ITSAR ISTITUTO TECNICO SUPERIORE
ANGELO RIZZOLI

Algoritmi per il Machine Learning

Dott. Antonio Giovanni Lezzi

MACHINE LEARNING

Dott. Antonio Giovanni Lezzi

ALGORITMI DI MACHINE LEARNING

- **Dimensionamento fattore K**
- Valutazione di un modello
- Matrice di confusione
- Vari metodi di valutazione in Python

Dott. Antonio Giovanni Lezzi

COME SCEGLIAMO IL FATTORE K?

- Una parte fondamentale dell'algoritmo K-Nearest Neighbors è la scelta di k.
- Nell'esempio della maglietta di Martina, relativo alla decisione della taglia da decidere, abbiamo utilizzato il valore 5 per k.
- In questo caso, ogni nuovo data point è previsto dai 5 suoi vicini più prossimi nel set di allenamento.
- Tuttavia, **k è un valore regolabile, possiamo impostarlo su qualsiasi valore compreso tra 1 e il numero di data points nel set di allenamento**.

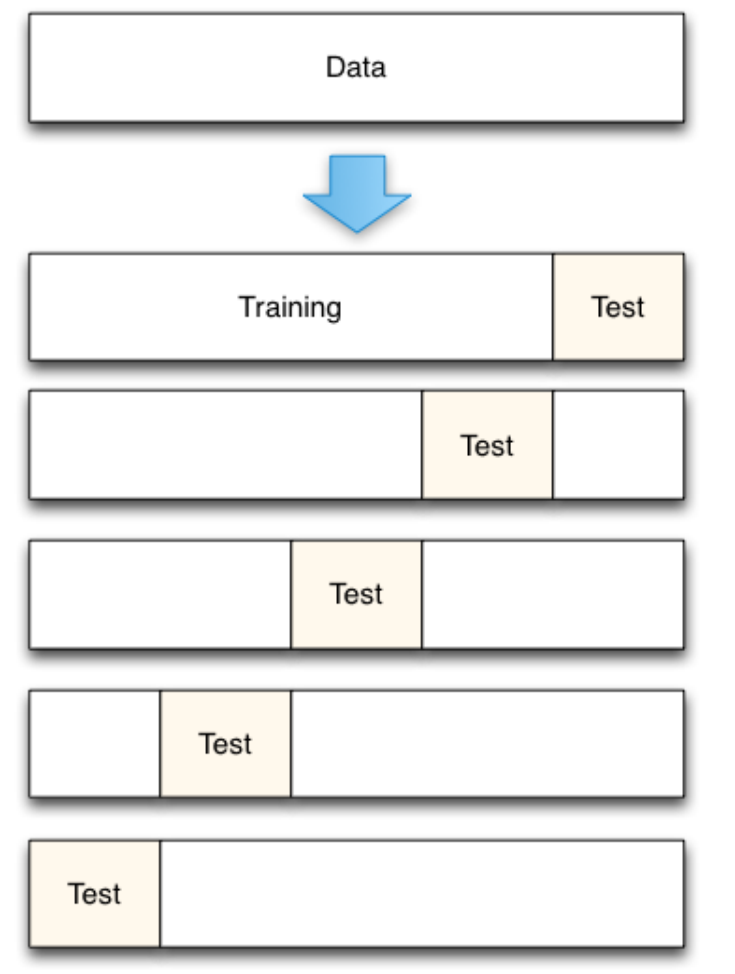
COME SCEGLIAMO IL FATTORE K?

- La scelta del valore di K è un compito cruciale per la buona riuscita della previsione.
- Tra i metodi più utilizzati troviamo il metodo della **convalida incrociata** o **cross validation**.
- L'idea generale di questo metodo è quella di **dividere il campione di dati in un numero k** definito di sottocampioni o segmenti ad estrazione casuale, disgiunti, che serviranno come **fase di allenamento** (training).
- Si applica poi il modello KNN per fare **previsioni sul segmento k-esimo** (ad esempio, utilizzando i segmenti k-1 come esempi) **e si valuta l'errore**.

COME SCEGLIAMO IL FATTORE K?

- In pratica, ogni volta uno dei sottoinsiemi k viene usato come set di test, mentre gli altri sottoinsiemi k-1 sono messi insieme per formare un set di allenamento.

- Come si può vedere, <u>ogni data point può trovarsi in un set di test esattamente una volta, e può trovarsi in un set di allenamento k-1 volte</u>.



COME SCEGLIAMO IL FATTORE K?

- Alla fine della procedura, **gli errori calcolati vengono mediati per fornire una misura della stabilità del modello** (ossia quanto bene il modello prevede nuove istanze).

- **I passaggi precedenti vengono quindi ripetuti per vari k e il valore che raggiunge l'errore più basso** (o la massima precisione di classificazione) viene così selezionato come valore ottimale per k (ottimale in un senso di convalida incrociata).

COME SCEGLIAMO IL FATTORE K?

- Si noti come la **convalida incrociata sia dispendiosa** dal punto di vista computazionale e si dovrebbe essere pronti a far funzionare l'algoritmo per un po' di tempo, specialmente quando la dimensione del campione di esempi è ampia.

- In alternativa, puoi specificare un **k ragionevole qualora si conosca il problema di analisi** (ad esempio, dalle precedenti analisi KNN che si è potuto condurre su dati simili).

ALGORITMI DI MACHINE LEARNING

- Dimensionamento fattore K
- **Valutazione di un modello**
- Matrice di confusione
- Vari metodi di valutazione in Python

Dott. Antonio Giovanni Lezzi

VALUTAZIONE

- Per valutare un algoritmo in genere si utilizzano le seguenti metriche:
  - **Matrice di confusione**
  - **Precisione**
  - **Richiamo**
  - **Punteggio F1**
  - **Supporto**

MATRICE DI CONFUSIONE

- La **matrice di confusione**, ossia un riepilogo dei risultati di previsione su un problema di classificazione.

- In questa matrice, il numero di previsioni corrette e non corrette viene riepilogato con valori di conteggio e suddiviso per ogni classe, per comprendere i modi in cui il modello di classificazione è confuso quando fa previsioni.

PRECISIONE

- La precisione è l'abilità di un classificatore **di non etichettare un'istanza positiva che è in realtà negativa**.
- Per ogni classe è definito come il rapporto tra veri positivi e la somma di veri e falsi positivi.
- Detto in un altro modo, "**per tutte le istanze classificate come positive, quale percentuale era corretta?**"

RICHIAMO

- È la capacità di un classificatore di **trovare tutte le istanze positive**.
- Per ogni classe è definito come il rapporto tra i veri positivi e la somma dei veri positivi e dei falsi negativi.
- Detto in altri termini, "**per tutte le istanze che erano effettivamente positive, quale percentuale è stata classificata correttamente?**"

PUNTEGGIO F1

- Il punteggio F1 è una **media armonica ponderata delle due precedenti metriche** in modo tale che il punteggio migliore sia 1,0 e il peggiore sia 0,0.

- Come regola generale, la media ponderata di F1 dovrebbe essere **utilizzata per confrontare i modelli di classificatore, non la precisione globale**.

SUPPORTO

- Il supporto è il **numero di occorrenze effettive della classe nel set di dati specificato**.
- Il supporto squilibrato nei dati di addestramento può indicare debolezze strutturali nei punteggi riportati del classificatore e potrebbe indicare la necessità di campionamento stratificato o ribilanciamento.
- Il supporto non cambia tra i modelli, ma invece diagnostica il processo di valutazione.

VALUTAZIONE

- I metodi **confusion_matrix** e **classification_report** di **sklearn.metrics** possono essere utilizzati per calcolare queste metriche.

```
# valutiamo l'algoritmo
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

- L'output dello script sopra appare così:

```
[[13  0  0]
 [ 0 14  0]
 [ 0  0  9]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        14
           3       1.00      1.00      1.00         9

   micro avg       1.00      1.00      1.00        36
   macro avg       1.00      1.00      1.00        36
weighted avg       1.00      1.00      1.00        36
```

VALUTAZIONE

- I risultati mostrano che l'algoritmo KNN è stato in grado di classificare tutti i 36 record nel set di test con una precisione del 100%, che è eccellente.
- Sebbene l'algoritmo funzioni molto bene con dei set di dati, non ci si deve aspettarsi gli stessi risultati per ogni volta che si eseguono queste righe di codice e per altre applicazioni.
- Infatti, KNN non sempre funziona altrettanto bene con le caratteristiche di elevata dimensione o di tipo categoriale.

CONFRONTO DEL TASSO DI ERRORE CON IL VALORE K

- Nella sezione di allenamento e previsione non c'è modo di sapere in anticipo quale valore di K produce i migliori risultati al primo tentativo.
- Si è scelto come valore di K il valore di default 5 e si dà il caso che si ottenga una precisione del 100%.
- Un modo per aiutare a trovare il valore migliore di K è quello di tracciare il grafico del valore K e il tasso di errore corrispondente per il set di dati.

CONFRONTO DEL TASSO DI ERRORE CON IL VALORE K

- Infatti tracceremo l'errore medio per i valori previsti del set di test per tutti i valori K compresi tra 1 e 40.
- Per fare ciò, calcoliamo innanzitutto la media dell'errore per tutti i valori previsti in cui K varia da 1 a 40.

```
# Valutiamo cosa succede al variare di K

error = []

# Calcoliamo l'errore per K che varia tra 1 e 40
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error.append(np.mean(pred_i != y_test))
```

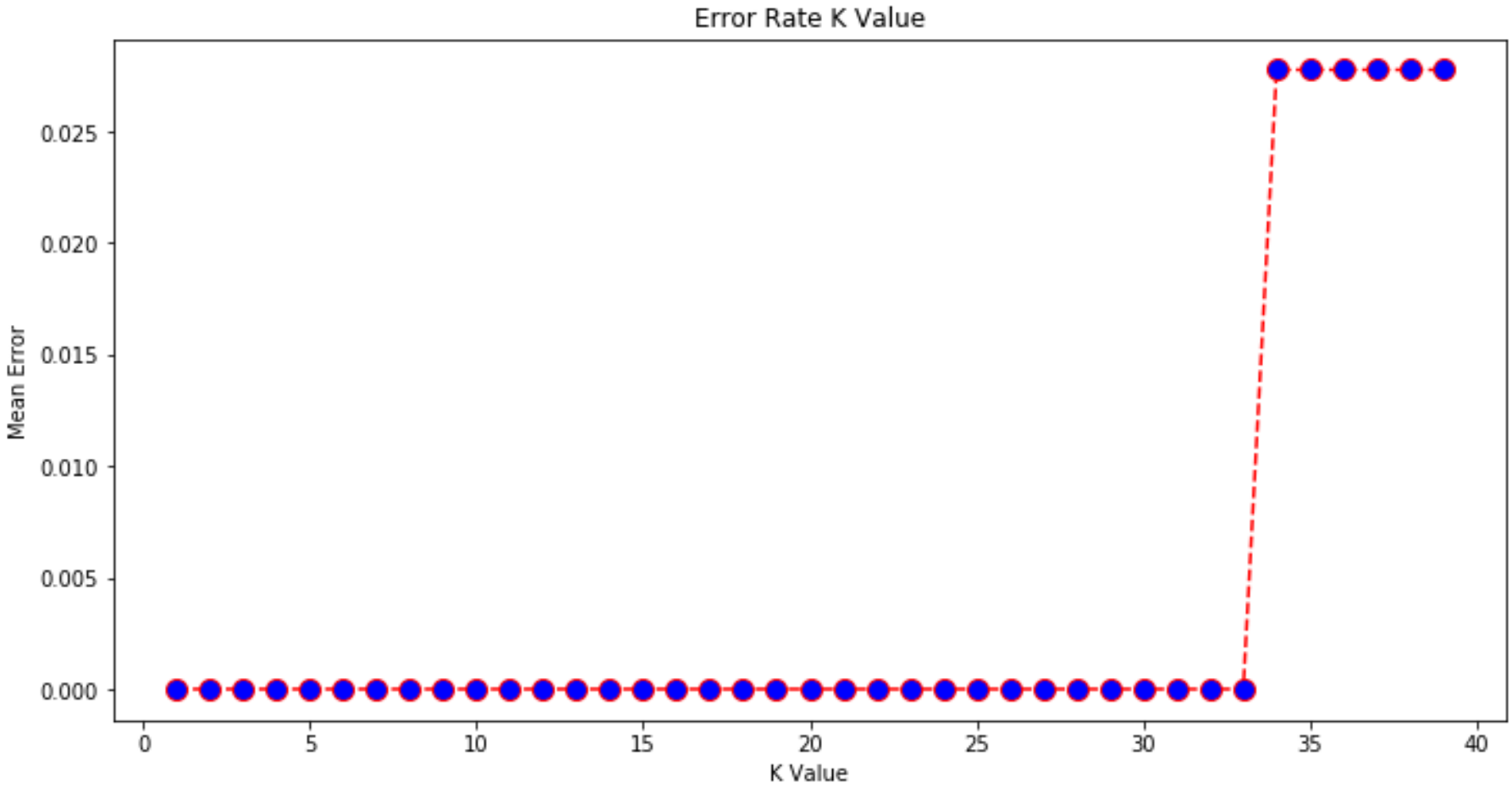
CONFRONTO DEL TASSO DI ERRORE CON IL VALORE K

- Lo script precedente esegue un ciclo da 1 a 40.
- In ogni iterazione viene calcolato l'errore medio per i valori previsti del set di test e il risultato viene aggiunto all'elenco degli errori.
- Il prossimo passo è tracciare i valori di errore rispetto ai valori di K.

```
#Rappresentiamo il variare di K su un grafico
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')

plt.xlabel('K Value')

plt.ylabel('Mean Error')
```


Error Rate K Value
Mean Error
0.025
0.020
0.015
0.010
0.005
0.000
0
5
10
15
20
25
30
35
40
K Value

CONFRONTO DEL TASSO DI ERRORE CON IL VALORE K

- Dal grafico possiamo vedere che l'errore medio è zero quando il valore di K è compreso tra 0 e 33.
- Si consiglia di giocare con il valore di K per vedere come influisce sulla precisione delle previsioni.

CONFRONTO DEL TASSO DI ERRORE CON IL VALORE K

- **Nota bene**: Ripetendo lo stesso codice molto probabilmente non si otterrà lo stesso risultato.

- Questo è dovuto al fatto che probabilmente il calcolo iterativo dell'errore varia di volta in volta, come anche il risultato della matrice di confusione cambia ad ogni esecuzione dell'algoritmo, in quanto non tiene in memoria quanto accaduto nelle esecuzioni precedenti dell'algoritmo stesso.

CONCLUSIONE

- KNN è un algoritmo di classificazione semplice ma potente.
- Non richiede alcuna formazione per fare previsioni, che di solito è una delle parti più difficili di un algoritmo di apprendimento automatico.
- L'algoritmo KNN è stato ampiamente utilizzato per trovare la somiglianza dei documenti e il riconoscimento dei pattern.
- È stato anche impiegato per lo sviluppo di sistemi di raccomandazione e per la riduzione della dimensionalità e le fasi di pre-elaborazione per la visione artificiale, in particolare i compiti di riconoscimento facciale.

ALGORITMI DI MACHINE LEARNING

- Dimensionamento fattore K
- Valutazione di un modello
- **Matrice di confusione**
- Vari metodi di valutazione in Python

Dott. Antonio Giovanni Lezzi

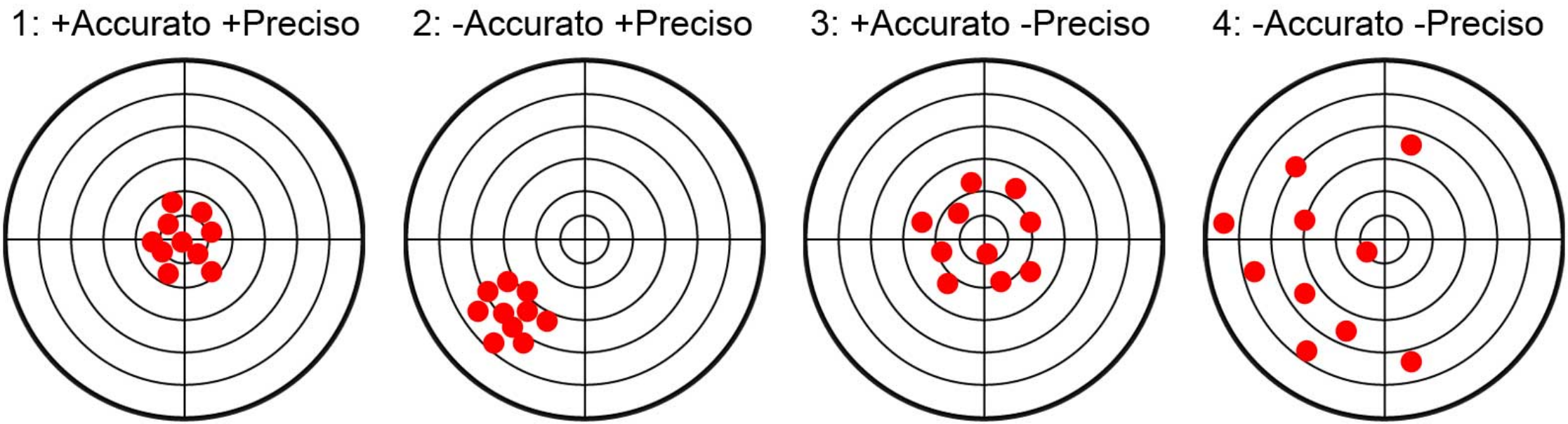
MATRICE DI CONFUSIONE

- il **falso positivo**, analogo all'**errore di primo tipo**, è il risultato di un test che porta erroneamente ad accettare l'ipotesi sulla quale esso è stato condotto.

- il **falso negativo**, analogo all'**errore di secondo tipo**, è il risultato di un test che porta erroneamente a rifiutare l'ipotesi sulla quale esso è stato condotto.

MATRICE DI CONFUSIONE

Tabella dei tipi di errore		**L'ipotesi <u>nulla</u> (*H*$_0$) è**	
		Vera	**Falsa**
La decisione circa l'ipotesi nulla (*H*$_0$) è	**Non rigetta**	Inferenza corretta ***(vero negativo)***	Errore di tipo 2 ***(Falso Negativo)***
	Rigetta	Errore di tipo 1 ***(Falso Positivo)***	Inferenza corretta ***(Vero Positivo)***


1: +Accurato +Preciso
2: -Accurato +Preciso
3: +Accurato -Preciso
4: -Accurato -Preciso

MATRICE DI CONFUSIONE

- Per valutare un modello di machine learning che analizza grandi dataset si può pensare di usare la **matrice di confusione** (meglio conosciuta come **confusion matrix**).
- Osserviamo con maggiore dettaglio il **principio di funzionamento** e come i valori al suo interno vengono calcolati.

CHE COSA É?

- Una matrice di confusione è una tabella in cui le **previsioni** sono rappresentate nelle colonne e lo **stato effettivo** è rappresentato dalle righe (a volte questo è invertito, con istanze reali in colonne e previsioni nelle righe).

- La tabella rende facile vedere se si è verificato un errore di denominazione e se le previsioni sono più o meno corrette.

MATRICE DI CONFUSIONE

- Dalla tabella è possibile comprendere le performance del modello predittivo di classificazione in modo da determinare quanto **sia accurato ed efficace**.
- Possiamo avere due tipi di matrice di confusione:
  - **A 2 Classi**: tentano di prevedere le performance di un modello predittivo a due classi (Si/No, Vero/Falso, e così via);
  - **Multiclasse**: tentano di prevedere le performance di un modello predittivo formato da più di due classi.

MATRICE DI CONFUSIONE A DUE CLASSI

- La matrice di confusione a due classi viene utilizzata per prevedere il risultato di un classificatore binario, ovvero due valori di classe o etichette, come Si / No e 1/0, per dati di input conosciuti.
- La classe di interesse viene solitamente indicata come "**positiva**" e l'altra come "**negativa**".
- Ipotizziamo di avere una matrice di confusione per un modello di **spam** su un modello a due classi:

n = 300	Previsti: NO	Previsti: SI
Effettivi: NO	100	30
Effettivi: SI	20	150

MATRICE DI CONFUSIONE

- Notiamo che ci sono due "**classi di previsione**": Si rappresenta lo spam di mail (classe positiva), No una mail normale non considerata spam (classe negativa).

- Il numero totale di previsioni eseguite è pari a 300 (mail testate dal modello). Di queste 300 previsioni, 180 sono mail di spam e 120 no.

- Effettivamente però sono solo 170 (20 + 150) che sono spam. Allo stesso modo le mail effettive che non sono considerate spam sono 130 (100 + 30).

n= 300	**Previsti: No**	**Previsti: Si**	
Effettivi: No	TN = 100	FP = 30	*130*
Effettivi: Si	FN = 20	TP = 150	*170*
	120	*180*	

MATRICE DI CONFUSIONE

- Per comprendere meglio l'incrocio tra valori effettivi abbiamo:
  - **True positive** (TP, o veri positivi): sono i casi in cui abbiamo previsto che la mail fosse spam, e lo sono realmente.
  - **True negative** (TN, o veri negativi): il modello ha previsto che la mail non è spam e non lo è veramente.
  - **False positive** (FP, o falsi positivi): sono i casi in cui abbiamo previsto che le mail fossero spam, ma in realtà non lo erano (conosciuti anche come errori di I tipo).
  - **False negative** (FN, o falsi negativi): il modello ha previsto che la mail non è spam, anche se in realtà lo è (conosciuti anche come errori di II tipo).

MATRICE DI CONFUSIONE

- Nella matrice di confusione precedente è subito evidente che il modello lascia passare email di spam (false negative).
- Circa 15 email di spam sono ritenute valide.
- Sono invece meno frequenti i falsi positivi, soltanto 5 email valide ritenute erroneamente

		CLASSI PREVISTE	
		SI	NO
CLASSI EFFETTIVE	SI	TRUE POSITIVE (TP) 35	FALSE NEGATIVE (FN) 15
	NO	FALSE POSITIVE (FP) 5	TRUE NEGATIVE (TN) 25

risposte corrette: 35+25 = 60
risposte errate: 15+5 = 20

METRICHE MATRICE DI CONFUSIONE

- Tra le metriche base e più diffuse della matrice di confusione possiamo trovare:
  - **Error Rate** (o **Tasso di errore**): è il tasso di errore (ERR) e viene calcolato come il numero di tutti i pronostici errati diviso per il numero totale del set di dati. Il miglior tasso di errore è 0, mentre il peggiore è 1.

$$ERR = \frac{FP + FN}{TN + FP + FN + TP}$$

		CLASSI PREVISTE	
		SI	NO
CLASSI EFFETTIVE	SI	TRUE POSITIVE (TP) **35**	FALSE NEGATIVE (FN) **15**
	NO	FALSE POSITIVE (FP) **5**	TRUE NEGATIVE (TN) **25**

tasso di errore

$$\frac{15 + 5}{35 + 25 + 15 + 5} = \frac{20}{80} = 25\%$$

risposte corrette: 35+25 = 60

risposte errate: 15+5 = 20

MATRICE DI CONFUSIONE

- **Accuracy (o accuratezza)**: indica l'accuratezza del modello verso il come dice il nome. Pertanto, la migliore accuratezza è 1, mentre la peggiore è 0.
  Può anche essere calcolato da 1 – ERR, o dalla formula qui riportata

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

		CLASSI PREVISTE	
		SI	NO
CLASSI EFFETTIVE	SI	TRUE POSITIVE (TP) **35**	FALSE NEGATIVE (FN) **15**
	NO	FALSE POSITIVE (FP) **5**	TRUE NEGATIVE (TN) **25**

Accuratezza

$$\frac{35 + 25}{35 + 25 + 15 + 5} = \frac{60}{80} = 75\%$$

risposte corrette: 35+25 = 60

risposte errate: 15+5 = 20

MATRICE DI CONFUSIONE

- **Precision (o precisione)**: La precisione è l'abilità di un classificatore di non etichettare un'istanza positiva che è in realtà negativa.
  Per ogni classe è definito come il rapporto tra veri positivi e la somma di veri e falsi positivi.
  Detto in un altro modo, "per tutte le istanze classificate come positive, quale percentuale era corretta?".

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

		CLASSI PREVISTE	
		SI	NO
CLASSI EFFETTIVE	SI	TRUE POSITIVE (TP) **35**	FALSE NEGATIVE (FN) **15**
	NO	FALSE POSITIVE (FP) **5**	TRUE NEGATIVE (TN) **25**

Precisione

$$\frac{35}{35+5} = \frac{35}{40} = 87.5\%$$

risposte corrette: 35+25 = 60

risposte errate: 15+5 = 20

MATRICE DI CONFUSIONE

- **Recall (o richiamo)**: detta anche sensitivity o true positive, è la capacità di un classificatore di trovare tutte le istanze positive.
  Per ogni classe è definito come il rapporto tra i veri positivi e la somma dei veri positivi e dei falsi negativi.
  Detto in altri termini, "per tutte le istanze che erano effettivamente positive, quale percentuale è stata classificata correttamente?".

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

		CLASSI PREVISTE	
		SI	NO
CLASSI EFFETTIVE	SI	TRUE POSITIVE (TP) **35**	FALSE NEGATIVE (FN) **15**
	NO	FALSE POSITIVE (FP) **5**	TRUE NEGATIVE (TN) **25**

Sensitività

$$\frac{35}{35 + 15} = \frac{35}{50} = 70\%$$

risposte corrette: 35+25 = 60

risposte errate: 15+5 = 20

MATRICE DI CONFUSIONE

- **Specificity (o specificità, SP)**: è calcolata come il numero di previsioni negative corrette diviso per il numero totale di negativi (N). Viene anche chiamato vero tasso negativo (TNR).

- La migliore specificità è 1, mentre la peggiore è 0.

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

		CLASSI PREVISTE	
		SI	NO
CLASSI EFFETTIVE	SI	TRUE POSITIVE (TP) **35**	FALSE NEGATIVE (FN) **15**
	NO	FALSE POSITIVE (FP) **5**	TRUE NEGATIVE (TN) **25**

Specificità (specificity)

$$\frac{25}{5 + 25} = \frac{25}{30} = 83.3\%$$

risposte corrette: 35+25 = 60

risposte errate: 15+5 = 20

MATRICE DI CONFUSIONE

- **False positive rate (o rateo falsi positivi)**: Il tasso di falsi positivi (FPR) viene calcolato come il numero di previsioni positive errate diviso per il numero totale di negativi.

$$FPR = \frac{FP}{TN + FP} = 1 - SP$$

- Il miglior tasso di falsi positivi è 0 mentre il peggiore è 1.

- Può anche essere calcolato come 1 – specificità.

		CLASSI PREVISTE	
		SI	NO
CLASSI EFFETTIVE	SI	TRUE POSITIVE (TP) **35**	FALSE NEGATIVE (FN) **15**
	NO	FALSE POSITIVE (FP) **5**	TRUE NEGATIVE (TN) **25**

Tasso dei falsi positivi

$$\frac{5}{5+25} = \frac{5}{30} = 16.6\%$$

risposte corrette: 35+25 = 60

risposte errate: 15+5 = 20

MATRICE DI CONFUSIONE

- **F-score (o punteggio f)**: è una media armonica ponderata delle metriche Precision e Recall in modo tale che il punteggio migliore sia 1 e il peggiore sia 0.

$$F - score = \frac{2\ x\ Recall\ x\ Precision}{Recall + Precision}$$

- Come regola generale, la media ponderata di F1 dovrebbe essere utilizzata per confrontare i modelli di classificatore, non la precisione globale.

$$FS = \frac{2 \cdot 0.70 \cdot 0.875}{0.70 + 0.875} = \frac{1.225}{1.575} = 0.77$$

MATRICE DI CONFUSIONE A PIÙ CLASSI

- La matrice di confusione può essere costruita anche su più classi. Non solo su due.
- Ecco un esempio pratico di confusion matrix multiclasse con tre classi.

		CLASSI PREVISTE		
		classe 1	classe 2	classe 3
CLASSI EFFETTIVE	classe 1			
	classe 2			
	classe 3			

MATRICE DI CONFUSIONE A PIÙ CLASSI

- Supponiamo di avere un campione di 25 animali, ad es. 7 gatti, 8 cani e 10 uccelli.
- La matrice di confusione del nostro algoritmo di riconoscimento può essere simile alla seguente tabella:

		Previsti		
		Cane	Gatto	Uccello
Effettivi	Cane	6	2	0
	Gatto	1	6	0
	Uccello	1	1	8

MATRICE DI CONFUSIONE A PIÙ CLASSI

- In questa matrice di confusione, il sistema predice correttamente **sei degli otto cani effettivi**, ma in due casi scambia un cane per un gatto.
- I sette **gatti** sono stati correttamente **riconosciuti in sei casi**, ma in un caso un gatto è stato considerato un cane.
- Di solito, è difficile prendere un uccello per un cane o un gatto, ma questo è quello che è successo al nostro classificatore in due casi. Eppure, **otto uccelli su dieci erano stati riconosciuti correttamente**.
- Puoi vedere che tutte le previsioni corrette si trovano nella **diagonale della tabella**, quindi gli errori di previsione possono essere facilmente trovati, poiché saranno rappresentati da valori esterni alla diagonale.

MATRICE DI CONFUSIONE A PIÙ CLASSI

- Per calcolare i valori di **True Positive**, **True Negative**, **False Positive** e **False Negative** in una matrice di confusione multiclasse si può procedere a calcolare il valore singolo di TP, TN, FP e FN specifico per quella classe, avremo che:

• **Classe Cane:** TP = 6 TN = 6 + 8 = 14 FP = 1 + 1 = 2 FN = 2	• **Classe Gatto:** TP = 6 TN = 6 + 8 = 14 FP = 1 FN = 3	• **Classe Uccello:** TP = 8 TN = 6 + 6 = 12 FP = 2 FN = 0

Metriche Matrice di confusione	Classe Cane	Classe Gatto	Classe uccello	Totale
ERR	0,167	0,167	0,091	0,141
Accuracy	0,833	0,833	0,909	0,859
Precision	0,750	0,857	0,800	0,802
Recall	0,750	0,667	1,000	0,806
SP	0,875	0,933	0,857	0,888
FPR	0,125	0,067	0,143	0,112
F-score	0,750	0,750	0,889	0,796

ALGORITMI DI MACHINE LEARNING

- Dimensionamento fattore K
- Valutazione di un modello
- Matrice di confusione
- **Vari metodi di valutazione in Python**

Dott. Antonio Giovanni Lezzi

METODO HOLDOUT

- Si è visto il metodo di validazione più semplice che consente di suddividere il data set in training set e test set, quest'ultimo di grandezza pari a 20-30%.
- Utilizzando il parametro random_state con un valore fisso, ad ogni esecuzione del codice si ottengono i medesimi risultati, perché la suddivisione rimane la medesima.
- Un esempio in python, in cui viene creata la lista X, test_size viene settato al 30% e random_state pari a 42

VALUTAZIONI IN PYTHON

- Un esempio in python, in cui viene creata la lista X, test_size viene settato al 30% e random_state pari a 42

```
##Holdout - test train split
from sklearn.model_selection import train_test_split

X = [[10, 7], [3, 8], [1, 4], [5, 2], [2, 4], [3, 6]]

X_train, X_test = train_test_split(
    X ,test_size= 0.3, random_state = 42)

print ("X_train: ", X_train)
print ("X_test: ", X_test)
```

```
X_train:  [[3, 6], [1, 4], [2, 4], [5, 2]]
X_test:  [[10, 7], [3, 8]]
```

LEAVE ONE OUT CROSS VALIDATION (LOOCV)

- Si divide il set di dati in due parti:
- In una parte abbiamo un'unica osservazione, che sono i nostri dati di test e nell'altra parte, abbiamo tutte le altre osservazioni del set di dati di allenamento (set di training).
- Se disponiamo di un set di dati con n osservazioni, i dati di allenamento contengono n-1 osservazioni e i dati di test contengono un'unica osservazione.

LEAVE ONE OUT CROSS VALIDATION (LOOCV)

- Questo processo è ripetuto per ciascun data points e la ripetizione di questo processo n volte genera n volte la metrica di accuratezza del modello.

- Non è consigliabile utilizzare questa convalida per dataset molto grandi, in quanto il tempo computazionale sarebbe molto elevato.

LEAVE ONE OUT CROSS VALIDATION (LOOCV)

```python
# LOOCV
import numpy as np
from sklearn.model_selection import LeaveOneOut

X = np.array([[10, 7], [3, 8], [1, 4], [5, 2], [2, 4], [3, 6]])

LOO = LeaveOneOut()
LOO.get_n_splits(X)

for train_index, test_index in LOO.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
```

```
TRAIN: [1 2 3 4 5] TEST: [0]
TRAIN: [0 2 3 4 5] TEST: [1]
TRAIN: [0 1 3 4 5] TEST: [2]
TRAIN: [0 1 2 4 5] TEST: [3]
TRAIN: [0 1 2 3 5] TEST: [4]
TRAIN: [0 1 2 3 4] TEST: [5]
```

LEAVE ONE OUT CROSS VALIDATION (LOOCV)

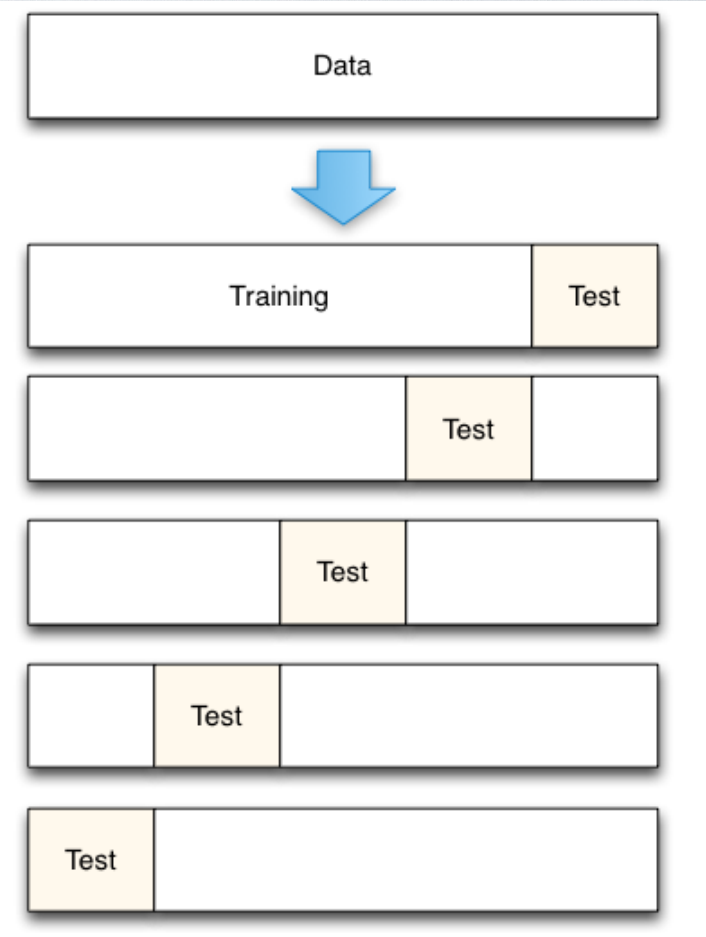
- Le suddivisioni eseguite sono pari a 6, come il numero di osservazioni nella variabile X.

- Da notare che vengono stampati gli indici di X_train e X_test. Ad esempio l'indice 1 fa riferimento all'elemento [3,8], l'indice 4 all'elemento [2,4], rispettivamente nella posizione 2 e 5 (perché si parte dall'indice 0) di X.

- Maggiori informazioni sulla convalida LOOCV le puoi trovare nella pagina di Sklearn.

CROSS VALIDATION K FOLD

- La convalida incrociata k-fold prevede di dividere casualmente il set di dati in k gruppi, dette anche pieghe (fold) di dimensioni approssimativamente uguali.
- La prima piega viene conservata per i test, mentre il modello viene addestrato sulle pieghe k-1.
- Il processo viene ripetuto K volte e ogni volta per la convalida vengono utilizzate pieghe diverse o un diverso gruppo di punti dati.
- Ciò significa che ogni data point deve trovarsi in un set di test esattamente una volta e deve trovarsi in un set di allenamento k-1 volte.



CROSS VALIDATION K FOLD

- Lo svantaggio di questo metodo è che l'algoritmo di allenamento deve essere rieseguito da zero k volte: ne consegue che ci vuole k volte più tempo per fare una valutazione.
- Se ad esempio la metrica che vogliamo ottimizzare è l'errore quadratico medio, con questo tipo di cross validation, otteniamo k volte Mean Squared Error (MSE): MSE_1, MSE_2, … MSE_K, quindi l'errore di questa cross validation viene calcolato prendendo la media dell'MSE sulle k suddivisioni.
- La convalida LOOCV, vista poco fa, è una variante di K fold, dove k = n. Tipicamente il valore di K in K-fold è 5 o 10.

CROSS VALIDATION K FOLD

```python
# KFOLD
import numpy as np
from sklearn.model_selection import KFold

X = np.array([[10, 7], [3, 8], [1, 4], [5, 2], [2, 4], [3, 6]])

kf = KFold (n_splits = 5)
kf.get_n_splits(X)

for train_index, test_index in kf.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
```

```
TRAIN: [2 3 4 5] TEST: [0 1]
TRAIN: [0 1 3 4 5] TEST: [2]
TRAIN: [0 1 2 4 5] TEST: [3]
TRAIN: [0 1 2 3 5] TEST: [4]
TRAIN: [0 1 2 3 4] TEST: [5]
```

CROSS VALIDATION K FOLD

- È possibile poi ottenere un array per visualizzare i risultati di ogni fold di test tramite il metodo **cross_val_score**.
- Maggiori informazioni in merito alla k-fold cross validation si trovano alla pagina di Sklearn.

CROSS VALIDATION STRATIFIED

- La stratificazione è una tecnica in cui riordiniamo i dati in modo tale che ogni piega abbia una buona rappresentazione dell'intero set di dati.
- Obbliga ogni piega ad avere almeno m istanze di ogni classe.
- Questo approccio garantisce che una classe di dati non sia sovrapesata soprattutto quando la variabile target è sbilanciata.
- Inoltre, aiuta a ridurre sia il bias che la varianza.

CROSS VALIDATION STRATIFIED

```
import numpy as np
from sklearn.model_selection import StratifiedKFold

X = np.array([[1,2,3,4,5], [3,4,5,6,7], [1,2,3,4,5], [3,4,5,6,7],
              [1,2,3,4,5],[3,4,5,6,7],[1,2,3,4,5], [3,4,5,6,7],
              [1,2,3,4,5], [3,4,5,6,7]])
y = np.array([0,0,0,0,0, 1,1,1,1,1])
kfs = StratifiedKFold (n_splits =5)
kfs.get_n_splits(X,y)

for train_index, test_index in kfs.split(X,y):
    print("TRAIN:", train_index, "TEST:", test_index)
```

```
TRAIN: [1 2 3 4 6 7 8 9] TEST: [0 5]
TRAIN: [0 2 3 4 5 7 8 9] TEST: [1 6]
TRAIN: [0 1 3 4 5 6 8 9] TEST: [2 7]
TRAIN: [0 1 2 4 5 6 7 9] TEST: [3 8]
TRAIN: [0 1 2 3 5 6 7 8] TEST: [4 9]
```

CROSS VALIDATION STRATIFIED

- A differenza degli altri metodi visti, questo tipo di convalida incrociata necessita obbligatoriamente della variabile target y (classe).
- È stato definito n_splits = 5, con questa impostazione occorre disporre di almeno 5 campioni per ciascuna delle classi che si hanno. Avendo due classi (0 e 1), avremo 10 campioni.

Classe	Numero di campioni per classe	Percentuale
0	5	50%
1	5	50%

CROSS VALIDATION STRATIFIED

- Nell'esempio StratifiedKFold si cerca di mantenere il rapporto numero di membri per classe / n_splits del 50% in ogni piega.
- Con n_splits = 5 significa che in una sola piega, sia per la classe 0 che 1 dobbiamo avere almeno 5/5 = 1 membro.
- Se avessimo ottenuto un valore minore di uno, anche solo per una delle due classi, avremmo ottenuto un errore.
- Maggiori informazioni sulla convalida StratifieldKFold le puoi trovare alla pagina di Sklearn.

QUANDO UTILIZZARE LA CROSS VALIDATION?

- Gli esempi di convalida incrociata suddivide i set di training e di test per fornire una misura più accurata della qualità del modello. Tuttavia, può richiedere più tempo per l'esecuzione, poiché stima più modelli.

- Quindi, dati questi compromessi, quando si deve usare ogni approccio?

- Il miglior consiglio è quello di sperimentare e trovare una tecnica per il problema che si sta analizzando che sia veloce e che produca stime ragionevoli delle prestazioni che si possa usare per prendere decisioni.

QUANDO UTILIZZARE LA CROSS VALIDATION?

- Per piccoli set di dati, dove l'onere computazionale aggiuntivo non è un grosso problema, è consigliabile eseguire la **convalida incrociata** (la standard è la **k-fold**).

- Per set di dati più grandi, ci si può avvalere di un **singolo set di convalida**. Il codice verrà eseguito più rapidamente e potresti disporre di dati sufficienti per cui non è necessario riutilizzarli per il controllo.

- Anche se non esiste una soglia semplice per ciò che costituisce un set di dati grande o piccolo, indicativamente quando un modello impiega un paio di minuti o meno per essere eseguito, probabilmente vale la pena passare alla convalida incrociata.

QUANDO UTILIZZARE LA CROSS VALIDATION?

- In alternativa, puoi eseguire la convalida incrociata e vedere se i punteggi per ciascun esperimento sembrano vicini. Se così è, forse è sufficiente un singolo set di validazione.

- Per set di dati non bilanciati, dove cioè per ogni classe troviamo un numero differente di osservazioni, allora si può pensare di utilizzare la **convalida incrociata stratificata**, in quanto tiene conto del fatto che in ogni piega la distribuzione dei campioni tra le classi viene mantenuta costante.

- La tecnica **LOOCV** può essere utile quando si cerca di bilanciare la varianza nelle prestazioni stimate, nella velocità di allenamento del modello e nella dimensione del set di dati.