S.Q.l.D

Davide Amoruso
matr. 05112114271

Contents

This page intentionally left blank.

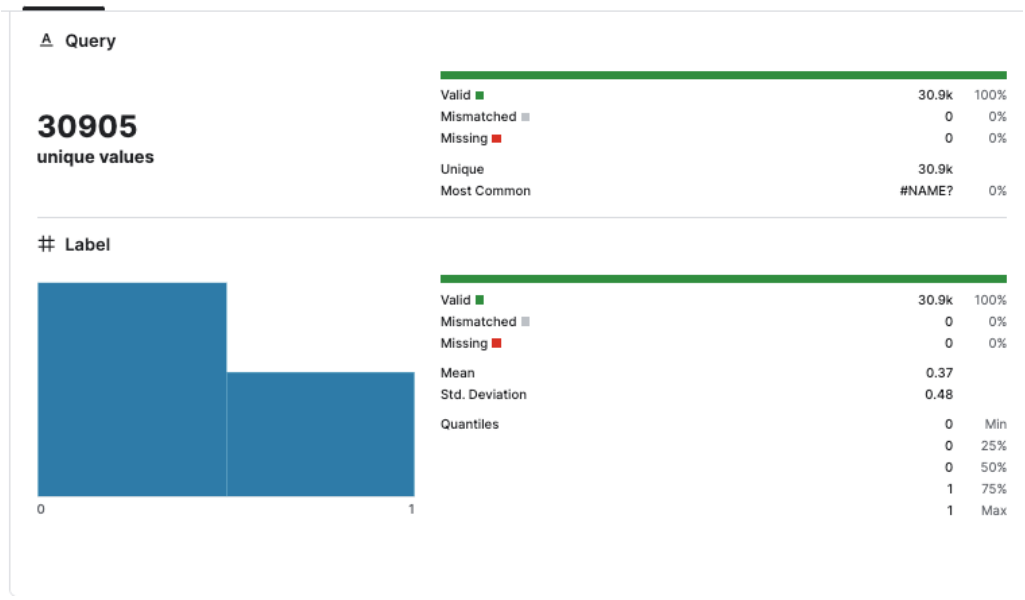
1 Introduzione

Nell'era della digitalizzazione, la sicurezza dei dati è diventata una preoccupazione primaria per le organizzazioni. I database, che sono il cuore pulsante di quasi tutte le attività digitali, sono spesso soggetti a attacchi e violazioni. Uno dei metodi più comuni utilizzati per compromettere la sicurezza dei database è tramite le SQLInjection, un tipo di attacco in cui un malintenzionato inserisce codice SQL dannoso in una query. Questo progetto si concentra sulla creazione di un modello di Machine Learning per rilevare tali anomalie nelle query SQL.
Per raggiungere questo obiettivo, è stato utilizzato un ampio set di dati , che sono stati poi pre-elaborati e utilizzati per addestrare diversi modelli di shallow Machine Learning.
Nel corso di questo documento, saranno descritte in dettaglio il processo di pre-elaborazione dei dati, l'approccio utilizzato per l'addestramento del modello e i risultati ottenuti.
Tutto il codice relativo all'addestramento ed all'implementazione del modello sono presenti su: https://github.com/Dabi1290/S.Q.I.D

2 Data Engineering

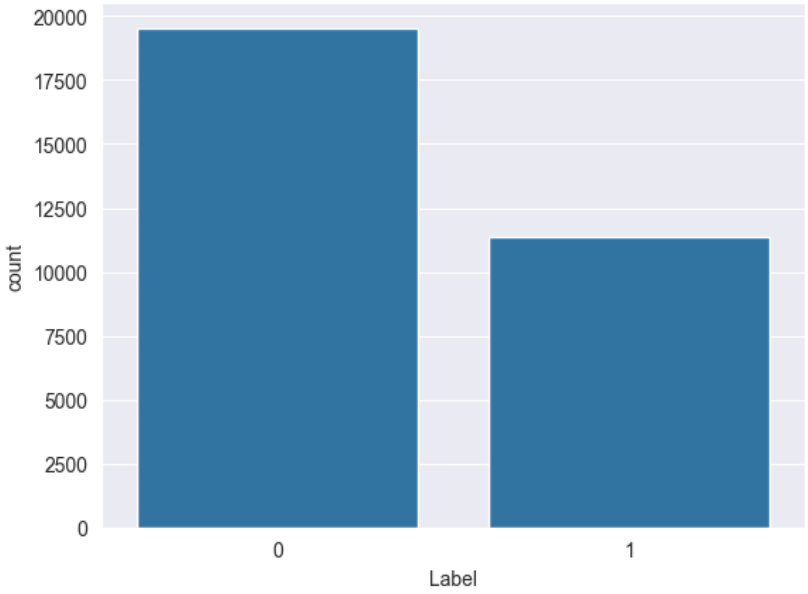
2.1 Data Sources

I dati che sono stati usati per l'addestramento dei classificatori sono stati presi dal sito Kaggle, più precisamente il dataset preso in considerazione è presente al link:
https://www.kaggle.com/datasets/sajid576/sql-injection-dataset
Il dataset è composto da un unica tabella in formato CSV, composta da 2 colonne: "Query" e "Label". il numero di elementi di cui è composto è : 30905. Gli elementi della colonna "Query" sono query SQL nel formato stringa, mentre nella colonna "Label" vi è la label corrispondente: 1 se la query è dannosa, 0 se non lo è. Dalla Scheda di Kaggle si può vedere che non sono presenti valori nulli o mancanti e che tutti i valori sono unici tra loro.



2.2 Data Esploration

Nell'esplorazione dei dati si procede a visualizzare quante query dannose e quante buone sono presenti.



Si procede inoltre a visualizzare i dati veri e propri

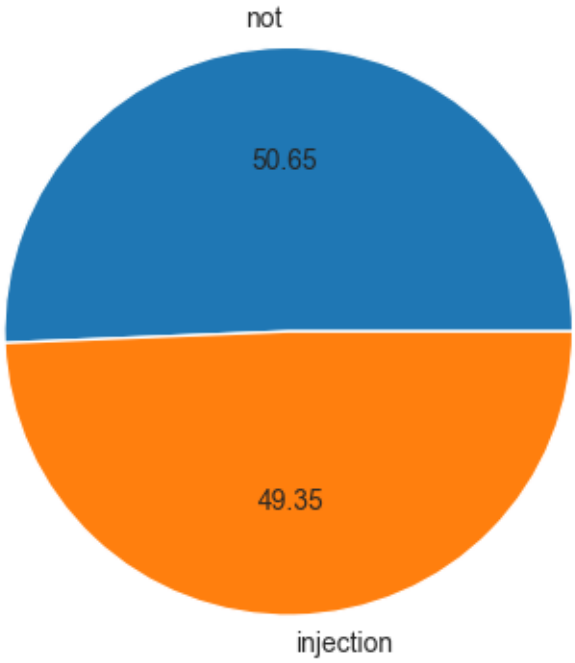
14.021-28.020 30919 rows × 2 columns pd.DataFrame

	Query	Label
140…	painter.stovin@tetdevelocidad.bn	0
140…	8466	0
140…	sim5e469	0
140…	trevetha	0
140…	38980059g	0
140…	mango_rizzo1@premiostv.ky	0
140…	3.61806E+15	0
140…	particular teresa orozco 101	0
140…	2.24328E+15	0

Dall'esplorazione dei dati si può vedere che il dataset è sbilanciato, ma anche che una grande quantità di dati all'interno del dataset non sono utili per l'addestramento del classificatore, quindi possono essere eliminati.

2.3 Data Balancing

Bisogna controllare se il dataset risulta sbilanciato, fortunatamente dopo le varie procedure di data cleaning il dataset risulta bilanciato



3 Feature Engineering

3.1 Text cleaning

Dato che i dati che vado a trattare sono di tipo testuale, bisogna effettuare diverse operazioni, tra cui:

- Rimozione delle Stopword
- Lemmatizzazione
- Contrazione
- rimozione della punteggiatura
- Portare tutto in minuscolo
- Tokenizzazione

3.1.1 Rimozione delle Stopword

È stato ritenuto opportuno non effettuare questa operazione dato che alcune Stopword potrebbero essere parte integrante del linguaggio SQL che potrebbe rivelarsi utile per il classificatore.

3.1.2 Lemmatizzazione

È stato ritenuto opportuno non effettuare questa operazione dato che alcuni verbi potrebbero essere nomi di tabelle sulle quali vengono effettuate maggiormente attacchi e questo dato sembra essere rilevante per i diversi domini presi in considerazione dal classificatore.

3.1.3 Contrazione

È stato ritenuto opportuno non effettuare questa operazione dato che le query SQL che fanno uso di forme contratte sono molto poche e quindi il fattore di rischio è molto basso.

3.1.4 rimozione della punteggiatura

Come per la rimozione delle Stopword buona parte della punteggiatura può essere un fattore rilevante per la classificazione di una query, basta pensare semplicemente ad un commento(Es. '–').

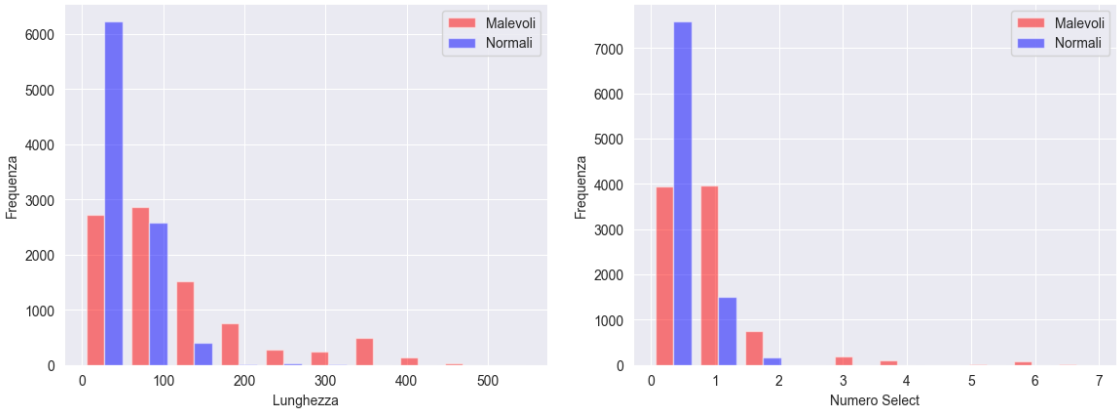
3.1.5 Portare tutto in minuscolo

È stato ritenuto opportuno effettuare questa operazione dato che portare in minuscolo le parole apporta benefici al classificatore.

3.1.6 Tokenizzazione

Questa operazione è fondamentale per far si che i dati testuali possano essere utilizzati durante la fase di addestramento e quindi è stato ritenuto opportuno effettuarla

3.2 Feature Extraction



Esaminando i dati si nota che le query dannose hanno una lunghezza maggiore rispetto alle query normali e che presentano l'istruzione "select" più volte, si procede quindi ad estrarre le features ed inserirle all'interno del dataframe.

1-10 18325 rows × 4 columns pd.DataFrame

	Query	Label	lunghezza	Select_Count
26851	fairly (s from volume right join	0	33	0
1866	1' in boolean mode) order by 1--	1	36	0
4949	1" and 3824 = benchmark (5000000,md5 …	1	65	0
26246	* from tone where several between '1996-07-…	0	63	0
8451	1" where 4445 = 4445 procedure analyse (…	1	166	0
8523	1)) (select (case when (…	1	321	1
3405	1'))) and updatexml (3393,…	1	195	1
24523	* from wherever order by consonant asc, wit…	0	51	0
7584	-3017")) as ljit where 9128 = 9	1	80	1

3.2.1 Feature scaling

Le nuove features aggiunte presentano una grande differenza in termini di varianza, e quindi c'è bisogno di effettuare un operazione di feature scaling, si procede attraverso la min-max normalization.

1-10 18325 rows × 4 columns pd.DataFrame

	Query	Label	lunghezza	Select_Count
0	" or pg_sleep (__time__) --	1	0.059783	0.000000
1	create user name identified by pass123 temp…	1	0.163043	0.000000
2	and 1 = utl_inaddr.get_host_address (…	1	0.393116	0.285714
4	* from users where id = 1 or 1#" (uni…	1	0.139493	0.142857
5	name from syscolumns where id = (se…	1	0.182971	0.142857
6	* from users where id = 1 +$+ or 1 = 1 …	1	0.085145	0.000000
7	1; (load_file (char (47,101,116,99…	1	0.168478	0.000000
8	* from users where id = '1' or ||/1 = 1…	1	0.148551	0.142857
9	* from users where id = '1' or) <) union	1	0.123188	0.142857

4 Model Development

I dati sono ora pronti per l'addestramento, essendo questo un problema di classificazione si è deciso di mettere a confronto diverse classi di modelli adatti a tale scopo: Naive Bayes,Support Vector Machines, Nearest Neighbour, Decision Tree e Random Forest.

4.1 Naive Bayes

Gli algoritmi Naive Bayes si basano sull'applicazione del teorema di Bayes con l'assunzione "naive". Il teorema di Bayes è la base degli algoritmi Naive Bayes, che è un principio fondamentale nella teoria delle probabilità. In termini semplici, il teorema di Bayes calcola la probabilità di un evento basandosi su conoscenze pregresse che sono correlate all'evento.
L'assunzione "naive" in Naive Bayes è che ogni caratteristica contribuisce in modo indipendente alla probabilità finale. Questa assunzione semplifica il calcolo, ma non è sempre accurata nel mondo reale dove le caratteristiche possono essere correlate.

4.2 Support Vector Machines

Gli algoritmi Support Vector Machines puntano a trovare l'iperpiano che rappresenta il massimo margine tra le classi nei dati di addestramento.
In termini semplici, SVM sono modelli che cercano di trovare un separatore che permetta di distinguere i dati in base a dove vengono posizionati nello spazio.
Inoltre, SVMs possono efficientemente eseguire una classificazione non lineare utilizzando ciò che viene chiamato il trucco del kernel, mappando implicitamente i loro input in spazi ad alta dimensione. 1

4.3 Nearest Neighbours

Gli algoritmi Nearest Neighbours si basano sul concetto di "vicinanza" o "distanza" tra punti in uno spazio multidimensionale.
In termini semplici, l'idea di base degli algoritmi Nearest Neighbors è che i punti "vicini" tra loro nello spazio delle caratteristiche tendono ad avere etichette simili. Ad esempio, in un problema di classificazione, un punto non etichettato avrà come etichetta l'etichetta della maggior parte dei suoi "vicini" più vicini.

4.4 Decision Tree

Gli algoritmi decision tree, utilizzano una struttura ad albero per rappresentare una serie di decisioni possibili e i loro possibili risultati. Si basa sull'entropia per scegliere il nodo in modo che esso sia più puro possibile.Ogni nodo interno dell'albero rappresenta una decisione (o "test") su un attributo, ogni ramo rappresenta il risultato del test, e ogni foglia dell'albero rappresenta un risultato (o "classe") della decisione.
Gli algoritmi decision tree sono particolarmente utili per problemi di classificazione ed offrono una buona explainability

4.5 Random Forest

Gli algoritmi Random Forest combinano molteplici alberi decisionali per creare un modello più potente e robusto. Questo approccio è noto come ensemble learning, dove si combinano le previsioni di più modelli per ottenere una previsione finale più accurata.
Ogni albero nel Random Forest viene addestrato su un sottoinsieme diverso dei dati di addestramento, scelto a caso. Inoltre, ad ogni nodo dell'albero, un sottoinsieme casuale delle caratteristiche viene scelto per la divisione. Questa randomizzazione aiuta a rendere il modello più robusto contro l'overfitting, che può essere un problema con i singoli alberi decisionali.

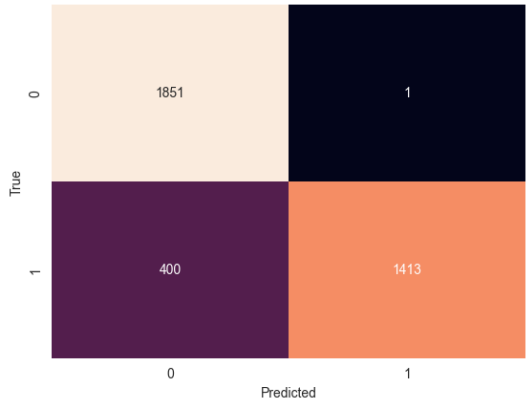
5 Evaluation

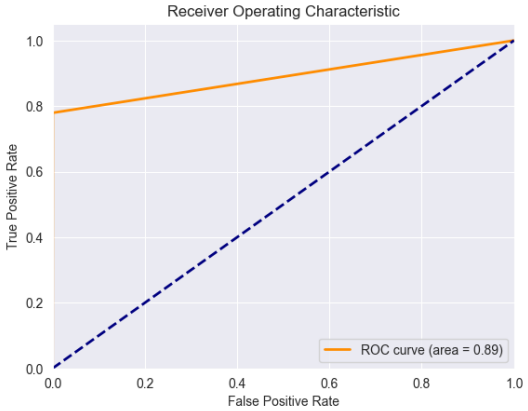
Nella fase di validazione sono state prese come riferimento Accuracy, Precision, Recall, F1-score, Confuzion matrix e ROC curve

5.1 Naive Bayes

GaussianNB

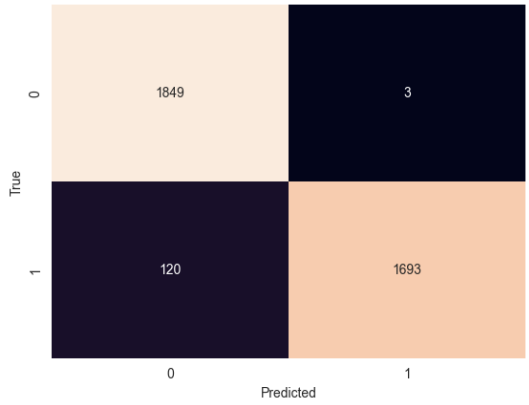
- Accuracy: 0.8905866302864939
- Precision: 0.9098552914583584
- Recall: 0.8905866302864939
- F1-score: 0.8891424649473036

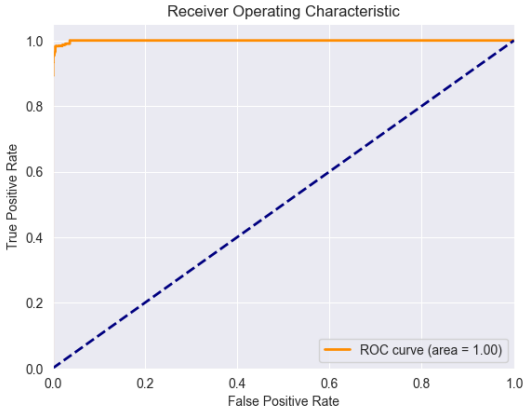




BernoulliNB

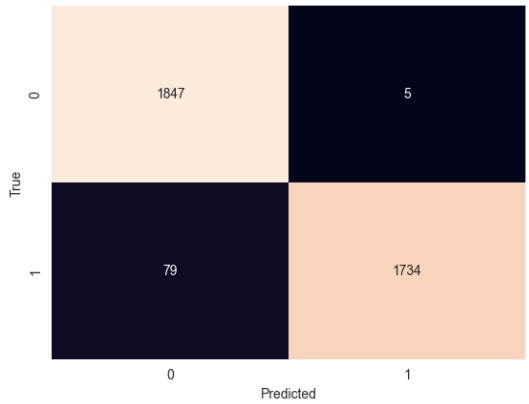
- Accuracy: 0.9664392905866303
- Precision: 0.9683283944231942
- Recall: 0.9664392905866303
- F1-score: 0.9663936047916323

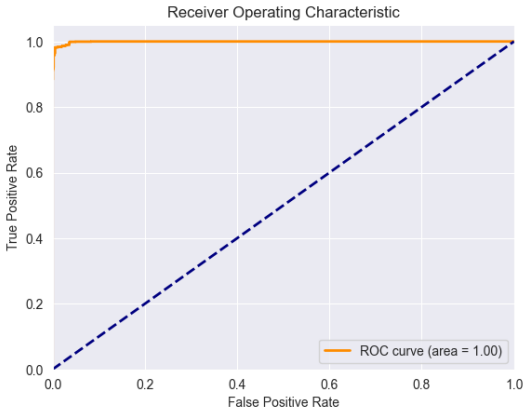




MultinomialNB

- Accuracy: 0.9770804911323329
- Precision: 0.9778506249323123
- Recall: 0.9770804911323329
- F1-score: 0.977066209410842

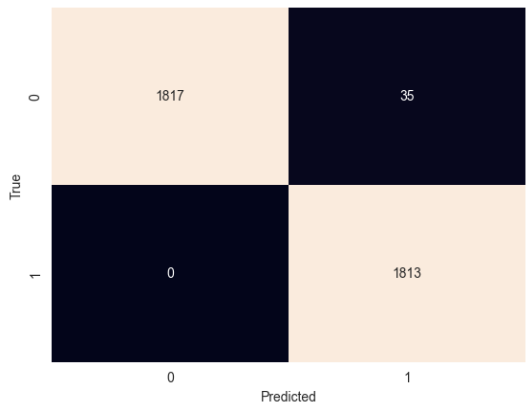


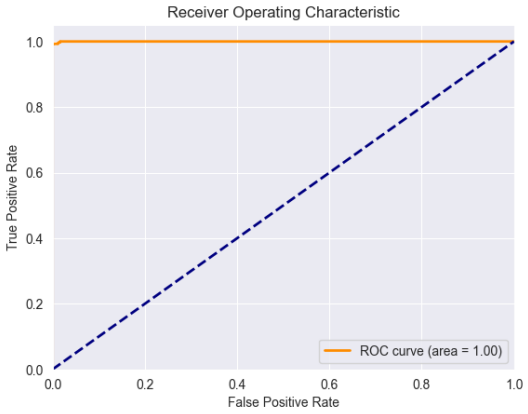


5.2 Support Vector Machines

SVC

- Accuracy: 0.990450204638472
- Precision: 0.9906310719748647
- Recall: 0.990450204638472
- F1-score: 0.9904503041732061





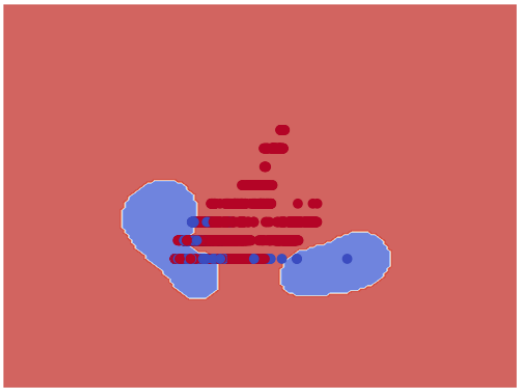
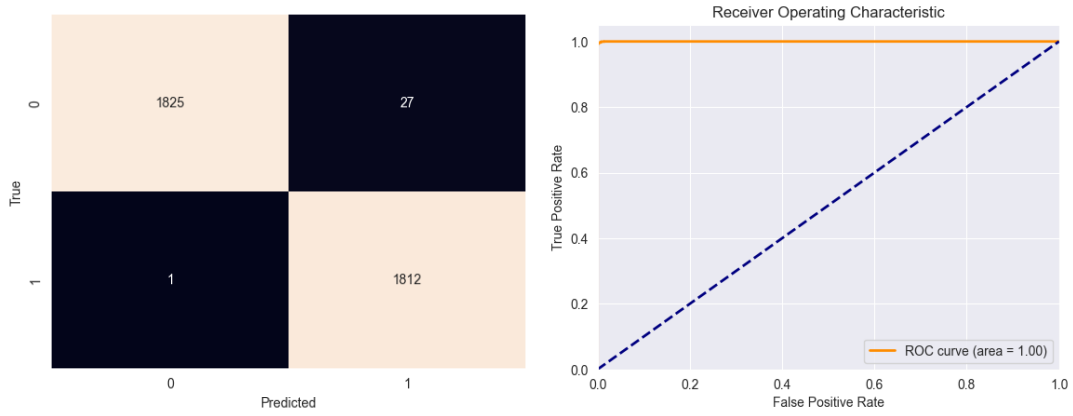


Figure 1: esempio di classificazione effettuata da SVM, l'esempio fa riferimento al risultato ottenuti in fase di test dall'algoritmo Support Vector classifier

LinearSVC

- Accuracy: 0.9923601637107776
- Precision: 0.9924604339501713
- Recall: 0.9923601637107776
- F1-score: 0.9923603559571965



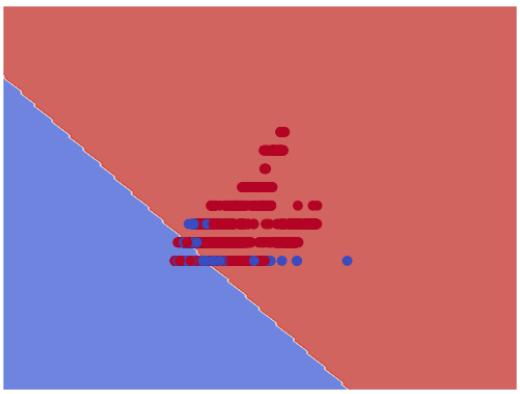
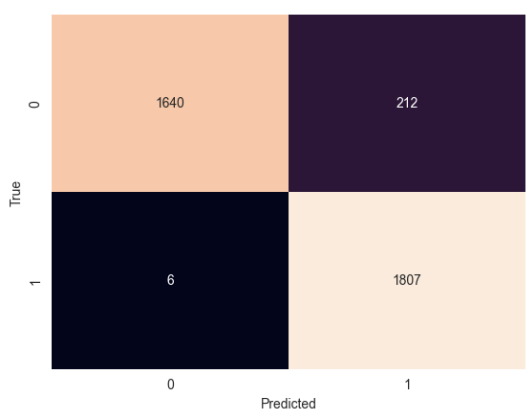


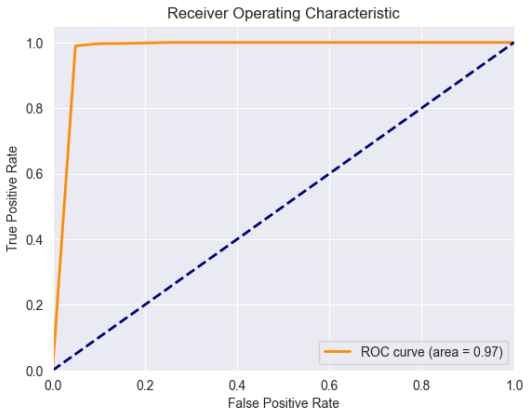
Figure 2: esempio di classificazione effettuata da SVM, l'esempio fa riferimento al risultato ottenuto in fase di test dall'algoritmo Linear support vector classifier

5.3 Nearest Neighbours

KNeighborsClassifier

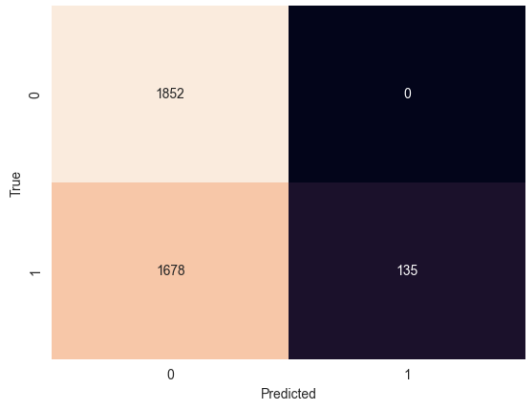
- Accuracy: 0.940518417462483
- Precision: 0.9462154430705103
- Recall: 0.940518417462483
- F1-score: 0.9403657589941742

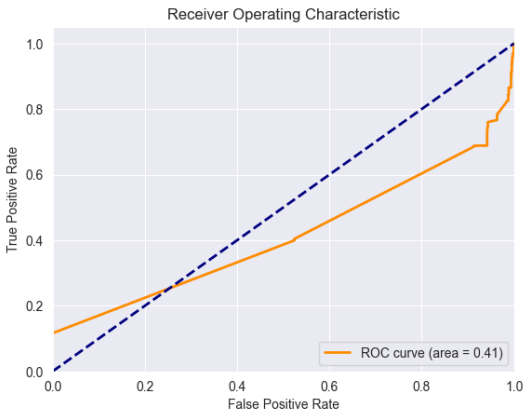




RadiusNeighborsClassifier

- Accuracy: 0.5421555252387449
- Precision: 0.7597937769807807
- Recall: 0.5421555252387449
- F1-score: 0.41633613295014854

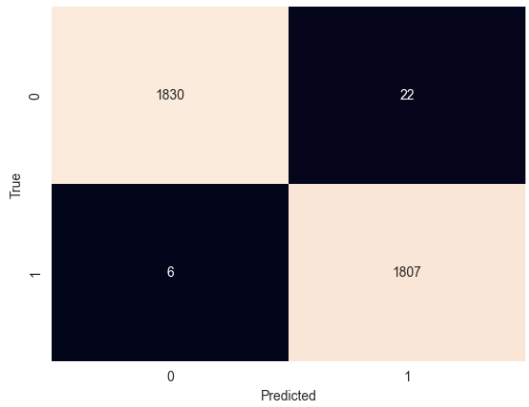


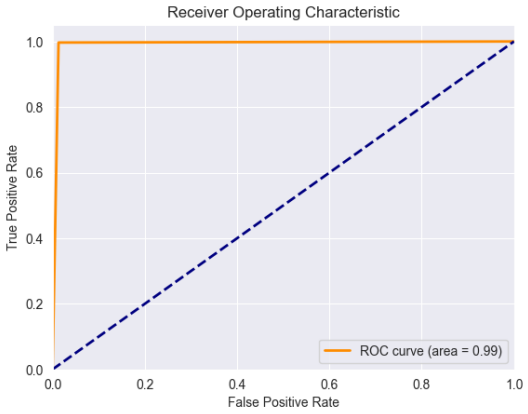


5.4 Decision Tree

DecisionTreeClassifier

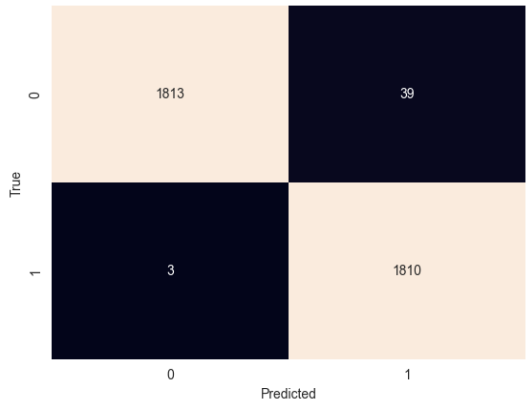
- Accuracy: 0.9923601637107776
- Precision: 0.9923984085434339
- Recall: 0.9923601637107776
- F1-score: 0.9923603730260981

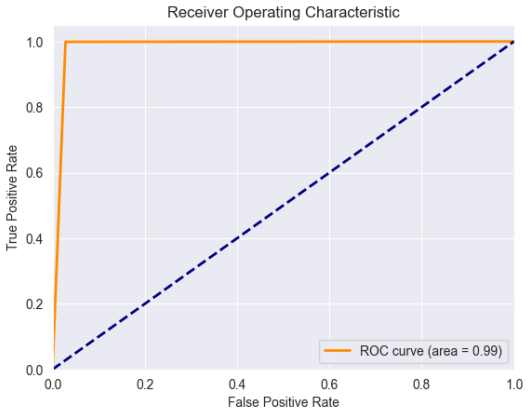




ExtraTreeClassifier

- Accuracy: 0.9888130968622101
- Precision: 0.9890147963574729
- Recall: 0.9888130968622101
- F1-score: 0.9888131584924373

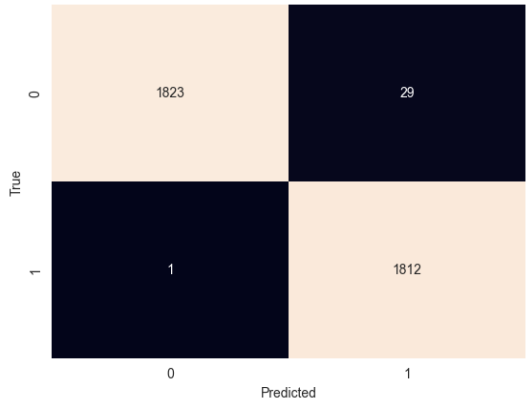


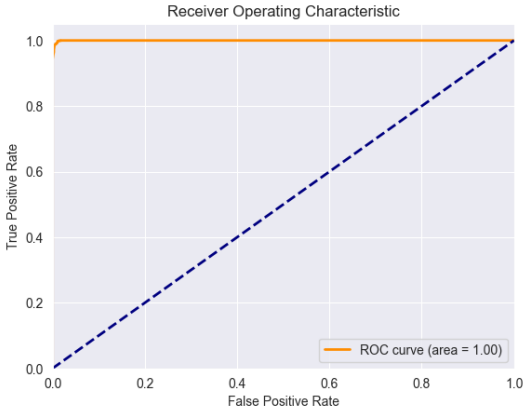


5.5 Random Forest

RandomForestClassifier

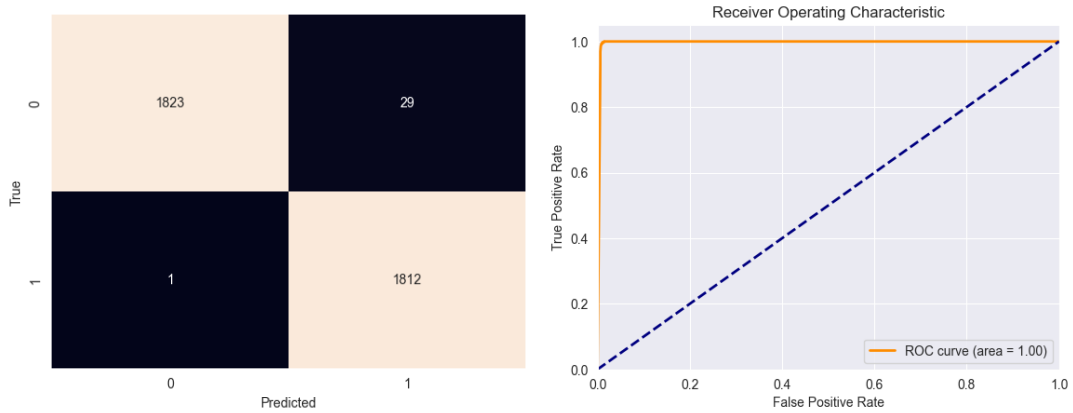
- Accuracy: 0.9918144611186903
- Precision: 0.9919306176702839
- Recall: 0.9918144611186903
- F1-score: 0.9918146488142275





ExtraTreesClassifier

- Accuracy: 0.992087312414734
- Precision: 0.9921953820040026
- Recall: 0.992087312414734
- F1-score: 0.9920875032794837



	Accuracy	Precision	Recall	F1-score
Gaussian Naive Bayes	0.8905	0.9098	0.8905	0.8891
Bernoulli Naive Bayes	0.9664	0.9683	0.9664	0.9663
Multinomial Naive Bayes	0.9770	0.9778	0.9770	0.9770
Support Vector Classifier	0.9904	0.9906	0.9904	0.9904
Linear Support Vector Classifier	0.9923	0.9924	0.9923	0.9923
K Nearest Classifier	0.9405	0.9462	0.9405	0.9403
Radius Nearest Classifier	0.5421	0.7597	0.5421	0.4163
Decision Tree Classifier	0.9923	0.9923	0.9923	0.9923
Extra Tree Classifier	0.9888	0.9890	0.9888	0.9888
Random Forest Classifier	0.9918	0.9919	0.9918	0.9918
Random Forest Extra Trees Classifier	0.9920	0.9921	0.9920	0.9920

Dalla tabella possiamo facilmente vedere che i migliori modelli sono quelli appartenenti alle classi: Support Vector Machines, Decision tree e Random Forest. Anche il Multinomial Naive Bayes ha dato buoni risultati, ma inferiori rispetto ai precedenti questo è stato dovuto all'approccio naive che quest'ultimo sfrutta. Inoltre vi è un modello che non è riuscito a superare la fase di validazione: Radius Nearest Classifier. Il fallimento di questo modello è causato da come i punti siano sparpagliati all'interno dello spazio, dato che sono molto distanti tra loro, per poter usare questo algoritmo si è dovuto impostare un raggio abbastanza grande da avere almeno k vicini, questo ha fatto si che le predizioni non fossere abbastanza accurate.

6 Deployment

Per il deployment è necessario che i modelli vengano esportati, inoltre per i modelli che sfruttano i dati anche in fase di production bisogna esportare anche i dati. Una volta esportati i modelli bisogna realizzare l'applicativo per poter interagire con questi. Per l'applicazione si è deciso di creare una webapp che simuli il comportamento che il classificatore possa avere in un contesto reale. La principale tecnologia usata per la creazione dell'applicativo è streamlit, una libreria pensata per costruire in modo rapido webapp per appllicativi riguardanti la datascience.

L'applicativo banalmente è composto da una textBox che serve per fornire l'input al classificatore che ci fornirà l'output di ogni modello in modo da poterli confrontare tra loro. Il risultato ottenuto per ogni modello può essere di due tipi: "Normale" o "Dannosa". Nel caso fosse "Normale" la query procederà verso il database, altrimenti la query verrà droppata