Università degli Studi di Pisa

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Cybersecurity

Appunti:
Artificial Intellingence For Security Project

Made by:

Alessandro Niccolini

Giacomo Vitangeli

Anno Accademico 2022-2023

Contents

0.1 Introduction 1

1 Project Explanation **2**
1.1 Model Comparison 4
1.2 Future Work 9

0.1 Introduction

Il progetto si concentra sull'analisi e nell'applicazione delle tecniche viste a lezione. Partendo dal *Data Mining*, cercando di capire le caratteristiche (come attributi e dati) del database scelto. Poi nella fase di *Data Cleaning* andando a pulire quelli che sono i dati presenti nel database controllando la presenza di valori NULL. Successivamente nel *Data Reduction* si cerca di ridurre la complessità e di bilanciare il database. Dopo ci troviamo nella fase di *Classification* dove andiamo a dividere il dataset in training/test set, con l'obiettivo di allenare il nostro modello per riconoscere nuovi oggetti e dire a quale classe appartengono. Nella fase di *Performance Evaluation* si vanno a usare le metriche per valutare se il nostro modello è buono oppure no, anche tramite l'utilizzo di approcci statistici come **Cross-Validation**.

1 Project Explanation

Il Goal di questo progetto è quello di allenare un modello in grado di riconoscere le email di phishing. In questo caso sono stati utilizzati due database, uno di URL e uno di email con i label già presenti. Nella figura 1.1 è già stato fatto il merge dei due database. Mentre dalla figura 1.2, possiamo vedere che il database non è bilanciato.

```
df.sample(10)
```

✓ 0.3s

	Content	Label
55531	www.indobase.com/events/	1
210688	makethestand.com/	1
202781	jeffersonpva.ky.gov/	1
156207	cantonesejobs.com/	1
220802	newbernnchotels.com/	1
284702	aspenlakehouse.com/	1
328792	facebook.com/pages/Christy-Chung/24838401265	1
708	Subject: re : pg & e texas contract 5098 - 695...	1
417876	psa-squash.com/entry/ranking.php?player=T02685	1
431079	siliconindia.com/	1

Figure 1.1

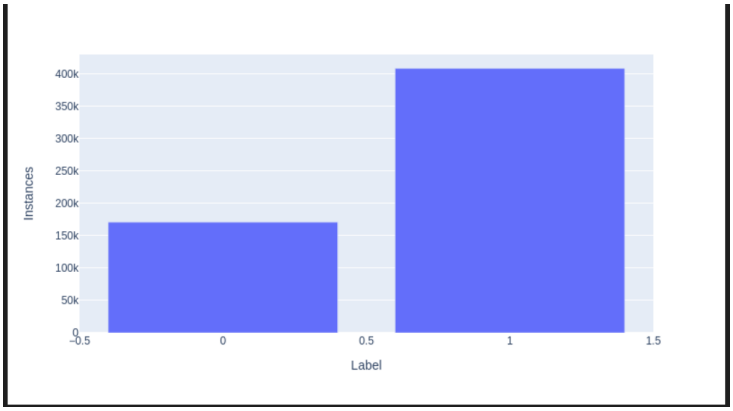


Figure 1.2

Per risolvere questo problema quello che viene fatto è un *Undersampling*,

andando quindi a ridurre la classe più numerosa. In questo caso viene applicato il *Random Undersampling*. Dalla figura 1.3, si vede il risultato di questa operazione. Successivamente viene utilizzata la *Pipeline*, che serve per raggruppare delle unità di elaborazione collegate in serie, in cui l'uscita di un elemento serve come ingresso per il successivo.

```
undersampled_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 342012 entries, 0 to 342011
Data columns (total 2 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   Content  342012 non-null  object
 1   Label    342012 non-null  object
dtypes: object(2)
memory usage: 5.2+ MB
```

Figure 1.3

Nella pipeline di figura 1.4, si vede che vengono utilizzati:

- TfidfVectorizer, che è uno strumento statistico utilizzato per valutare il significato della parola in un documento o in un gruppo di documenti. Per estrarre caratteristiche pertinenti dai testi, viene spesso utilizzato nelle attività di information retrieval e di Natural Language Processing (NLP). Segue il principio che le parole che compaiono più frequentemente in un testo sono più significative, mentre i termini che compaiono in tutti i documenti hanno una rilevanza minore. Questo strumento viene accompagnato con l'utilizzo di *stopword*, ovvero, parole comuni che di solito vengono filtrate dalle attività di NLP perché non forniscono molte informazioni significative sul contenuto di un testo (ad esempio a, un, e, ma, etc.).

- LogisticRegression, che è un algoritmo di apprendimento automatico utilizzato per la classificazione. I modelli di regressione di questo tipo sono utilizzati per prevedere esiti binari, come in questo caso, phishing oppure no. Il modello apprende una relazione lineare tra le caratteristiche di input e la probabilità della classe target nella regressione logistica.

- KFold, si tratta di un metodo di Cross-validation utilizzato per valutare le prestazioni di un modello di apprendimento automatico. Consiste nel dividere i dati in K fold, dove K è un numero specificato dall'utente ($k = 10$ in questo caso). Il modello viene quindi addestrato e valutato K volte, utilizzando ogni volta un fold diverso come set di test e le fold rimanenti come set di addestramento. Le prestazioni del modello vengono quindi calcolate come media di tutte le K iterazioni.

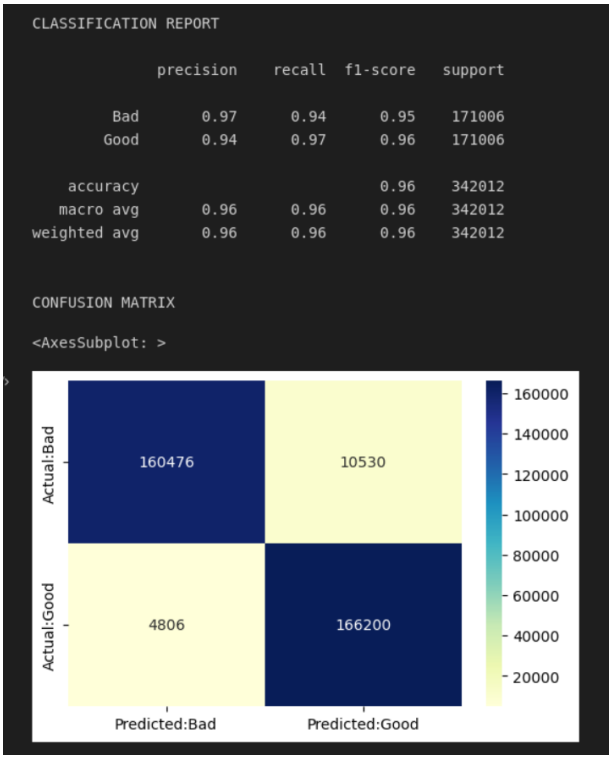
```
start = time()
pipe = Pipeline([('vect', TfidfVectorizer(stop_words = stopwords)), ('clf', LogisticRegression())])
y_pred = cross_val_predict(pipe, undersampled_df.Content.values, under_label, cv=kf)
LR_tfidf = cross_validate(pipe,
                          undersampled_df.Content.values,
                          under_label,
                          scoring = {'precision_ham': make_scorer(precision_score,pos_label = 1),
                                     'precision_spam': make_scorer(precision_score,pos_label = 0),
                                     'recall_ham': make_scorer(recall_score,pos_label = 1),
                                     'recall_spam': make_scorer(recall_score,pos_label = 0),
                                     'accuracy': make_scorer(accuracy_score),
                                     'fscore_spam': make_scorer(f1_score,pos_label = 0),
                                     'fscore_ham': make_scorer(f1_score,pos_label = 1)},
                          return_estimator = True,
                          cv = kf,
                          n_jobs = 12) # Number of jobs to run in parallel.
                                       # Training the estimator and computing the score
                                       # are parallelized over the cross-validation splits.

print_metrics(LR_tfidf)

elapsed = time()-start
print(elapsed)
```

Figure 1.4

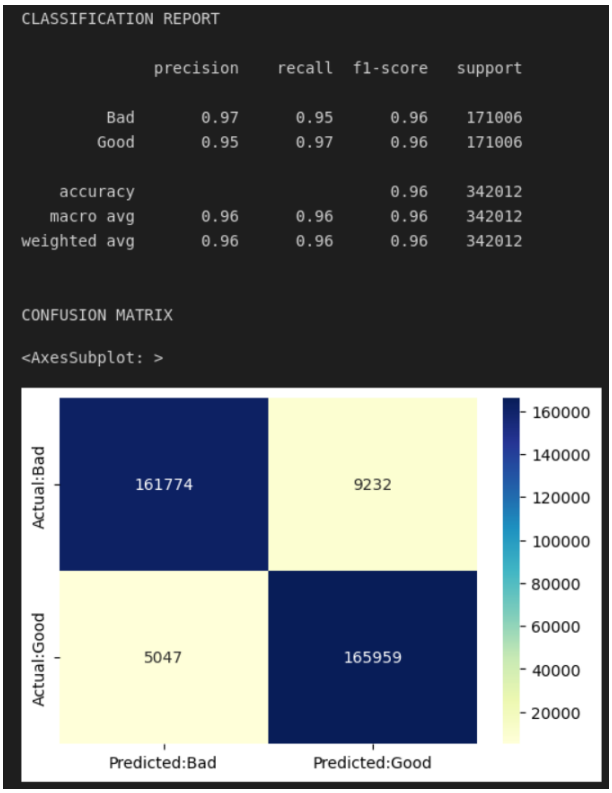
Dopo viene utilizzata la *matrice di confusione*, che non è altro che una tabella utilizzata per valutare le prestazioni di un modello di classificazione. Mostra il numero di previsioni true positive, true negative, false positive e false negative fatte dal modello. Come si vede in figura 1.5a.

1.1 Model Comparison

Nella prima comparazione mettiamo a confronto due Modelli che fanno uso entrambi della *LogisticRegression*, mentre come strumento statistico utilizzano, il primo il **TfidfVectorizer** e il secondo il **CountVectorizer**. Di seguito sono riportati i risultati della comparazione. La differenza principale tra i due è che, CountVectorizer si limita a contare il numero di volte in cui una parola appare in un documento, mentre il TF-IDF Vectorizer considera sia la frequenza di una parola sia la sua importanza per l'intero corpo del



(a) Confusion Matrix LogisticRegression with TfidfVectorizer



(b) Confusion Matrix LogisticRegression with CountVectorizer

Figure 1.5

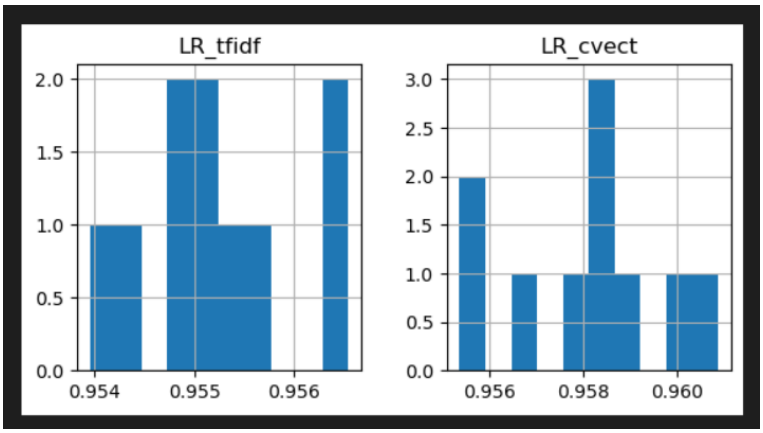
documento. A tal fine, i termini che compaiono spesso in tutti i documenti vengono penalizzati, riducendo il loro numero perché probabilmente non sono così significativi.

Oltre alla metrica di *Accuracy*, che rappresenta la percentuale di previsioni corrette fatte dal modello, calcolata come il numero di previsioni true positive e true negative diviso per il numero totale di previsioni. Dopo è stato fatto il confronto anche con la metrica *fscore_spam* che combina Precision e Recall, anche in questo caso si evidenzia come il modello che fa uso del CountVectorizer sia migliore.

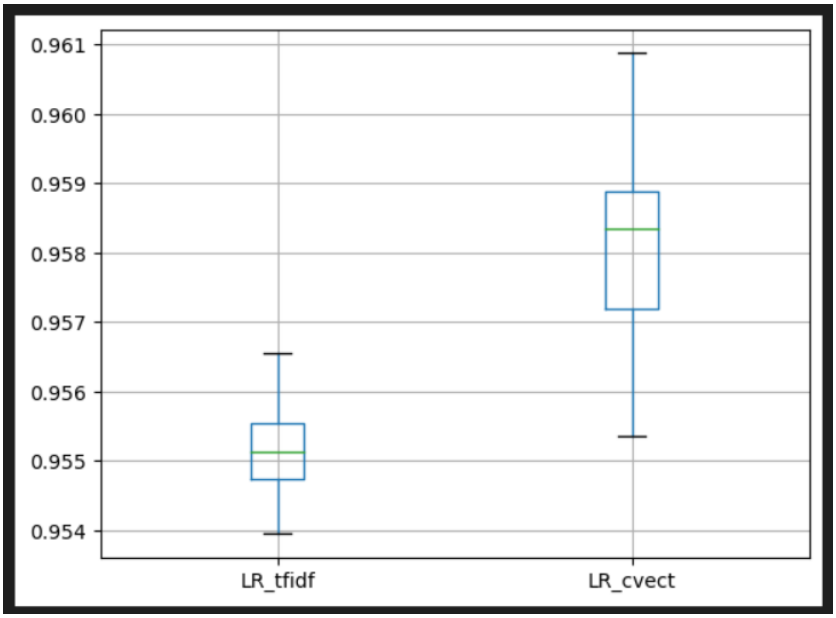
Dopodiché abbiamo fatto il confronto tra il modelli costruito usando **LinearSVC** e quello con **LogisticRegression**. Il LinearSVC è una variante

	LR_tfidf	LR_cvect
0	0.955032	0.955353
1	0.954739	0.957985
2	0.953949	0.956931
3	0.956317	0.960878
4	0.954329	0.958656
5	0.955323	0.959884
6	0.955235	0.958949
7	0.955615	0.958276
8	0.954767	0.955762
9	0.956551	0.958422

(a)



(b)



(c)

Figure 1.6: LS with CountVectorizer VS LS with TfidfVectorizer (Accuracy)

della tecnica Support Vector Machines (SVM)[1] basata su un kernel lineare, che la rende adatta all'uso con dati linearmente separabili. Poiché è veloce da addestrare e prevedere e offre una serie di iperparametri utili che consentono di regolare il comportamento del modello, LinearSVC è un'opzione molto valida per la classificazione.

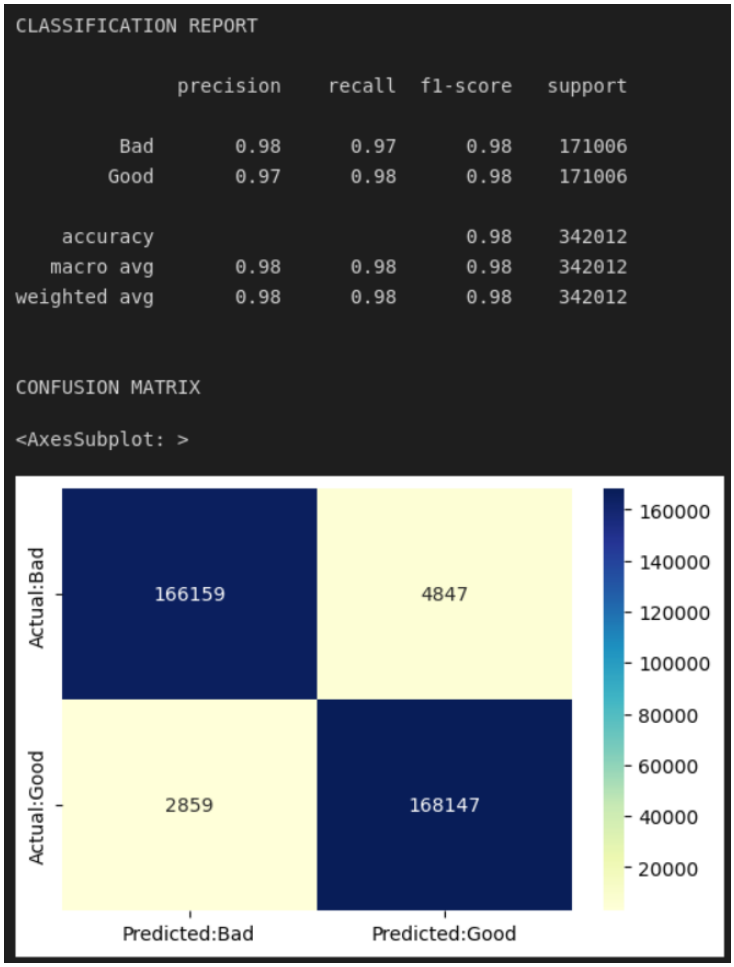


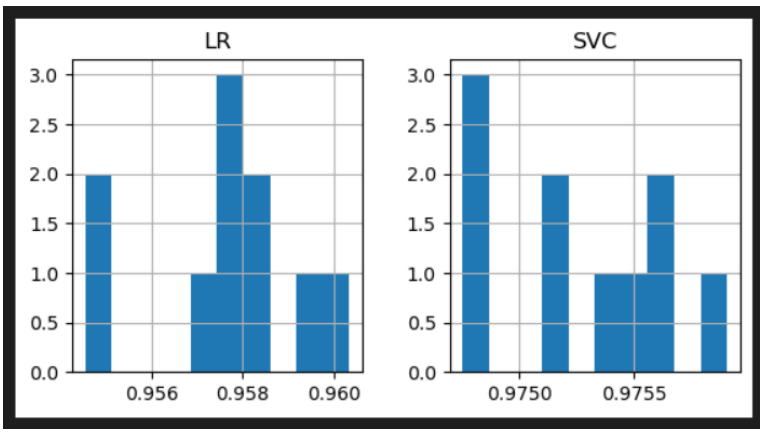
Figure 1.7

Come si osserva dalle figure precedenti, il modello allenato con LinearSVC

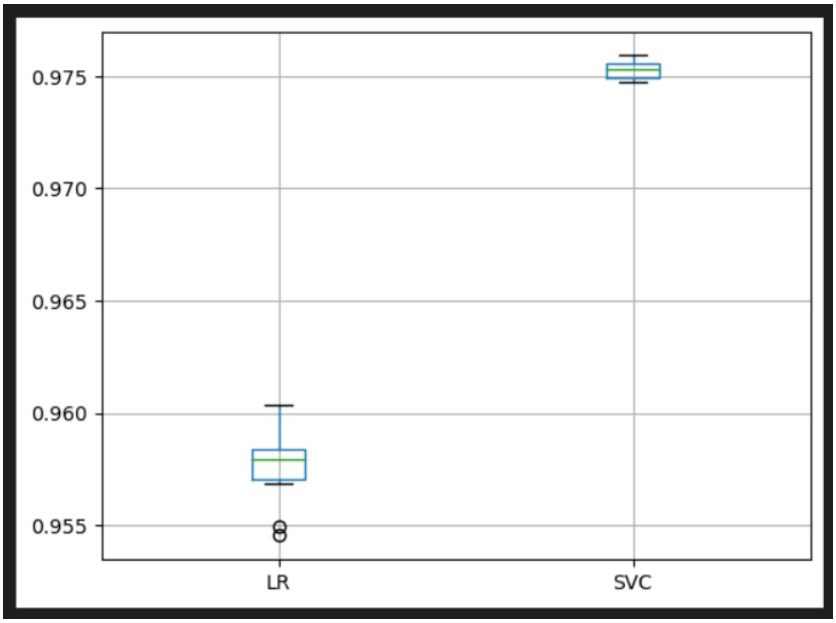
[1]Le SVM sono algoritmi di apprendimento automatico supervisionato che possono essere applicati alla classificazione, alla regressione e ad altri problemi. Si basano sulla nozione di individuare un iperpiano in uno spazio ad alta densità che segreghi al meglio le varie classi.

	LR	SVC
0	0.954539	0.975435
1	0.957479	0.975468
2	0.956888	0.974855
3	0.960349	0.974784
4	0.958166	0.975140
5	0.959343	0.975180
6	0.958434	0.974757
7	0.958006	0.975571
8	0.954921	0.975655
9	0.957837	0.975903

(a)



(b)



(c)

Figure 1.8: LS with CountVectorizer VS SVC with CountVectorizer (F-score)

ha risultati visivamente migliori. Questi risultati si notano anche andando ad analizzare la metrica di *Accuracy*.

Mentre nell'ultimo confronto si vedono i risultati della comparazione tra due modelli allenati con *LinearSVC* che fanno uso, uno di **CountVectorizer** e uno di **TfidfVectorizer**. Vengono confrontati usando le metriche di *Accuracy e F-score*, ed in entrambe si nota che quello che fa uso di TfidfVectorizer risulta essere il migliore.

Wilcoxon Test
Un test statistico non parametrico chiamato *Wilcoxon test* viene utilizzato per rilevare se due campioni correlati provengono dalla stessa distribuzione. Quando i dati non sono distribuiti normalmente o le dimensioni del campione sono minime, viene spesso utilizzato per confrontare le medie di due campioni simili. Per determinare la probabilità di vedere una statistica di prova almeno altrettanto estrema di quella ottenuta, nell'ipotesi che la *NULL-Hyphotesis* sia vera, è possibile calcolare il valore p utilizzando la statistica di Wilcoxon. È possibile rifiutare la NULL-Hyphotesis e dedurre che i due campioni provengono da distribuzioni distinte se il valore p è inferiore a una soglia predeterminata (in genere 0,05).

1.2 Future Work

Per migliorare ulteriormente il modello potrebbe essere aggiunto al database una parte aggiuntiva di email di phishing, in quanto rispetto agli URL sono in numero decisamente inferiore. Un altra possibilità è quella di usare un algoritmo di machine learning supervisionato più accurato ad esempio la variante dell'SVC non-lineare.