INTRODUZIONE

Il progetto mira ad analizzare e creare un modello predittivo che sia in grado di classificare, a partire da altre informazioni, la fascia di età di un determinato individuo. A tal proposito verranno utilizzati alberi di decisioni, random forest e reti neurali, per poi fare un confronto delle prestazioni.

DATASET

Variable Name	Role	Type	Demographic	Description	Units	Missing Values
SEQN	ID	Continuous		Respondent Sequence Number		no
age_group	Target	Categorical	Age	Respondent's Age Group (senior/non-senior)		no
RIDAGEYR	Other	Continuous	Age	Respondent's Age		no
RIAGENDR	Feature	Continuous	Gender	Respondent's Gender		no
PAQ605	Feature	Continuous		If the respondent engages in moderate or vigorous-intensity sports, fitness, or recreational activities in the typical week		no
BMXBMI	Feature	Continuous		Respondent's Body Mass Index		no
LBXGLU	Feature	Continuous		Respondent's Blood Glucose after fasting		no
DIQ010	Feature	Continuous		If the Respondent is diabetic		no
LBXGLT	Feature	Continuous		Respondent's Oral		no
LBXIN	Feature	Continuous		Respondent's Blood Insulin Levels		no

Il dataset è formato da sette features e un target, le variabili SEQN e RIDAGEYR non verranno utilizzate.

Il target è la variabile age_group, che può essere (Senior/non-senior),

le features a partire da RIAGENDR sono rispettivamente:

- Genere
- Attività sportiva
- BMI
- Glucosio
- Diabete
- Test di tolleranza al glucosio

- Insulina

Da segnalare un forte sbilanciamento della classe age_group, (circa 84% degli individui appartiene alla classe “Adult”), questo riscontra un ostacolo nella classificazione degli individui della classe “Senior”.

PREPROCESSING DATI

Dopo la selezione delle features, per poter permettere la classificazione binaria di un individuo secondo la sua fascia di età, viene mappata la variabile age_group in 0 se “Adult” o 1 se “Senior”.

Per la rete neurale le features sono state normalizzate sottraendo per la media e dividendo per la deviazione standard.

Per affrontare il forte sbilanciamento dei dati, è stata utilizzata una tecnica di oversampling, “SMOTE”, ovvero vengono generati nuovi esempi della classe minoritaria (utilizzando K-Neighbors) in modo da bilanciare il dataset. Questa tecnica ha aumentato le prestazioni di modelli come le reti neurali (in particolare, abbiamo un aumento di circa del 25% in precisione e recall, a scapito dell’accuratezza).

ALBERO DI DECISIONE

Il primo modello applicato è un albero decisionale

```
# Divide il dataset in set di addestramento e test
X_train, X_test, y_train, y_test = train_test_split(features, targetMapped, random_state=0)

# Addestra il classificatore ad albero di decisione
dtree = DecisionTreeClassifier(random_state=0)
dtree.fit(X_train, y_train)

# Effettua previsioni sul set di test
predictions = dtree.predict(X_test)
```

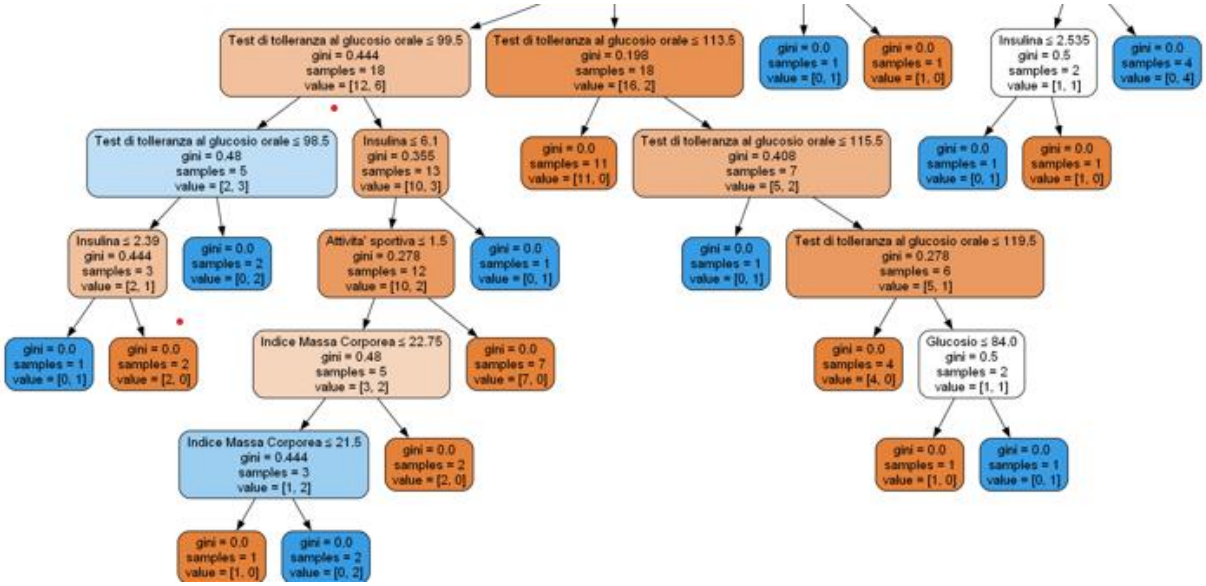
Iperparametri:

- random_state, per garantire la riproducibilità del modello ogni volta che il classificatore viene eseguito con gli stessi parametri e dati

- class_weight, ulteriore parametro per gestire un dataset sbilanciato, attribuisce un peso maggiore alla classe minoritaria quando impostato su “balanced”

Il dataset viene diviso in training set e test set, la divisione se non specificata è di default 70% per il training set e 30% per il test set.

Una volta creato l’albero con vengono fatte previsioni sul set di test, per poi permettere di valutare il modello.



Questo è un piccolo frammento dell’albero decisionale, ad ogni nodo o foglia abbiamo:

- Gini, ovvero l’impurità
- Samples, ovvero il numero di individui su cui stiamo lavorando
- Value, ovvero la classificazione che viene fatta dal modello

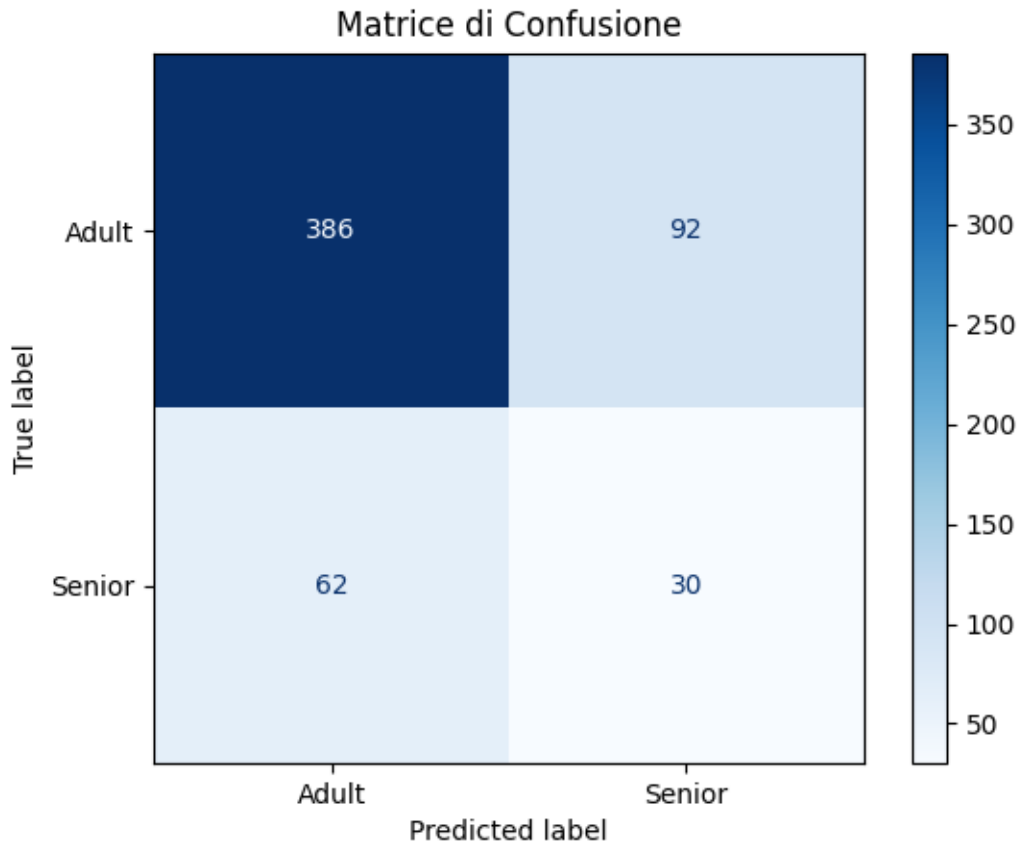
Nei nodi che non siano foglie troviamo in cima l’attributo utilizzato per la suddivisone binaria. Le foglie arancioni indicano gli individui classificati come “Adult” (0), quelle blu indicano gli individui classificati come “Senior” (1).

Per valutare il modello verranno utilizzate le seguenti metriche:

- Accuracy: 0.73
- Precision: 0.25

- Recall: 0.33
- F1 Score: 0.28

Per fornire ulteriore valutazione del modello viene calcolata anche la matrice di confusione:



Notiamo un alto numero di individui “Adult” classificati come “Senior”, e viceversa un alto numero di Senior classificati come “Adult”. In generale il modello sembra fare fatica nel classificare correttamente i “Senior”.

RANDOM FOREST

Il secondo modello utilizzato è random forest

```
# Addestra il classificatore Random Forest
rf = RandomForestClassifier(n_estimators=500, random_state=0, class_weight='balanced')
rf.fit(X_train_smote, y_train_smote)
```

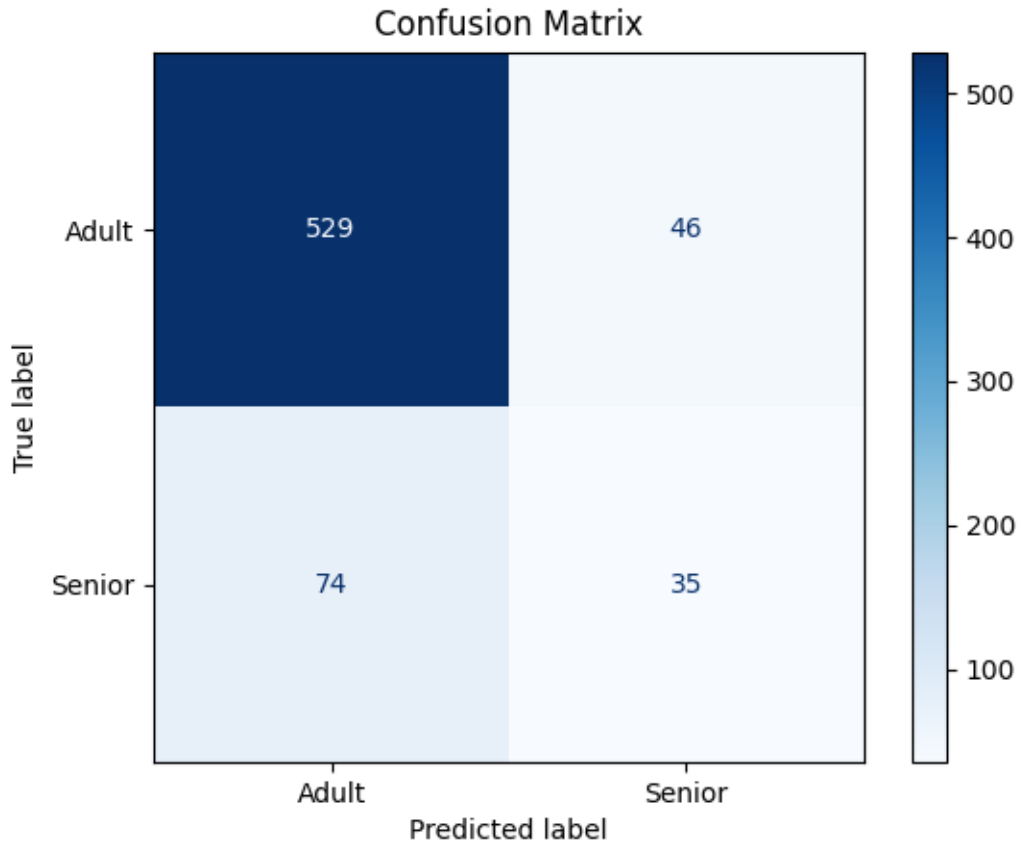
Iperparametri:

- N_estimators, ovvero il numero di alberi che vengono utilizzati
- Random_state, come nel decision tree
- Class_weight, come nel decision tree.

Le prestazioni sono le seguenti:

- Accuracy: 0.82
- Precision: 0.43
- Recall: 0.32
- F1 Score: 0.37

Matrice di confusione:



RETI NEURALI

Infine, reti neurali:

```python
# Definisce il modello di rete neurale sequenziale
model = Sequential([
    Dense(64, activation='relu', input_shape=(7,)),  # Primo strato con funzione di attivazione ReLU
    Dense(64, activation='relu'),  # Secondo strato nascosto con funzione di attivazione ReLU
    Dense(1, activation='sigmoid')  # Strato di output con funzione di attivazione sigmoide per classificazione binaria
])


# Compila il modello specificando ottimizzatore, funzione di perdita e metriche
model.compile(optimizer='adam',
              loss='binary_crossentropy',  # Funzione di perdita per la classificazione binaria
              metrics=['accuracy', Precision(), Recall()])

# Addestra il modello sui dati normalizzati
history = model.fit(X_train_scaled, y_train_smote, epochs=8, validation_split=0.2, class_weight = class_weights_dict)
```

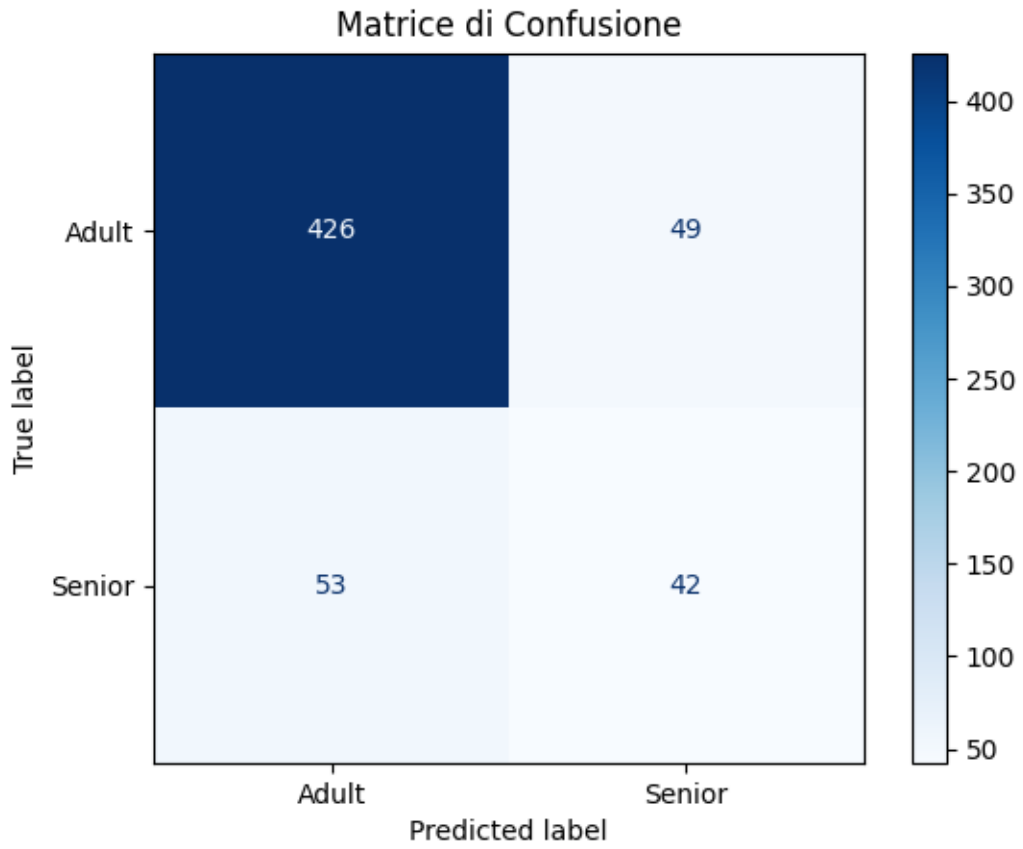
Iperparametri:

- Due strati con ciascuno 64 neuroni, uno strato di output
- Gli strati nascosti utilizzano come funzione di attivazione ReLU, lo strato di output utilizza la sigmoidea per fornire una classificazione binaria
- Il modello viene compilato con l'ottimizzatore Adam
- Funzione di perdita' binary_crossentropy, per fornire una misura di quanto il bene il modello stia facendo le sue previsioni
- Epochs, il numero di volte in cui l'algoritmo lavorerà sull'intero data set di addestramento
- Validation split, usato per valutare il modello durante l'addestramento utilizzando una parte del training set
- Class_weight, anche qui per bilanciare le classi

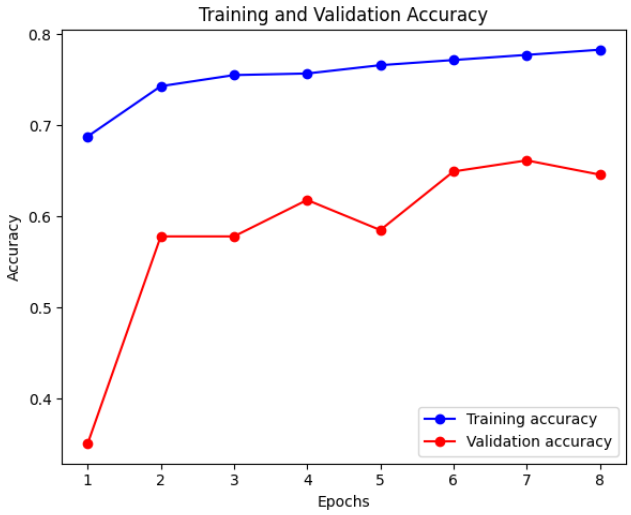
Le prestazioni sono le seguenti:

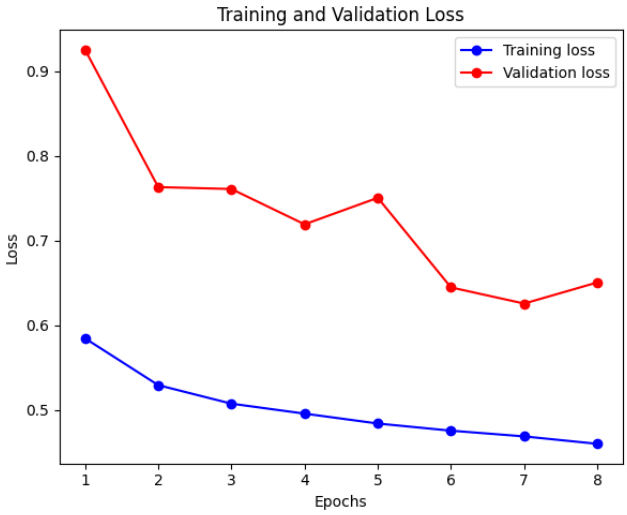
- Test set accuracy: 80.70%
- Precision: 0.4257
- Recall: 0.4526
- F1 Score: 0.4388

Matrice di confusione:



Curve di apprendimento:





MODELLI A CONFRONTO

Alberi di decisione:

- Intuitivo e facilmente interpretabile
- Difficoltà nell'individuare individui "Senior"
- Tende all'overfitting

Random forest:

- Riduzione dell'overfitting
- Meno interpretabile rispetto ai decision tree
- Netto miglioramento nell'individuare "Senior"

Reti neurali:

- Risultati migliori nella classificazione
- Più esigenti a livello computazionale

In conclusione, con queste implementazioni dei modelli, sembrerebbe che le reti neurali portino i migliori risultati, tuttavia, ci sono margini di miglioramento in tutti e tre i casi. Gli iperparametri utilizzati sono stati testati e dovrebbero portare i risultati migliori senza sacrificare troppo a livello computazionale e/o causare overfitting.