PROGETTO ARTIFICIAL INTELLIGENCE E MACHINE LEARNING

Riccardo Fidanza

1-Introduzione

Per lo sviluppo del progetto ho immaginato di essere un'azienda che ha bisogno di assumere dei dipendenti, con diverse posizioni disponibili tante quanti sono i candidati, ogni posizione disponibile ha un suo punteggio di visibilità normalizzato, le posizioni sono ordinate in ordine decrescente, (la prima posizione ha il punteggio di visibilità più elevato e l'ultima ha il punteggio più basso), anche i candidati hanno un punteggio di skill normalizzato e sono disposti in ordine decrescente in base alla skill (il primo candidato ha il punteggio di skill più elevato e l'ultimo ha la il punteggio di skill più basso), attraverso questi due punteggi sono state calcolate la welfare e la fairness di ogni combinazione.

Ogni combinazione è stata poi confrontata con tutte le altre combinazioni mediante i valori di welfare e fairness, ed è stato assegnato un punteggio di score calcolando quante volte una combinazione sia migliore delle altre (welfare più alta e fairness più bassa).

L'obiettivo è quello di determinare quali sono le combinazioni di candidati e posizione che ci restituiscono uno score più elevato, permettendo all'azienda di prendere una decisione valida senza dover valutare tutte le possibili combinazioni.

Per raggiungere l'obiettivo è stato creato un modello di regressione lineare che ha permesso di analizzare come un candidato in una determinata posizione potesse influire sul valore di score della combinazione, permettendoci di scegliere la miglior soluzione.

2-Dataset

I dataset da me utilizzati sono 2, sono stati generati partendo da candidati con il relativo valore di skill e posizioni con il valore della visibilità, generando combinazioni diverse, ogni dataset è composto da 1000 istanze diverse (1000 possibili combinazioni di candidati), e 13 colonne dove le prime 10 rappresentano le diverse posizioni, 2 colonne rappresentano i valori di welfare e fairness per la permutazione e l'ultima colonna rappresenta il valore score.

3-Librerie Utilizzate

Librerie utilizzate in **Python**:

- Pandas: utilizzata per la gestione dei dati, per il salvataggio e per la conversione delle variabili in variabili dummy
- Math: operazioni di calcolo
- Matplotlib: creazione di grafici
- Statsmodel: per modello di regressione lineare, report della regressione
- Numpy: array

Librerie utilizzate in **R**:

- FastDummies: per la conversione delle variabili in variabili dummy
- Gamlss: per l'implementazione della regressione lineare

4-Metodi Utilizzati

Creazione dei dati

Il primo passo per lo sviluppo del progetto è stata la generazione dei valori di skill e di visibilità entrambi normalizzati (la somma totale è pari ad 1), in Python con la libreria random e in R con la funzione sample (implementata nel pacchetto base di R), questi due punteggi sono stati assegnati in ordine decrescente, le skill ai candidati e la visibilità alla posizione.

Per la creazione dei dati sono stati selezionati 10 candidati rinominati con il nome candidato(n) (candidato1, …candidato10), a ciascuno di essi è stato assegnato un punteggio di skill e sono stati gestiti con una lista di tuple (candidato(n), skill).

Attraverso shuffle di sklearn in Python e replicate di R sono state generate 1000 permutazioni per i candidati senza ripetizioni.

Per ciascuna permutazione sono stati calcolati i punteggi di welfare e fairness, il welfare è stato calcolato moltiplicando la skill di ciascun candidato per la visibilità della posizione in cui si trova e sommando su tutti i candidati, la fairness è stata calcolata come una divergenza di Kullback-Leibler (è una misura non simmetrica della differenza tra due distribuzioni di probabilità P e Q.) moltiplicando la skill di ciascun candidato per il logaritmo in base 2 della skill del candidato diviso la visibilità della posizione in cui si trova sommando su tutti i candidati. Per ogni permutazione più il valore di fairness sarà basso più la permutazione sarà fair.

A questo punto ogni permutazione è stata confrontata con tutte le altre, ogni volta che il valore di welfare è più alto e il valore di fairness è più basso rispetto ad un'altra permutazione viene aggiunto 1 ad un contatore, alla fine del confronto di ogni permutazione il contatore viene diviso per il totale delle permutazioni meno la permutazione che viene confrontata con le altre, in questo caso 999, e viene aggiunto questo valore come valore di score per la permutazione, lo score ci indica quanto una permutazione sia migliore delle altre.

Il dataset alla fine delle operazioni è composto da 1000 istanze, e 13 colonne contenenti le posizioni, i valori di welfare fairness e score. È stato poi salvato per evitare di dover ripetere le operazioni di creazione.

	0	1	2	3	4	5	6	7	8	9	welfare	fairness	score
0	candidato7	candidato9	candidato5	candidato2	candidato1	candidato6	candidato3	candidato10	candidato4	candidato8	0.098590	0.570590	0.420420
1	candidato9	candidato6	candidato10	candidato2	candidato1	candidato5	candidato3	candidato4	candidato8	candidato7	0.085115	0.796815	0.098098
2	candidato7	candidato6	candidato4	candidato1	candidato8	candidato3	candidato5	candidato9	candidato2	candidato10	0.109742	0.411800	0.774775
3	candidato5	candidato9	candidato4	candidato8	candidato6	candidato7	candidato3	candidato10	candidato1	candidato2	0.096402	0.745237	0.290290
4	candidato4	candidato2	candidato1	candidato8	candidato5	candidato6	candidato9	candidato3	candidato10	candidato7	0.119515	0.353200	0.943944
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	candidato9	candidato4	candidato8	candidato5	candidato7	candidato10	candidato6	candidato3	candidato1	candidato2	0.083772	0.979708	0.040040
996	candidato6	candidato9	candidato10	candidato1	candidato3	candidato2	candidato4	candidato5	candidato8	candidato7	0.087315	0.770558	0.137137
997	candidato2	candidato5	candidato10	candidato4	candidato3	candidato6	candidato9	candidato7	candidato1	candidato8	0.105138	0.555107	0.623624
998	candidato7	candidato10	candidato1	candidato3	candidato2	candidato8	candidato6	candidato5	candidato4	candidato9	0.095433	0.663819	0.313313
999	candidato10	candidato5	candidato4	candidato1	candidato8	candidato3	candidato6	candidato9	candidato7	candidato2	0.097418	0.659587	0.369369

1000 rows × 13 columns

Sviluppo

Una volta caricati i dati, le colonne contenenti il welfare e la fairness sono state rimosse poiché non sono significative per il nostro studio, quindi il dataset sarà composto ora da 1000 istanze e 11 colonne.

Come primo passo le variabili (i candidati) che sono di tipo qualitativo sono state trasformate in variabili dummy, cioè variabili fittizie che permettono di lavorare con valori numerici anche quando la variabile di partenza è di tipo qualitativo, codificando con 0 ed 1 la presenza o meno di un determinato candidato all'interno della permutazione, in questo modo ogni posizione potrà assumere 10 diversi valori, il dataset adesso è composto da 101 variabili e 1000 istanze.

	0_candidato1	0_candidato10	0_candidato2	0_candidato3	0_candidato4	0_candidato5	0_candidato6	0_candidato7	0_candidato8	0_candidato9	...	9_can
0	0	0	0	0	0	0	0	1	0	0	...	
1	0	0	0	0	0	0	0	0	0	1	...	
2	0	0	0	0	0	0	0	1	0	0	...	
3	0	0	0	0	0	1	0	0	0	0	...	
4	0	0	0	0	1	0	0	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
995	0	0	0	0	0	0	0	0	0	1	...	
996	0	0	0	0	0	0	1	0	0	0	...	
997	0	0	1	0	0	0	0	0	0	0	...	
998	0	0	0	0	0	0	0	1	0	0	...	
999	0	1	0	0	0	0	0	0	0	0	...	

1000 rows × 101 columns

A questo punto attraverso Statsmodel in Python e Gamlss in R è stata creata una regressione lineare con il metodo dei minimi quadrati (una tecnica di ottimizzazione che permette di trovare una funzione rappresentata da una curva che si avvicini il più possibile ai dati) utilizzando come variabile dipendente lo score e come variabili indipendenti le posizioni dei candidati.

Il passo successivo è stato quello di analizzare i risultati della regressione, e verificare come la posizione specifica di un determinato candidato influisse sullo score finale.

5-Risultati

Python

Con la funzione summary di statsmodel sono stati stampati i risultati

OLS Regression Results

Dep. Variable:	score	R-squared:	0.969
Model:	OLS	Adj. R-squared:	0.966
Method:	Least Squares	F-statistic:	349.0
Date:	Tue, 01 Feb 2022	Prob (F-statistic):	0.00
Time:	11:08:54	Log-Likelihood:	1561.1
No. Observations:	1000	AIC:	-2958.
Df Residuals:	918	BIC:	-2556.
Df Model:	81		
Covariance Type:	nonrobust		

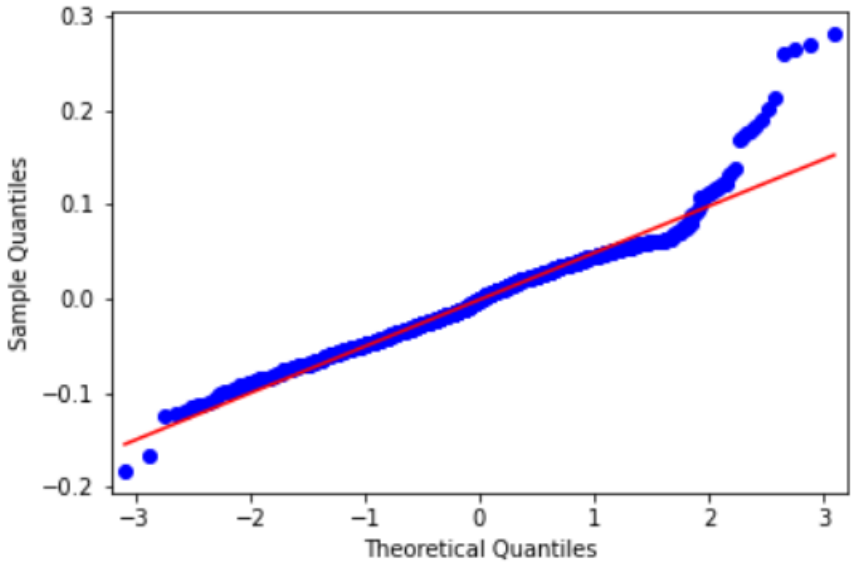
Dep.variable: mostra quale è la variabile dipendente

Model: è il modello di regressione utilizzato, OLS (ordinary least squares)

Method: il metodo implementato nella regressione, appunto quello dei minimi quadrati

R-squares: l'R quadro può assumere valori che vanno da 0 ad 1, in questo caso ha un valore di 0.969, ci indica che il modello spiega il 96.9% della varianza della variabile dipendente, ciò implica che il fenomeno ha forte capacità di rimanere concentrato attorno alla sua retta di regressione, minimizzando gli scostamenti da questa.

È stata fatta un'analisi dei residui attraverso un q-q plot per vedere se i residui del nostro modello si distribuissero come i residui di una normale.



I nostri residui si dispongono all'incirca come i residui di una normale, tranne che per le code cove essi si discostano, questo poiché la nostra distribuzione presenta un'asimmetria positiva che possiamo vedere nel coefficiente di skewness e nella media che è maggiore della mediana.

```
minimo -0.18262564530376402
primo quartile: -0.03442298090754536
mediana: -0.0007155884098664078
media: 1.6547874182037958e-16
terzo quartile: 0.03258481299727567
massimo: 0.2810845685913222
Skew:                          0.835
```

Dopo aver analizzato il modello, è stato studiato come la posizione di un candidato influenzasse il risultato dello score.

Per il candidato 1, che ha un valore di skill più elevato le posizioni migliori sono le prime quindi quelle che hanno visibilità più elevata.

	coef	std err	t	P>|t|	[0.025	0.975]
0_candidato1	0.1785	0.005	39.045	0.000	0.169	0.187
1_candidato1	0.1614	0.005	35.206	0.000	0.152	0.170
3_candidato1	0.0719	0.005	14.983	0.000	0.062	0.081

5 candidato1	0.0356	0.005	7.615	0.000	0.026	0.045
7_candidato1	-0.0514	0.005	-11.270	0.000	-0.060	-0.042
9_candidato1	-0.0834	0.005	-15.587	0.000	-0.094	-0.073

Come primo studio andiamo ad analizzare il P-value (più è basso più indica che il nostro test ha una significatività statistica), per valori inferiori allo 0.05 possiamo dire che la variabile nel nostro modello è statisticamente significativa, in questo caso tutti i P-Value sono minori di 0.05.

Attraverso il coefficiente che indica la correlazione tra la variabile indipendente e la variabile dipendente (se è positivo c'è correlazione positiva, se è negativo c'è correlazione negativa) possiamo vedere come all'aumentare della presenza del candidato 1 nelle prime 5 posizioni lo score della combinazione sia influenzato positivamente, ma partendo dalla 6° questa influenza diventa negativa, da questo possiamo dedurre che i candidati con un alto valore di skill sono più adatti a ricoprire posizioni con alto valore di visibilità.

Questi valori sono confermati per i primi 5 candidati, cioè quelli con valore di skill più elevato.

Se andiamo ad analizzare il sesto candidato sempre attraverso il coefficiente ci accorgiamo che anche esso ha una maggiore influenza positiva nelle prime posizioni ma anche quando è presente in posizioni con minor valore di visibilità non influenza negativamente lo score della combinazione.

	coef	std err	t	P>|t|	[0.025	0.975]
0_candidato6	0.0762	0.005	16.842	0.000	0.067	0.085
1 candidato6	0.0701	0.004	15.789	0.000	0.061	0.079
3_candidato6	0.0585	0.005	11.379	0.000	0.048	0.069
5 candidato6	0.0400	0.004	8.960	0.000	0.031	0.049
7_candidato6	0.0192	0.005	3.959	0.000	0.010	0.029
9_candidato6	0.0159	0.005	3.204	0.001	0.006	0.026

Anche in questo caso i P value sono inferiori allo 0.05 questo ci indica che le nostre variabili sono statisticamente significative.

È subito evidente come all'aumentare della presenza del candidato 6 che ha valori di skill che potremmo definire 'medi' tra quelli presenti, l'influenza positiva sullo score della permutazione non sia così elevata come quella del candidato 1, la forte differenza è che anche nelle posizioni con un valore di visibilità più basso il candidato 6 non influenza negativamente lo score della combinazione a differenza dei candidati con un elevato valore di skill. Questo discorso è valido solamente per i candidati 6 e 7.

Se andiamo ad analizzare il candidato 10 possiamo vedere come il suo comportamento è opposto da quello del candidato 1.

	coef	std err	t	P>|t|	[0.025	0.975]
0_candidato10	-0.1870	0.005	-40.603	0.000	-0.196	-0.178
1_candidato10	-0.1451	0.005	-29.934	0.000	-0.155	-0.136
3_candidato10	-0.0107	0.004	-2.380	0.018	-0.019	-0.002

5_candidato10	0.0596	0.004	13.559	0.000	0.051	0.068
7 candidato10	0.2137	0.005	43.927	0.000	0.204	0.223
9_candidato10	0.2563	0.005	56.457	0.000	0.247	0.265

I P-Value sono minori dello 0.05

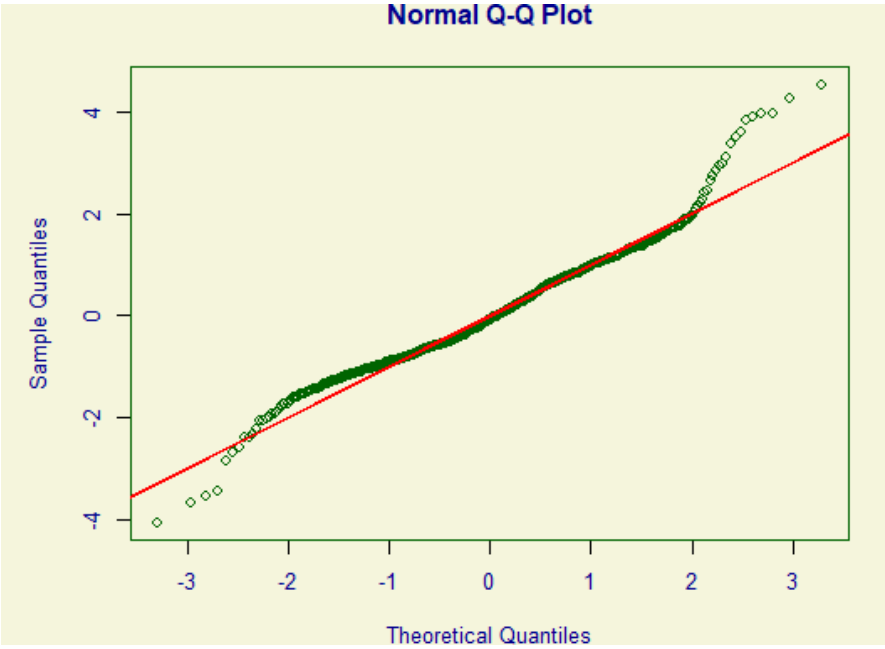
Guardando i coefficienti notiamo come l'aumento della presenza del candidato 10 nelle prime posizioni (quelle con un punteggio di visibilità più elevato) abbia un'influenza negativa sullo score della combinazione, mentre l'aumentare della sua presenza nelle posizioni con punteggio di visibilità più basso influisca positivamente sullo score. Questo discorso è valido per i candidati 8 ,9 e 10, quelli con un punteggio di skill più basso.

R

Con summary in R sono stati stampati i risultati della regressione lineare con Gamlss

Anche in questo caso è stata applicata una regressione lineare con il metodo dei minimi quadrati, l'R quadro è di 0.979 e quindi il nostro modello spiega il 97.9% della varianza `[1] 0.979508`

L'analisi dei residui anche qui è stata fatta attraverso un q-q plot



Analizzando il q-q plot i residui si dispongono sulla normale tranne che per le code, anche in questo caso attraverso il coefficiente di skewness e la media che è maggiore della mediana possiamo dire di avere una distribuzione con asimmetria positiva

```
coef. of skewness  =  0.4495911
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-4.0599 -0.6824 -0.0815  0.0000  0.6847  4.5268
```

Dopo aver analizzato il modello, è stato analizzato come la posizione di un candidato influenzasse il risultato dello score.

Per il candidato 2, che ha un valore di skill più elevato le posizioni migliori sono le prime quindi quelle che hanno visibilità più elevata, a conferma dello studio precedentemente effettuato in Python.

	Estimate	Std. Error	t value	Pr(>|t|)	
V1 candidato2	0.715731	0.009450	75.738	< 2e-16	***
V3_candidato2	0.641479	0.009168	69.966	< 2e-16	***
V5_candidato2	0.615946	0.008863	69.494	< 2e-16	***
V7_candidato2	0.324346	0.008969	36.164	< 2e-16	***
V9_candidato2	0.034641	0.008905	3.890	0.000107	***

Il primo valore che andiamo a guardare è il P-Value, ed essendo minore di 0.05 possiamo dire che le nostre variabili sono statisticamente significative.

In R per comodità vicino al P-Value sono rappresentati dei simboli che ci aiutano a capire quanto una variabile sia significativa.

Possiamo notare attraverso i valori stimati dei coefficienti (estimate) come anche in questo caso i candidati con un valore di skill più elevato abbiano un'influenza positiva sullo score maggiore se si trovano nelle prime posizioni, ma a differenza della regressione sviluppata con Python, se questa categoria di candidati si trova nelle posizioni con valore di visibilità più basso, influenza lo score comunque in maniera positiva.

Il passo successivo è stato quello di analizzare il candidato 6 che possiamo dire avere un valore di skill 'medio' tra i valori di skill dei nostri candidati.

	Estimate	Std. Error	t value	Pr(>|t|)	
V1_candidato6	0.383630	0.009387	40.870	< 2e-16	***
V3_candidato6	0.347476	0.008758	39.674	< 2e-16	***
V5_candidato6	0.317536	0.009238	34.373	< 2e-16	***
V7_candidato6	0.170268	0.008890	19.153	< 2e-16	***
V9_candidato6	0.022400	0.009159	2.446	0.014648	*

La prima cosa che possiamo notare è che il P-Value della posizione 9 non è significativo come quello delle altre posizioni per il candidato 6, anche nella posizione 1 il candidato 6 non influenza eccessivamente lo score della combinazione, mantenendo circa lo stesso livello di influenza anche per le altre posizioni.

Possiamo quindi dire che i candidati che hanno un punteggio di skill non troppo alto ma neanche troppo basso possono assumere le diverse posizioni centrali come la posizione 4-5-6 influenzando comunque in modo positivo lo score della combinazione.

Si è poi proceduto nell'analizzare i candidati con un valore di skill basso, come il candidato 9, cercando di vedere come lo score fosse influenzato dalla posizione di questo candidato nella combinazione.

	Estimate	Std. Error	t value	Pr(>|t|)	
V1_candidato9	0.118728	0.009453	12.560	< 2e-16	***
V3_candidato9	0.106716	0.008758	12.185	< 2e-16	***

```
V5_candidato9  0.103985   0.009298  11.183  < 2e-16 ***
V7_candidato9  0.052774   0.009147   5.769 1.09e-08 ***
V9_candidato9  0.008487   0.009031   0.940 0.347601
```

Il P-Value in tutte le posizioni escluse le ultime 2 posizione (8-9) è significativo.

Come possiamo vedere attraverso la stima dei coefficienti l'influenza del candidato 9 anche nelle prime posizioni è bassa rispetto a quella di altri candidati con valori di skill superiori, e a partire già dalla posizione 7 possiamo dire che sia quasi nulla.

Con questo si conferma quanto già notato in precedenza, i candidati con punteggio di skill basso non influenzano significativamente una combinazione in qualsiasi posizione essi si trovino.

6-Conclusioni

Attraverso la regressione lineare sono state individuate le posizioni migliori da assegnare a ciascun candidato.

Possiamo dire che i nostri dieci candidati possono essere divisi in 3 'gruppi', nel primo gruppo sono presenti i candidati con elevati valori di skill, nel secondo quelli con valori di skill 'medi' e nel terzo quelli con valori bassi.

Per un'azienda che deve assumere dei candidati le migliori combinazioni sono sicuramente quelle che hanno nelle prime posizioni un candidato con un valore di skill elevato. Per quanto riguarda le posizioni con una visibilità non eccessivamente alta possiamo dire che i candidati con valori di skill 'medi' sono i più adatti, poiché se fossero assegnati alle posizioni con alta visibilità non influenzerebbero in modo eccessivo lo score della combinazione a differenza dei candidati con valori di skill elevati. Per i candidati con valori di skill bassi le posizioni migliori sono quelle dove anche il valore della visibilità è basso, evitando così di influenzare negativamente lo score della combinazione.

Le combinazioni che sono sicuramente da evitare sono quelle in cui i candidati con basso valore di skill si trovano nelle posizioni con alto valore di visibilità, poiché oltre a non influenzare lo score eccessivamente, c'è anche il rischio che possano influenzare lo score in modo negativo. Lo stesso discorso lo si può fare per i candidati con skill più elevata che se assegnati a posizioni con visibilità bassa possono influenzare negativamente lo score.

Attraverso questo studio, si è messo in evidenza come candidati con valori di skill 'medi' possano essere assegnati a quasi tutte le posizioni, candidati con valori di skill bassi non siano adatti a ricoprire posizioni ad alta visibilità, e che lo score della combinazione è maggiormente influenzato dai candidati con punteggio di skill elevato.