Metodo di Machine Learning per la stima delle attività di un progetto Software

1.1 Introduzione

Nell'ambito del *project management* l'attività principale svolta da un project manager è quella di gestire e portare a termine (con successo) progetti con risorse limitate entro un certo limite di tempo. Per risorse si intende l'effort speso dal team che lavora sul determinato progetto.
La gestione del progetto ad oggi è demandata completamente al project manager tramite azioni del tutto manuali (planning, meeting, revisioni, ..) allo scopo di completare il progetto in tempo e con le risorse disponibili.
Ad oggi inoltre la previsione del completamento di un progetto viene effettuata tramite una stima fornita dai vari membri del team, ma questa stima attualmente è influenzata dagli errori umani di valutazione delle varie persone del team.
La seguente relazione illustrata un metodo di Machine Learning che permette di aiutare il project manager a stimare l'effort delle attività del progetto in modo da capire se saranno in linea con la deadline valutandone i rischi e modificando conseguentemente la pianificazione. Infatti un approccio di ML, che impara dall'esperienza dei progetti precedenti elimina l'influenza degli errori umani sulla stima e permette di prevedere se un'attività sarà completata in tempo o meno.

1.2 Problema (formale)

Dato un progetto *p* formato da *m* attività, creare un algoritmo di Machine Learning che dall'esperienza pregressa dei progetti, fornisca una stima e_m (in ore) delle *m* attività. Inoltre si vuole calcolare la classe d'appartenenza *k* dell'attività che indica se una attività appartenente a un progetto sarà completato in tempo (*true*) o no (*false*).

1.3 Esempio

Un progetto software su cui lavorano 3 sviluppatori (100h in totale) è formato da 6 attività. Le attività contengono informazioni sul tipo di attività, sulla priorità, la persona che ci lavora, se è stato completato in tempo e molte altre informazioni.
L'algoritmo di Machine Learning stima i giorni necessari per il completamento dei vari tasks considerando i progetti passati (di cui le attività sono state stimate manualmente) e stima se un task verrà completato in tempo o meno. Quindi in questo caso per le 6 attività avremo 6 stime (in ore) e un valore booleano che indica se il task verrà completato in tempo.
Ad esempio, dopo la computazione l'output fornito dall'algoritmo di Machine Learning riguardante le sei attività è così composto:

- *Attività 1*: 5 ore, true
- *Attività 2*: 34 ore, false
- *Attività 3*: 23 ore, true
- *Attività 4*: 12 ore, true
- *Attività 5*: 10 ore, false
- *Attività 6*: 25 ore, true

Il totale delle stime è di 109 ore, quindi in ritardo di 9 ore con le risorse attualmente disponibili di 100h. Inoltre l'algoritmo di Machine Learning deve permettere di capire se è l'attività sarà classificata

come non completata (quindi in ritardo) o no (quindi completata secondo il tempo previsto) Nell'esempio due tasks sono state classificati come attività che non saranno completate in tempo. Grazie a questo algoritmo il project manager potrà pianificare le attività cercando di rispettare le risorse disponibile ed evitando eventuali ritardo sulla consegna del progetto.

2 Dataset

Il dataset utilizzato per il training e il testing dell'algoritmo di Machine Learning descritto in questa relazione è specifico nell'ambiente del Project Management in ambito di sviluppo software. In particolare i dati riguardano le attività relative alle varie versioni di un software di una multinazionale. Infatti una versione del software (ad. esempio v.1.2) può essere considerato un progetto formato da *t* attività che passerà dalle fasi di planning, sviluppo, testing, rilascio, manutenzione, ecc..
Il dataset è formato da *m* osservazioni che indicano le attività dei vari progetti e da n caratteristiche (features) che descrivono le attività, come il tipo di attività, la priorità, la persona che ci lavora, lo status, se è stato completato in tempo, la versione a cui appartiene l'attività e molte altre informazioni. Il campo fixVersion indica il progetto a cui appartiene il task.
La sorgente dei dati proviene da un tool per la gestione di progetti software, chiamato Jira (di Atlassian). Essendo un progetto a scopo didattico i dati sensibili sono stati modificati manualmente rendendoli anonimi.
I valori di verità presenti nel dataset sono due:

- la stima del task, che ricordiamo sono stime effettuate manualmente dai vari membri del team
- il completamento del task, che indica se il task è stato completato in tempo (true, false)

Durante la computazione, alcune features sono state eliminate in modo da ridurre la complessità. In particolare è stata eseguita una feature selection manuale, secondo la conoscenza pregressa sui dati (dettagli nella sezione dell'algoritmo).

Nome dataset: *tasks.csv*

Esempio di osservazione (dopo la feature selection):

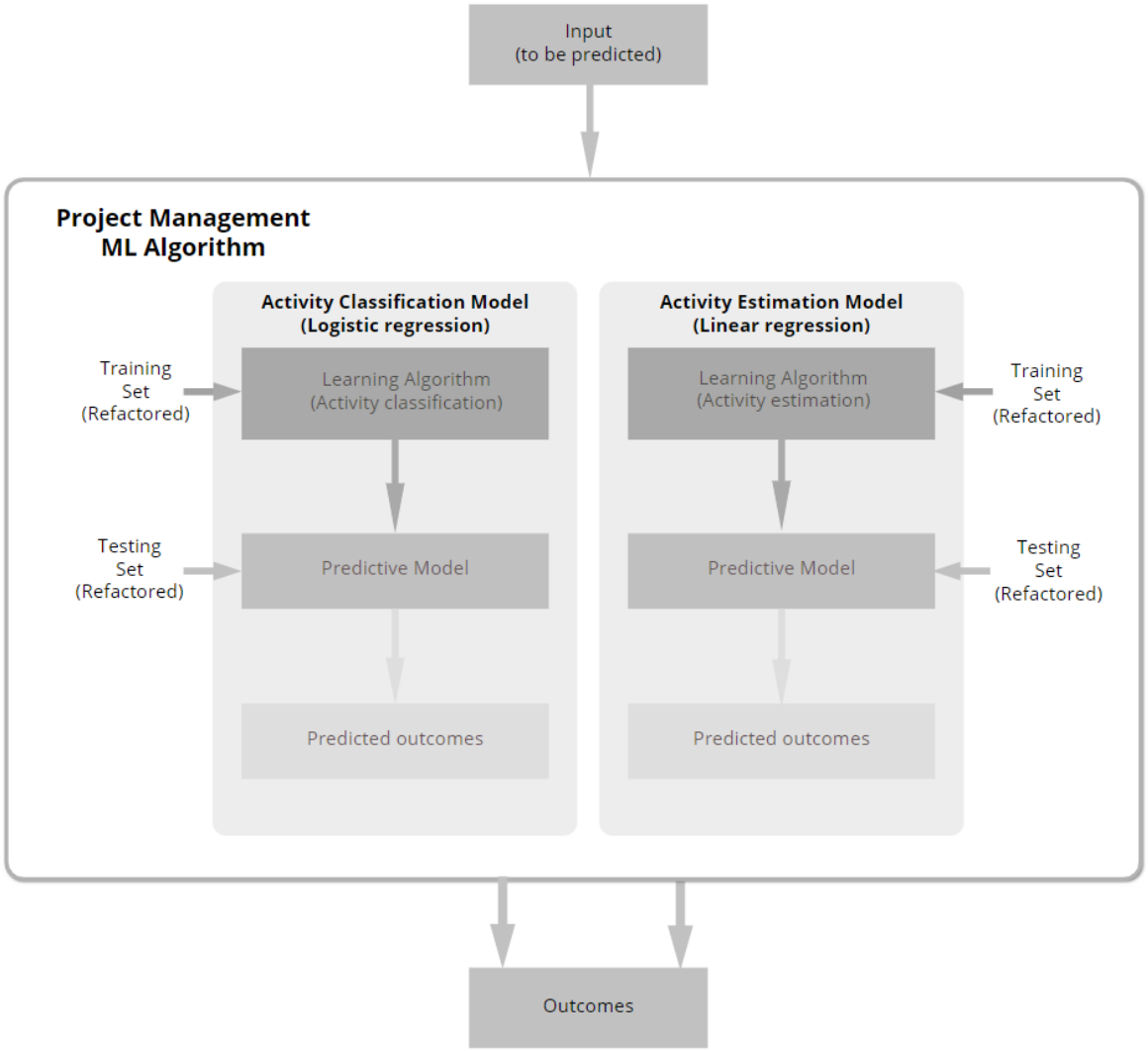
Issue key	Issue Type	Status	Priority	Resolution	Assignee	Reporter	Creator	Last Viewed	Resolved	Release Date	Versi
KKK-19201	story	1	Minor	Won't Fix	r.asdf	andrssan	andrssan	03/May/21 7:53 PM	2021-01-29 12:50:00	2021-03-01 11:23:00	Rel
KKK-19194	Bug	0	Major	Working as expected	i.covrig	andrssan	andrssan	03/May/21 7:53 PM	2021-02-11 13:57:00	2021-03-01 11:23:00	Rel
KKK-19160	Task	1	Major	Done	member2ymanski	member3	member3	03/May/21 7:53 PM	2021-02-11 14:53:00	2021-03-02 11:23:00	Rel
KKK-								03/May/21	2021-03-	2021-	Rel

3.1 Algoritmo

I problemi che l'algoritmo deve risolvere sono due: la stima in ore dell'attività e la classificazione dell'attività se è stata completata in tempo o no.

La risoluzione del primo problema si basa sull'algoritmo di Machine Learning della Regressione Lineare. Infatti il valore di verità da calcolare è un valore numerico reale che indica la stima delle ore da lavorare per completare l'attività.
La risoluzione del secondo problema si basa invece sull'algoritmo di Machine Learning della Regressione Logistica. Infatti il valore di verità da calcolare è un valore booleano che indica se l'attività è stata completata in tempo o no.

Di seguito uno schema generale dell'algoritmo di ML implementato:



Nel dettaglio l'algoritmo è diviso in quattro blocchi principali:

1. Refactor dei dati: ai dati sono state applicate alcune trasformazioni in modo da mantenere solo le informazioni necessarie e ridurre la complessità. In dettaglio:
    a. Sono state eliminate le features che in tutti samples risultano nulli
    b. E' stata applicata una features selection manuale basata dall'esperienza nell'ambito. Sono state eliminate features, per lo più testuali, che secondo l'esperienza non portano valore.
    c. E' stata effettuata una riformattazione delle features uguali creando un'unica feature che indica il numero totale di occorrenze. Esempio: in un sample potremmo avere 3

commenti del task con i valori testuali; invece di lasciare i valori testuali, è stata creata una feature che indica il numero di commenti del task, in questo caso 3.

   d. Ridenominazione di alcune features in modo da renderle leggibili.
   e. Categorizzazione di alcune features. Esempio: in un sample invece di mantenere la feature "Type" testuale (con valori del tipo "Story", "Bug", "Defect"), sono stati trasformati in numeri interi (in questo caso la trasformazione sarebbe: "Story" → 1, "Bug" → 2, "Defect" → 3). Altre trasformazioni sono da tipo testuale a tipo numerico o tipo datetime.
   f. Eliminazione dei samples che hanno i valori di verità (la stima, lo stato del completamento del task) nullo
   g. Normalizzazione dei dati secondo il metodo zscore
2. Creazione di un modello di regressione lineare multivariato che prende in input le features dei sample e in output il valore di verità, in questo caso la stima del task.
   a. Esecuzione del regressore lineare e calcolo delle loss sul dataset di training (la funzione di loss utilizzata è la MSE [Mean Square Error]). Applicazione fino a convergenza del metodo del gradient descent.
   b. Esecuzione del regressore lineare e calcolo delle loss sul dataset di test, applicazione fino a convergenza del metodo del gradient descent
   c. Valutazione dei risultati e tuning degli iperparametri se necessario. Rappresentazione del grafico epoche vs loss e REC.
3. Creazione di un modello di regressione logistico che prende in input le features dei sample e in output il valore di verità, in questo caso la la classe d'appartenenza che indica se un task è stato completato in tempo o meno.
   a. Esecuzione del regressore logistico e calcolo delle loss sul dataset di training (la funzione di loss utilizzata è la Binary Cross Entropy. Applicazione fino a convergenza del metodo del gradient descent.
   b. Esecuzione del regressore logistico e calcolo delle loss sul dataset di test, applicazione fino a convergenza del metodo del gradient descent
   c. Valutazione dei risultati e tuning degli iperparametri se necessario. Rappresentazione della matrice di confusione, REC, valore di accuracy, precision, recall e F1 score.

L'algoritmo ha tre modalità di esecuzione:

1. Training e testing del modello del regressore lineare
2. Training e testing del modello del regressore logistico
3. Inferenza del modello, che preso in input uno o più sample restituisce i corrispettivi valori di verità

3.2 Codice

Il codice e' stato diviso in 4 sezioni (*per dettagli vedere il notebook ML.ipynb*):

1. Caricamento Data Set, che permette l'upload del dataset
2. Preprocessing dei dati, che effettua trasformazioni di tipo, features selection, eliminazioni di feature e sample nulli, categorizzazione delle features
3. Creazione, training e testing del regressore lineare
4. Creazione, training e testing del regressore logistico
5. Inferenza su dati da predire

4.1 Valutazione

Le misure di valutazioni dell'algoritmo sono divise in due blocchi:

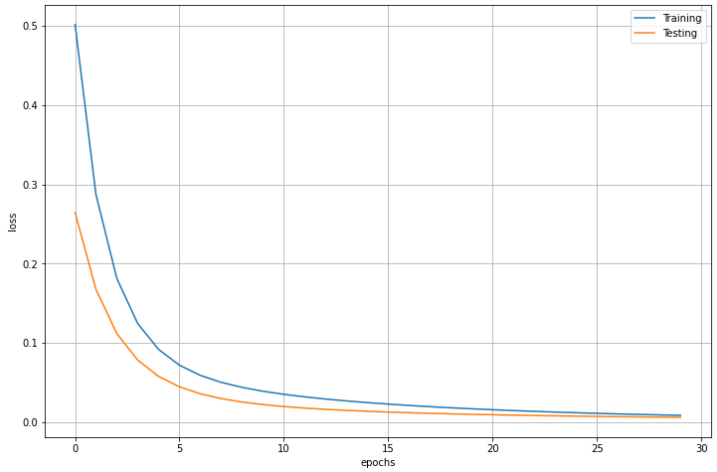
1. Valutazione del regressore lineare
    a. Rappresentazione e confronto del grafico epoche vs loss tra il dataset di training e di test
    b. Calcolo del MSE, MAE, RMSE di training e di test
    c. Plot del grafico REC di training e di test
2. Valutazione del regressore logistico
    a. Calcolo della Confusion Matrix
    b. Calcolo dei valori di accuracy, precision, recall, F1 score di training e di test

5.1 Esperimenti

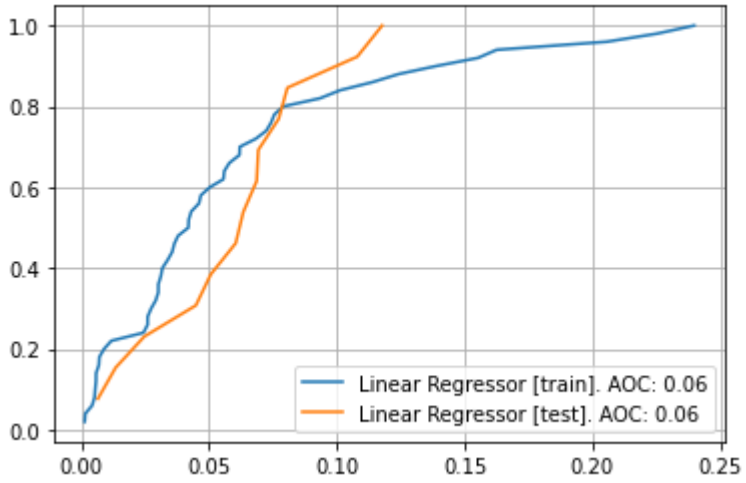
Gli esperimenti effettuati si basano sul training del modello e sulla valutazione del modello tramite i dataset di test.
Dopo aver caricato il dataset ed avere effettuato il pre processing dei dati è stato fatto il training del modello del regressore lineare utile a stimare i valori della stima delle attività. In dettaglio sono state effettuate 3 prove, cambiando il learning rate e le epoche poiché sin da subito il modello approssima bene i valori di verità del dataset di training e di test. Di seguito la valutazione sui 3 esperimenti:

Linear Regression Model	Learning Rate	Epoche	MSE	MAE	RMSE
Esperimento 1	0.01	60	Training: 0.07 Testing: 0.08	Training: 0.27 Testing: 0.30	Training: 0.22 Testing: 0.29
Esperimento 2	0.05	40	Training: 0.01 Testing: 0.02	Training: 0.11 Testing: 0.15	Training: 0.07 Testing: 0.07
Esperimento 3	0.1	30	Training: 0.01 Testing: 0.01	Training: 0.09 Testing: 0.08	Training: 0.08 Testing: 0.13

Di questo l'esperimento con risultati migliori è stato il numero 3.
Di seguito è rappresentato il grafico che confronta le epoche vs loss dell'esperimento num 3 dove si evince che dopo quasi 30 epoche la loss tra training e test è simile.



Di seguito è rappresentato il grafico REC dell'esperimento num 3 dove si evince che l'area sottesa sotto la curva di training e di test sono quasi uguali e quindi si può concludere che il modello generalizza bene



Successivamente è stato eseguito il training del modello del regressore logistico utile a stimare il valore booleano che indica se un'attività è stata completata in tempo o no. In dettaglio sono state effettuate 3 prove, cambiando il learning rate e le epoche. Di seguito la valutazione la matrice di confusione sul risultato migliore (learning rate: 0.1, epoche: 30) con i rispettivi valori di accuracy, precision, recall e F1 score:

Confusion Matrix	Predicted Positive	Predicted Negative
True	100% (TP)	0% (TN)
False	0% (FP)	100% (FN)

Accuracy di training: 0.46
Accuracy di test: 0.38

Precision di training: 0.46
Precision di test: 0.38

Recall di training: 0.46
Recall di test: 0.38

F1 di training: 0.46
F1 di test: 0.38

Come si nota dalla matrice di confusione ma anche dagli altri parametri di valutazione, il classificatore non riesce a classificare le etichette negative. Il risultato ottenuto dopo la computazione è la classificazione di tutte le classi predette in modo positivo (tutte le etichette predette sono TRUE). Il modello di regressione logistica quindi non riesce ad approssimare il pattern dei dati, nonostante il cambiamento di vari parametri.

Dall'inferenza quindi i risultati previsti sono quello di avere una buona approssimazione delle stime dei tasks (tramite la regressione logistica) e una classificazione sempre positiva (errata) se il task verrà completato in tempo. Di seguito il risultato di una delle inferenze effettuate dove si può notare la differenza tra i valori di verità originali e quelli predetti:

Issue key	Original Delivery Date	Delivery Date Predicted	Original Status	Status Predicted
KKK-25888	30	36	0	1
KKK-25889	17	23	0	1
KKK-25160	18	24	1	1

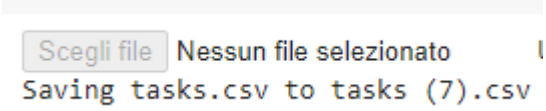
5.2 Demo

La demo è stata effettuata tramite Google Colaboratory.
Tramite la prima sezione si può caricare il dataset e settare i valori dei parametri:

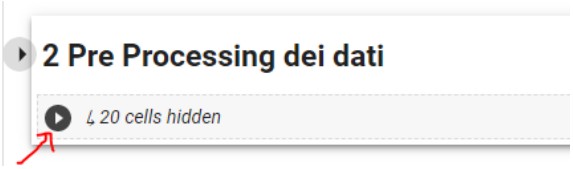
- Regressione Lineare: learning rate, epoche, modalità (training o test).
- Regressione Logistica: learning rate, epoche, modalità (training o test).
- Inferenza: osservazioni da inferire.

Seguire la seguente procedura per effettuare la demo:

1) Aprire il file ML.ipynb in Google Colaboratory
2) Runnare il codice del paragrafo *1.1 Caricamento Data Set*, cliccare il bottone upload e caricare il dataset tasks.csv



3) Runnare l'intero codice del paragrafo *2 Pre Processing dei dati* come mostrato di seguito:



4) Settare i parametri (learning rate e epoche) nel codice della sezione *1.2 Training e Testing del regressore lineare*:

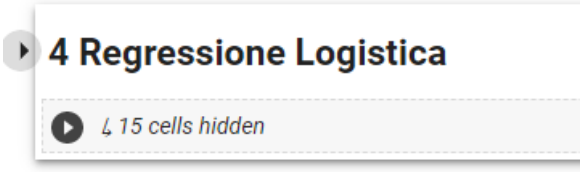
```
[ ]  lr_linear = 0.1
     epochs_linear = 30
     mode_train_linear = True #modalità test (True = attiva)
     mode_test_linear = True #modalità test (True = attiva)
```

5) Runnare il training/testing del modello di regressione lineare tramite la sezione *3 Regressione Lineare* (per vedere i risultati espandere la sezione):

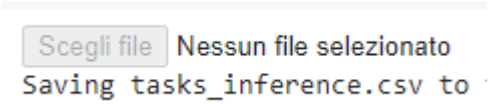
6) Settare i parametri (learning rate e epoche) nel codice della sezione *1.3 Training e Testing del regressore logistico*:

```
[ ]  lr_logistic = 0.01
     epochs_logistic = 50
     mode_train_logistic = True #modalità test (True = attiva)
     mode_test_logistic = True #modalità test (True = attiva)
```

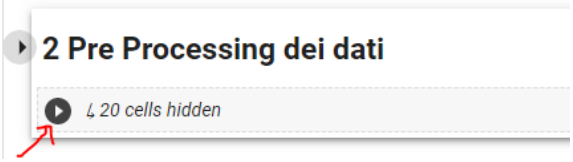
7) Runnare il training/testing del modello di regressione lineare tramite la sezione *4 Regressione Logistica* (per vedere i risultati espandere la sezione):



8) Per effettuare l'inferenza dei dati runnare il codice del paragrafo *1.4 Caricamento Data Set Inferenza*, cliccare il bottone upload e caricare il dataset tasks_inference.csv:



9) Runnare l'intero codice del paragrafo *2 Pre Processing dei dati* come mostrato di seguito:



10) Runnare l'intero codice del paragrafo *5 Inferenza dei dati* in modo da generare la tabella dei risultati

In allegato a questa relazione è disponibile un video dimostrativo *demo.mp4* esplicativo di tutti i passi da seguire.

6 Conclusioni

L'applicazione del Machine Learning nell'ambito del project management non è ancora molto adottato soprattutto perché molte delle attività di gestione possono (ad ora) solo essere svolte dall'intelligenza critica del project manager.
Il metodo presentato in questa relazione però può aiutare in alcune attività fondamentali della gestione del progetto, come la stima delle attività. Infatti grazie al metodo proposto si riesce a prevedere l'effort necessario per il completamento del progetto e delle sue relative attività. In questo caso la regressione lineare ha approssimato in modo ottimale il pattern dei dati.
Nel metodo presentato però non si è riusciti a trovare un modello di regressione logistica che permetta la classificazione delle attività che sono in ritardo o meno. Un improvement che si puo effettuare è applicare un altro metodo di classificazione in modo da capire se si può risolvere il problema.
Il metodo si può estendere ad altre attività del project management come il controllo della velocity del team che lavora sul progetto, in modo da prevedere se alcune attività potrebbero subire ritardo, e su altri ambiti del project management.