Progetto Sistemi Intelligenti su Internet

Spam detection per i commenti di YouTube

Di Fonzo, Illuminati

Sommario

Introduzione

Lo spam è un contenuto informativo irrilevante e indesiderato per chi lo riceve.
In questo progetto si è implementato un classificatore che rilevi in modo automatico i commenti di YouTube ritenuti spam.

Panoramica progetto

Il progetto ha riguardato diverse fasi di studio. Si sono dapprima esaminate e studiate le API di YouTube; sono state quindi elaborate 21 feature di classificazione per determinare se un commento di YouTube fosse spam o meno; si sono quindi estratti tramite API i dati utili, salvati in

forma grezza in un file json e in forma elaborata in un file csv per essere utilizzati come feature; utilizzando Weka si è quindi addestrato il classificatore e se ne sono valutati i risultati.

Estrazione dei commenti

Si è costruito un piccolo software in Java in grado di:

- estrarre, dato un video di YouTube, i commenti più rilevanti e più recenti, con i dati più interessanti per la rilevazione di spam;
- creare un file .txt con struttura json con i dati grezzi estratti;
- creare un file .csv con i dati elaborati in 21 feature utili per il classificatore.

Feature e dati estratti

Sono state pensate 21 feature utili per la classificazione di un commento in lingua inglese di YouTube, di cui alcune tratte da testi e documenti di ricerca riguardanti lo spam detection.

- **isReply**
  booleano
  Indica se il commento è una risposta a un altro commento.
  Con questa feature è possibile distinguere commenti dalle risposte ai commenti e determinare una possibile differenza di presenza di spam.
- **authorNameInComment**
  booleano
  Indica se è presente il nome dell'autore del video nel commento.
  E' probabile che un commento in cui è presente il nome dell'autore del video sia rilevante.
- **isCommentFromAuthor**
  booleano
  Indica se il commento è stato scritto dall'autore del video.
  Un commento da parte dell'autore del video non può essere spam.
- **commenterNameInComment**
  booleano
  Indica se è presente nel commento il nome dell'utente che ha scritto il commento stesso.
  Difficilmente un utente si autocita in un commento; se ciò invece accade può trattarsi di un caso di spam.
- **numbersInCommenterName**
  numerico
  Indica il numero massimo di numeri consecutivi presenti nel nome dell'utente che ha scritto il commento.
  Un nickname con molti numeri potrebbe essere un bot che scrive commenti di spam in modo automatico.
- **likes**
  numerico
  Indica il numero di likes (pollici in su) che ha ricevuto il commento.
  Un commento con molti apprezzamenti difficilmente sarà spam.

- **replies**
  numerico
  Indica il numero di risposte che ha ricevuto il commento.
  Un commento che ha molte risposte vuol significare che ha generato una discussione, tale che l'utenza è motivata a intervenire, pertanto difficilmente sarà spam.
- **daysInterval**
  numerico
  Indica l'intervallo di giorni passati dalla creazione del video a quello del commento.
  Più giorni sono passati dal caricamento del video, meno probabilmente tale video sarà rilevante, tale da scrivere su di esso un commento, e quindi potrebbero aumentare i casi di spam.
- **repeatedWords**
  numerico
  Indica il numero massimo di parole identiche consecutive presenti nel commento.
  Se un commento possiede molte parole ripetute consecutivamente potrebbe essere un caso di spam.
- **mentionTags**
  numerico
  Indica il numero di mention tags (etichetta preceduta dal segno '+' che si usa per citare o rivolgersi a un utente specifico di YouTube) presenti nel commento.
  Il numero di mention tags potrebbe essere fattore rilevante per la classificazione, in particolare più sono numerosi, più la copertura (raggiungibilità e visibilità) del commento aumenta.
- **hashtags**
  numerico
  Indica il numero di hashtags (parole chiave preceduta dal segno '#' che si usa per etichettare il commento, associarlo a un concetto) presenti nel commento.
  Il numero di hashtags potrebbe essere fattore rilevante per la classificazione, in particolare più sono numerosi, più la copertura (raggiungibilità e visibilità) del commento aumenta.
- **commenterSubscribers**
  numerico
  Indica il numero di iscritti (followers) dell'utente che ha scritto il commento.
  Se l'utente possiede un gran numero di iscritti, difficilmente potrà scrivere commenti contenenti spam.
- **videoMomentReference**
  booleano
  Indica se è presente nel commento un riferimento a un istante preciso del video (la cui sintassi è 'ora:minuto:secondo').
  Un commento che fa riferimento a un momento preciso del video difficilmente sarà irrilevante e indesiderato.
- **homePageLink**
  booleano
  Indica se è presente nel commento un link a una home page.

Commenti contenenti link, e in particolare link di homepage, sono più propensi a essere casi di spam.

- **redirectingLink**
  booleano
  Indica se è presente nel commento un link che possiede almeno una redirezione.
  Commenti con link con più redirezioni possono essere fortemente sospetti di spam.
- **IP**
  booleano
  Indica se è presente nel commento un indirizzo IP.
  Commenti contenenti indirizzi IP sono sospetti e probabilmente si trattano di casi di spam.
- **percentageTitleWordsInComment**
  numerico
  Indica la percentuale di parole, appartenenti al titolo del video, presenti nel commento.
  Un commento che possiede parole appartenenti allo stesso campo semantico di parole utilizzate dall'autore del video per descrivere quest'ultimo, è meno propenso a essere un caso di spam.
- **percentageTagsInComment**
  numerico
  Indica la percentuale di parole, appartenenti ai tag del video, presenti nel commento.
  Un commento che possiede parole appartenenti allo stesso campo semantico di parole utilizzate dall'autore del video per descrivere quest'ultimo, è meno propenso a essere un caso di spam.
- **percentageCapsLock**
  numerico
  Indica la percentuale di caratteri maiuscoli presenti nel commento.
  Un commento contenente parole maiuscole potrebbe essere un caso di spam in quanto ha intenzione di richiamare l'attenzione di chi legge.
- **percentageSpellingErrors**
  numerico
  Indica la percentuale delle parole presenti nel commento sbagliate grammaticalmente.
  Un commento fortemente sgrammaticato potrebbe essere un caso di spam.
- **blackWords**
  numerico
  Indica il numero di parole, appartenenti a una lista di parole chiave ritenute indicatrici di un messaggio di spam, presenti nel commento.
  Un commento contenente un numero significativo di black words potrebbe trattarsi molto probabilmente di spam.

Per realizzare queste feature si sono quindi estratti tramite API i dati necessari. E' possibile vederne l'implementazione con relativi commenti sul codice della classe *FeatureProcessing.java*.

Addestramento del classificatore

Algoritmi utilizzati

Di seguito sono spiegati i metodi di classificazione presi in considerazione per l'addestramento del sistema di spam detection. Sono stati scelti vari algoritmi di classificazione in modo da avere una visione più ampia di quanto siano efficaci per l'addestramento il dataset raccolto e le feature selezionate. Ad ogni utilizzo del filtro SMOTE gli algoritmi sono stati nuovamente testati per verificare i benefici ottenuti dall'incremento e dal bilanciamento del dataset.

ZeroR

Il Classificatore ZeroR assegna tutte le istanze alla classe di maggiore dimensione presente nel training-set senza mai considerare gli attributi di ciascuna istanza. È un classificatore in genere poco accurato dal momento che non usa un vero e proprio algoritmo.

J48

È una variante dell'algoritmo C4.5, che consiste nel generare un albero di classificazione in cui ad ogni livello vi è una regola. Quando si deve classificare un'istanza si percorre l'albero seguendo le regole presenti nei vari nodi, fino ad arrivare a una foglia che indica la classe a cui appartiene l'istanza.

NaiveBayes

È un classificatore bayesiano semplificato, in cui si fa l'ipotesi che le feature siano indipendenti tra loro, cioè che la presenza o l'assenza di una certa feature non è correlata alla presenza o assenza di altre feature. Per evitare problemi di divisione per zero, dovuti a probabilità nulle, i contatori sono incrementati di 1.

BayesNet

È un metodo per apprendere tramite reti bayesiane. La struttura della rete può essere fissata o può essere appresa dall'algoritmo, e si possono modificare sia l'euristica usata per la ricerca della struttura della rete, sia il metodo usato per stimare le probabilità condizionate.

K-NN

È il metodo dei k nearest neighbor. L'elemento in analisi viene classificato in base a come sono classificati i suoi k vicini cioè gli elementi più simili ad esso. Nelle analisi successive si sono presi in considerazione per k i valori 1, 3 e 5, in modo da avere risultati più vari.

Metriche di valutazione

Le metriche prese in considerazione in questa relazione sono elencate e spiegate di seguito. I risultati dettagliati possono essere visionati nell'apposita cartella "risultati_addestramenti".

La precision indica, in percentuale, quante istanze classificate in un certo modo che sono state classificate correttamente.

La recall indica, in percentuale, quante istanze di una certa classe sono state classificate correttamente.

Il concetto di veri positivi coincide con quello di recall, mentre i falsi positivi sono le istanze classificate erroneamente in una certa classe.

L'accuratezza è una metrica relativa al classificatore stesso e rappresenta la percentuale di istanze classificate correttamente sul numero totale di istanze classificate.

L'errore assoluto relativo (relative absolute error in inglese) prende l'errore assoluto totale e lo normalizza dividendo per l'errore assoluto totale del semplice previsore. La formula che lo calcola è

$$RAE = \frac{\sum_{i=1}^{N} |\hat{\theta}_i - \theta_i|}{\sum_{i=1}^{N} |\bar{\theta} - \theta_i|}$$

dove $\bar{\theta}$ è il valore medio di θ.

Weka

Training set

Inizialmente è stato fatto un tentativo di addestramento con l'intero training set per osservare i risultati ottenuti in questo modo e per confrontarli con quelli ottenuti dalla Cross Validation, metodo molto più usato nella pratica.

ZeroR: accuratezza del 77%. Errore assoluto relativo 100%. Classifica correttamente solo i commenti non spam.

J48: accuratezza 92%. Errore assoluto relativo 37%. Non riconosce lo spam in un caso su quattro circa. Riconosce il non spam in quasi tutti i casi.

NaiveBayes: accuratezza 46%. Errore relativo assoluto 137%. Riconosce bene lo spam nell'88% dei casi, il non spam viene riconosciuto una volta su tre.

BayesNet: accuratezza 84%. Errore assoluto relativo 64%. Riconosce lo spam circa una volta su tre. Riconosce il non spam in quasi tutti i casi.

1NN: risultati perfetti, quindi l'algoritmo non addestra bene.

3NN: accuratezza 90% circa. Errore assoluto relativo 37%. Riconosce lo spam nel 70% dei casi e il non spam nel 95% dei casi.

5NN: simile a 3NN, lieve peggioramento nel riconoscere lo spam (65%).

Commento generale
La maggior parte degli algoritmi non danno risultati soddisfacenti. J48 e 3NN hanno dato dei risultati abbastanza buoni, ma migliorabili soprattutto nel riconoscimento dello spam. I risultati

sono tali che potrebbero essere usati in pratica, ma tenendo conto che non si è usato il metodo della Cross Validation e che quindi i risultati e le metriche potrebbero peggiorare.

Cross Validation

L'addestramento con Cross Validation è sempre stato fatto dividendo il dataset in 10 parti.

J48: accuratezza 83%. Errore assoluto relativo 61%. Riconosce lo spam in circa la metà dei casi, e il non spam nel 93%.

NaiveBayes: accuratezza 49%. Errore assoluto relativo 138%. Riconosce lo spam nell'80% dei casi, e il non spam nel 40%.

BayseNet: accuratezza 83%. Errore assoluto relativo 69%. Riconosce lo spam una volta su tre, e il non spam nel 97% dei casi.

1NN: accuratezza 80%. Errore assoluto relativo 56%. Riconosce lo spam nella metà dei casi e il non spam nell'88%.

3NN: accuratezza 83%. Errore assoluto relativo 59%. Riconosce lo spam nella metà dei casi e il non spam nel 92%.

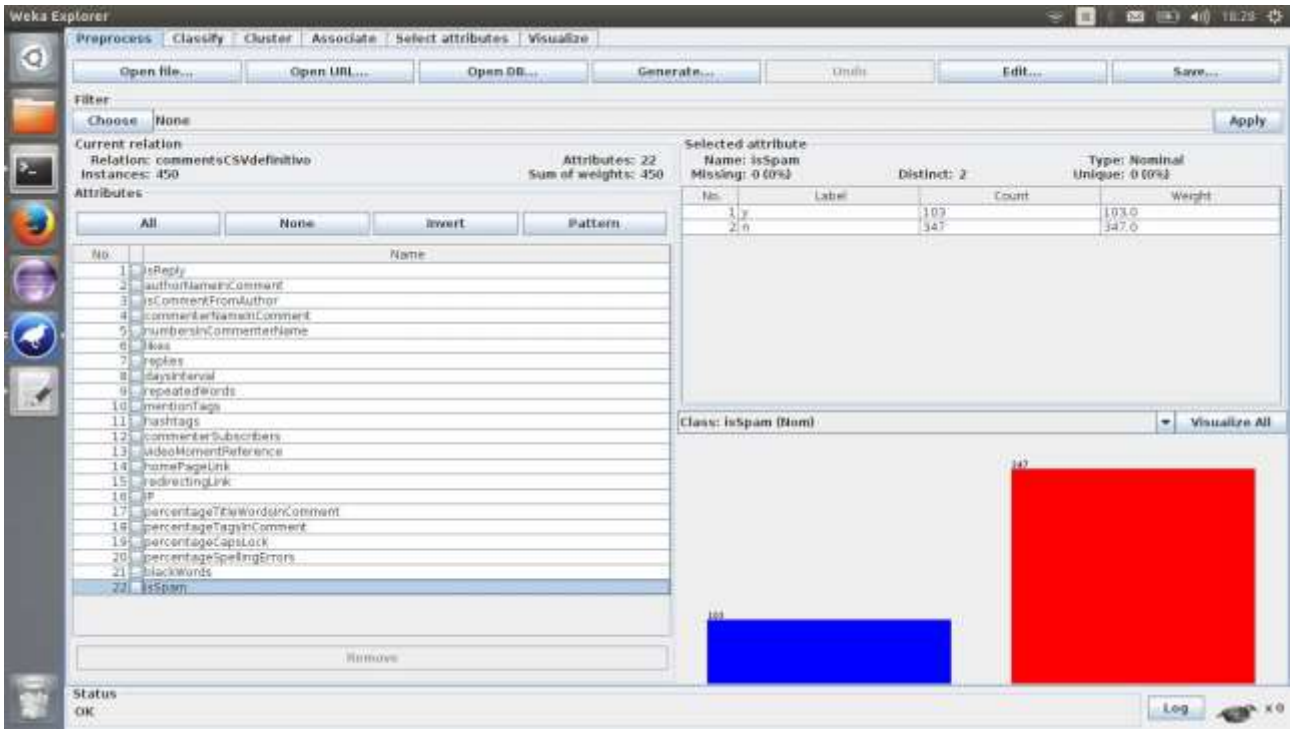
5NN: accuratezza 83%. Errore assoluto relativo 64%. Riconosce lo spam nella metà dei casi e il non spam nel 92%.

Commento generale
Con la tecnica Cross Validation i risultati sono meno buoni che usando l'intero training set, perché il sistema si addestra solo su una parte del dataset. Gli algoritmi migliori sono J48, 3NN e 5NN che riconoscono bene i commenti non spam, e quelli spam nella metà dei casi. I risultati quindi non sono ancora sufficienti.

Problema del dataset sbilanciato

Un problema che si è posto durante il progetto è stato quello della costruzione del dataset, per due motivi principali: il reperimento di commenti contenenti spam, e la loro etichettatura.
Le difficoltà relative al primo caso riguardano il fatto che lo spam non è facilmente reperibile su YouTube, sia perché è la piattaforma stessa a rimuoverlo, sia perché gli autori dei video gestiscono la propria sezione commenti, rimuovendo messaggi indesiderati.
Le difficoltà relative al secondo caso riguardano il fatto che le API di YouTube non offrono un'etichettatura ben implementata, o comunque completa, dei commenti (l'attributo *moderationStatus*, che dovrebbe indicare se un commento è stato etichettato come spam, da tutti i commenti che sono stati estratti durante il progetto, è sempre risultato con valore null).
Si è dovuto perciò eseguire un'analisi e etichettatura a mano dei commenti estratti, piuttosto onerosa a causa della poca frequenza dei commenti spam.
Si è quindi ottenuto un dataset contenente quasi 500 commenti, di cui circa 100 spam.
Questo dataset è quindi sbilanciato, con un rapporto di circa 4:1. Se il dataset è piccolo, come in

questo caso, tale sproporzione può provocare un cattivo addestramento del classificatore, mentre per dataset dell'ordine di oltre 10.000 istanze la differenza è sempre meno percepita.

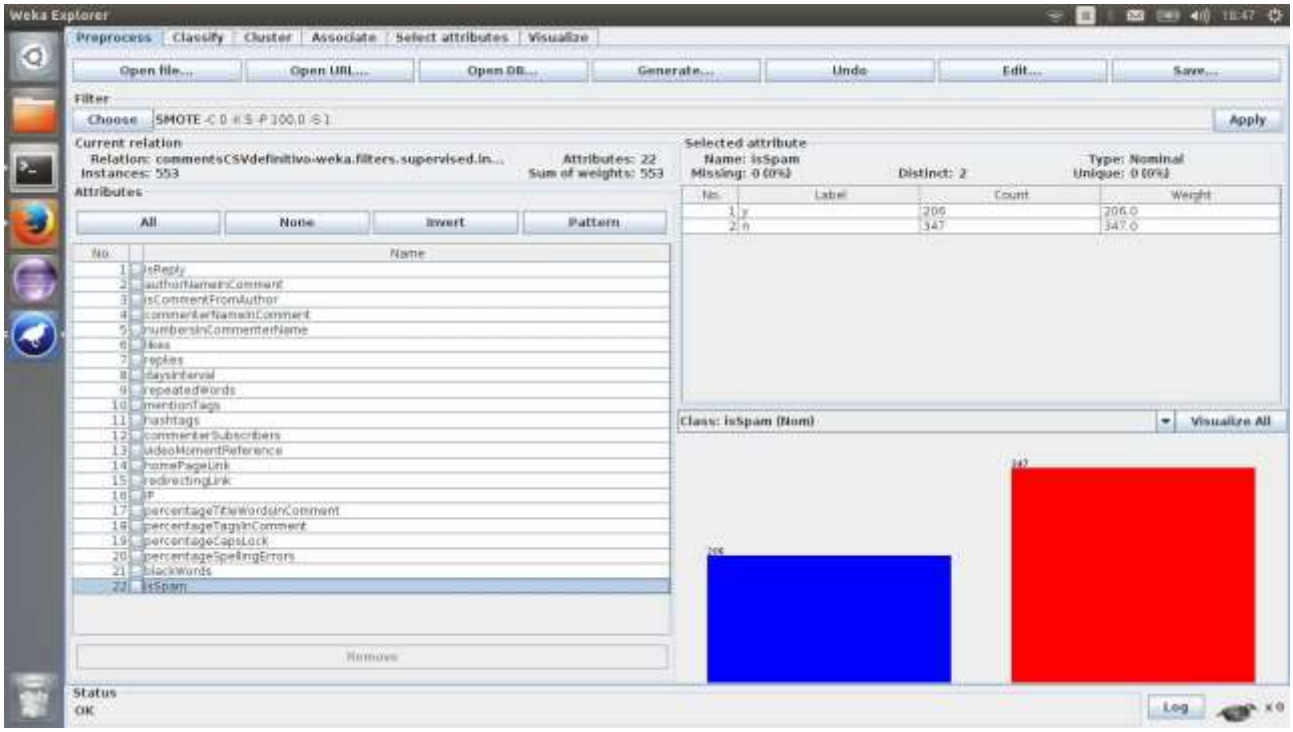


Casi come questo sbilanciamento sono comuni nella classificazione e vi son diversi metodi per porvi rimedio. In particolare, se non è possibile recuperare dati più numerosi, sono possibili due approcci:

- *Over-sampling*: aggiungere copie delle istanze della classe meno numerosa, o istanze con attributi simili, in modo da incrementarne la cardinalità.
- *Under-sampling*: eliminare parte delle istanze della classe più numerosa, in modo da diminuirne la cardinalità.

Per evitare di ridurre eccessivamente le dimensioni del dataset, si è scelto il primo approccio. È stato possibile effettuare efficientemente l'over-sampling grazie a un apposito filtro di Weka, denominato SMOTE.

Primo utilizzo del filtro SMOTE



J48: accuratezza 83%. Errore assoluto relativo 42%. Riconosce lo spam nel 75% dei casi e il non spam nell'88%.

NaiveBayes: accuratezza 53%. Errore assoluto relativo 97%. Riconosce lo spam nel 92% dei casi e il non spam nel 30%.

BayesNet: accuratezza 84%. Errore assoluto relativo 49%. Riconosce lo spam nel 72% dei casi e il non spam nel 91%.

1NN: accuratezza 80%. Errore assoluto relativo 42%. Riconosce lo spam nel 76% dei casi e il non spam nell'82%.

3NN: accuratezza 82%. Errore assoluto relativo 46%. Riconosce lo spam nel 76% dei casi e il non spam nell'86%.

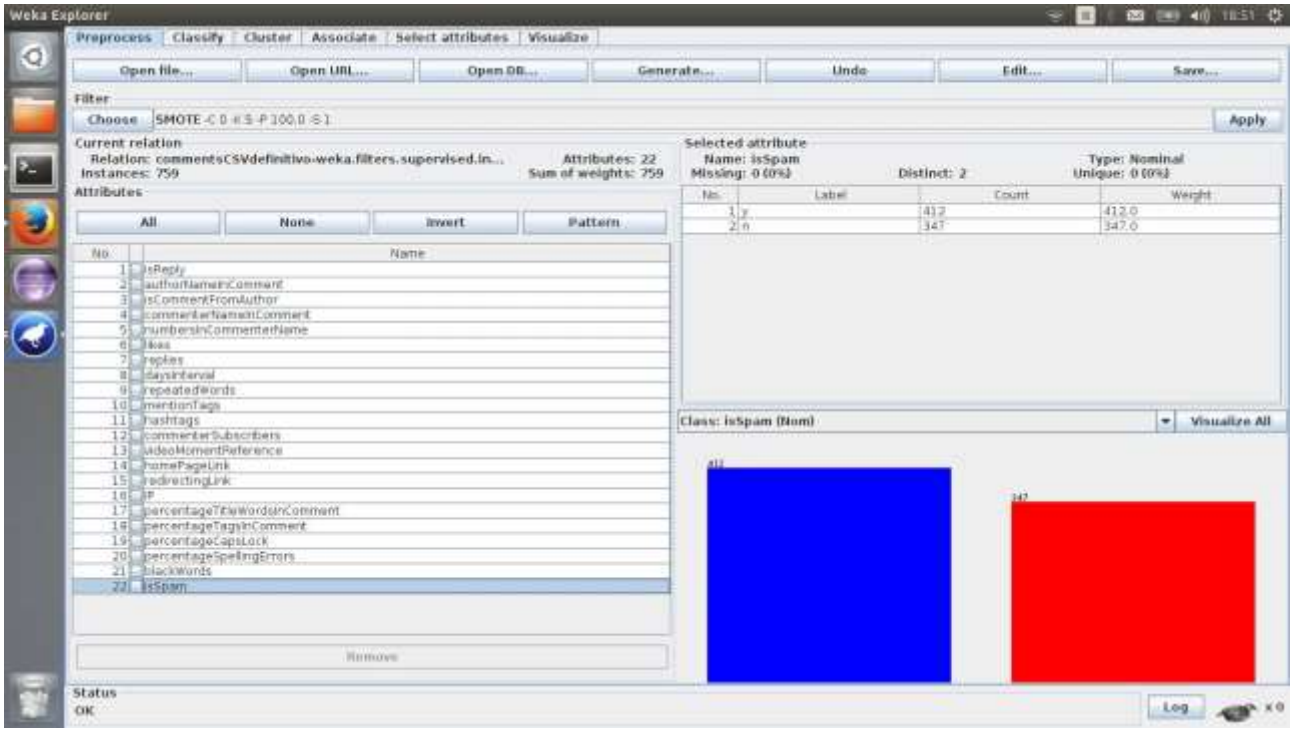
5NN: accuratezza 83%. Errore assoluto relativo 48%. Riconosce lo spam nel 73% dei casi e il non spam nell'89%.

Commento generale
In generale grazie al filtro SMOTE, e quindi con un dataset più bilanciato, i risultati osservati migliorano considerevolmente. Si può osservare che tutti gli algoritmi migliorano nella rilevazione positiva di spam, la classe che prima era meno rappresentata nel dataset. Vi è un leggero peggioramento nel riconoscimento di commenti non spam, ma nonostante questo l'errore assoluto relativo diminuisce in tutti i casi, anche di molto. Questo perché i miglioramenti sono molto maggiori dei peggioramenti.

Osservando nel dettaglio si nota che tuttavia NaiveBayes fornisce ancora risultati insufficienti, in quanto tende a catalogare come spam molti commenti che non lo sono. Gli altri algoritmi hanno risultati simili tra loro e tali che possono essere utilizzati in piattaforme come YouTube, dove

classificare erroneamente commenti come spam è più grave che non riconoscere lo spam, mantenendo comunque sufficiente accuratezza.

Secondo utilizzo del filtro SMOTE



J48: accuratezza 87%. Errore assoluto relativo 31%. Riconosce lo spam nel 88% dei casi e il non spam nell'87%.

NaiveBayes: accuratezza 62%. Errore assoluto relativo 74%. Riconosce lo spam nel 94% dei casi e il non spam nel 24%.

BayesNet: accuratezza 86%. Errore assoluto relativo 33%. Riconosce lo spam nel 82% dei casi e il non spam nel 92%.

1NN: accuratezza 87%. Errore assoluto relativo 26%. Riconosce lo spam nel 90% dei casi e il non spam nell'83%.

3NN: accuratezza 85%. Errore assoluto relativo 34%. Riconosce lo spam nel 89% dei casi e il non spam nell'81%.

5NN: accuratezza 82%. Errore assoluto relativo 40%. Riconosce lo spam nel 86% dei casi e il non spam nell'78%.

Commento generale
In generale applicando SMOTE gli algoritmi migliorano, alcuni di più, altri meno, mantenendo un livello dei risultati comunque accettabile. Questo perché un dataset più equilibrato permette un addestramento migliore. Si inizia inoltre a notare un diverso comportamento tra i vari algoritmi.

J48 è rimasto stabile nel riconoscere i commenti non spam ed è migliorato nel riconoscimento dello spam (+13%).

NaiveBayes rimane ancora non sufficiente, in particolare tende a catalogare tutti i commenti come spam.

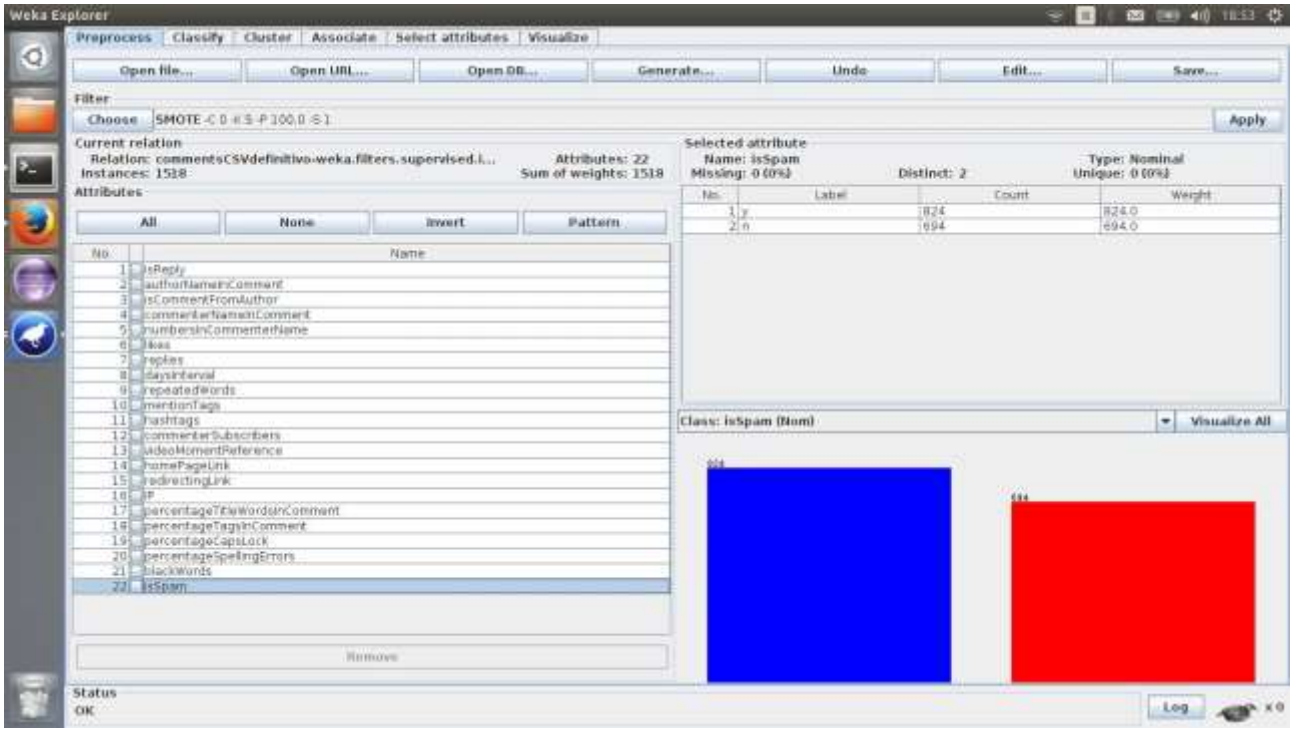
BayesNet è quello che dà i risultati migliori, catalogando ottimamente i commenti non spam e in modo molto buono quelli spam (+ 10%).

1NN migliora molto nel riconoscere lo spam (+14%) e di molto poco anche per il non spam (+1%).

3NN e 5NN sono migliorati sensibilmente nel riconoscimento dello spam (+13% ciascuno) ma hanno subito un lieve peggioramento nel riconoscimento del non spam (rispettivamente 5% e 11%).

Gli algoritmi migliori da usare nella pratica sono dunque J48 e BayesNet. Da notare che 3NN e 5NN in questo caso marcano come spam commenti che non lo sono (una volta su cinque circa), per cui su YouTube potrebbero non essere ottimali.

Quarto utilizzo del filtro SMOTE



J48: accuratezza 91%. Errore assoluto relativo 22%. Riconosce lo spam nel 93% dei casi e il non spam nell'89%.

NaiveBayes: accuratezza 62%. Errore assoluto relativo 74%. Riconosce lo spam nel 94% dei casi e il non spam nel 25%.

BayesNet: accuratezza 84%. Errore assoluto relativo 39%. Riconosce lo spam nel 84% dei casi e il non spam nell'83%.

1NN: accuratezza 91%. Errore assoluto relativo 18%. Riconosce lo spam nel 93% dei casi e il non spam nell'88%.

3NN: accuratezza 90%. Errore assoluto relativo 23%. Riconosce lo spam nel 94% dei casi e il non spam nell'86%.

5NN: accuratezza 88%. Errore assoluto relativo 27%. Riconosce lo spam nel 91% dei casi e il non spam nell'85%.

Commento generale
J48 ha delle metriche con valori che si aggirano intorno al 90%.

NaiveBayes non ha subito cambiamenti rispetto al caso precedente ed è rimasto stabile.

BayesNet è peggiorato nel riconoscere i commenti non spam.

1NN è migliorato in ogni parametro, e si osserva che ha l'errore assoluto relativo minore di tutti.

3NN e 5NN sono migliorati discretamente e tra i due si osserva che il primo dà risultati migliori.

Aumentando le dimensioni del dataset si nota quindi che si possono usare nella pratica quasi tutti gli algoritmi, tranne NaiveBayes. Inoltre è da notare che BayesNet, avendo subito dei peggioramenti, potrebbe non essere ottimale rispetto a come si era pensato inizialmente.

Feature selection

Abbiamo analizzato come varia l'*info gain* in base al numero di istanze del dataset.

La feature 'IP' ha sempre guadagno 0 perché nel dataset l'attributo ha sempre valore 'false'. Questo perché è molto raro trovare indirizzi IP in un commento.

Al contrario si può notare che al variare del dataset ci sono alcune feature che rientrano sempre tra le prime 10. Queste, in ordine di *info gain* medio, sono 'blackWords', 'percentageCapsLock', 'mentionTags', 'videoMomentReference' e 'percentageTagsInComment'.

Si va ora a analizzare l'efficacia degli algoritmi selezionando solo alcune feature con diversi criteri:

- le prime 10 in ordine di guadagno;
- le feature con *info gain* maggiore di una certa threshold;
- eliminando feature potenzialmente dipendenti tra loro.

Poiché gli algoritmi sono più efficaci dopo la quarta applicazione di SMOTE, è stato ritenuto più interessante applicare la selezione in questo caso. Inoltre la seguente analisi è limitata all'algoritmo NaiveBayes, perché si è notato che gli altri non producevano cambiamenti apprezzabili.

Le prime 10 in ordine di guadagno

NaiveBayes: accuratezza 62%. Errore assoluto relativo 75%. Riconosce lo spam nel 94% dei casi e il non spam nel 25%.

Threshold > 0.05

NaiveBayes: accuratezza 65%. Errore assoluto relativo 71%. Riconosce lo spam nel 91% dei casi e il non spam nel 33%.

Threshold ≥ 0.05

NaiveBayes: accuratezza 59%. Errore assoluto relativo 78%. Riconosce lo spam nel 94% dei casi e il non spam nel 17%.

Threshold ≥ 0.01

NaiveBayes: accuratezza 62%. Errore assoluto relativo 74%. Riconosce lo spam nel 94% dei casi e il non spam nel 25%.

NaiveBayes con una threshold alta ha avuto un lieve miglioramento nei risultati.

Per quanto riguarda i miglioramenti di efficienza invece, questi non sono stati apprezzabili in quanto il dataset ha dimensioni ridotte.

Eliminazione feature potenzialmente dipendenti

Tra le feature che potevano essere considerate dipendenti tra loro sono state eliminate quelle con *info gain* minore. Questa modifica non ha comunque portato cambiamenti apprezzabili nei risultati che sono rimasti sostanzialmente invariati.

Caso dataset di grandi dimensioni

Per completezza sono stati provati gli algoritmi con un dataset molto maggiore (circa 12000 istanze) ottenuto dall'applicazione ripetuta del filtro SMOTE. Ogni algoritmo ha dato dei risultati che oscillavano tra il 97% e il 98%, con l'eccezione di NaiveBayes e BayesNet, rimasti sostanzialmente invariati.

Caso Naive Bayes con dataset perfettamente bilanciato

Data la scarsa efficacia di classificazione dell'algoritmo NaiveBayes si è approfondito lo studio relativo a questo algoritmo.
Si è notato che avendo le istanze con attributi di classe perfettamente equilibrati i risultati sono visibilmente migliorati, diventando paragonabili a quelli degli altri algoritmi presi in considerazione. Aumentando il numero di istanze con SMOTE tuttavia i risultati non migliorano ulteriormente quanto gli altri algoritmi. Ciò può significare che l'over-sampling è meno efficace dell'under-sampling sul classificatore bayesiano analizzato. Questo perché bilanciare il dataset con l'under-sampling ha portato risultati migliori rispetto al bilanciamento dovuto all'over-sampling automatizzato con il filtro SMOTE. Per approfondire al meglio l'analisi si ritiene necessario

aumentare la dimensione del dataset e la varietà dei commenti estraendo e classificandone di nuovi.

Conclusioni e sviluppi futuri

Durante il progetto è stato sviluppato un prototipo di applicazione per l'estrazione dei commenti di YouTube, con elaborazione di feature rilevanti dal punto di vista della spam detection, e del relativo classificatore.

Per la realizzazione dell'estrazione dei commenti ed elaborazione delle feature sono stati utilizzati:

- le API di YouTube versione 3.0 fornite da Google;
- la API di Google per la rappresentazione delle date;
- la libreria Jazzy per la verifica di correttezza dei vocaboli.

I test di classificazione hanno dato buoni risultati, soprattutto con gli algoritmi J48 e K-NN.

Per quanto riguarda invece NaiveBayes i risultati sono sufficienti ma possono ancora essere migliorati. Essendo questo un algoritmo rilevante, sia per importanza che per utilizzo, per la classificazione, è probabilmente necessario migliorare l'implementazione delle feature e modificarne parzialmente il pool (nonostante quello attuale sia ritenuto completo e sufficientemente vario). Inoltre si ritiene necessario incrementare le dimensioni del dataset, estraendo ed etichettando nuovi commenti.