Sopra ogni algoritmo implementato vi è la spiegazione, in poche righe, di cosa svolge ogni funzioni che ho usato.

Prima di discutere dei risultati ottenuti dai modelli che ho implementato reputo necessario spiegare come ho trattato i dati prima di darli in input ai vari modelli.

Per quanto riguarda la divisione del dataset ho sempre diviso nella seguente maniera: 60% training set, 20% validation set e 20% testing set. Per ottenere risultati migliori ogni volta ho implementato la model selection tramite k fold cross validation (k=3) in modo da trovare gli iperparametri che permetto di far performare al meglio i nostri modelli.

Per quanto riguarda il pre-processamento dei dati, a prescindere dalla tipologia del problema ho sempre eliminato dal dataset le prime due colonne (URL, timedelta) le quali non sono predittive, cioè non influenzano il risultato della shares che otterrà l’articolo. In base al problema ho modificato diversamente i dati:

* Nel caso della Classificazione ho sempre aggiunto al dataset una colonna di valori binari, tamite un’apposita funzione. Se il valore associato a un sample in questa colonna è 1 indica che questo avrà un numero di condivisioni > 1400, altrimenti minore.

Nel modello DecisonTree ho manipolato i dati di tutte le feature discretizzandoli cosi da ottenere un albero di decisione binario. Per la discretizzazione di questi per ogni colonna mi sono calcolato la media e l’ho usata come soglia per sostituire i valori in base a se fossero sopra o sotto tale trashold. Gli altri modelli li ho sempre normalizzati , più precisamente li ho normalizzati in maniera tale che avessero una media zero e una deviazione standard unitaria.

* Nel caso della regressione anche ho sempre normalizzato i dati secondo la tecnica descritta sopra.

La scelta della normalizzazione dei dati deriva, come detto anche a lezione, dal fatto che questa tecnica migliora i risultati ottenuti da un modello. Un esempio di miglioramento di un modello è la rimozione degli "outliners" cioè valori molto estremi in una certa feature che possono disturbare l'apprendimento del modello. La normalizzazione attenua l'impatto di questi valori riducendoli nel loro range normalizzato.

**Regressione**

- Regressione senza regolarizzazione (RMSE: 7319.2188): Questo è il modello di base, che non utilizza alcuna tecnica di regolarizzazione per evitare l'overfitting. Implementata tramite Gradient Descent con una certa tolleranza per comprendere quando si è giunti a convergenza e uscire dal ciclo delle epoche.

- Regressione con regolarizzazione Lasso (RMSE: 7314.9264): La regressione Lasso utilizza la regolarizzazione L1 per ridurre l'overfitting e selezionare le variabili più rilevanti. In questo caso, il modello Lasso ha ottenuto un RMSE leggermente inferiore rispetto al modello senza regolarizzazione, il che suggerisce che potrebbe esserci una leggera riduzione dell'overfitting. Implementazione con Gradient Descent.

- Regressione con regolarizzazione Ridge (RMSE: 7321.5825): In questo caso, il modello Ridge ha ottenuto un RMSE leggermente superiore rispetto al modello senza regolarizzazione. Una possibile spiegazione per questo risultato potrebbe essere che la regolarizzazione Ridge sta cercando di ridurre la complessità del modello penalizzando i coefficienti delle variabili meno importanti, il che potrebbe portare il modello a adattarsi meno ai dati e di conseguenza performare peggio. Sottolineo come la regressione con regolarizzazione Ridge l’ho implementata attraverso la forma chiusa invece che con il Gradient Descent, questo perché ho notato come tale forma ottenesse risultati leggermente migliori.

- Regressione con regolarizzazione Elastic (RMSE: 7319.0741): La regressione Elastic Net combina le regolarizzazioni L1 e L2, cercando di ottenere il meglio da entrambi i metodi. In questo caso, il modello Elastic Net ha ottenuto un RMSE simile al modello senza regolarizzazione e al modello Lasso. Questo suggerisce che la combinazione di entrambe le regolarizzazioni non ha portato a miglioramenti significativi. Anche questa regressione è stata implementata tramite Gradient Descent.

- k-Nearest Neighbors (RMSE: 7634.2372): Il modello k-NN è un modello non parametrico che si basa sulla distanza tra i samples per effettuare predizioni. In questo caso, a dispetto del problema di classificazione, il modello k-NN non ha avuto in generale una ottima previsione sul target rispetto agli altri modelli, bensì nella media. Questo potrebbe essere dovuto al fatto che KNN è un algoritmo istance based e che cerca di prevedere la popolarità di una notizia basandosi sulla sua similarità con altre notizie nel dataset (facendo table look up per cercare i sample più vicini a quello di cui si vuole prevedere la ground truth e le loro label) e quindi proprio perché si basa su altri sample vicini non è in grado di cogliere correlazioni più complicate tra le variabili indipendenti e la variabile dipendente. In aggiunta, il risultato potrebbe essere influenzato negativamente da una penalizzazione dovuta alla previsione di un campione di test che si basa sulla media delle etichette corrette dei campioni vicini. Pertanto, potrebbe verificarsi la situazione in cui i campioni vicini non siano sufficientemente simili e, di conseguenza, al campione in questione verrebbe assegnata una previsione distante da quella reale. Nell’implementazione uso la funzione cdist di scipy.spatial.distance per calcolare le distanze euclidee perché il calcolo delle distanze realizzato da me tramite una scansione del set di training impiegava un tempo eccessivo.

- Rete neurale (RMSE: 10016.8902) (valori variabili a causa dell'inizializzazione casuale dei pesi): Le reti neurali sono modelli potenti e flessibili che possono apprendere rappresentazioni complesse dei dati. Tuttavia, in questo caso, la rete neurale ha ottenuto un RMSE significativamente superiore rispetto agli altri modelli, il che suggerisce che potrebbe essere non configurata correttamente per questo problema, suppongo sia legato al vincolo posto sul numero di middle layer.

Quindi i modelli di regressione lineare hanno ottenuto i valori migliori di Rmse, specificatamente il più performante è stato l’algoritmo di regressione lineare che sfrutta la regolarizzazione di Lasso (rmse: 7314.9264).

Osservando le performance degli altri modelli si nota che quelli lineari hanno ottenuti i migliori risultati quindi suppongo che vi sia una relazione lineare tra le feature e la variabile target. I modelli non lineari come knn e rete neurale non riescono a catturare questa relazione, quindi ottengono prestazioni peggiori.

E’ importante sottolineare come la differenza tra le prestazioni conseguite tra i modelli di regressione è abbastanza piccola, quindi potrebbe essere utile esplorare ulteriormente i dati e sperimentare con diverse tecniche di pre-elaborazione e di Feature selection per ottenere risultati migliori.

**Classificazione**

- Albero di decisione: Ha ottenuto buoni risultati con un’ accuratezza del 60.39% (altezza massima 6) che può essere attribuita alla sua capacità di identificare pattern complessi all'interno del dataset e relazioni non lineari fra features e ground truth. Gli alberi di decisione sono algoritmi che inizialmente in fase di training costruiscono una struttura ad albero per dividere il dataset in sottoinsiemi in base alle caratteristiche degli input e successivamente in fase di testing utilizzano la struttura ad albero creata in fase di training per classificare le istanze di test. Implementato tramite una classe Nodo che rappresenta gli elementi che compongono l’albero e una classe DecisionTreeClassifier che rappresenta il classificatore dell'albero decisionale. La funzione grow\_tree è la funzione che popola l’albero di decisione in fase di training per poi usarlo in fase di testing.

- Regressione lineare: eseguita sia senza regolarizzazione che con varianti Ridge, Lasso ed Elastic, ha mostrato un'accuratezza del 58%. La regressione lineare assume una relazione lineare tra le variabili indipendenti (feature) e la variabile dipendente (ground truth), tuttavia, potrebbe non essere in grado di catturare al massimo la complessità del dataset se ci sono relazioni non lineari o variabili non rilevanti per la classificazione. Inoltre sottolineerei come il fatto che i risultati ottenuti con 3 diverse regolarizzazioni (no regolarizzazione, Ridge ed Elastic) siano identici potrebbe essere dovuto al fatto che la regolarizzazione non ha avuto un impatto significativo sulla prestazione del modello. Invece per quanto riguarda il fatto che la regressione lineare con regolarizzazione Lasso abbia ottenuto la stessa accuracy delle altre lo riconduco alla trashold posta nella predict, usata per ottenere la previsione di una classificazione, magari con valori differenti di questa si sarebbe discostata dagli altri risultati di classificazione con regressione. Infatti a differenza delle altre regressioni questa ha un vettore dei pesi totalmente differente (stampato a schermo nelle ultime celle) , il che sottolinea come la regolarizzazione ha impattato sul modello, ma comunque la accuracy ottenuta sia uguale alle altre. Implementazione: a differenza dell’implementazione per il problema di regerssione, qui anche la ridge è stata implementata con Gradient Descent

-Regressione Logistica: accuracy del 64.10% la più alta fra i modelli. Dato che i risultati migliori sono stati ottenuti da modelli non lineari (Logistica, Knn, Reti Neurali, Decisiontree) reputo che questo modello abbia un ottima accuracy per la presenza di relazioni non lineari fra dati in input e la label e a quanto risulta la funzione sigmoide, a cui viene applicata una soglia (0.5) per la classificazione, coglie molto bene tale relazione.

- K-Nearest Neighbors (KNN): L'accuratezza del 63.08% ottenuta con il KNN può essere attribuita alla capacità dell'algoritmo di classificare un'istanza di test assegnandole l'etichetta più comune tra i suoi K vicini più prossimi (sulla base della distanza dei sample nel dataset) nel training set. In generale l'algoritmo funziona bene quando i dati sono uniformemente distribuiti nello spazio delle caratteristiche. Se la distribuzione è molto diversa e i gruppi sono molto diversi tra loro, questo potrebbe influenzare negativamente le prestazioni del kNN. Avendo ottenuto una accuracy alta ciò potrebbe indicare che il dataset è omogeneo e l'omogeneità del dataset potrebbe essere dovuta al fatto che ho eseguito una modellazione del dataset normalizzando le features.

- Rete neurale: La rete neurale ha ottenuto un'accuratezza del 61% (con valori variabili a causa dell'inizializzazione casuale dei pesi). Le reti neurali sono modelli complessi in grado di catturare relazioni non lineari complesse nel dataset. Reputo che la rete neurale non abbia ottenuto i risultati migliori, come invece mi sarei aspettato, perché la performance di questi modelli è fortemente influenzata dall'architettura della rete e noi abbiamo posto un forte vincolo su questa cioè quello di avere un singolo middle layer. Implementazione: a dispetto di come è stata implementata per il problema di regressione qui l’ho implementata creando la sua classe

L'accuratezza più alta è stata ottenuta dal Regressione Logistica 64.10%, seguita da k-NN Classifier con 63.08%, ciò nonostante anche il modello più accurato non ha raggiunto una accuracy così sostanzialmente elevata, il che suggerisce che il problema potrebbe essere complesso. Performance migliori potrebbero essere ottenute utilizzando modelli più complessi come le reti neurali con architetture più articolate, o modelli ensemble come Random Forest, AdaBoost e Gradient Boosting, che spesso forniscono ottime prestazioni combinando più modelli.

Tuttavia, reputo interessante analizzare i risultati ottenuti dai modelli in ulteriori tre metriche: Precision, Recall e F1.

La precisione misura la proporzione di predizioni positive corrette fatte dal modello rispetto al totale delle predizioni positive fatte dal modello. In altre parole, la precisione indica quanto il modello è preciso nel classificare correttamente gli esempi positivi. La recall misura la capacità del modello di identificare correttamente tutti gli esempi positivi presenti nel set di dati. Quindi questa metrica evidenzia la proporzione di esempi positivi correttamente identificati dal modello rispetto al numero totale di esempi positivi presenti nel set di dati.

Sostanzialmente la precisione si concentra sulla qualità delle predizioni positive fatte dal modello, mentre la recall si concentra sulla capacità del modello di identificare correttamente gli esempi positivi senza trascurarne nessuno.

F1-score tiene conto sia della capacità del modello di identificare correttamente gli esempi positivi (recall) che della sua capacità di evitare falsi positivi (precision). È particolarmente utile quando le classi di interesse sono sbilanciate, cioè quando una classe è rappresentata in modo significativamente maggiore rispetto all'altra

Considerando la precision, il k-NN Classifier ottiene il valore più alto (0.66), seguito dal Logistic regression (0.64) e dal NN Classifier (0.61)(ha valori variabili a causa dei pesi inizializzati random). Questo indica che il k-NN Classifier è più preciso nel classificare correttamente gli esempi positivi rispetto agli altri due modelli.

Il recall è più alto lo hanno le regressioni lineari (0.93). Ciò significa che sono in grado di identificare correttamente una maggiore proporzione di esempi positivi rispetto agli altri modelli. Hanno la capacità di catturare la maggior parte degli articoli con un alto numero di condivisioni nel set di dati.

F-1 score le regressioni ottengono il punteggio più alto avendo una così alta recall, seguite dalla Logistic Regression (0.678) e dal Decision Tree (0.672). Quindi le regressioni hanno un migliore equilibrio tra precisione e recall rispetto agli altri modelli.

In base a queste analisi si può notare come anche avendo la accuracy più alta il modello di regressione logistica non sempre è il modello migliore se si considerano altre metriche. Nelle altre metriche non ha i risultati migliori ma comunque si pone sempre fra i primi quindi reputo che sia il modello più performante che esegue classificazione fra tutti quelli implementati