**Politecnico di Torino**

COLLEGIO DI MATEMATICA
Corso di Laurea Magistrale in Ingegneria Matematica

# Descriptor Silhouette come identificatore del class-based concept drift

Un'analisi della sua generalità applicativa

Candidato:
**Tedesco Valeria**

Relatore:
**Ing. Tania Cerquitelli**

**2020**

*A Carlo*

# Ringraziamenti

Prima di procedere con la trattazione, vorrei dedicare qualche riga a tutti coloro che mi sono stati vicini in questo percorso di crescita personale e professionale.
In primo luogo, vorrei ringraziare i miei genitori e tutta la mia famiglia, per avermi sempre supportata e per aver creduto in me anche quando ho pensato di non farcela.
Un sentito grazie va alla mia relatrice, Tania Cerquitelli, e a Francesco Ventura, per la loro disponibilità, la loro pazienza e per le conoscenze trasmesse durante tutto il percorso di stesura dell'elaborato.
Ringrazio inoltre Antonella, Giulia, Julia, Sara e Valentina, con cui ho condiviso parte del mio percorso accademico; grazie per le risate durante le giornate di lezione che sembravano non finire mai e per il vostro supporto.
Un grazie di cuore va ad Alessandro, Alice e Simone, senza di voi non sarei la persona che sono ora.
Infine, vorrei ringraziare Andrea, che è stato la mia forza in questo ultimo periodo; grazie per essermi stato sempre vicino, per avermi supportata in ogni mia scelta e per avermi dato il coraggio di cui avevo bisogno per portare a termine questo percorso.

# Indice

# Elenco delle figure

# Prefazione

In molti contesti, che vanno dal text mining alla produzione industriale, lo streaming dei dati collezionati per questi processi è spesso continuo ed è naturale che ci siano dei cambiamenti nella loro distribuzione. Questo fenomeno è detto *concept drift*, e il suo studio è diventato necessario nella fase di training e learning dei modelli di machine learning.
In questo elaborato, dopo aver descritto il problema del concept drift in tutte le sue forme, andremo a focalizzarci su quelle variazioni nel flusso dei dati dovute alla comparsa di una loro nuova classe, non presente in fase di training (*class-based concept drift*). In particolare, andremo ad analizzare la soluzione al problema proposta in [1], andandone a valutare l'effettiva efficacia su dei nuovi dataset.

# Capitolo 1

# Il Concept Drift

> *"Concept drift is a phenomenon in which the statistical properties of a target domain change over time in an arbitrary way"* [4]

Il *Concept Drift* descrive cambiamenti imprevedibili nella distribuzione di dati nel tempo. Detta in altri termini, le proprietà statistiche della variabile target, che il modello sta cercando di prevedere, possono cambiare nel tempo in modi imprevisti.
Per questo motivo, il concept drift è stato identificato come la causa della diminuzione di efficacia in molti data-driven information systems. Un esempio banale di questo fenomeno è com'è cambiato l'utilizzo dei telefoni cellulari negli ultimi 20 anni: si è passati dall'usare quasi solo esclusivamente le chiamate a non usarle quasi più, preferendo l'uso di internet, dei contenuti multimediali e della fotocamera.
La presenza di concept drift e l'incertezza del tipo di dati e della loro distribuzione sono fattori intrinsechi dei big data; per questo motivo è diventato necessario aggiungere lo studio del concept drift nella fase di training/learning dei modelli di machine learning, affrontando le fasi di *drift detection* (se

il drift avviene o meno), *drift understanding* (quando, dove e come avviene) e *drift adaptation* (reazione alla presenza del drift).

## 1.1 Descrizione e analisi del problema

Dati che adesso potrebbero essere visti come semplice rumore in un futuro potrebbero rivelarsi dati importanti per l'analisi e per la costruzione di modelli. Per questo motivo, è molto importante lo studio del concept drift. Andiamo quindi ad analizzare nel dettaglio il problema, iniziando fornendone una descrizione formale: [2] [3]

Dato un periodo $[0, t]$, definiamo un insieme di samples $S_{0,t} = \{d_0, \ldots, d_t\}$, dove $d_i = (X_i, y_i)$ è una osservazione, $X$ è il vettore delle caratteristiche, $y$ è il vettore delle etichette e $S_{0,t}$ segue una certa distribuzione di probabilità $F_{0,t}(X, y)$. Si dice che avviene un concept drift al tempo $t + 1$ quando si ha che

$$F_{0,t}(X, y) \neq F_{t+1,\infty}(X, y)$$

ovvero se $\exists t : P_t(X, y) \neq P_{t+1}(X, y)$.
Dato che vale che $P_t(X, y) = P_t(X) \times P_t(y|X)$ possiamo andare a definire tre fonti di concept drift, che vengono descritti nella figura 1.1:

- **Virtual drift**: avviene quando $P_t(X) \neq P_{t+1}(X)$ ma $P_t(y|X) = P_{t+1}(y|X)$. (space drift)
- **Actual drift**: avviene quando $P_t(y|X) \neq P_{t+1}(y|X)$ ma $P_t(X) = P_{t+1}(X)$. Questo causa una diminuzione dell'accuratezza del modello. (decision boundary drift)
- Misto delle due precedenti fonti: avviene quando $P_t(X) \neq P_{t+1}(X)$ e $P_t(y|X) \neq P_{t+1}(y|X)$.

La figura 1.1 mostra molto bene come queste fonti differiscono tra loro nel caso di dataset a due feature:

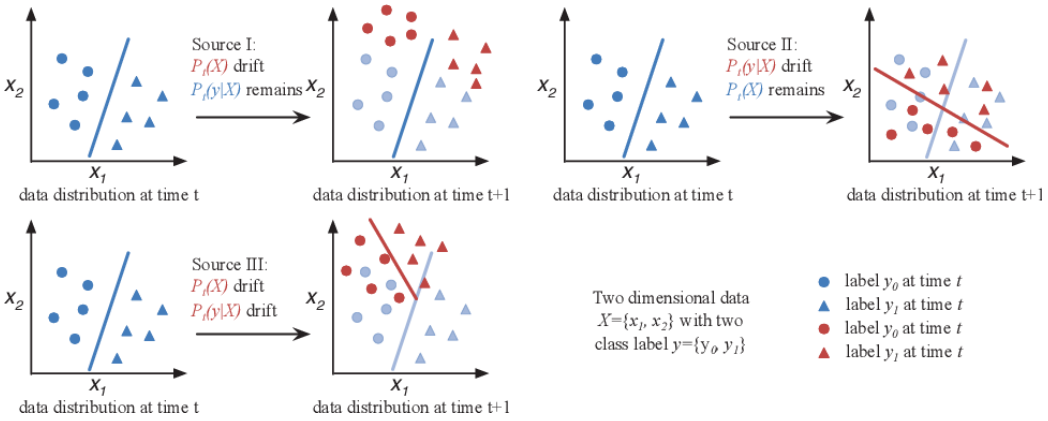



Figura 1.1: Le varie fonti di concept drift, [6]

Inoltre, si possono definire 4 tipi di concept drift, schematizzati nella figura 1.2:

- **Sudden drift**: avviene quando esiste un momento preciso in cui i dati osservati prima e dopo quel punto sono generati da due fonti diverse.

- **Gradual drift**: simile al sudden drift, ma in questo caso si alternano i dati delle due fonti.

- **Incremental drift**: avviene una modifica graduale del dato, che passa dal tipo 1 al tipo 2.

- **Recurring concepts**: ogni tanto si ripresentano dati del vecchio tipo.

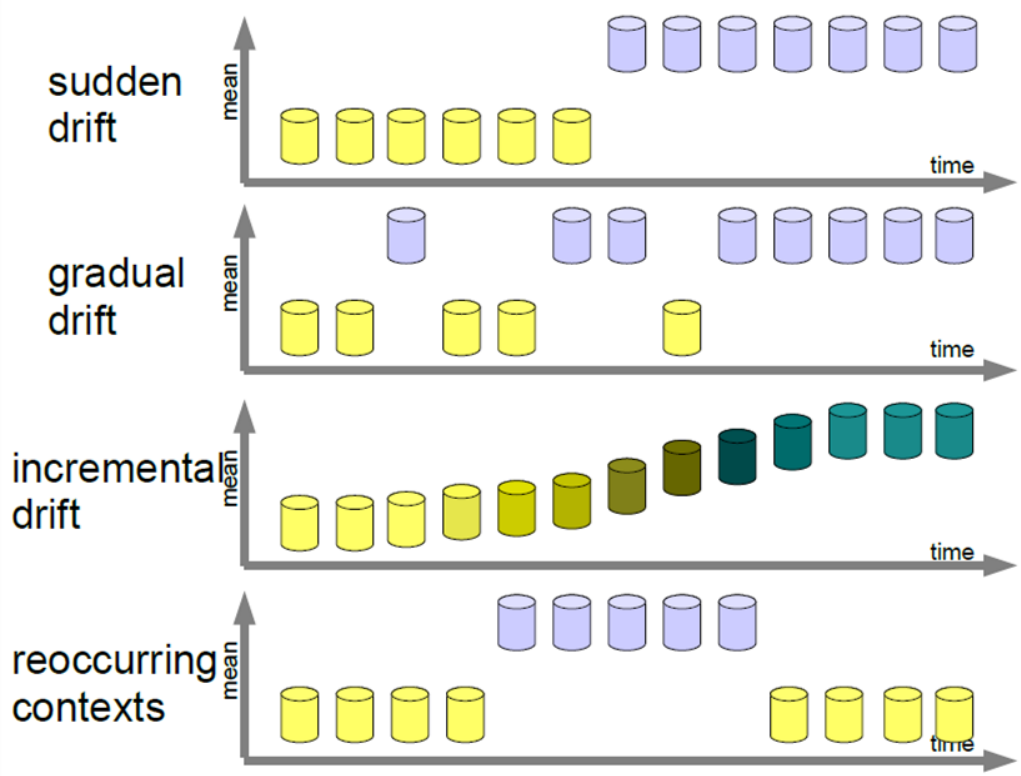



Figura 1.2: I tipi di concept drift, [5]

### 1.1.1 Fase I: Concept drift detection

> *"Drift detection refers to the techniques and mechanisms that characterize and quantify concept drift via identifying change points or change time intervals"* [7]

In generale, si possono identificare 4 fasi per l'identificazione del drift:

1. **Data Retrieval**: seleziona blocchi di dati significativi per l'analisi.

2. **Data Modeling**: identifica le feature che impattano di più sul sistema se si presenta un drift; questa fase è opzionale.

3. **Test Statistics Calculation**: misura la dissimilarità/stima la distanza. Quantifica la "gravità" del drift ed è considerata una fase cruciale per l'identificazione del drift, in quanto non è ancora chiaro come definire una buona misura di dissimilarità. In questa fase vengono creati i "test statistics" per la fase 4.

4. **Hypothesis test**: valuta il significato statistico del cambiamento rilevato nella fase 3. Determina l'accuratezza dell'identificazione del drift provando i limiti statistici dei "test statistics" proposti in fase 3. Questa fase è molto importante perché, senza di essa, i risultati ottenuti nella fase di *test statistics calculation* sarebbero inutili in quanto non è definito quanto è "probabile" che il cambiamento rilevato sia effettivamente un drift e non del semplice rumore. I test d'ipotesi più utilizzati sono: *stima della distribuzione del test statistics*, *bootstrapping*, *permutation test* e *Hoeffding's inequality-based bound identification*.

### Algoritmi di concept drift detection

Esistono sostanzialmente tre tipi di algoritmi per l'individuazione del drift, di cui ne andremo a fornire una generale trattazione.

- *Error Rate-Based Drift Detection*: si basano sul tener traccia dell' *online error rate* dei classificatori di base; se si riesce a provare che un aumento o un decremento dell'errore è statisticamente significativo si passerà ad aggiornare il modello.

- *Data Distribution-Based Drift Detection*: usano una funzione che calcola la distanza per valutare quanto la distribuzione dei dati storici e quelli nuovi siano dissimili. Come prima, se la distanza è statisticamente significativa, allora bisogna addestrare un nuovo modello. Questi

algoritmi sono in grado di identificare in maniera accurata il tempo a cui avviene il drift e di dare informazioni riguardo il luogo in cui avviene il drift, ma quasi sempre hanno un costo computazionale molto elevato. Inoltre, necessitano una definizione di *time window* dei dati storici e di quelli nuovi da parte dell'utente.

- *Multiple Hypothesis Test Drift Detection*: Applicano tecniche simili a quelle usate dai due metodi precedenti, ma utilizzano *multiple hypothesis test* per identificare il drift in vari modi.

### 1.1.2 Fase II: Concept drift understanding

In questa fase vengono esplicate informazioni riguardo al "quando", "come" e "dove" avviene il concept drift. Queste informazioni risultano come output della fase precedente (*concept drift detection*) e fanno da input per la fase successiva (*concept drift adaptation*).

#### Il momento in cui avviene il drift

Nella fase di drift detection viene identificato il momento $t$ in cui avviene il drift, ovvero il momento nel quale avviene che

$$P_t(X, y) \neq P_{t+1}(X, y)$$

È estremamente necessario essere molto accurati in questa fase in quanto un errore (un ritardo o un "falso allarme") nella individuazione dell'istante in cui avviene il drift porta allo sviluppo di un learning system impreciso. Tipicamente, negli algoritmi di drift detection viene settato un allarme che compare ogni volta che viene identificato un drift; questi allarmi sono di solito controllati da un tasso predefinito di errore, oltre al quale l'errore viene valutato come significativo e segnalato. Come abbiamo già visto in

precedenza non esiste un solo tipo di drift, per cui bisogna identificare quando il drift inizia, il change period e il punto in cui il drift termina.
Si può evincere facilmente che il timestamp del drift negli algoritmi esistenti è ritardato rispetto al momento reale del drift, in quanto questi algoritmi hanno bisogno di un numero minimo di dati che confermino o meno la presenza di drift; per questo motivo, alcuni algoritmi contengono un warning che si scatena quando un drift potrebbe essere accaduto.

### Valutazione della gravità del drift

Valutare la "gravità" del drift significa misurare quanto sono dissimili i dati nuovi da quelli vecchi; in formule, si può introdurre una variabile $\Delta$ così definita

$$\Delta = \delta(P_t(X, y), P_{t+1}(X, y))$$

e dire che maggiore è il valore di $\Delta$, maggiore sarà l'entità del drift.
L'individuazione del drift basata sulla valutazione del tasso di errore, però, non può valutare in maniera diretta la gravità del drift; in questi casi, il grado di diminuzione della learning accuracy ce ne può dare una stima: se la learning accuracy è dinimuita in maniera significativa dopo la comparsa del drift, allora si può dire che i nuovi dati sono di tipo differente rispetto ai vecchi (il drift è "netto").
I data distribution-based drift detection methods, invece, sono in grado di quantificare in maniera diretta l'entità del drift in quanto la misura usata per confrontare due data samples riflette direttamente le differenze tra i dati. L'individuazione dell'entità del drift è importante perché serve per individuare qual è la migliore strategia da utilizzare nella fase di drift adaptation.

#### Dove avviene il drift

Vengono identificate le regioni tra un tipo di dato e un altro. Queste regioni vengono individuate nello spazio $X$ nel quale $P_t(X, y)$ e $P_{t+1}(X, y)$ hanno differenze significative. Le tecniche utilizzate per identificare queste regioni dipendono strettamente dal modello utilizzato per identificare il drift.

### 1.1.3 Fase III: Drift Adaptation

In questa fase vengono aggiornati i modelli predittivi esistenti secondo il drift individuato. Si possono identificare tre gruppi di metodologie di drift adaptation: *simple retraining, ensemble retraining* e *model adjusting.*

#### Simple retraining: strategia per il global drift

Il modo più diretto per reagire al concept drift è quello di addestrare un nuovo modello sulla base dei dati identificati come drift dal classificatore. In questo caso, è importante individuare in maniera precisa il momento da cui è necessario l'utilizzo di un nuovo modello.

#### Ensemble retraining: strategia per il recurring drift

Nei casi di recurring drift, una strategia molto utile è quella di mantenere i vecchi modelli e riutilizzarli. I metodi ensemble comprendono una serie di classificatori base con diversi parametri; l'output di ognuno di questi viene combinato usando determinate regole per prevedere i nuovi dati in arrivo.

#### Model adjusting: strategia per il regional drift

Un alternativa al creare un nuovo modello ogni volta che avviene il drift è quella di creare un unico modello che si adatta a ciò che impara dai nuovi

dati e dalla loro diversità rispetto a quelli vecchi. In pratica, questi modelli sono in grado di modificarsi in maniera automatica quando viene identificato un cambio di pattern nei dati. Come si può intuire, questo approccio è sicuramente più efficiente rispetto a quello di addestrare continuamente nuovi modelli.

# Capitolo 2

# Un nuovo approccio per il Class-Based Concept Drift

In alcune applicazioni, collezionare dataset contenenti sample in grado di rappresentare tutte le classi esistenti può essere molto difficile o addirittura impossibile. Dunque, quando andiamo a lavorare con grandi moli di dati, è possibile incontrare dei sample appartenenti a classi non analizzate in fase di addestramento del modello; questo fenomeno è detto *class-based concept drift*.

Una soluzione ovvia, ma sicuramente molto costosa in termini computazionali, sarebbe quella di riaddestrare il modello in maniera molto frequente. Questo approccio, però, richiede l'utilizzo costante di esperti per tradurre i nuovi dati analizzati in appropriate scelte del classificatore, azione che può richiedere molto tempo; come si può immaginare, questo in molti contesti non è attuabile.

In [1] viene presentato un processo unsupervised in grado di individuare in maniera automatica il class-based concept drift e di gestire il riaddestramento del modello valutando la degradazione della predizione ottenuta per i nuovi

dati. Detto in altri termini, si va ad identificare quando i nuovi dati potrebbero appartenere ad un tipo non identificato nella fase di addestramento; in questi casi, le nuove classi vengono automaticamente aggiunte e viene addestrato un nuovo modello, in grado di classificare in maniera corretta anche i nuovi dati.
La metodologia proposta gode delle seguenti proprietà:

1. *Automatic detection*: ovvero, self-evaluation;

2. *Absence of the ground-truth labels*: la soluzione è basata su una stima unsupervised, cosa che permette sia l'avviamento automatico del processo di riaddestramento del modello predittivo, sia l'immediata descrizione dei cambiamenti nella distribuzione delle etichette di classe che hanno motivato l'update del modello (permettendone così una facile lettura);

3. *General-purpose approach*: l'approccio non è fatto ad hoc per un determinato dominio di applicazione o per un determinato tipo di dati;

4. *Scalability*: l'algoritmo è stato disegnato al fine di essere orizzontalmente scalabile per i vari contesti Big Data e applicabile in ambienti in tempo reale; è stato implementato su Apache Spark.

## 2.1 Gestione automatizzata del concept drift

La metodologia proposta si propone di identificare quando sono necessarie delle nuove etichette di classe, senza richiedere delle etichette ground-truth (ovvero basate su dati empirici veri, raccolti tramite osservazione diretta) per i nuovi dati e utilizzando un approccio scalabile. Questo viene fatto essenzialmente in tre step, come mostrato in figura 2.3:

1. **Self-evaluation della degradazione del modello:** questa fase è eseguita da un nuovo approccio unsupervised che valuta la degradazione del modello predittivo nel tempo.

2. **Semi-supervised data labeling:** assegnazione delle etichette alle nuove classi individuate in maniera automatizzata; un piccolo sottoinsieme di sample significativi di ogni nuova classe viene manualmente ispezionata dagli esperti.

3. **KDD (Knowledge Discovery Process) automatizzato per addestrare il nuovo modello:** viene creato un nuovo modello in grado di classificare correttamente i nuovi dati.

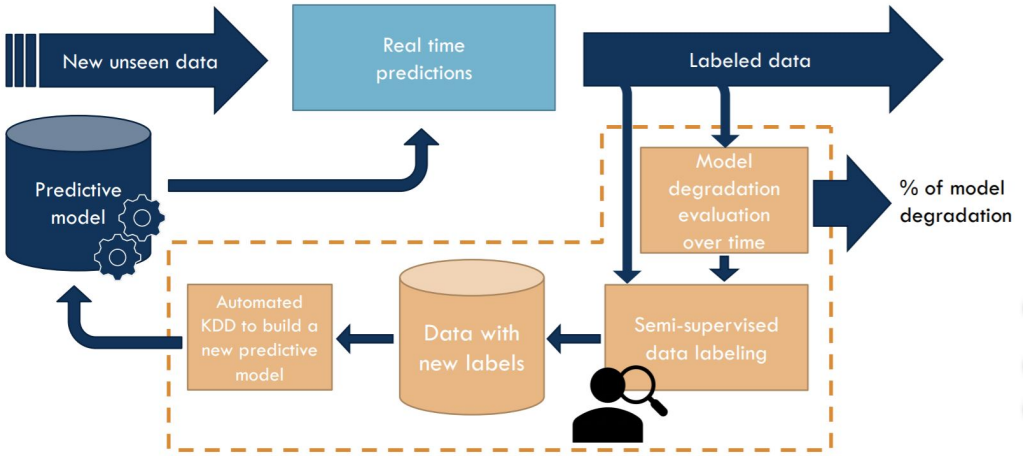



Figura 2.1: Building blocks dell'approccio proposto, [1]

Il contributo di [1] si focalizza sullo step 1, al fine di dare una soluzione che permetta al modello di valutare la sua degradazione nel tempo, con lo scopo di identificare concept drift.

## 2.2 Self-evaluation della degradazione del modello

I modelli predittivi si basano sulle informazioni ricavate dai dati del training set; per questo motivo, prevedere la classe di dati che dascrivono fenomeni in evoluzione può diventare difficile. Per catturare in maniera efficente questa degradazione del modello nel tempo, in [1] vengono effettuati:

(i) *Baseline computation*, calcolando metriche di qualità insupervised sul training set;

(ii) *Self-evaluation*, ricalcolando in maniera periodica la stessa metrica sui nuovi dati e comparando i risultati ottenuti con quelli di (i).

Mentre i nuovi dati vengono continuamente classificati dal modello nel tempo, la qualità della valutazione deve essere stimata in maniera periodica; dato che quest'ultima dipende dalla cardinalità dei nuovi dati e dalle loro etichette, in questo approccio viene innescata la self-evaluation quando una qualsiasi etichetta è affetta dall'accrescimento di una certa percentuale (ad esempio, almeno il 20% di dati in più rispetto a prima) di dati nuovi.
Infine, per automatizzare in maniera completa il processo, si va a definire un valore soglia (definito, in base ai vari casi, dagli esperti) per la degradazione, superato il quale si va a riaddestrare il modello.

**La Silhouette: un indice per la valutazione automatica della qualità del modello nel tempo**

A causa dell'assenza nella fase di training di tutte le possibili classi, non è possibile valutare l'accuratezza del modello predittivo utilizzando i metodi tradizionali (f-measure, precision, recall, ...); viene quindi introdotto un nuovo indice unsupervised in grado di quantificare il grado di coesione tra elementi della stessa classe e il grado di separazione tra elementi di classi diverse. La degradazione del modello viene quindi definita come un cambio in negativo nel valore dell'indice dei nuovi dati classificati rispetto ai valori ottenuti nel training set.

Bisogna sottolineare che l'approccio proposto è valido per ogni scelta dell'indice che valuta la qualità del modello. In [1] viene utilizzata la *Silhouette*, indice che misura quanto simile è un sample ai dati della classe a cui è stato assegnato (coesione), rispetto a quanto lo è dei dati delle altre classi (separazione). I valori della Silhouette variano da $-1$ a $+1$, dove un valore tendente a $+1$ sta ad indicare una assegnazione corretta del sample alla sua classe di appartenenza; in altre parole, più il valore dell'indice è alto, più la similitudine di un sample rispetto agli elementi della classe a cui è stato assegnato è maggiore della similitudine di questo con gli elementi delle altre classi.

Per il calcolo di questo indice devono dunque essere calcolate le distanze di ogni sample con ognuno dei dati presenti nel dataset; questo però richiede un costo computazionale elevato e quindi rende l'approccio non applicabile a contesti con grandi moli di dati (Big Data) o contesti in cui abbiamo flussi di dati costanti (il costo computazionale è $O(N^2)$, dove $N$ è la cardinalità del dataset). Per questo motivo, è conveniente utilizzare la *Descriptor Silhouette*, già proposta in [8].

L'indice *Descriptor Silhouette* è basato sull'idea che la forma geometrica di

un gruppo di punti può essere descritta attraverso un numero limitato di descrittori, ben distribuiti nello spazio; per ogni etichetta, vengono quindi calcolati i vari descrittori sfruttando i centroidi estratti dall'implementazione dell'algoritmo K-Means. In questo modo, vengono ridotti drasticamente i calcoli, in quanto non si andrà a calcolare la distanza di un sample da tutti i punti del dataset, ma si andrà a valutare la distanza di questo dai vari descrittori (la complessità computazionale si riduce a $O(N \cdot C \cdot D)$, dove $C$ è il numero delle classi e $D$ è il numero dei descrittori calcolati per ogni classe). Il fatto che il numero dei descrittori sia limitato è dovuto al fatto che gli algoritmi di clustering (tra cui il K-Means) sono in grado di spiegare i dataset attraverso un numero relativamente piccolo di centroidi. In [1] e in [8], per misurare la distanza tra i dati è stata sfruttata la distanza euclidea. Questa metodologia, però, può essere sviluppata utilizzando anche metriche diverse. Riportiamo qui di seguito la costruzione della silhouette illustrata in [10]. Nel caso di cluster ben separati e di distanze normalizzate, per ogni elemento $i$ del dataset, denotiamo con $A$ il cluster a cui è stato assegnato. Chiamiamo $a(i)$ la diversità media di $i$ da tutti gli altri oggetti di $A$; per ogni cluster $C$ diverso da $A$ calcoliamo la diversità media di $i$ da ogni elemento di $C$ $d(i, C)$ e definiamo

$$b(i) = min_{C \neq A} d(i, C)$$

Il cluster per cui si ottiene il minimo viene denotato con $B$ ed è detto *vicino* dell'elemento $i$ (e si pone $d(i, B) = b(i)$). Il cluster B può essere interpretato come la "seconda miglior scelta" per $i$. Per comprendere meglio quanto detto finora si faccia riferimento all'immagine 2.2; qui la diversità media si può vedere come la lunghezza media dei link tra $i$ e tutti gli elementi dei vari cluster.

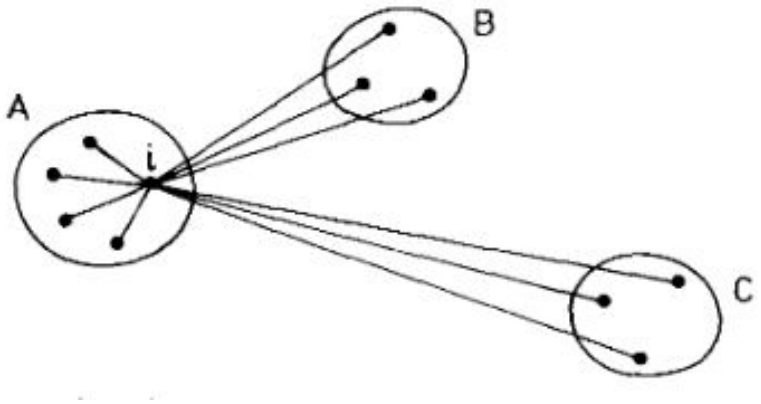



Figura 2.2: Esempio degli elementi coinvolti nel calcolo della silhouette, [10]

Il valore della silhouette, infine, si ottiene combinando $a(i)$ e $b(i)$ nel modo seguente:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

Da questa formula risulta chiaro che, necessariamente, $-1 \leq s(i) \leq +1$. Inoltre, si può capire perché un risultato che tende a $+1$ indica che il dato in esame è stato ben classificato: mettiamoci in un caso limite in cui $a(i)$ è molto minore di $b(i)$, ovvero in un caso in cui la distanza media tra $i$ e gli elementi del cluster $A$ è molto minore della distanza media tra $i$ e gli elementi del cluster $B$; il numeratore sarà dunque circa uguale a $b(i)$, che è il valore che assumerà il denominatore e il loro rapporto tenderà ad uno. Il risultato di questo calcolo, dunque, dà informazioni riguardo la qualità dell'assegnazione di un record ad una certa classe; infatti, se il valore $s(i)$ è minore di 0, la classe a cui è stato assegnato il sample in questione è scorretta; viceversa, se il valore è positivo, l'assegnazione risulta corretta.

Il motivo per cui questo indice è utile per individuare il *class-based concept drift* è semplice: supponiamo di aver addestrato il modello su due classi (e quindi abbiamo definito la presenza di solo 2 cluster); all'arrivo dei nuovi dati, non appartenenti alle due classi già definite, gli algoritmi di clustering tenderanno a incorporare il nuovo cluster nei due cluster esistenti, al fine di ridurre il numero totale di cluster a 2. La silhouette è in grado evidenziare questo problema, in quanto l'azione degli algoritmi di clustering porta ad un aumento della distanza media all'interno di $A$ (e di conseguenza aumenteà il valore di $a(i)$). Più grande è $a(i)$, più il valore di $s(i)$ tenderà ad essere piccolo, andando ad evidenziare un errore di classificazione. Infine, tutti i valori $s(i)$ trovati vengono ordinati in maniera crescente e viene creata la *curva di silhouette*. Questo procedimento va svolto sia sui dati di training (per creare la *Base Silhouette*), sia in fase di test (*Degraded Silhouette*): la qualità delle assegnazioni viene valutata andando a confrontare le due curve.

**Stima della degradazione del modello**

Per la stima della degradazione del modello, per ogni classe, vengono messi a confronto i valori dell'indice calcolati inizialmente solo sui dati di training e poi utilizzando anche i nuovi dati. Vengono così plottate le varie curve per ogni step, in cui i vari punti sono stati ordinati in maniera crescente. Uno shift verso l'alto della nuova curva rispetto a quella vecchia rappresenta una miglioria del modello: i dati nelle varie classi sono più coesi nel caso di dati appartenenti alla stessa classe e più separati nel caso di dati di classi diverse; uno shift verso il basso, invece, rappresenta una degradazione del modello.

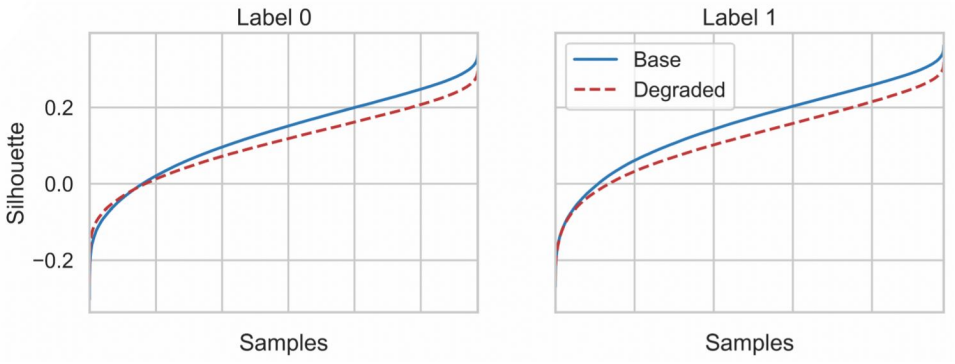



Figura 2.3: Silhouette che rappresenta un improvement in termini di intra-class cohesion e inter-class separation, [1]

Al fine di quantificare lo shift della curva di Silhouette, viene utilizzato il MAAPE (Mean Arc-tangent Absolute Percentage Error).
Dato un modello predittivo addestrato su un set di C classi al tempo $t_0$, il cui training set ha un valore di Silhouette pari a $Sil_{t_0}$, la degradazione della classe $c \in C$ al tempo $t$ è data da:

$$DEG(c,t) = \alpha \cdot MAAPE(Sil_{t_0}, Sil_t) \cdot \frac{N_c}{N}$$

dove $N_c$ è il numero di nuovi record assegnati alla classe $c$, $N$ il numero totale di nuovi sample (troncato a $N_{train}$ se $N > N_{train}$, al fine di poter fare una comparazione sensata), $Sil_t$ è calcolata su una finestra contente gli ultimi $N$ dati (Con $N$ limitato a $N_{train}$) e $\alpha$ così definita:

$$\alpha = \begin{cases} 1 & \overline{Sil_{t_0}} \geq \overline{Sil_t} \\ -1 & \overline{Sil_{t_0}} < \overline{Sil_t} \end{cases}$$

Il coefficiente $\alpha$ definisce se la degradazione è positiva, ovvero se c'è una possibile riduzione nella performance del classificatore, o negativa, quando

c'è una migliore coesione tra elementi della stessa classe e una maggior separazione tra elementi di classi distinte.
Infine, la degradazione dell'intero modello al tempo $t$ è calcolata sommando il valore di degradazione dato da ogni classe, ovvero:

$$DEG_t = \sum_{c \in C} DEG(c, t)$$

L'approccio in esame avvia in maniera automatica una *self-evaluation* quando almeno una delle classi raggiunge un valore di nuovi dati ($N_C$), in percentuale rispetto ai dati precedenti, superiore ad un determinato valore soglia (ad esempio, il 20%); questo permette di evitare ritardi nell'individuazione del concept drift.
Infine, viene eseguito un riaddestramento del modello quando la degradazione totale (o la degradazione di almeno una classe) supera un certo threshold (ad esempio, il 10-15% per la degradazione totale e il 5-10% nel caso di singola classe).

# Capitolo 3

# Casi di studio

Nella pratica, per attuare quanto descritto finora, l'approccio adottato in [1] si divide essenzialmente in tre fasi:

1. Creazione delle finestre temporali: viene definita la percentuale dei dati da usare come train set (nei casi in esame di seguito, si è scelto di prendere il 60% dei dati di due classi); a questo punto, si vanno a creare le finestre temporali su cui testare la validità del modello. Si fanno vari test, prendendo diverse percentuali dei dati rimanenti: si inizia con il 10%, per poi passare al 20% e così via, fino ad arrivare al 100% dei dati rimanenti. Ovvero:

   Test_window_size : [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

   Queste finestre temporali contengono sia parte dei dati delle due classi già presenti in fase di training, sia dati di una classe sconosciuta. Ipotizzando di chiamare 0 e 1 le due classi conosciute e 2 la classe sconosciuta, si passeranno percentuali diverse per ogni classe in ogni finestra, nella maniera seguente:

| | | | |
|---|---|---|---|
| Test 1 | 0: 0.5 | 1: 0.5 | 2. 0.0 |
| Test 2 | 0: 0.45 | 1: 0.45 | 2: 0.1 |
| Test 3 | 0: 0.4 | 1: 0.4 | 2: 0.2 |
| Test 4 | 0: 0.35 | 1: 0.35 | 2: 0.3 |
| Test 5 | 0: 0.3 | 1: 0.3 | 2: 0.4 |
| Test 6 | 0: 0.25 | 1: 0.25 | 2: 0.5 |
| Test 7 | 0: 0.2 | 1: 0.2 | 2: 0.6 |
| Test 8 | 0: 0.15 | 1: 0.15 | 2: 0.7 |
| Test 9 | 0: 0.1 | 1: 0.1 | 2: 0.8 |
| Test 10 | 0: 0.05 | 1: 0.05 | 2: 0.9 |
| Test 11 | 0: 0.0 | 1: 0.0 | 2: 1.0 |

Tabella 3.1: Distribuzione dei dati nelle finestre temporali

2. Creazione del modello di classificazione: si addestrano vari classificatori con i dati del training set, andando poi a scegliere quello con le performance migliori.

3. Valutazione del degrado del modello: è la fase in cui entra in gioco la silhouette. Per ogni classe e per ogni finestra temporale, si va a valutare quanto il modello è in grado di classificare in maniera corretta i dati in entrata. Nel tempo, si va ad inserire sempre più percentuale di dato di una classe sconosciuta; per questo motivo, ci si aspetta un degrado crescente (ovvero: il degrado rilevato nel test uno della prima finestra sarà minore del degrado nel test 11 della stessa finestra temporale).

Riportiamo ora i risultati più significativi ottenuti dall'analisi, dando una descrizione generale dei vari dataset e spiegando nel dettaglio i risultati ottenuti. Tutti i dataset in analisi, ove necessario, sono stati normalizzati.

Bisogna sottolineare che questa è solo una prima analisi, a cui dovranno seguirne altre più dettagliate, dove si andrà a rendere più generale possibile l'approccio; questo sarà possibile, ad esempio, andando a cambiare le metriche utilizzate nel calcolo della silhouette o cambiando l'indice in questione. Tutti i test sono stati svolti in un cluster node con CPU 8-core e 40 GB di RAM.

## 3.1 KDDCup99 Dataset

Il primo dataset analizzato è un dataset reale, scaricabile dal sito web UCI: Machine Learning Repository.

Come spiegato in [9], il quale ne presenta un'analisi dettagliata, KDDCup99 è uno dei dataset più usati per la valutazione dei metodi di rilevazioni di anomalie dei dati. È una raccolta effettuata da *Stolfo et al.* prendendo i dati acquisiti nella valutazione IDS di DARPA '98, che ha analizzato il traffico di rete e i log di controllo di una rete di simulazione e ha identificato sessioni di attacco (e il loro tipo) tra le varie attività ritenute normali.

Il caso in analisi è una versione contenente solamente il 10% dei dati effettivi del dataset, in quanto la mole di dati del dataset completo era veramente grande e il 10% di esso è sufficiente per un'analisi esaustiva; per cui, il caso in esame presenta **494021 sample**, **41 attributi** e **23 classi**; una descrizione accurata degli attributi è riportata in figura 3.1.

Le etichette sono state rimappate a dei valori numerici tramite la classe *LabelEncoder* di *sklearn* su Python.

### Analisi dei risultati ottenuti

Si è scelto di addestrare il modello sulle classi 9 e 18, in quanto sono due delle classi che hanno più sample all'interno del dataset. Gli elementi esterni alle due classi sono sample appartenenti alla classe 11.

Il 60% degli elementi della classe 9 e il 60% degli elementi della classe 18 vengono usati per addestrare il modello; si vanno poi a simulare delle finestre temporali introducendo percentuali diverse dei rimanenti dati, includendo in maniera graduale i dati della classe 11.

Come primo risultato illustriamo la matrice dei degradi: questa matrice riassume le percentuali di degrado totale (ovvero: la somma delle percentuali

| Nr | Features | |
|---|---|---|
| | Name | Description |
| 1 | duration | duration of connection in seconds |
| 2 | protocol_type | connection protocol (tcp, udp, icmp) |
| 3 | service | dst port mapped to service (e.g. http, ftp, ..) |
| 4 | flag | normal or error status flag of connection |
| 5 | src_bytes | number of data bytes from src to dst |
| 6 | dst_bytes | bytes from dst to src |
| 7 | land | 1 if connection is from/to the same host/port; else 0 |
| 8 | wrong_fragment | number of 'wrong' fragments (values 0,1,3) |
| 9 | urgent | number of urgent packets |
| 10 | hot | number of 'hot' indicators (bro-ids feature) |
| 11 | num_failed_logins | number of failed login attempts |
| 12 | logged_in | 1 if successfully logged in; else 0 |
| 13 | num_compromised | number of 'compromised' conditions |
| 14 | root_shell | 1 if root shell is obtained; else 0 |
| 15 | su_attempted | 1 if 'su root' command attempted; else 0 |
| 16 | num_root | number of 'root' accesses |
| 17 | num_file_creations | number of file creation operations |
| 18 | num_shells | number of shell prompts |
| 19 | num_access_files | number of operations on access control files |
| 20 | num_outbound_cmds | number of outbound commands in an ftp session |
| 21 | is_hot_login | 1 if login belongs to 'hot' list (e.g. root, adm); else 0 |
| 22 | is_guest_login | 1 if login is 'guest' login (e.g. guest, anonymous); else 0 |
| 23 | count | number of connections to same host as current connection in past two seconds |
| 24 | srv_count | number of connections to same service as current connection in past two seconds |
| 25 | serror_rate | % of connections that have 'SYN' errors |
| 26 | srv_serror_rate | % of connections that have 'SYN' errors |
| 27 | rerror_rate | % of connections that have 'REJ' errors |
| 28 | srv_rerror_rate | % of connections that have 'REJ' errors |
| 29 | same_srv_rate | % of connections to the same service |
| 30 | diff_srv_rate | % of connections to different services |
| 31 | srv_diff_host_rate | % of connections to different hosts |
| 32 | dst_host_count | count of connections having same dst host |
| 33 | dst_host_srv_count | count of connections having same dst host and using same service |
| 34 | dst_host_same_srv_rate | % of connections having same dst port and using same service |
| 35 | dst_host_diff_srv_rate | % of different services on current host |
| 36 | dst_host_same_src_port_rate | % of connections to current host having same src port |
| 37 | dst_host_srv_diff_host_rate | % of connections to same service coming from diff. hosts |
| 38 | dst_host_serror_rate | % of connections to current host that have an S0 error |
| 39 | dst_host_srv_serror_rate | % of connections to current host and specified service that have an S0 error |
| 40 | dst_host_rerror_rate | % of connections to current host that have an RST error |
| 41 | dst_host_srv_rerror_rate | % of connections to the current host and specified service that have an RST error |
| 42 | connection_type | |

Figura 3.1: Descrizione delle feature del dataset KDDCup99, [11]

di degrado di entrambe le classi) su ogni test fatto, per ogni finestra. La matrice va letta in tre modi:

- da sinistra verso destra: dato che aumenta la percentuale di dati nuovi inseriti, ci si aspetta un degrado crescente;
- dall'alto verso il basso: in questo caso aumenta la percentuale di dato esterno alle due classi conosciute dal classificatore e ciò dovrebbe portare ad un aumento del degrado;
- diagonale principale: vengono mediati i due punti precedenti, per cui ci si aspetta un degrado crescente anche in questo caso.

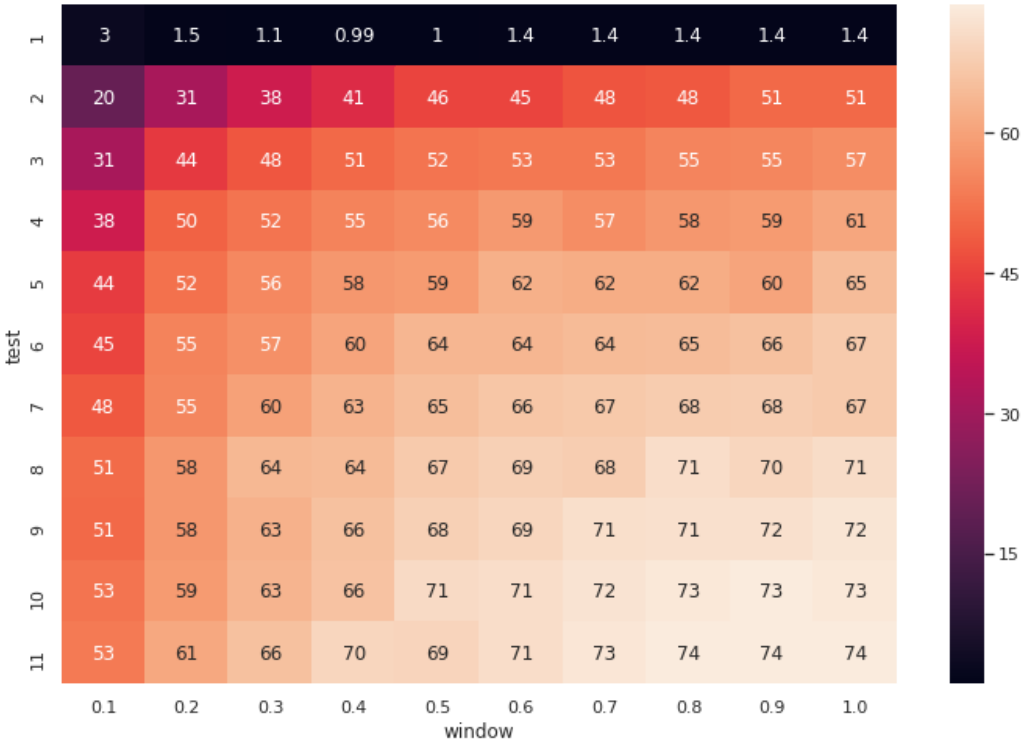


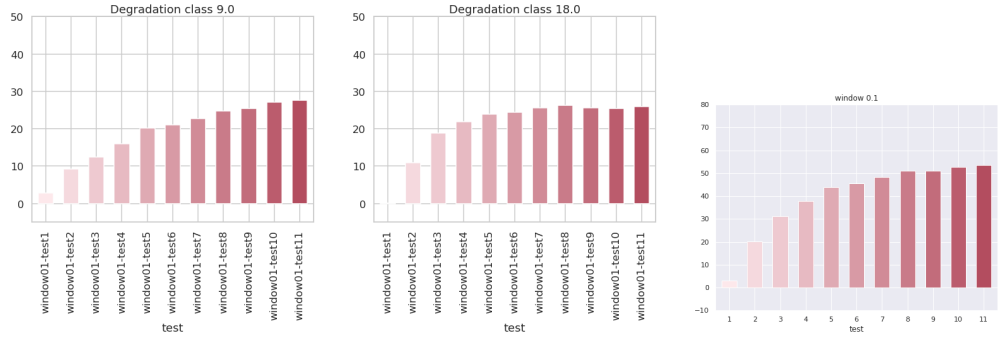

Figura 3.2: KDDCup99 Dataset: matrice dei degradi

Come mostra la legenda a destra nella figura 3.2, più il degrado è alto, più il colore tenderà ad essere chiaro. La prima riga della matrice rappresenta il degrado del modello quando si cerca di classificare solamente dati appartenenti alle due classi su cui è stato addestrato il modello; si nota un trend inizialmente decrescente, che poi cresce ed infine si stabilizza. Questo potrebbe essere dovuto al fatto che i dati del training set non spiegavano completamente il dataset, per cui i primi dati nuovi risultavano distanti dal nucleo centrale del cluster; l'introduzione di una percentuale maggiore di dati ha portato ad una maggiore coesione tra le classi e ad una stabilizzazione della percentuale di degrado. Il degrado in questa prima riga rimane molto basso, indicando una quasi sempre esatta classificazione dei dati da parte del classificatore.

Dalla seconda riga in poi, si va ad introdurre in maniera graduale una percentuale di dato della classe 11. Ciò che salta subito all'occhio è come la percentuale di degrado aumenti in maniera evidente appena si introduce una percentuale minima di dato sconosciuto; la percentuale di degrado, infine, aumenta in maniera graduale sia aumentando la percentuale di dato della classe 11, sia il numero di sample da analizzare. I dati in questo dataset, dunque, sono ben separati in base alle loro classi e permettono un'identificazione quasi immediata di una nuova classe sconosciuta.

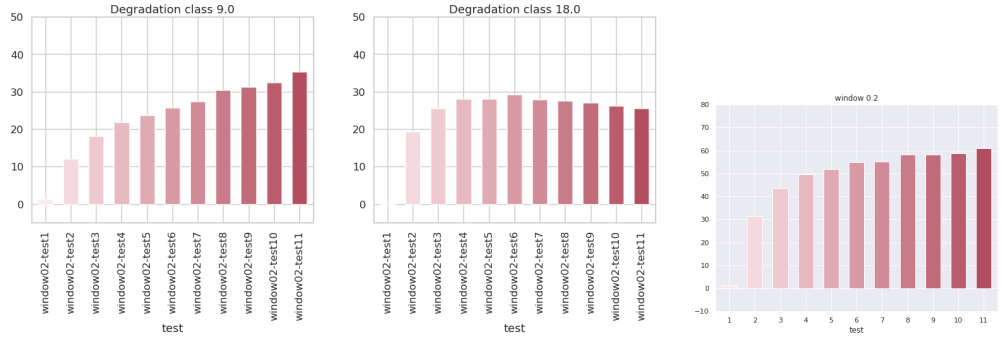
Andando ad analizzare nel dettaglio il problema, la classe 9 presenta un degrado crescente ben evidente. La classe 18, invece, presenta un degrado abbastanza costante, che tende ad essere leggermente decrescente nel tempo. La figura 3.3 rappresenta la percentuale di degrado quando si cerca di classificare una quantità di dati pari al 10% del totale dei dati utilizzati per addestrare il modello; sia per la classe 9, sia per la classe 18, il degrado nel caso in cui non è presente nessun sample della classe 11 è quasi nullo. La

classe 9 ha percentuale di degrado che aumenta in maniera graduale all'aumentare della percentuale di dati della classe 11, mentre la classe 18 figura un degrado che aumenta in maniera più netta, per poi stabilizzarsi dal test 5 (ovvero test in cui si vanno ad analizzare un gruppo di dati composto al 30% da dati di classe 9, 30% di dati di classe 18 e 40% di dati di classe 11) in poi.




*Degrado della classe 9* *Degrado della classe 18* *Degrado totale*

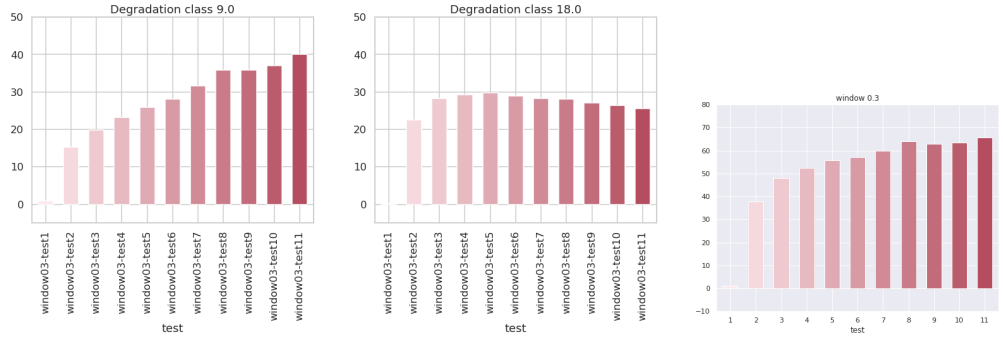
Figura 3.3: KDDCup99 Dataset: Window 0.1, percentuale di degrado




*Degrado della classe 9* *Degrado della classe 18* *Degrado totale*

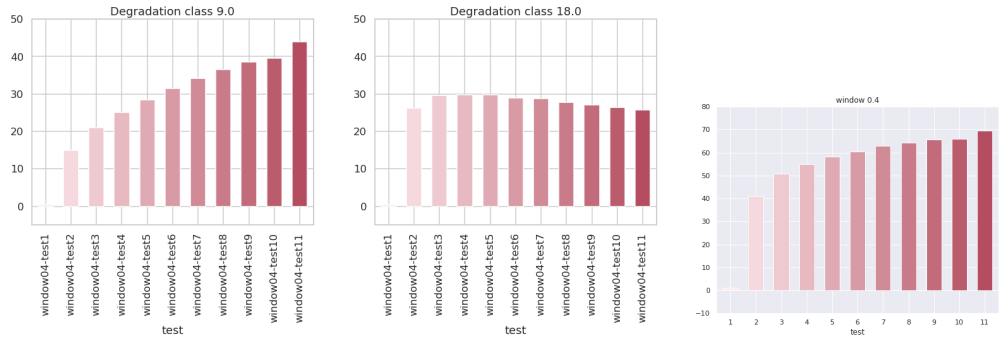
Figura 3.4: KDDCup99 Dataset: Window 0.2, percentuale di degrado

Come possiamo notare, questo trend persiste aumentando il numero di dati passati al classificatore. La classe 9 presenta un degrado che aumenta in maniera graduale in tutte le finestre temporali, così come la classe 18 presenta sempre una percentuale di degrado che aumenta in maniera più netta nei primi test, per poi stabilizzarsi.



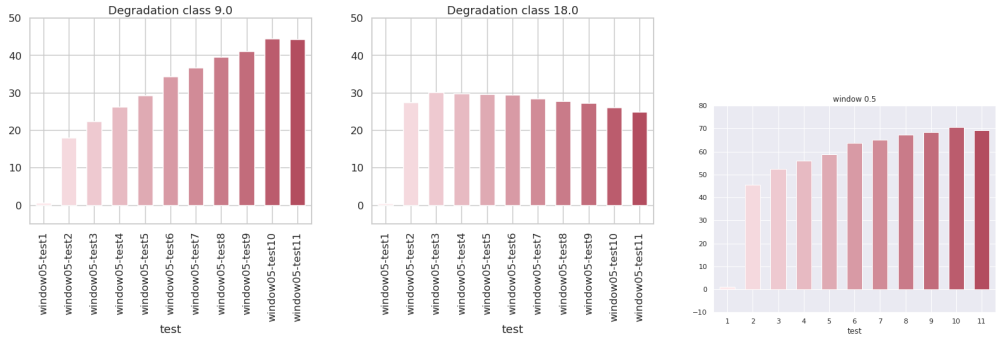

*Degrado della classe 9* *Degrado della classe 18* *Degrado totale*

Figura 3.5: KDDCup99 Dataset: Window 0.3, percentuale di degrado

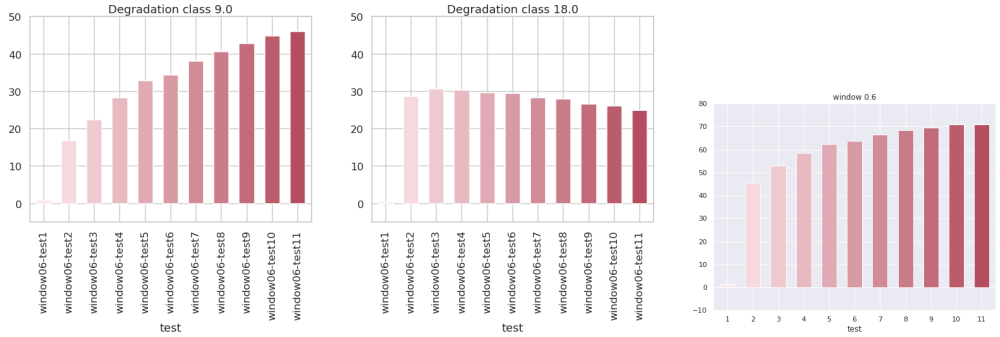



*Degrado della classe 9* *Degrado della classe 18* *Degrado totale*

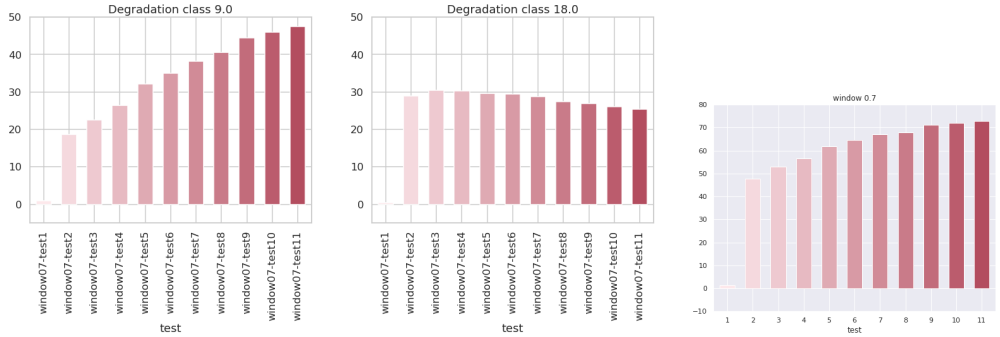
Figura 3.6: KDDCup99 Dataset: Window 0.4, percentuale di degrado




*Degrado della classe 9* *Degrado della classe 18* *Degrado totale*

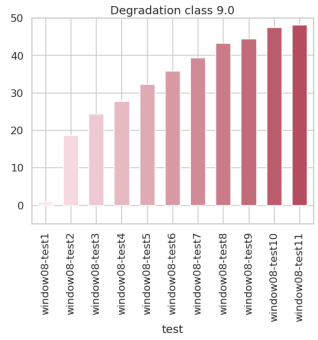



*Degrado della classe 9* *Degrado della classe 18* *Degrado totale*

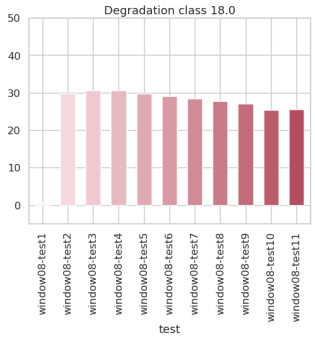



*Degrado della classe 9* *Degrado della classe 18* *Degrado totale*

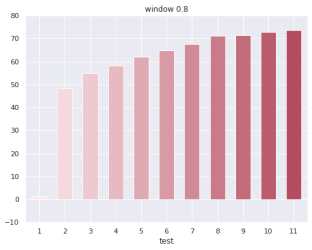
Figura 3.7: KDDCup99 Dataset: Windows 0.5, 0.6, 0.7, percentuale di degrado



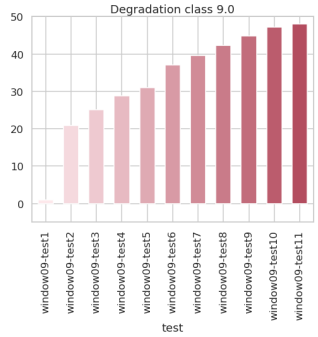

*Degrado della classe 9*



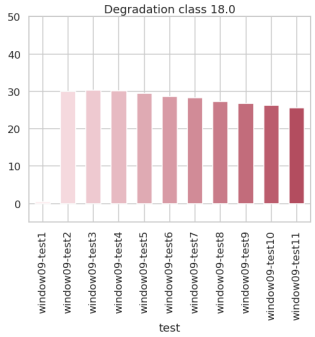

*Degrado della classe 18*



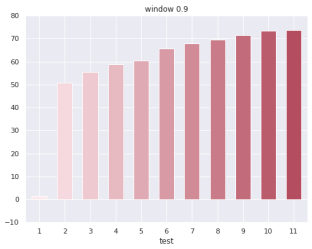

*Degrado totale*



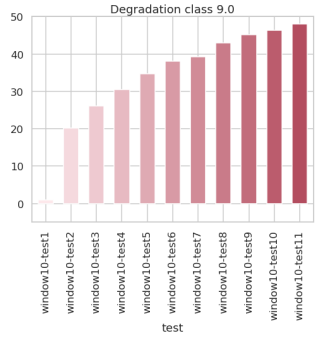

*Degrado della classe 9*



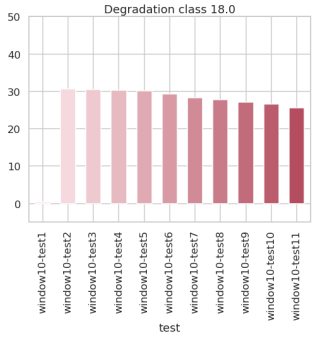

*Degrado della classe 18*



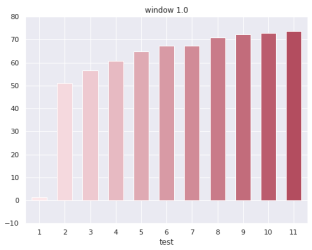

*Degrado totale*




*Degrado della classe 9*



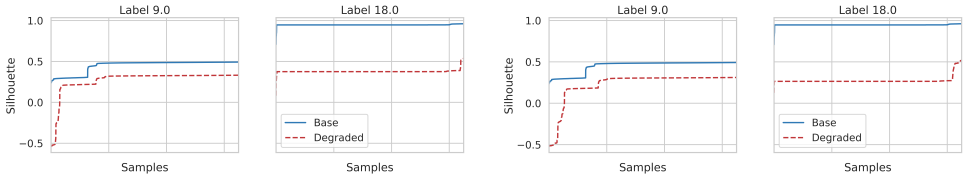

*Degrado della classe 18*




*Degrado totale*

Figura 3.8: KDDCup99 Dataset: Window 0.8, 0.9, 1.0, percentuale di degrado

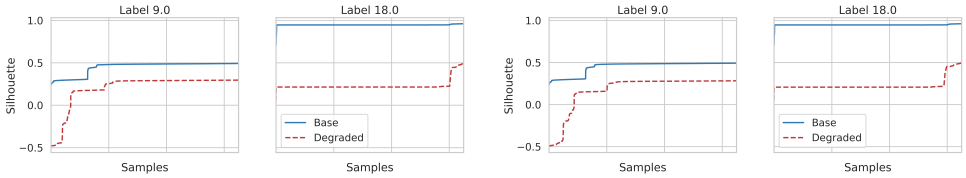
La percentuale totale presenta un trend crescente; i grafici evidenziano inoltre in maniera netta quanto anche una piccola percentuale di dato della classe 11 introdotta faccia aumentare esponenzialmente il degrado totale del modello. Questo indica che i dati della classe non presente in fase di addestramento del classificatore sono ben distanti dai dati delle altre due classi.
Andando ad analizzare nel dettaglio il grado di coesione intra-classe e separazione inter-classi esplicato dalla silhouette, si evince che la classe 18 presentava in fase di training una forte coesione tra i suoi elementi, che spiega la crescita veloce di degrado nei primi test riscontrata delle figure sopra. Infatti, nel momento in cui il classificatore assegna dei valori della classe 11 alla classe 18, la coesione degli elementi di questa diminuisce nettamente, e ciò è ben visibile nei grafici della silhouette: mentre la silhouette base presenta valori prossimi a uno, il grafico della silhouette degraded risulta molto traslato verso il basso rispetto a quello della silhouetta base, che si sposta sempre più verso il basso, andando a risultare prossimo allo zero, man mano che si vanno a valutare sempre più dati.
Per quanto riguarda la classe 9, evidenzia anch'essa un livello di coesione intra-classe minore nel caso della silhouette degraded rispetto a quella base, ma in questo caso il distacco è meno netto.




*Window 0.1 Test 5*



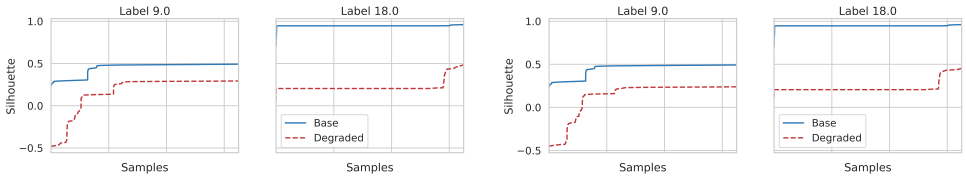

*Window 0.2 Test 5*




*Window 0.3 Test 5*



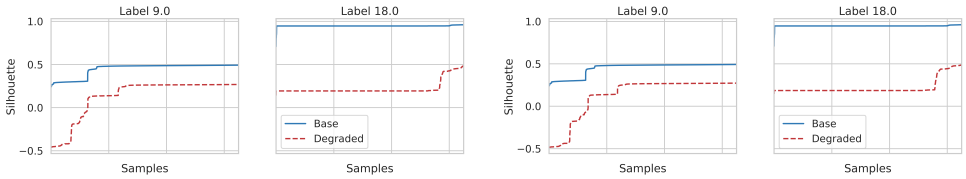

*Window 0.4 Test 5*




*Window 0.5 Test 5*



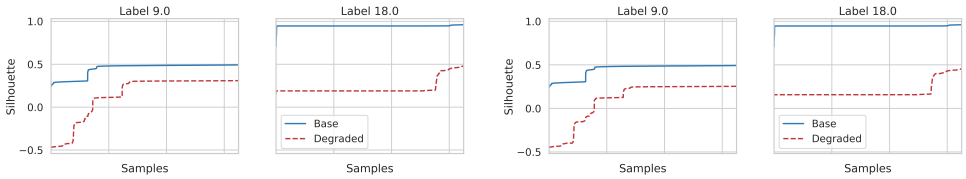

*Window 0.6 Test 5*




*Window 0.7 Test 5*




*Window 0.8 Test 5*




*Window 0.9 Test 5*




*Window 1.0 Test 5*

Figura 3.9: KDDCup99 Dataset: silhouette

## 3.2 Covertype Dataset

Il secondo dataset preso in analisi è un dataset reale, scaricabile dal sito web UCI: Machine Learning Repository. Si tratta di una raccolta di dati riguardanti i tipi di alberi presenti in determinate aree geografiche e include informazioni sulla copertura dell'ombra, la distanza dai punti di riferimento vicini (come, ad esempio, le strade), il tipo di terreno e la topografia locale. Lo studio è stato fatto su quattro aree situate nella Foresta Nazionale di Roosevelt, nel nord del Colorado, descritte in figura 3.10. Sono stati scelti questi luoghi perché le loro foreste presentano disturbi minimi da parte dell'uomo, per cui i forest covertype sono quasi esclusivamente il risultato di processi ecologici. Le quattro aree in esame sono: Rawah, Neota, Comanche Peak e Cache la Poudre, come osservabile in figura 3.10.

Il dataset presenta **5810126 osservazioni**, ha **54 attributi** e **7 classi**:

1. Spruce/Fir
2. Lodgepole Pine
3. Ponderosa Pine
4. Cottonwood/Willow
5. Aspen
6. Douglas-fir
7. Krummholz

Per effettuare l'analisi, il dataset è stato normalizzato. Tutti gli attributi, quindi, hanno valori tra 0 e 1 e sono stati descritti nel dettaglio in tabella 3.2.

| Name | Data Type | Measurement | Description |
|---|---|---|---|
| Elevation | quantitative | meters | Elevation in meters |
| Aspect | quantitative | azimuth | Aspect in degrees azimuth |
| Slope | quantitative | degrees | Slope in degrees |
| Horizontal Distance To Hydrology | quantitative | meters | Horz Dist to nearest surface water features |
| Vertical Distance To Hydrology | quantitative | meters | Vert Dist to nearest surface water features |
| Horizontal Distance To Roadways | quantitative | meters | Horz Dist to nearest roadway |
| Hillshade 9am | quantitative | 0 to 255 index | Hillshade index at 9am, summer solstice |
| Hillshade Noon | quantitative | 0 to 255 index | Hillshade index at noon, summer soltice |
| Hillshade 3pm | quantitative | 0 to 255 index | Hillshade index at 3pm, summer solstice |
| Horizontal Distance To Fire Points | quantitative | meters | Horz Dist to nearest wildfire ignition points |
| Wilderness Area (4 binary columns) | qualitative | 0 (absence) or 1 (presence) | Wilderness area designation |
| Soil Type (40 binary columns) | qualitative | 0 (absence) or 1 (presence) | Soil Type designation |
| Cover Type (7 types) | integer | 1 to 7 | Forest Cover Type designation |

Tabella 3.2: Descrizione degli attributi del dataset

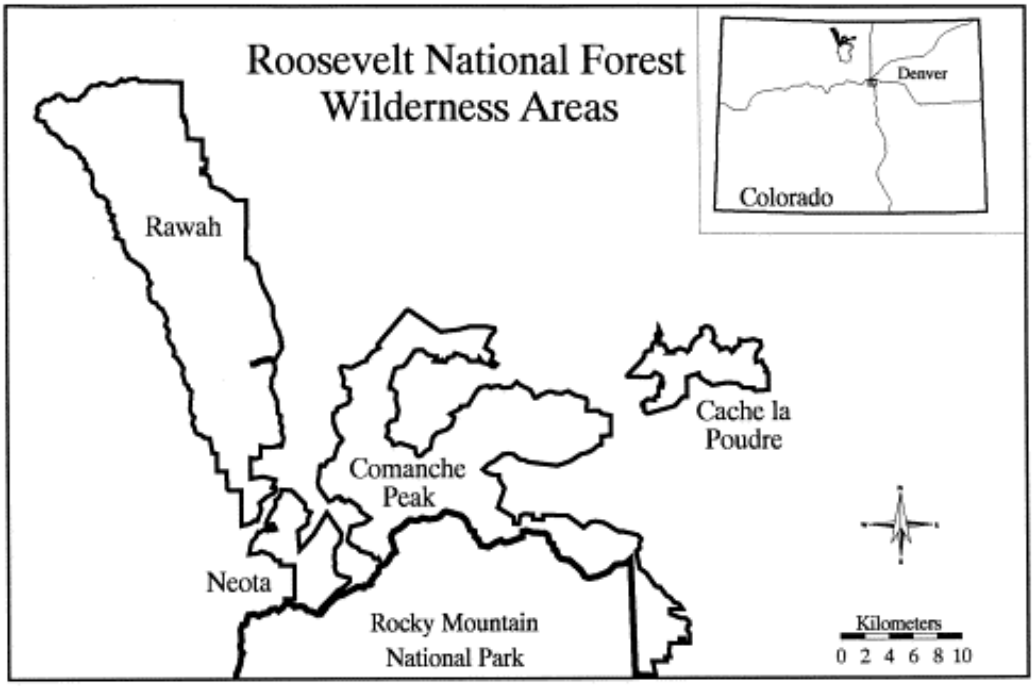



Figura 3.10: Covertype Map, fonte: sciencedirect.com

**Analisi dei risultati ottenuti**

Come per il dataset precedente, si è scelto di addestrare il modello sulle classi 3 e 7, in quanto sono tra le classi che hanno più sample all'interno del dataset; si è poi introdotto come dato della classe "sconosciuta" gli elementi della classe 6, anch'essa una delle classi più popolate del dataset.
Per avere una overview di come l'avvento di una nuova classe influenza il modello di classificazione, viene mostrata la matrice dei degradi totali che riassume i risultati di tutti i test svolti su tutte le finestre temporali.
In tutte e tre le direzioni in cui va letta la matrice viene rispettato un trend crescente del degrado. Una cosa apparentemente anomala in questa matrice è la presenza di una zona più chiara in alto a destra (e non in basso a destra, come ci saremmo aspettati). Questo risultato può essere dovuto a una non

sufficiente coesione dei dati delle due classi nel training set. Approfondiremo questo aspetto quando andremo ad analizzare le curve di silhouette.

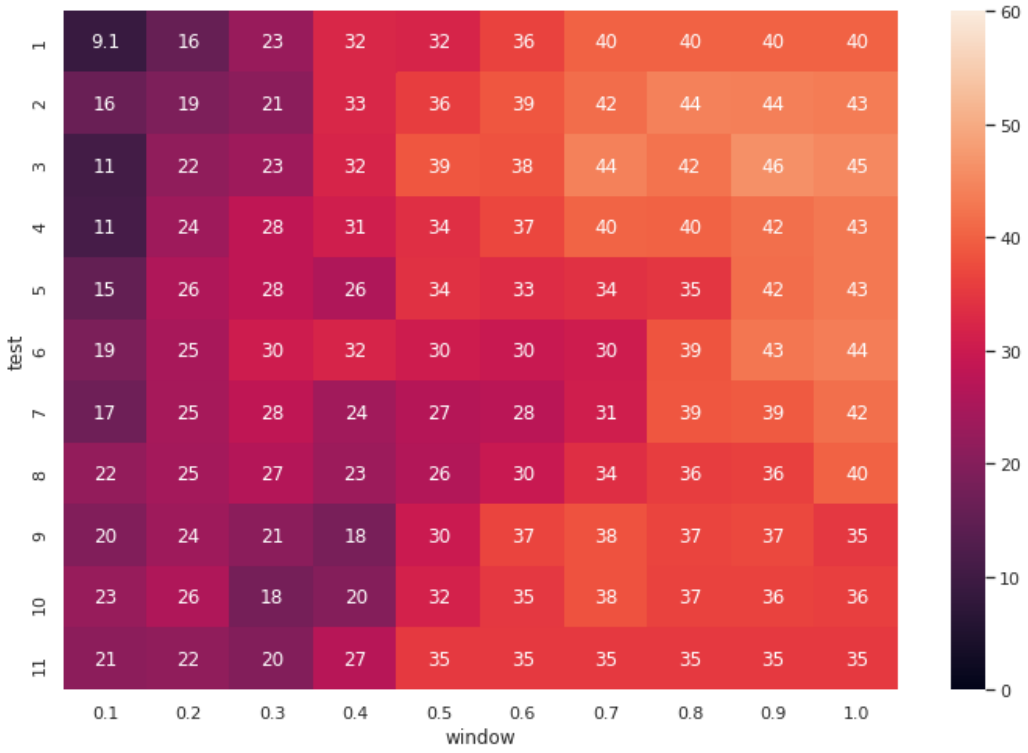
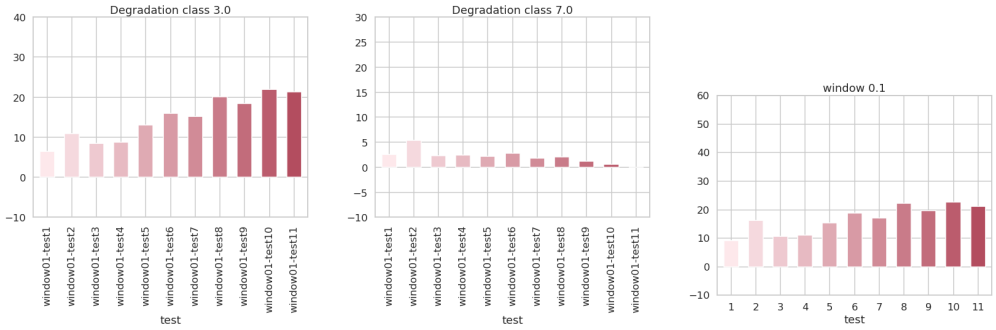



Figura 3.11: Covertype Dataset: matrice dei degradi

Ciò che si evince dai grafici della degradazione del modello è che la classe più interessata da degrado è la 3, mentre la classe 7 tende ad avere un degrado molto basso, che decresce nel tempo. Questo sta ad indicare che il classificatore tende ad assegnare più spesso i dati della classe nuova, non conosciuta, alla classe 3, mentre riesce ad identificare il fatto che gli elementi della classe sconosciuta non appartengono alla classe 7.

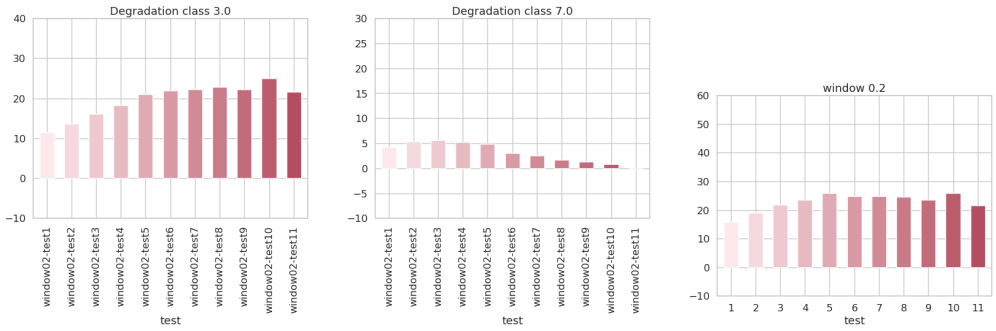
Nella prima finestra, nella quale si analizzano una quantità molto piccola di dati, si evidenzia un degrado quasi nullo nella classe 7 e un degrado crescente nella classe 3.




*Degrado della classe 3* *Degrado della classe 7* *Degrado totale*

Figura 3.12: Covertype Dataset: Window 0.1, percentuali di degrado

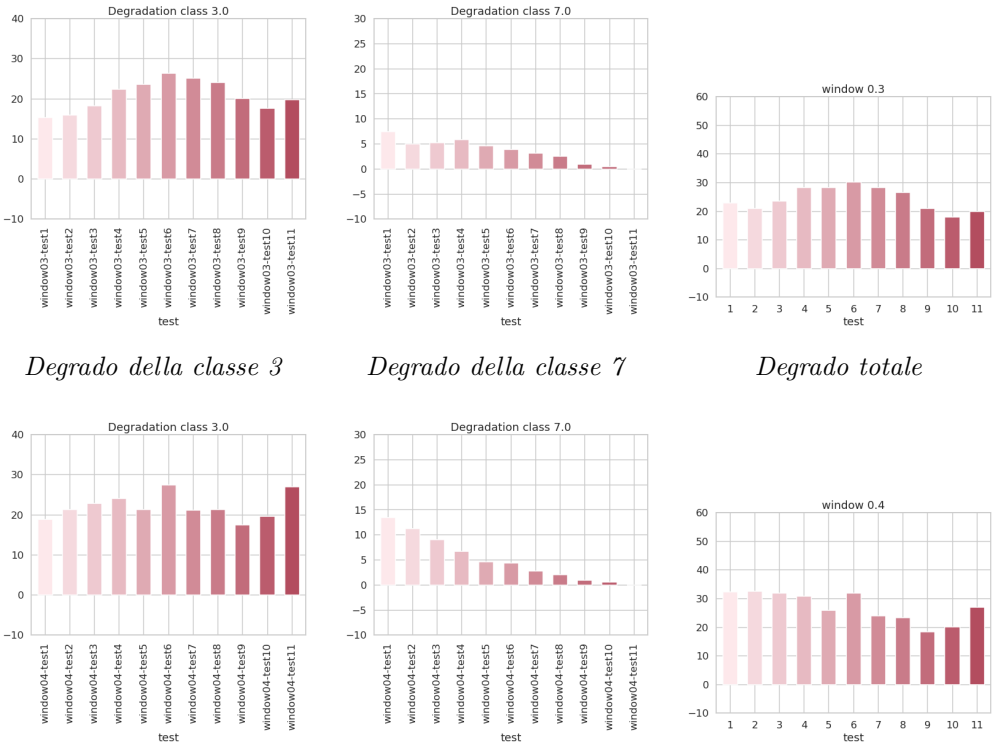
Dalla finestra 0.2 in poi, la classe 7 tende ad avere una percentuale di degrado che decresce nel tempo, diventando nullo nel caso in cui i dati passati al classificatore sono tutti appartenenti alla classe 6.




*Degrado della classe 3* *Degrado della classe 7* *Degrado totale*

Figura 3.13: Covertype Dataset: Window 0.2, percentuali di degrado

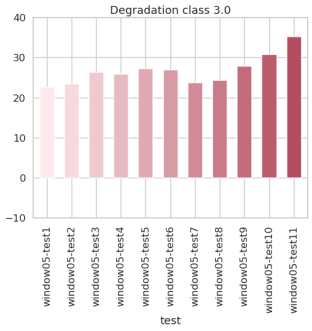
La classe 3, dalla finestra 0.3 in poi, manifesta un degrado che oscilla: si può visualizzare un trend prima crescente e poi decrescente. Per spiegare meglio questo fenomeno è necessario andare a studiare il comportamento dell'indice di silhouette nelle varie finestre e in tre test in particolare: nel test uno, nel quale è completamente assente la classe 6, il test 5 in cui le tre classi sono circa bilanciate (30% di dati della classe 3, 30% di dati della classe 7 e 40% di dati della classe 6) e il test 11, nel quale è presente esclusivamente la classe 6.



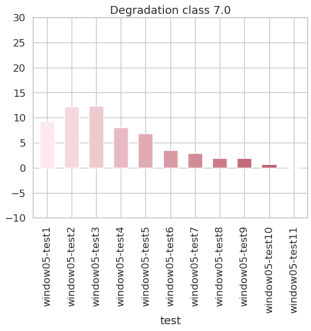

*Degrado della classe 3* *Degrado della classe 7* *Degrado totale*

*Degrado della classe 3* *Degrado della classe 7* *Degrado totale*

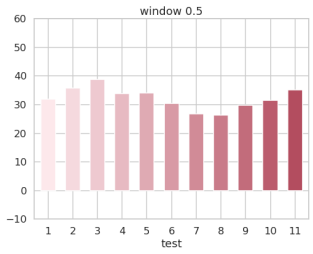
Figura 3.14: Covertype Dataset: Windows 0.3, 0.4, percentuali di degrado



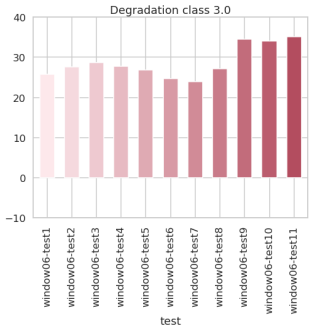

*Degrado della classe 3*



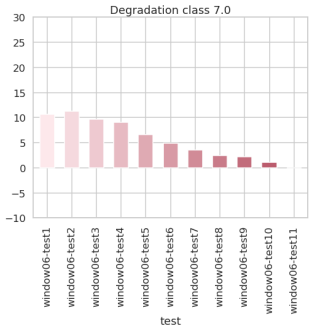

*Degrado della classe 7*



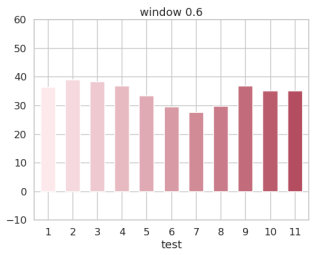

*Degrado totale*




*Degrado della classe 3*

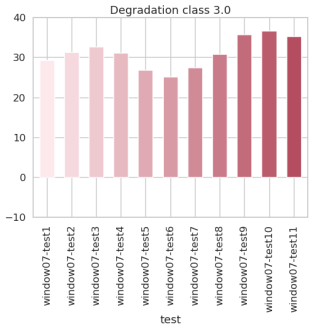



*Degrado della classe 7*

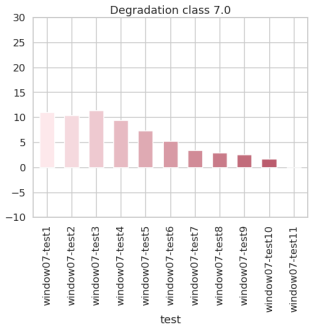



*Degrado totale*

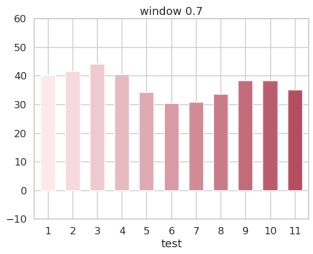
Figura 3.15: Covertype Dataset: Window 0.6, percentuali di degrado




*Degrado della classe 3*

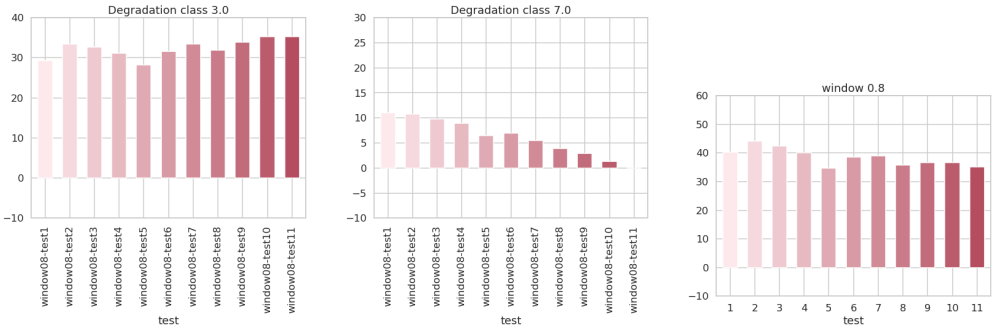



*Degrado della classe 7*

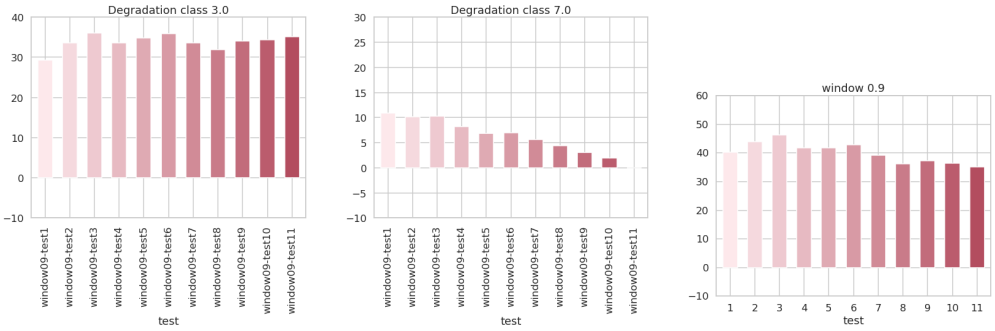



*Degrado totale*

Figura 3.16: Covertype Dataset: Windows 0.5, 0.6, 0.7, percentuali di degrado

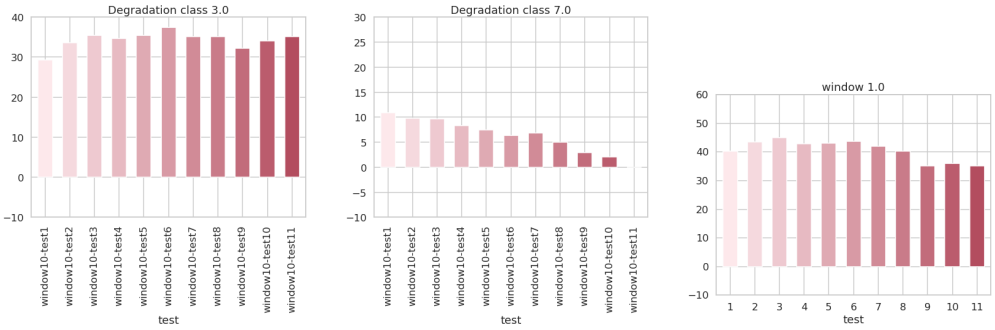



*Degrado della classe 3* *Degrado della classe 7* *Degrado totale*




*Degrado della classe 3* *Degrado della classe 7* *Degrado totale*

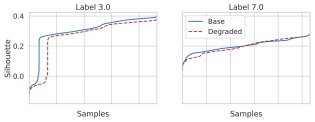
Figura 3.17: Covertype Dataset: Window 0.9, percentuali di degrado



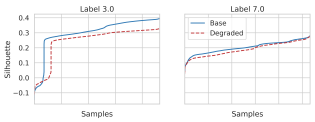

*Degrado della classe 3* *Degrado della classe 7* *Degrado totale*

Figura 3.18: Covertype Dataset: Windows 0.8, 0.9, 1.0, percentuali di degrado

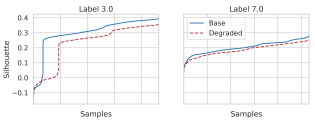
In via generale, una curva di silhouette che indica una miglioria nella coesione degli elementi di una stessa classe e nella separazione di elementi di classi diverse si manifesta in una curva che sta al di sopra della curva di silhouette base. Questo fenomeno non si può presentare quando si vanno ad inserire elementi di una classe sconosciuta al classificatore: infatti, quando questo si troverà a dover assegnare ad una classe uno di questi elementi, lo assegnerà necessariamente a una delle classi conosciute, commettendo un errore e portando la classe in questione a essere meno coesa.
Dal grafico che rappresenta i valori dell'indice di silhouette in figura **??**, possiamo valutare com'è cambiata la coesione tra elementi della stessa classe e la separazione di un elemento con gli elementi delle altre classi.
Ciò che salta subito all'occhio è che, in tutte le finestre, per ciascuna classe, la silhouette base sta sempre sopra a quella calcolata aggiungendo i dati nuovi: ciò sta ad indicare una maggiore separazione tra gli elementi interni ad una stessa classe e una coesione più accentuata tra elementi di classi distinte. Andando ad approfondire, si può notare come la classe 3 sia quella inizialmente più coesa e che maggiormente viene influenzata dalla presenza dei dati della classe 6. La classe 7, invece, presenta una curva di silhouette degraded meno distante da quella base.
La classe 7 presenta, in generale, una curva di silhouette degraded che tende ad avvicinarsi alla curva base: al diminuire del numero di elementi della classe 7 passati, essa risulta essere più coesa. Ciò indica che gli elementi della classe 7 presenti in fase di addestramento del modello non rappresentavano completamente la classe. Inoltre, il classificatore è in grado di riconoscere che gli elementi della classe 6 non appartengono alla classe 7, altrimenti avremmo registrato una percentuale di degrado crescente nel tempo.



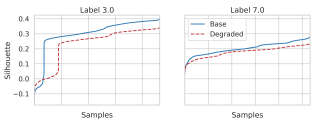

*Window 0.1 Test 1*



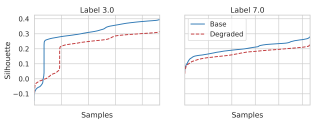

*Window 0.1 Test 5*



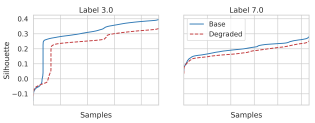

*Window 0.1 Test 11*



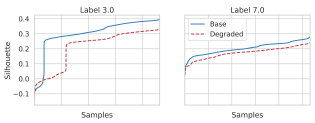

*Window 0.2 Test 1*



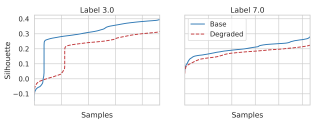

*Window 0.2 Test 5*



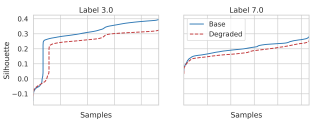

*Window 0.2 Test 11*



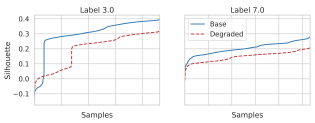

*Window 0.3 Test 1*



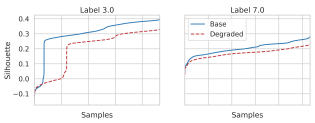

*Window 0.3 Test 5*



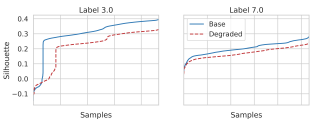

*Window 0.3 Test 11*



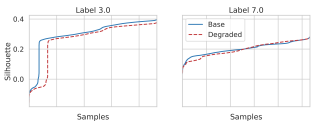

*Window 0.4 Test 1*



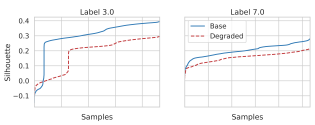

*Window 0.4 Test 5*



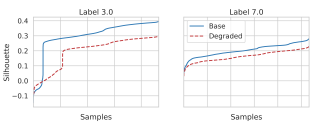

*Window 0.4 Test 11*



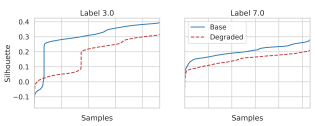

*Window 0.5 Test 1*



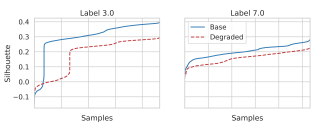

*Window 0.5 Test 5*



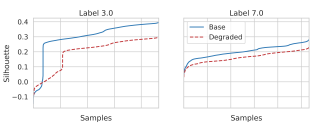

*Window 0.5 Test 11*



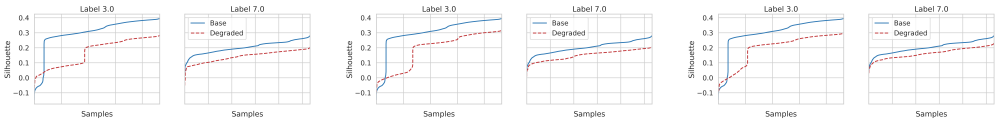

*Window 0.6 Test 1*



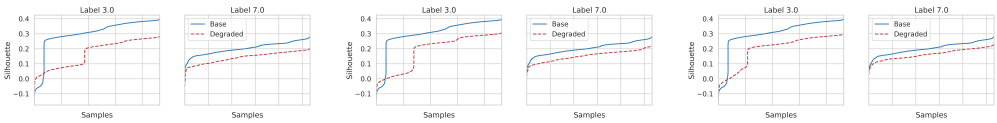

*Window 0.6 Test 5*



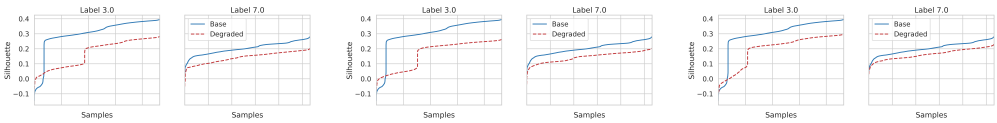

*Window 0.6 Test 11*



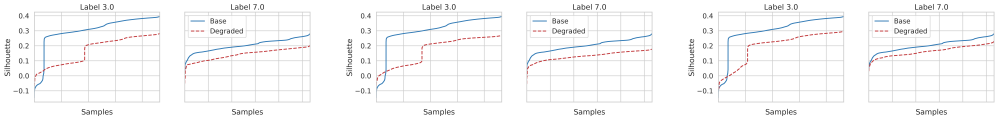

*Window 0.7 Test 1* *Window 0.7 Test 5* *Window 0.7 Test 11*




*Window 0.8 Test 1* *Window 0.8 Test 5* *Window 0.8 Test 11*




*Window 0.9 Test 1* *Window 0.9 Test 5* *Window 0.9 Test 11*




*Window 1.0 Test 1* *Window 1.0 Test 5* *Window 1.0 Test 11*

Figura 3.19: Covertype Dataset: Silhouette

Andando a confrontare le curve di silhouette per la classe 3 nei test 1, 5 e 11, si nota che la curva di silhouette tende ad alzarsi dal test 1 (in cui non è presente dato della classe 6) al test 5, per poi riabbassarsi nel test 11 (in cui vi è solo dato della classe 6). Detto in altre parole, il test uno introduce solamente dati delle classi conosciute dal classificatore: una percentuale di degrado alta in questo caso indica che i dati del training set sotto-rappresentavano le loro classi e l'introduzione di nuovi elementi le ha rese meno coese. Per cercare di capire meglio il fenomeno, si possono pensare le classi come un insieme di punti appartenenti ad una sfera; in fase di addestramento del classificatore, i dati passati erano solo quelli vicini al centro della sfera e ora stiamo andando ad introdurre i dati più vicini al bordo. Di-

minuendo la percentuale di dati della classe 3 (test 5) questa separazione tra i dati diminuisce, per cui la curva di silhouette tende a riavvicinarsi a quella base, per poi abbassarsi nuovamente nel test 11, in cui vengono passati esclusivamente dati della classe 6, che necessariamente andranno ad aumentare il livello di separazione dei dati interni alla classe 3.

## 3.3 Poker Hand Dataset

Viene presentato ora un dataset per cui il metodo presentato, allo stato in cui è, non è efficace.

Il dataset Poker Hand è un insieme di sample che rappresentano una "mano" nel gioco del poker. Ogni mano consiste in 5 carte prese da un mazzo standard da 52 carte; ogni carta è descritta da due attributi, il primo indica il numero della carta (i.e., un numero da 1 a 13) e il secondo il suo seme (i semi sono indicati con i numeri da 1 a 4). La classe in questo caso descrive la "poker hand"; nel dettaglio:

| Valore | Descrizione |
|---|---|
| 0 | Nulla in mano |
| 1 | Una coppia |
| 2 | Due coppie |
| 3 | Un tris |
| 4 | Scala |
| 5 | Flush: 5 carte dello stesso seme |
| 6 | Full House: una coppia e un tris |
| 7 | Poker: 4 carte dello stesso valore |
| 8 | Straight Flush: 5 carte in sequenza dello stesso colore |
| 9 | Royal Flush: scala formata dalle carte più alte, tutte dello stesso seme |

Tabella 3.3: Distribuzione dei dati nelle finestre temporali

Il dataset presenta **829201 osservazioni**, ha **10 attributi** e **10 classi**.

### Analisi dei risultati ottenuti

Si è scelto di addestrare il modello sulle classi 4 e 5 e di introdurre in seguito gli elementi della classe 6.
La scelta è ricaduta su queste classi perché sono quelle che più evidenziano i problemi della silhouette come metodo di identificazione del class-based concept drift su questo dataset; ad esempio, si è provato ad addestrare il modello sulle classi 0 e 1 e introdurre successivamente i dati della classe 2. In questo caso la classe 0 in generale non è affetta da degrado, mentre la classe 1 sì, ma mai con un trend crescente ben evidente. Questo ha portato ad approfondire in maniera ulteriore l'analisi, provando ad addestrare il modello su altre classi del dataset.
La matrice delle percentuali del degrado totale mostra in maniera evidente un'incapacità del metodo di rilevare la presenza di dati di una classe aggiuntiva rispetto a quelle analizzate in fase di training. Infatti, non è possibile rilevare un trend né crescente né decrescente nelle percentuali di degrado in nessuna delle tre direzioni in cui è possibile leggere la matrice. I risultati attesi sarebbero quelli di una matrice i cui colori sfumano dal viola scuro all'arancione chiaro sia dall'alto verso il basso, sia da sinistra verso destra, sia sulla diagonale principale.
Come visibile nei grafici sottostanti, è quasi impossibile rilevare un trend nella percentuale di degrado, soprattutto nella classe 4.
Il grafico in figura 3.21 rappresenta la percentuale di degrado nella prima finestra temporale, ovvero quella in cui si vuole classificare una quantità di dati nuovi pari al 10% dei dati analizzati in fase di addestramento del modello.

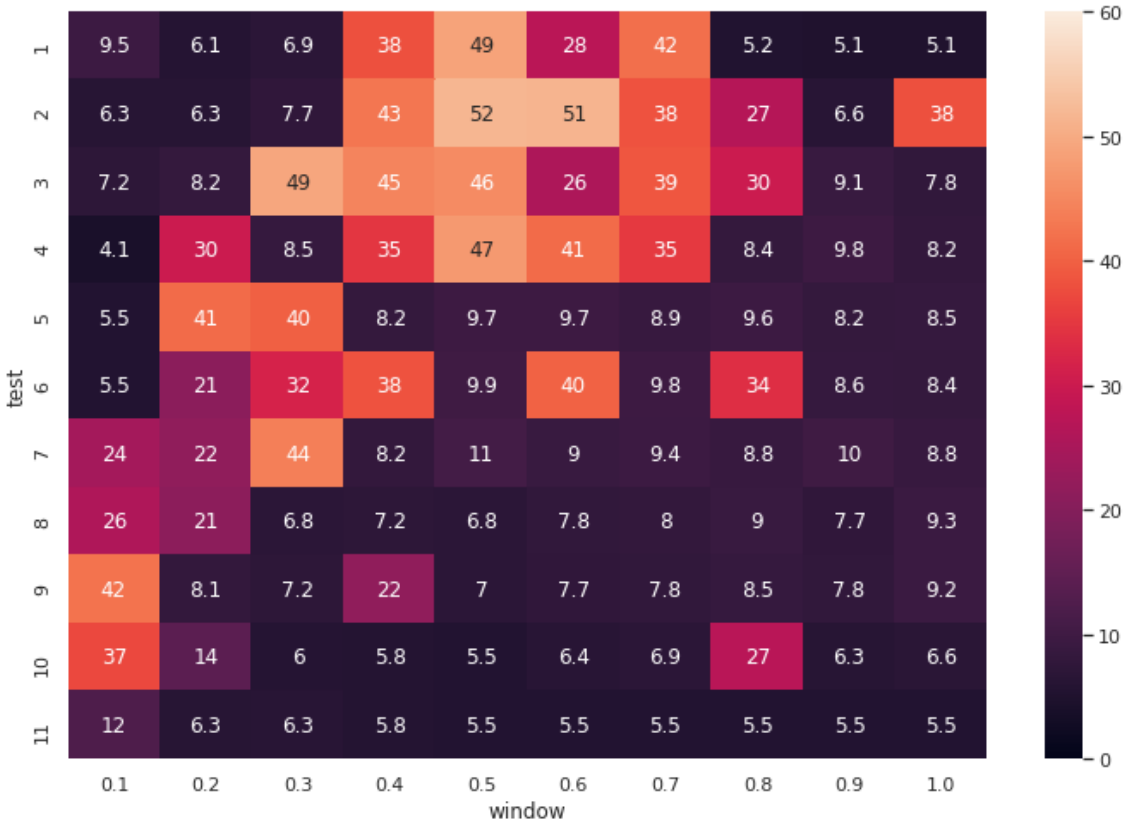
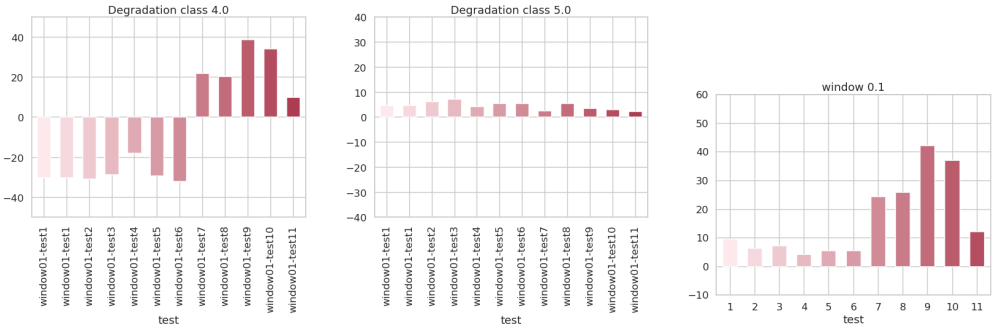



Figura 3.20: Poker Hand Dataset: Matrice dei degradi

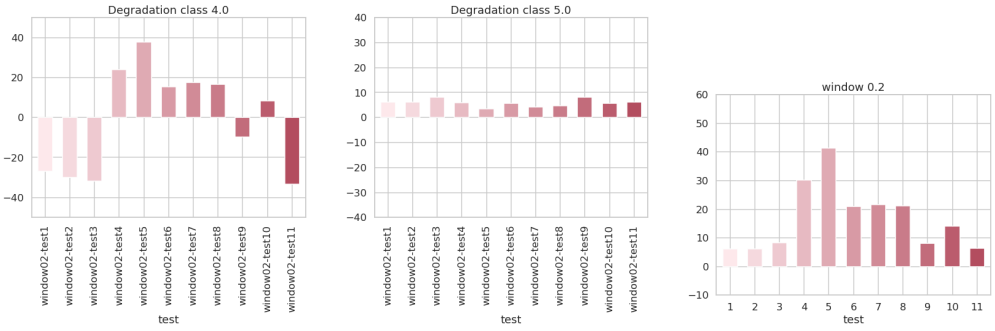
La classe 5 presenta un degrado minimo quasi costante, mentre la classe 4 presenta inizialmente un degrado negativo, come ad indicare una maggiore coesione tra i suoi elementi, per poi variare in un degrado positivo. Ciò potrebbe portare a credere che la classe 4 e la classe 6 non sono abbastanza separate, per cui un numero esiguo di sample della classe 6 molto vicini alla classe 4 potrebbe portare ad una maggiore coesione degli elementi della classe 4, mentre aumentando i dati della classe nuova, questa risulta più definita e distinguibile, cosa che si manifesta in un aumento della percentuale di degrado.




*Degrado della classe 4* *Degrado della classe 5* *Degrado totale*

Figura 3.21: Poker Hand Dataset: Window 0.1, percentuali di degrado

Se quanto ipotizzato sopra fosse vero, però, ci aspetteremmo un degrado che cresce in maniera graduale, passando da negativo a positivo. Come si può notare in figura 3.22, e anche poi nelle successive, questo fenomeno non si presenta.

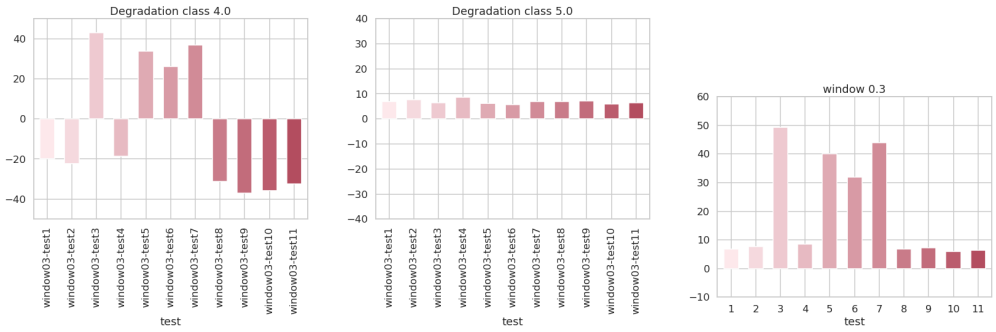



*Degrado della classe 4* *Degrado della classe 5* *Degrado totale*

Figura 3.22: Poker Hand Dataset: Window 0.2, percentuali di degrado

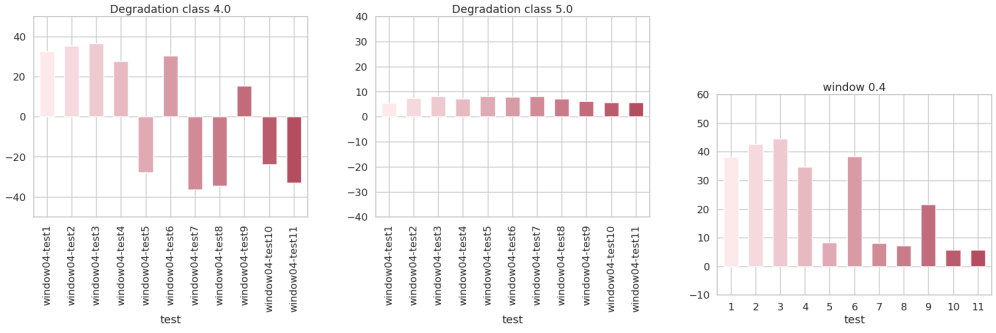
Nella figura citata, infatti, rileviamo un trend prima negativo, poi positivo e poi nuovamente negativo. Il significato di questo grafico è il seguente: introducendo fino al 20% di dati di una classe sconosciuta, i dati della classe 4 risultano più coesi; dal 30% al 70% i dati della classe 6 classificati come

dati di classe 4 rendono gli elementi di quest'ultima più separati, per poi tornare ad essere più coesi quando i dati passati al classificatore sono solo dati della classe 6. Ciò, se si pensa al significato di coesione e separazione di dati appartenenti alla stessa classe, non ha senso. Infatti, andando a classificare erroneamente come dati appartenenti alla classe in esame una quantità sempre maggiore dati esterni ad essa, meno i suoi dati saranno coesi (i.e. aumenterà la percentuale di degrado).



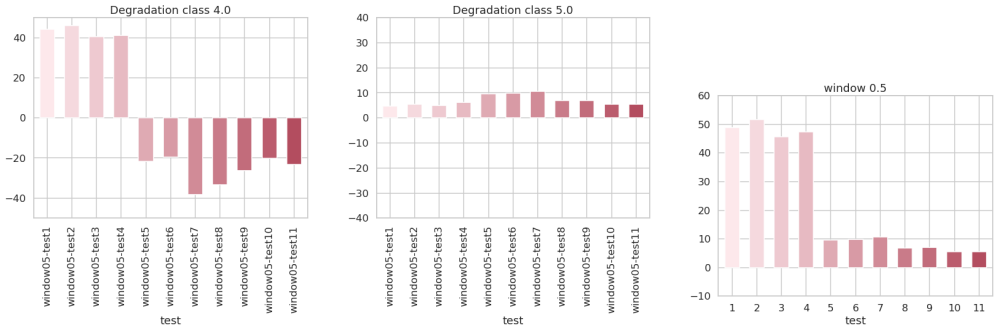

*Degrado della classe 4* *Degrado della classe 5* *Degrado totale*

Figura 3.23: Poker Hand Dataset: Window 0.3, percentuali di degrado




*Degrado della classe 4* *Degrado della classe 5* *Degrado totale*

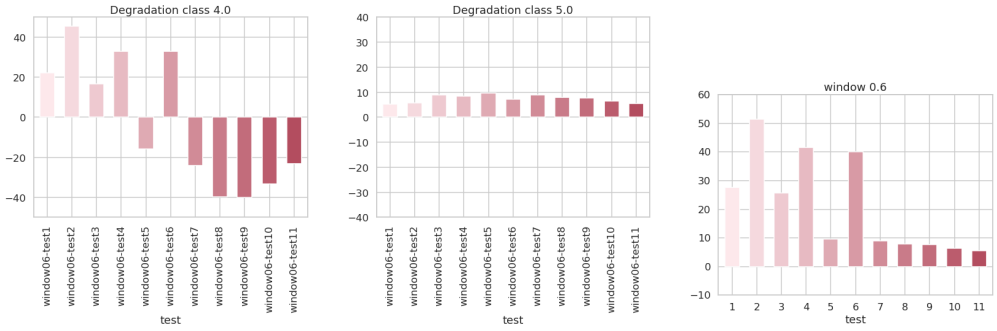
Figura 3.24: Poker Hand Dataset: Window 0.4, percentuali di degrado




*Degrado della classe 4* *Degrado della classe 5* *Degrado totale*

Figura 3.25: Poker Hand Dataset: Window 0.5, percentuali di degrado

Nella finestra 0.5, figura 3.25, ovvero quella in cui viene chiesto al modello di classificare una mole di dati pari al 50% di quelli che sono serviti per addestrare il modello, si registra un degrado molto alto quando si introduce una piccola percentuale di dato esterno alla classe 4. Questo diventa poi negativo all'aumentare della quantità di dati della classe 6 analizzati.

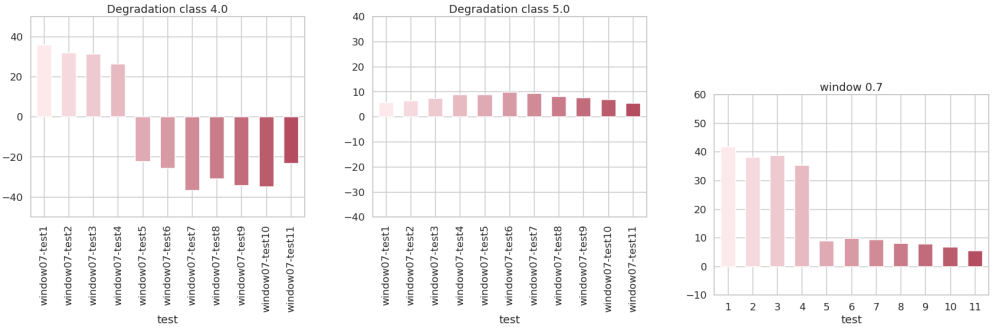



*Degrado della classe 4* *Degrado della classe 5* *Degrado totale*

Figura 3.26: Poker Hand Dataset: Window 0.6, percentuali di degrado

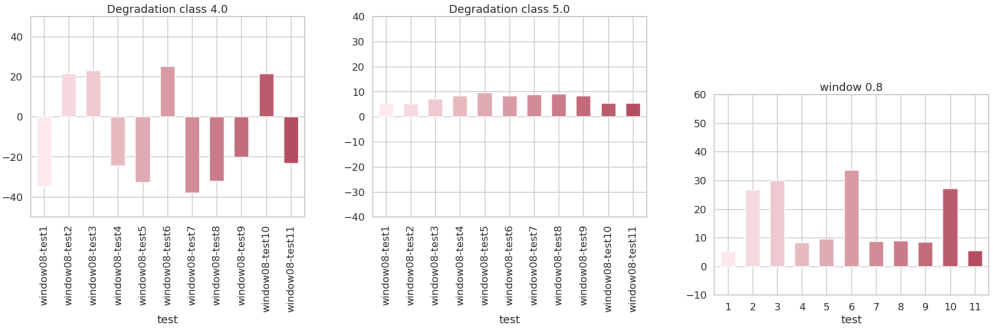
Ciò indica una maggiore separazione dei dati nella classe 4 (i.e. il classificatore ha assegnato un'etichetta sbagliata ai dati, indicando come elementi di classe 4 una serie di dati che in realtà non lo erano) in una fase iniziale, che

si trasforma in una forte coesione di questi dal test 5 in poi. Questo trend si registra poi nuovamente nella finestra 0.7, figura 3.27. Se questo trend si fosse presentato in tutte le finestre, evidenziando un degrado decrescente in maniera graduale, avremmo potuto concludere che le classi 4 e 6 sono molto vicine tra loro, per cui aumentando i dati di classe 6 in analisi i due cluster tendono ad essere sempre più vicini, confondendosi in un unico cluser (con risultato una maggiore coesione dei dati della classe 4).




*Degrado della classe 4* *Degrado della classe 5* *Degrado totale*

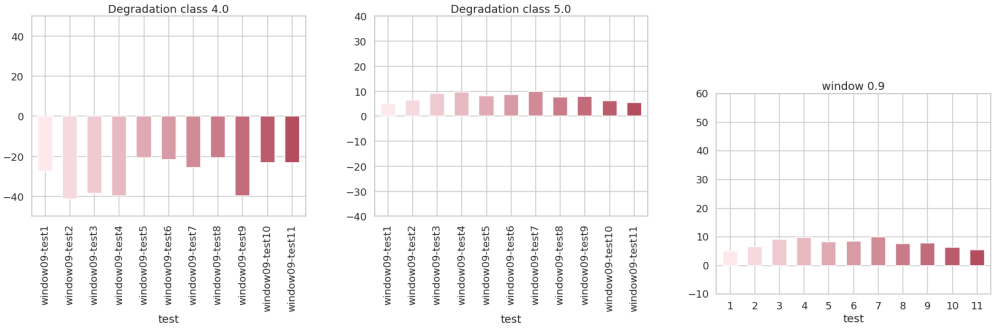
Figura 3.27: Poker Hand Dataset: Window 0.7, percentuali di degrado




*Degrado della classe 4* *Degrado della classe 5* *Degrado totale*

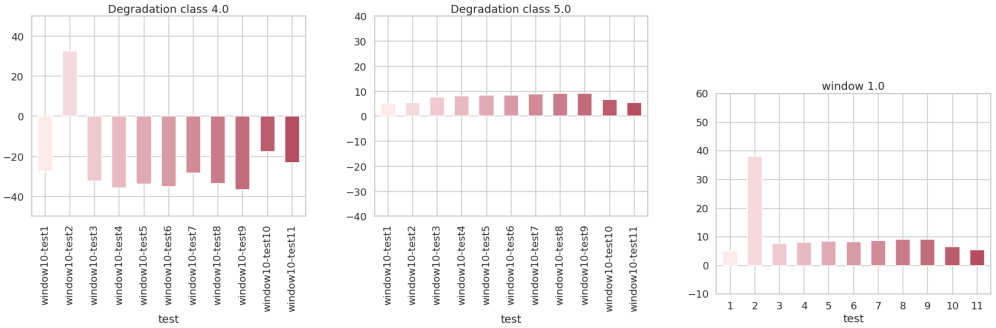
Figura 3.28: Poker Hand Dataset: Window 0.8, percentuali di degrado

Le ultime due finestre registrano un degrado della classe 4 quasi sempre negativo: quando la quantità di dati da analizzare è molto grande (e dunque, anche il numero di dati di classe 6 è molto significativo), la classe 4 registra un grado di coesione maggiore rispetto a quello evidenziato in fase di addestramento del modello.




*Degrado della classe 4* *Degrado della classe 5* *Degrado totale*

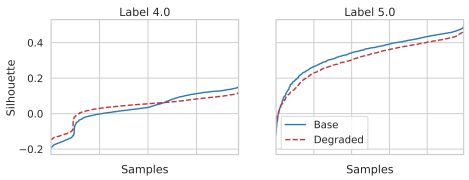
Figura 3.29: Poker Hand Dataset: Window 0.9, percentuali di degrado



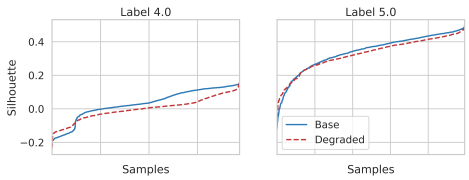

*Degrado della classe 4* *Degrado della classe 5* *Degrado totale*

Figura 3.30: Poker Hand Dataset: Window 1.0, percentuali di degrado

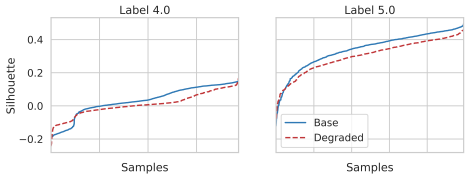
Ciò però non può essere vero, in quanto sappiamo che stiamo introducendo elementi di una classe esterna ad esso, che il classificatore assegna alla classe 4 o alla classe 5. Dato il livello di degrado praticamente assente nella classe 5, si può dedurre che quasi tutti i dati della classe 6 sono stati assegnati alla classe 4. Questo dovrebbe tradursi in un degrado della classe 4 che cresce al crescere della percentuale di dati della classe 6 analizzata (i.e. il degrado dovrebbe essere basso nel test 1 e alto nel test 11).
Quanto detto finora è riassunto negli indici di silhouette presentati in figura 3.31; si è scelto di mostrare, per ogni finestra, i valori dell'indice calcolati nel test 5 (ovvero, quando le percentuali dei dati delle classi 4,5,6 sono rispettivamente 30%,30%,40%).
La silhouette per la classe 5 indica che gli elementi interni ad essa presentavano una moderata coesione già nel training set; la curva di silhouette calcolata a seguito dell'introduzione di nuovi dati evidenzia una separazione sempre maggiore dei dati interni alla classe: infatti, nella finestra 0.1 la curva di silhouette degraded giace leggermente sotto alla curva di silhouette base, mentre aumentando la percentuale di dati nuovi passati al classificatore la curva tende ad abbassarsi sempre di più. Andando ad analizzare, invece, la silhouette per i dati della classe 4, si evidenzia una coesione quasi assente dei dati anche in fase iniziale; la curva di silhouette base è molto vicina a zero, andando ad indicare che i valori $a(i)$ e $b(i)$ sono quasi uguali e quindi non è chiaro se il valore $i$ debba essere assegnato alla classe $A$ o alla classe $B$. Questa incertezza si propaga anche nella curva di silhouette degraded, la quale in quasi tutte le finestre temporali tende a sovrapporsi a quella base.



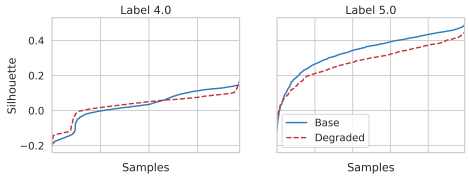

*Window 0.1 Test 5*



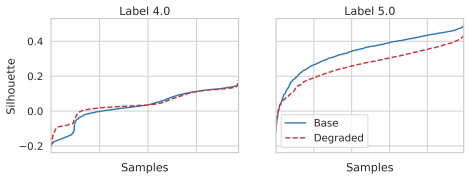

*Window 0.2 Test 5*



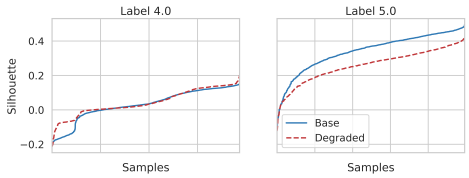

*Window 0.3 Test 5*



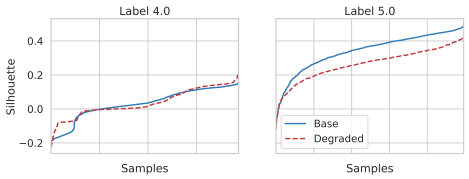

*Window 0.4 Test 5*



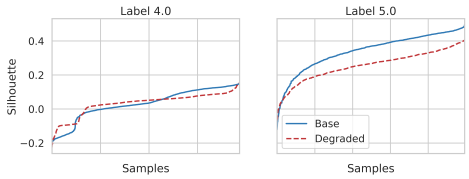

*Window 0.5 Test 5*



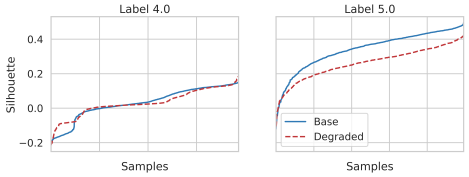

*Window 0.6 Test 5*



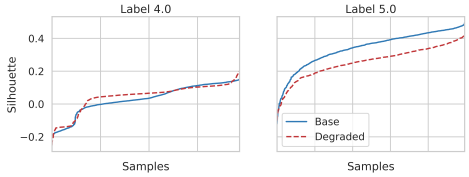

*Window 0.7 Test 5*




*Window 0.8 Test 5*




*Window 0.9 Test 5*




*Window 1.0 Test 5*

Figura 3.31: Poker Hand Dataset: silhouette

Dal comportamento della silhouette nella classe 4, si evince un'incapacità del classificatore nel classificare i dati anche senza l'introduzione di sample della classe 6. Un'ipotesi del perché di questo fenomeno potrebbe essere che le classi del dataset in questione non siano ben separate tra loro; infatti, un indice di silhouette (che, ricordiamo, si basa sul concetto di distanza del punto in analisi con quelli già presenti nei vari cluster) vicino allo zero indica che il dato poteva appartenere sia alla classe a cui è stato assegnato, sia ad un'altra. Questo è confermato dal metodo del gomito per l'algoritmo di clustering K-Means illustrato in figura 3.32; il metodo in questione è un metodo grafico che permette di trovare il numero di cluster ottimale per il dataset in analisi. Nella pratica, l'algoritmo K-Means raggruppa i vari dati in $k$ cluster, dove $k$ è definito dall'utente; per trovare il $k$ ottimale si itera l'algoritmo andando a variare ogni volta il numero di cluster che l'algoritmo dovrebbe riconoscere. Si stampa poi la somma delle distanze al quadrato tra ogni centroide ed i punti del proprio cluster in relazione al numero $k$ di cluster; a questo punto si legge il grafico creato da destra a sinistra e si definisce il $k$ ottimale quello tale per cui la curva, dopo di esso, tende a crescere in maniera consistente. Nella figura 3.32, il $k$ ottimale risulta essere 3, quando in realtà il numero di cluster realmente esistenti è 10.

```
[1]: import pandas as pd

[2]: df=pd.read_csv(r"C:\Users\Valeria\.spyder-py3\poker.csv")

[3]: X = df.drop(["y"], axis=1)

[4]: from sklearn.cluster import KMeans

     distortions = []
     K = range(1,30)
     for k in K:
         kmeanModel = KMeans(n_clusters=k)
         kmeanModel.fit(X)
         distortions.append(kmeanModel.inertia_)

[5]: import matplotlib.pyplot as plt
     plt.figure(figsize=(16,8))
     plt.plot(K, distortions, 'x-')
     plt.xlabel('k')
     plt.ylabel('Distortion')
     plt.title('Metodo del gomito per la ricerca del k ottimale')
     plt.show()
```

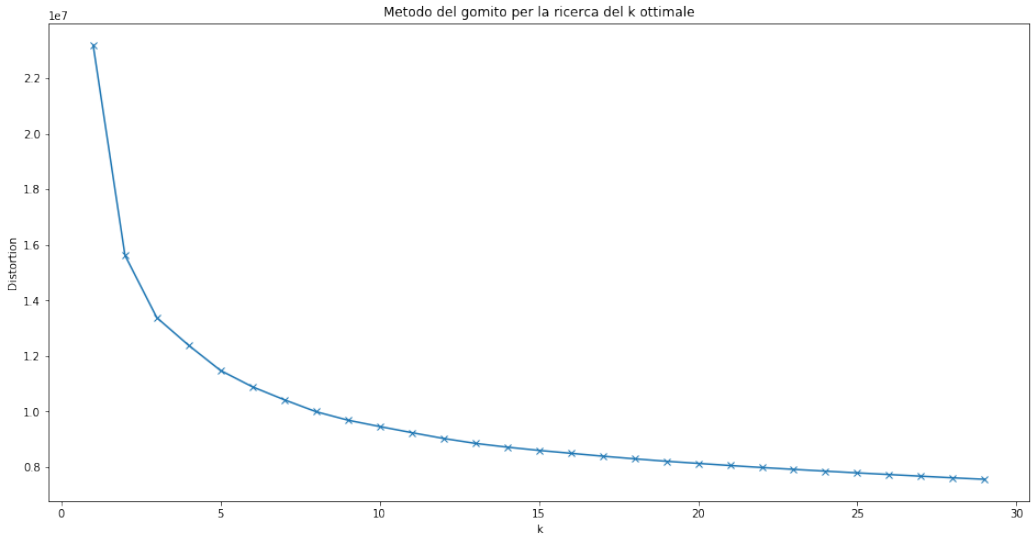



Figura 3.32: Metodo del gomito per la ricerca del numero ottimale di cluster

Per cui, rietichettando i sample del dataset assegnandoli ai loro cluster naturali, ci aspettiamo che il metodo sia in grado di rilevare la presenza di class-based concept drift.
Abbiamo dunque addestrato il modello sulle classi 0 e 1 e abbiamo introdotto in maniera graduale gli elementi della classe 2.

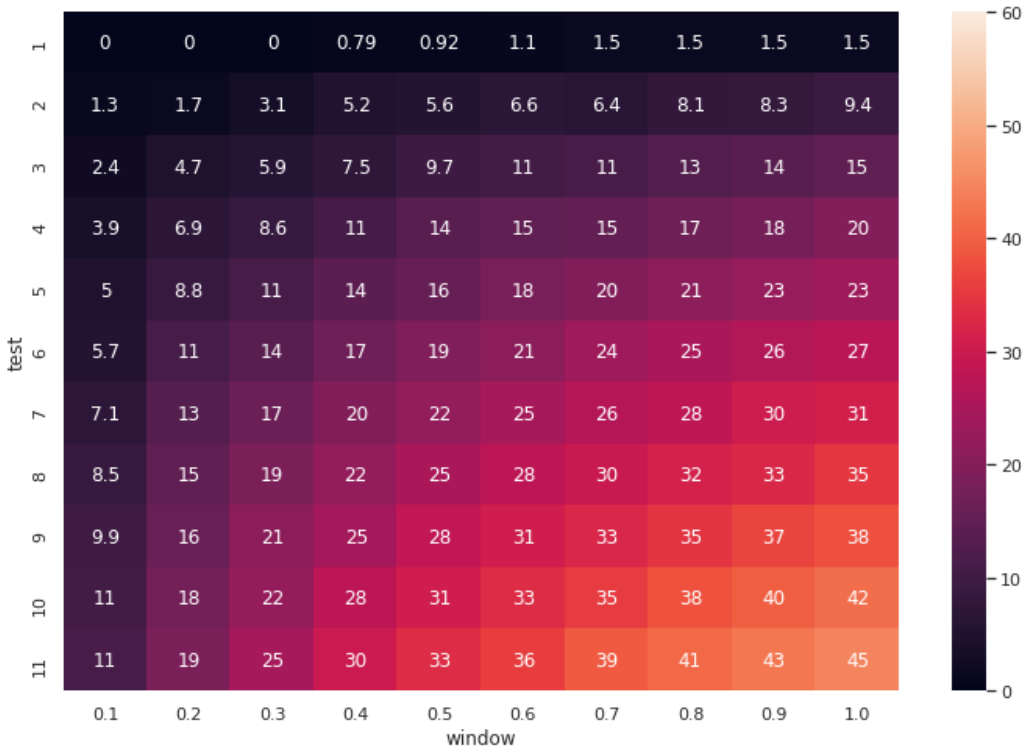
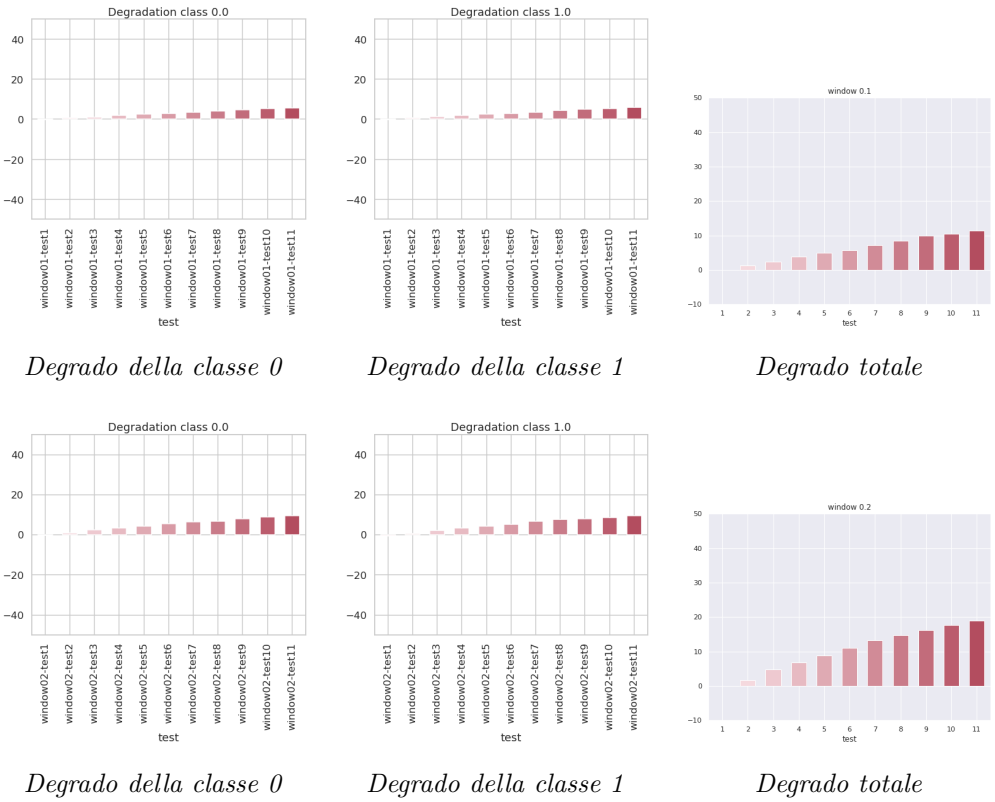



Figura 3.33: Poker Hand Dataset Modified: Matrice dei degradi

La matrice in figura 3.33 mostra un degrado crescente in tutte e tre le direzioni; è dunque evidente la miglioria del modello rispetto al dataset di partenza. Nel caso di classi ben separate tra loro, infatti, la silhouette è in grado di evidenziare in maniera chiara la corretta o scorretta assegnazione di un dato ad una classe da parte del classificatore.

Essendoci ricondotti ad un caso ideale, i grafici della percentuale di de-

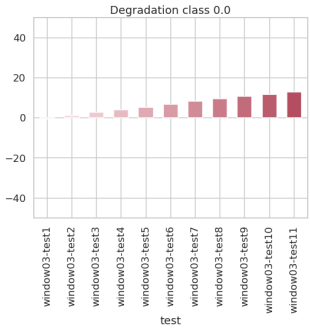
grado delle due classi e quello della percentuale di degrado totale riveleranno un trend crescente e graduale, che aumenta sia all'aumentare della percentuale di dato esterno alle due classi analizzate in fase di addestramento del modello, sia all'aumentare del numero totale di dati analizzati. Nella realtà, però, è difficile trovare dei dataset le cui classi sono così ben distinte tra loro; i risultati ottenuti nei capitoli 3.1 e 3.2, seppur non perfetti come quelli illustrati in questa sezione, sono esempi di come, nella realtà, l'indice di silhouette è in grado di evidenziare la presenza di class-based concept drift.




*Degrado della classe 0* *Degrado della classe 1* *Degrado totale*

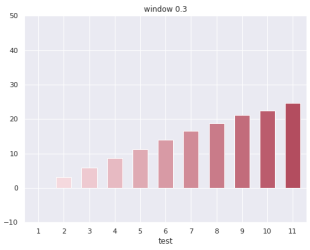
*Degrado della classe 0* *Degrado della classe 1* *Degrado totale*

Figura 3.34: Poker Hand Dataset Modified: Windows 0.1, 0.2, percentuali di degrado

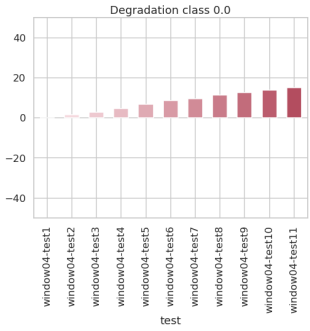



*Degrado della classe 0*

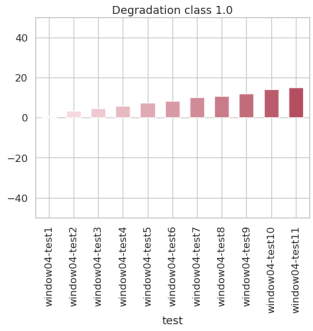

*Degrado della classe 1*



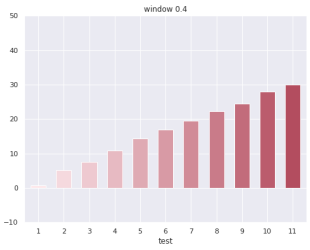

*Degrado totale*



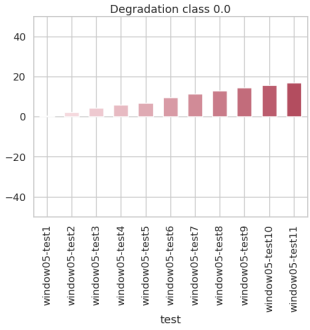

*Degrado della classe 0*



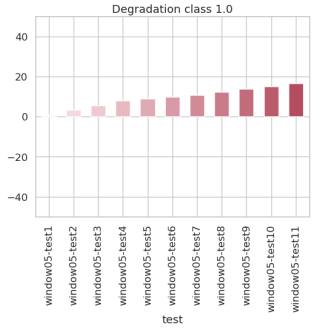

*Degrado della classe 1*



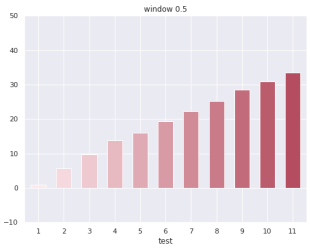

*Degrado totale*




*Degrado della classe 0*

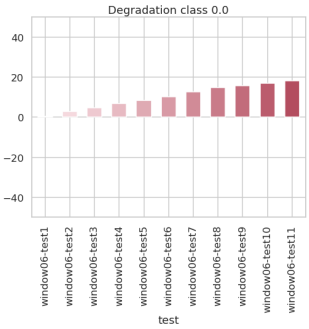



*Degrado della classe 1*



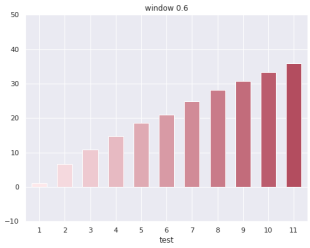

*Degrado totale*

Figura 3.35: Poker Hand Dataset Modified: Windows 0.3, 0.4, 0.5, percentuali di degrado

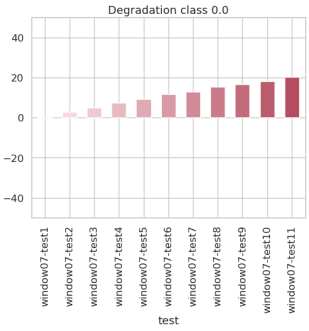



*Degrado della classe 0*

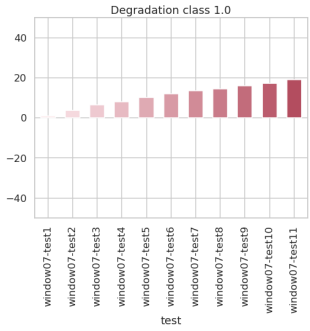

*Degrado della classe 1*



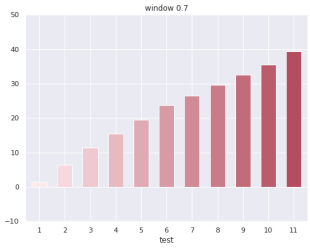

*Degrado totale*



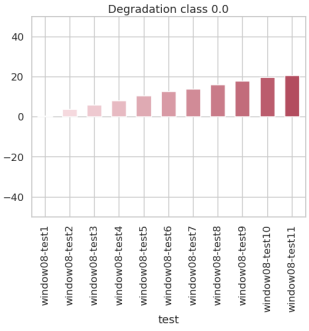

*Degrado della classe 0*



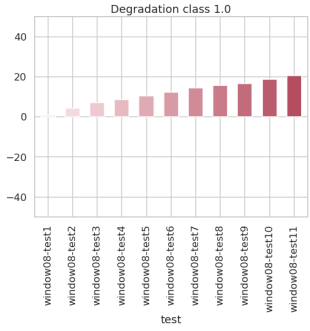

*Degrado della classe 1*



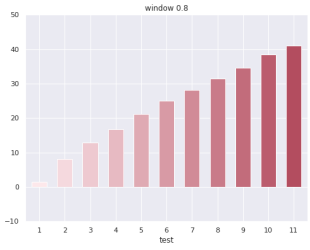

*Degrado totale*




*Degrado della classe 0*

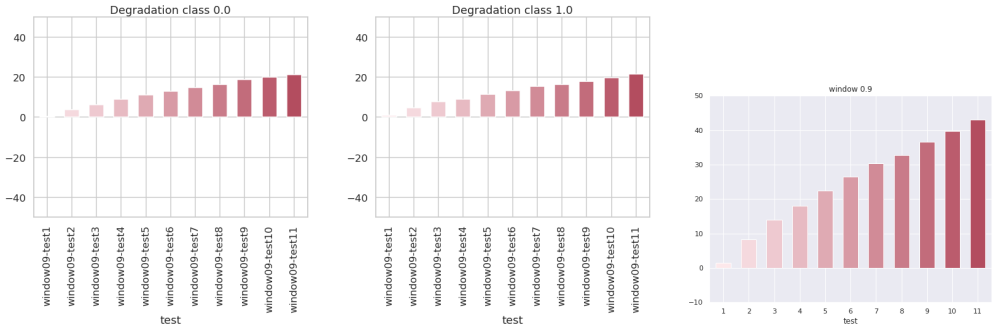



*Degrado della classe 1*

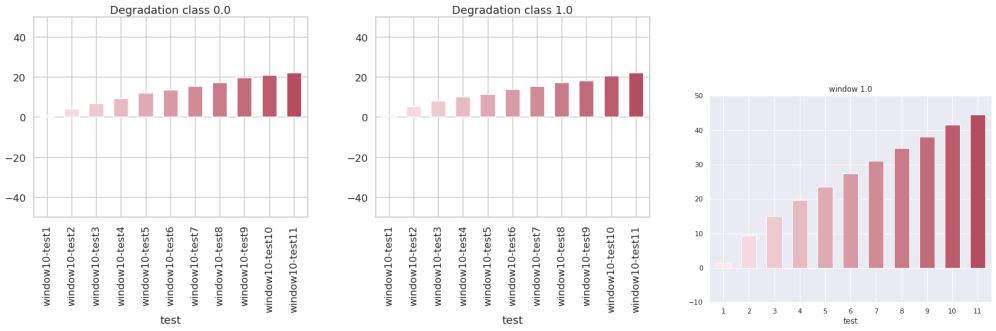



*Degrado totale*

Figura 3.36: Poker Hand Dataset Modified: Windows 0.6, 0.7, 0.8, percentuali di degrado



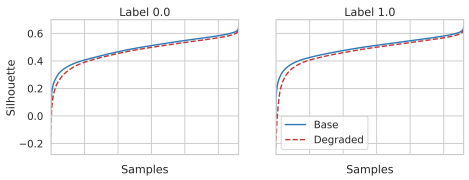

*Degrado della classe 0* *Degrado della classe 1* *Degrado totale*



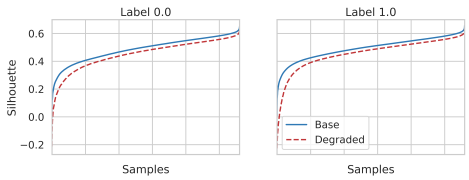

*Degrado della classe 0* *Degrado della classe 1* *Degrado totale*

Figura 3.37: Poker Hand Dataset Modified: Windows 0.9, 1.0, percentuali di degrado

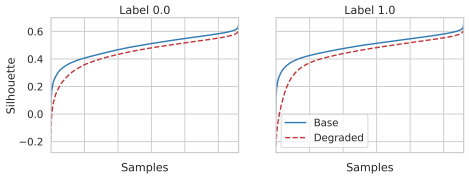
Le curve di silhouette rappresentate nella figura 3.38 evidenziano quanto detto finora: infatti, sia nel caso della classe 0, sia nel caso della classe 1, la silhouette di base giace sopra a quella degraded, andando ad evidenziare che la precisione del modello è peggiorata. Questo fenomeno si evidenzia nel tempo per entrambe le classi; andando a confrontare le curve nelle varie finestre, notiamo che la curva di silhouette degraded tende ad avere valori sempre più bassi man mano che aumentano i dati da analizzare, confermando il ragionamento fatto in precedenza.



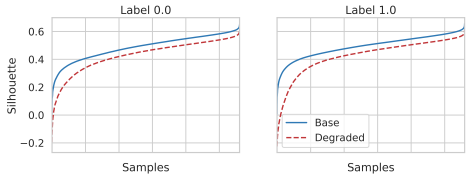

*Window 0.1 Test 5*



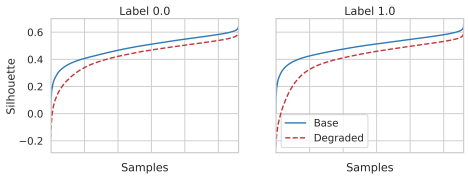

*Window 0.2 Test 5*



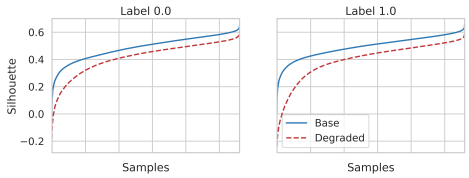

*Window 0.3 Test 5*



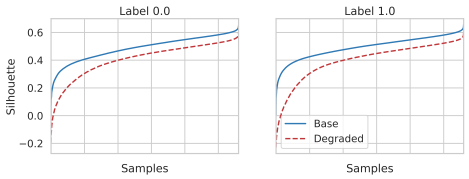

*Window 0.4 Test 5*



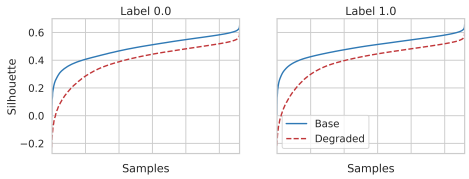

*Window 0.5 Test 5*



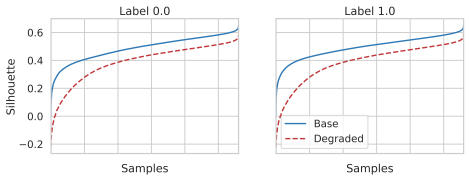

*Window 0.6 Test 5*



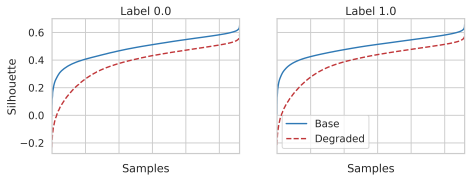

*Window 0.7 Test 5*




*Window 0.8 Test 5*




*Window 0.9 Test 5*




*Window 1.0 Test 5*

Figura 3.38: Poker Hand Dataset Modified: silhouette

# Conclusioni

Il problema dell'individuazione del class-based concept drift è complesso e a oggi non presenta una soluzione universale. In questo documento ci si è occupati della verifica della generalità di una metodologia unsupervised che si propone di risolvere questo problema. Il metodo proposto sfrutta l'indice di silhouette per valutare se l'assegnazione del classificatore è stata corretta o meno, utilizzando il concetto di distanza tra i punti: se la distanza media del sample in questione dai punti della classe a cui è stato assegnato è maggiore della distanza media di questo con i punti di un'altra classe, allora l'assegnazione risulta scorretta.

Si è scelto di analizzare tre dataset ampiamente utilizzati nello studio del concept drift; i primi due hanno fornito i risultati attesi. Il terzo dataset, invece, rappresenta un caso in cui l'indice di silhouette risulta incapace di valutare la qualità dell'assegnazione; infatti, già in fase di addestramento la curva di silhouette per una delle classi risulta essere prossima allo zero, evidenziando una situazione in cui quasi tutti i sample potevano essere assegnati in maniera equivalente ad un'altra classe. Ciò ha portato a concludere che, nel caso di dataset ben separati tra di loro, la metodologia allo stato in cui si trova è efficace.

Sviluppi futuri di questo studio sono, in prima istanza, l'estensione della validazione sperimentale a nuovi dataset per migliorare la conoscenza delle condizioni di funzionamento corretto della metodologia proposta; inoltre, per rendere più generale possibile l'approccio, questo va esteso a nuove misure di distanza, in base al tipo di dato che il dataset presenta.
Un ulteriore e interessante sviluppo è quello di ricercare nuove metriche non supervisionate alternative alla silhouette, valutandone l'efficacia e confrontando i risultati ottenuti con quelli riscontrati applicando l'indice di silhouette.

# Bibliografia


[1] *Towards a real-time unsupervised estimation of predictive model degradation*, T. Cerquitelli, S. Proto, F. Ventura, D. Apiletti, E. Baralis.

[2] *Learning under Concept Drift: A review*, J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang.

[3] *A Concept Drift-tolerant casa-base editing technique*, N. Lu, J. Lu, G. Zhang, R. Lopez de Mantaras.

[4] *Concept Drift detection via competence models*, N. Lu, G. Zhang, J. Lu.

[5] *Learning under concept drift: an overview*, I. Zliobaite.

[6] *Concept Drift adaptation for learning with streaming data*, A. Liu.

[7] *Detection of Abrupt Changes: Theory and Applications*, M. Basseville, I.V. Nikiforov.

[8] *A new unsupervised predictive-model self-assessment approach that SCALEs*, F. Ventura, S. Proto, D. Apiletti, T. Cerquitelli, S. Panicucci, E. Baralis, E. Macii.

[9] *A Detailed Analysis of the KDD CUP 99 Data Set*, M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani.

[10] *Silhouettes: a graphical aid to the interpretation and validation of cluster analysis*, P. J. Rousseeuw.

[11] *Extracting salient features for network intrusion detection using machine learning methods*, R. C. Staudemeyer.