Università degli Studi di Milano Bicocca
**Scuola di Scienze**
**Dipartimento di Informatica, Sistemistica e Comunicazione**
**Corso di laurea magistrale in Informatica**

# ASSIGNEMENT 1
# Advance Machine Learning

Default Payments of Credit Card Prediction

Andrea Guzzo, 761818

## Descrizione dell'assignement

L'assignement consiste nella predizione di probabilità di default riguardo a dati delle carte di credito di alcuni clienti utilizzando una rete neurale.

Il set di dati contiene informazioni sui pagamenti in default, i fattori demografici, i dati di credito, la cronologia dei pagamenti e gli estratti conto dei clienti delle carte di credito a Taiwan da aprile 2005 a settembre 2005.

L'analisi compiuta è reperibile anche al seguente indirizzo su GitHub:
https://github.com/JeyDi/CreditCard-NeuralNet

## Modello dei dati

Il dataset consegnato è diviso in test e training, all'interno dei files forniti sono presenti 25 variabili che descrivono il problema.
La descrizione delle variabili e il loro utilizzo è riportato all'interno del notebook contenente la soluzione proposta.

## Analisi esplorativa

L'analisi esplorativa per comprendere da un punto di vista funzionale e statistico le variabili e il dataset è divisa in due momenti differenti:

1. Report in PDF: dataMaid_training_dataset.pdf all'interno della cartella consegnata, generato utilizzando Data Maid in R
2. Analisi esplorativa e statistica più approfondita all'interno del Jupyter Notebook

## Descrizione della rete neurale

Per risolvere il problema richiesto è stata costruita una rete neurale utilizzando Keras su un Tensorflow backend.
La rete è stata costruita utilizzando la seguente configurazione:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_41 (Dense)             (None, 10)                240
_________________________________________________________________
dense_42 (Dense)             (None, 10)                110
_________________________________________________________________
dense_43 (Dense)             (None, 2)                 22
=================================================================
Total params: 372
Trainable params: 372
Non-trainable params: 0
_________________________________________________________________
```

È quindi composta da 3 layers densi, due composti da 10 unità utilizzando come funzione di attivazione: Relu (Recrified Linear Unit), mentre il layer finale composto da 2 unità che si occupano di prevedere le due classi utilizzando come funzione di attivazione: Sigmoid.
In questo modo normalizzando e vettorizzando il dataset con valori compresi tra 0 e 1 è stato possibile ottenere una classificazione sufficiente per ottenere un buon grado di performance del nostro modello.
La due labels classificate sono appunto:

0: Non c'è probabilità di Default

1: Buona probabilità di Default

Il peso delle classi all'interno del dataset di training è rispettivamente di:

0: 0,22

1: 0,78

Oltre alle label è possibile trovare anche la distribuzione di probabilità data dalla rete neurale, una versione coerente di quanto specificato è possibile trovarla all'interno del file: result.csv contente i risultati ottenuti dalla predizione utilizzando la rete e il file di test fornito con la consegna.

La rete oltre ai 3 layers utilizza anche:

- Funzione di Loss: RMSLE (Mean Squared Logarithmic Error)
- Optimizer: Adam
- Metrics: Accuracy

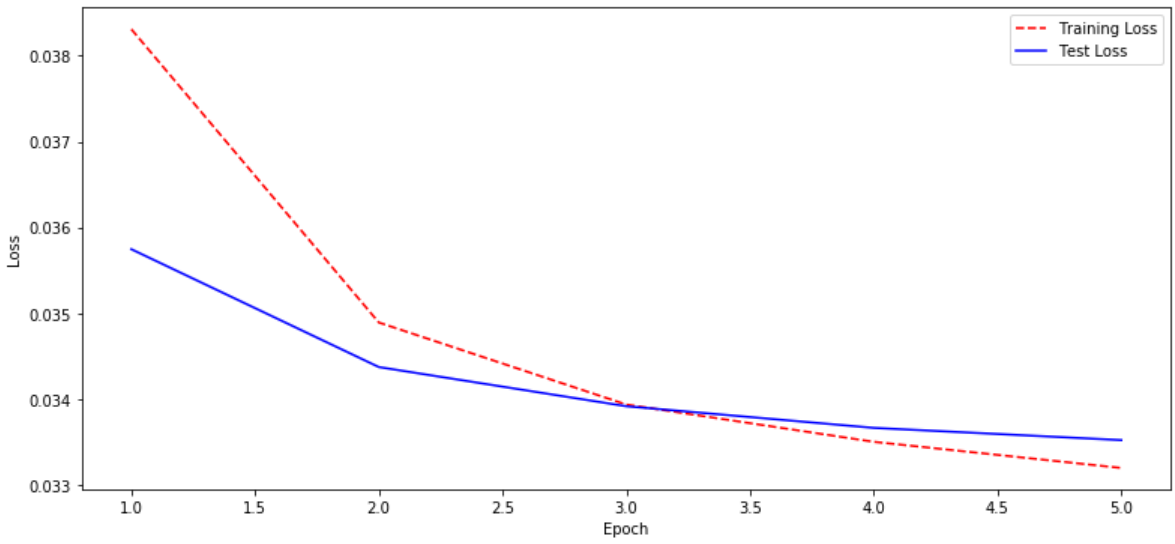
Le epoche sono state impostate a 5 in quanto siccome non si aveva a disposizione un dataset molto grande era inutile prevedere un numero superiore, così come il valore di batch_size che è stato impostato a 92 (4 volte il numero delle feature in input che è di 23 esclusa la label). Precedenti esecuzioni rispetto a quella finale hanno evidenziato quanto al crescere delle epoche il modello tendeva a overfittare rispetto al dataset di training, si è quindi rivelato inutile utilizzare un numero più ampio di epoche e di barch_size.

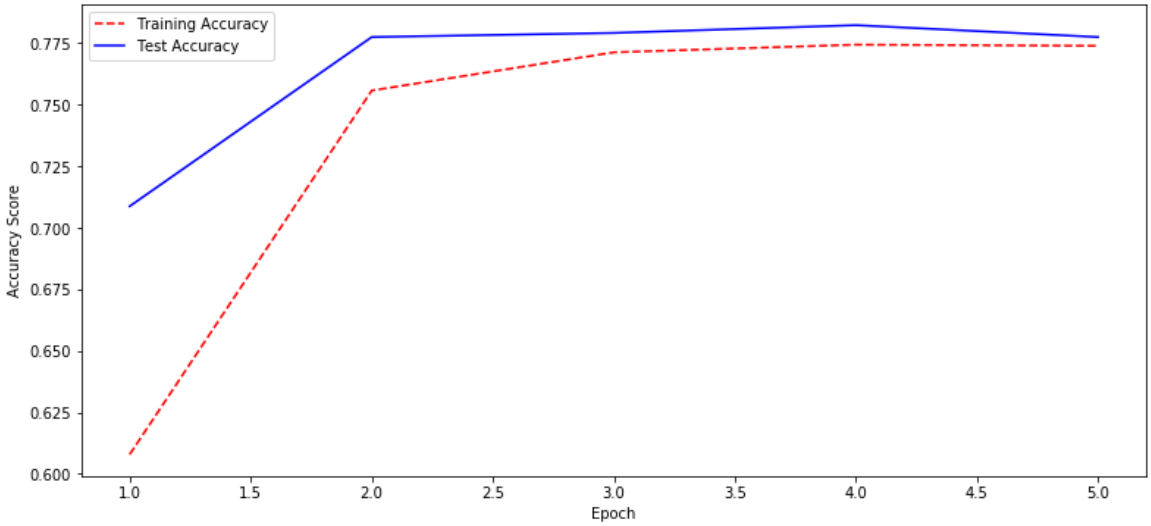
Il risultato finale ottenuto è il seguente

```
Number of rows predictors: 27000
Number of cols predictors: 23
Number of rows target: 27000
Number of cols target: 2
Class Weight: {0: 0.22207407407407406, 1: 0.7779259259259259}
Train on 21600 samples, validate on 5400 samples
Epoch 1/5
21600/21600 [==============================] - 1s 26us/step - loss: 0.0383 - acc: 0.6079 - val_loss: 0.0357 - val_acc: 0.7087
Epoch 2/5
21600/21600 [==============================] - 0s 11us/step - loss: 0.0349 - acc: 0.7557 - val_loss: 0.0344 - val_acc: 0.7774
Epoch 3/5
21600/21600 [==============================] - 0s 10us/step - loss: 0.0339 - acc: 0.7712 - val_loss: 0.0339 - val_acc: 0.7791
Epoch 4/5
21600/21600 [==============================] - 0s 10us/step - loss: 0.0335 - acc: 0.7743 - val_loss: 0.0337 - val_acc: 0.7822
Epoch 5/5
21600/21600 [==============================] - 0s 10us/step - loss: 0.0332 - acc: 0.7738 - val_loss: 0.0335 - val_acc: 0.7774
```

Il valore di Accuratezza ottimale finale della rete è di: 0,77

Di seguito vengono riportati anche i grafici di Loss e Accuratezza tra trainingset e validation set






## Aggiunte

Oltre alle varie funzionalità fin ora illustrate, il codice all'interno del file Jupyter consegnato prevede il salvataggio del modello e dei pesi in formato json e il loro caricamento.

È stato implementato lo scoring sul dataset di test fornito caricando il modello precedentemente salvato e il salvataggio in formato csv dell'output ottenuto.

In questo modo il modello è facilmente deployabile all'interno di un'architettura e può essere utilizzato anche da altre applicazioni.