

Classical Data Analysis

Master in Big Data Solutions 2020-2021



Victor Pajuelo Madrigal

victor.pajuelo@bts.tech

Today's Objective

- Review concepts from previous sessions
- Logistic regression
 - Logistic regression theory
 - Logistic regression with Python

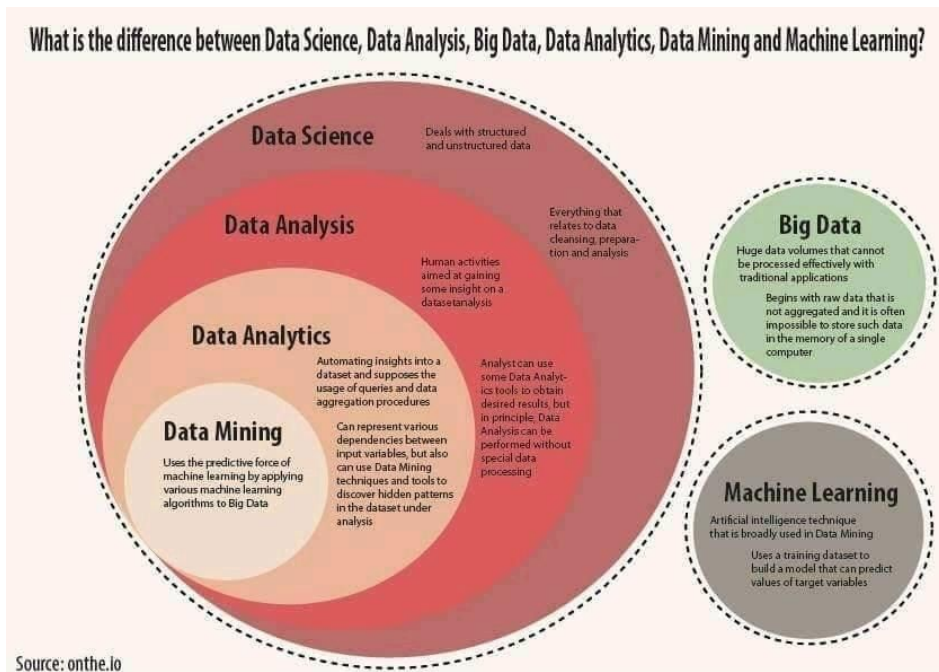
Contents

- Introduction to data analysis
- Linear Regression
- **Logistic Regression**
- Regression analysis (Polynomial, Ridge, Lasso, etc.)
- Neural Networks (MLPs in depth)
- Support Vector Machines (SVM)
- Decision Trees
- Ensemble Methods (Random Forests, Bagging, etc.)
- Other Classifier (KNN, Naïve Bayes)
- K-Means
- PCA
- Hierarchical Clustering

Previous Session

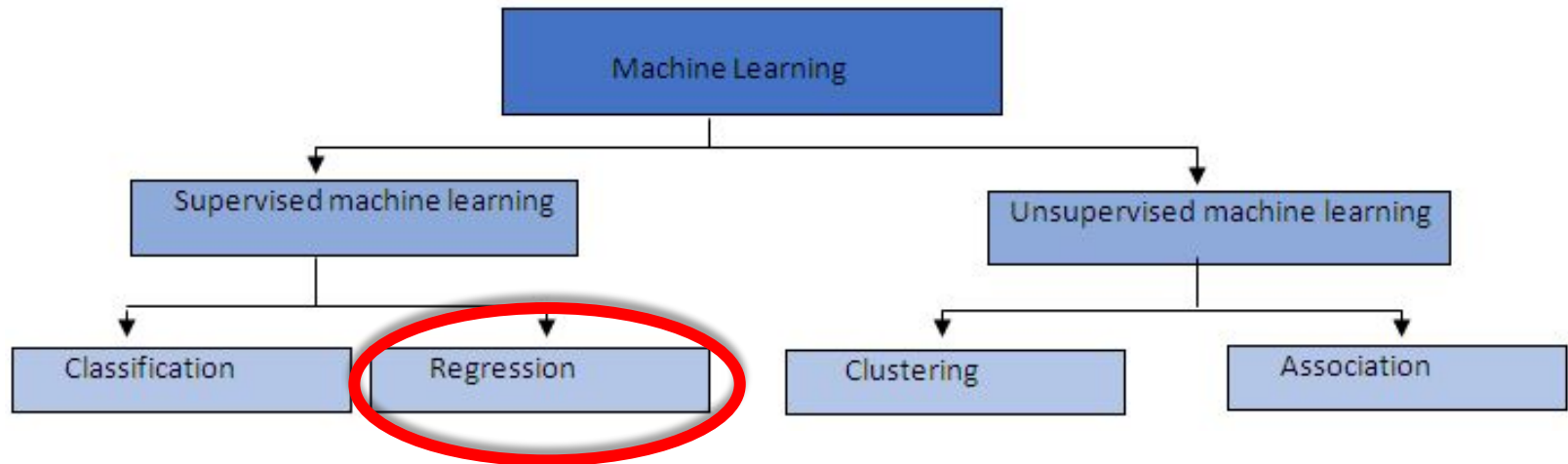
What is Data Analysis

"Extracting, cleaning, transforming, modeling and visualization of data with an intention to uncover meaningful and useful information that can help in deriving conclusion and take decisions."



Data Analysis is characterized by a wide use of **Data Mining algorithms**

Supervised learning: Regression



Supervised learning: Regression

- **Regression:** Predict continuous valued output
- **Examples:**
 - Predict stock market index based on other indicator
 - Predict the total amount of sales of a company based on the total budget spent for advertising
 - Predict the price of a house based based on its characteristic
 - Other examples?

Supervised learning: Linear Regression

Given a dataset

x_1	x_2	y
Living area (feet ²)	#bedrooms	Price (1000\$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
\vdots	\vdots	\vdots

- x_1 and x_2 are the explanatory variables (aka independent variables). They can be either discrete or continuous.
- y is the target (aka dependent variable). It must be continuous.

- Linear Regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So it finds out a linear relationship between x (input) and y (output).
- The goal is to find the **function that best fits the data** minimizing the error.
- The error in a regression task is the difference between the prediction of the regression model $h(X)$ and the actual target value y . It can be expressed as:

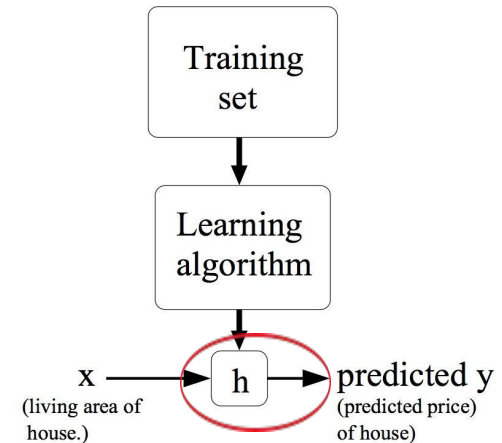
Absolute error: $\sum_i |y_i - h_\theta(x_i)|$

Squared error: $\sum_i (y_i - h_\theta(x_i))^2$

Supervised learning: Linear Regression

Training Set

Size in feet ² (x)	Price (\$) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...



We assume a linear model $h_{\theta}(x) = \theta_0 + \theta_1 x$

Given some estimates of the **coefficients** θ_0 and θ_1 we predict future observations using:

$$h_{\theta}(x) = \hat{y} = \hat{\theta}_0 + \hat{\theta}_1 x$$

we want to come up with values for the **parameters** θ_0 and θ_1 so that $h_{\theta}(x)$ (the prediction) is close to y (the actual value) for our training set (X, Y) .

Supervised learning: Linear Regression

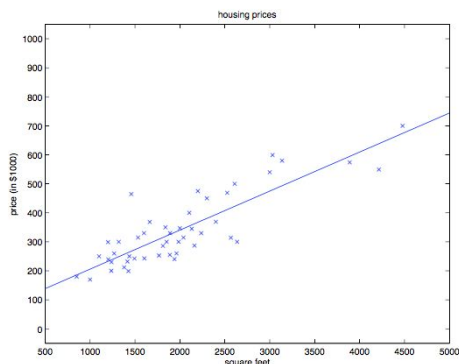
Univariate VS Multivariate Linear regression

Univariate (Simple LR)

Size in feet ² (x)	Price (\$) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

Hypothesis:

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

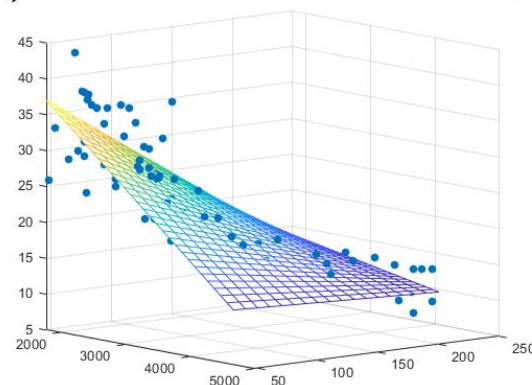


Multivariate

Living area (feet ²)	#bedrooms	Price (1000\$)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

Hypothesis:

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$



Supervised learning: Linear Regression

Univariate (simple) Linear regression

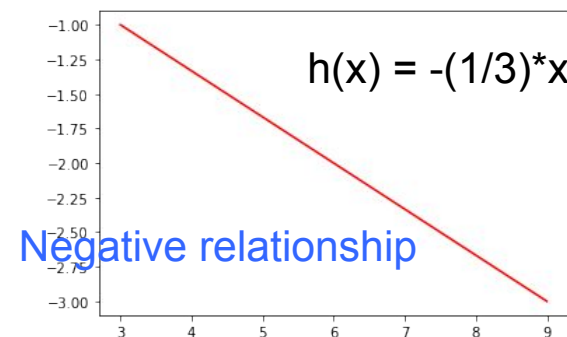
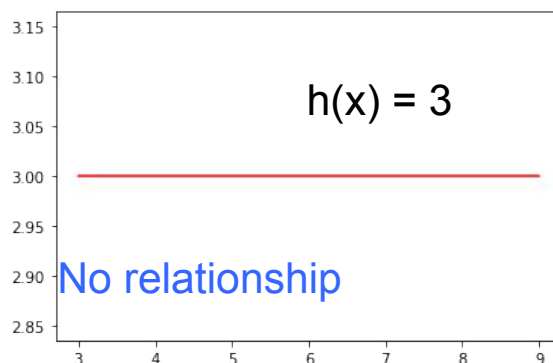
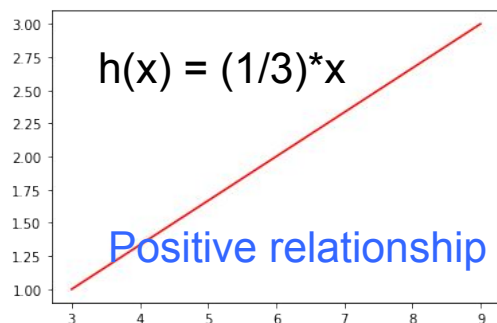
Hypothesis:

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

θ_0 : is the intercept of the line. It is the expected value of y when $x=0$

θ_1 : is the slope of the line. A value very close to 0 indicates little to no relationship; large positive or negative values indicate large positive or negative relationships, respectively.

Examples

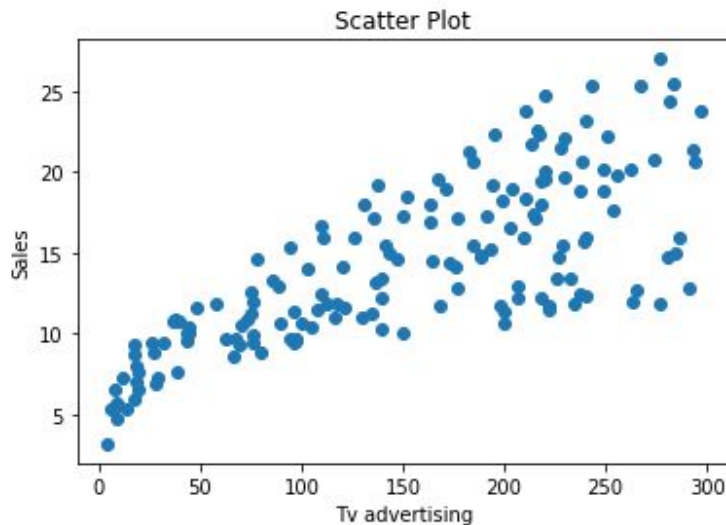


Supervised learning: Linear Regression

Assumptions Of Linear Regression Algorithm

Hypothesis:

1. Linear Relationship between the features (x) and target (y)



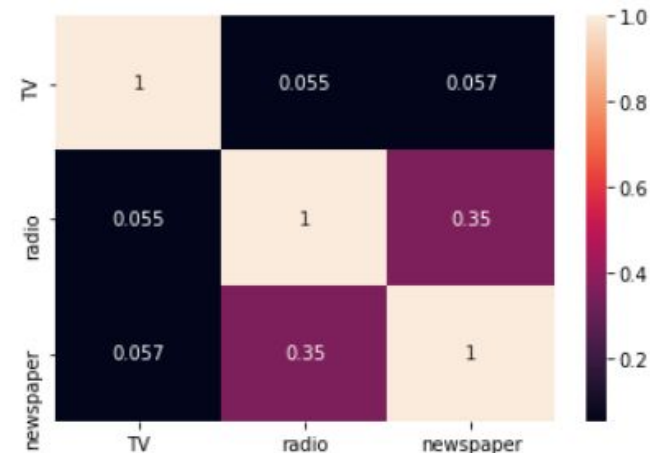
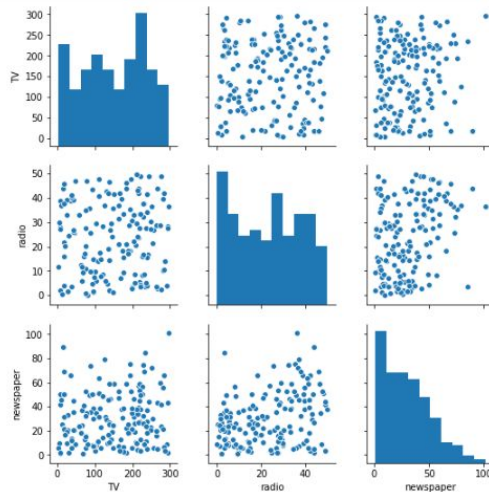
It can be validated by plotting a scatter plot between the features and the target.

Supervised learning: Linear Regression

Assumptions Of Linear Regression Algorithm

Hypothesis:

2. Little or no Multicollinearity between the features, i.e., very high inter-correlations or inter-associations among the independent variables



Pair plots and heatmaps(correlation matrix) can be used for identifying highly correlated features.

Supervised learning: Linear Regression

Evaluation Metrics

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2}$$

RMSE (Root Mean Square Error) – Particularly used because

$$\text{MAE} = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j|$$

MAE (Mean Absolute Error) - is a linear score

$$\hat{R}^2 = 1 - \frac{\sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{\sum_{i=1}^n (Y_i - \bar{Y})^2} = 1 - \frac{\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{\frac{1}{n} \sum_{i=1}^n (Y_i - \bar{Y})^2}$$

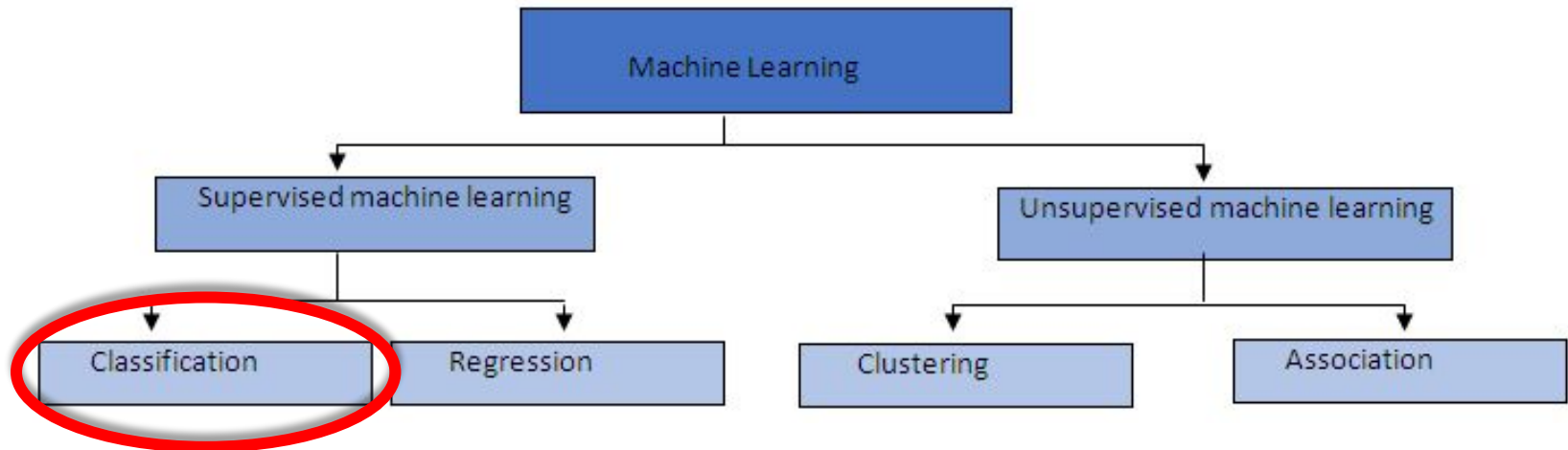
R Squared - The maximum is 1 but minimum can be negative infinity (even if it is unlikely scenario, usually the minimum is 0)

$$R_{adj}^2 = 1 - \left[\frac{(1 - R^2)(n - 1)}{n - k - 1} \right]$$

Adjusted R Squared

Logistic Regression

Supervised learning: Classification



The Logistic Regression algorithm solve **classification** problems. Don't get confused by the name "Regression"!

Logistic Regression

- The main goal of logistic regression is to perform binary classification
- Examples of applications:
 - In medical research, in the field of **predictive food microbiology**, to describe bacterial growth/no growth interface, in the 0–1 format.
 - Predict the **risk of developing a given disease** based on observed characteristics of the patient.
 - In credit risk modeling in identifying **potential loan defaulters**.

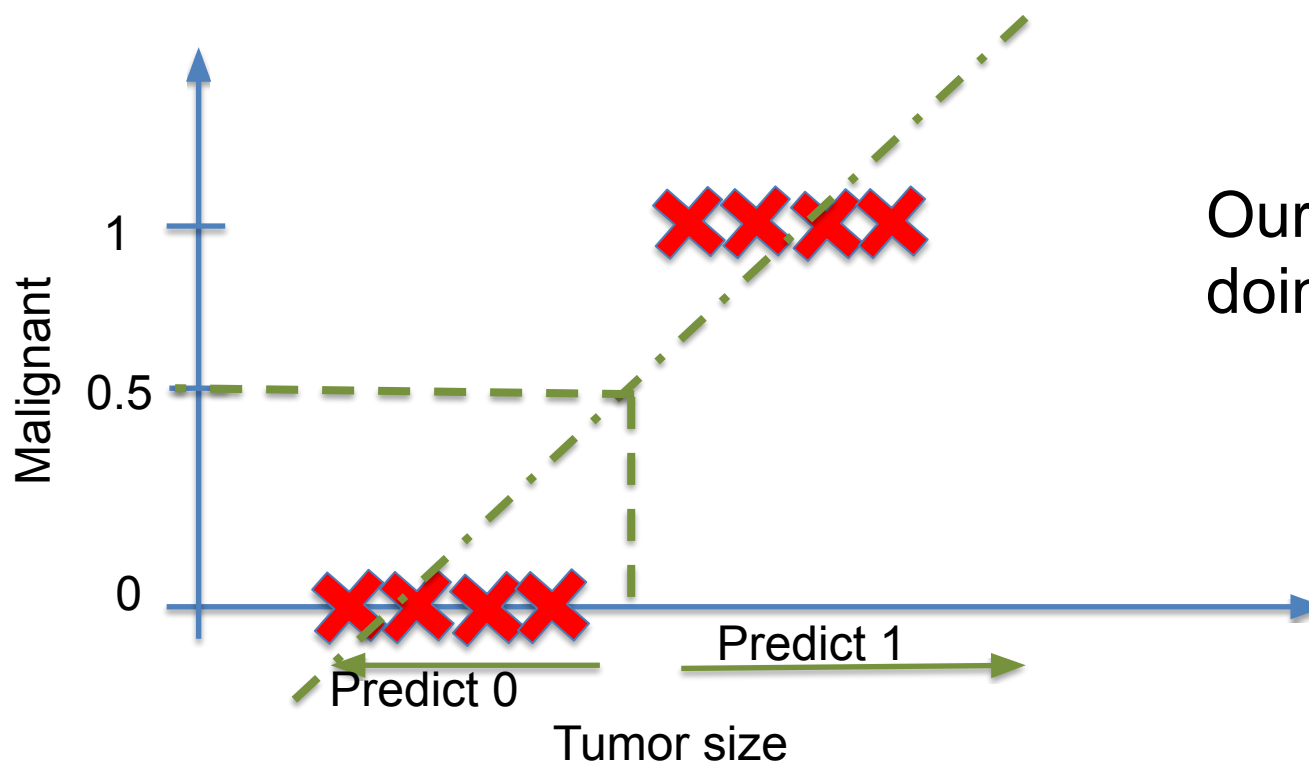
All the above problems are **binary** classification problems.

Possible results:

- $y = 1$
 - $y = 0$
- Feature Engineering plays an important role in regards to the performance of Logistic

Logistic Regression

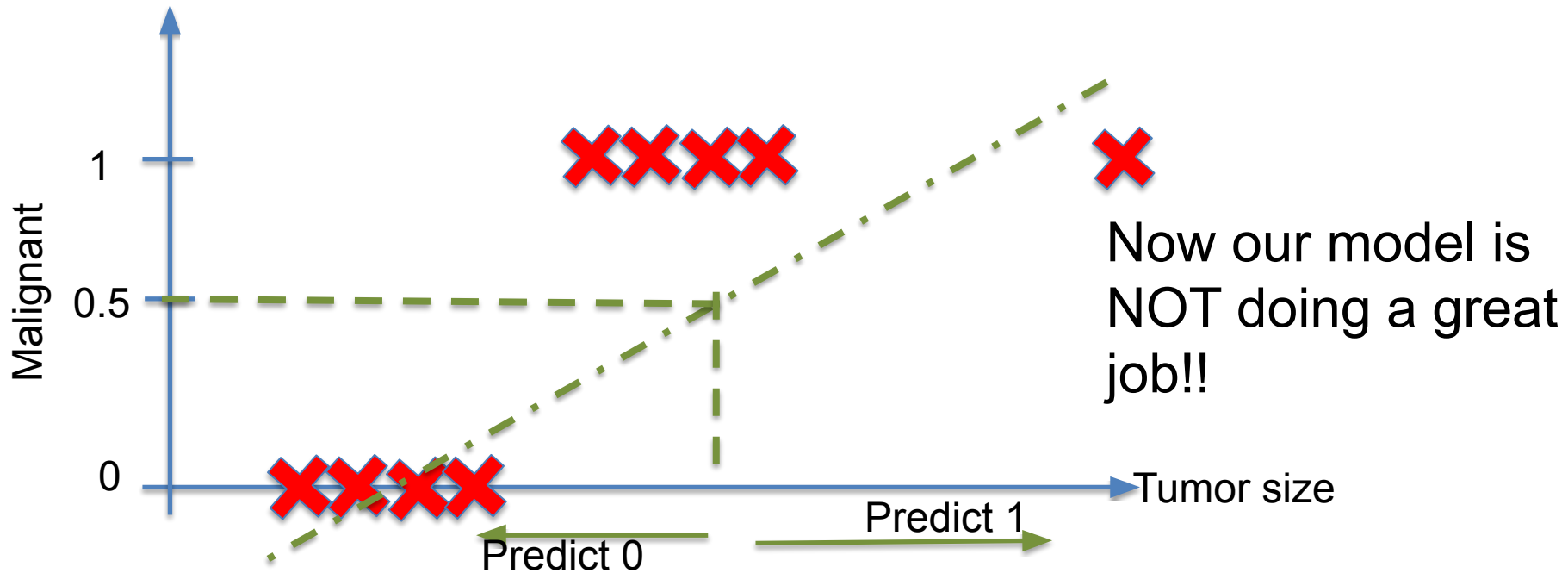
- How to develop a classification algorithm?
 - Let's try applying a linear regression model to classify a tumor as malignant or not!



Our model is
doing a great job!!

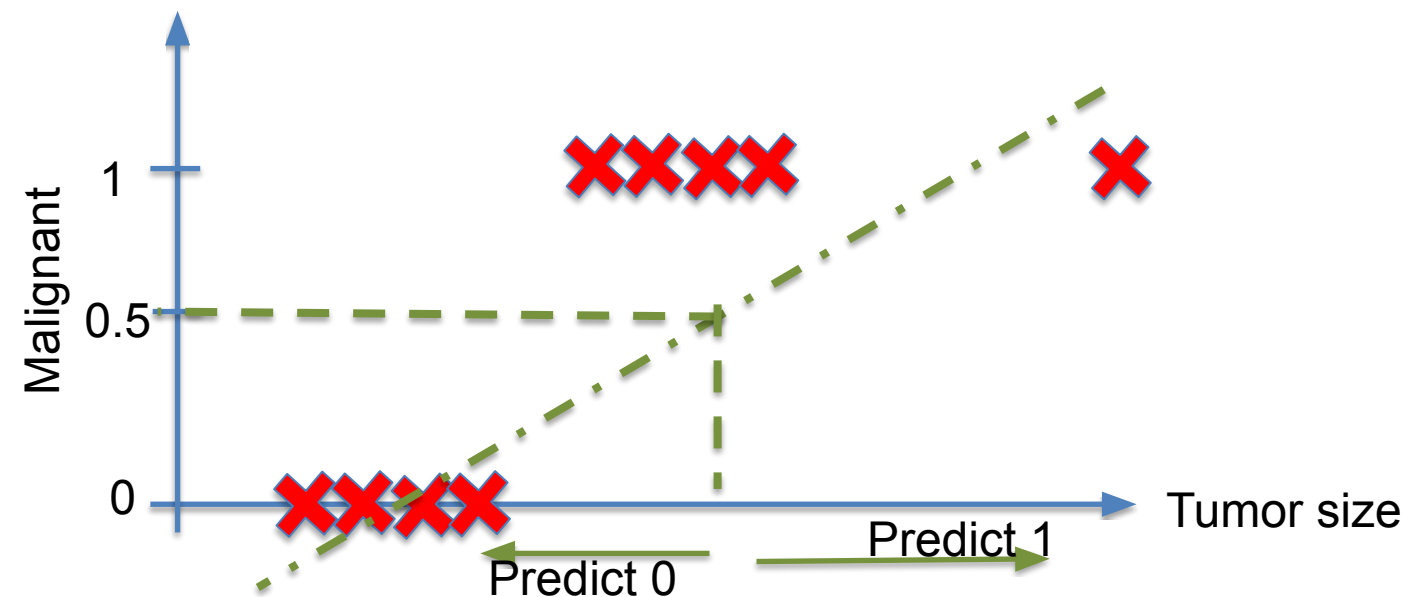
Logistic Regression

- How to develop a classification algorithm?
 - Let's try applying a linear regression model to classify a tumor as malignant or not!
 - Let's add an additional example



Many positive examples are now classified as negative.

Logistic Regression



- Many positive examples are now classified as negative.
- It also doesn't make sense for $h_{\theta}(x)$ to take values larger than 1 or smaller than 0 when we know that $y \in \{0, 1\}$.

Logistic Regression

Linear regression hypothesis function:

$$h_{\theta}(x) = \theta^T x = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

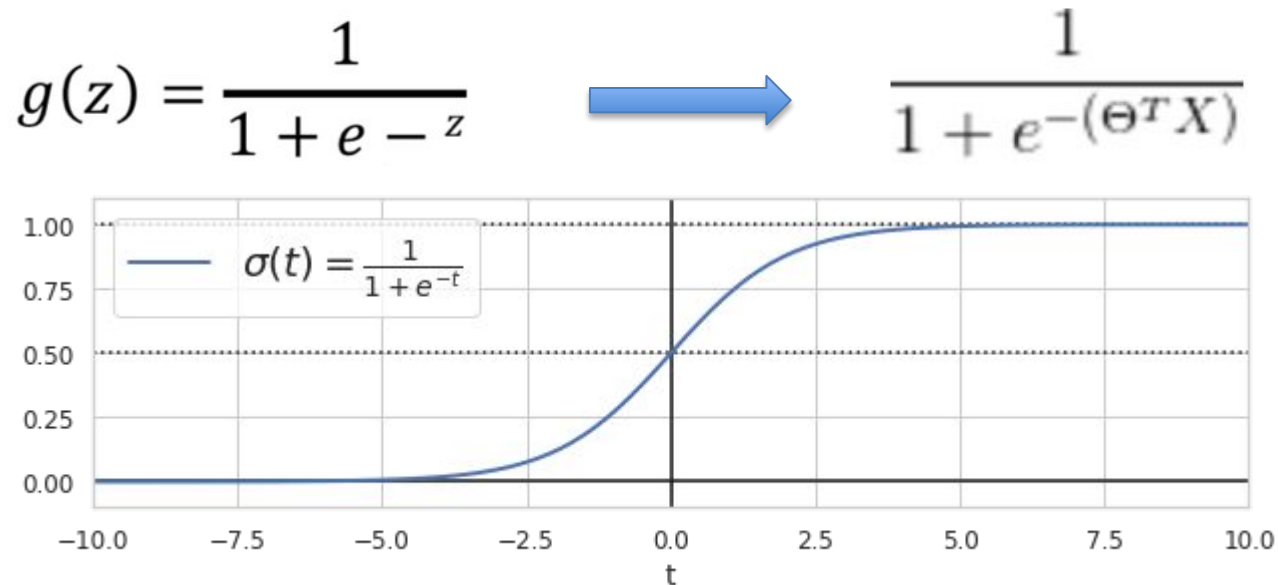
In Logistic regression we want the output of the hypothesis function to be between 0 and 1. We apply the **sigmoid function (aka logistic function)** to the output of the linear regression, obtaining the hypothesis function for the logistic regression:

$$h_{\theta}(x) = g(\theta^T x) = g(\theta_0 + \theta_1 x_1 + \dots + \theta_n x_n)$$

Logistic Regression is **just like a linear regression model**, it computes a weighted sum of the input features (plus a bias), but instead of outputting the linear result directly, it outputs the logistic of that result

Logistic Regression

- Sigmoid function:

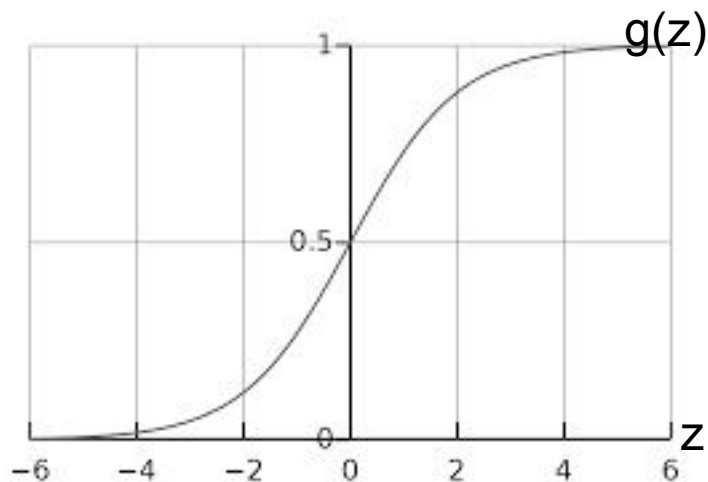


The logistic is a sigmoid function that outputs a number between 0 and 1

Logistic Regression

Logistic Regression hypothesis function:

$$g(z) = \frac{1}{1 + e^{-z}}$$



When our hypothesis $h_{\theta}(\mathbf{x})$ outputs a number, we treat that value as the estimated probability that $y=1$ on input \mathbf{x}

Notice that $g(z)$ tends towards 1 as $z \rightarrow \infty$, and $g(z)$ tends towards 0 as $z \rightarrow -\infty$. Moreover, $g(z)$, and hence also $h(\mathbf{x})$, is always bounded between 0 and 1.

Logistic Regression aka *Logit Regression*

- It is commonly used to estimate the **probability that an instance belongs to a particular class**
 - What is the probability that an email is spam?
 - What is the probability that I will pass this class?
- If the estimated probability is above 50%, the model is predicting that the instance belongs to that class (positive class, labelled 1)
- If the estimated probability is below 50%, the model predicts that the instance belongs to the other class (negative class, labelled 0)

Making predictions is easy...

$$\hat{y} = \begin{cases} 0 \rightarrow \hat{p} < 0.5 \\ 1 \rightarrow \hat{p} \geq 0.5 \end{cases}$$

Logistic Regression

Cost function

- Alright, we know now that Logistic Regression estimates probabilities and makes predictions out of it but...
 - How is it trained?
- The **objective** of the training is to set the parameter vector $h_{\theta}()$ so that...
 - Estimates high probabilities for positive instances ($y=1$)
 - Estimates low probabilities for negative instances ($y=0$)

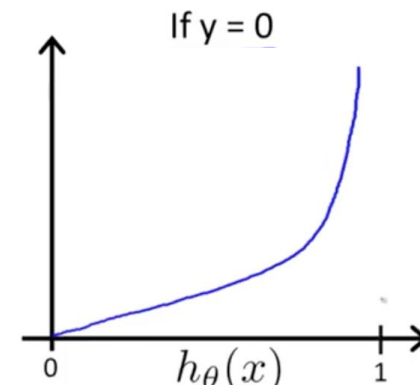
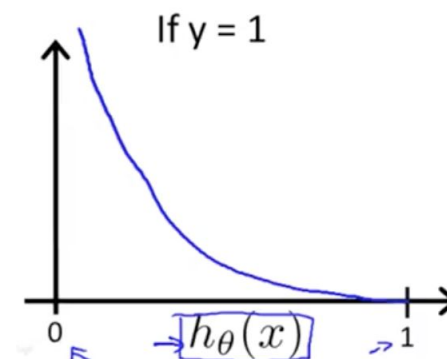
$$\text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$

Logistic Regression

Cost function

$$\text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$

- In this example for a single instance (x) , $-\log(h_{\theta}(x))$, grows very large when $h_{\theta}(x)$ approaches 0
 - So the cost will be very large if the model predicts a probability close to 0 for a positive instance
- $-\log(1 - h_{\theta}(x))$, grows very large when $h_{\theta}(x)$ approaches 1
 - So the cost will be very large if the model predicts a probability close to 1 for a negative instance

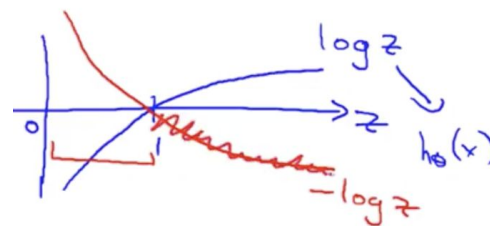
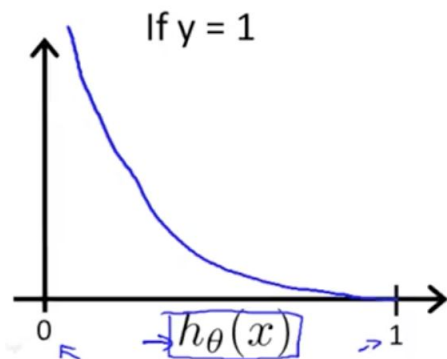


Logistic Regression

Cost function

- Given the logistic regression model, how do we fit the θ parameters?

$$\text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$



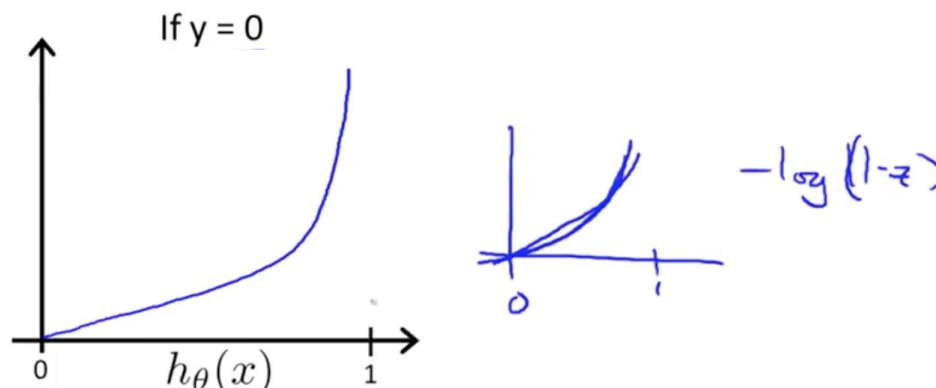
- if the hypothesis predicts that $y = 1$ and $y = 1$ cost = 0.
- if the hypothesis predicts that $y = 0$ and $y = 1$ cost $\rightarrow \text{inf.}$

Logistic Regression

Cost function

- Given the logistic regression model, how do we fit the θ parameters?

$$\text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$



- if the hypothesis predicts that $y = 0$ and $y = 0$ cost = 0.
- if the hypothesis predicts that $y = 1$ and $y = 0$ cost $\rightarrow \text{inf.}$

Logistic Regression

Cost function

- Given the logistic regression model, how do we fit the θ parameters?

$$\text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$

- This cost function can be written as follow

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

- This cost function can be derived from statistics using the principle of maximum likelihood estimation
- We can apply the **gradient descent** algorithm to find the parameter that minimize the cost function.

Logistic Regression

Guaranteed success! (almost)

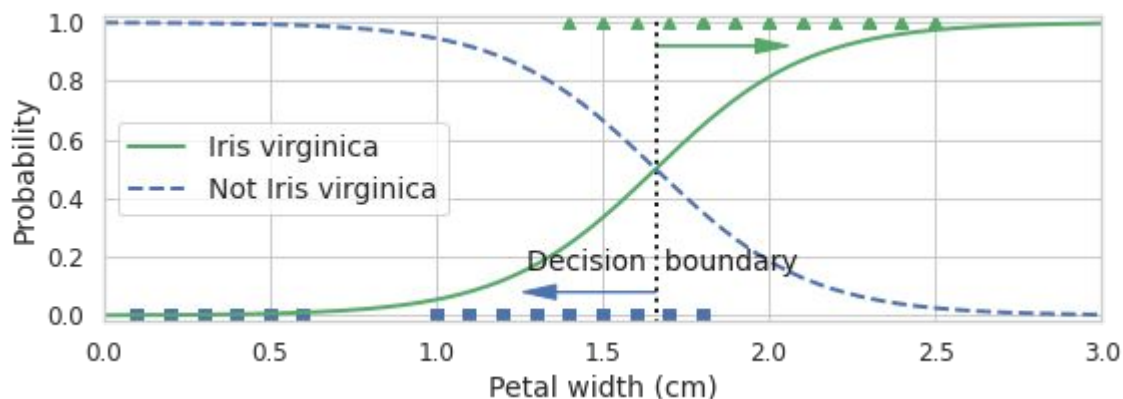
$$-\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\theta}(x^{(i)})) \right]$$

- The cost function for the logistic regression is **convex**, so that *Gradient Descent* or any other optimization algorithm, will guarantee to **find the global minimum**
 - That is... if the learning rate is not too large and one has enough time to wait
 - This can be controlled with the **tolerance criteria** and the **number of iterations** defined in Sklearn

Logistic Regression

A note on Decision Boundaries

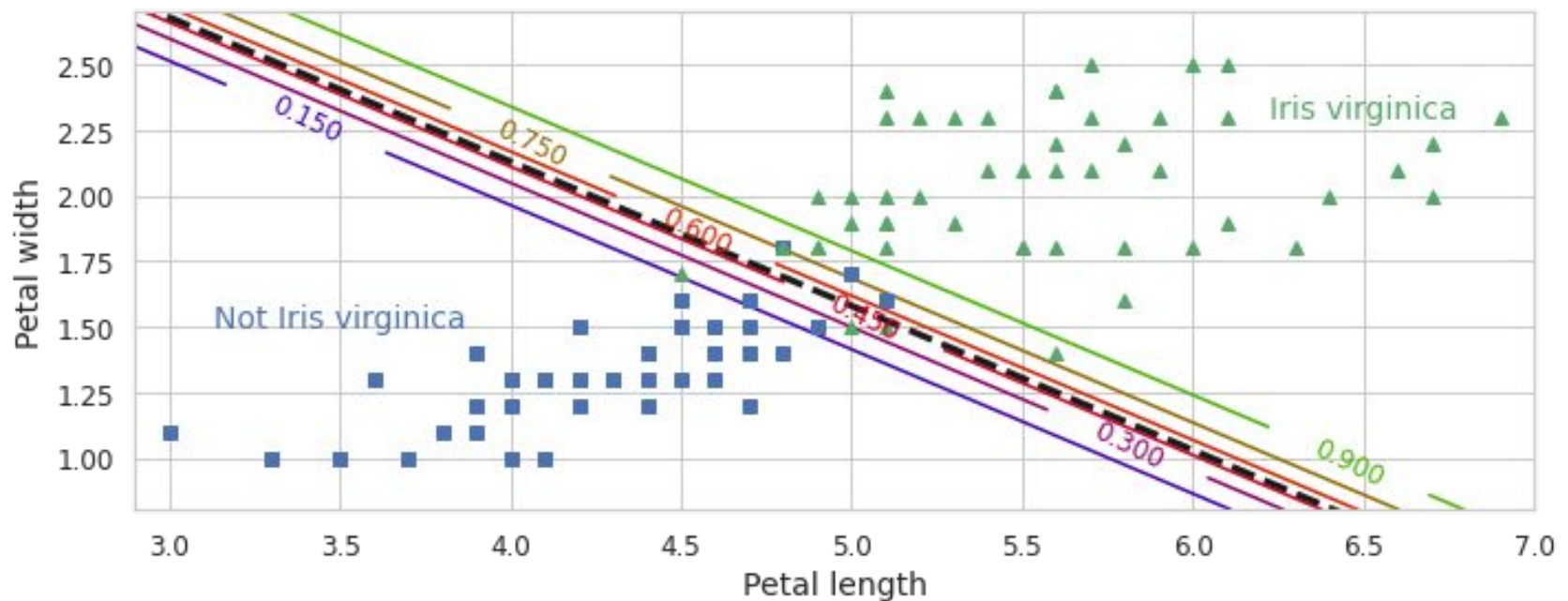
- Decision boundaries in LogReg are usually set when negative and positive probabilities cross at $y=0.5$
- If you want to get probabilistic results rather than classes you can use `sklearn's predict_proba()` instead of `predict()`
- That can be used for further refinement of the decision boundary (in case that one wants to classify something as positive if it goes below 0.5)



Logistic Regression

A note on Decision Boundaries

- `predict_proba()` returns the probabilities of the class to be the class that represents in its vector space



Logistic Regression

A note on Decision Boundaries

```
>> np.set_printoptions(suppress=True,  
formatter={'float_kind': '{:0.9f}'.format})  
>> y_proba = log_reg.predict_proba(X_new)
```

```
[0.999999998, 0.000000002]
```

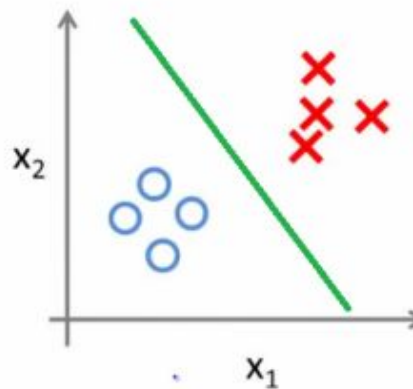
- Probabilities in results always sum up to 1

Logistic Regression

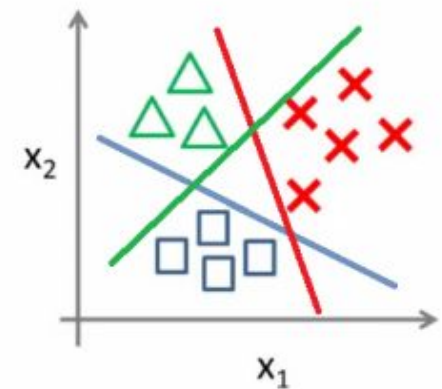
Multi-class Classification

- **One vs. all classification**
 - classification problem with N distinct classes
 - Train N distinct binary classifiers, each designed for recognizing a particular class

Binary classification:



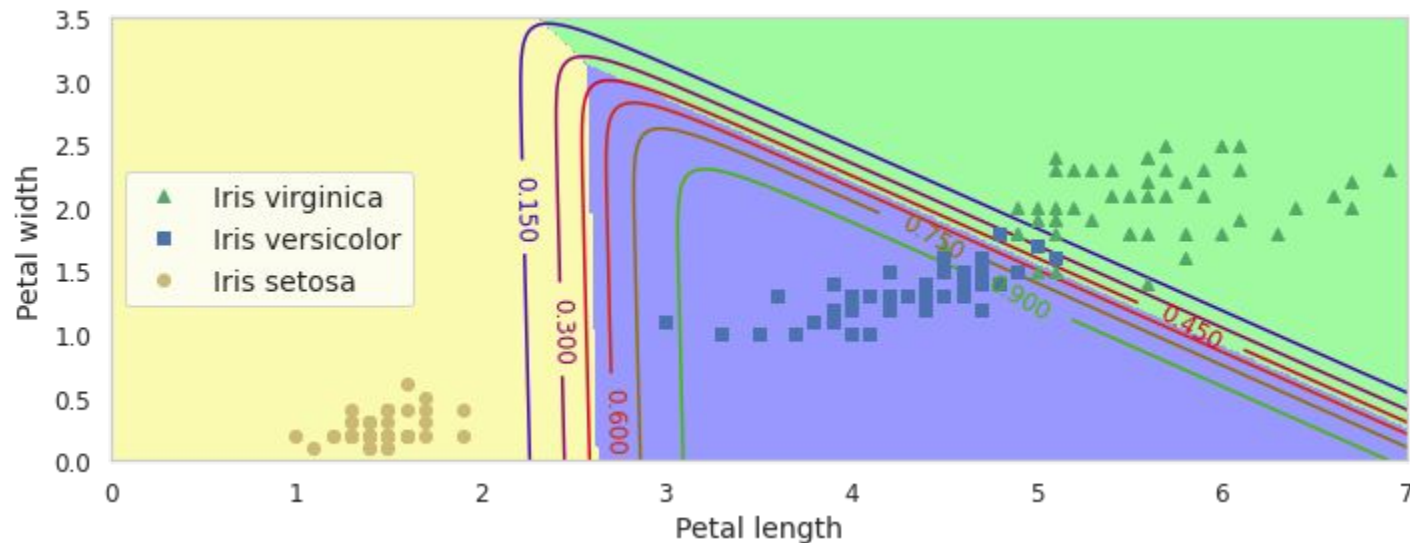
Multi-class classification:



Logistic Regression

Multi-class Classification

We can use multi class classification in Sklearn LogReg by passing the `multiclass` parameter



Logistic Regression

Dealing with overfit and underfit

A model generalization error can be expressed as:

- **BIAS:** this part of the generalization error is due to wrong assumptions, such as assuming that the data is linear when it is actually quadratic. High-bias model is highly likely to underfit the training data
- **VARIANCE:** this error is due to having a model that is too sensitive to small variations in training data. A model with a lot of degrees of freedom, will have high variance and it will overfit our training data

Logistic Regression

How do we regularize those errors?

- Lasso, L1 penalty
 - *L1 penalizes sum of absolute value of weights.*
- Ridge, L2 penalty (added by default by Sklearn)
 - *L2 regularization penalizes sum of square weights.*

L1 Regularization

$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M |W_j|$$

L2 Regularization

$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M W_j^2$$

Loss function

Regularization
Term

In Sklearn, this value is represented by C, and it is the inverse of lambda. So, the higher the value of C, the less the model will be regularized

Huge part of next session will be spend on understanding this fully

Parameter recap

Logistic Regression

penalty	Regularization penalty, default is L2
dual	Used in rare cases, default is False. Has to do with the Lagrangian dual problem of convex optimization. We will see it in NNs
tol	stopping criteria, good to control when to stop training
C	the inverse of lambda, smaller means more regularization (more sticks in the wheel, less overfit)
class_weight	pass weight to classes, useful when too lazy to oversample or undersample
solver	the solving optimizer, usually leave default. If you have an immense dataset use <i>sag</i> or <i>saga</i>
max_iter	for how many iterations to go on training
multi_class	select “multinomial” for multi classification

Check literature and more parameters in official docs:





https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Recap on metrics for classification

Logistic Regression

Evaluation Metrics

- Confusion Matrix

	$\hat{Y} = 0$ NEGATIVE	$\hat{Y} = 1$ POSITIVE
$Y = 0$ NOT PREGNANT	<p>TRUE NEGATIVE</p>  <p>You're not pregnant</p>	<p>FALSE POSITIVE</p>  <p>You're pregnant</p> <p>TYPE 1 ERROR</p>
$Y = 1$ PREGNANT	<p>FALSE NEGATIVE</p>  <p>You're not pregnant</p> <p>TYPE 2 ERROR</p>	<p>TRUE POSITIVE</p>  <p>You're pregnant</p>

Logistic Regression

Evaluation Metrics

n=165		Predicted: NO	Predicted: YES	
Actual: NO		TN = 50	FP = 10	60
Actual: YES		FN = 5	TP = 100	105
		55	110	

Accuracy:

- Overall, how often is it **correct**?
- $(TP + TN) / \text{total} = 150/165 = 0.91$

- It may not be reliable, especially in case of imbalanced dataset.

Logistic Regression

Evaluation Metrics

n=165		Predicted: NO	Predicted: YES	
Actual: NO		TN = 50	FP = 10	60
Actual: YES		FN = 5	TP = 100	105
		55	110	

Misclassification Rate
(Error Rate):

- Overall, how often is it **wrong**?
- $(FP + FN) / \text{total} = 15/165 = 0.09$

Logistic Regression

Evaluation Metrics

- **Precision:** percentage of your results which are relevant. Among all the elements that your model classify as positive, how many are really positive examples?

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

- **Recall:** percentage of total relevant results correctly classified by your algorithm.

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

Logistic Regression

Evaluation Metrics

- **F1-Score:** It is the weighted average of the Precision and the recall scores.

$$F1Score = 2\left(\frac{Precision \times Recall}{Precision + Recall}\right)$$

Resources

- Tan, Pang-Ning. *Introduction to data mining*. Pearson Education India, 2006.
- Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. New York: Springer series in statistics, 2001.
- Aurelien Geron, Hands On Machine Learning with SKlearn, Keras and TF

