Big data, data mining and data analytics

Lecture 8

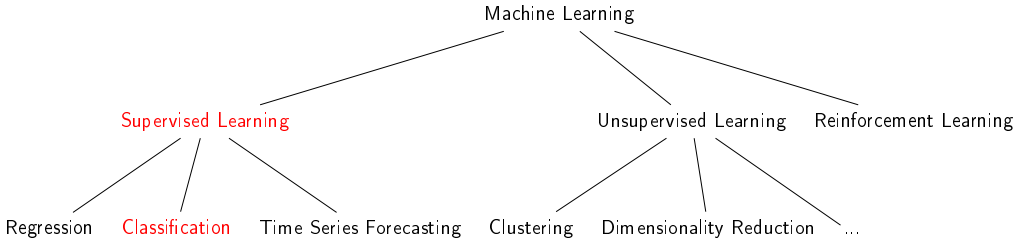
Matteo Tellarini, Federico Bertoni

2024-04-18

Outline

Topic

Machine Learning



Machine Learning

Definition (Classification)

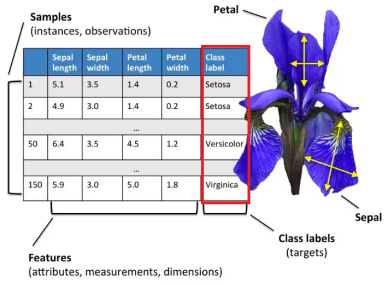
The problem of predicting a qualitative response Y from a set of predictors $X_1, \ldots, X_p$ is called classification.
Given n observations of the response Y and p features, we have

$$\text{Target Y:} \quad \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \qquad \text{X:} \quad \begin{pmatrix} x_{1\,1} & x_{1\,2} & \ldots & x_{1\,p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n\,1} & x_{n\,2} & \ldots & x_{n\,p} \end{pmatrix}$$

where y_i take values from a finite set $\mathcal{Y}$ and $\vec{x}_i = (x_{i\,1}, \ldots, x_{i\,p})$ is a point in $\mathbb{R}^p$.
A classification rule is a function $h : \mathbb{R}^p \to \mathcal{Y}$, so that when we observe a new $\vec{x}_0$ we can predict y to be $h(\vec{x}_0)$.

Machine Learning

Example (Classification)



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica

Figure: For example, $\mathcal{Y} = \{\text{virginica}, \text{versicolor}, \text{setosa}\}$ and the features are sepal length sl, sepal width sw, petal length pl and petal width pw. The classification rule is $h : (sl, sw, pl, pw) \to \mathcal{Y}$. Image from [1].

Machine Learning

Classification

The goal is to find the classification rule h that makes accurate predictions, i.e. one for which the true error rate, defined as

$$L(h) = \mathbb{P}(\{h(X) \neq Y\})$$

is low.

It can be shown that the true error rate is minimized, on average, for a classifier that assigns each observation to the most likely class, given its predictor values. In other words, we assign an observation with predictors $\vec{x}_0$ to the class j for which

$$\mathbb{P}(Y = j | X = \vec{x}_0)$$

is largest. This is the conditional probability, i.e. the probability that $Y = j$ given the predictors $\vec{x}_0$.

Classification

Suppose that the response Y has two levels, 0 and 1, and to know the conditional distribution of Y given X. Then the optimal decision rule[1] is

$$h(X) = \begin{cases} 1 & \text{if} \quad \mathbb{P}(Y = 1|X = \vec{x}_0) > \mathbb{P}(Y = 0|X = \vec{x}_0) \\ 0 & \text{if} \quad \mathbb{P}(Y = 1|X = \vec{x}_0) < \mathbb{P}(Y = 0|X = \vec{x}_0) \end{cases}$$

The points at which $\mathbb{P}(Y = 1|X = \vec{x}_0) = \mathbb{P}(Y = 0|X = \vec{x}_0)$ defines the **decision boundary**.

[1] This is known as the Bayes classification rule.

Machine Learning

Example (Classification)

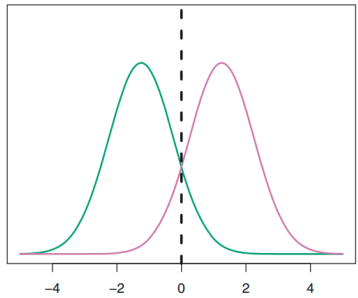


Figure: Suppose to know the conditional probability for a two-class problem. The probability density function for class 1 is shown in green. The probability density function for class 2 is shown in purple. The dashed vertical line represents the Bayes decision boundary. Image from [2].

Machine Learning

Example (Classification)

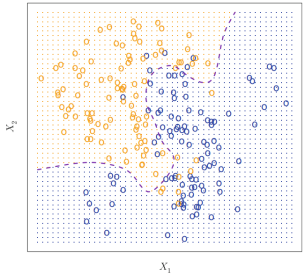


Figure: A two-class problem with two predictors, X_1 and X_2, for which we know the conditional probability. The purple dashed line represents the Bayes decision boundary. The orange background grid indicates the region in which an observation will be assigned to the orange class, and the blue background grid indicates the region in which an observation will be assigned to the blue class Image from [2].

Classification

It is straightforward to generalize to the case for which Y can take on j levels,

$$h(x) = \text{argmax}_k \, \mathbb{P}(Y = k | X = \vec{x})$$

where argmax_k means the value of k that maximizes the following expresssion, i.e. $\mathbb{P}(Y = k | X = \vec{x})$. In this case, k is a value between 1 and j.

Machine Learning

Example (Classification)

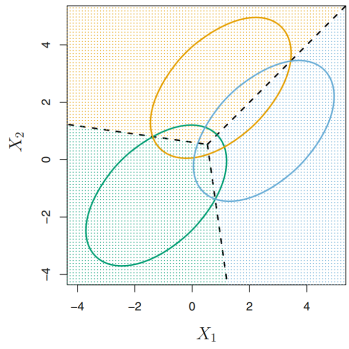


Figure: A three-class problem with two predictors, X_1 and X_2, for which we know the conditional probability. The ellipses contain 95% of the probability for each of the three classes. The dashed lines are the Bayes decision boundaries. Image from [2].

Classification

However, in practice we do not know the conditional probability $\mathbb{P}(Y = j|X = \vec{x_0})$. Also, the true error rate is the one that we would get if we could test the classifier on the entire population. But we don't have access to that.

We only have access to observations, that we could use to estimate the conditional probability and the error rate.

Machine Learning

Classification

To estimate the error rate using a sample of n training data, we can use

$$\hat{L}(h) = \frac{1}{n} \sum_{i=1}^{n} I(h(X_i) \neq Y_i),$$

where $I(h(X_i) \neq Y_i)$ is an indicator variable,

$$I(h(X_i) \neq Y_i) = \begin{cases} 1 & \text{if} \quad h(X_i) \neq Y_i \\ 0 & \text{if} \quad h(X_i) = Y_i \end{cases}$$

In practice, $\hat{L}(h)$ computes the fraction of incorrect classifications. But, since it uses the training data, generally underestimates the true value of $L(h)$. However, ss in the regression setting, we can use techniques such as the validation set approch or the k-fold cross-validation method to improve the estimate.

Machine Learning

Example (Classification)

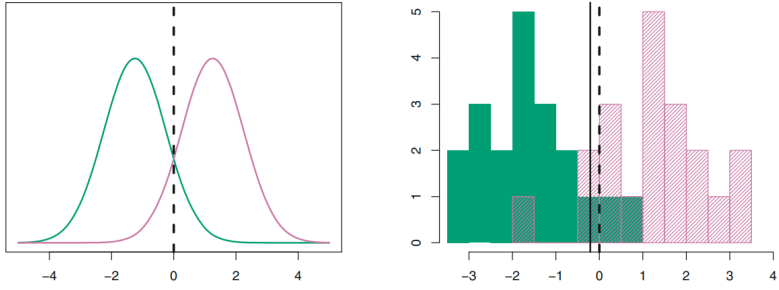


Figure: Right: observations were drawn from the 2 classes, shown as histograms. The Bayes decision boundary is again shown as a dashed vertical line. The solid vertical line represents the decision boundary estimated from the training data. Image from [2].

Machine Learning

Example (Classification)

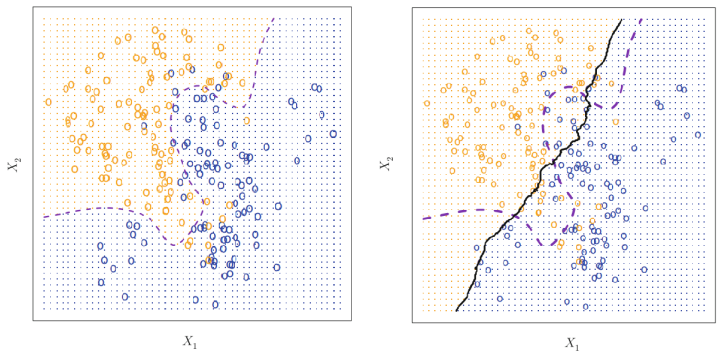


Figure: Right: the decision boundary (solid black line) estimated from the data points. The purple dashed line represents the Bayes decision boundary. Image from [2].

Machine Learning

Example (Classification)

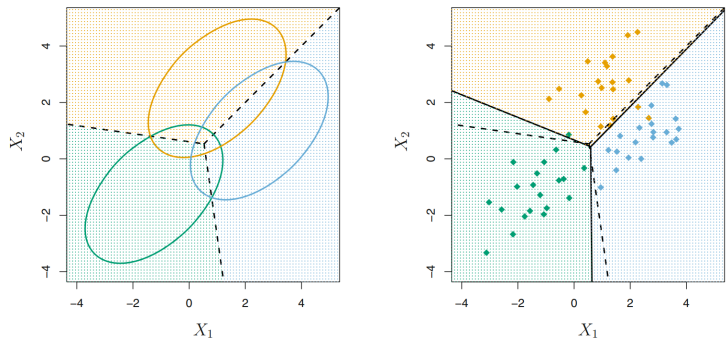


Figure: Right: observations were generated from each class, and the corresponding decision boundaries are indicated using solid black lines. The Bayes decision boundaries are shown as dashed lines. Image from [2].

Machine Learning

Example (Bias-Variance Trade-off in Classification)

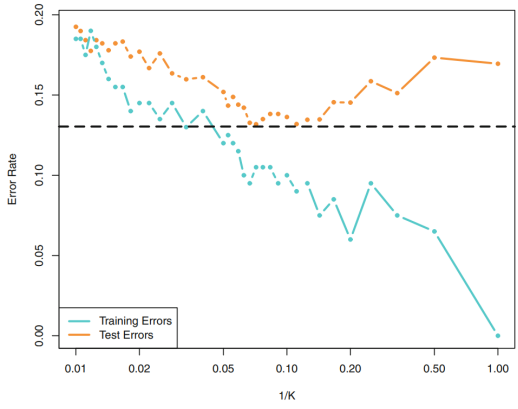


Figure: Training and test error rates as a function of the model flexibility, represented as $1/K$. The black dashed line indicates the bayes error. Image from [2].

Classification

Recipe

Ingredient	Example
Dataset	
Model	KNN, Logistic Regression, Naive Bayes, ...
Cost function	$\hat{L}(h)$, ...
Optimization Method	Maximum Likelihood, ...
Assement	Confusion Matrix, ...

Classification

k-nearest neighbors

Suppose that we have a qualitative response variable with j classes. Given n observations, $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$, an integer value for k and a prediction point $\vec{x}_0$, the k-nearest neighbors classifier works as follows:

1. Identifies the k observations among the n data points that are closest to $\vec{x}_0$. Let's call $\mathbb{D}$ this set of points.
2. Estimate the conditional probability for class j as the fraction of points in $\mathbb{D}$ whose response values equal j:

$$\mathbb{P}(Y = j | X = \vec{x}_0) = \frac{1}{k} \sum_{i \in \mathbb{D}} I(y_i = j)$$

3. Once $\mathbb{P}(Y = j | X = \vec{x}_0)$ is known for all the j classes, classify the new observation $\vec{x}_0$ to the class with the largest probability

Classification

Example (k-nearest neighbors)

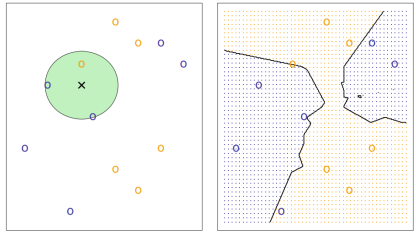


Figure: Left: a new observation is shown with a black cross. Let's use a knn classifier with $k = 3$ to predict whether the new point belongs to the blue or the orange class. The 3 closest points are shown in the green circle. The majority of them are blue, thus the new point is predicted to be of the blue class. Right: The corresponding knn decision boundary. The blue (orange) grid indicates the region in which a point will be classified as blue (orange). Image from [2].

Classification

Logistic Regression

Suppose that we have a qualitative response variable Y with two classes, 0 and 1, and one predictor X.
Given n observations, $(x_1, y_1), \ldots, (x_n, y_n)$, can we estimate the conditional probability using a linear regression model?

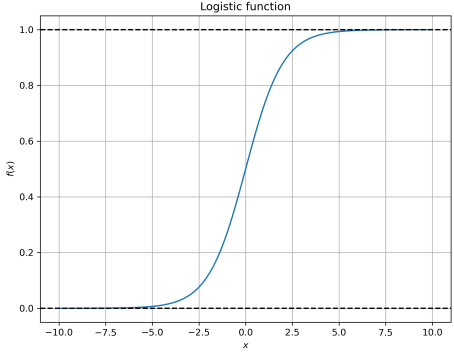
$$\begin{aligned}\mathbb{P}(Y = 1|X = x) &\stackrel{?}{=} \beta_0 + \beta_1 X \\ \mathbb{P}(Y = 0|X = x) &= 1 - \mathbb{P}(Y = 1|X = x)\end{aligned}$$

We can use the least squares method to find the estimates for β_0 and β_1.
However, $\mathbb{P}(Y = 1|X = x)$ and $\mathbb{P}(Y = 0|X = x)$ will not be bounded between 0 and 1, as required by the axioms of probability.

Classification

Logistic Regression

The logistic function provides a remedy to this issue. It is an S-shaped curve, defined as $f(x) = \frac{e^x}{1+e^x}$, that gives outputs between 0 and 1.



Classification

Logistic Regression

We can use the logistic function to model the conditional probability. Therefore, the logistic regression model is

$$\mathbb{P}(Y_i = 1|X = x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}$$
$$\mathbb{P}(Y_i = 0|X = x) = 1 - \mathbb{P}(Y_i = 1|X = x)$$

Classification

Logistic Regression: coefficient estimates

How to estimate the unknown coefficients β_0 and β_1?
The **maximum likelihood** method is usually adopted.

Basically, we seek estimates for β_0 and β_1 such that the estimated conditional probability for each observation $\hat{\mathbb{P}}(Y = 1|X = x)$ corresponds as closely as possible to the observed class. Technically, this is done by finding the values for β_0 and β_1 that maximize the likelihood function, yielding the estimates $\hat{\beta}_0$ and $\hat{\beta}_1$.

The maximum likelihood procedure is already implemented in statistical software packages, so we do not need to worry about that.

Classification

Logistic Regression: interpretation of the coefficients

Let's write for simplicity $p(X) \equiv \mathbb{P}(Y = 1|X = x)$. The logistic regression can be rewritten as

$$p(x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}} \Rightarrow \frac{p(x)}{1 - p(x)} = e^{\beta_0+\beta_1 x}$$

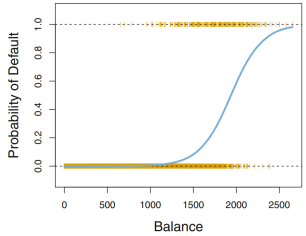
where $\frac{p(x)}{1-p(x)}$ is called the odds[2]. The odds can take on any value between 0 and $+\infty$. A value close to 0 indicate very low probability for $Y = 1$, a large value that it is very likely.

In a logistic regression model, increasing X by one unit changes the odds by a factor of e^{β_1}. In general, if β_1 is positive, increasing X will increase $p(X)$. If β_1 is negative, increasing X wil crease $p(X)$.

[2]Odds are traditionally used instead of probabilities in horse-racing

Classification

Example (Logistic Regression)



	Coefficient	Std. error	Z-statistic	P-value
Intercept	−10.6513	0.3612	−29.5	<0.0001
balance	0.0055	0.0002	24.9	<0.0001

Figure: Left: The dataset (orange ticks) shows the monthly credit card balances for a number of individuals (x axis), indicating on the y axis whether they defaulted on their credit card payments (1) or not (0). The blue curve shows the predicted probabilities of default, given the monghtly balance, using logistic regression. Right: estimated regression coefficients, the related standard errors plus the results of the hypothesis test $H_0 : \beta_i = 0$. Image from [2].

Logistic Regression

Logistic Regression

- What if there are p predictors?
  As in the multiple linear regression setting

$$\mathbb{P}(Y_i = 1|X = x) = \frac{e^{\beta_0+\sum_{j=1}^{p}\beta_j x_{ij}}}{1 + e^{\beta_0+\sum_{j=1}^{p}\beta_j x_{ij}}}$$

- What if a predictor is categorical?
  Code it with a dummy variable
- What if the qualitative response takes more than two levels?
  Extensions to a multi-level qualitative response exist and are implemented in statistical software

Classification

Naive Bayes

Suppose that we have a qualitative response variable Y with j classes and p predictors $X = (x_1, \ldots, x_p)$.
Using the **Bayes Theorem**, we can write the conditional probability of $Y = j$ as

$$\mathbb{P}(Y = j | x_1, \ldots, x_p) = \frac{\mathbb{P}(x_1, \ldots, x_p | Y = j)\mathbb{P}(Y = j)}{\mathbb{P}(x_1, \ldots, x_p)}$$

where $\mathbb{P}(Y = j)$ is called the **prior**, $\mathbb{P}(x_1, \ldots, x_p | Y = j)$ the **likelihood**, $\mathbb{P}(Y = j | x_1, \ldots, x_p)$ the **posterior** probability.

Classification

Naive Bayes

The Naive Bayes classifier makes two simplifications:

1. Assume conditional independence on each predictor given the class label j. Thus,

$$\begin{aligned}\mathbb{P}(x_1, \ldots, x_p | Y = j) =& \mathbb{P}(x_1 | Y = j) \times \mathbb{P}(x_2 | Y = j) \times \cdots \times \mathbb{P}(x_p | Y = j) \\ =& \prod_{i=1}^{p} \mathbb{P}(x_i | Y = j)\end{aligned}$$

2. Remove the term $\mathbb{P}(x_1, \ldots, x_p)$. It appears as the denominator of $\mathbb{P}(Y = j | x_1, \ldots, x_p)$ for all the class labels j and thus has no impact on the relative probabilities

Classification

Naive Bayes

Under these assumptions, the Naive Bayes classifier approximates the conditional probability to

$$\mathbb{P}(Y = j|x_1, \ldots, x_p) \propto \prod_{i=1}^{p} \mathbb{P}(x_i|Y = j)\mathbb{P}(y = j)$$

But since these terms are not known, we have to estimate them from the data. $\hat{\mathbb{P}}(y = j)$ is simply the fraction of training data points that belongs to class j. Estimating $\mathbb{P}(x_i|Y = j)$ depends on the nature (categorical or numerical) of the feature x_i,

Classification

Naive Bayes

Finally, the Naive Bayes classifier assigns a class to an observation using the rule

$$h(x) = \text{argmax}_k \, \hat{\mathbb{P}}(y = k) \prod_{i=1}^{p} \hat{\mathbb{P}}(x_i | Y = j)$$

An observation is predicted to be of class j if it gives the highest value in the expression above, compared to the value computed for all the remaining classes.

Classification

Example (Naive Bayes)

Suppose to use a Naive Bayes classifier to predict whether a person would go out or stay in on one day, based on weather conditions "Sunny", "Rainy", and "Cloudy".
The feature weather condition is categorical and the response is binary, "Stay In" or "Go Out".
Let's say that we have 10 observation for the behaviour of that person.

Classification

Example (Naive Bayes)

X (Weather)	Y (Decision)
Sunny	Go Out
Rainy	Stay In
Cloudy	Go Out
Sunny	Go Out
Sunny	Stay In
Rainy	Stay In
Sunny	Go Out
Cloudy	Stay In
Rainy	Stay In
Sunny	Go Out

Classification

Example (Naive Bayes)

Let's calculate the priors, i.e. the overall probabilities of going out and staying in:

$$\mathbb{P}(\text{Go Out}) = 5/10 = 0.5$$
$$\mathbb{P}(\text{Stay In}) = 5/10 = 0.5$$

Classification

Example (Naive Bayes)

Next, calculate the likelihoods, i.e. the probabilities of seeing each weather condition given each decision:

$$\mathbb{P}(\text{Sunny}|\text{Go Out}) = 3/5 = 0.6$$
$$\mathbb{P}(\text{Rainy}|\text{Go Out}) = 0/5 = 0$$
$$\mathbb{P}(\text{Cloudy}|\text{Go Out}) = 2/5 = 0.4$$
$$\mathbb{P}(\text{Sunny}|\text{Stay In}) = 1/5 = 0.2$$
$$\mathbb{P}(\text{Rainy}|\text{Stay In}) = 3/5 = 0.6$$
$$\mathbb{P}(\text{Cloudy}|\text{Stay In}) = 1/5 = 0.2$$

Classification

Example (Naive Bayes)

Finally, suppose that the wheather is sunny, what behaviour would the Naive Bayes algorithm predict?

$$\mathbb{P}(\text{Go Out}|\text{Sunny}) \propto \mathbb{P}(\text{Sunny}|\text{Go Out})\mathbb{P}(\text{Go Out}) = 0.6 \times 0.5 = 0.3$$
$$\mathbb{P}(\text{Stay In}|\text{Sunny}) \propto \mathbb{P}(\text{Sunny}|\text{Stay In})\mathbb{P}(\text{Stay In}) = 0.2 \times 0.5 = 0.1$$

The Naive Bayes classifier predicts that the person will go out.

Classification

Example (Gaussian Naive Bayes)

If x_i is a continuous variable, the likelihood is usually assumed to be Gaussian

$$\mathbb{P}(x_i|Y=j) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x_i-\mu)^2}{\sigma^2}}$$

Then, the mean μ and the variance σ^2 are estimated from the data.

Topic

Standardization

Standardization is a preprocessing step that transforms the features so that they have a mean of 0 and a standard deviation of 1.
The transformation, also known as z-score normalization, goes as follows

$$x_i \Rightarrow z_i = \frac{x_i - \bar{x}}{s}$$

where $\bar{x} = \sum x_i$ and $s = \frac{1}{n}\sum(x_i - \bar{x})^2$.
Standardization brings all features to the same scale, which is important for many machine learning algorithms.
WARNING: when a model is trained using a standardized variable, the same transformation (using the same $\bar{x}$ and s) must to be applied to any new data (including the test set) before classification!

Standardization

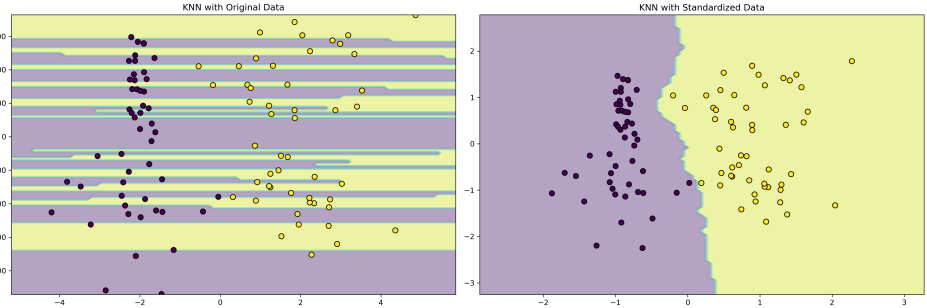


Figure: Left: the estimated decision boundary is affected by the different scale of the two features. Right: using standardization we get the expected decision boundary.

Topic

Confusion Matrix

A confusion matrix is a convenient way to display the performance of a classifier

	Predicted: Positive	Predicted: Negative
Actual: Positive	True Positive (TP)	False Negative (FN)
Actual: Negative	False Positive (FP)	True Negative (TN)

TP and TN are the correct guesses. In general, a good classifier should have large TP and TN and small (ideally zero) numbers for FP and FN.

Confusion Matrix

- **TP**: number of positive instances the classifier correctly identified as positive
- **FP**: number of instances the classifier identified as positive but in reality are negative
- **TN**: number of negative instances the classifier correctly identified as negative
- **FN**: number of instances classified as negative but in reality are positive

Confusion Matrix

- **Accuracy**: ratio of the total number of correct predictions oever the total number of predictions

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

- **Precision**: ratio of true positives to the sum of true and false positives

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

It is the ability of the classifier not to label as positive a sample that is negative

Confusion Matrix

- **Recall** (or **sensitivity**): ratio of true positives to the sum of true positives and false negatives

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

  It is the ability of the classifier to find all the positive samples

- **Specificity**: ratio of true negatives to the sum of true negatives and false positives

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

  It is the ability of a classifier to avoid false positives

- **F1 Score**: weighted mean of Precions and Recall

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Confusion Matrix

Example (Testing for a disease)

Suppose that a classifier predicted

- 100 people correctly as having the disease (True Positives)
- 870 people correctly as not having the disease (True Negatives)
- 20 people incorrectly as having the disease (False Positives)
- 10 people incorrectly as not having the disease (False Negatives)

The resulting confusion matrix is

	Predicted: Disease	Predicted: No Disease
Actual: Disease	100 (TP)	10 (FN)
Actual: No Disease	20 (FP)	870 (TN)

Confusion Matrix

Example (Testing for a disease)

Let's compute the metrics:

- Accuracy: $(100 + 870)/(100 + 20 + 10 + 870) = 0.97$.
  The model correctly predicted the disease status for 97% of the patients.
- Precision: $100/(100 + 20) = 0.83$.
  When the model predicts a patient has the disease, it's correct 83% of the time.
- Recall: $100/(100 + 10) = 0.91$.
  The model correctly identifies 91% of all patients with the disease.
- F1 Score: $2 \times (0.83 \times 0.91)/(0.83 + 0.91) = 0.87$.
  The F1 score is a balance between Precision and Recall.
- Specificity: $870/(870 + 20) = 0.98$.
  The model correctly identifies 98% of all patients without the disease.

Confusion Matrix

Example (Testing for a disease)

In a clinical setting, depending on the severity of the disease, we might want to maximize Recall to ensure we catch as many true cases as possible, even at the risk of some False Positives.

Confusion Matrix

In a multiclass classification problem, the confusion matrix can still be used to describe the performance of a classification model.
Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class.
For example, the confusion matrix for a 3-class classification problem is

	Predicted C1	Predicted C2	Predicted C3
Actual C1	TP1	FN12	FN13
Actual C2	FN21	TP2	FN23
Actual C3	FN31	FN32	TP3

Confusion Matrix

The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier.
For example, for each class we can compute:

- Precision: Ratio of correctly predicted positive observations to the total predicted positive observations for that class

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

Confusion Matrix

- Recall (or sensitivity): Ratio of correctly predicted positive observations to the all observations in actual class

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- F1 Score: weighted average of Precision and Recall

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

These metrics are computed for each class and then averaged

Topic

Dataset

Example (sklearn)

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
# Load the Breast Cancer Wisconsin dataset
cancer = load_breast_cancer()
nrows, ncols = cancer.data.shape
print(f"Dataset with {nrows} and {ncols} features")
# Split the dataset into a training set and a test set
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, test_size=0.2,
    stratify=cancer.target, random_state=42
)
```

```
Dataset with 569 and 30 features
```

Standardization

Example (sklearn)

```
from sklearn.preprocessing import StandardScaler
# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

KNN

Example (sklearn)

```
from sklearn.neighbors import KNeighborsClassifier
# Train a KNN classifier on the training set
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
# Predict the labels of the test set
y_pred = knn.predict(X_test)
```

Confusion Matrix

Example (sklearn)

```
from sklearn.metrics import confusion_matrix
# Print the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

```
Confusion Matrix:
[[40  2]
 [ 0 72]]
```

Classification metrics

Example (sklearn)

```
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score
)
# Compute metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
tn, fp, fn, tp = cm.ravel()
specificity = tn / (tn+fp)
```

Classification Metrics

Example (sklearn)

```
print(f'Accuracy: {accuracy:.3f}, Precision: {precision:.3f}')
print(f'Recall: {recall:.3f}, F1 Score: {f1:.3f}')
print(f'Specificity: {specificity:.3f}')
```

```
Accuracy: 0.982, Precision: 0.973
Recall: 1.000, F1 Score: 0.986
Specificity: 0.952
```

Logistic Regression

Example (statsmodels)

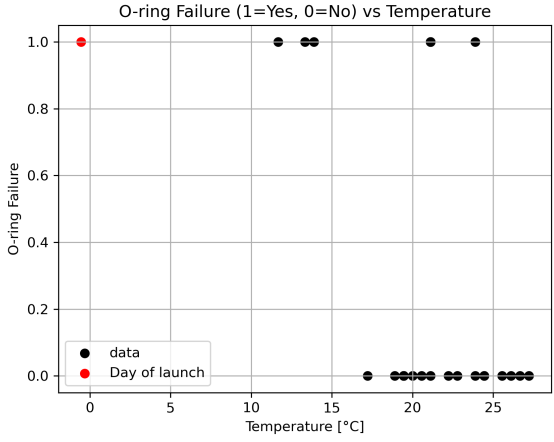
```
import os
import pandas as pd
import statsmodels.api as sm

def fahrenheit_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5.0/9.0

# Load the Challenger O-ring dataset into a pandas DataFrame
df = pd.read_csv(os.path.join('..', 'tutorials', 'challenger.csv'))
# Define the feature columns and the target column
X, y = fahrenheit_to_celsius(df['Temperature']), df['Failure']
```

Logistic Regression

Example (statsmodels)



Logistic Regression

Example (statsmodels)

```
# Add a constant to the feature columns
X = sm.add_constant(X)

# Train a logistic regression model on the dataset
model = sm.Logit(y, X)
result = model.fit()

# Print the model summary
print(result.summary())
```

Logistic Regression

Example (statsmodels)

```
Optimization terminated successfully.
         Current function value: 0.479801
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                Failure   No. Observations:                   24
Model:                          Logit   Df Residuals:                       22
Method:                          MLE   Df Model:                            1
Date:                Mon, 15 Apr 2024   Pseudo R-squ.:                 0.2052
Time:                        17:42:51   Log-Likelihood:                -11.515
converged:                       True   LL-Null:                       -14.487
Covariance Type:            nonrobust   LLR p-value:                  0.01477
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.3931      3.054      1.766      0.077      -0.592      11.378
Temperature   -0.3084      0.150     -2.053      0.040      -0.603      -0.014
==============================================================================
```

- Info about model creation
- Degrees of freedom
- Pseudo R^2 is the proportion of variance explained by the model, similar to R^2 in linear regression

Logistic Regression

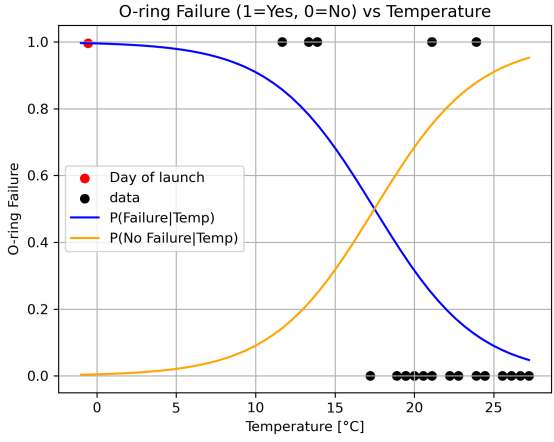
Example (statsmodels)

```
Optimization terminated successfully.
         Current function value: 0.479801
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                Failure   No. Observations:                   24
Model:                          Logit   Df Residuals:                       22
Method:                          MLE   Df Model:                            1
Date:               Mon, 15 Apr 2024   Pseudo R-squ.:                 0.2052
Time:                       17:42:51   Log-Likelihood:                -11.515
converged:                      True   LL-Null:                       -14.487
Covariance Type:            nonrobust   LLR p-value:                  0.01477
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.3931      3.054      1.766      0.077      -0.592      11.378
Temperature   -0.3084      0.150     -2.053      0.040      -0.603      -0.014
==============================================================================
```

- p-value of the test that the predictors have no effect (null hypothesis), against the alternative that at least one predictor have an impact
- Intercept and Temperature coefficients, followed by
  - estimated value
  - estimated SE
  - z-statistic
  - p-value
  - 95% confidence interval

Logistic Regression

Example (statsmodels)



Logistic Regression

Example (sklearn)

```
from sklearn.linear_model import LogisticRegression
X = fahrenheit_to_celsius(df[['Temperature']]).to_numpy()
y = df['Failure'].to_numpy()
# Train a logistic regression model on the training set
clf = LogisticRegression(random_state=0).fit(X, y)
# Print the coefficients of the logistic regression model
print(f"beta_1={clf.coef_}, beta_0={clf.intercept_}")
```

```
beta_1=[[-0.30166293]], beta_0=[5.25855951]
```

Logistic Regression

Example (sklearn)

```
new_temp = np.array([[-0.56]])
pred_class = clf.predict(new_temp)
est_prob = clf.predict_proba(new_temp)
print(f"Predicted class: {pred_class}")
print(f"Est. prob. class 0 - No Fail: {est_prob[0, 0]:.3f}")
print(f"Est. prob. class 1 - Fail: {est_prob[0, 1]:.3f}")
```

```
Predicted class: [1]
Est. prob. class 0 - No Fail: 0.004
Est. prob. class 1 - Fail: 0.996
```

Naive Bayes

Example (sklearn)

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
# Load the Iris dataset
iris = load_iris()
# Split the dataset into a training set and a test set
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, test_size=0.5,
    stratify=iris.target, random_state=42
)
```

Naive Bayes

Example (sklearn)

```
from sklearn.naive_bayes import GaussianNB
# Train a Gaussian Naive Bayes classifier on the training set
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Predict the labels of the test set
y_pred = gnb.predict(X_test)
```

Naive Bayes

Example (sklearn)

```
from sklearn.metrics import confusion_matrix
# Compute the confusion matrix
cm = confusion_matrix(y_test, y_pred)
# Print the confusion matrix
print(cm)
```

```
[[25  0  0]
 [ 0 24  1]
 [ 0  3 22]]
```

Naive Bayes

Example (sklearn)

```
from sklearn.metrics import classification_report
cr = classification_report(y_test, y_pred)
print(cr)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	25
1	0.89	0.96	0.92	25
2	0.96	0.88	0.92	25
accuracy			0.95	75
macro avg	0.95	0.95	0.95	75
weighted avg	0.95	0.95	0.95	75

Topic

Homeworks

- Read chapter 2.1.5, 2.2.3, 4.1, 4.2, 4.3 from An introduction to Statistical Learning
- `08-1_boston_houses_revisited.ipynb`
- `08-2_breast_cancer.ipynb`
- Complete the remaining classworks (if any)

Topic

References I

[1] *Exploring the Iris flower dataset*. https://eminebozkus.medium.com/exploring-the-iris-flower-dataset-4e000bcc266c. Accessed: 2024-04-03.

[2] G. James et al. *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer New York, 2014. ISBN: 9781461471370.