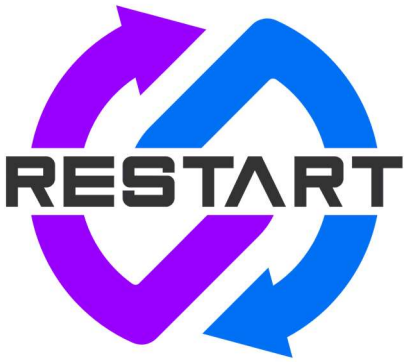


Documentazione modello IA ReStart

Versione	1.0
Data	12/02/2024
Destinatario	Prof.re Fabio Palomba
Presentato da	Gianfranco Barba, Francesco Corcione, Luigi Guida, Tullio Mansi, Matteo Panza
Approvato da	/

Sommario

Team Members

Ruolo	Nome e Cognome	Acronimo	Email
TM	Gianfranco Barba	GB	g.barba14@studenti.unisa.it
TM	Francesco Corcione	FC	f.corcione5@studenti.unisa.it
TM	Luigi Guida	LG	l.guida15@studenti.unisa.it
TM	Tullio Mansi	TM	t.mansi@studenti.unisa.it
TM	Matteo Panza	MP	m.panza13@studenti.unisa.it

1. Introduzione

1.1. Descrizione del problema

"Restart" è un progetto volto a facilitare la piena reintegrazione degli ex-detenuti nella società. Il problema affrontato dal nostro classificatore è il difficile reinserimento lavorativo degli ex-detenuti nella società. Durante il periodo di reclusione, molti di loro perdono contatto con il mondo del lavoro e con le competenze necessarie per interfacciarsi efficacemente nel mercato attuale. Di conseguenza, si trovano ad affrontare notevoli ostacoli nel cercare un nuovo impiego una volta rilasciati. Il nostro obiettivo è sviluppare un modello in grado di individuare professioni adatte alle competenze degli ex-detenuti, facilitando così il loro reinserimento professionale.

1.2. Motivazione e rilevanza

La reintegrazione degli ex-detenuti è un tema di grande importanza sociale. Un adeguato reinserimento lavorativo non solo riduce il rischio di recidiva, ma contribuisce anche alla costruzione di una società più inclusiva e solidale. Tuttavia, la mancanza di supporto nel trovare un impiego adatto rappresenta uno dei principali fattori che ostacolano questo processo. Il nostro modello si propone di colmare questa lacuna fornendo un'importante risorsa per facilitare il reinserimento lavorativo degli ex-detenuti, contribuendo così al benessere individuale e alla stabilità sociale.

1.3. Obiettivi

- Sviluppare un classificatore in grado di analizzare le competenze e le esperienze lavorative degli ex-detenuti.

- Identificare professioni che corrispondano alle competenze e alle preferenze dei singoli individui.

- Valutare l'efficacia del modello nel facilitare il reinserimento lavorativo degli ex-detenuti attraverso monitoraggi e valutazioni periodiche.

1.4. Specifica PEAS

PEAS per il Modello di Classificazione dei Titoli di Lavoro

Performance (Prestazione)

- Misura di Performance: la prestazione del nostro agente (il modello di classificazione) è valutata sulla base della sua capacità di classificare correttamente i titoli di lavoro. L'obiettivo è massimizzare l'accuratezza, la precisione, la recall e l'F1 score, assicurando che il modello possa identificare con precisione il titolo di lavoro appropriato in base alle caratteristiche fornite.

Environment (Ambiente)

- Ambiente Operativo: il nostro agente opera nell'ambiente professionale e aziendale, dove le dinamiche sono influenzate dalle tendenze del mercato del lavoro, dalle competenze richieste e dai cambiamenti nelle strutture organizzative.
- Dinamico: l'ambiente è dinamico poiché i requisiti di lavoro e le competenze cambiano nel tempo.
- Episodico: ogni classificazione è indipendente dalle altre; la classificazione di un candidato non influisce su quella successiva.
- Totalmente Osservabile: il modello ha accesso a tutte le informazioni necessarie per effettuare la classificazione al momento dell'addestramento e della valutazione.
- Noto: le regole e i criteri utilizzati per la classificazione dei titoli di lavoro sono noti e definiti nel modello.
- Singolo: l'ambiente considera l'azione di un unico agente, il nostro modello, che opera indipendentemente.

Actuators (Attuatori)

- Output del Modello: gli attuatori del nostro agente sono rappresentati dai meccanismi di classificazione che assegnano titoli di lavoro alle istanze in ingresso. L'output è il titolo di lavoro classificato che viene fornito in risposta alle caratteristiche dell'utente.

Sensors (Sensori)

- Input al Modello: i sensori del nostro agente sono le caratteristiche degli utenti, come età, esperienza, livello di istruzione e competenze. Questi dati vengono raccolti attraverso il sistema e costituiscono lo stream di input che il nostro modello utilizza per percepire l'ambiente ed effettuare le classificazioni.

2. Panoramica del modello

2.1. Teoria del modello

Nell'ambito del nostro studio, abbiamo selezionato due approcci distinti di apprendimento automatico per affrontare il problema di classificazione: l'albero decisionale (Decision Tree) e il classificatore a foreste casuali (Random Forest Classifier). L'impiego di questi due modelli ci consente di valutare e confrontare l'efficacia di un metodo semplice e interpretabile rispetto a uno più complesso e robusto all'interno del nostro specifico dominio applicativo.

- Albero Decisionale (Decision Tree)
  L'albero decisionale è un modello predittivo che mappa le osservazioni sulle caratteristiche di un oggetto a conclusioni riguardo al valore target. La sua struttura è quella di un albero binario, dove ciascun nodo interno rappresenta una "domanda" su una delle caratteristiche, ogni ramo rappresenta la decisione presa in risposta a quella domanda, e ciascun nodo foglia rappresenta un valore di classe (decisione finale). Questo modello è particolarmente apprezzato per la sua facilità di interpretazione, poiché è possibile visualizzare il processo decisionale in maniera intuitiva.

- Classificatore a Foreste Casuali (Random Forest)
  Il classificatore a foreste casuali, invece, è un ensemble di alberi decisionali, il che significa che costruisce una moltitudine di alberi decisionali durante l'addestramento e restituisce la classe che è la moda delle classi (classificazione) o la media delle previsioni (regressione) degli alberi individuali. La "casualità" è inserita nella costruzione degli alberi attraverso il metodo di bagging (bootstrap aggregating) e mediante la selezione casuale di sottoinsiemi di caratteristiche da utilizzare per dividere ciascun nodo. Questo modello è noto per avere una maggiore capacità predittiva e robustezza rispetto al singolo albero decisionale, riducendo il rischio di overfitting.

2.2. Architettura del modello

- Architettura dell'Albero Decisionale

  L'architettura del nostro modello di albero decisionale è stata definita con la seguente configurazione: per quanto riguarda l'architettura del modello basato su Albero Decisionale, abbiamo optato per mantenere i parametri di default forniti dalla libreria **scikit-learn**.
  In particolare, abbiamo utilizzato i seguenti parametri predefiniti:

  - **Criterio di Pura**: Gini; per misurare la qualità delle divisioni, cercando di massimizzare la purezza delle classi in ciascun nodo foglia.

  - **Profondità Massima**: Non specificata; permettendo così all'albero di espandersi fino alla massima profondità possibile.

  - **Numero Minimo di Campioni per Dividere un Nodo: 2.**

  - **Numero Minimo di Campioni per Nodo Foglia**: 1; consentendo anche ai singoli campioni di costituire una foglia.

- Architettura del Classificatore a Foreste Casuali

  L'architettura del Classificatore a Foreste Casuali è stata, invece, attentamente calibrata attraverso un processo di ottimizzazione iperparametrica. Utilizzando una combinazione di griglia di iperparametri e ricerca casuale (Random Search), abbiamo selezionato un set di parametri che hanno migliorato le prestazioni del modello sul nostro insieme di dati. La configurazione finale è stata la seguente:

  - **Numero di Alberi** (n_estimators): 369; questo è il numero di alberi utilizzati nel modello. Determinato attraverso una ricerca ottimale per bilanciare la capacità di calcolo e le prestazioni del modello

  - **Criterio di Pura**: Gini; come per l'albero decisionale, anche qui abbiamo adottato questo indice.

  - **Profondità Massima:** 13; rappresenta il livello massimo di profondità che ogni albero può raggiungere. È selezionata per evitare l'overfitting, consentendo al modello di imparare efficacemente senza memorizzare i dati.

  - **Peso delle Classi** (class_weight): None; usato per bilanciare le classi nel dataset, assegnando un peso maggiore alle classi meno rappresentate. Questo aiuta a ottenere un apprendimento equilibrato tra le classi.

  - **Stato Casuale** (random_state): 42; serve a garantire la riproducibilità dei risultati fissando il seme per la generazione di numeri casuali, che influisce sulla costruzione del modello.

3. Set di dati

3.1. Fonte

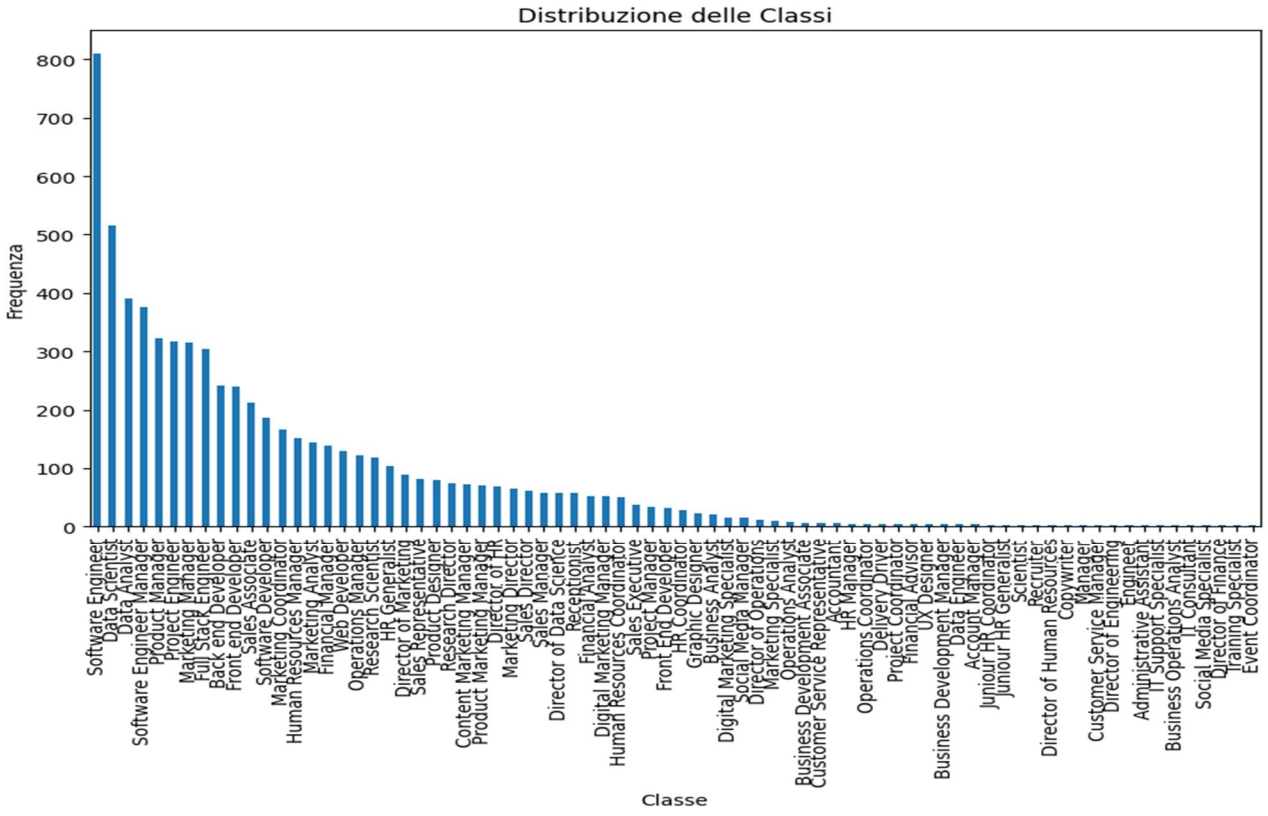
Il dataset utilizzato per questo progetto è stato acquisito da Kaggle al seguente link. Questo dataset contiene informazioni sulle retribuzioni in base al titolo del lavoro e al paese di provenienza dei dati.

3.2. Esplorazione dei dati

L'esplorazione dei dati è stata un passo essenziale nel comprendere la composizione del nostro dataset, permettendoci di identificare pattern, anomalie, e tendenze che avrebbero potuto influenzare le prestazioni dei modelli di classificazione.

Distribuzione delle Classi

Abbiamo esaminato la distribuzione delle classi della nostra variabile dipendente, "Job Title". Utilizzando la visualizzazione dei dati, abbiamo potuto osservare la frequenza delle diverse categorie professionali all'interno del dataset. Questo ha rivelato come alcune posizioni lavorative fossero più comuni di altre, suggerendo un potenziale squilibrio di classe. Un'analisi dettagliata della distribuzione delle classi è fondamentale perché squilibri significativi possono portare a modelli di classificazione che favoriscono le classi maggioritarie a scapito di quelle minoritarie.



3.3. Pre-elaborazione

Nel processo di pre-elaborazione dei dati, abbiamo adottato una serie di misure per garantire che il set di dati fosse accuratamente pulito e preparato per l'addestramento dei modelli di classificazione. Le operazioni eseguite sono state le seguenti:

- **Rimozione delle Colonne Irrilevanti**: abbiamo escluso dal dataset quelle colonne che non apportavano informazioni utili ai fini del problema in esame. Questa selezione è stata effettuata sulla base di una valutazione dell'importanza delle variabili e della conoscenza del dominio applicativo.

- **Gestione dei Dati Mancanti**: abbiamo condotto verifica della presenza di dati mancanti all'interno del dataset che ha avuto esito negativo.

- **Codifica delle Variabili Categoriche**: le colonne contenenti variabili categoriche sono state trasformate in formati adatti all'elaborazione mediante modelli di machine learning. In particolare, abbiamo convertito queste variabili in variabili booleane, permettendo così un loro utilizzo diretto negli algoritmi di classificazione.

- **Rimozione delle classi minoritarie**: l'identificazione delle classi minoritarie (classi in un dataset che hanno un numero significativamente inferiore di osservazioni o esempi rispetto ad altre classi) è stata effettuata manualmente osservando il grafico di distribuzione delle classi, e escludendo quelle istanze che presentavano una scarsa frequenza della variabile dipendente. Più specificatamente, abbiamo rimosso dal dataset le righe in cui la variabile di risposta si manifestava solo poche volte, allo scopo di prevenire distorsioni nei risultati del modello causate da campioni eccessivamente rari.

- **Aggiunta della Colonna "Skills"**: in una delle due versioni del dataset, abbiamo introdotto una nuova colonna "Skills", derivata dall'elaborazione di informazioni aggiuntive, al fine di esplorare la potenziale correlazione tra le competenze espresse e la variabile dipendente.

- **Codifica One-Hot della Variabile Dipendente**: infine, abbiamo applicato la tecnica di codifica One-Hot alla variabile dipendente. Questo approccio ha trasformato la variabile target in un formato adatto per l'analisi da parte dei nostri modelli, rappresentando ogni categoria con una nuova colonna binaria.

Ognuna di queste operazioni è stata fondamentale per assicurare che i dati fossero pronti e adeguati all'addestramento dei modelli, e per migliorare la qualità e l'efficacia del processo di apprendimento automatico.

4. Implementazione

4.1. Ambiente di sviluppo

Nel corso dello sviluppo del nostro progetto di classificazione, abbiamo optato per un ambiente di sviluppo che combinasse flessibilità, efficienza e facilità di integrazione. Di seguito sono elencati i componenti chiave del nostro ambiente di sviluppo:

- **Linguaggio di Programmazione**: abbiamo scelto Python per la sua vasta adozione nella comunità scientifica e di data science, nonché per la ricchezza delle sue librerie dedicate all'apprendimento automatico e all'analisi dei dati.
- **Piattaforma di Sviluppo**: per la scrittura e l'esecuzione del codice Python, abbiamo utilizzato Google Colab. Questa scelta ci ha permesso di beneficiare di un ambiente di sviluppo basato su cloud, facilitando la collaborazione e l'accesso a risorse computazionali elevate senza necessità di configurazioni hardware complesse.
- **Librerie per l'Analisi dei Dati e il Machine Learning**:
    - **Pandas**: utilizzata per la manipolazione e l'analisi dei dati. Questa libreria ci ha fornito strumenti potenti per la gestione dei dataset, consentendoci di eseguire operazioni di pulizia, esplorazione e trasformazione dei dati in modo efficiente.
    - **Scikit-learn (sklearn)**: scelta per lo sviluppo e la validazione dei modelli di classificazione, grazie alla sua vasta collezione di algoritmi di apprendimento automatico, metodi di pre-elaborazione dei dati e utilità per la valutazione delle prestazioni.
    - **NumPy**: fondamentale per il supporto a operazioni matematiche complesse e alla gestione di array multidimensionali. La sua integrazione con Pandas e Scikit-learn ha garantito un'elaborazione dei dati fluida e performante.

- **Integrazione e Deploy del Modello**:
    - **Flask**: abbiamo adottato Flask per sviluppare un'interfaccia web attraverso la quale interagire con il modello. Questo microframework web ci ha permesso di creare facilmente endpoint API per le richieste e le risposte del modello, facilitando l'integrazione del modello in applicazioni esistenti.
    - **Joblib**: per l'esportazione e il salvataggio dei modelli addestrati, abbiamo utilizzato Joblib. Questo strumento si è rivelato particolarmente efficace per la serializzazione di grandi strutture di dati, come i modelli di machine learning, consentendone un facile caricamento e riutilizzo senza la necessità di riallenarli.

L'insieme di queste tecnologie e librerie ha costituito la base per lo sviluppo, l'addestramento e l'integrazione del nostro modello di classificazione, permettendoci di affrontare con successo le sfide poste dal problema in esame.

4.2. Dettagli di implementazione

Il processo di sviluppo del nostro modello di classificazione è stato strutturato in quattro fasi principali, ciascuna delle quali ha richiesto l'adozione di specifiche tecniche e metodologie.

1. **Preparazione ed Esplorazione dei Dati**

   La fase iniziale ha visto la ricerca e selezione del dataset adeguato, seguita da un'accurata pulizia e analisi preliminare dei dati. Utilizzando la libreria Pandas, abbiamo caricato il dataset in un DataFrame, selezionato le colonne pertinenti eliminando quelle superflue, e identificato i valori mancanti attraverso **.isnull().sum()**. Per gestire le variabili categoriche, abbiamo adottato il metodo **.map()** per convertirle in formati booleani e utilizzato **.value_counts()** per analizzare la distribuzione delle classi nella variabile dipendente. Successivamente, abbiamo:

   - Suddiviso il dataset in set di addestramento e test;
   - Applicato la codifica One-Hot alla variabile dipendente mediante LabelEncoder;
   - Normalizzato le variabili numeriche con uno scaler.

2. **Addestramento del Modello**

   Per l'addestramento, abbiamo istanziato i modelli RandomForestClassifier e DecisionTreeClassifier. La valutazione iniziale del Decision Tree, basata su metriche quali precisione, accuratezza, recall e F1 score, ha fornito una prima indicazione delle prestazioni del modello. Analogamente, per il Random Forest, l'analisi di queste metriche, insieme a una valutazione attraverso convalida incrociata, ha evidenziato la necessità di un'ottimizzazione. Abbiamo quindi impiegato tecniche di ricerca degli iperparametri, come Grid Search e Random Search, per identificare la configurazione più efficace, ottenendo un significativo miglioramento delle prestazioni del modello.

3. **Integrazione del Modello**

   L'integrazione del modello in un'applicazione esistente è stata realizzata mediante l'uso del framework Flask, che ha permesso la creazione di un server HTTP. Abbiamo gestito le richieste in ingresso convertendo i dati ricevuti in DataFrame, gestendo le eventuali colonne nulle dovute a dati mancanti forniti dall'utente, e applicando il metodo .predict() per ottenere le previsioni del modello. L'output è stato poi tradotto in una forma comprensibile attraverso il metodo inverse_transform() del LabelEncoder.

5. Addestramento modello e Risultati

Durante la fase di addestramento, abbiamo perseguito due distinte strategie per valutare l'impatto della colonna "Skills" sulle prestazioni dei nostri modelli di classificazione, Decision Tree e Random Forest. Di seguito, sono descritti i dettagli e i risultati ottenuti da ciascuna strategia.

- **Strategia con Colonna "Skills"**

Nel primo approccio, abbiamo incluso la colonna "Skills" nel nostro dataset, procedendo con l'addestramento dei modelli sia Decision Tree che Random Forest sul set di training e successivamente testandoli sul set di test. Le prestazioni sono state valutate utilizzando metriche standard nel campo della classificazione:

- **Accuracy (Accuratezza)**: Misura la percentuale di predizioni corrette sul totale. Un valore elevato indica un alto grado di corrispondenza tra le previsioni del modello e le classi reali.
- **Precision (Precisione)**: Rappresenta la proporzione delle identificazioni positive che erano effettivamente corrette. Una precisione elevata indica una bassa frequenza di falsi positivi.
- **Recall (Sensibilità)**: Misura la proporzione di positivi reali che sono stati identificati correttamente. Un valore alto indica che il modello è capace di riconoscere la maggior parte dei casi positivi.

I risultati ottenuti sono stati notevolmente positivi, con valori di Accuracy, Precision, e Recall rispettivamente pari a 0.983, 0.988, e 0.988, dimostrando l'elevata efficacia dei modelli nel classificare correttamente le istanze.

- **Strategia senza Colonna "Skills"**

Nella seconda strategia, abbiamo escluso la colonna "Skills" concentrandoci esclusivamente sull'utilizzo del Random Forest. Inizialmente, le metriche di valutazione, compreso l'F1 Score (la media armonica tra Precision e Recall), erano pari a 0.67, indicando una prestazione moderata del modello.

Dopo aver eseguito una convalida incrociata, i risultati sono rimasti consistenti, spingendoci a adottare tecniche di ricerca degli iperparametri come Grid Search e Random Search. Queste tecniche si sono

concentrate su parametri chiave quali **n_estimators**, **max_depth**, e **class_weight**. Nonostante l'ottimizzazione, le metriche di valutazione non hanno mostrato miglioramenti significativi.

6. Interpretazione risultati

6.1. Prestazioni del modello

La valutazione delle prestazioni dei nostri modelli di classificazione, Decision Tree e Random Forest, ha seguito due distinte strategie di addestramento: una che includeva la colonna "Skills" e una che ne era priva. Questa analisi ha fornito risultati significativi, ma richiede una riflessione critica in merito all'overfitting.

- **Con la Colonna "Skills"**

Includendo "Skills", abbiamo ottenuto valori eccellenti nelle metriche chiave di valutazione per entrambi i modelli, con Accuracy, Precision, e Recall rispettivamente pari a 0.983, 0.988, e 0.988. Questi risultati indicano un'alta efficacia dei modelli nel classificare correttamente le istanze. Tuttavia, l'alta performance può essere influenzata da un overfitting, ovvero dal fatto che il modello ha appreso troppo bene i pattern specifici del set di addestramento, compromettendo la sua capacità di generalizzare su dati non visti.

- **Senza la Colonna "Skills"**

L'approccio senza "Skills", focalizzandoci sul solo Random Forest, ha mostrato prestazioni moderate con tutte le metriche intorno a 0.67. Anche se l'ottimizzazione degli iperparametri non ha portato a miglioramenti significativi, questa strategia evidenzia come l'esclusione di una caratteristica influente possa impattare negativamente le capacità predittive del modello.

7. Conclusioni

7.1. Riepilogo

Nel corso di questo progetto, abbiamo sviluppato e valutato modelli di classificazione basati su Decision Tree e Random Forest, esplorando l'impatto dell'inclusione di una caratteristica specifica, "Skills", sulle prestazioni dei modelli. Abbiamo adottato un approccio metodico che includeva la pulizia e l'esplorazione dei dati, l'addestramento dei modelli con e senza la caratteristica "Skills", e la valutazione delle prestazioni attraverso metriche standard come Accuracy, Precision, Recall e F1 Score.

I risultati hanno evidenziato che l'inclusione della colonna "Skills" ha notevolmente migliorato le prestazioni dei modelli, suggerendo l'importanza di questa caratteristica per la predizione accurata delle classi. Tuttavia, abbiamo anche rilevato potenziali segnali di overfitting, particolarmente in relazione al

modello che includeva "Skills", indicando che il modello potrebbe aver appreso troppo bene i dettagli specifici del set di addestramento a scapito della capacità di generalizzare a nuovi dati.

7.2. Limitazioni

Nel corso di questo progetto, ci siamo imbattuti in diverse sfide e limitazioni che hanno influenzato l'approccio alla modellazione e l'interpretazione dei risultati:

- **Selezione del Dataset**: una delle principali difficoltà incontrate è stata trovare un dataset adeguatamente allineato con il problema in esame. La limitata disponibilità di set di dati pre-esistenti che soddisfacessero pienamente i requisiti del nostro progetto ci ha costretti a un processo di selezione e adattamento più complesso.

- **Generazione della Colonna "Skills"**: l'assenza di dati direttamente correlati alla variabile di interesse ci ha portato alla decisione di generare artificialmente la colonna "Skills". Sebbene questa scelta abbia migliorato le prestazioni dei modelli, ha anche introdotto il rischio di overfitting, poiché il modello potrebbe aver appreso pattern troppo specifici legati a questa caratteristica artificiale, riducendo così la sua capacità di generalizzazione su dati non visti.

- **Overfitting**: la generazione della colonna "Skills" e l'alta dipendenza da questa caratteristica specifica hanno accentuato il problema dell'overfitting, evidenziando la sfida di bilanciare l'accuratezza del modello con la sua capacità di generalizzare. Questo sottolinea l'importanza di approcci metodici e cautelativi nella generazione di caratteristiche e nella selezione delle variabili.

- **Generalizzazione e Robustezza dei Modelli**: le limitazioni sopra menzionate sollevano interrogativi sulla generalizzazione e la robustezza dei modelli sviluppati. La capacità di questi modelli di performare bene su set di dati diversi da quelli utilizzati per l'addestramento rimane una questione aperta.

7.3. Prospettive future

Per affrontare queste limitazioni, progettiamo di:

- **Esplorare Fonti di Dati Alternative**: Ampliare la ricerca di dataset più adatti o esplorare metodi per integrare set di dati multipli potrebbe fornire una base di addestramento più ricca e variegata.

- **Valutazione Critica delle Caratteristiche Generate**: Approfondire l'analisi sull'impatto delle caratteristiche generate artificialmente, esplorando strategie per minimizzare il rischio di overfitting associato.

- **Tecniche Avanzate per la Riduzione dell'Overfitting**: Implementare e valutare l'efficacia di tecniche avanzate di regolarizzazione e di validazione, come la validazione incrociata k-fold o l'uso di metodi di ensemble, per migliorare la generalizzazione dei modelli.

Queste azioni rappresentano passi importanti verso il superamento delle limitazioni attuali e il miglioramento dell'efficacia e dell'affidabilità dei modelli di classificazione in futuri progetti di apprendimento automatico

8. Appendice

8.1. Codice sorgente

Il codice sorgente completo del progetto, inclusi script per la pulizia dei dati, l'addestramento dei modelli, la valutazione delle prestazioni e l'integrazione dei modelli tramite Flask, è disponibile pubblicamente nella nostra repository su GitHub. Questo permette di esaminare in dettaglio le implementazioni, le scelte metodologiche e tecniche adottate durante lo sviluppo del progetto. Inoltre, la repository offre la possibilità di scaricare il codice per replicare l'analisi, testare le prestazioni dei modelli con nuovi dati o estendere il lavoro con ulteriori ricerche e sperimentazioni.

- https://github.com/Fxller/RestartFIA.git

8.2. Riferimenti

- **Libri e materiale teorico:**
  - https://elearning.informatica.unisa.it/el-platform/course/view.php?id=950
- **Risorse online:**
  - **Scikit-Learn Documentation**: https://scikit-learn.org/stable/index.html
  - **Pandas Documentation**: https://pandas.pydata.org/pandas-docs/stable/
  - **Flask Documentation**: https://flask.palletsprojects.com/en/latest/