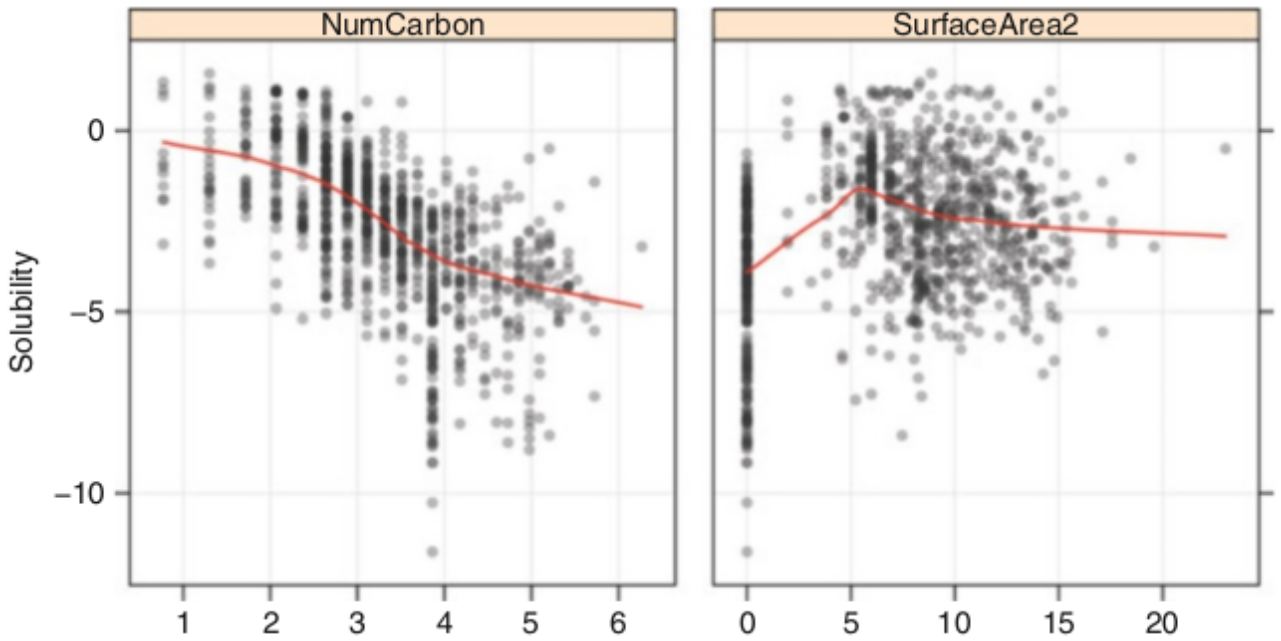


Figure: $R_1^2 = 0.37,\ R_2^2 = 0.07,\ pseudo - R_2^2 = 0.22$
With pseudo-R squared, SurfaceArea2 improves the rank from 17/20 to 7/20.
Credit: Applied predicitive model

The maximal information coefficient

Maximum of Mutual information over all possible grid

$$I^*(D, x, y) = \max I(D|_G)$$

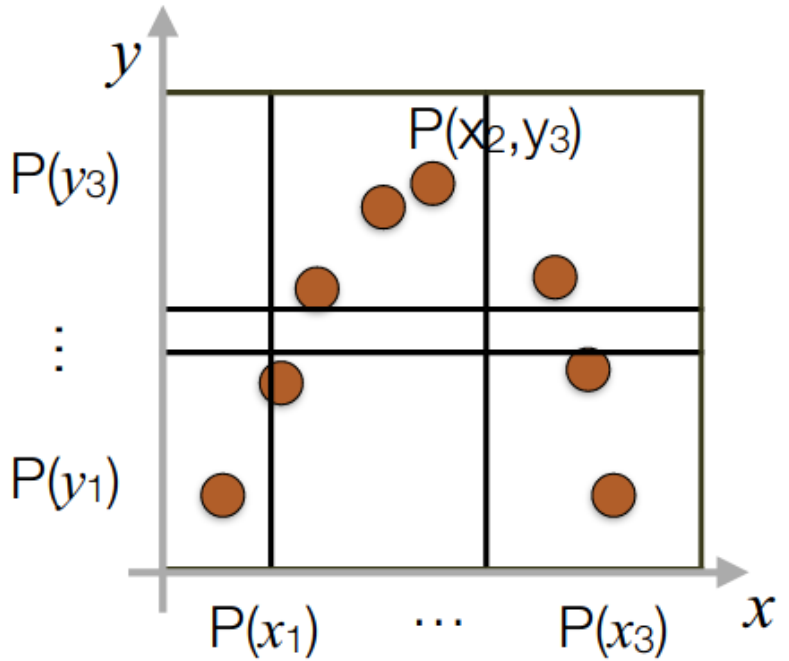
$$I(X;Y) = \int_Y \int_X p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right) dxdy$$

Characteristic Matrix

$$M(D)_{x,y} = \frac{I^*(D, x, y)}{\log \min\{x, y\}}$$

Maximal Information Coefficient

$$\mathbf{MIC}(D) = \max_{xy<B(n)} \{M(D)_{x,y}\}$$



t-statistic

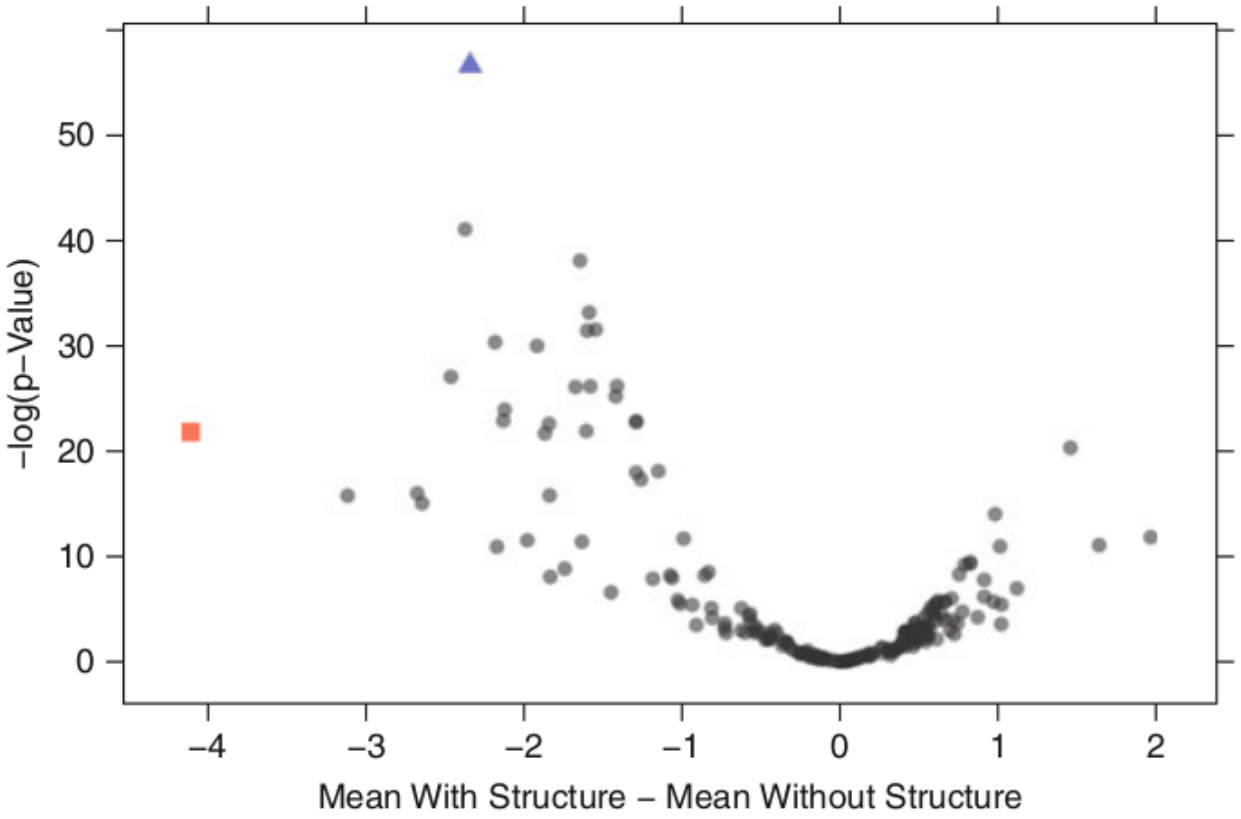
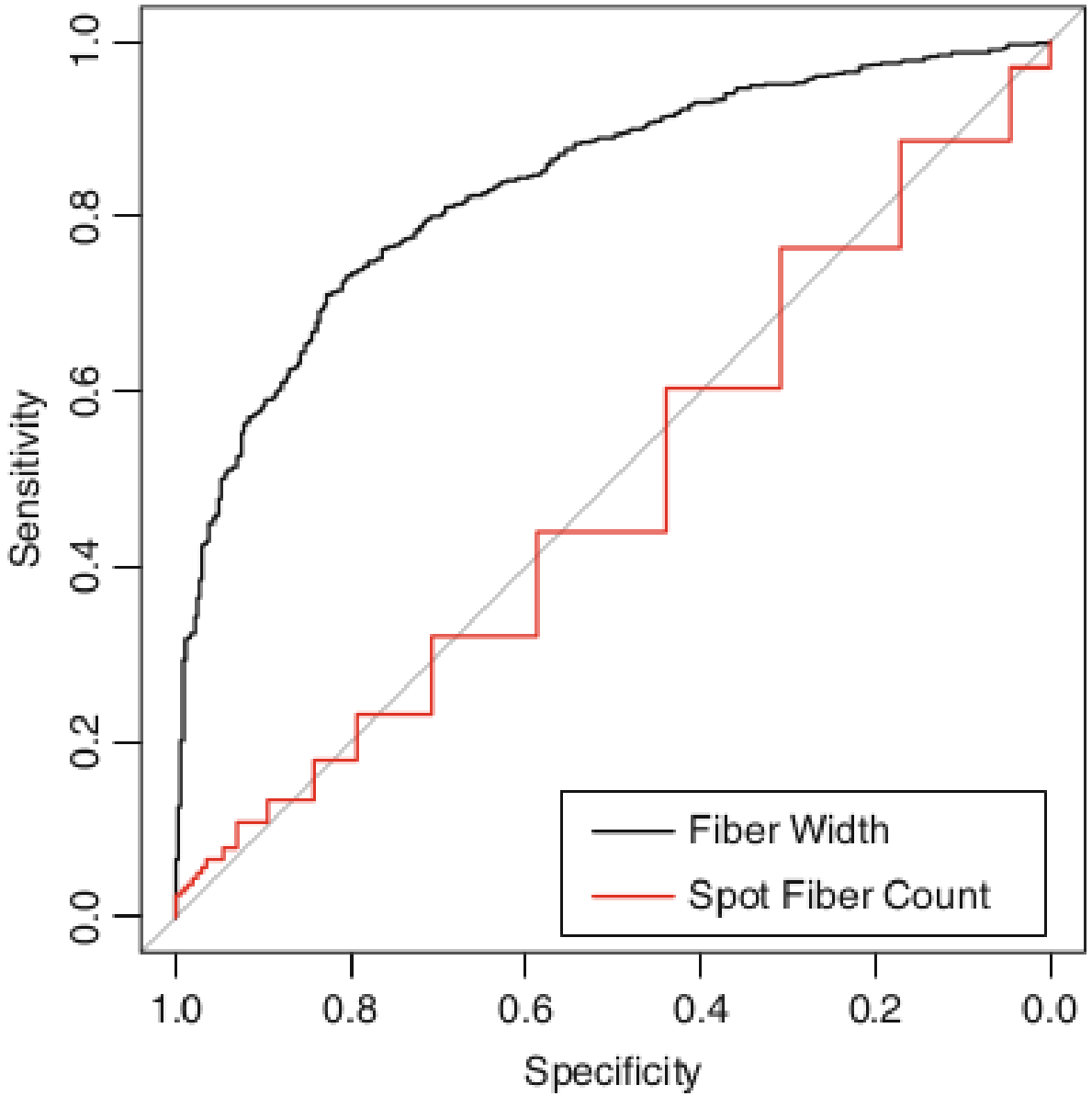


Figure: The red square has the largest difference in means, while the blue triangle is the most statistically significant. Credit: Applied predicitive model

Categorical Outcomes

- AUC - the Area under the ROC curve
- t-statistic, Odds ratio
- Information gain ratio
- The maximal information coefficient

AUC


Sensitivity
1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.8
0.6
0.4
0.2
0.0
Specificity
Fiber Width
Spot Fiber Count

Categorical Outcomes

	Grant success					
	Yes	No	%	OR	p-value	Gain ratio
Sponsor						
62B	7	44	14			
Other	3226	3356	49	6.0	$2.6e^{-07}$	0.04726
CVB						
Band unknown	644	2075	24			
Band known	2589	1325	66	6.3	$1.7e^{-263}$	0.13408
RFCD code						
240302	13	15	46			
Other code	3220	3385	49	1.1	$8.5e^{-01}$	0.00017

References

[1] Vu Huu Tiep, Gioi thieu ve Feature Engineering, https://machinelearningcoban.com/general/2017/02/06/featureengineering/
[2] VEF Academy, Machine Learning, 2019