SAN FRANCISCO CRIME PREDICTION

Artificial Intelligence & Machine Learning Project

Nicola Santillo

Corso di laurea “Tecniche informatiche per la gestione dei dati”

a.a. 2023/2024

Indice

Abstract— Questo articolo presenta diversi approcci di machine learning implementati per prevedere la probabilità delle categorie di crimine avvenuti nella città di San Francisco. Il focus del progetto è quello di effettuare un'analisi approfondita delle principali tipologie di reati avvenuti nella città e determinare come le varie features contribuiscono agli specifici crimini. Per condurre lo studio sono state utilizzate varie tecniche di classificazione, come Decision Tree, Random Forest , Logistic Regression, Gaussian Naive Bayes, XGboost e la rete neurale di tipo feedforward Multi-Layer Perceptron. I risultati forniscono una base importante per lo sviluppo di strategie di controllo e l'allocazione delle risorse delle forze dell'ordine in una Smart City.

1. Introduzione

Il concetto di Smart City è inteso come uno dei mezzi per migliorare la vita delle persone che vivono in città adottando iniziative intelligenti in una varietà di settori come lo sviluppo urbano, la sicurezza, l'energia e così via. Uno dei fattori che determina la qualità della vita in città è il tasso di criminalità ivi presente. Anche in presenza di molti progressi tecnologici in città, il requisito fondamentale della sicurezza dei cittadini rimane comunque. La criminalità, quindi, continua a rappresentare una minaccia per la società e richiede una seria considerazione se si vuole sperare di ridurne l'incidenza o le ripercussioni da essa causate.

Centinaia sono i crimini che vengono registrati quotidianamente dai data office che lavorano a fianco delle autorità di contrasto in tutti gli Stati Uniti. Molte città degli Stati Uniti hanno aderito all'iniziativa Open Data, rendendo così accessibili al grande pubblico, tra gli altri, i dati sulla criminalità. L'adesione a questa iniziativa sta aumentando la partecipazione dei cittadini al processo decisionale, utilizzando questi dati per scoprire informazioni utili. La città di San Francisco è una delle tante ad aver aderito a questo progetto. I data scientist e gli ingegneri che lavorano a fianco del Dipartimento di Polizia di San Francisco hanno registrato oltre 100.000 casi, solo nel 2016, di crimine sotto forma di denunce pervenute alla polizia. Con l'aiuto di questi dati storici, possono essere scoperte molte informazioni. Ciò aiuterà a prevedere i crimini che potranno verificarsi in futuro e quindi supporterà la polizia cittadina nel salvaguardare al meglio la popolazione della città.

1.1 Obiettivo

Predire il tipo di tipo di crimine in base alle varie features.

2. Dataset

Il set di dati utilizzato in questo progetto proviene da SF OpenData. Esso contiene un totale di 150500 crimini avvenuti nell'anno 2016 nei 10 distretti di San Francisco. Per ciascun incidente vengono forniti tredici campi dati:

- *IncidntNum* : numero di incidente.
- *Date* : data e ora in cui è avvenuto il crimine. Da notare che l'ora è sempre fissa alle 12:00, pertanto per approfondimenti relativi all'ora in cui è stato commesso il crimine prenderemo in considerazione la variabile 'Time'.
- *Time* : ora in cui è avvenuto il crimine. L'intervallo dell'orario varia dalle 00:00 alle 23:59
- *Category* : il tipo di reato. Esso è composta da 39 classi e in fase di classificazione sarà la variabile target da prevedere:

LARCENY/THEFT	ASSAULT	VANDALISM	VEHICLE THEFT	WARRANTS	BURGLARY
SUSPICIOUS OCC	MISSING PERSON	DRUG/NARCOTIC	ROBBERY	FRAUD	SECONDARY CODES
TRESPASS	WEAPON LAWS	SEX OFFENSES, FORCIBLE	STOLEN PROPERTY	RECOVERED VEHICLE	DISORDERLY CONDUCT
PROSTITUTION	FORGERY/COUNTERFEITING	DRUNKENNESS	DRIVING UNDER THE INFLUENCE	ARSON	KIDNAPPING
EMBEZZLEMENT	LIQUOR LAWS	RUNAWAY	SUICIDE	BRIBERY	EXTORTION
FAMILY OFFENSES	LOITERING	SEX OFFENSES, NON FORCIBLE	BAD CHECKS	GAMBLING	PORNOGRAPHY/OBSCENE MAT
TREA	NON-CRIMINAL	OTHER OFFENSES			

- *Description* : descrizione dettagliata dell'incidente. Questo campo dati fornisce informazioni più dettagliate sull'accaduto.
- *DayOfWeek* : il giorno della settimana in cui si è verificato il crimine. Ci sono sette classi possibili: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday.
- *PdDistrict* : la circoscrizione del distretto di polizia dove è avvenuto il crimine. In totale troviamo dieci PD: Bayview, Central, Ingleside, Mission, Northern, Park, Richmond, Southern Taraval, Tenderloin.
- *Resolution* : Come è stato risolto il reato (ad esempio arrestando o incriminando il colpevole).

- *Address* : l'indirizzo stradale dove è avvenuto il crimine.
- *X* : longitudine del luogo del crimine. Gli intervalli di longitudine della città di San Francisco variano da -122.5136 a -122.3649.
- *Y* : latitudine del luogo del crimine. La latitudine della città di San Francisco varia da 37.70788 al 37.81998.
- *Location* : combinazione di latitudine (Y) e longitudine (X)
- *PdID:* identificativo del distretto di polizia.

Di seguito un esempio delle prime righe del dataset:

	IncidntNum	Category	Descript	DayOfWeek	Date	Time	PdDistrict	Resolution	Address	X	Y	Location	PdId
0	120058272	WEAPON LAWS	POSS OF PROHIBITED WEAPON	Friday	01/29/2016 12:00:00 AM	11:00	SOUTHERN	ARREST, BOOKED	800 Block of BRYANT ST	-122.403405	37.775421	(37.775420706711, -122.403404791479)	12005827212120
1	120058272	WEAPON LAWS	FIREARM, LOADED, IN VEHICLE, POSSESSION OR USE	Friday	01/29/2016 12:00:00 AM	11:00	SOUTHERN	ARREST, BOOKED	800 Block of BRYANT ST	-122.403405	37.775421	(37.775420706711, -122.403404791479)	12005827212168
2	141059263	WARRANTS	WARRANT ARREST	Monday	04/25/2016 12:00:00 AM	14:59	BAYVIEW	ARREST, BOOKED	KEITH ST / SHAFTER AV	-122.388856	37.729981	(37.7299809672996, -122.388856204292)	14105926363010
3	160013662	NON-CRIMINAL	LOST PROPERTY	Tuesday	01/05/2016 12:00:00 AM	23:50	TENDERLOIN	NONE	JONES ST / OFARRELL ST	-122.412971	37.785788	(37.7857883766888, -122.412970537591)	16001366271000
4	160002740	NON-CRIMINAL	LOST PROPERTY	Friday	01/01/2016 12:00:00 AM	00:30	MISSION	NONE	16TH ST / MISSION ST	-122.419672	37.765050	(37.7650501214668, -122.419671780296)	16000274071000

2.1 Data Cleaning

Analizzando la composizione del dataset ho ricercato gli eventuali valori nulli presenti, ed ho riscontrato un singolo valore NaN in corrispondenza della variabile PdDistrict. Ho quindi eliminato il record corrispondente portando le osservazioni a 150499.

2.2 Esplorazione e analisi dei dati (EDA)

In *Figura 1* vengono riportate le varie categorie dei crimini commessi a San Francisco nel 2016 nelle varie aree della città (utilizzando latitudine e longitudine forniti nel dataset).

Figura 1- Categorie di crimini commessi del 2016

Nella Figura 2 si può notare che alcune categorie di crimine sono molto frequenti (come Larceny/Theft, Other Offenses), mentre altre sono molto rare (Pornography/Obscene Mat, Trea).

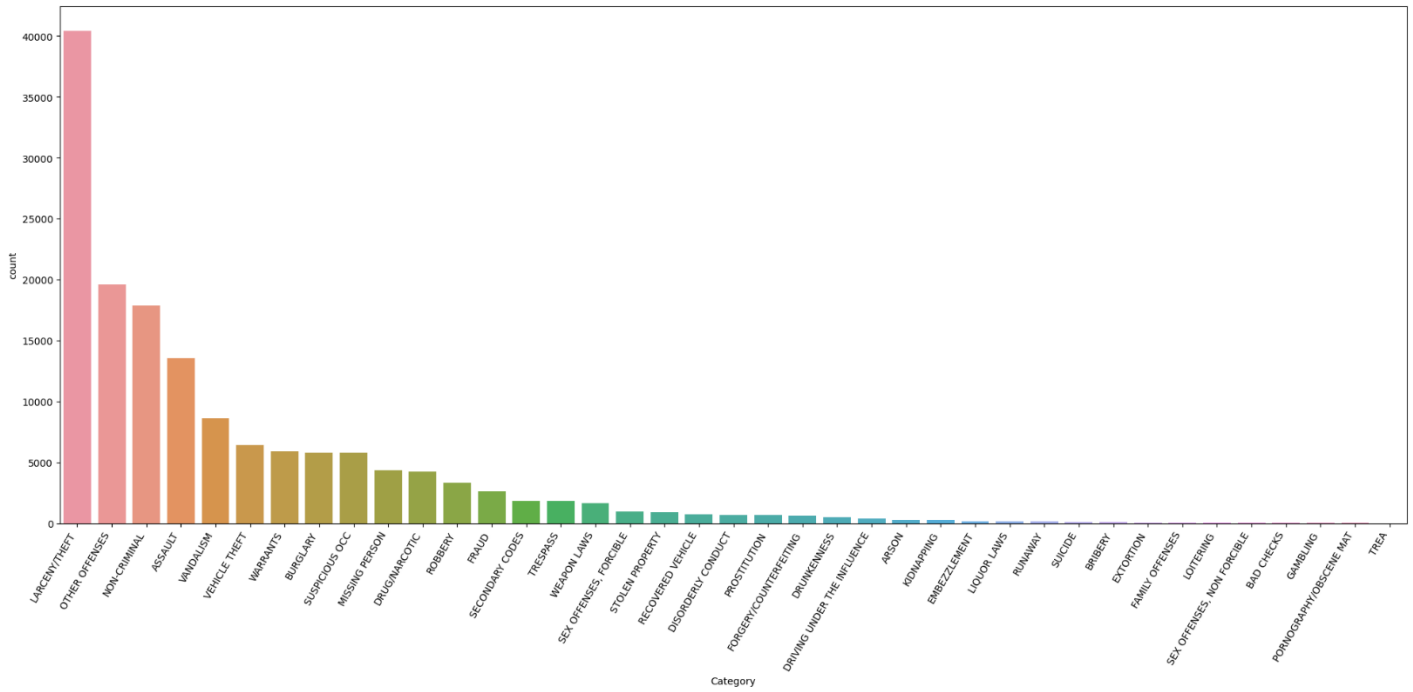

Figura 2 – Frequenza dei crimini

Nei vari distretti la frequenza dei crimini varia, pur rimanendo i furti (Larceny/Theft) la tipologia più diffusa. Dalla *Figura 3* si può notare che il numero dei crimini dipende molto dai distretti. Ad esempio, nel distretto Southern ne avvengono di più.

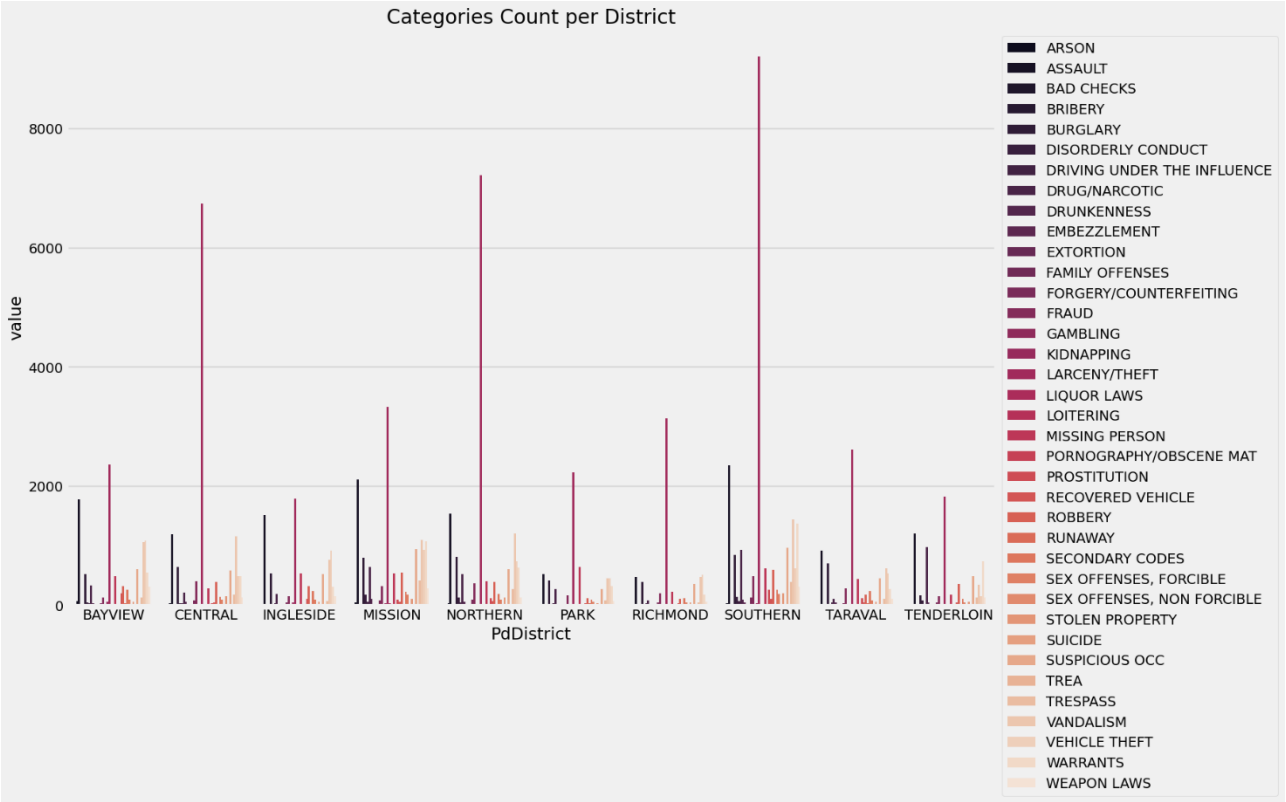

Figura 3 – Distribuzione dei crimini per distretto

In *Figura 4*, *Figura 5* e *Figura 6* si mostra l'andamento dei crimini in base ai mesi, ai giorni della settimana e all'ora del giorno.

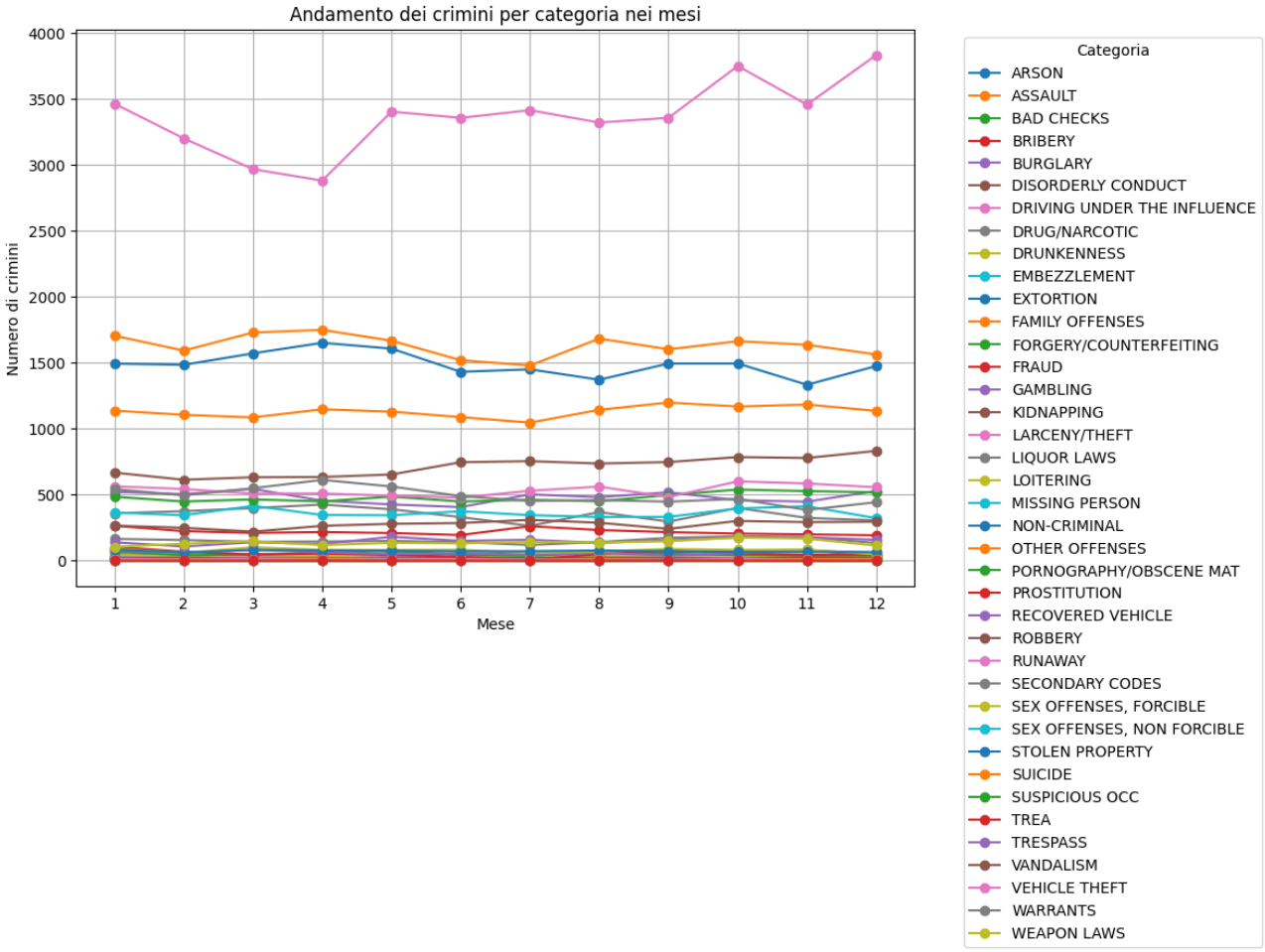

Figura 4 – Frequenza dei crimini nei mesi dell'anno

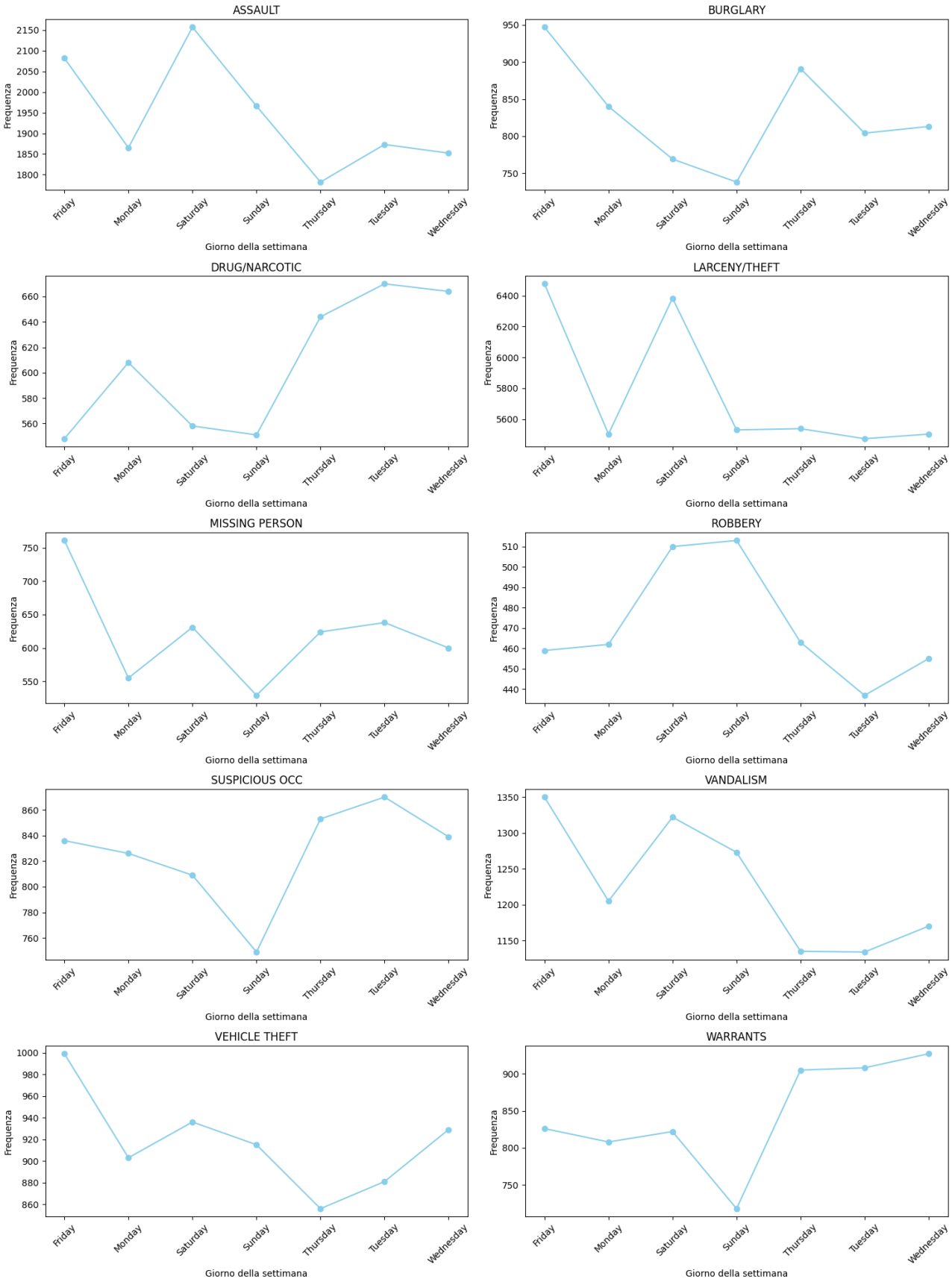

Figura 5 - Frequenza dei primi 10 crimini nei vari giorni della settimana

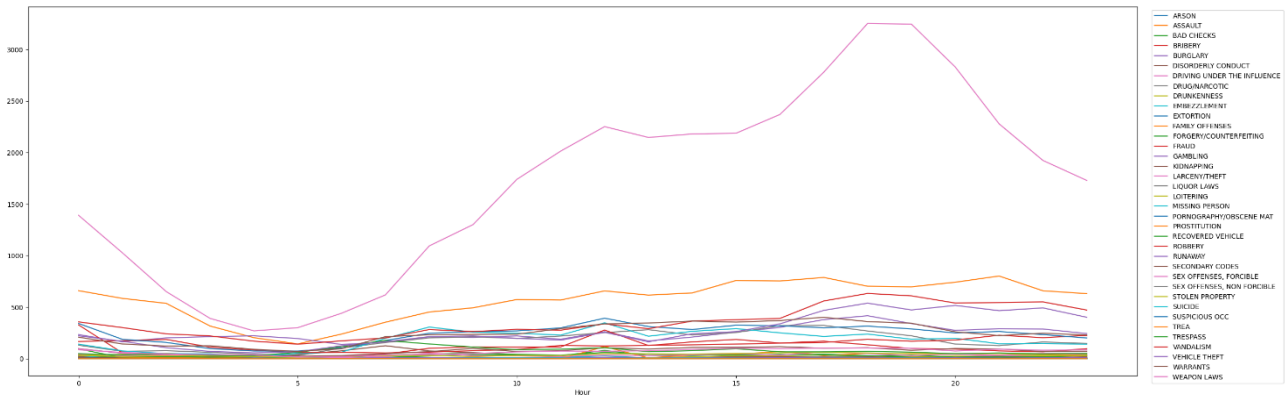

Figura 6 - Frequenza dei crimini nelle varie ore del giorno

2.3 Pre-Processamento dei dati

Per poter ottenere migliori risultati di classificazione procedo innanzitutto con il pre-processamento dei dati.

2.3.1 Features Engineering

Nella *Figura 6* si vede come la frequenza dei crimini dipenda dai vari periodi del giorno. Ho quindi generato una nuova feature 'DayPeriod' per dividere le ore del giorno in cinque periodi:

- Dopo mezzanotte, tra le 02:00 e le 08:00
- Mattina, tra le 08:00 e le 12:00
- Pomeriggio, tra le 12:00 e le 18:00
- Sera, tra le 18:00 e le 22:00
- Notte, tra le 22:00 e le 02:00

Dato che l'analisi per indirizzo risulta troppo puntuale, a partire dalla feature 'Address' ne ho generato una nuova, 'StreetType', contenente la tipologia di strada dove è avvenuto il crimine: avenues, streets, ways, boulevards, highways, courts, walks, plazas e incroci di strade (Indirizzi con /):

- INT
- ST
- AV
- BL
- DR
- WY
- PZ
- CT
- RD
- LN
- HY
- PL

- TR
- CR
- AL
- WAY

Ho quindi proceduto con la rimozione delle features ridondanti:

- 'Time' perchè ho considerato il periodo del giorno, ‘DayPeriod’.
- 'Location' , ‘X’ , ‘Y’ perché troppo di dettaglio rispetto a ‘PdDistrict’ e ‘StreetType’.
- 'Date', perché non c’è una grande dipendenza dal giorno e dal mese specifico.
- 'Descript' perchè fornisce solo ulteriori dettagli rispetto alla feature 'Category'.
- 'Address' perchè già considero la tipologia di indirizzo e il distretto (‘StreetType’ e ‘PdDistrict’).
- 'IncidntNum' e 'PdId' perché sono rispettivamente identificativi degli incidenti e del poliziotto.
- ‘Resolution’ perché la maggioranza dei record contiene valore ‘NONE’ (107779 su 150500).

Le features considerate per la classificazione sono quindi:

- ‘DayOfWeek'
- 'PdDistrict'
- 'Day_Period'
- 'StreetType'

2.3.2 Variabile target

Nel dataset esaminato risultano varie categorie che sono troppo generiche o non attinenti alla classificazione che si vuole effettuare:

- NON-CRIMINAL: Guardando la descrizione di questi record si vede che si tratta di eventi non criminali (come rapporti di decessi, incendi ecc.).
- OTHER OFFENSES: E’ una categoria troppo generica (aborto, cospirazione, cani che abbaiano ecc.).

Ho quindi rimosso i record con questi valori dal dataset, riducendo così le osservazioni da 150499 a 113034. Per il prosieguo di questo studio, prenderò in considerazione il comportamento delle restanti 37 categorie di reati.

Allo scopo di sviluppare un più accurato sistema di classificazione dei crimini e per evitare una categorizzazione troppo spinta, ho raggruppato (per similarità) le categorie di crimini nei tre gruppi più significativi:

- ***‘PropertyCrimes’*** (crimini relativi alla proprietà***)***: ’LARCENY/THEFT’, ‘VEHICLE THEFT’, ‘VANDALISM’, ‘ROBBERY’, ‘STOLEN PROPERTY’, ‘ARSON’, ‘RECOVERED VEHICLE’, ‘EMBEZZLEMENT’, ‘FRAUD’, ‘BURGLARY’, ‘EXTORTION’
- ***‘ViolentCrimes’*** (crimini violenti): ‘ASSAULT’, ’KIDNAPPING’, ‘SEX OFFENSES, FORCIBLE’, ‘SECONDARY CODES’, ‘SUICIDE’
- ***‘QoLCrimes’***(crimini relativi allo stile di vita): ‘LOITERING’, ’PROSTITUTION’, ’LIQUOR LAWS’, ’TRESPASS’, ’TREA’, ‘DRUG/NARCOTIC’, ’DRIVING UNDER THE INFLUENCE’,

‘DRUNKENNESS’, ‘FAMILY OFFENSES’, ‘SEX OFFENSES, NON FORCIBLE’, ‘GAMBLING’, ‘DISORDERLY CONDUCT’

Ho infine eliminato i record con le seguenti categorie perché troppo generici o variegati e quindi difficilmente raggruppabili:

- ‘SUSPICIOUS OCC’
- ’MISSING PERSON’
- ’FORGERY/COUNTERFEITING’
- ’RUNAWAY’
- ’BRIBERY’
- ’BAD CHECKS’
- ’PORNOGRAPHY/OBSCENE MAT’
- ‘WARRANTS’
- ‘WEAPON LAWS’

In questo modo le osservazioni su cui verranno addestrati i modelli di machine learning diventano 94479.

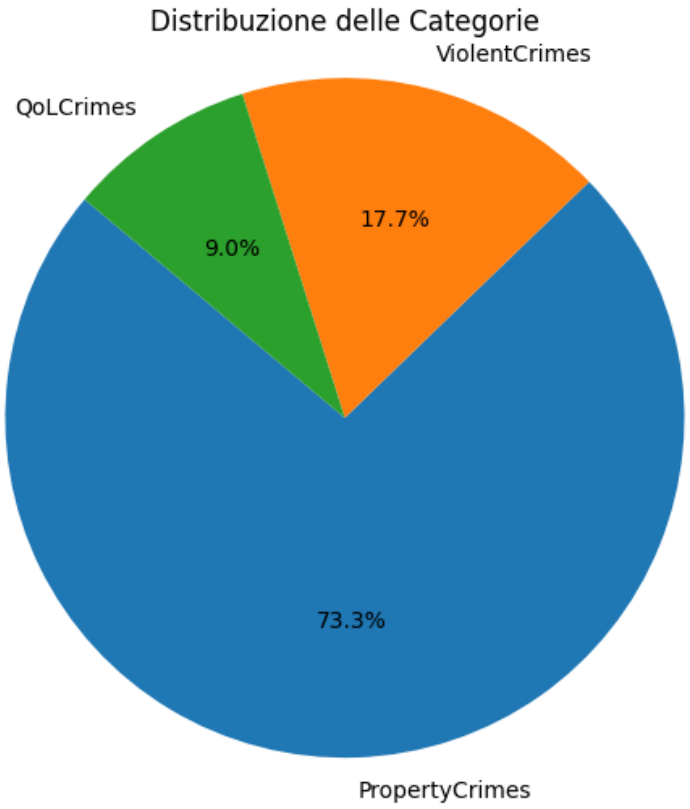

Figura 7 - Percentuali dei vari gruppi di categorie

2.3.3 Encoding

Un modello di machine learning ha bisogno che i dati raccolti siano forniti in formato numerico. Ho quindi applicato due tecniche di encoding.

Ho applicato OneHotEncoder sulle variabili indipendenti (X) di interesse per il mio studio:

- 'Day_Period'
- 'DayOfWeek'
- 'PdDistrict'
- 'StreetType'

La tecnica ***One-Hot Encoding*** (anche OHE) è una delle tecniche di codifica più utilizzate nel machine learning per le variabili di tipo categorico. Essa crea tante colonne quanti sono i valori della variabile, ognuna con il nome del valore: si assegna 1 alla colonna il cui nome coincide con il valore della variabile e 0 negli altri casi, creando così dei vettori binari.

Sulla variabile target, Category (Y), ho applicato invece il ***Label Encoder***, particolarmente efficiente per i modelli ad albero.

Label	**Category**
0	'PropertyCrimes'
1	'QoLCrimes':
2	'ViolentCrimes'

2.3.4 Feature Scaling

Ho adottato la standardizzazione come metodo di feature scaling. Essa centra la media di ogni feature a 0 e i dati sono scalati in modo che la distribuzione dei punti per ogni feature ha deviazione standard pari a 1. Questa tecnica di preprocessing è utile per le Neural Network.

3. Modelli di Machine Learning e Metriche

I modelli di machine learning utilizzati per la classificazione sono:

- Decision Tree
- Random Forest
- Logistic Regression
- Gaussian Naive Bayes
- XGboost
- Multi-Layer Perceptron, rete neurale di tipo feedforward

3.1 Decision Tree

Il Decision tree o albero di classificazione è un albero in cui ciascun nodo interno (non foglia) è etichettato con una funzionalità di input. Gli archi provenienti da un nodo etichettato con una caratteristica di input sono etichettati con ciascuno dei possibili valori della caratteristica di destinazione oppure l'arco conduce a un nodo decisionale subordinato su una caratteristica di input diversa. Ogni foglia dell'albero è etichettata con una classe o una distribuzione di probabilità sulle classi, a significare che l'insieme di dati è stato classificato dall'albero in una classe specifica o in una particolare distribuzione di probabilità (che, se l'albero decisionale è ben costruito, è sbilanciato verso determinati sottoinsiemi di classi).

Un albero viene costruito suddividendo l'insieme di origine, che costituisce il nodo radice dell'albero, in sottoinsiemi, che costituiscono i figli successori. La suddivisione si basa su una serie di regole di suddivisione basate su caratteristiche di classificazione. Questo processo viene ripetuto su ciascun sottoinsieme derivato in modo ricorsivo chiamato partizionamento ricorsivo. La ricorsione viene completata quando il sottoinsieme in un nodo ha tutti gli stessi valori della variabile di destinazione o quando la suddivisione non aggiunge più valori alle previsioni. Questo processo di induzione top-down degli alberi decisionali (TDIDT) è un esempio di algoritmo greedy ed è di gran lunga la strategia più comune per applicare alberi decisionali ai dati.

3.2 Random Forest

L'algoritmo Random Forest è un'estensione del metodo di bagging in quanto utilizza sia il bagging che la casualità delle caratteristiche per creare una foresta non correlata di strutture ad albero decisionali. La casualità delle caratteristiche, nota anche come bagging delle caratteristiche o "metodo del sottospazio casuale", genera un sottoinsieme casuale di caratteristiche, che garantisce una bassa correlazione tra le strutture ad albero decisionali. Si tratta di una differenza fondamentale tra strutture ad albero decisionali e Random Forest. Mentre le strutture ad albero decisionali considerano tutte le possibili suddivisioni delle caratteristiche, gli algoritmi Random Forest selezionano solo un sottoinsieme di queste caratteristiche.

3.3 Logistic Regression

La regressione logistica multinomiale è un metodo di classificazione che generalizza la regressione logistica a problemi multiclasse, ovvero con più di due possibili risultati discreti. Esso è un modello utilizzato per prevedere le probabilità dei diversi possibili risultati di una variabile dipendente distribuita categoricamente, dato un insieme di variabili indipendenti. Si tratta di una soluzione particolare ai problemi di classificazione che utilizza una combinazione lineare delle features osservate e alcuni parametri specifici del problema per stimare la probabilità di ciascun valore particolare della variabile dipendente. I migliori valori dei parametri per un dato problema sono solitamente determinati da alcuni dati di training.

3.4 Gaussian Naive Bayes

L'algoritmo Gaussian Naive Bayes applica la regola di Bayes che assume l'indipendenza tra le features. La probabilità congiunta è quindi il prodotto delle probabilità condizionali di ogni feature.

L'algoritmo è composto da due fasi:

- Fase di apprendimento

Per ogni valore di uscita (classe tra k classi) si stima la probabilità di quella classe mediante le istanze nel dataset S. Per ogni valore dell'attributo si stima la probabilità condizionale mediante le istanze nel dataset S

- Fase di applicazione

Per un'istanza incognita con m attributi si calcolano le probabilità condizionali per tutte le classi. Si attribuisce l'istanza alla classe che presenta la probabilità condizionale maggiore

3.5 XGboost

XGBoost, che sta per Extreme Gradient Boosting, è una libreria di apprendimento automatico con albero decisionale gradient-boost (GBDT) scalabile e distribuito. Fornisce il potenziamento degli alberi paralleli ed è la principale libreria di machine learning per problemi di regressione e classificazione. Per comprendere XGBoost è fondamentale comprendere innanzitutto i concetti e gli algoritmi di apprendimento automatico su cui si basa XGBoost: apprendimento automatico supervisionato, alberi decisionali, apprendimento di insieme e potenziamento del gradiente.

Un Gradient Boosting Decision Trees (GBDT) è un algoritmo di apprendimento dell'insieme di alberi decisionali simile alla Random Forest, per la classificazione. Gli algoritmi di apprendimento d'insieme (ensemble learning) combinano più algoritmi di apprendimento automatico per ottenere un modello migliore.

3.6 Multi-Layer Perceptron

Multi-Layer Perceptron è il nome di una moderna rete neurale artificiale feedforward, costituita da neuroni completamente connessi con una funzione di attivazione di tipo non lineare, organizzata in almeno tre strati, nota per essere in grado di distinguere dati che non sono linearmente separabili. È un termine improprio perché il percettrone originale utilizzava una funzione di gradino di Heaviside, invece di una funzione di attivazione di tipo non lineare (utilizzata dalle reti moderne).

Le moderne reti feedforward vengono addestrate utilizzando il metodo di backpropagation e vengono colloquialmente chiamate reti neurali "vanilla".

3.7 Metriche

I modelli di classificazione utilizzati saranno valutati secondo le seguenti metriche così definite:

- Accuracy = (TP + TN) / (TP + FP + TN + FN).
- Sensitivity = Recall = TP/t = TP / (TP + FN).
- Precision = TP/p = TP / (TP + FP).
- F1 Score = 2 * (precision * recall) / (precision + recall).

Dove nella matrice di confusione:

- TP è vero positivo
- TN è vero negativo
- FP è falso positivo
- FN è falso negativo

In questo contesto la precisione si riferisce alla percentuale effettiva di criminalità prevista dal modello di classificazione. D'altro canto, la recall misura la percentuale di individuazione corretta dei casi positivi. La F1 Score è una misura della precisione e della recall bilanciati (media armonica tra precisione e recall). L'accuracy è una misura della percentuale di predizioni corrette rispetto al totale delle predizioni effettuate dal modello.

4. Risultati dei vari modelli

Il dataset è stato suddiviso in due parti: 75% per il training e 25% per il test.

4.1 Dataset senza bilanciamento

4.1.1 Cross-Validation

La Cross-Validation è una tecnica per valutare i modelli ML addestrandoli su sottoinsiemi dei dati in ingresso disponibili e valutandoli sulla base di un sottoinsieme complementare dei dati. È stato utilizzato il metodo k-fold cross-validation a 5 fold. Il modello viene addestrato su 4/5 dei dati e testato su 1/5 dei dati, questo processo viene ripetuto 5 volte utilizzando un diverso insieme di dati per il test ogni volta.

I risultati ottenuti sono riportati nella seguente tabella:

Modello	**Media dei risultati Cross-Validation**
DecisionTreeClassifier	0.73
GaussianNB	0.13
RandomForestClassifier	0.73
LogisticRegression	0.73
XGBClassifier	0.73

Il peggior risultato è stato prodotto dall'algoritmo di classificazione GaussianNB che ha ottenuto il punteggio medio più basso, mentre i restanti modelli utilizzati hanno ottenuto una buona valutazione media.

In base agli esiti elencati, sono andato a studiare ogni singolo modello in base alle metriche spiegate nel paragrafo 3.7 cercando di trovare le caratteristiche che li contraddistinguono.

Di seguito i risultati dei modelli.

4.1.2 Random Forest

4.1.2.1 Risultati con parametri standard

Utilizzando tutti i parametri di default il classificatore ha restituito i seguenti risultati:

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.7401	0.732	0.5486	0.3558	0.3307

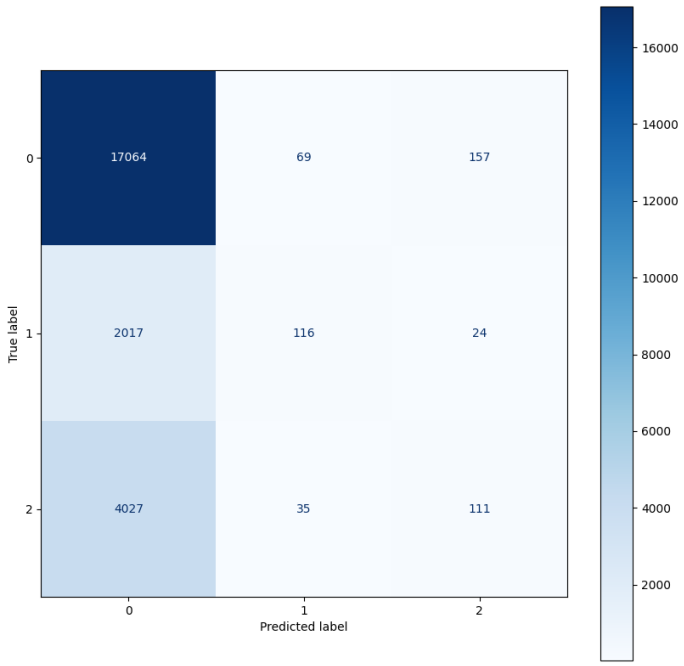

Figura 8- Matrice di confusione Random Forest

Di seguito vengono riportate invece le metriche per ogni singola classe:

Classi	Precision	Recall	F1 Score
0	0.74	0.99	0.84
1	0.53	0.05	0.10
2	0.38	0.03	0.05

I risultati ottenuti, per quanto riguarda l'accuracy, risultano essere buoni. La precisione, al 50%, indica che la metà delle predizioni positive fatte dal modello è stata corretta rispetto al totale delle predizioni positive. Guardando la matrice di confusione possiamo notare però come la classe 'ViolentCrimes'(2) e 'QoLCrimes'(1) vengano spesso predette erroneamente come 'PropertyCrimes'(0).

4.1.2.2 Hyperparameter Tuning

L'ottimizzazione degli iperparametri o tuning è il problema di scegliere un insieme di iperparametri ottimali per un algoritmo di apprendimento. Un iperparametro è un parametro il cui valore è utilizzato per controllare il processo di apprendimento.

L'ottimizzazione degli iperparametri trova una tupla di iperparametri che produce un modello ottimale e minimizza una funzione di perdita predefinita su dati indipendenti forniti. La funzione obiettivo prende una tupla di iperparametri e restituisce la perdita associata.

Saranno stimati i migliori valori sui seguenti parametri:

- 'n_estimators ' : numero di alberi nella foresta.
- 'max_features' : numero massimo di features considerate per la suddivisione di un nodo.
- 'min_samples_leaf' : numero minimo di data points consentiti in un nodo foglia.

4.1.2.2.1 GridSearchCV

La GridSearch è una tecnica utilizzata nel machine learning per trovare i migliori iperparametri di un modello e consiste nel definire una griglia di valori per ogni iperparametro, testando tutte le possibili combinazioni di valori.

Sono stati stimati i migliori valori dei parametri specificati precedentemente tra quelli compresi tra parentesi quadre:

- 'n_estimators ' : : [100,200,300,500]
- 'max_features' : ['sqrt', 'log2', None]
- 'min_samples_leaf' : [1,10,20]

I parametri per cui si ottiene un aumento, seppur lieve, dell'accuracy sono i seguenti: *{'max_features': 'sqrt', 'min_samples_leaf': 10, 'n_estimators': 500}*.

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.736	0.734	0.61	0.3448	0.306

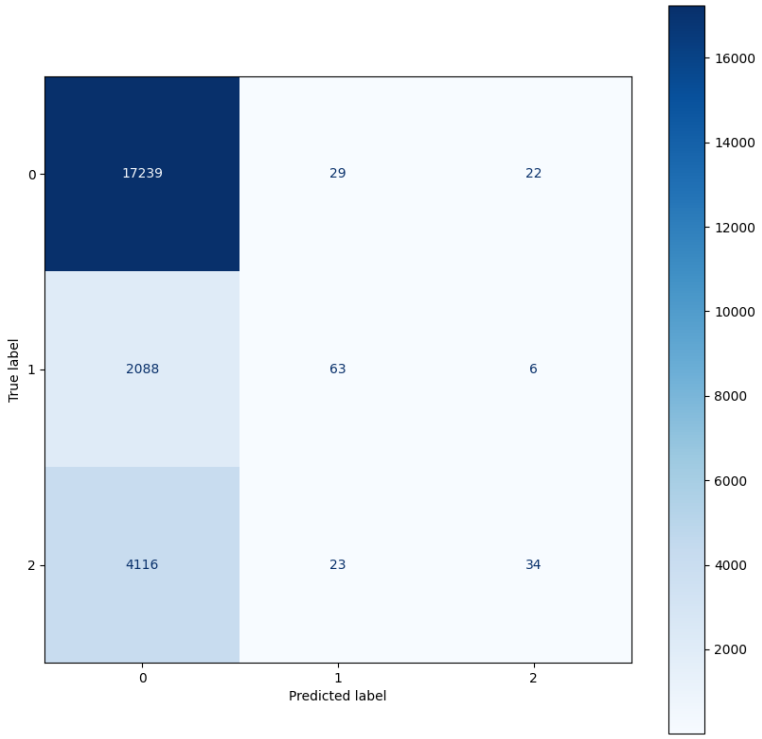

Figura 9- Matrice di confusione Random Forest con GridSearchCV

Di seguito le metriche per singole classi:

Classi	Precision	Recall	F1 Score
0	0.74	1.00	0.85
1	0.55	0.03	0.06
2	0.55	0.01	0.02

E’ aumentata rispetto al modello con i parametri di default la Precision per le classi ‘ViolentCrimes’(2) e ‘QoLCrimes’(1). Da notare inoltre come la Recall per la classe ‘PropertyCrimes’ (0) sia massima, cioè che non ci sono falsi negativi e il modello è riuscito quindi a identificare tutti i TP.

4.1.2.2.2 BayesSearchCV

Questo approccio utilizza l'ottimizzazione bayesiana graduale per esplorare gli iperparametri più promettenti nello spazio del problema.

In breve, l'ottimizzazione bayesiana trova il minimo di una funzione obiettivo in ampi spazi problematici ed è molto adatta a valori continui. Per fare ciò utilizza la regressione del processo gaussiano sulla funzione obiettivo. Nel nostro caso la funzione obiettivo è arrivare al miglior risultato del modello dati con i parametri del modello che specifichiamo.

L'approccio di ottimizzazione bayesiana offre il vantaggio di poter fornire una gamma molto più ampia di valori possibili, poiché nel tempo esploriamo automaticamente le regioni più promettenti e scartiamo quelle meno promettenti.

Una semplice ricerca su griglia, come svolta da GridSearch, richiederebbe secoli per esplorare tutti i possibili valori. Poiché ci muoviamo in modo molto più efficace, possiamo consentire un range di valori molto più ampio

In questo caso sono stati stimati i migliori valori passando gli stessi parametri e intervalli utilizzati per la GridSearch in modo tale da capire se ci siano differenze, oltre che di tempo computazionale, anche nelle scelte.

I parametri torvati sono i seguenti:

{('max_features', None), ('min_samples_leaf', 20), ('n_estimators', 500)}

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.7365	0.7336	0.58	0.3503	0.318

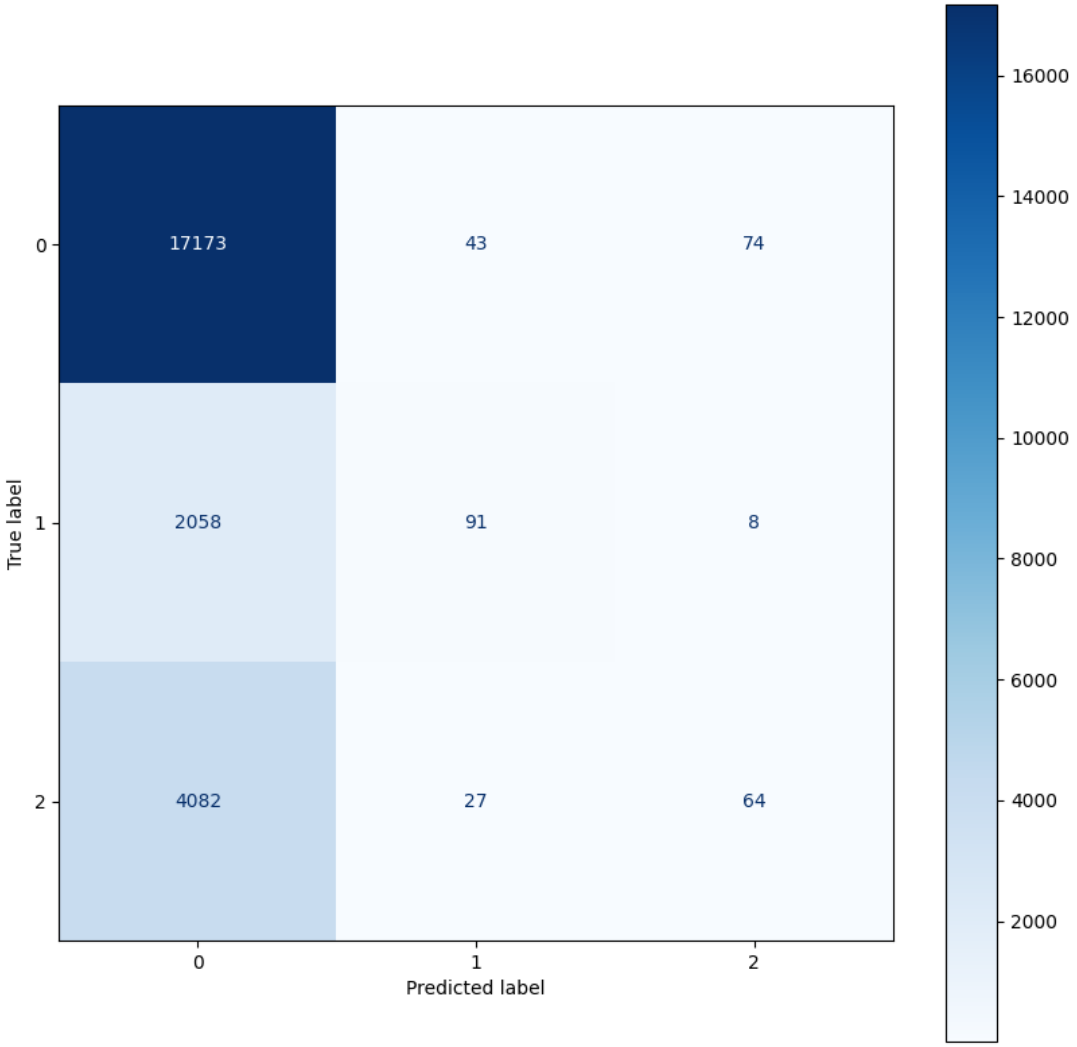

Figura 10- Matrice di confusione Decision Tree con BayesSearch

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.74	0.99	0.85
1	0.57	0.04	0.08
2	0.44	0.02	0.03

4.1.3 Decision Tree

4.1.3.1 Risultati con parametri standard

Utilizzando tutti i parametri di default il classificatore ha restituito i seguenti risultati:

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.7401	0.7328	0.5581	0.3551	0.3288

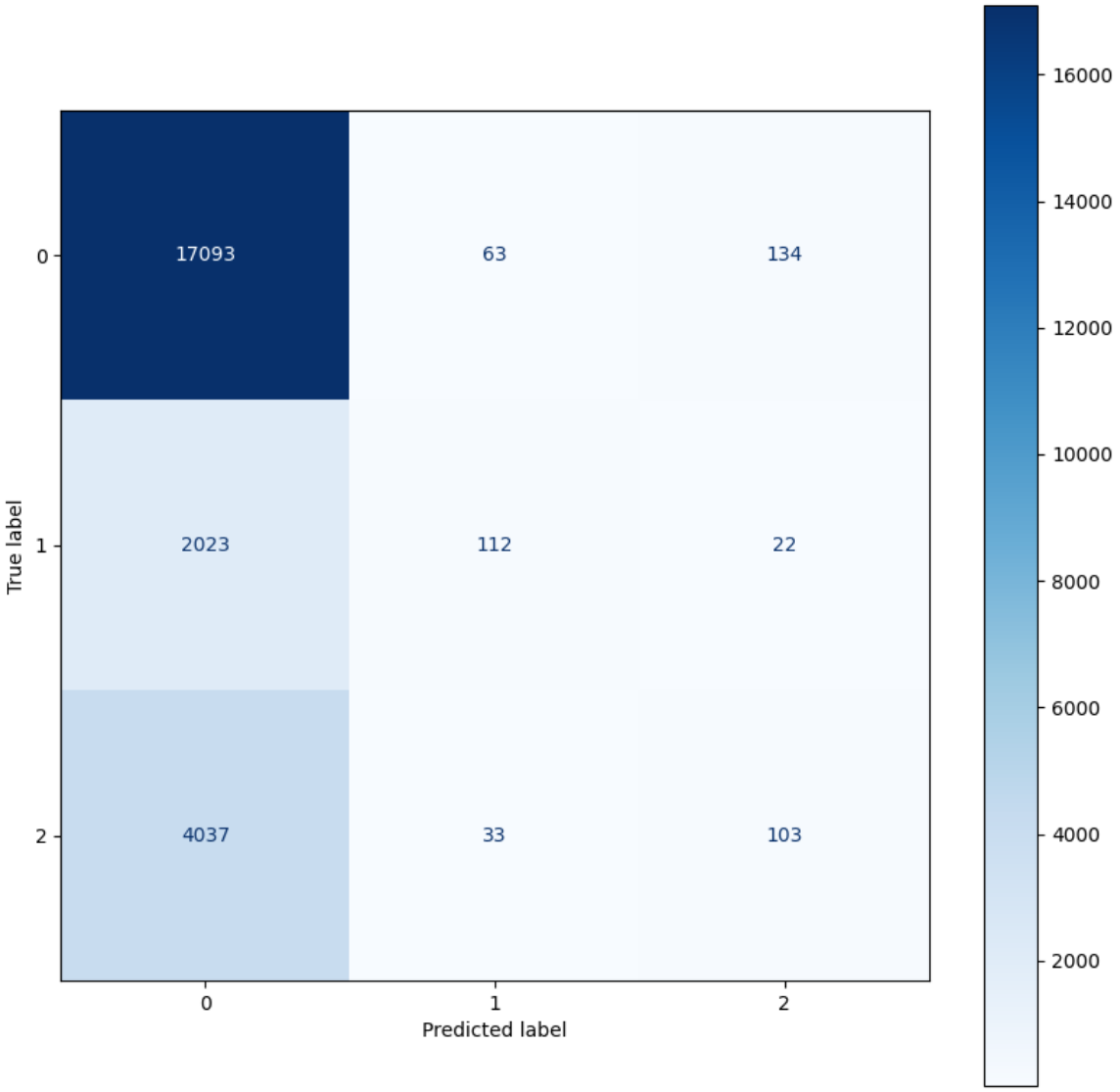

Figura 11- Matrice di confusione Decision Tree

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.74	0.99	0.85
1	0.57	0.05	0.09
2	0.40	0.02	0.05

I risultati sono abbastanza buoni considerando l'accuracy. Discreti invece per la precision (intorno al 55%). I risultati e le caratteristiche sono molto simili al modello utilizzato precedentemente.

4.1.3.2 Hyperparameter Tuning

Sono stati stimati i migliori valori, tra quelli presenti, dei seguenti parametri:

- 'max_features'
- 'criterion'

- 'min_samples_leaf'

4.1.3.2.1 GridSearchCV

Sono stati stimati i migliori valori dei parametri specificati precedentemente tra quelli compresi tra parentesi quadre:

- 'max_features' : ['sqrt', 'log2', None]
- 'criterion' : ['gini' , 'entropy'],
- 'min_samples_leaf': [1,5,10,20]

I parametri per cui si ottiene, seppur lieve, un aumento dell'accuracy sono i seguenti: *{'criterion': 'gini', 'max_features': 'log2', 'min_samples_leaf': 20}*.

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.7352	0.732	0.5425	0.3431	0.3043

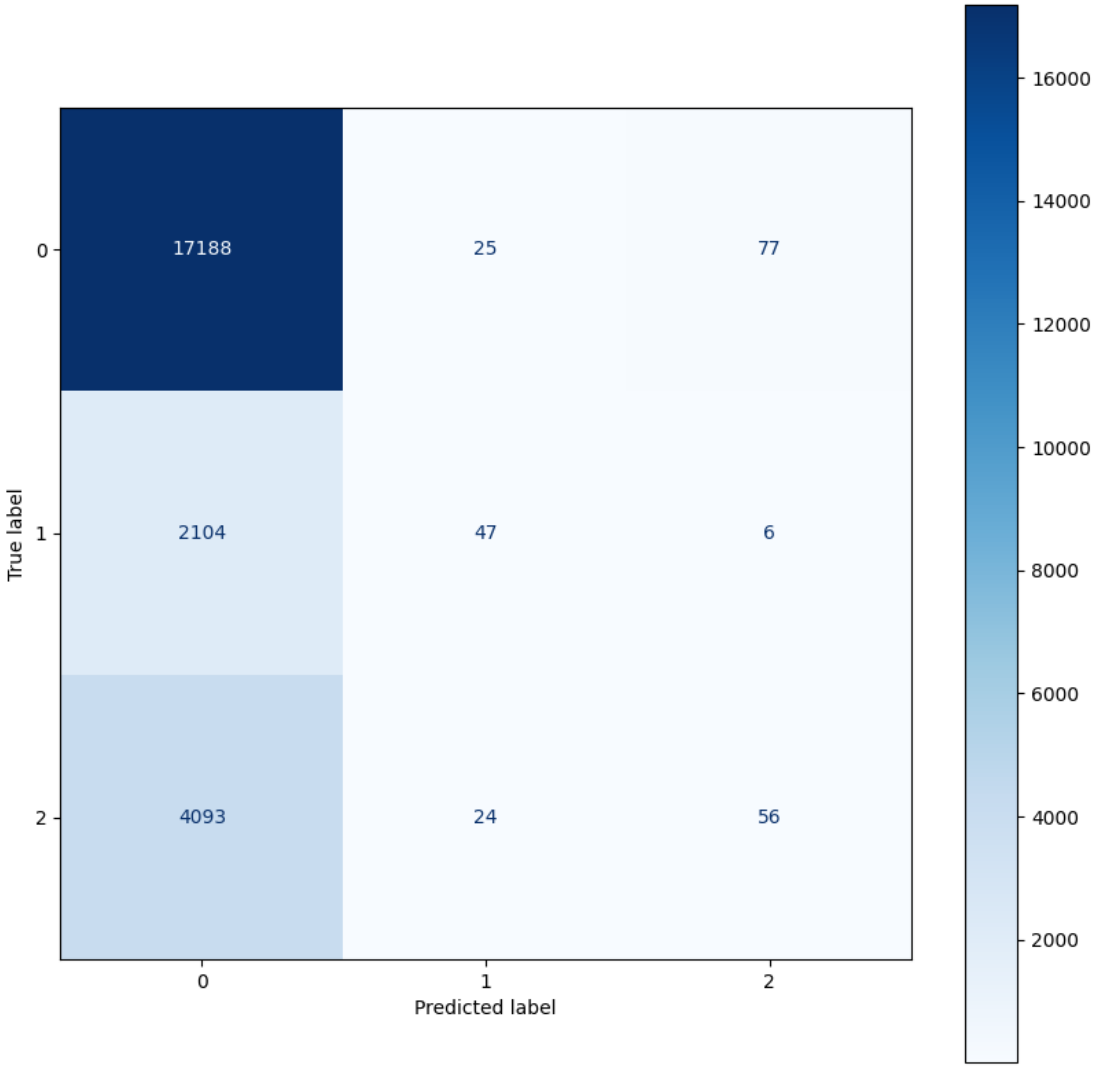

Figura 12- Matrice di confusione Decision Tree con GridSearchCV

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.74	0.99	0.85
1	0.49	0.02	0.04
2	0.40	0.01	0.03

4.1.3.2.2 BayesSearchCV

Sono stati stimati i migliori valori passandogli gli stessi parametri e intervalli utilizzati per la GridSearch in modo tale da capire se ci siano differenze, oltre che di tempo computazionale, anche nelle scelte.

I parametri trovati sono i seguenti*:*

{('criterion', 'gini'), ('max_features', 'log2'), ('min_samples_leaf', 20)}

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.7351	0.7331	0.569	0.3459	0.3094

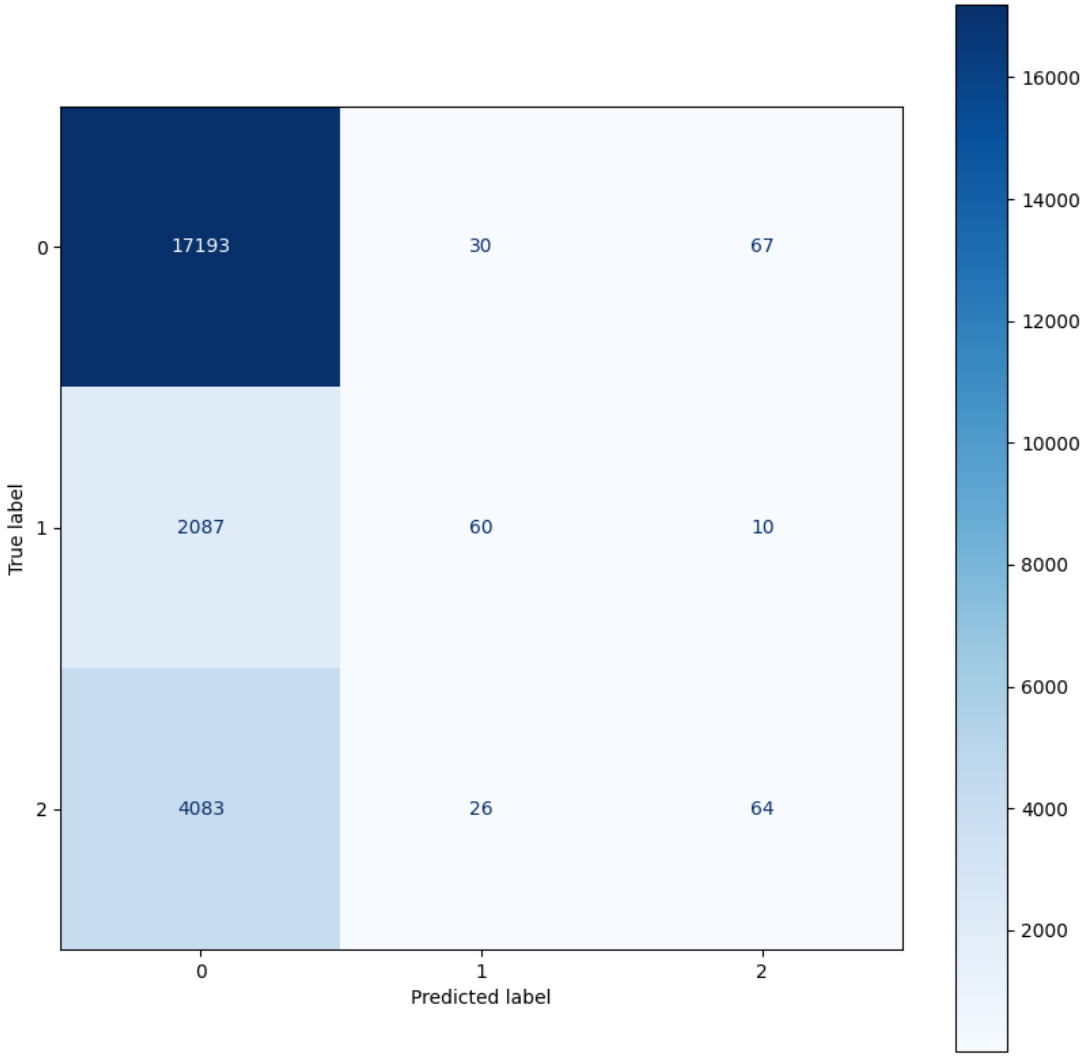

Figura 13 -- Matrice di confusione Decision Tree con BayesSearch

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.74	0.99	0.85
1	0.52	0.03	0.05
2	0.45	0.02	0.03

C'è stato un lieve miglioramento nell'accuracy pur rimanendo le difficoltà nel classificare correttamente le classi che non sono 'PropertyCrimes' (0).

4.1.4 Gaussian Naive Bayes

Utilizzando tutti i parametri di default il classificatore ha restituito i seguenti risultati:

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.1542	0.1542	0.4214	0.3543	0.123

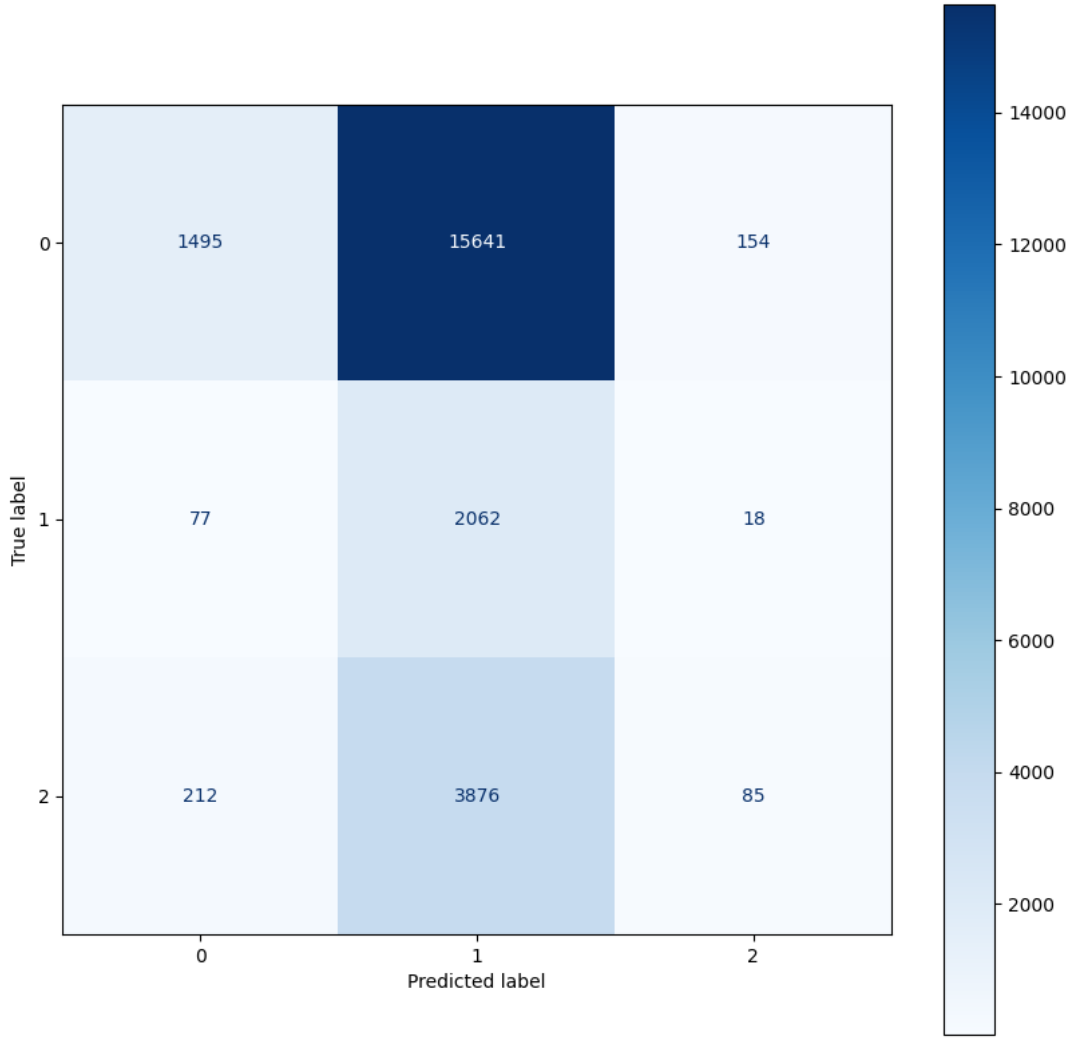

Figura 14- Matrice di confusione Gaussian Naive Bayes

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.84	0.09	0.16
1	0.10	0.96	0.17
2	0.33	0.02	0.04

I risultati sono estremamente negativi rispetto ai modelli precedenti. Ciò significa che questo modello non predirà quasi mai nella maniera corretta, risultando così il modello meno affidabile tra quelli descritti finora. Si evidenzia inoltre come la classe 'QoLCrimes' (1) abbia il 96% di recall, ossia le predizioni fatte su quella classe sono sempre positive, anche se non appartengono alla categoria, tant'è che per 15641 volte la classe 'Property Crimes' (0) viene 'scambiata' per QoLCrimes' (1).

4.1.5 Logistic Regression

Utilizzando tutti i parametri di default il classificatore ha restituito i seguenti risultati:

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.7343	0.7326	0.6162	0.3374	0.291

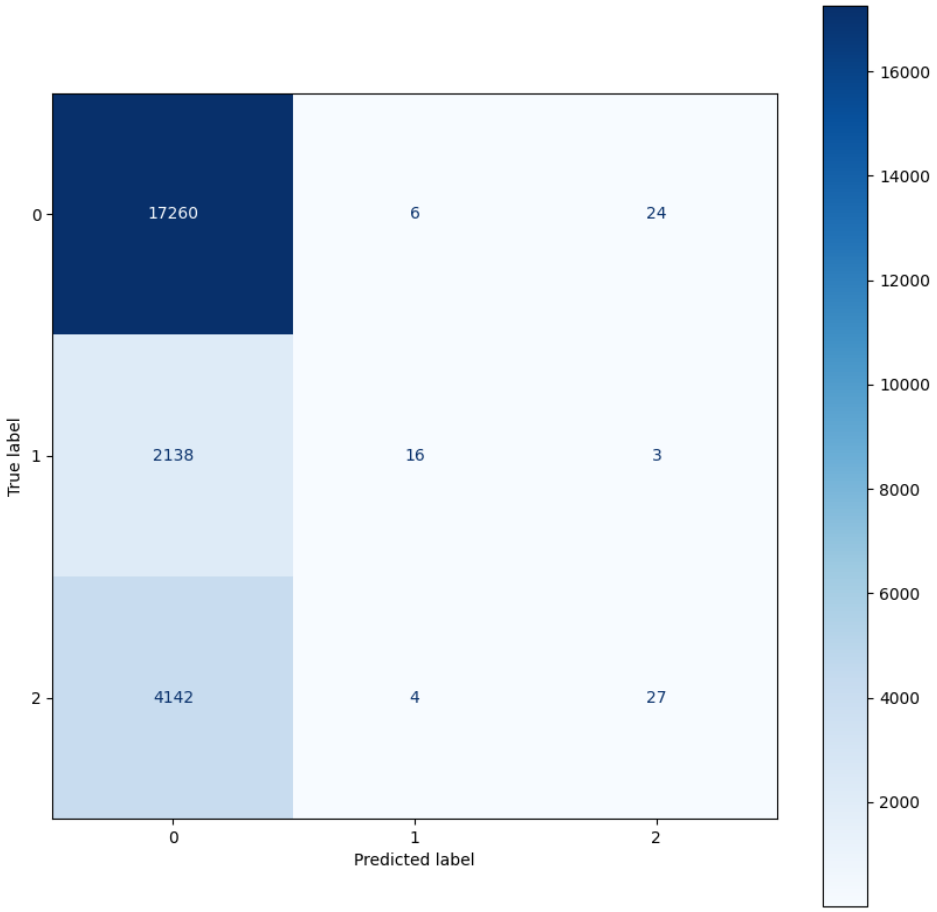

Figura 15- Matrice di confusione Logistic Regression

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.73	1.00	0.85
1	0.62	0.01	0.01
2	0.50	0.01	0.01

4.1.6 XGBoost

Utilizzando tutti i parametri di default il classificatore ha restituito i seguenti risultati:

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.7382	0.7339	0.5942	0.3485	0.3143

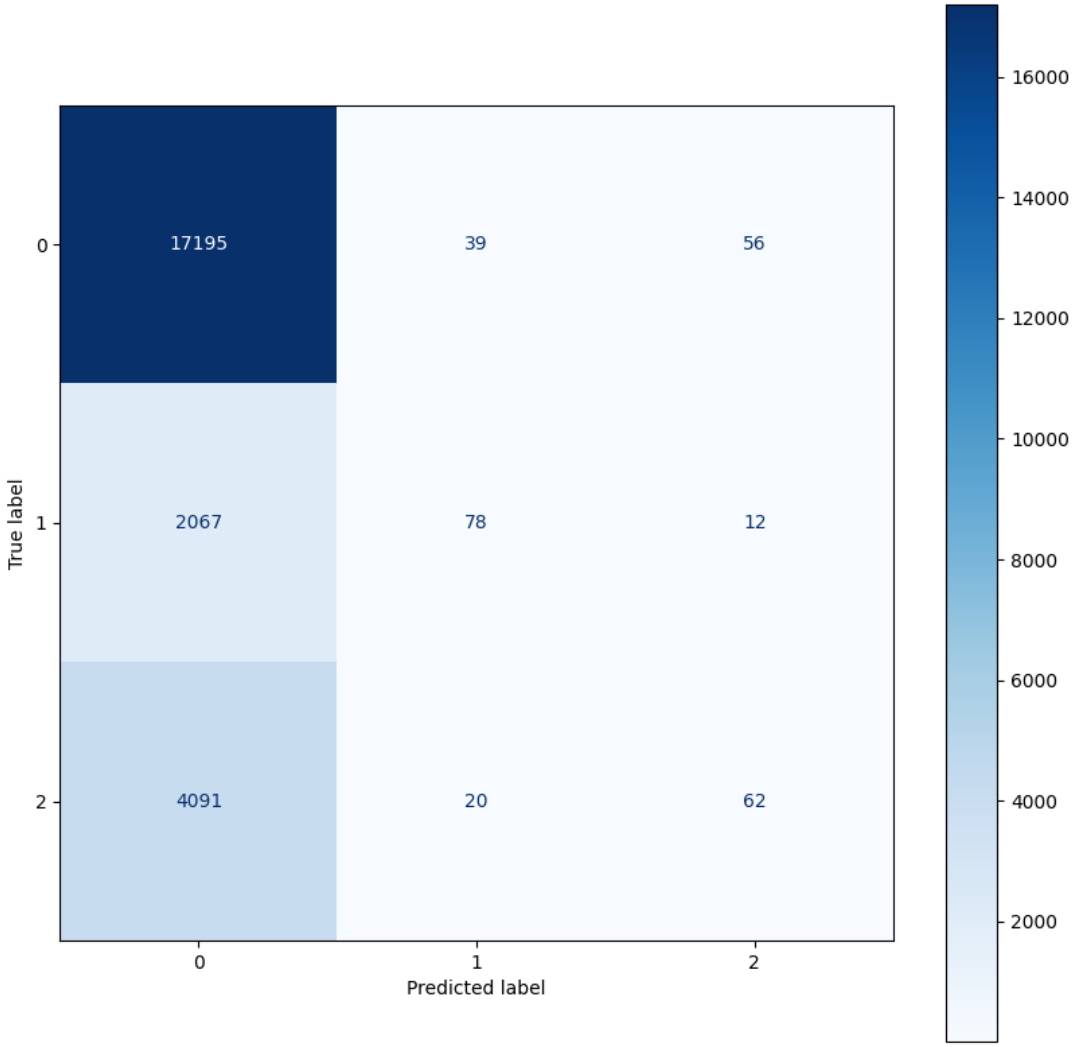

Figura 16 - Matrice di confusione XGBoost

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.74	0.99	0.85
1	0.57	0.04	0.04
2	0.48	0.01	0.03

4.1.7 Neural Network

4.1.7.1 Hyperparameter Tuning con 1 layer intermedio

Sono stati stimati i migliori valori dei seguenti parametri tramite GridSearchCV:

- 'hidden_layer_sizes': [(10,),(20,),(50,), (100,)]
- 'activation': ['tanh', 'relu']
- 'solver': ['sgd', 'adam','lbfgs']
- 'alpha': [0.0001, 0.05]

- 'learning_rate': ['constant','adaptive']

I parametri trovati sono i seguenti:

{'activation': 'relu', 'alpha': 0.05, 'hidden_layer_sizes': (100,), 'learning_rate': 'adaptive', 'solver': 'adam'}

I risultati ottenuti, inserendo i migliori parametri sono stati:

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.7368	0.7339	0.5895	0.3485	0.3135

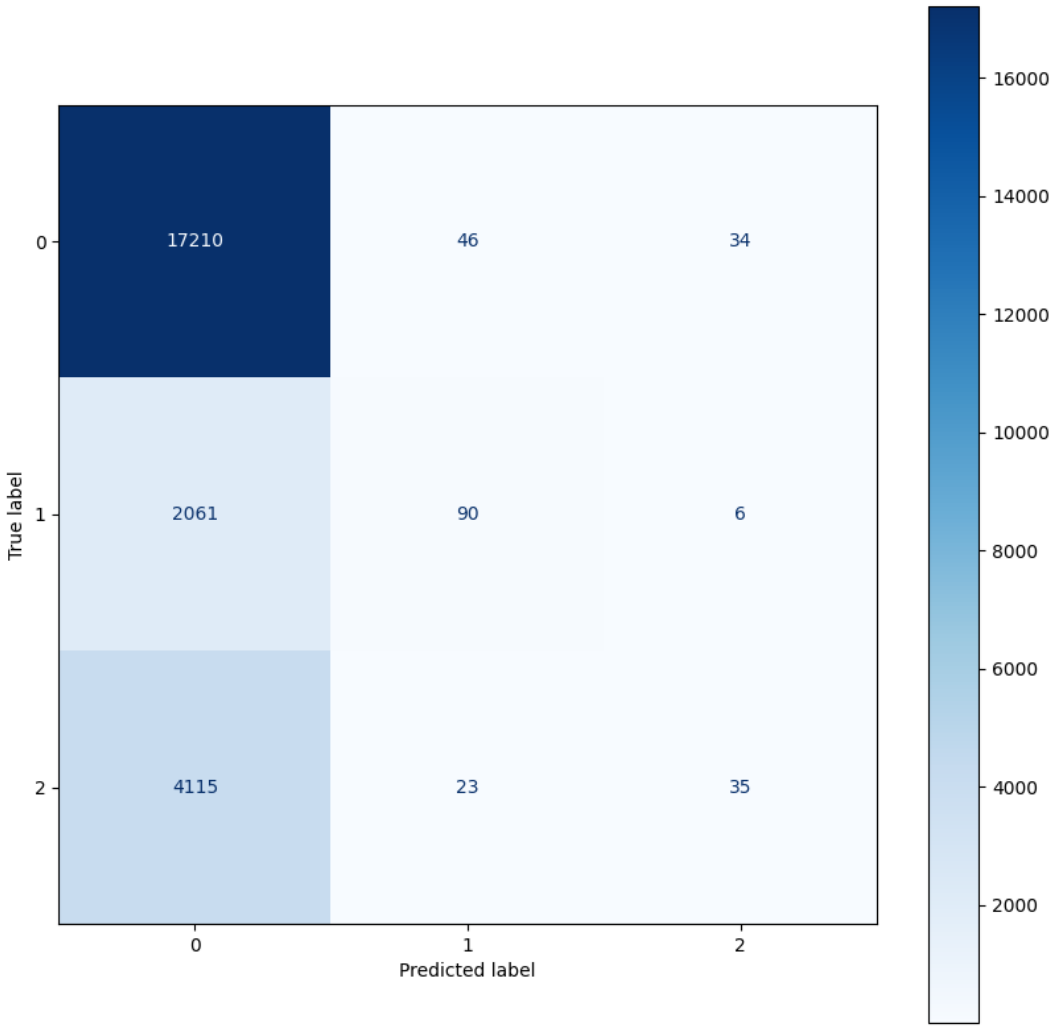

Figura 17 - Matrice di confusione Multilayer Perceptron con 1 strato intermedio

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.74	1.00	0.85
1	0.57	0.04	0.08
2	0.47	0.01	0.02

4.1.7.2 Hyperparameter Tuning con 2 layer intermedi

Sono stati stimati i migliori valori dei seguenti parametri tramite GridSearchCV:

- 'hidden_layer_sizes': [(10,10),(20,20),(50,50)]
- 'activation': ['tanh', 'relu']
- 'solver': ['sgd', 'adam','lbfgs']
- 'alpha': [0.0001, 0.05]
- 'learning_rate': ['constant','adaptive']

I parametri trovati sono i seguenti:

{'activation': 'relu', 'alpha': 0.05, 'hidden_layer_sizes': (50, 50), 'learning_rate': 'constant', 'solver': 'adam'}

I risultati ottenuti, inserendo i migliori parametri sono stati:

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.7372	0.7341	0.5858	0.3502	0.3171

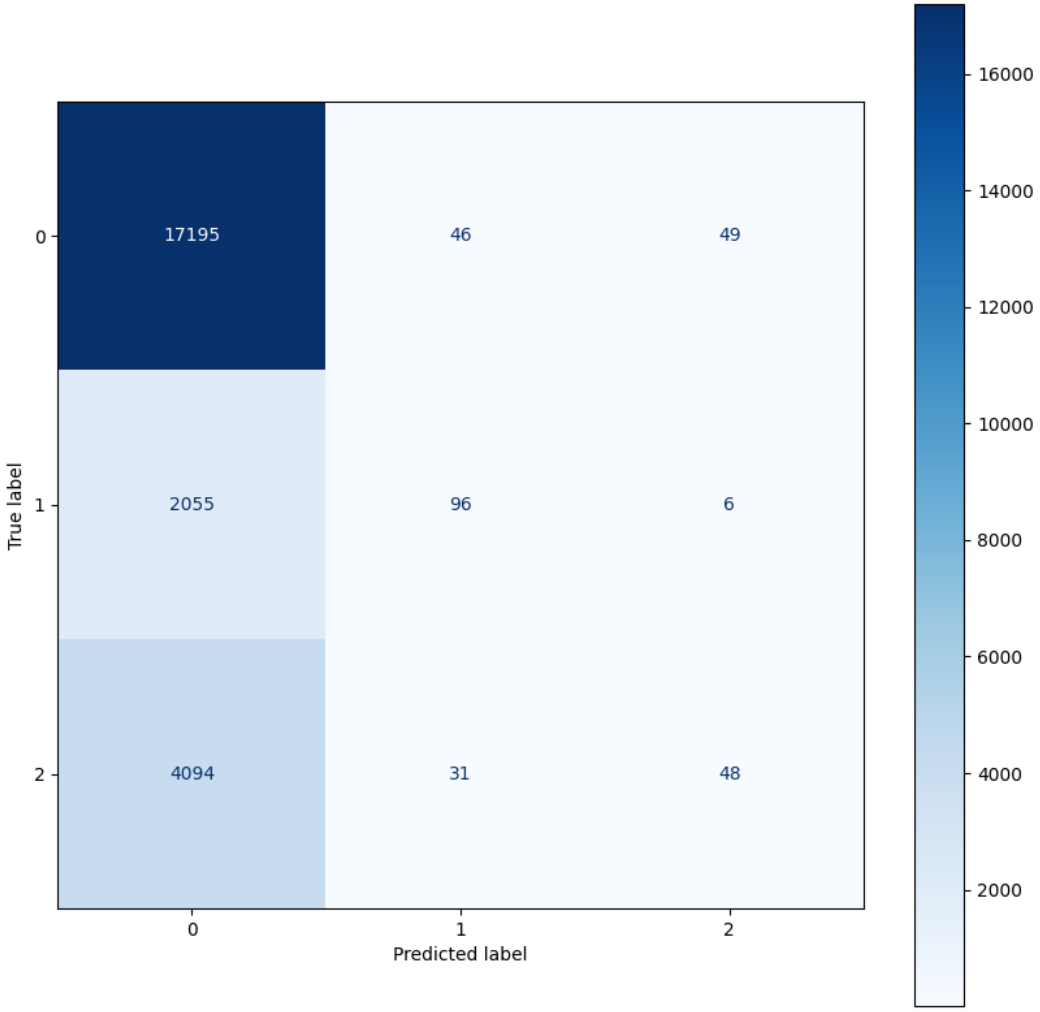

Figura 18 - Matrice di confusione Multilayer Perceptron con 2 strati intermedi

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.74	0.99	0.85
1	0.55	0.04	0.08
2	0.47	0.01	0.02

4.1.7.3 Hyperparameter Tuning con 3 layer intermedi

Sono stati stimati i migliori valori dei seguenti parametri tramite GridSearchCV:

- 'hidden_layer_sizes': [(10,10,10),(20,20,20),(20,50,20)]
- 'activation': ['tanh', 'relu']
- 'solver': ['sgd', 'adam','lbfgs']
- 'alpha': [0.0001, 0.05]

I parametri trovati sono i seguenti:

{'activation': 'relu', 'alpha': 0.05, 'hidden_layer_sizes': (20, 20, 20), 'learning_rate': 'adaptive', 'solver': 'lbfgs'}

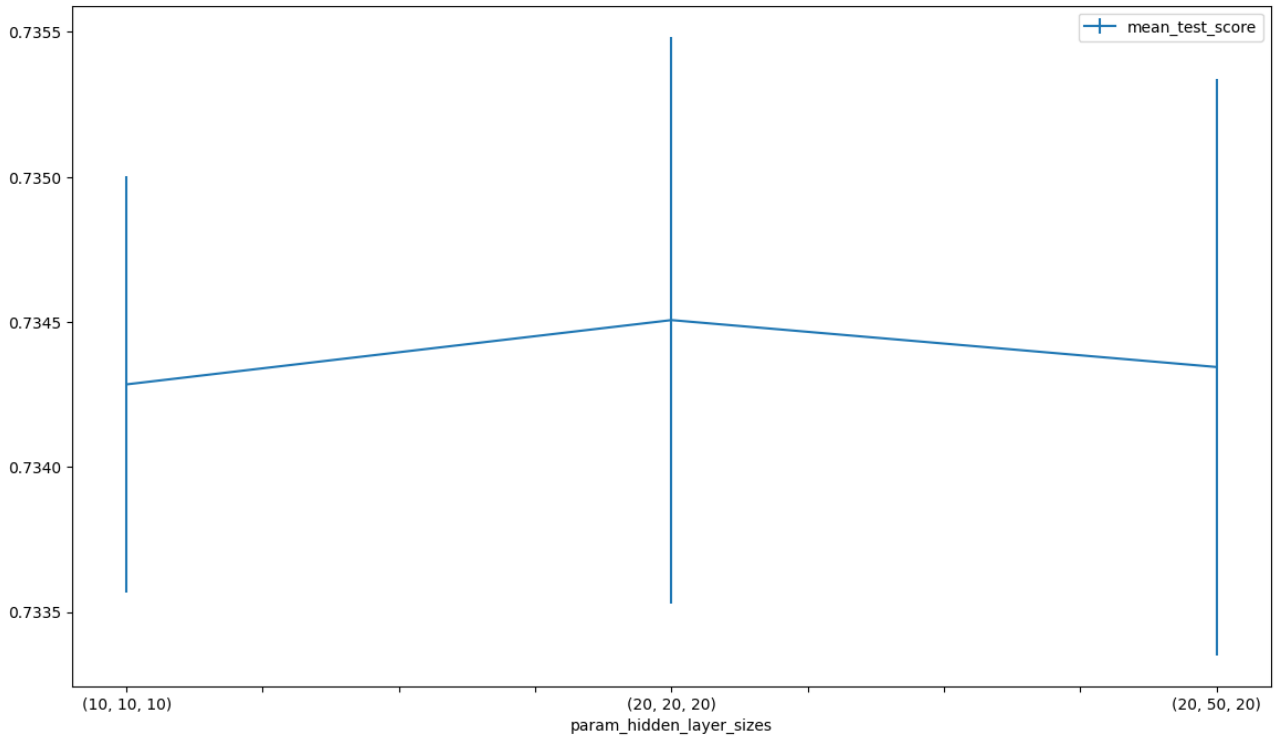

Figura 19 - Test accuracy al variare del numero di nodi dei 3 layer intermedi

Come possiamo notare dalla Figura 19, il valore più alto (seppur di poco) di 'Test accuracy' lo abbiamo con il paramentro *'hidden_layer_sizes': [(20,20,20)]*.

I risultati ottenuti, inserendo i migliori parametri sono stati:

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.7361	0.7341	0.5935	0.3502	0.3174

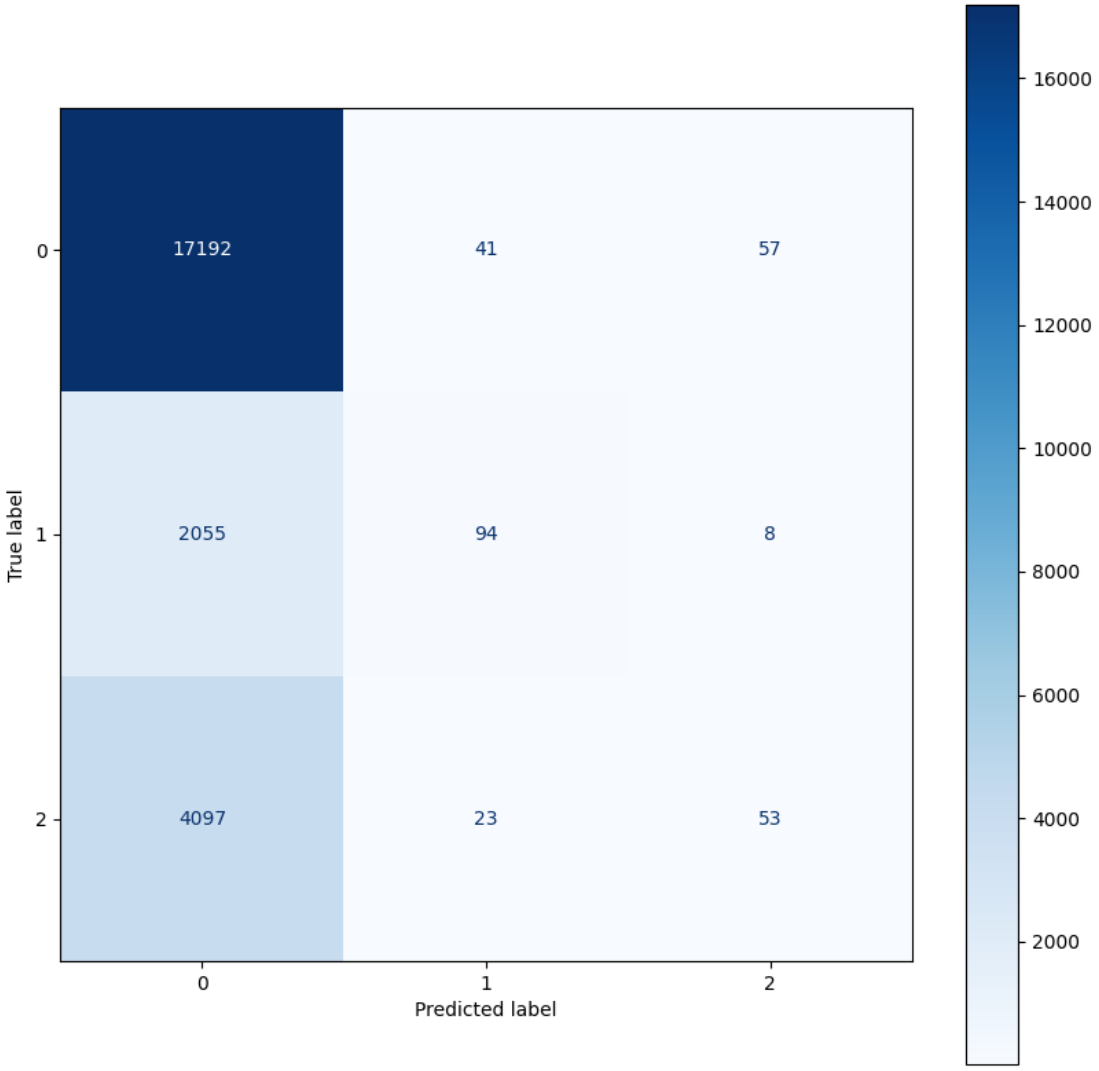

Figura 20 - Figura 17 - Matrice di confusione Multilayer Perceptron con 3 strati intermedi

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.74	0.99	0.85
1	0.59	0.04	0.08
2	0.45	0.01	0.02

Il modello con 3 strati intermedi risulta essere il migliore tra quelli analizzati. Se per l'Accuracy i valori sono pressoché identici, il valore della 'Precision' è leggermente migliore.

4.2 Dataset con tecnica di sovracampionamento

La tecnica utilizzata per il sovracampionamento è SMOTE (Synthetic Minority Oversampling Technique).

Questo approccio prevede che si vadano a prendere le osservazioni più vicine (distanza euclidea) tra quelle della classe minoritaria, si fa la differenza tra i due vettori di features e si moltiplica questo valore per un numero casuale tra 0 e 1. In altre parole, si va ad applicare una perturbazione alla

distanza tra due punti della classe minoritaria. Così facendo si creano osservazioni artificiali che accrescono il patrimonio di dati ma non ne modificano troppo il valore.

Dopo aver bilanciato il dataset, per poter valutare la performance dei modelli di apprendimento sul dataset di addestramento, ho utilizzato ancora la Cross-Validation (nello stesso modo del paragrafo 4.1.1).

I risultati ottenuti sono stati:

Modello	**Media dei risultati Cross-Validation**
DecisionTreeClassifier	0.53
GaussianNB	0.36
RandomForestClassifier	0.54
LogisticRegression	0.46
XGBClassifier	0.53

I risultati non sono dei migliori. L'algoritmo di classificazione GaussianNB è risultato di nuovo tra i peggiori, ottenendo il punteggio medio più basso mentre quelli con risultati leggermente migliori sono Random Forest, DecisionTree e XGBClassifier.

In base agli esiti elencati, la tecnica SMOTE non ha migliorato le prestazioni rispetto all'utilizzo del dataset senza bilanciamento.

Di seguito i risultati dei modelli che hanno avuto i risultati migliori.

4.2.1 Decision Tree

Utilizzando tutti i parametri di default il classificatore ha restituito i seguenti risultati:

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.5514	0.47	0.4073	0.4585	0.3853

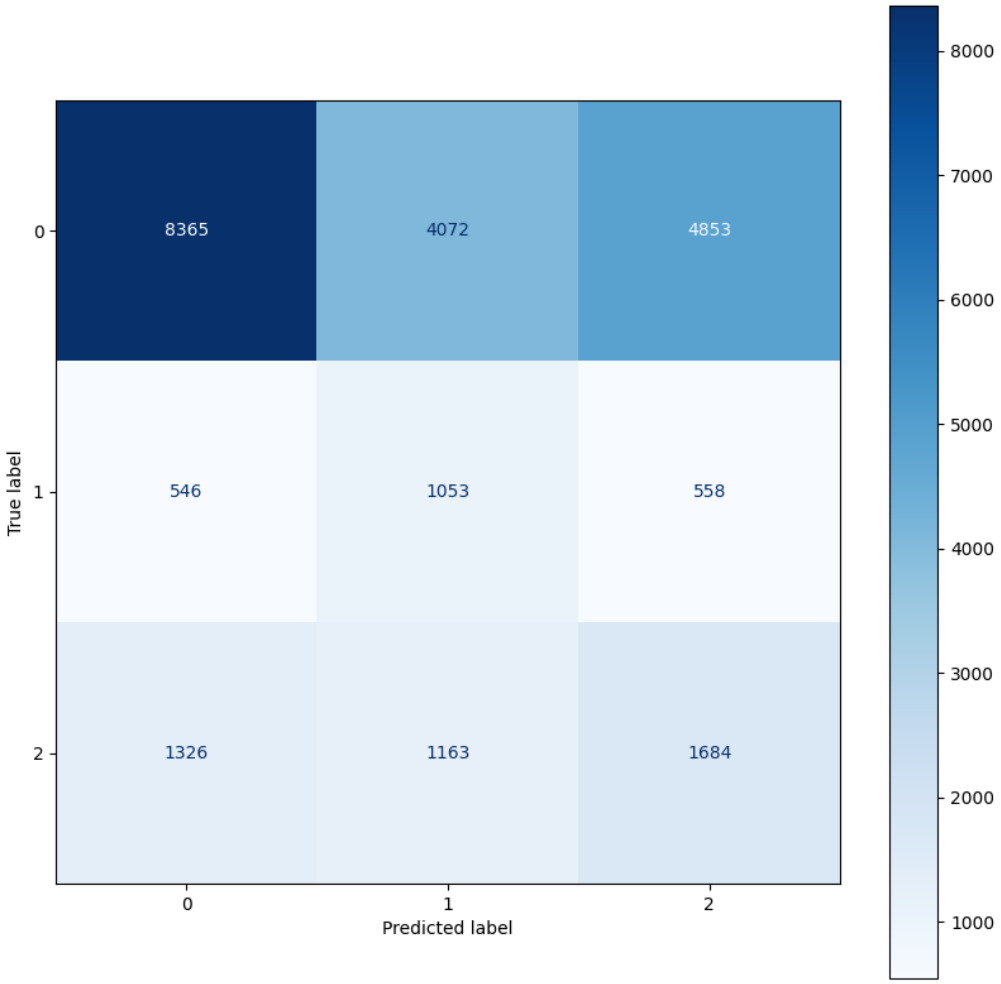

Figura 21- Decision Tree con tecnica SMOTE

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.82	0.48	0.61
1	0.17	0.49	0.25
2	0.24	0.40	0.30

4.2.2 Random Forest

Utilizzando tutti i parametri di default il classificatore ha restituito i seguenti risultati:

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.5514	0.47	0.408	0.4594	0.3858

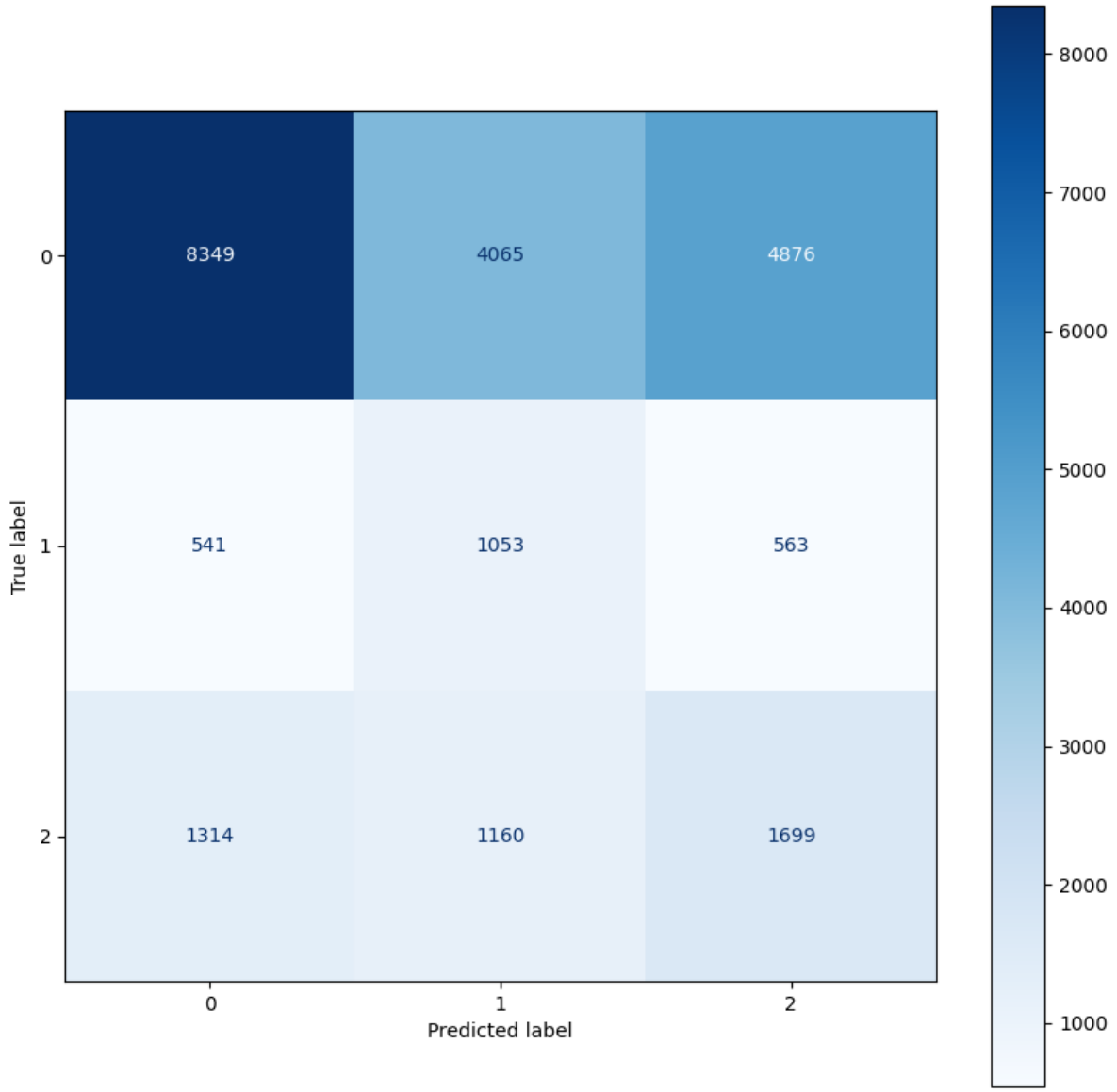

Figura 22 - Random Forest con tecnica SMOTE

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.82	0.48	0.61
1	0.17	0.49	0.25
2	0.24	0.41	0.30

I risultati ottenuti per questo modello sono molto simili a quelli riscontrati con Decision Tree, l'appartenenza alla stessa classe di modelli (ad albero) potrebbe aver giocato un ruolo decisivo.

4.2.3 XGBoost

Utilizzando tutti i parametri di default il classificatore ha restituito i seguenti risultati:

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.5514	0.47	0.408	0.4594	0.3858

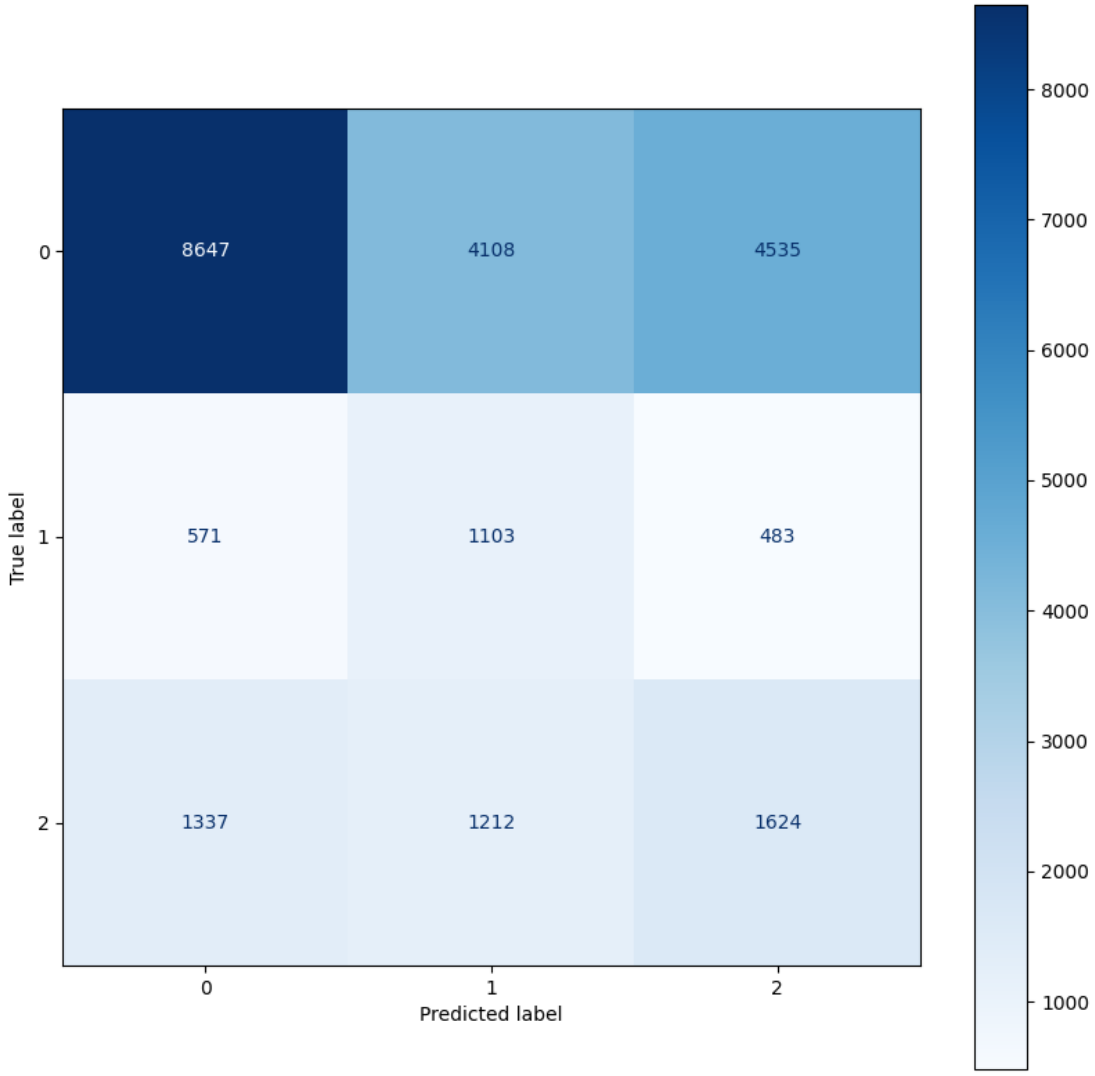

Figura 23 - XGBoost con tecnica SMOTE

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.82	0.50	0.62
1	0.17	0.51	0.26
2	0.24	0.39	0.30

Rispetto ai modelli precedenti si nota un aumento della Recall (46%), il valore più alto riscontrato. Ciò significa che il modello individua quasi la metà delle osservazioni positive.

4.3 Dataset con tecnica di sottocampionamento

La tecnica utilizzata per il sottocampionamento è il Near Miss-1. Essa è utilizzata per affrontare il problema della classe maggioritaria dominante nei dataset sbilanciati, dove la classe maggioritaria (cioè quella con il maggior numero di campioni) può avere una rappresentazione molto più grande rispetto alla classe minore.

La tecnica NearMiss, basata sul criterio della distanza tra le istanze, seleziona un sottoinsieme di campioni dalla classe maggioritaria in modo che i campioni rimossi siano "vicini" ai campioni della classe minore. Ciò significa che vengono rimossi solo i campioni della classe maggioritaria che sono più vicini dai campioni della classe minore, mentre i campioni della classe maggioritaria che sono più lontanii vengono mantenuti.

Near-Miss 1 mantiene le istanze della classe di maggioranza che hanno la minima distanza media dalle 3 istanze più vicine della classe di minoranza

Per poter valutare la performance dei modelli di apprendimento sul dataset di addestramento, ho utilizzato ancora la Cross-Validation (nello stesso modo del paragrafo 4.1.1).

I risultati ottenuti sono stati:

Modello	**Media dei risultati Cross-Validation**
DecisionTreeClassifier	0.48
GaussianNB	0.39
RandomForestClassifier	0.49
LogisticRegression	0.45
XGBClassifier	0.49

I risultati non sono dei migliori. L'algoritmo di classificazione peggiore è ancora GaussianNB che ha ottenuto il punteggio medio più basso mentre quelli con risultati leggermente migliori sono sempre Random Forest, DecisionTree e XGBClassifier.

In base agli esiti elencati nemmeno la tecnica Near-Miss 1 ha migliorato le prestazioni rispetto all'utilizzo del dataset senza bilanciamento.

Di seguito i risultati dei modelli che hanno avuto i risultati migliori.

4.3.1 Random Forest

Utilizzando tutti i parametri di default il classificatore ha restituito i seguenti risultati:

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.5279	0.3666	0.3815	0.3896	0.3135

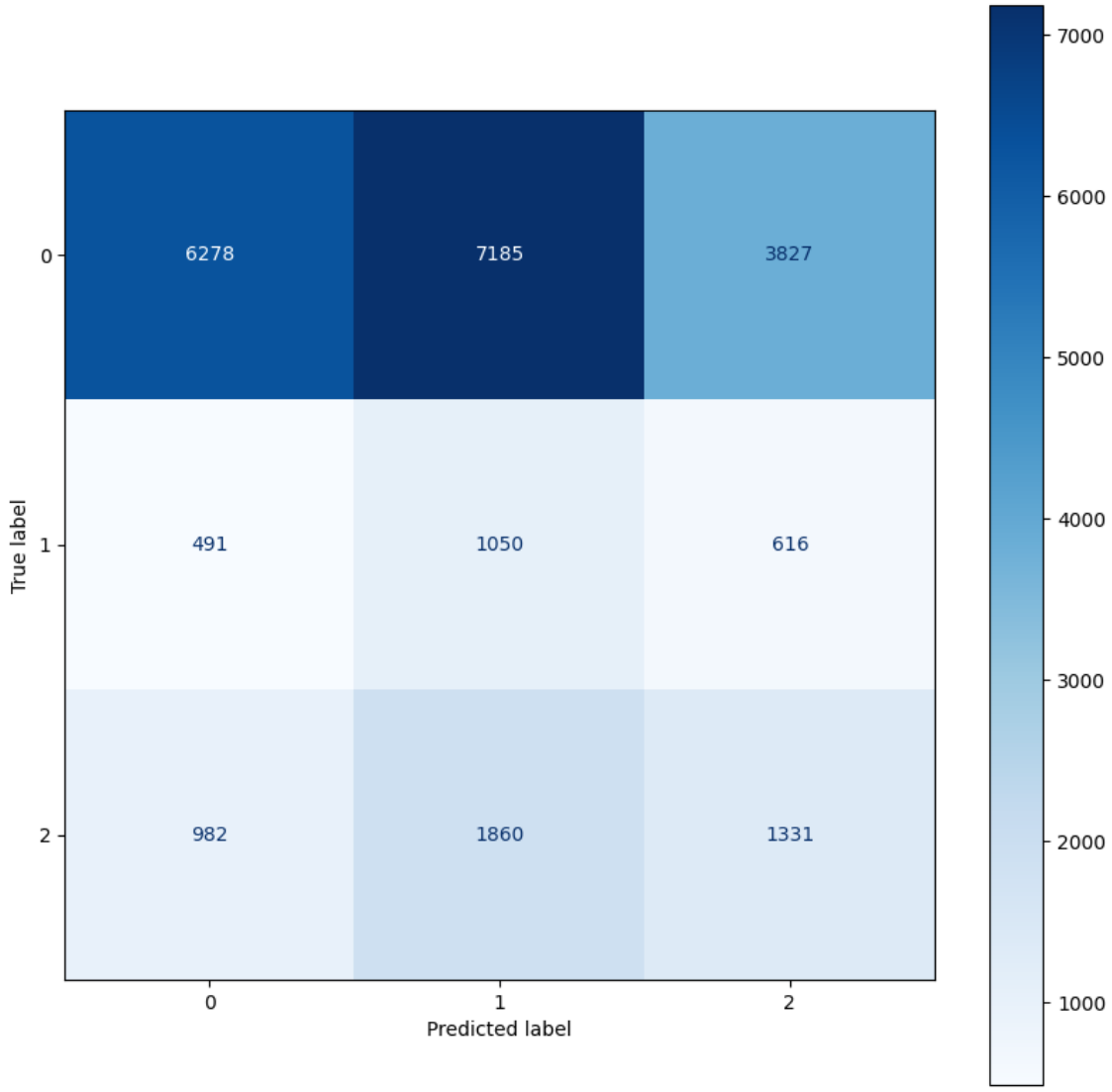

Figura 24- Random Forest con tecnica Near-Miss 1

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.81	0.36	0.50
1	0.10	0.49	0.17
2	0.23	0.32	0.27

4.3.2 DecisionTree

Utilizzando tutti i parametri di default il classificatore ha restituito i seguenti risultati:

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.5281	0.3771	0.3814	0.3925	0.3189

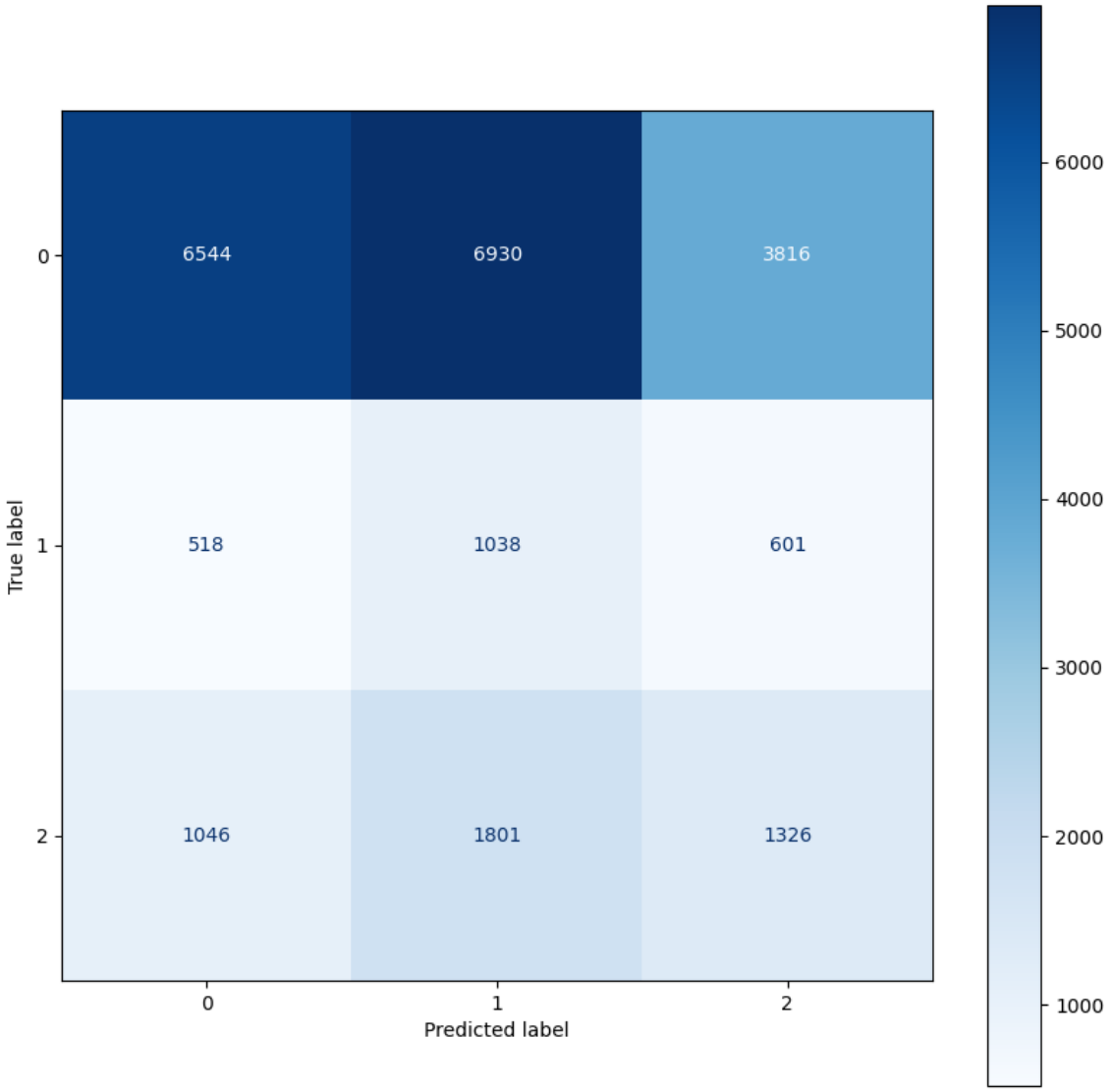

Figura 25 – Decision Tree con tecnica Near-Miss 1

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.81	0.38	0.52
1	0.11	0.48	0.17
2	0.23	0.32	0.27

4.3.3 XGBoost

Utilizzando tutti i parametri di default il classificatore ha restituito i seguenti risultati:

Training Accuracy	*Test Accuracy*	*Precision*	*Recall:*	*F1 Score*
0.5244	0.3725	0.3853	0.3985	0.32

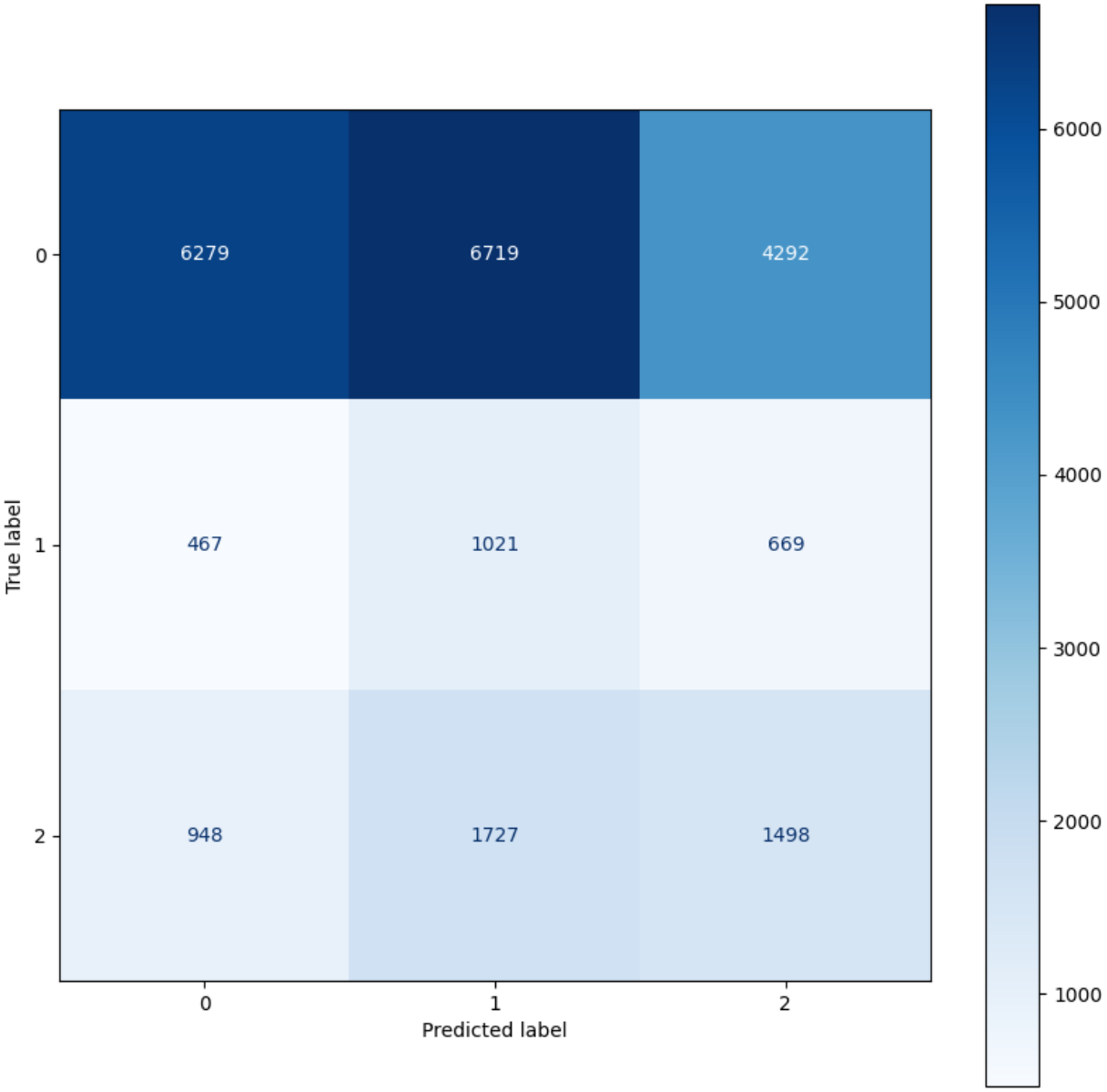

Figura 26 – XGBoost con tecnica Near Miss-1

Di seguito le metriche per le singole classi:

Classi	Precision	Recall	F1 Score
0	0.81	0.36	0.50
1	0.11	0.47	0.18
2	0.23	0.36	0.28

5. Conclusioni

Il problema iniziale di classificare 39 diverse categorie di crimini è un problema multi-classe che non consente di ottenere risultati con una precisione elevata. Ho quindi suddiviso le categorie di crimine in tre gruppi (relativi alla proprietà, violenti o relativi allo stile di vita) e ho eliminato quelle meno significative per l'obiettivo che volevo raggiungere. Analogamente ho ridotto le 12 features della variabile indipendente a sole quattro, di cui due già presenti (PdDistrict, DayOfWeek) e le altre due ricavate dal dataset iniziale (StreetType, DayPeriod).

Ho quindi utilizzato sei modelli di machine learning sul dataset ottenuto **senza applicare alcuna tecnica di bilanciamento**:

- Decision Tree
- Random Forest
- Logistic Regression
- Gaussian Naive Bayes
- XGboost
- Multi-Layer Perceptron

In termini di 'Accuracy' i vari modelli si sono comportati più o meno nello stesso modo (73-74%) ad eccezione di 'Gaussian Naive Bayes' (15%). In particolare, i migliori sono stati i modelli ad albero 'Decision Tree e 'Random Forest'. La 'Precision' migliore invece è stata ottenuta con il modello 'Logistic Regression' (61%).

Successivamente ho applicato le tecniche di hyper-parameter tuning GridSearch e BayesSearch ai modelli con 'Accuracy' migliore (Decision Tree e Random Forest). In entrambi i casi ci sono stati piccoli miglioramenti nella 'Test Accuracy' dimostrando così una riduzione dell'overfitting.

Per il modello di Neural Network, Multi-layer Perceptron, ho effettuato l'hyper-parameter tuning per 3 casi, con 1 livello intermedio, con 2 livelli intermedi e con 3 livelli intermedi. Come previsto, il risultato migliore è stato ottenuto con 3 livelli intermedi, anche se di poco.

In seguito, ho **bilanciato il dataset** con due tecniche:

- **SMOTE**, per il sovracampionamento
- **Near Miss-1**, per il sottocampionamento

In entrambi i casi, i risultati ottenuti sono risultati peggiori rispetto al dataset non bilanciato, indicando un problema di overfitting.

In conclusione, per il dataset analizzato le tecniche di hyper-parameter tuning, di sovracampionamento e di sottocampionamento non hanno fornito grandi miglioramenti. I modelli che hanno funzionato meglio sono stati:

- Decision Tree
- Random Forest
- Multi-Layer Perceptron, con 3 layer intermedi

6. Sitografia

1. https://www.thesmartcityjournal.com/en/cities/smart-city-san-francisco
2. https://datasf.org/opendata/
3. https://en.wikipedia.org/wiki/
4. https://www.ibm.com/it-it/topics/random-forest
5. https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/
6. https://medium.com/analytics-vidhya/comparison-of-hyperparameter-tuning-algorithms-grid-search-random-search-bayesian-optimization-5326aaef1bd1
7. https://scikit-optimize.github.io/stable/modules/generated/skopt.BayesSearchCV.html
8. https://scikit-learn.org/stable/modules/neural_networks_supervised.html
9. https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/