Progetto di Analisi: Employee Retention

Basato sul dataset https://www.kaggle.com/datasets/tawfikelmetwally/employee-dataset

Raffaele Manzo

Matricola: 05121 16484

Contents

1 Introduzione

Il progetto mira a sviluppare un modello di machine learning per risolvere un problema di classificazione binaria nel contesto della gestione delle risorse umane. In particolare, l'obiettivo è determinare se un dipendente dovrebbe rimanere nell'organizzazione o lasciarla. Questa analisi fornirà supporto strategico nella gestione dei dipendenti, aiutando a identificare i fattori chiave che influenzano il turnover e a ottimizzare la produttività aziendale.

1.1 Obiettivo e Problema

L'obiettivo principale del progetto è costruire un modello predittivo in grado di classificare i dipendenti in due categorie:

- **0**: Il dipendente rimane nell'organizzazione.
- **1**: Il dipendente lascia l'organizzazione.

Il problema può essere formalizzato come una classificazione supervisionata binaria, dove la variabile target *LeaveOrNot* indica la classe di appartenenza. Le caratteristiche dei dipendenti, come età, livello di istruzione, anni di esperienza, genere, e altri fattori, verranno utilizzate per addestrare il modello e fare previsioni.

1.2 Obiettivi del Progetto

1. Analizzare e preprocessare il dataset per garantire la qualità dei dati, inclusa la gestione dei valori mancanti.
2. Sviluppare un modello predittivo utilizzando tecniche di machine learning supervisionato.
3. Valutare le performance del modello attraverso metriche come accuracy, precision, recall e F1-score.

2 Analisi dei Dati

2.1 Descrizione del Dataset

Il dataset contiene informazioni sui dipendenti di un'azienda, inclusi i loro background educativi, la storia lavorativa, i dati demografici e i fattori legati all'occupazione. È stato anonimizzato per proteggere la privacy dei dipendenti, pur mantenendo informazioni utili per l'analisi della forza lavoro.

Colonna	Descrizione
Education	Qualifiche educative, inclusi titolo di studio, istituzione e campo di studio
JoiningYear	Anno di assunzione, indica la durata del servizio
City	Città di base o lavoro del dipendente
PaymentTier	Classificazione in diverse fasce salariali
Age	Età del dipendente, fornisce informazioni demografiche
Gender	Identità di genere, utile per analisi di diversità
EverBenched	Indica se il dipendente è stato temporaneamente senza lavoro assegnato
ExperienceInCurrentDomain	Anni di esperienza nel dominio attuale
LeaveOrNot	Variabile target: 0 (rimane), 1 (lascia)

Table 1: Descrizione delle variabili del dataset

2.2 Formato e Caricamento del Dataset

Il dataset è fornito in formato `CSV` (Comma-Separated Values), un formato di testo leggibile e ampiamente supportato. Questo file può essere facilmente importato in Python utilizzando librerie come `pandas`. Per questo progetto, il dataset è stato caricato e analizzato con il seguente codice:

```
import pandas as pd

# Caricamento del dataset
file_path = "Employee_with_missing.csv"
data = pd.read_csv(file_path)

# Visualizzare le prime righe del dataset
data.head()

data.info()
```

2.3 Tabella dei Tipi delle Colonne del Dataset

Colonna	Tipo
Education	Categoriale
JoiningYear	Numerica
City	Categoriale
PaymentTier	Numerica
Age	Numerica
Gender	Categoriale
EverBenched	Categoriale
ExperienceInCurrentDomain	Numerica
LeaveOrNot	Numerica

Table 2: Tipi delle Colonne del Dataset

3 Analisi ed Elaborazione dei Dati

3.1 Variabile Dipendente

La variabile dipendente *LeaveOrNot* rappresenta il target del modello ed è una variabile binaria:

- **0**: Il dipendente rimane nell'organizzazione.
- **1**: Il dipendente lascia l'organizzazione.

Questa variabile sarà analizzata per comprendere la distribuzione delle classi e identificare eventuali sbilanciamenti che potrebbero influire sulle performance del modello. Grafici a torta e istogrammi saranno utilizzati per visualizzare la distribuzione delle classi.

3.2 Valori Mancanti

Il dataset aggiornato contiene 5653 righe e 9 colonne. La presenza di valori mancanti è stata simulata per riprodurre scenari realistici e rappresenta un aspetto fondamentale della fase di preprocessing. I valori mancanti sono distribuiti casualmente in tutte le colonne del dataset:

- **Education**: 108 valori mancanti.
- **JoiningYear**: 105 valori mancanti.

- **City**: 115 valori mancanti.
- **PaymentTier**: 104 valori mancanti.
- **Age**: 95 valori mancanti.
- **Gender**: 114 valori mancanti.
- **EverBenched**: 99 valori mancanti.
- **ExperienceInCurrentDomain**: 102 valori mancanti.
- **LeaveOrNot**: 118 valori mancanti.

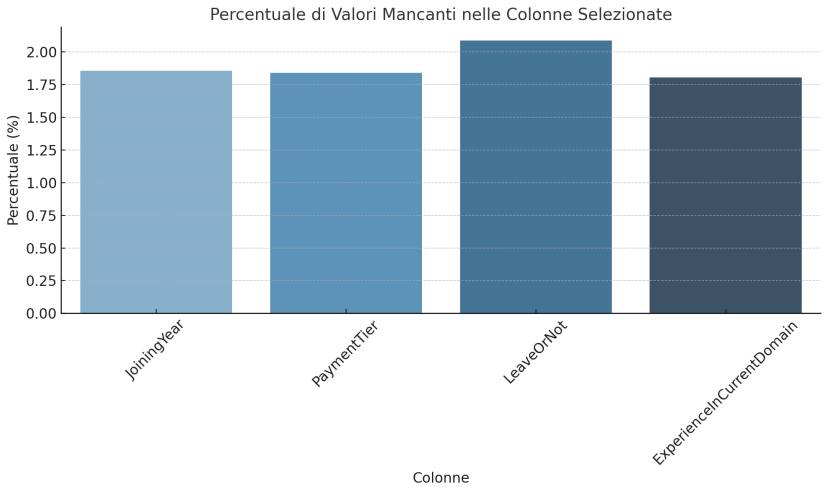

Figure 1: Percentuale di Valori Mancanti nelle Colonne Selezionate

3.3 Gestione dei Valori Mancanti

La gestione dei valori mancanti rappresenta una fase critica del preprocessing. Dopo un'analisi delle colonne, è stato deciso di eliminare la colonna *City* in quanto ritenuta poco rilevante per il contesto del problema. La decisione si basa sul fatto che la città di lavoro non incide significativamente sui fattori chiave che determinano la permanenza o l'abbandono del dipendente. Per le altre colonne, sono state adottate le seguenti tecniche:

- **Imputazione per la Media**: Per la colonna numerica *Age*, i valori mancanti sono stati riempiti con la media, in quanto l'età rappresenta un attributo continuo.

- **Imputazione per la Moda**: Per colonne categoriali come *Gender*, i valori mancanti sono stati riempiti con la moda, ossia il valore più frequente.
- **Creazione di una Categoria per i Valori Mancanti**: Per colonne come *EverBenched*, una nuova categoria *Non Definito* è stata aggiunta per rappresentare i valori mancanti.
- **Rimozione di Righe per le Colonne Critiche**: Le righe con valori mancanti nelle colonne *JoiningYear*, *PaymentTier*, *LeaveOrNot* ed *ExperienceInCurrentDomain* sono state eliminate per preservare l'integrità e la qualità dei dati critici.

3.3.1 Gestione dei Valori Mancanti nelle Colonne Critiche

Per garantire la qualità e l'affidabilità del modello di machine learning, è stata adottata la strategia di rimuovere le righe con valori mancanti nelle seguenti colonne: *JoiningYear*, *PaymentTier*, *LeaveOrNot* ed *ExperienceInCurrentDomain*. La scelta è motivata dai seguenti aspetti:

- **Importanza delle Colonne per il Modello**:
  - *JoiningYear*: Indica la durata del servizio del dipendente, un fattore chiave per l'analisi della retention.
  - *PaymentTier*: Classifica i dipendenti in fasce salariali, una variabile fortemente correlata alla motivazione e al turnover.
  - *LeaveOrNot*: È la variabile target del modello; i valori mancanti in questa colonna renderebbero inutilizzabili quelle righe per l'addestramento.
  - *ExperienceInCurrentDomain*: Fornisce informazioni sull'esperienza nel dominio attuale, un predittore significativo delle performance e della permanenza del dipendente.
- **Preservazione della Qualità del Dataset**: L'eliminazione delle righe con valori nulli in queste colonne garantisce che il dataset finale sia composto da dati completi e affidabili. Questo minimizza il rischio di introdurre bias nel modello.
- **Proporzione Ridotta di Valori Mancanti**: La percentuale di righe con valori mancanti in queste colonne è relativamente bassa rispetto al volume totale del dataset. Pertanto, l'impatto dell'eliminazione sarà limitato, preservando comunque un dataset sufficientemente rappresentativo.

- **Compatibilità con il Modello**: La presenza di valori nulli in queste colonne critiche potrebbe causare problemi durante il processo di addestramento del modello o richiedere strategie di gestione complesse. La rimozione rappresenta una soluzione semplice ed efficace per evitare questi ostacoli.

3.4 Grafici Associati alla Variabile Dipendente

Per comprendere meglio le relazioni tra la variabile dipendente e le altre variabili, saranno generati grafici esplorativi:

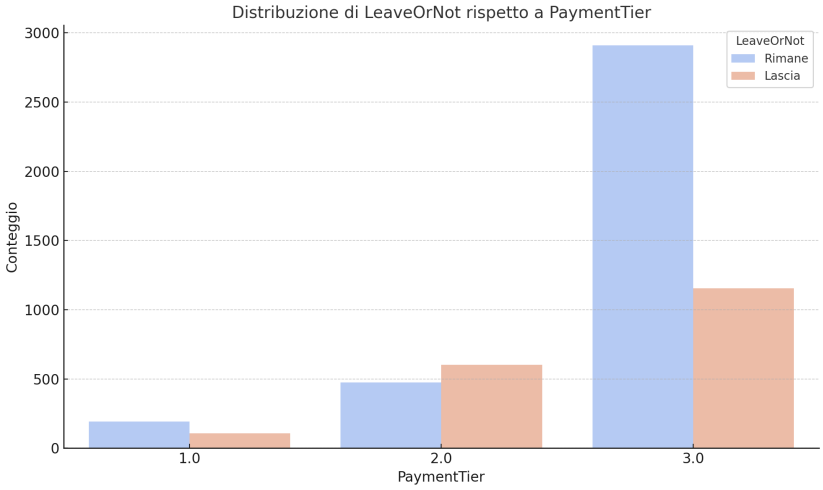

Figure 2: Distribuzione di LeaveOrNot rispetto a PaymentTier

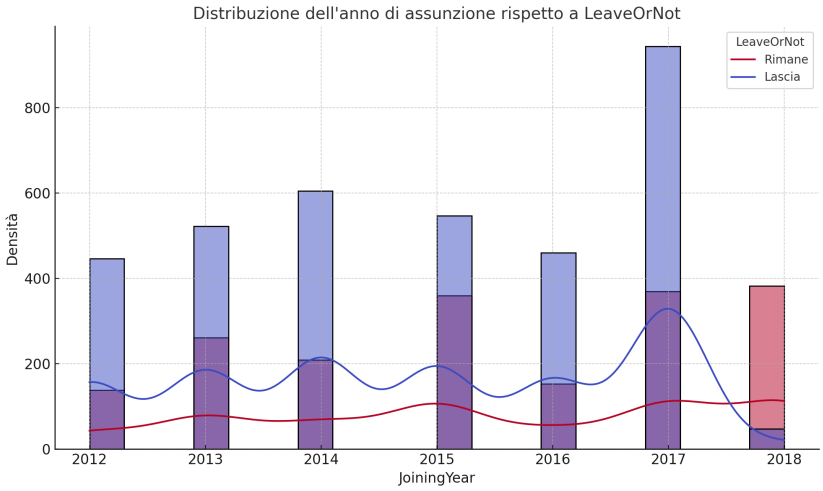

Figure 3: Distribuzione dellánno di assunzione rispetto a LeaveOrNot

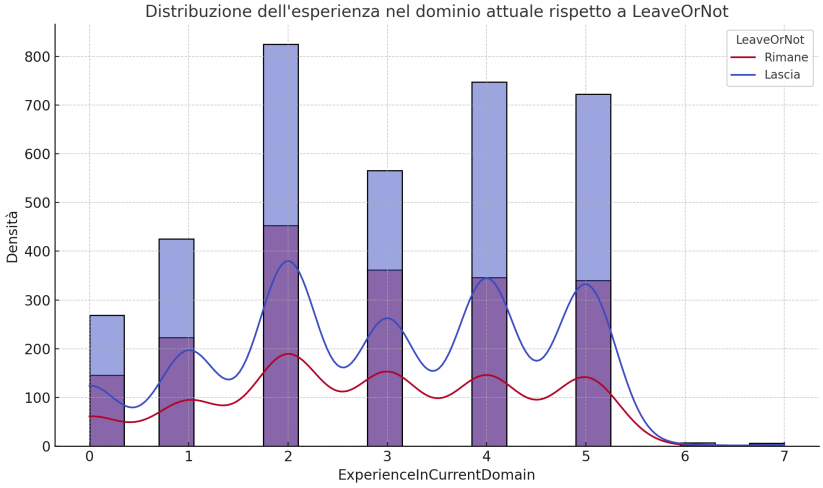

Figure 4: Distribuzione dellésperienza nel dominio attuale rispetto a LeaveOrNot

3.5 Encoding delle Variabili Categoriali

Le variabili categoriali saranno convertite in formato numerico per essere utilizzate nei modelli di machine learning. Verranno utilizzati i seguenti metodi:

- **Label Encoding**: Per colonne binarie come *EverBenched* e *Gender*.

- **One-Hot Encoding**: Per colonne con più categorie come *City* ed *Education*.

Queste tecniche garantiranno che tutte le variabili siano rappresentate in un formato compatibile con gli algoritmi di machine learning.

Feature	**Type**
JoiningYear	Numerical
PaymentTier	Numerical
Age	Numerical
ExperienceInCurrentDomain	Numerical
LeaveOrNot	Numerical
Education_Bachelors	Encoded Categorical
Education_Masters	Encoded Categorical
Education_PhD	Encoded Categorical
Gender_Female	Encoded Categorical
Gender_Male	Encoded Categorical
EverBenched_No	Encoded Categorical
EverBenched_Yes	Encoded Categorical

Table 3: Feature del Dataset Dopo Encoding Completo e Rimozione della Colonna City

3.5.1 Esempio di Codice

Un esempio di implementazione in Python:

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

# Label Encoding
label_encoder = LabelEncoder()
data['Gender'] = label_encoder.fit_transform(data['Gender'])
data['EverBenched'] = label_encoder.fit_transform(data['EverBenched'])

# One-Hot Encoding
data = pd.get_dummies(data, columns=['Education'], drop_first=False)
```

3.5.2 Bilanciamento della Variabile Target e Correlazione delle Feature Codificate

Dopo aver eseguito l'encoding delle variabili categoriali, è stata effettuata l'analisi della correlazione tra tutte le variabili del dataset. La heatmap

risultante è mostrata di seguito:

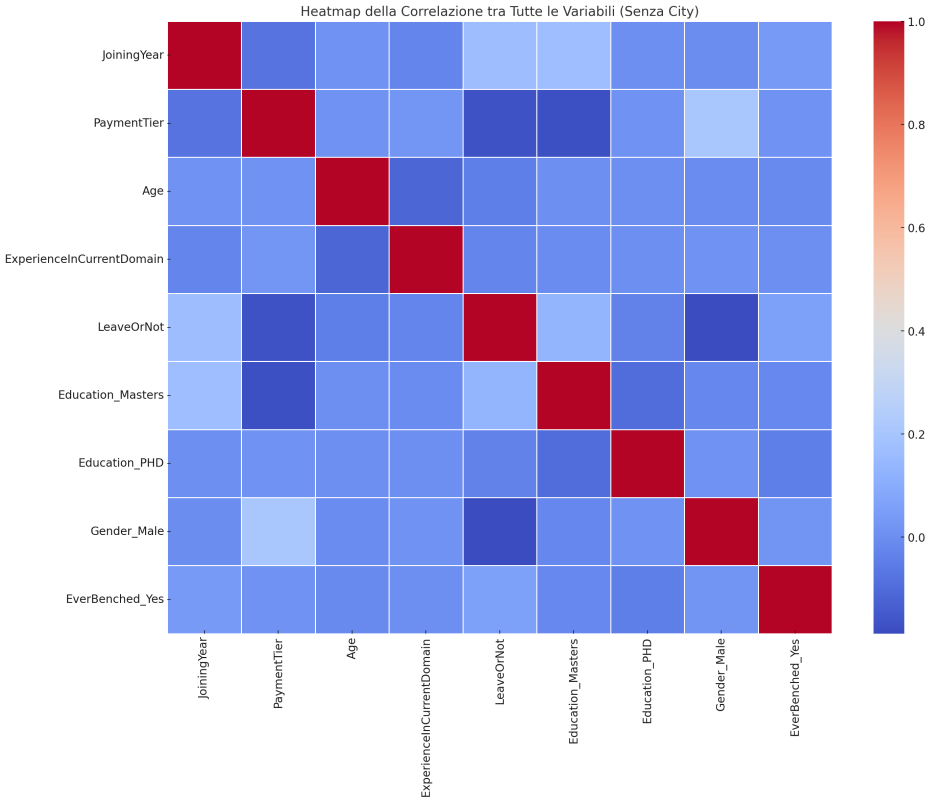

Figure 5: Heatmap della Correlazione tra Tutte le Variabili (Numeriche e Codificate)

La distribuzione della variabile target *LeaveOrNot* dopo l'encoding rimane invariata rispetto all'originale, come mostrato nel seguente grafico:

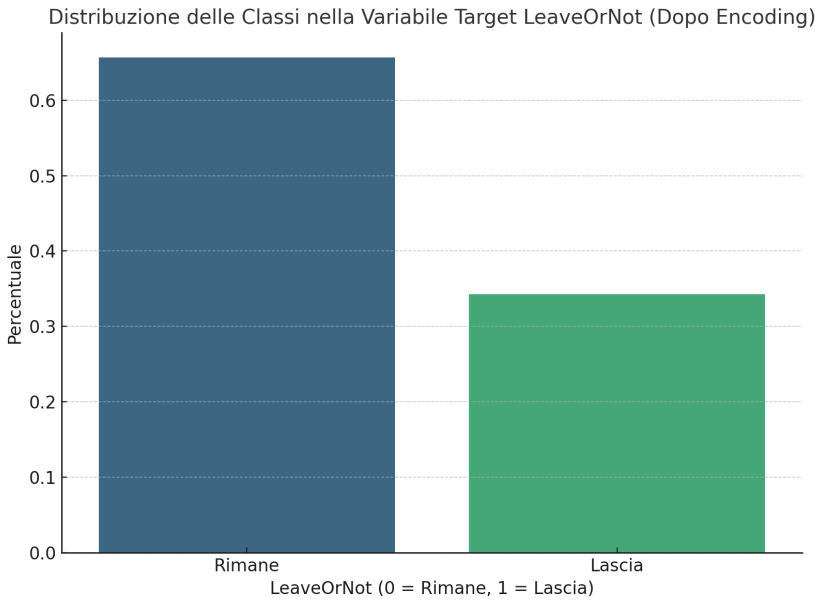

Figure 6: Distribuzione delle Classi nella Variabile Target LeaveOrNot (Dopo Encoding)

Per bilanciare le classi durante il training, utilizzeremo la tecnica di oversampling con SMOTE, che genera nuovi dati sintetici per la classe minoritaria, mantenendo la distribuzione delle feature basata sui vicini più prossimi.

4 Addestramento e Valutazione del Modello

4.1 Scelta dei Modelli

Per affrontare il problema di classificazione binaria, sono stati selezionati due modelli da testare: la regressione logistica e Random Forest. La scelta è stata guidata da considerazioni pratiche e basate sull'obiettivo di trovare un equilibrio tra semplicità e capacità predittiva.

4.1.1 Perché Logistic Regression?

La regressione logistica è un modello lineare che offre una base solida per comprendere come le variabili del dataset influenzano la decisione finale. È stata scelta perché:

- È intuitiva: il modello permette di interpretare facilmente il contributo di ogni variabile al risultato finale.

- È rapida: con dataset relativamente piccoli come il nostro, il tempo di addestramento è minimo.
- È affidabile: fornisce risultati chiari e prevedibili, particolarmente utili come punto di partenza per valutazioni più complesse.

4.1.2 Perché Random Forest?

Il modello Random Forest è stato selezionato per la sua capacità di adattarsi a problemi più complessi e gestire variabili con relazioni non lineari. Ecco perché lo consideriamo:

- È versatile: riesce a catturare interazioni tra le variabili che potrebbero sfuggire a modelli più semplici.
- È robusto: riduce il rischio di errori gravi grazie alla combinazione di più alberi decisionali.
- È esplicativo: oltre a fare previsioni, fornisce una classifica dell'importanza delle variabili, utile per analisi approfondite.

Questo modello è particolarmente adatto per situazioni in cui vogliamo massimizzare l'accuratezza predittiva e ottenere una visione più dettagliata delle dinamiche del dataset.

4.2 Valutazione dei Modelli

Per confrontare le performance dei due modelli, utilizzeremo metriche che ci permettono di misurare sia la qualità complessiva delle previsioni sia la capacità di distinguere tra classi diverse. Le metriche selezionate sono:

- **Accuracy**: Ci dice quanto spesso il modello classifica correttamente un esempio.
- **Precision**: Indica quanto il modello sia affidabile nel predire i casi positivi (es. predizioni di abbandono che si rivelano corrette).
- **Recall**: Misura la capacità del modello di individuare tutti i casi positivi, anche quelli più difficili.
- **F1-Score**: Una combinazione bilanciata di precision e recall, utile quando vogliamo dare uguale peso a entrambi gli aspetti.

- **ROC-AUC**: Fornisce una visione d'insieme sulla capacità del modello di separare correttamente le classi, indipendentemente dalla soglia utilizzata.

Queste metriche ci aiuteranno a identificare quale dei due modelli, Logistic Regression o Random Forest, sia più adatto per il nostro problema di classificazione e a prendere decisioni basate non solo sulla precisione, ma anche sull'affidabilità e sull'adattabilità del modello.

4.3 Divisione del Dataset

La divisione del dataset in training e test rappresenta una fase cruciale per garantire che il modello sia robusto e generalizzabile. In questa sezione, viene descritto l'approccio adottato per una divisione ottimale del dataset.

4.3.1 Dimensione del Dataset

Il dataset, contenente circa **5653 campioni** dopo il preprocessing, è considerato di piccole dimensioni. Pertanto, è stata adottata una divisione **80%-20%**, riservando l'80% dei dati all'addestramento e il 20% alla valutazione.

4.3.2 Codice per la Divisione del Dataset

Il seguente codice Python è stato utilizzato per implementare la divisione del dataset:

```
from sklearn.model_selection import train_test_split

# Divisione in training e test set (80%-20%)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Verifica della distribuzione della variabile target
print("Distribuzione nel Training Set:")
print(y_train.value_counts(normalize=True))

print("\nDistribuzione nel Test Set:")
print(y_test.value_counts(normalize=True))
```

5 Valutazione dei Risultati

5.1 Logistic Regression

- **Senza bilanciamento**: La regressione logistica mostra una performance moderata con un'accuratezza del **65%** e un ROC-AUC di **0.68**. Tuttavia, la precision per la classe minoritaria (*1.0*) è bassa, evidenziando difficoltà nel bilanciare le predizioni.
- **Con bilanciamento**: Dopo l'applicazione di SMOTE, l'accuratezza rimane simile, ma la recall della classe minoritaria migliora leggermente, indicando un incremento nella capacità di identificare dipendenti che lasciano l'azienda.

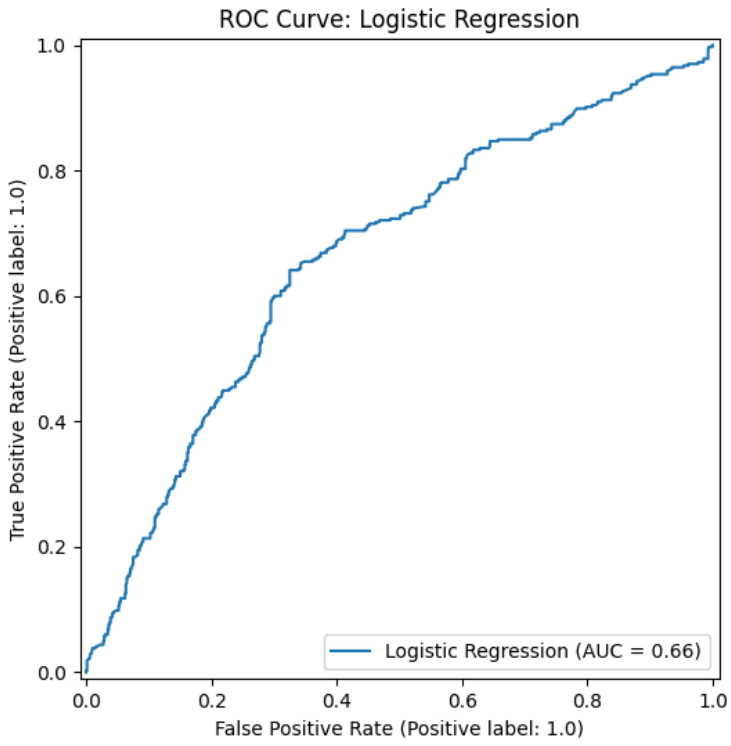

Figure 7: Roc Curve per Logistic Regrssion

5.2 Random Forest

- **Senza bilanciamento**: Random Forest supera Logistic Regression in termini di accuratezza (**75%**) e ROC-AUC (**0.776**). Questo è probabilmente dovuto alla capacità di Random Forest di modellare relazioni non lineari e gestire feature interattive.
- **Con bilanciamento**: Con SMOTE, la performance rimane stabile, ma si osserva un piccolo miglioramento nella recall della classe minoritaria. La precision della classe *1.0* migliora rispetto alla Logistic Regression.

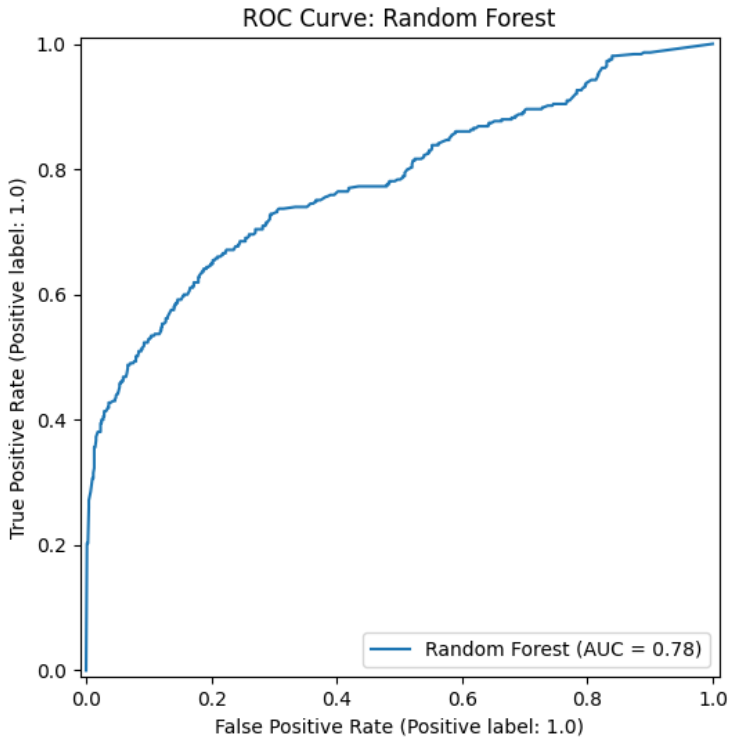

Figure 8: Roc Curve per Random Forest

5.3 Confusion Matrices

Le matrici di confusione forniscono una visione dettagliata dei falsi positivi e dei falsi negativi:

- **Logistic Regression**: Mostra più falsi negativi, indicando difficoltà nel classificare correttamente i dipendenti che lasciano l'azienda.
- **Random Forest**: Offre una migliore distribuzione tra veri positivi e falsi negativi rispetto alla Logistic Regression.

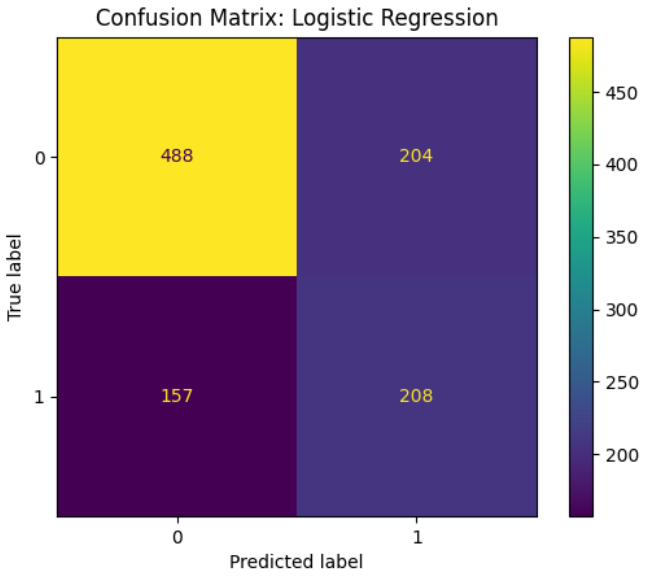

Figure 9: Matrice di confusione per Logistic Regression

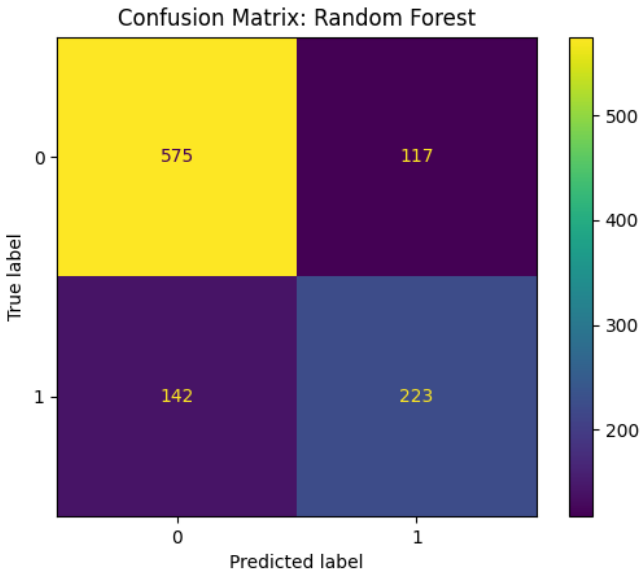

Figure 10: Matrice di confusione per Random Forest

5.4 Considerazioni Finali

- **Random Forest** sembra essere la scelta migliore per questo dataset, grazie alla sua robustezza e capacità di modellare relazioni complesse.
- La bilanciatura con SMOTE migliora leggermente le metriche della classe minoritaria, ma il guadagno complessivo è limitato.