Spambase

Report per l'Esame di Fondameti di Machine Learning

Marco Moroni

129056
Ingegneria informatica
255699@studenti.unimore.it

Giugno 2021

Indice

Abstract

In questo progetto si andrà a trattare ed analizzare il problema delle email spam che oggigiorno è sempre più diffuso e nocivo, soprattutto per le grandi aziende. Grazie al dataset fornito e alle dovute analisi compiute su di esso si è riuscito a catalogare le email e predirre in modo pressochè perfetto la natura di una mail (veritiera o spam).

1 Introduzione

Con Spam si intende l'invio o la ricezione di messaggi pubblicitari di posta elettronica che non sono stati richiesti e che l'utente che li riceve non ha autorizzato il mittente ad inviare.

Obiettivo degli spammer è la pubblicità: comuni offerte commerciali, proposte di vendita di materiale pornografico o illegale, farmaci senza prescrizione medica. Il loro scopo è quello di carpire dati personali, login e password di utenti, numeri di carte di credito e di conto corrente ecc...

Gli spammer sono, a tutti gli effetti, dei criminali. Ad esempio quelli che inviano mail simili alla famosa truffa "alla nigeriana" dove una persona che non conosci ti chiede aiuto per sbloccare enormi quantità di denaro e ti propone di dividere parte di quel denaro, ma ti chiede, per fare questa operazione, una serie di dati bancari che sono il vero e unico obiettivo della mail.

Lo Spam è anche strumento di truffa, ti propone improbabili progetti finanziari, cerca di farti credere che hai fatto una favolosa vincita di denaro o sei stato designato erede di grandi fortune e per mandarti questo fiume di denaro chiede le credenziali di accesso al tuo conto corrente online.

Gli spammers inviano le mail pubblicitarie o truffaldine a migliaia di indirizzi, questi indirizzi sono raccolti in rete in molteplici modi:

- automaticamente indirizzi mail da pagine personali, blog, forum o newsgroup;
- usando appositi software che costruiscono gli indirizzi di mail usando nomi e cognomi comuni;
- pubblicando falsi siti web che catturano indirizzi, promettendo vantaggi e offerte mirabolanti;
- acquistando indirizzi mail da altri spammers.

Sorgono sempre più spesso notizie di grandi aziende, che per colpa di un dipendente non abbastanza istruito sul comportamento coretto da adottare di fronte ad una email non sicura, si ritrovano a dover sborsare migliaia di dollari da pagare in cryptovalute.

Motivo per il quale negli ultimi anni sempre più persone si stanno istruendo, e stanno istruendo gli altri sull'identificare 'a vista d'occhio' una mail spam, o comunque una mail sospetta.
I vari servizi anti-spam che mettono a disposizione i diversi provider di mail (come Gmail, Outlook, Protonmail, Tutanota, Yahoo,...) il più delle volte non sono sufficienti a fermare i criminali che si nascondono dietro una banalissima ed innoqua email inviata dal 'cuggino di 4^{o} grado con un patrimonio milionario bloccato alle isole mauritius'.

2 Analisi dei dati

In statistica, l'analisi esplorativa del dati (EDA - exploratory data analysis) è un approccio di analisi del set di dati per riassumere le loro caratteristiche principali, spesso usando grafici statistici e altri metodi di visualizzazione dei dati. Un modello statistico può essere usato o meno, ma principalmente l'EDA serve a vedere 'cosa i dati possono dirci'.L'EDA è diversa dall'analisi iniziale dei dati (IDA), che si concentra più strettamente sulla verifica delle ipotesi richieste per l'adattamento del modello e la verifica delle ipotesi, e sulla gestione dei valori mancanti e sulle trasformazioni delle variabili secondo necessità.

2.1 Strumenti utilizzati

Per questo lavoro ho trovato necessario l'utilizzo di alcune librerie open-source, fondamentali per l'analisi dei dati e per semplificare il lavoro dell'analista:

- **pandas**: potente e veloce libreria nata per analizzare e manipolare i dati in input, il suo nome deriva dal gioco di parole "python data analysis";
- **pyplot**: sottolibreria di matpolotlib che rende possibile la creazione di grafici, figure e linee, e permette inoltre di poter cambiare colori, forma e dimensioni.
- **sklearn**: è una libreria di apprendimento automatico che contiene al suo interno algoritmi di ogni tipo;
- **seaborn**: libreria basata su matpolotlib, che come pyplot permette la creazione di grafici;
- **numpy**: libreria basata su python e c++ fondamentale per lavorare con array di grandi dimensioni. A differenza delle liste di python, numpy salva i dati in memoria in maniera contigua, l'accesso, incrementando notevolmente le performance, stimate al 5000%;

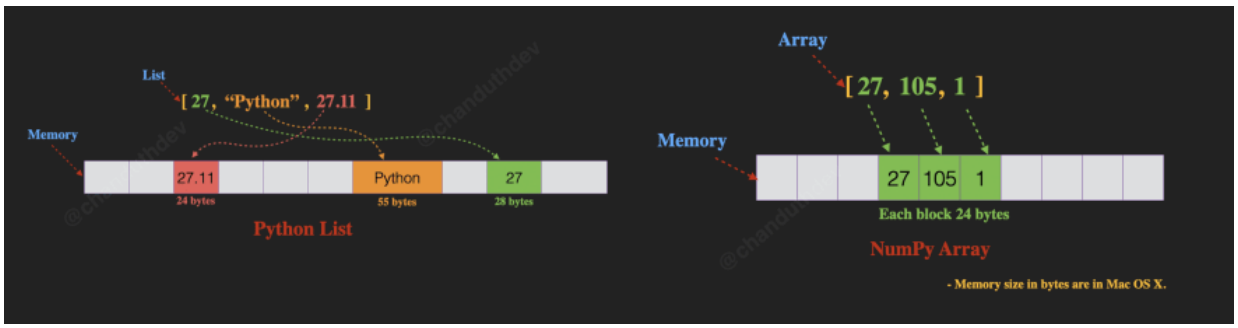


Figure 1

2.2 Analisi inziale

Per prima cosa è necessario capire con che tipo di dati si andrà a lavorare, che siano numeri interi o decimali, stringhe o booleani, è fondamentale saperlo prima di cominciare l'analisi. Per fare ciò la libreria *pandas* viene in supporto fornendo delle semplici funzioni ad-hoc, come ***pandas.info()*** che permette di ricavare il numero totale di righe e colonne, e per ogni attributo, specifica il tipo di dato della colonna. Un'altra funzione utile è

```
RangeIndex: 4601 entries, 0 to 4600
Data columns (total 58 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   word_freq_make         4601 non-null   float64
 1   word_freq_address      4601 non-null   float64
 2   word_freq_all          4601 non-null   float64
 3   word_freq_3d           4601 non-null   float64
 4   word_freq_our          4601 non-null   float64
 5   word_freq_over         4601 non-null   float64
 6   word_freq_remove       4601 non-null   float64
 7   word_freq_internet     4601 non-null   float64
 8   word_freq_order        4601 non-null   float64
 9   word_freq_mail         4601 non-null   float64
```

Figure 2

pandas.describe() che per ogni colonna permette di ricavare dati come il numero di istanze non nulle, la media, la deviazione standard, il massimo e il minimo.

	word_freq_make	word_freq_address	word_freq_all	word_freq_3d	word_freq_our	word_freq_over	word_freq_remove
count	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000
mean	0.104553	0.213015	0.280656	0.065425	0.312223	0.095901	0.114208
std	0.305358	1.290575	0.504143	1.395151	0.672513	0.273824	0.391441
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.420000	0.000000	0.380000	0.000000	0.000000
max	4.540000	14.280000	5.100000	42.810000	10.000000	5.880000	7.270000

Figure 3

Il dataset è composto da 4601 istanze di email e 57 attributi rappresentati da un numero reale e un attributo classe rappresentato da un booleano che indica appunto se il sample è uno spam (1) o non spam (0). Il dataset spambase in questione non è bilanciato, difatti le classi spam e non spam corrispondono rispettivamente al 39.4% e 60.6% dei samples raccolti.

2.3 Correlazione

La matrice di correlazione è una tabella che mostra come gli attributi siano effettivamente legati tra loro. Quando due attributi sono tra di loro correlati, al crescere dell'uno cresce anche l'altro.

Il coefficiente di correlazione è un numero, compreso tra -1 e + 1 che esprime il grado di dipendenza tra due attributi, ed in base al valore che assume si può dedurre che se il coefficente è:

- == 1: i due attributi hanno la stessa curva, al crescere dell'uno cresce anche l'altro;
- == -1: i due attributi sono perfettamente decorrelati con una curva inversamente proporzionale, al crescere del primo il secondo decresce;
- == 0: i due attributi sono indipendenti l'uno dall'altro.

Si può notare dal grafico in figura 4 che la diagonale è composta da tutti 1, dato che un attributo messo in relazione con se stesso seguirà la stessa curva.

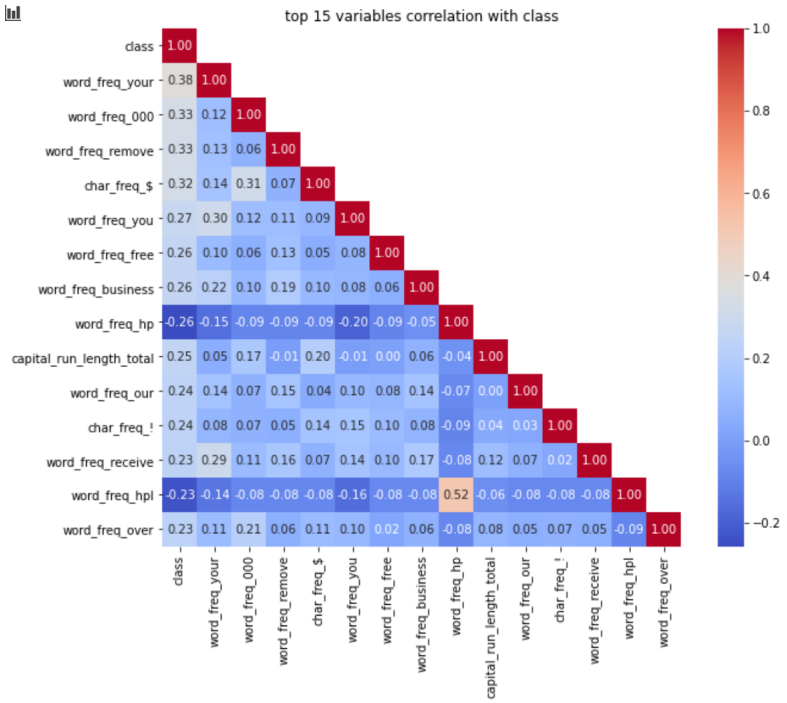


Figure 4

Il grafico di correlazione riesce a mettere in evidenza gli attributi più collegati con la classe (spam o non spam). In ordine, si notano le parole "your", "0000", "remove", "$", "you", "free", "business", "hp", "capital_run_lenght_total" (numero di lettere maiuscole), "our", "!", "receive", "hpl", "over".

Raffigurati ci sono i primi 15 attributi più collegati alla classe spam, anche se non si nota una stretta coerelazione, dato che i valori si muovono tra 0.23 e 0.38.

Si va quindi a restringere il campo per analizzare, sempre graficamente, la corrispondenza tra i cinque attributi più correlati. Si denota facilmente, dalla figura 5, che la parola

”your” è più presente nelle email non spam, mentre le parole ”000”, ”remove”, ”$” e ”you” sono nettamente più frequenti nelle email spam.

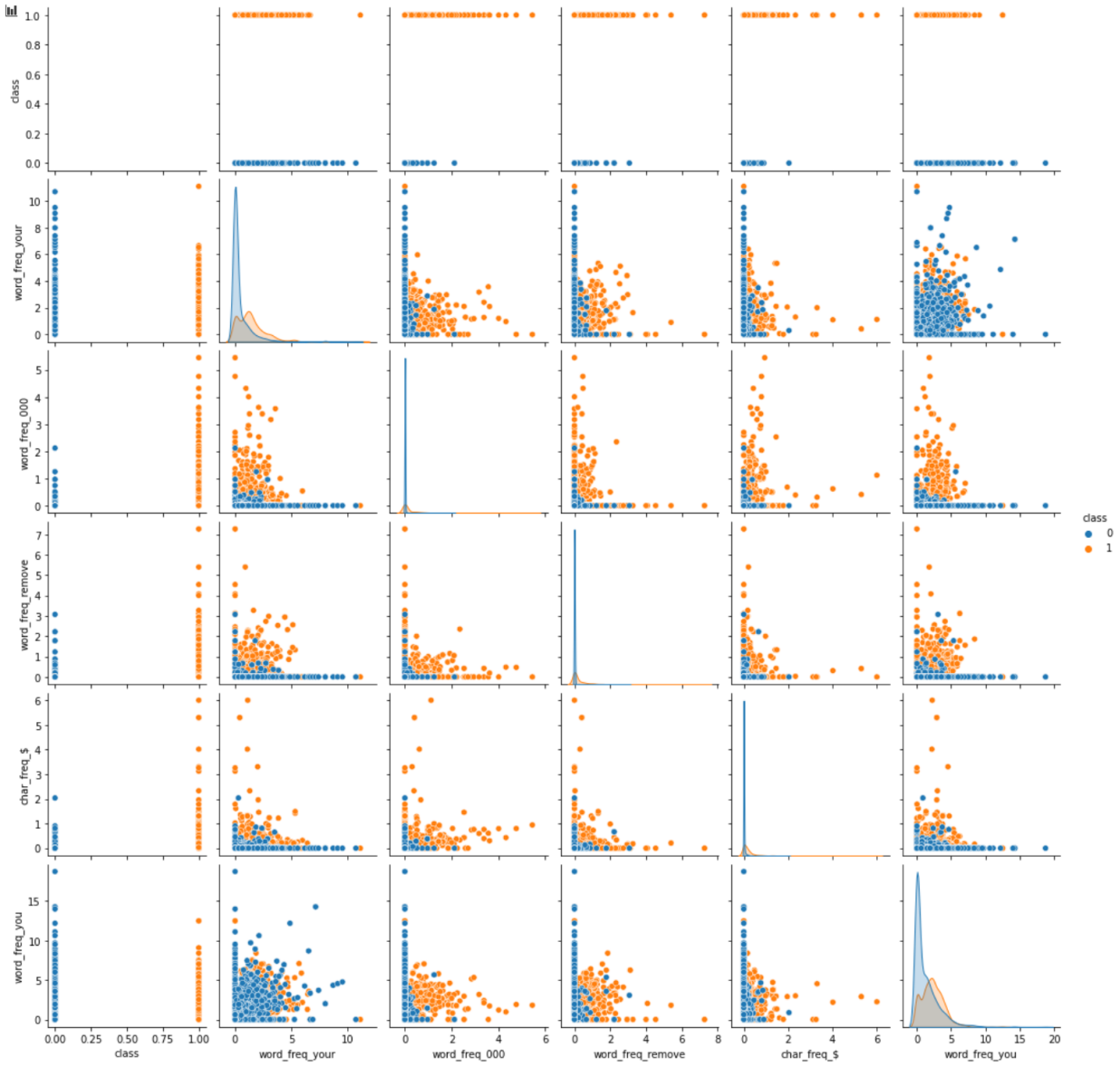


Figure 5

2.4 Comparazione parole

In questa sezione si andranno ad analizzare quali saranno le parole effettive da tenere maggiormente in considerazione tramite 3 grafici esplicativi.

Il grafico in figura 6 rappresenta la percentuale della frequenza di ogni singola parola trovata nelle email che sono state poi selezionate come spam. In questo grafico non si tiene conto dell'influenza delle email non spam. Il grafico conferma le analisi svolte nei capitoli precedenti, questa volta calcolando solo l'impatto delle email non spam. Le due parole più inserite sono sempre "you" e "your".

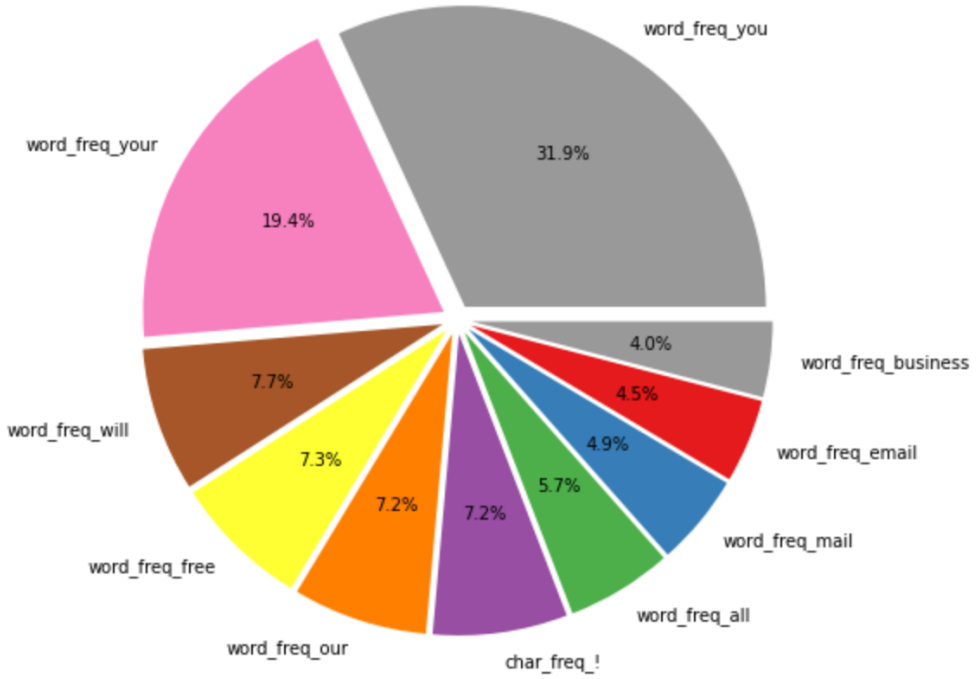


Figure 6

Per eseguire un'analisi più scrupolosa si va invece a calcolare la differenza delle frequenze tra le email non spam e le email spam. Per fare ciò è prima necessario creare una tabella pivot che calcoli, per entrambe le classi, la media di attributo, restituendo una tabella (figura 7) che abbia sulle ascisse le parole chiave e sulle ordinate la classe di appartenenza; i valori corrispondenti saranno la media delle frequenze della parola X nella rispettiva classe di appartenenza Y.

class	char_freq_!	char_freq_#	char_freq_$	char_freq_(	char_freq_;	char_freq_[	word_freq_000	word_freq_1999	word_freq_3d
0	0.109984	0.021713	0.011648	0.158578	0.050281	0.022684	0.007088	0.197744	0.000886
1	0.513713	0.078877	0.174478	0.108970	0.020573	0.008199	0.247055	0.043469	0.164672

Figure 7

Una volta calcolata la tabella pivot si notano immediatamente, dal grafico in figura 8, le parole che hanno un maggiore impatto sulla scelta della classe di appartenenza. In base al valore che ogni barra assume sulle y si determina che se:

- y == 0: la parola è equamente presente sia nelle email spam che non spam;
- y < 0: la parola è maggiormente presente nelle email spam;
- y > 0: la parola è maggiormente presente nelle email non spam.

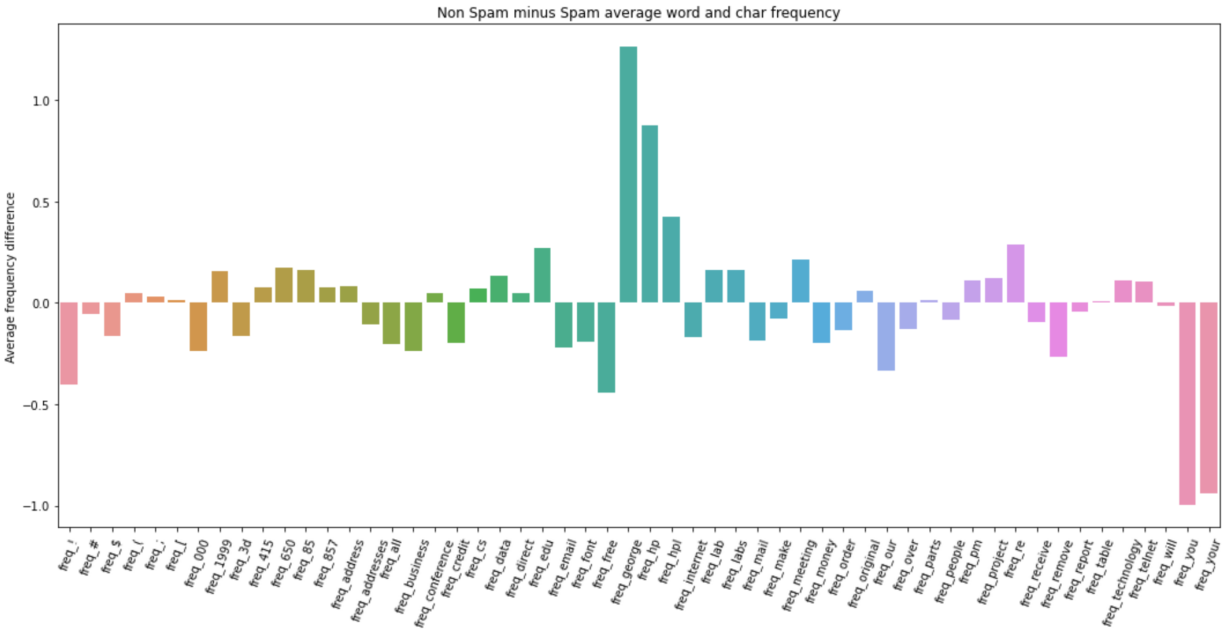


Figure 8

Spiccano all'occhio, quindi, che le parole "you" e "your" sono nettamente più frequenti nelle email spam, mentre le parole come "george" e "hp" sono maggiormente più frequenti nelle email vere. Questo perchè spesso, il malintenzionato che spedisce la mail malevola, non conosce il nome delle proprie vittime e si rivolge a loro con il pronome "you" ("tu", in inglese), piuttosto che con il loro vero nome.

2.5 Lettere maiuscole

Nel dataset, oltre alla frequenza delle singole parole, sono presenti tre campi aggiuntivi:

- capital_run_lenght_average: indica la media delle lunghezze di ininterrote sequenze di lettere maiuscole;
- capital_run_length_longest: indica la sequenza di lettere maiuscole più lunga;
- capital_run_length_total: indica la somma totale delle lettere maiuscole.

Analizzando questi tre fattori presenti all'interno della email, si deduce, leggendo il grafico a figura 9 con le stesse valutazioni del grafico a figura 8, che nelle email spam è nettamente maggiore il numero di lettere maiuscole inserite, che nelle email non spam. Il malintenzionato preferisce, quindi, utilizzare le lettere maiuscole per carpire l'attenzione della vittima.

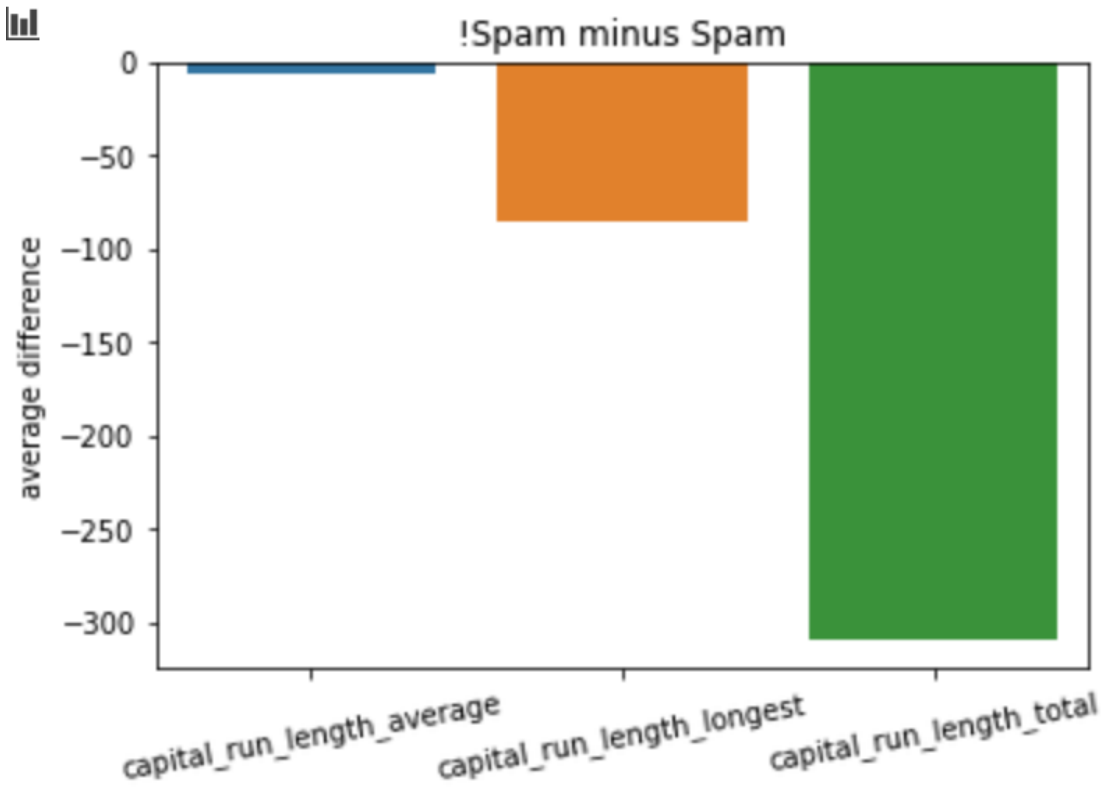


Figure 9

3 Discussione dei modelli

In questo capitolo si andranno ad analizzare gli algoritmi utilizzati per prevedere la natura di una mail.

Il problema in questione fa chiaramente riferimento ad un modello di classificazione binaria, poichè ogni sample è rappresentato da un numero intero e può assumere solamente due valori:

- 0: non spam;
- 1: spam.

Per questo progetto ho selezionato, tra gli algoritmi svolti durante le lezioni, tre che ho reputato più adatti per il riconoscimento, che si andranno ad esaminare più approfonditamente nelle prossime sezioni:

- K-Nearest Neighbors;
- Alberi decisionali;
- Support Vector Machine (SVM)

3.1 K-Nearest Neighbors

K-Nearest Neighbors è un algoritmo che si basa sulle caratteristiche dei K-vicini (dove K è un numero intero positivo che indica il numero di vicini da prendere in considerazione) per determinare la classe di appartenenza di ogni sample. In questo tipo di classificazione, il risultato è l'appartenenza ad una certa classe.

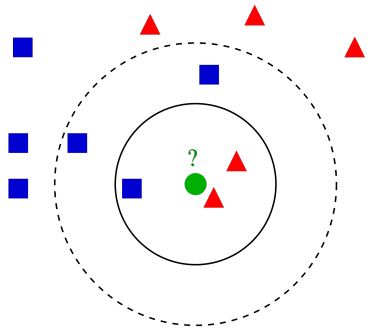


Figure 10

Questo algoritmo, quindi descrive la similarità tra i diversi sample, in base al numero di vicini che vengono presi in considerazione. Per esempio, per prevedere la classe della pallina verde in figura 10 la scelta di un k diverso cambia il risultato:

- k == 3: si considerano i tre oggetti più vicini e si tiene conto del maggiore, 2 triangoli rossi e 1 quadrato blu, quindi la pallina verde viene associata alla stessa classe dei triangoli rossi, in quanto numero maggiore dei quadrati blu;
- k == 5: ragionamento analogo, si contano 3 quadrati blu e 2 triangoli rossi, la pallina verde viene quindi associata alla stessa classe dei quadrati blu.

La scelta del valore k può quindi rovesciare la previsione dell'intero algoritmo. Solitamente k è un numero non molto grande, e per una classificazione binaria è preferibile un k dispari, per evitare di ritrovarsi in situazioni di parità (3 quadrati blu e 3 triangoli rossi).

Questo tipo di algoritmo è molto influenzato dall'uniformità dei dati ed il dataset preso in considerazione non è bilanciato (60% non spam e 40% spam), per ciò per il seguente algoritmo i parametri scelti per ottimizzare al massimo l'algoritmo sono stati:

- **weight** = "distance": nell'esempio di prima dei triangoli, ogni elemento aveva lo stesso peso nella conta; utilizzando questo parametro, gli elementi più vicni alla pallina verde avranno un peso maggiore, mentre quelli più distanti avranno un peso minore;
- **k = 14**: nonostante si tratti di una classificazione binaria, utilizzando la distanza pesata, perde di senso utilizzare per forza un numero dispari per evitare di imbattersi in condizioni di parità.

3.2 Alberi decisionali

Con alberi decisionali s'intende l'implementazione degli alberi binari, composti da:

- **nodi**: funzioni condizionali booleane che testano il valore dello specifico attributo;
- **rami**: collegamenti tra i nodi, rappresentano il risultato della condizione;
- **foglie**: una sottoclasse dei nodi, rappresentano il risultato finale della previsione di classificazione.

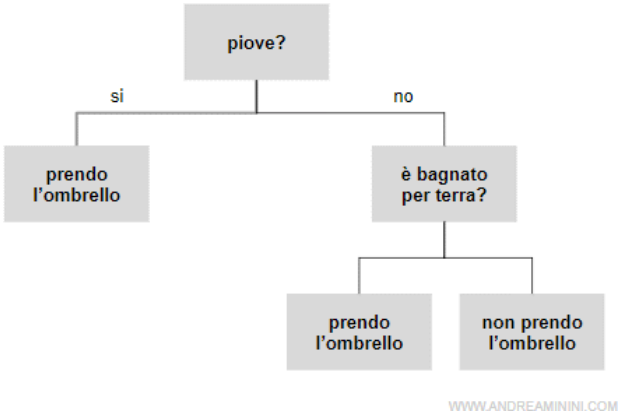


Figure 11

Una regola di suddivione è necessaria per decidere quale (o quali) variabile deve essere utilizzata nella condizione di un nodo per dividere il set di dati in sttogruppi, e soprattutto quale soglia utilizzare sulle variabili. Il metodo con cui vengono effettuate queste suddivisioni è basato sull'impurità dei nodi.

Si definisce "nodo impuro" un nodo che ha come risultato più classi diverse, mentre un "nodo puro" è un nodo che ha una sola classe come risultato. L'impurità in un nodo è quindi:

- **massima** quando tutte le classi nel nodo (in questo caso due) sono presenti in maniera casuale;
- **minima** quando nel nodo è presente una sola classe di appartenenza.

L'indice di Gini e l'entropia sono due criteri per calcolare l'impurità di un nodo, quindi la probabilità che una particolare variabile sia classificata in modo errato quando scelta casualmente.

Per il seguente algoritmo i parametri scelti per ottimizzare al massimo l'algoritmo sono stati:

- **l'indice di Gini**: varia tra 0 e 1, come si è discusso precedentemente, quando il valore è massimo (1) gli elementi son distribuiti casualmente nelle classi, minimo (0) quando tutti gli elementi appartengono alla stessa classe, e medio (0.5) quando gli elementi sono equamente distribuiti tra le due classi;

- **profondità massima = 10**: cioè la il numero massimo di livelli che ci possono essere nella rappresentazione dell'albero (per esempio, nella figura 11, sono presenti 3 livelli, compreso il nodo radice).

Di default non c'è un limite alla profondità massima che un albero può raggiungere, ma così facendo, si rende la previsione soggetta ad *overfitting*.

L'*overfitting* è una condizione nella quale il modello creato si adatta perfettamente ai sample con cui ha eseguito "l'allenamento" cioè con quali istanze di email ha creato l'albero; ma non riesce poi ad adattarsi nello stesso modo con nuovi sample. Di norma, più si incrementa il numero massimo di profondità, più l'albero diventa complesso e più specifico a determinate condizioni, di conseguenza meno flessibile ed adattabile, soggetto quindi ad overfitting.

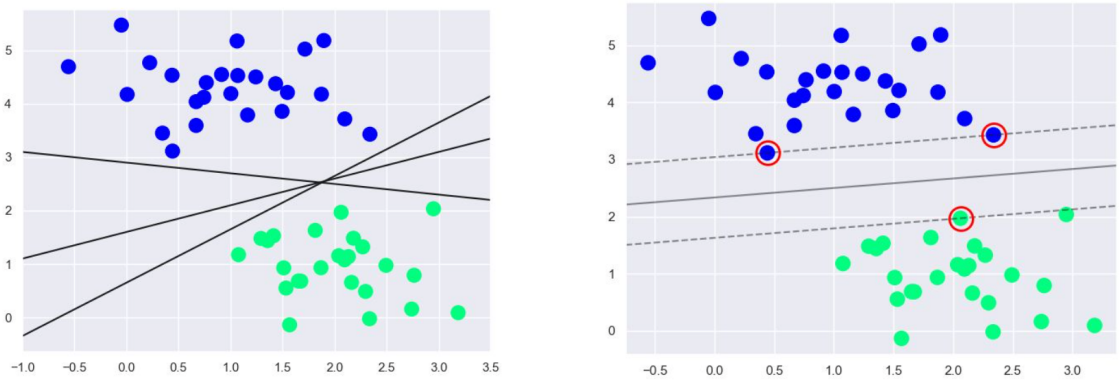
Può verificarsi anche la situazione opposta dove si imposta un numero massimo di livelli troppo basso, così da non lasciare possibilità al modello di creare le condizioni minime per classificare i sample e generalizzando troppo il risultato. Questa condizione specifica prende il nome di *underfitting*.

3.3 Support Vector Machine

Le macchine a vettori di supporto sono dei modelli di apprendimento supervisionato con l'obiettivo di identificare l'iperpiano linerare che meglio divide i sample in classi. In caso di classificazione con due sole dimensioni spaziali (x, y) l'iperpiano si riduce ad una riga, mentre a tre dimensioni viene rappresentato da un piano a tutti gli effetti, invece con più di tre dimensioni viene appunto definito *iperpiano*.

Si definiscono

- vettori di supporto: i punti più vicini all'iperpiano;
- margine: distanza tra i due vettori di supporto (di due classi differenti) più vicini all'iperpiano.



(a) (b)

Figure 12

L'algoritmo quindi disegna un numero n di linee per separare le palline blu da quelle verdi e calcola quale tra queste risulta ottimale per la separazione delle due classi.

Per modelli più complessi è necessario poter utilizzare, oltre che a linee rette, anche curve e regioni complesse nello spazio. Questo viene fatto attraverso un *kernel*, cioè una funzione che sposta i dati da un piano a bassa dimensione in uno spazio e dimensione superiore dove è possibile separarli tramite un piano.

Per questo algoritmo si è scelto di utilizzare un kernel **RBF** (radial basis function) che permette di creare regioni complesse nello spazio delle features.

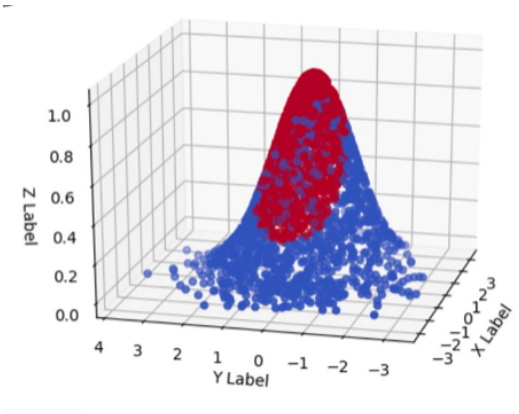


Figure 13

3.4 Ensamble

Con *ensamble* s'intende l'utilizzo modelli multipli per ottenere una migliore prestazione predittiva rispetto ai modelli singoli da cui è costituito. Ogni modello effettua la predizione della classe di appartenenza, e successivamente viene considerata valida la predizione fatta dal maggior numero di essi. Esistono tre tecniche differenti di ensamble:

- **bagging**: questa tecnica mira a creare un insieme di classificatori aventi la stessa importanza. All'atto della classificazione, ciascun modello voterà circa l'esito della predizione e l'output complessivo sarà la classe che avrà ricevuto il maggior numero di voti;
- **boosting**: a differenza del bagging, ciascun classificatore influisce sulla votazione finale con un certo peso. Tale peso sarà calcolato in base all'errore di accuratezza che ciascun modello commetterà in fase di learning;
- **stacking**: mentre nel bagging l'output era il risultato di una votazione, questa tecnica consente a un algoritmo di addestramento di raggruppare diverse altre previsioni di algoritmi di apprendimento simili.

Ho applicato questa la tecnica dell'ensamble all'algoritmo con i risultati migliori: gli alberi decisionali. L'algoritmo di ensamble degli alberi decisionali viene denominato **Random Forest**, che è un estensione del bagging. Questo algoritmo prende in esame n alberi diversi e, oltre a prendere un sottocampione diverso di dati per ogni albero, estrae anche le caratterstiche (attributi) in modo casuale, piuttosto che usare sempre tutte le caratteristiche per fare "crescere gli alberi".
L'algoritmo prende quindi il nome di Random Forest (Foresta casuale) dato che si hanno molteplici alberi casuali.

Utilizzando questa tecnica sull'algoritmo migliore, la previsione è incrementata di circa l'1%.

3.5 Scelta indicatori

Per l'analisi delle prestazioni di un algoritmo è necessario tener conto degli indicatori in base a diversi fattori, sia al tipo di classificazione che al dataset. Gli indicatori classici che

solitamente si vanno ad osservare sono:

- **accuracy**: indica il rapporto tra i valori corretti (sia positivi che negativi) sul totale dei sample

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

- **precision**: indica il rapporto tra le osservazioni positive correttamente predette e il totale delle osservazioni positive predette

$$precision = \frac{TP}{TP + FP} \tag{2}$$

- **recall**: indica il rapporto tra le osservazioni positive correttamente predette e tutte le osservazioni effettive della classe

$$recall = \frac{TP}{TP + FN} \tag{3}$$

- **f1 score**: indica la media ponderata tra precision e recall e prende quindi in considerazione sia i falsi positivi che i falsi negativi

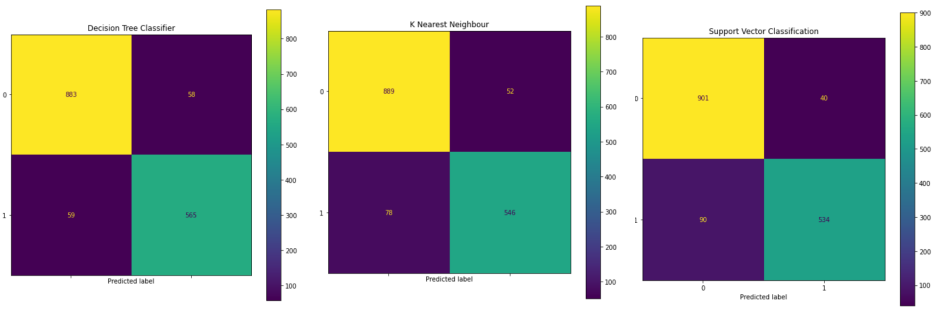
$$f1\_score = \frac{2 * Recall * Precision}{Recall + Precision} \tag{4}$$

In una classificazione binaria, come in questo caso, quando il dataset non è uniforme, cioè il numero di sample delle due classi non è simile ma si discosta di molto l'uno dall'altro come in questo caso (60% non spam, 40% spam) è necessario fare delle osservazioni.

Utilizzare l'accuracy sarebbe non totalmente corretto, perchè questo indice di misurazione è preciso solo se si sta lavorando con un dataset bilanciato (non è il caso in considerazione); eventualmente si potrebbe utilizzare l'*accuracy per classe* che va ad indicare l'accuratezza della previsione per ogni classe.

Per questo progetto si è scelto di utilizzare come indice l'**f1-score**, che da una rappresentazione più reale e corretta dei dati in questione. L'f1-score prende in considerazione sia i falsi negativi che i falsi positivi ed è meno intuitivo da interpretare, ma molto più adatto dell'accuracy quando il dataset non è uniforme.

Di seguito vegono riportate le *confusion matrix* per ogni algoritmo, che vanno a rappresentare graficamente il risultato di ogni classificatore binario.



(a) (b) (c)

Figure 14

4 Risultati

In questa sezione vengono esposti i risultati raggiunti e come poter facilmente testare il funzionamento dell'algoritmo.

Tutti i parametri precedentemente utilizzati nei modelli sono stati scelti tramite una *cross validation* a griglia. Sklearn dispone di una funzione chiamata *GridSearchCV* che esegue la previsione con lo stesso algoritmo, ma con diversi parametri (precedentemente impostati dall'utente) e calcola la combinazione di parametri che **meglio si adatta al dataset**. Questi metodi hanno un tempo di esecuzione ed un costo computazionale solitamente alto, motivo per il quale una volta calcolati, sono stati salvati nel codice, o come si dice nel gergo informatico, sono *"hard-coded"*.

I sample, prima di essere stati utilizzati negli agoritmi, sono stati normalizzati, per migliorare di gran lunga le previsioni. Vegnono rappresentate tre tabelle contenenti gli indicatori: la prima utilizzando i dati "crudi" senza nessun tipo di preprocessamento o miglioria dei parametri dei modelli; la seconda con i dati normalizzati; e la terza con sia il preprocessamento dei dati che i parametri discussi nel capitolo precedente.

Come si può velocemente notare dall'indicatore **f1-socre**, parametro maggiormente preso in considerazione, grazie a queste migliorie si riesce a perfezionare di quasi un 80% l'algoritmo con previsioni peggiori, mentre di poco più di un 2% negli alberi decisionali dove ha mantenuto le predizioni migliori.

	Model	Accuracy	Acc spam	Acc non spam	Precision	Recall	F1-Score	AUC
1	Decision Tree Classifier	0.910543	0.936238	0.871795	0.900662	0.871795	0.885993	0.904016
0	K Nearest Neighbour	0.789137	0.839532	0.713141	0.746644	0.713141	0.729508	0.776337
2	Support Vector Classification	0.716933	0.901169	0.439103	0.746594	0.439103	0.552977	0.670136

Figure 15: Sample senza ottimizzazioni

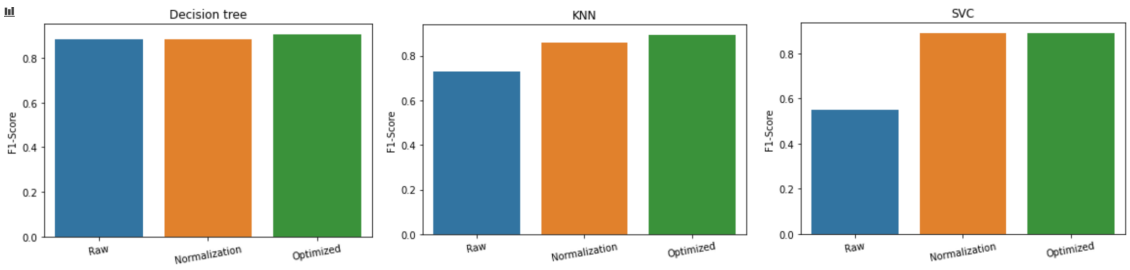
	Model	Accuracy	Acc spam	Acc non spam	Precision	Recall	F1-Score	AUC
2	Support Vector Classification	0.916933	0.957492	0.855769	0.930314	0.855769	0.891486	0.906631
1	Decision Tree Classifier	0.907348	0.926674	0.878205	0.888169	0.878205	0.883159	0.902439
0	K Nearest Neighbour	0.891374	0.929862	0.833333	0.887372	0.833333	0.859504	0.881598

Figure 16: Sample normalizzati

	Model	Accuracy	Acc spam	Acc non spam	Precision	Recall	F1-Score	AUC
1	Decision Tree Classifier	0.925240	0.938363	0.905449	0.906902	0.905449	0.906175	0.921906
0	K Nearest Neighbour	0.916933	0.944740	0.875000	0.913043	0.875000	0.893617	0.909870
2	Support Vector Classification	0.916933	0.957492	0.855769	0.930314	0.855769	0.891486	0.906631

Figure 17: Sample normalizzati e perfezionamento parametri

Da come è possibile notare dai grafici in figura 20 gli algoritmi attraverso queste ottimizzazioni hanno migliorato decisamente le previsioni della classe di appartenenza dell'email.



(a) (b) (c)

Figure 18

5 API

Attraverso l'utilizzo di Fast Api, una libreria python, si è implementata un'interfaccia grafica, facilmente raggiungibile tramite un browser, che permette di testare velocemente il corretto funzionamento del progetto.

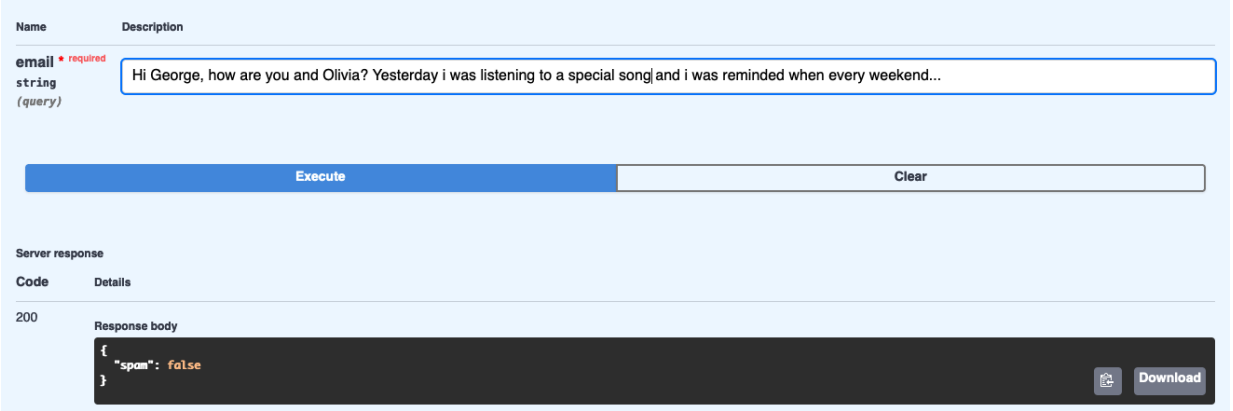


Figure 19: Non spam

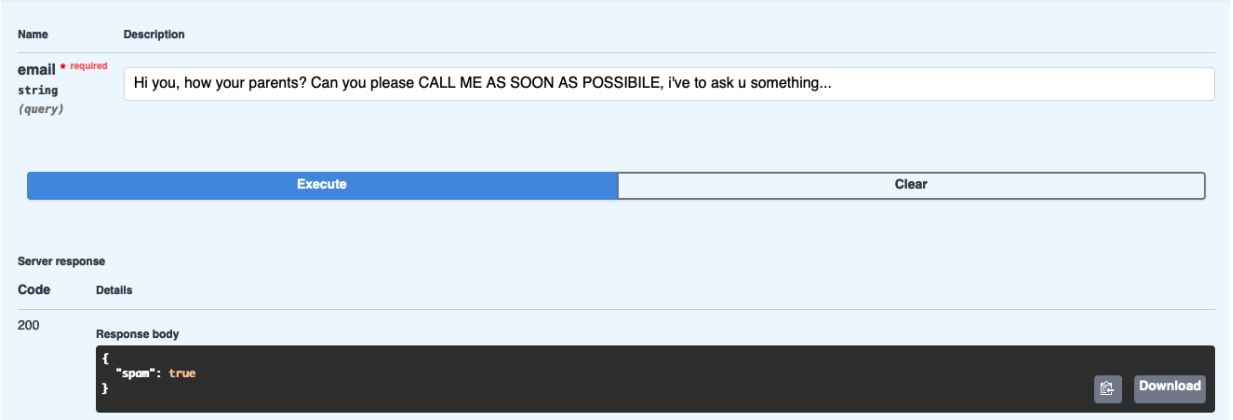


Figure 20: Spam

Bibliografia

[1] Silvia Cascianelli, *Video lezioni e dispense di Machine Learning.*

[2] Sito internet, *it.wikipedia.org*

[3] Sito internet, *stackexchange.com*

[4] Sito internet, *stackoverflow.com*

[5] Sito internet, *scikit-learn.org*

[6] Sito internet, *blog.exsilio.com*

[7] Sito internet, *knowledgehut.com*

[8] Sito internet, *machinelearningmastery.com*

[9] Sito internet, *machinecurve.com*

[10] Sito internet, *lorenzogovoni.com*

[11] Sito internet, *towardsdatascience.com*