
Gabriele Marconi

Matricola: 0000999689

Realizzazione di valutatore di mazzi KeyForge tramite tecniche di Machine Learning

ABSTRACT

Lo scopo del progetto è la realizzazione di un sistema che dato un **mazzo** (o deck) del gioco di carte [KeyForge](#) realizzi la valutazione automatica del mazzo.

Attualmente, esistono diversi siti che realizzano la valutazione algoritmica della qualità dei mazzi tramite alcuni punteggi, tra i più popolari ci sono [SAS](#) e [AERC](#).

Gli algoritmi esistenti si basano su una serie di regole, scopo del progetto è sperimentare diverse tecniche di machine learning per effettuare questo procedimento in modo automatico.

E' possibile sfruttare dataset già esistenti per analisi, apprendimento e testing.

CAPITOLI

- I. *Introduzione al tema.*
- II. *Realizzazione dei dataset.*
- III. *Creazione, Training e Testing del modello.*

STRUMENTI UTILIZZATI

Il progetto, sviluppato interamente in Python, ha usufruito di 2 ambienti. La creazione dei dataset è avvenuta in locale con Visual Studio Code ed i metodi di Pandas, mentre la piattaforma di Colab e Tensorflow hanno permesso di gestire il modello di Machine Learning.

Capitolo I - Introduzione

IL GIOCO

KeyForge è un gioco di carte collezionabili che adotta un modello di vendita molto diverso dalla concorrenza. Infatti, è possibile comprare unicamente dei mazzi pre-fatti, e lo scambio di carte è totalmente impossibile.

Questo perché ogni mazzo è generato da un algoritmo che decide quali carte inserire, creando così una combinazione unica nel mondo.

Un mazzo di KeyForge è identificato dal proprio nome (anche questo generato alitmicamente) e dalle tre “casate” (o fazioni) che lo compongono. Una fazione contribuisce con 12 carte al mazzo.


Al lancio il gioco ha pubblicato sette casate, ognuna con la propria lista di carte.

DECKS OF KEYFORGE

decksofkeyforge.com fornisce un’ampia raccolta di deck (caricati dagli utenti stessi) e di statistiche.

Queste ultime, ovvero SAS e AERC, riassumono grazie ad algoritmi costruiti ad hoc la potenza di una carta o mazzo.

In questo progetto, la statistica considerata per l’analisi è il “**SAS rating**”.

Name	Houses	SAS ↓	SASStars	SAS%	Synergy	Antisyn	Raw AERC
"Maga", Spadaccina di Furiarigida		70	★★★ ▲	84.3	5	1	64

Il sito inoltre offre file .csv contenenti le proprietà di tutti i deck presenti nel proprio database fino ad una certa data e di tutte le carte rilasciate fino ad oggi. Questi sono risultati fondamentali per agevolare la realizzazione dei dataset utilizzati.

Capitolo II - Realizzazione dei Dataset

I dataset (di *training* e *testing*) sono organizzati in:

- **righe:** ogni riga rappresenta un mazzo.
- **colonne:** ogni colonna rappresenta una proprietà (o feature) relativa ad un mazzo.

Per facilitare lo sviluppo e contenere il numero di esempi da inserire nel dataset (diminuendo quindi i tempi di esecuzione), sono stati considerati solamente i deck usciti nella prima espansione “*Call of the Archons*”.

COMPOSIZIONE DEI FILE DI DATI

A questo punto è necessario distinguere tra i due file che descrivono i mazzi e le carte:

- **AllDecks_CoA.csv** contiene tutti i deck della prima espansione e le loro proprietà:
 - **sas_rating:** il punteggio SAS del mazzo calcolato dall’algoritmo.
 - **houses_names:** i nomi delle tre casate.
 - **card_names:** la lista dei nomi delle carte che compongono il mazzo, delimitati da un carattere speciale.
- **AllCards_CoA.csv** contiene tutte le carte pubblicate nella prima espansione e le loro proprietà, che saranno approfondite nel paragrafo sulla scelta delle feature.

SMISTAMENTO DEI MAZZI

La prima fase della costruzione dei dataset consiste nel dividere AllDecks_CoA.csv in dieci file .csv numerati, ognuno contenente i mazzi che appartengono ad una certa categoria basata sul punteggio SAS. Ciò è stato fatto per poter controllare più efficientemente la composizione dei dataset stessi.

Nello specifico, una decisione importante durante lo sviluppo ha riguardato il metodo di smistamento dei mazzi. Ogni categoria, infatti, è legata ad un certo intervallo dei valori SAS.

Ad esempio, un mazzo con punteggio minore di 40 potrebbe appartenere al primo file, ovvero alla prima categoria; mentre un mazzo con punteggio nell'intervallo 41-50 potrebbe appartenere al secondo file.

Si sono dunque presentate due strategie per la definizione degli intervalli:

- **Percentile:** segue intervalli definiti da DoK in una statistica chiamata SASStars che considera la percentuale dei deck caricati con quel punteggio e li giudica con un valore da 1 a 10.
- **Costante:** dopo aver trovato il punteggio massimo e minimo, si definiscono intervalli equidistanti.

Vari tentativi di testing hanno mostrato come la strategia **costante** fosse la più efficace.

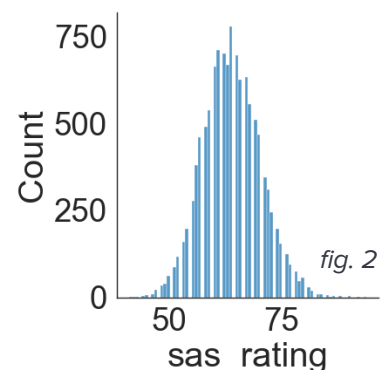
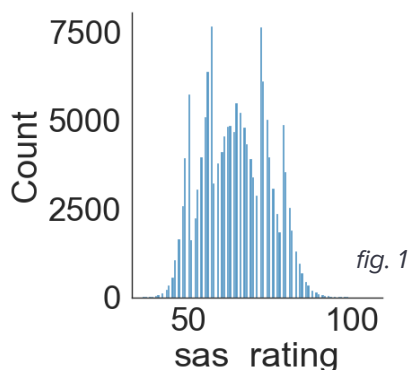
DISTRIBUZIONE DEI DATASET

Che logica seguire, ora, per scegliere quanti mazzi di ogni categoria devono essere inseriti nei dataset finali ?

Anche la seconda fase presenta una decisione fondamentale sulla logica da utilizzare:

- **Frazionaria:** viene raccolta una percentuale di mazzi per ogni categoria, rappresentando fedelmente la reale distribuzione di punteggi. Questa logica comporterebbe una maggior precisione nelle categorie più popolate, a scapito delle restanti.
- **Equilibrata:** si cerca di raccogliere lo stesso numero di mazzi da ogni categoria. Questa logica, teoricamente, porterebbe a maggior costanza nell'accuratezza delle previsioni ma con picchi minori rispetto all'alternativa.

L'analisi dei risultati ottenuti dopo vari tentativi ha portato a scegliere la logica **equilibrata** per il dataset di training (*fig. 1*). Tuttavia, per il dataset di testing è stata scelta la logica **frazionaria** (*fig. 2*) al fine di simulare un ambiente più realistico.



Alla fine di questo processo, otteniamo due file .csv, uno di training e uno di testing, che contengono ancora le informazioni “raw” del file originale.

La prossima fase si occupa di trasformare queste informazioni in features adatte all’input di un modello di Machine Learning.

SCELTA DELLE FEATURES

I dataset completi, ottenuti dopo la terza fase, sono composti da due categorie di feature:

Card features

Il nome di ogni carta è convertito in **22 features** attraverso un'operazione di lookup sul file *AllCards_CoA.csv*.

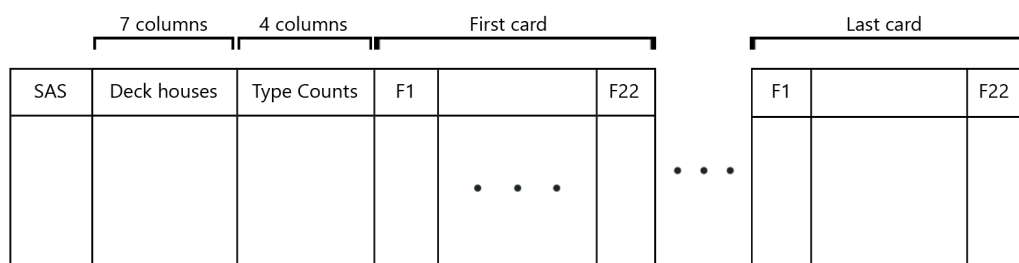
Molte proprietà presenti nel file fanno riferimento agli algoritmi SAS e AERC, sarebbe dunque illogico (e fin troppo semplice) utilizzarle considerato l’obiettivo del progetto. Le 22 features scelte, dunque, cercano di descrivere al meglio le statistiche e potenzialità di ogni carta basandosi su informazioni visibili e non pre-processate.

In tutto abbiamo **792** (36*22) features relative alle carte nel mazzo. Questo numero importante è causato dall’utilizzo del **one-hot encoding** per la maggior parte delle proprietà.

Deck features

Riguardano statistiche utili del mazzo:

- Il rating SAS, che sarà utilizzato come **label**.
- Le tre casate del mazzo in **one-hot encoding**.
- Il numero di ogni **tipo** di carta nel mazzo. (ad esempio: 22 creature, 5 artefatti...)



Capitolo III - Creazione, Training e Testing del modello

Analizzando il problema, si capisce chiaramente come il modello debba usare la **Regressione** per ottenere un valore unico (il punteggio SAS) partendo da un certo input.

Esistono vari tipi di modelli in grado di implementare la regressione; in questo caso, dopo una prova con strategia **Random Forest** (ricerca basata su alberi decisionali) che non ha fornito risultati particolarmente soddisfacenti, la scelta è ricaduta su una DNN costruita grazie agli strumenti forniti da tensorflow.

STRUTTURA DELLA RETE NEURALE

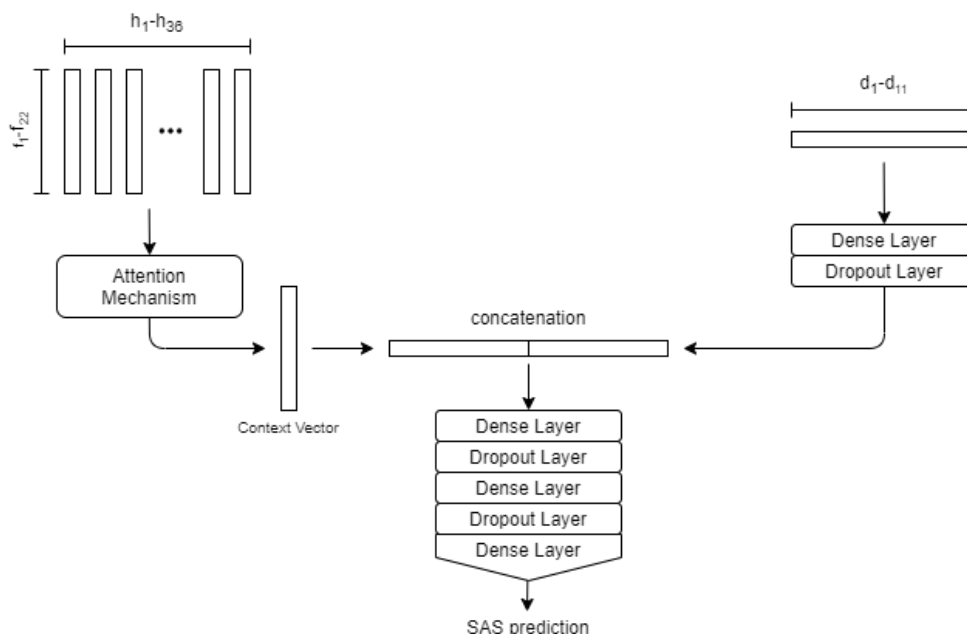
Il modello si basa principalmente sull'utilizzo di livelli **Dense** e **Dropout**. Tuttavia, la natura sequenziale e ripetitiva del dataset nella descrizione delle 36 carte, ha suggerito l'implementazione del meccanismo di **Attention** (più precisamente, Additive Attention).

Con questa strategia è stato possibile “riassumere” le informazioni contenute nella porzione di dataset relativa alle carte e rendere l'esecuzione del modello più efficiente ed efficace.

L'output del meccanismo è un **vettore contesto** che va concatenato con il resto del dataset.

Il modello ha quindi due **input** (considerando un solo mazzo):

- Una matrice 36x22 sotto forma di tensore, ottenuta dalla porzione delle *card features* nel dataset.
- Le *deck features* (senza il SAS) come DataFrame Pandas contenente una sola riga.



TRAINING CON HYPEROPT

La scelta di un dataset di training più equilibrato nasce dai risultati ottenuti con l'ottimizzazione degli iperparametri.

Hyperopt è una libreria Python che automatizza la ricerca dei parametri ottimali per gli strati (o layers) di una rete neurale durante la fase di training.

Inizialmente si definisce lo spazio di ricerca per tutti i parametri ottimizzabili. In questo progetto si ha:

- le **units** per gli strati Dense
- il **rate** per gli strati Dropout
- il **learning rate** per Adam optimizer

Successivamente si definisce la metrica da minimizzare: il **Mean Absolute Error** o MAE restituito dall'output della valutazione del modello è la scelta migliore, anche perché rappresenta la **loss** della rete.

```
def hyperopt_fcn(params):  
    model = train_fcn(params)  
    mae = test_fcn(model)  
    return {'loss': mae, 'status': STATUS_OK}
```

Infine, il metodo fmin utilizza l'algoritmo TPE per effettuare una ricerca dei parametri migliori nello spazio definito paragonando sequenzialmente i valori MAE.

```
best = fmin(hyperopt_fcn, search_space, algo=tpe.suggest,...)
```

Poiché il processo di ricerca può richiedere molto tempo, è stato utilizzato l'**Early Stopping** come callback del processo di training per velocizzare l'esecuzione ed evitare l'overfitting.

```
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',  
patience=10, restore_best_weights=True)
```

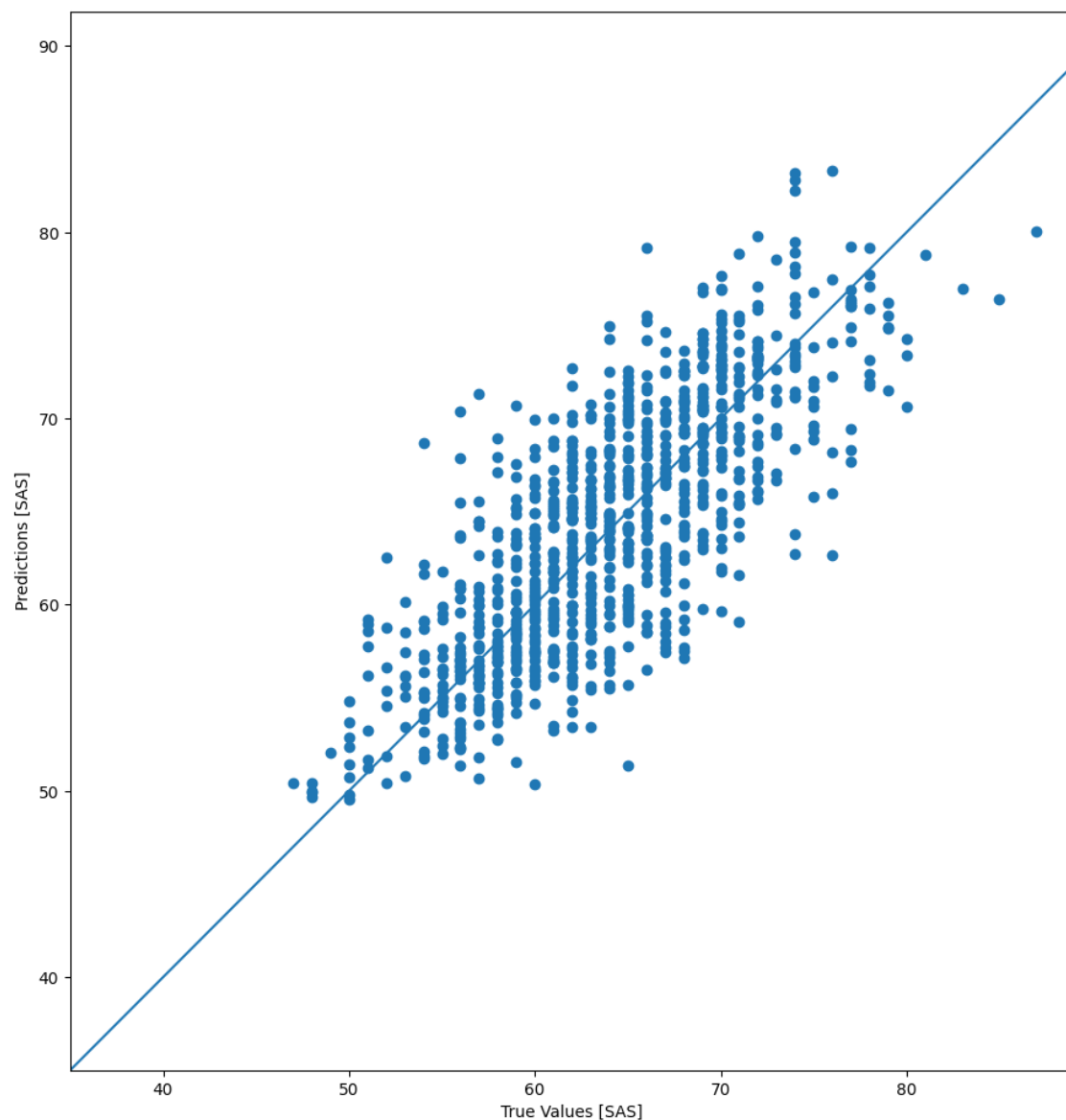
TESTING E RISULTATI

La valutazione del modello finale restituisce un MAE di **3.63 punti SAS**.

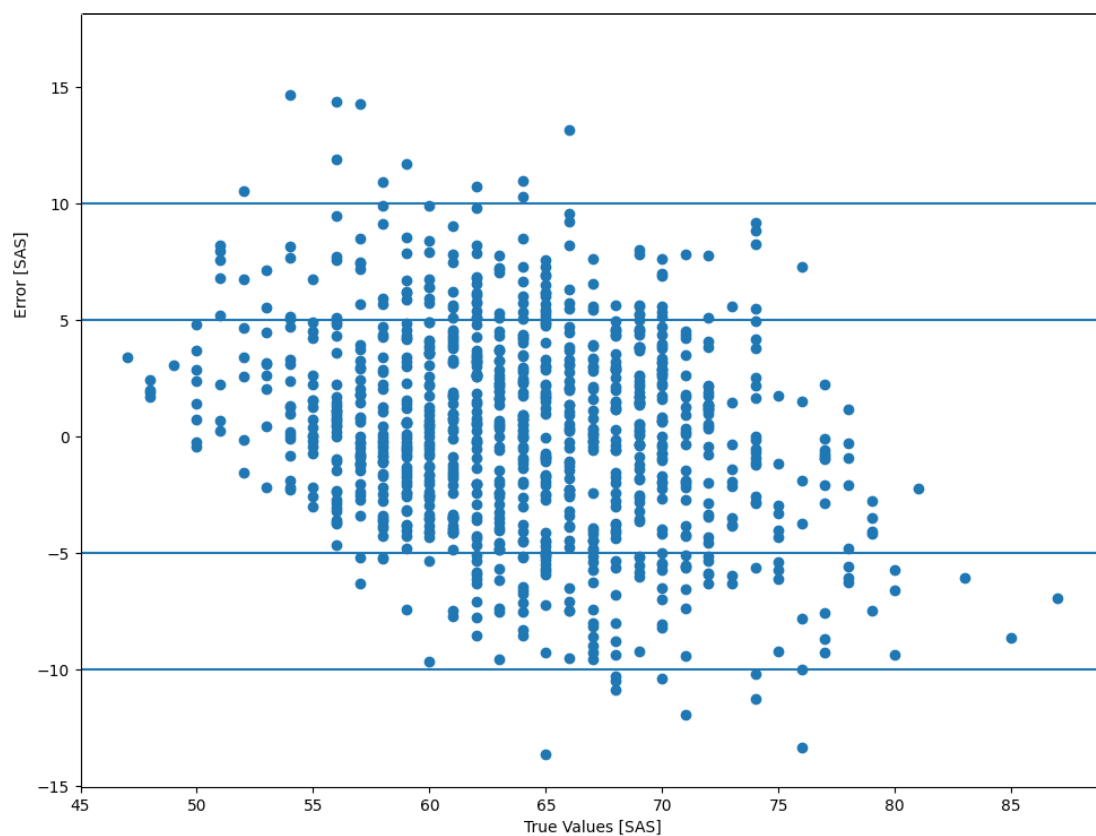
L'accuratezza, invece, è misurata in un intervallo di confidenza $[x-5, x+5]$ dove x è il vero punteggio del mazzo. Il risultato mostra il **77% di accuratezza generale**.

Matplotlib ha permesso di visualizzare queste statistiche in modo più chiaro:

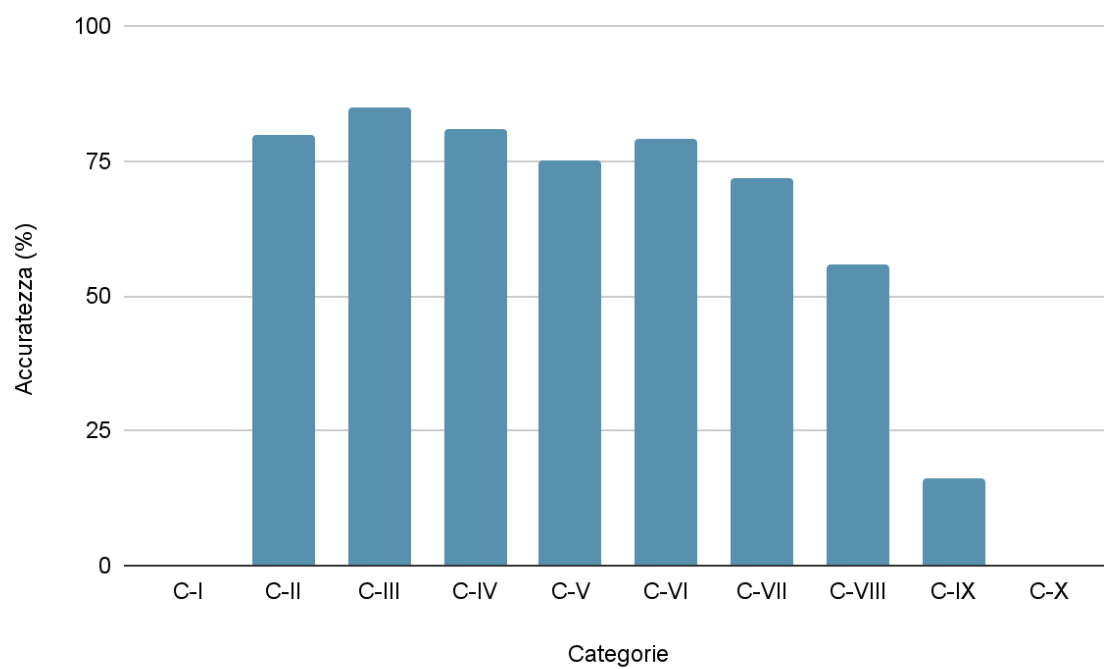
Rapporto tra predizioni e valori veri



Visualizzazione errore e intervalli



Accuratezza tra le categorie



Conclusioni

Grazie all'inclusione di Hyperopt e del meccanismo di Attention, è stato possibile raggiungere un risultato soddisfacente ed equilibrato.

La scalabilità del sistema, inoltre, potrebbe portare all'inclusione delle altre espansioni; questo però causerebbe una grande estensione sia del numero di esempi da inserire nei dataset che delle feature che rappresentano la singola carta, aumentando di molto la complessità del problema.

Bibliografia

Decks of KeyForge per lo studio del problema e per le risorse di dati:

<https://decksofkeyforge.com>

Documentazione Hyperopt:

<http://hyperopt.github.io/hyperopt/>

Un ringraziamento al Dott. Andrea Galassi per la condivisione del suo paper sull'Attention:

<https://ieeexplore.ieee.org/document/9194070>