Supervised Learning - I

M. Vazirgiannis

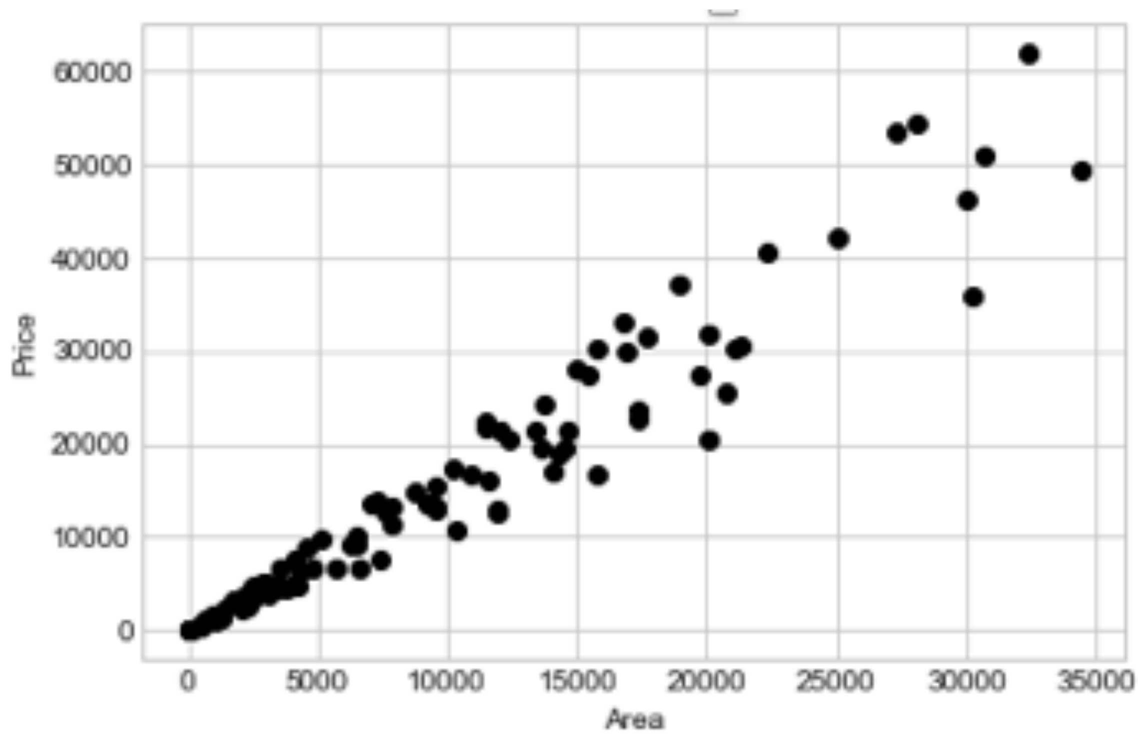
DaSciM Data Science and Mining Team École Polytechnique, LIX, École Polytechnique

September 2024

Outline

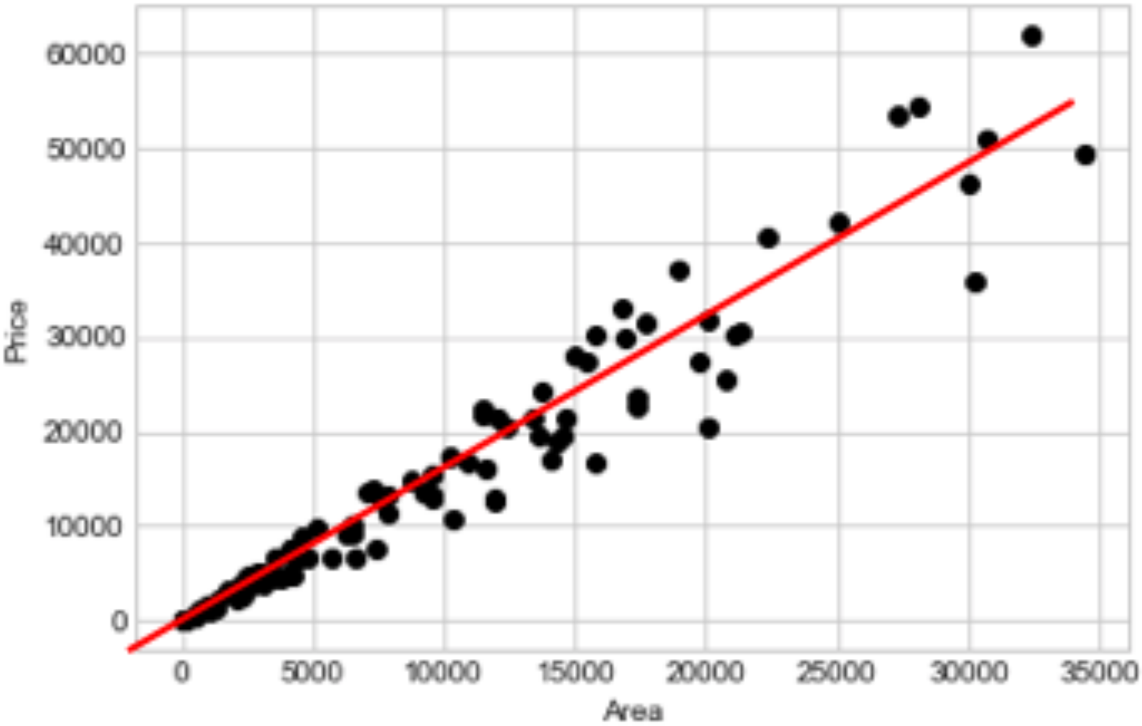
- Introduction to supervised learning
- K-nn
- Naïve Bayes
- Logistic Regression

Supervised Learning



Can we predict the price of a parcel based on its size (surface in $m\hat{2}$) ?

Supervised Learning



Can we predict the price of a parcel based on its size (surface in $m\hat{2}$) ?

Supervised learning

- $X = X_1, X_2, ..., X_p$: input variables (features)
- Y: "output" or target variable that we try to predict
- A pair (x_i, y_i) is called a training example, where:

$$x_i \in X = X_1 \times ... \times X_p \text{ and } y_i \in Y$$

- The training set is a list of training examples:

$$\{(x_i, y_i)\ ;\ i = 1...m\}$$

- The supervised learning problem is formulated as:
  - Given a training set,
  - Learn a function $h : X \to Y$ such that $h(x)$ is "good" predictor for the corresponding value of y.

For historical reasons, the function h is called a *hypothesis*.

Supervised Learning – Performance evaluation

Goal: find a hypothesis $h : X \rightarrow Y$ minimizing

$$Error(h) = E_{X,Y} L(Y, h(X))$$

Classification (Y label values, i.e. Y=yes,no) - Error: 0-1 loss:

$$L(Y, h(X)) = 1(Y! = h(X))$$

Regression Y: number, i.e. Y = stock_price - Error: square loss:

$$L(Y, h(X)) = (Y - h(X))^2$$

Generative vs. Discriminative classifiers

- Generative algorithm
  - Data generated by a distribution of feature
  - Assume x features of an animal and y the animal genre (i.e cat, dog).
  - For unknown classification data x find the class maximizing the posterior $p(y|x)$ based on Bayes rules

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

  - Therefore

$$p(y|x) = argmax_y(p(x|y)p(y))$$

Generative vs. Discriminative classifiers

- Discriminative algorithm
  - Learn directly $p(y|x)$or learn mappings directly from the space of inputs X to the labels $0, 1$
  - Do not care about how the data was generated - simply categorizes a given vector
  - Generally discriminative classifiers are more effective - better accuracy.

Some Classes of classifiers

- Class-conditional probabilistic based on $p(x \mid c_k)$
  - Naïve Bayes: simple, often effective in high dimensions
  - Parametric generative models, e.g., Gaussian (can be effective in low-dimensional problems: leads to quadratic boundaries in general)
- Regression-based $p(c_k \mid x)$ directly
  - Logistic regression: simple
  - Neural networks: non-linear extension of logistic regression

Some Classes of classifiers

- Discriminative models, focus on locating optimal decision boundaries
  - Linear discriminants: perceptron - simple, sometimes effective.
  - Support vector machines: Generalization of linear discriminants, can be quite effective, computational complexity is an issue
  - Nearest neighbor: Simple, can scale poorly in high dimensions
  - Decision trees: Often effective in high dimensions

Classification: Results & Evaluation

- Confusion matrix

Predicted class	Actual class 1	Actual class 0
1	True Positive	False positive
0	False negative	True negative

- Precision $\frac{\text{TP}}{\text{TP+FP}}$
- Recall $\frac{\text{TP}}{\text{TP+FN}}$
- Accuracy $\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$
- $F1 = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$

Classification: Results & Evaluation

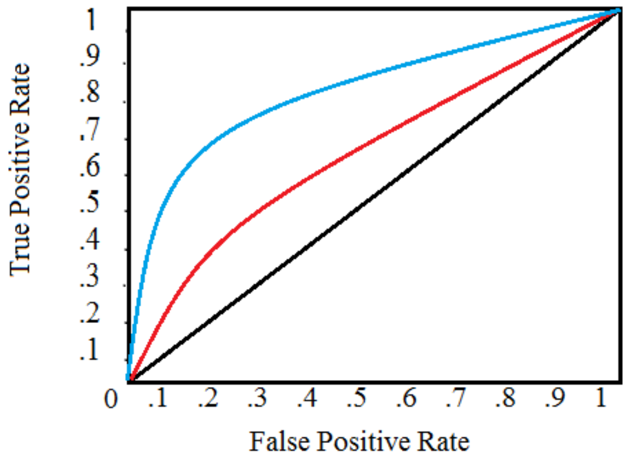
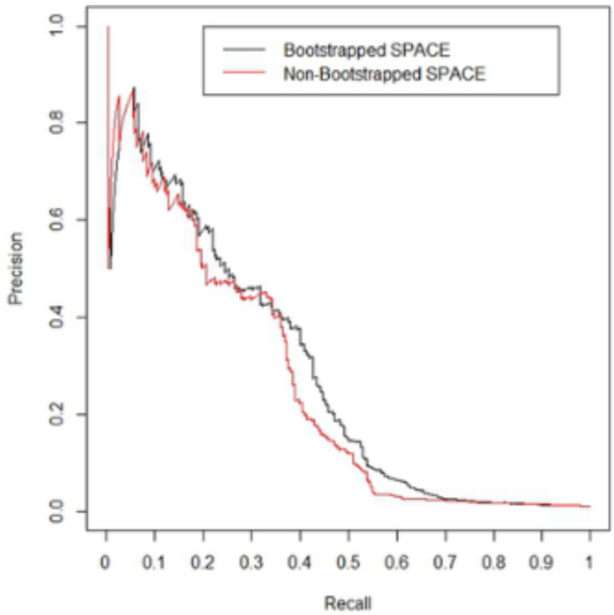


Figure: ROC curve



Precision recall curve

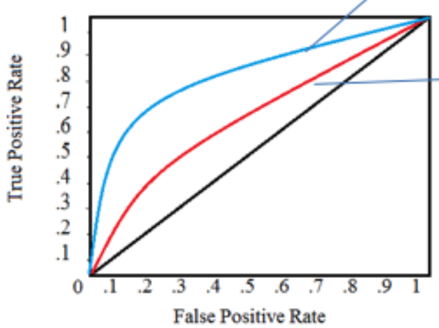
ROC/AUC curve

- True positive rate:

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

- False negative rate:

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



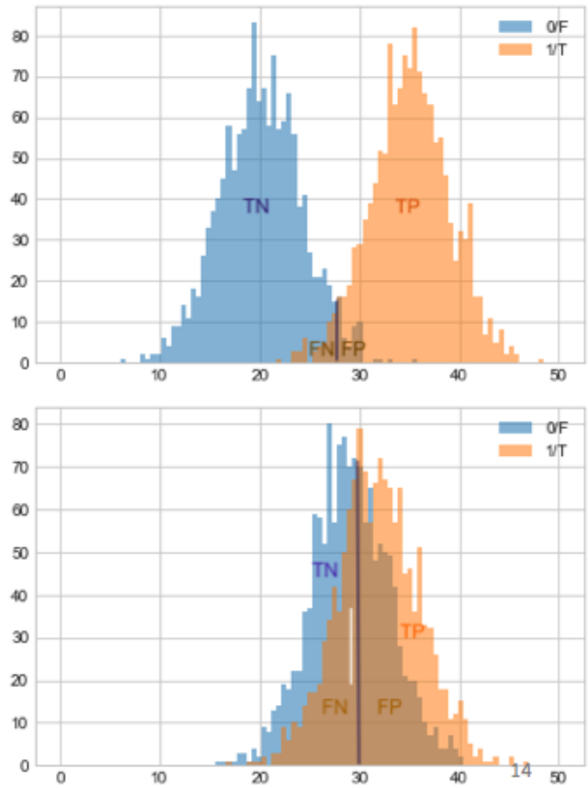


Figure: overlapping of postive / negative classes

Classification: Results & Evaluation

- Log-loss: more refined evaluation of the classification
- Capitalizing on the classifier probability

$$\log\text{-loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log p_i + (1 - y_i) \log p_i(1 - p_i)$$

- Log loss   cross entropy $H(p, q) = -\sum_x p(x) \log q(x)$ between the distribution of true labels and predictions
- Closely related to Kullback-Leibler divergence

$$KL(p|q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

- Minimizing cross entropy, maximize accuracy of the classifier.

Log Loss Analysis

Advantages of Log Loss

- Interprets Probabilities: directly evaluates how well the predicted probabilities match the true labels, making it ideal for probabilistic models.

- Penalizes Overconfidence: Models overconfident but wrong heavily penalized, encouraging models to be more conservative with their predictions.

Disadvantages of Log Loss

- Sensitive to Imbalanced Classes: for data highly imbalanced, log loss is misleading as it penalizes wrong predictions more than simply misclassifying the majority class.

- Difficult to Interpret: raw value log loss not as intuitive as accuracy or other simple metrics, though very useful for comparing models.

Outline

- Introduction to supervised learning
- **K-nn**
- Naïve Bayes
- Logistic Regression

k-NN Algorithm

- K-NN is an non parametric lazy learning algorithm.
  - No assumptions for the data (i.e. distribution, linearly separable)
  - No training lack of generalization
- K-nn classification: by majority voting
- K-nn regression: average value of k nearest neighbors

K-NN classification example

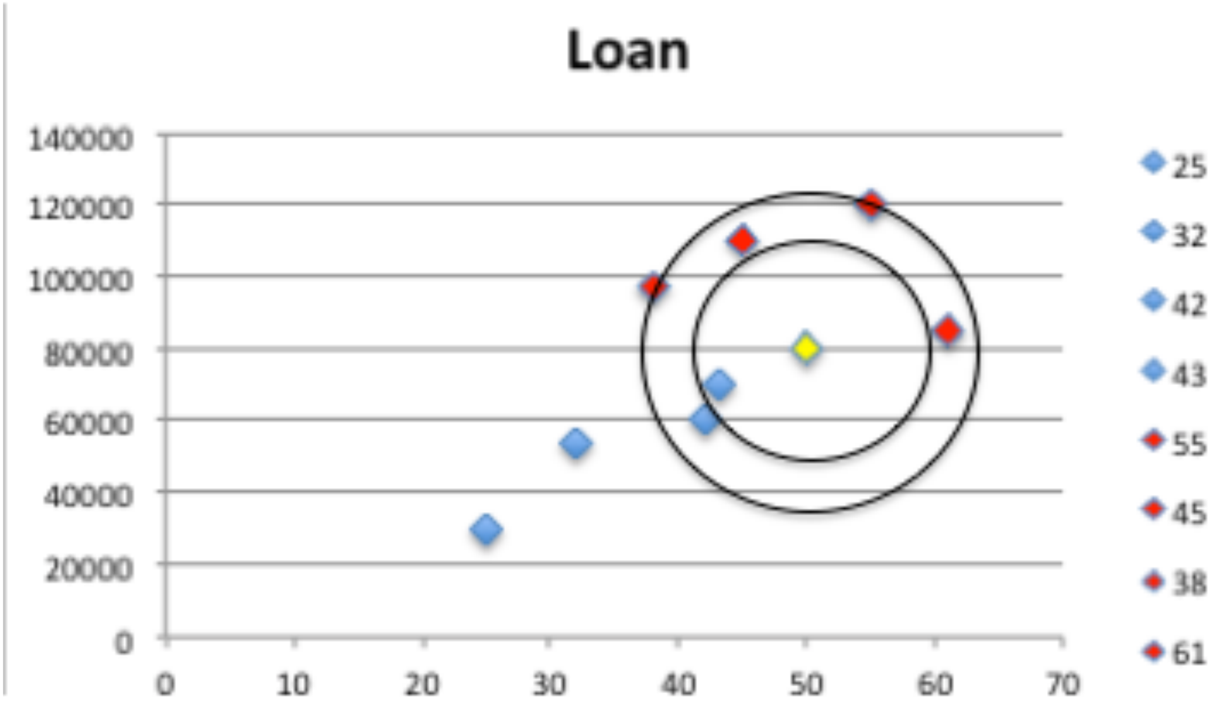


Figure: Loan amount vs. age: decide class by majority of nns

K-NN classification example

Age	Loan	Default	Distance
25	30000	N	50000,0062
32	54000	N	26000,0062
42	60000	N	20000,0016
43	70000	N	10000,0024
55	120000	Y	40000,0003
45	110000	Y	30000,0004
38	97000	Y	17000,0042
61	85000	Y	5000,0121
50	80000	?	

Figure: Loan amount vs. age: decide class by majority of nns

$$D(x,y) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

K-NN classification example

Age	Loan	Default	Distance
0	0	N	0,88932
0,1944	0,2667	N	0,57746
0,4722	0,3333	N	0,31427
0,5	0,4444	N	0,22395
0,8333	1	Y	0,46564
0,5556	0,8889	Y	0,36111
0,3611	0,7444	Y	0,38313
1	0,6111	Y	0,31056
0,6944	0,5556	?	

Figure: Loan amount vs. age: decide class by majority of nns

$$\frac{X_i - min(X)}{max(X) - min(X)}$$

K-NN regression

- k-NN algorithm to estimate continuous variables.
- weighted average of the k nearest neighbors, weighted by the inverse of their distance:
  - Compute distance of query example to the labeled examples.
  - Order labeled examples by increasing distance.
  - Find a heuristically optimal number k of nearest neighbors, based on RMSE – with cross validation.
  - Calculate inverse distance weighted average with k-nearest multivariate neighbors.

K-NN issues

- K: larger values reduce noise effect but make classes boundaries blur
- Dimensionality affects the performance
- Feature selection/scaling (mutual information)
- Binary classification: k odd number
- popular way for empirically optimal k via bootstrap method
- Assume large number of points and c classes:

$$E_{k-nn} \leq E_{Bayes}(2 - \frac{c}{c-1} E_{Bayes})$$

Outline

- Introduction to supervised learning
- K-nn
- **Naïve Bayes**
- Logistic Regression

Bayesian Classification: Why?

- Probabilistic learning
  - Calculate explicit probabilities for hypothesis,
  - practical approaches to certain types of learning problems
- Incremental
  - Each training example can incrementally increase/decrease the probability that a hypothesis is correct.
  - Prior knowledge combined with observed data.
- Probabilistic prediction
  - Predict multiple hypotheses, weighted by their probabilities

Bayesian Classification

- Problem formalized using a-posteriori probabilities:
- $p(C|X)$ = prob. vector $X =< x_1, \ldots, x_k >$ is class C.
    - e.g. $p(class = N|outlook = sunny, windy = true, \ldots)$
- Assign to sample X class label C: $p(C|X)$ is maximal
- Bayes theorem:

$$p(C|X) = \frac{p(X|C)p(C)}{p(X)}$$

- $p(X)$: prior probability of vector X
- $p(C)$ = prior probability of class C in training data
- $p(X|C)$ = probability of X given C
- $p(C|X)$ = probability of X given C

Learn probabilistic models with dependence

- Problem: computing $P(X|C)$ not unfeasible! – Why?
- $p(C)$: assume p classes
- $p(X =< x_1, \dots, x_k >)$: k binary features
- $p^{(2k-1)}$ parameters
- Likelihood: $p(X|C)$
    - Need a value for each possible $p(X =< x_1, \dots, x_k > |C))$

Naive Bayesian Classification

- Naïve assumption: attribute independence

$$p(X|C) = \prod_i p(x_i|C)$$

- $p(x_i|C)$: relative frequency of x_i as i-th attribute in class C
- Computationally feasible
- Generative probabilistic model with conditional independence assumption
- Prediction:

$$p(C|X) = argmax_C(p(X|C)p(C)) = argmax_C(\prod_i^k p(x_i|C)p(C))$$

Naive Bayesian Classification

- Simple to train
- estimate conditional probabilities for each feature-class pair
- Often very good baseline
- Feature selection can be helpful, e.g., information gain
- However. . . . on most problems can usually be outperformed by a more complex model

Playtennis example: estimating $P(xi|C)$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

p(P) = 9/14
p(N) = 5/14

outlook	
p(sunny|P) = 2/9	p(sunny|N) = 3/5
p(overcast|P) = 4/9	p(overcast|N) = 0
p(rain|P) = 3/9	p(rain|N) = 2/5
temperature	
p(hot|P) = 2/9	p(hot|N) = 2/5
p(mild|P) = 4/9	p(mild|N) = 2/5
p(cool|P) = 3/9	p(cool|N) = 1/5
humidity	
p(high|P) = 3/9	p(high|N) = 4/5
p(normal|P) = 6/9	p(normal|N) = 2/5
windy	
p(true|P) = 3/9	p(true|N) = 3/5
p(false|P) = 6/9	p(false|N) = 2/5

Naive Bayesian Classification

- unseen sample $X =< rain, hot, high, false >$
- $p(X|P)p(P) = p(rain|P)p(hot|P)p(high|P)p(false|P)p(P) = 3/9 * 2/9 * 3/9 * 6/9 * 9/14 = 0.010582$
- $p(X|N)p(N) = P(rain|N)P(hot|N)P(high|N)P(false|N)P(N) = 2/5 * 2/5 * 4/5 * 2/5 * 5/14 =$ **0.018286**
- Sample X is classified in class N (don't play)

Outline

- Introduction to supervised learning
- K-nn
- Naïve Bayes
- **Logistic Regression**

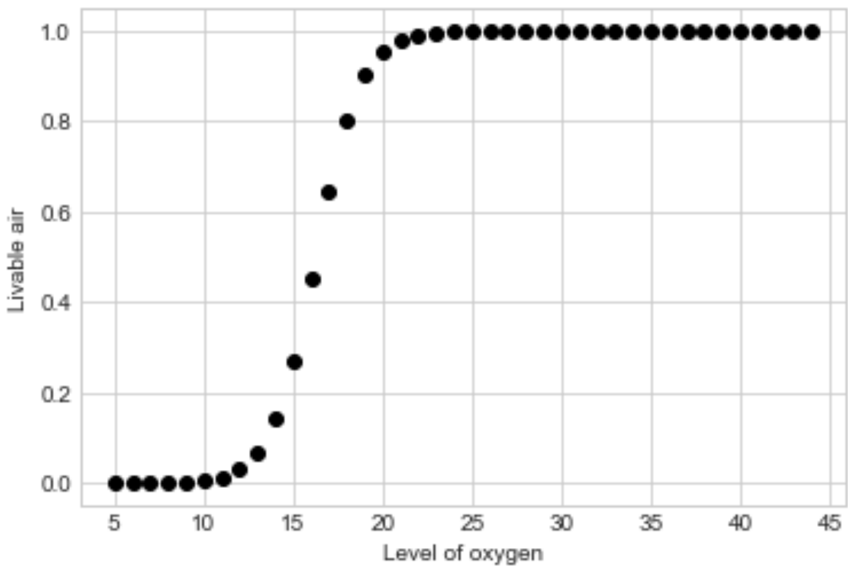
Solving a supervised learning problem

Given data, a real-world classification problem, and constraints, you need to determine:

- classifier to use
- optimization method to employ
- loss function to minimize
- features to consider from the data
- evaluation metric to use

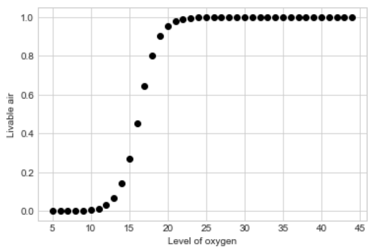
Logistic Regression

- Fits binary classification problems
- Output in [0,1]
- Need a function that takes data vector x and produces as output p(x) in [0,1]



Why linear regression does not do..

- Linear regression: output y is continuous – need binary classification
- Need a $p(X)$ in [0,1] – linear regression does not guarantee it.
- homoscedasticity assumption: variance of Y constant across values of X.
- significance testing assumes prediction errors (y-f(x)) are normally distributed.
- But y takes values 0 and 1,



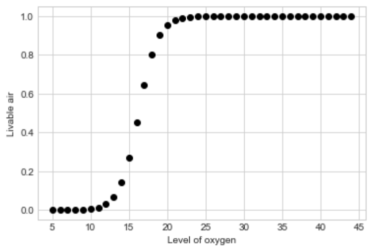
Logistic function – design

- Classification: $\mathbf{x} \in R^p, y \in \{0, 1\}$
- Assumption (odds p(y=1)):

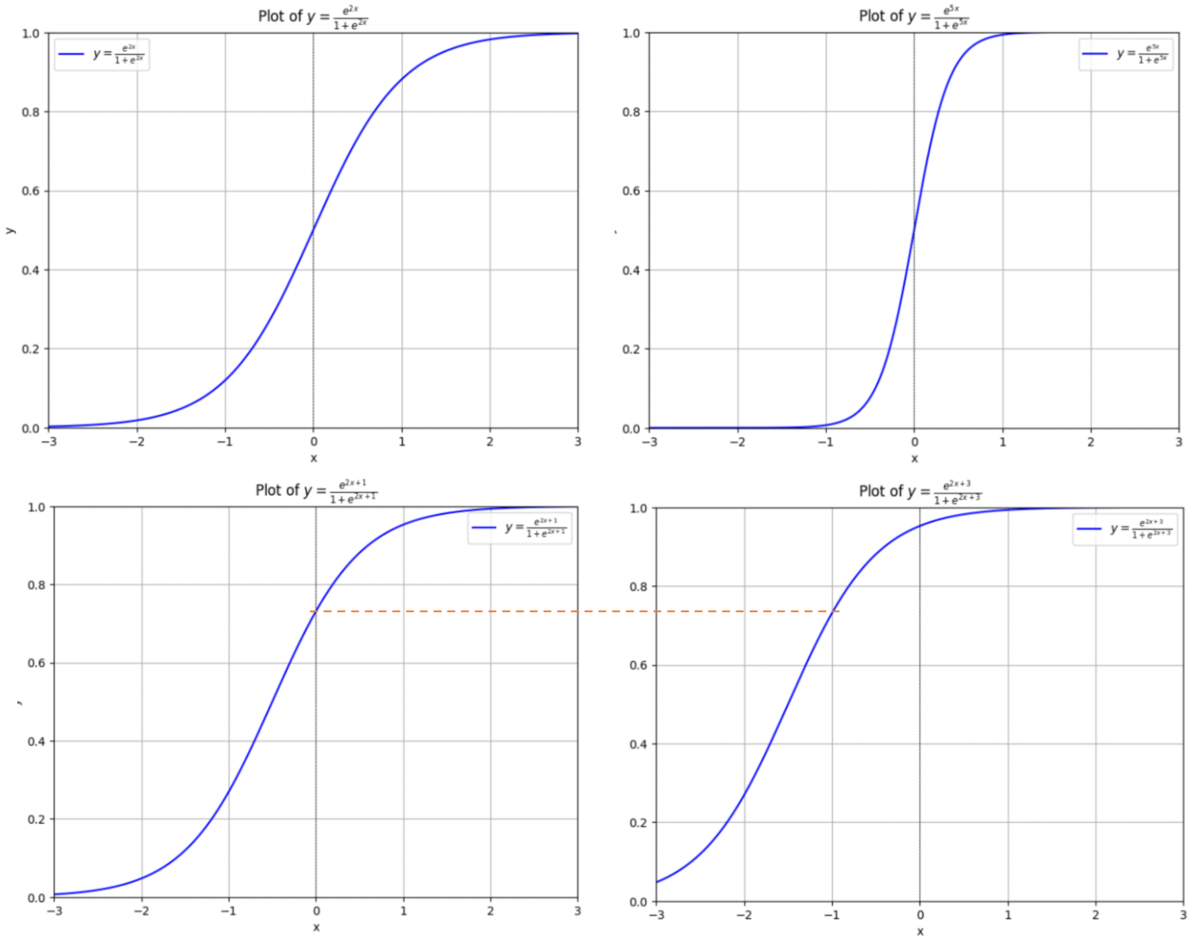
$$log(\frac{p(y=1|\mathrm{x})}{1-p(y=1|\mathrm{x})}) = \alpha\mathrm{x} + \beta$$

- therefore

$$p(y=1|\mathrm{x}) = \frac{e^{\alpha\mathrm{x}+\beta}}{1+e^{\alpha\mathrm{x}+\beta}} = \frac{1}{1+e^{-(\alpha\mathrm{x}+\beta)}}$$



Examples of logistic function - parameters



Click prediction example

- Web site publishing ads
- Predict user i click an ad.
- Assume class c_i (1: clicked, 0: otherwise)
- x_i data for user i (history of URLs visited)
- logit(P(¡user i clicks on the ad¿) = linear function of the features x_i

Click Prediction: Logistic regression model

- linear model for c_i,
- take the log of the odds ratio:

$$log(p(c_i = 1|x_i)/(1 - p(c_i = 1|x_i)) = \alpha + \beta^T x_i$$

- Or

$$p(c_i = 1|x_i) = \frac{e^{\alpha+\beta^T x_i}}{1 + e^{\alpha+\beta^T x_i}}$$

- α: base rate, unconditional probability of $c_i = 1$ (click)
- β: dependence on the user data
  - slope of the logit function
  - Determines the relevance of features (i.e. pages visited) to the likelihood of the ad being clicked

Click Prediction: α,β parameter estimation

- Parameters: $\theta = \{\alpha, \beta\}$,
- Likelihood: $L(\theta|X_1, ..., X_n) = p(X|\theta) = p(X_1|\theta)...p(X_n|\theta)$,
- X_i users are independent:

$$\theta_{MLE} = argmax_{\theta} \prod_{k=1}^{N} p(X_i|\theta)$$

- Setting:

$$p_i = \frac{1}{1 + e^{-(\alpha + \beta^T x_i)}}$$

- Then:

$$\theta_{MLE} = argmax_{\theta} \prod_{k=1}^{N} p_i^c (1 - p_i)^c$$

  c: number of clicks

- MLE with either i. Newton-Raphson method or ii. Stochastic gradient descent