INGEGNERIA DELLA CONOSCENZA

Progetto a cura di Pompeo Francesco Lippolis

Introduzione

Sono Pompeo Francesco Lippolis, matricola 597996.
Il progetto che ho voluto sviluppare prende spunto dalle competizioni online sul Machine Learning come Kaggle, dove si cerca di costruire il modello che meglio predice la classe di appartenenza di esempi, non visti prima, appartenenti ad un dataset dato.
Ho cercato un dataset adeguato al compito, imbattendomi nel sito del Machine Learning Repository dell'Università della California, Irvine dove il dataset sull'insorgenza del diabete ha attirato la mia attenzione in quanto è un problema presente nella mia famiglia.

Ho sviluppato una serie di classificatori addestrati sul dataset:

- Regressione Lineare
- Regressione Logistica
- Albero Decisionale
- Rete Neurale
- Support Vector Machine
- Random Forest
- ADAptive BOOSTing
- K-Nearest Neighbors
- Naive Bayes

Li ho successivamente valutati e ho scelto il modello più adeguato al compito, tra quelli sviluppati. Il vincitore è stato utilizzato per sviluppare una semplice applicazione che predice il rischio di insorgenza del diabete.

Software utilizzato

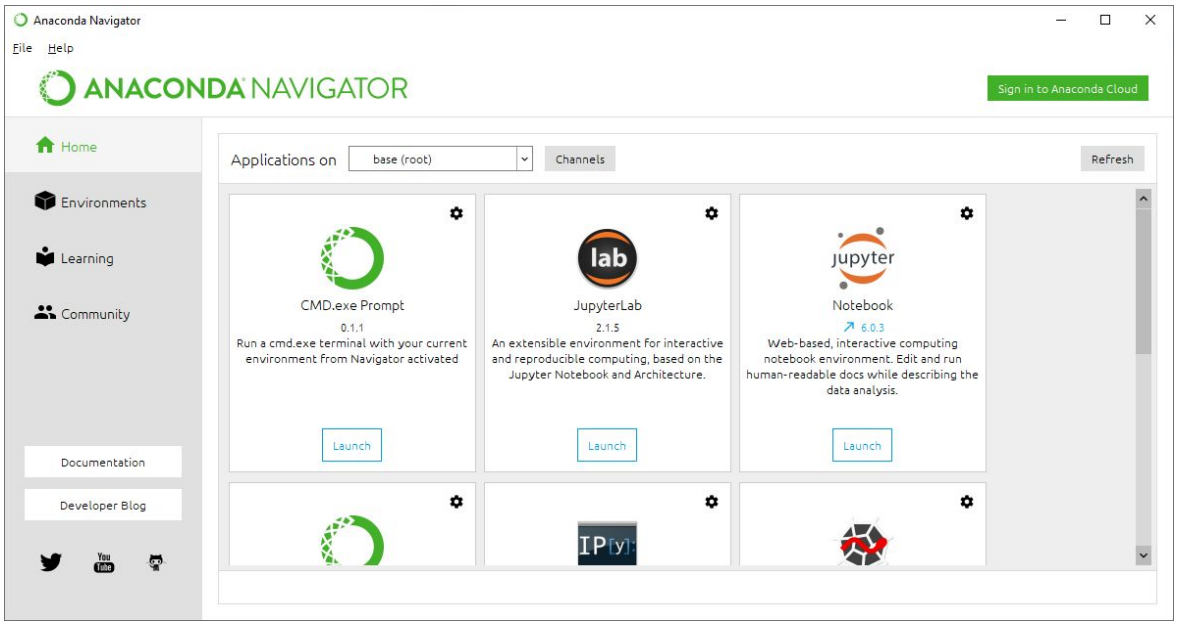
Per sviluppare i vari classificatori, ho usato l'applicazione Jupyter in quanto consente di creare e condividere documenti contenenti codice, equazioni, grafici e testo descrittivo.
Il codice utilizzato è in Python, con alcune istruzioni dell'estensione IPython del linguaggio.
Le librerie sono pandas, numpy, matplotlib, scikit-plot, IPython, scikit learn, tensorflow 2.0 e keras tuner.

Successivamente ho sviluppato una semplice applicazione in Python che predice il rischio di insorgenza del diabete utilizzando il modello considerato migliore tra quelli sviluppati.

Ho sviluppato la GUI dell'applicazione usando la libreria PySimpleGUI.

È possibile visualizzare il lavoro effettuato nello sviluppo dei vari modelli senza installare alcun software: online visualizzando il file nel repository oppure tramite l'export html ICon project.html. Il notebook contiene codice, risultati dell'esecuzione dello stesso, il tutto brevemente commentato.

Per visionare la versione interattiva del notebook (ICon project.ipynb) è necessario installare Jupyter sul proprio PC. Il modo più semplice è installando la distribuzione Anaconda, che contiene Jupyter, Python e molte delle librerie utilizzate. Una volta installato si potrà far partire il server Jupyter notebook dall'applicazione Anaconda Navigator:



La maggior parte delle librerie sono pre-installate o installabili tramite Anaconda Navigator

Le librerie mancanti possono essere installate tramite il package installer for python (pip):

Pandas: pip install pandas
Numpy: pip install numpy
MatPlotLib: python -m pip install -U matplotlib
Scikit-plot: pip install scikit-plot
IPython: pip install ipython
Scikit learn: pip install -U scikit-learn
Tensorflow 2: pip install tensorflow
Keras Tuner: pip install -U keras-tuner
PySimpleGUI: pip install pysimplegui

Preparazione del dataset

Per permettere alla maggior parte dei classificatori di poter operare sul dataset, ho effettuato la codifica delle etichette (Maschio, Femmina; Sì, No; Positivi, Negativi) in 0 e 1.
Normalmente etichette di categoria andrebbero codificate con il one-hot encoding; tuttavia, essendo i valori di etichette binari per ogni colonna, ho preferito codificare i valori della colonna come booleano rispetto alla relazione; ad esempio maschio(individuo): 1 se l'individuo è uomo, 0 se l'individuo è donna; questo metodo di descrizione risulta più compatto.
Ho successivamente discretizzato l'età in fasce d'età: ogni individuo è classificato secondo l'età più vicina in decadi. Questo è utile per limitare la presenza di outliers: se nel dataset ogni individuo con, per esempio, 24 anni d'età ha il diabete, un classificatore potrebbe essere tentato di classificare ogni 24enne come positivo, quando ciò è, ovviamente, falso.
Ho poi normalizzato il dataset, portando l'età su valori tra 0 e 1, in quanto determinati modelli si comportano male se addestrati su valori con scale diverse.

Fase di addestramento

Ho suddiviso il dataset in dati di training e di test con proporzione 4:1, considerando che un numero più alto di dati di training avrebbero reso il test set troppo piccolo.

La maggior parte dei modelli è stata addestrata usando un metodo di Cross validation (Lo Stratified K-Fold Cross Validation) per poter mettere a punto i parametri del modello.
L'unica eccezione è il Neural Network, addestrato sempre dividendo il set in training e validation per la messa a punto dei parametri, ma senza uno schema di CV (non supportato direttamente da Keras Tuner).
I parametri generalmente si riferiscono o alla struttura interna del classificatore o a parametri di regolarizzazione. Descrizioni specifiche per ogni modello sono presenti nel notebook.

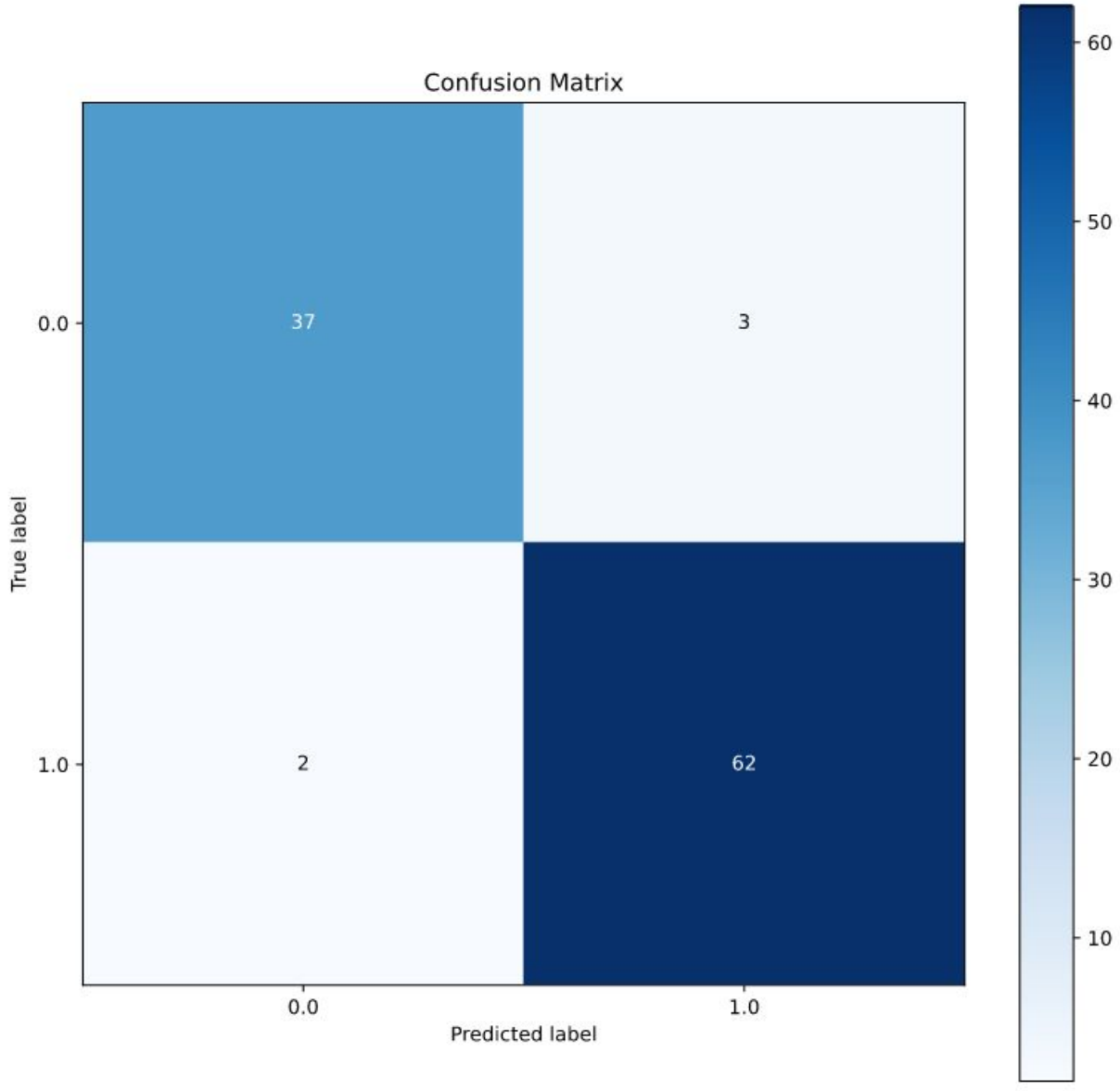
Raccolta di metriche

Per ogni modello sono state raccolte le seguenti metriche: sul training set è stata calcolata l'accuratezza (per valutare un eventuale overfitting dei modelli) invece sul test set vengono calcolati accuratezza, precisione e richiamo e i valori necessari a mostrare graficamente la matrice di confusione e la curva di precisione-richiamo.

Scelta del modello

Ho valutato la performance di ogni modello sul test set (non usato nell'addestramento)
Il modello che ho scelto tra i vari addestrati è la SVM, con le seguenti metriche:

	Train accuracy	Test accuracy	Precision	Recall
Support Vector Machine	99.52	95.19	95.38	96.88



L'ho scelto in quanto è il classificatore con la minor presenza di Falsi Negativi (alto richiamo)
Ogni falso negativo corrisponde ad un diabetico a cui non viene diagnosticata la malattia, cosa che vogliamo decisamente evitare.

L’ADAptive BOOSTing presenta lo stesso numero di Falsi Negativi, tuttavia ho scelto la SVM in quanto ha un numero decisamente più basso di Falsi Positivi; una metrica importante, seppur secondaria rispetto ai Falsi Negativi, in quanto un falso positivo corrisponde ad un esame di accertamento prescritto in più, non grave quanto una malattia non diagnosticata ma certamente da evitare.