Abraham Lemus Ruiz A01207538

Intelligent Systems 2 Report

Logistic Regression and NN with Categorical Embeddings with Pytorch for School Deserting Dataset

Introduction

Mexican universities have a compromise with their students. Some of the challenges they face are to guarantee everyone has access to a quality education, form ethical leaders of tomorrow, and maximize the student's success.

Anahuac University wants to make their students take off the most of their school activities, that is why they compiled a lot of data about their students, their programs, studying patterns and performance so they can create a predictive model that helps them identify those students who need social and emotional help, to prevent them from deserting or failing.

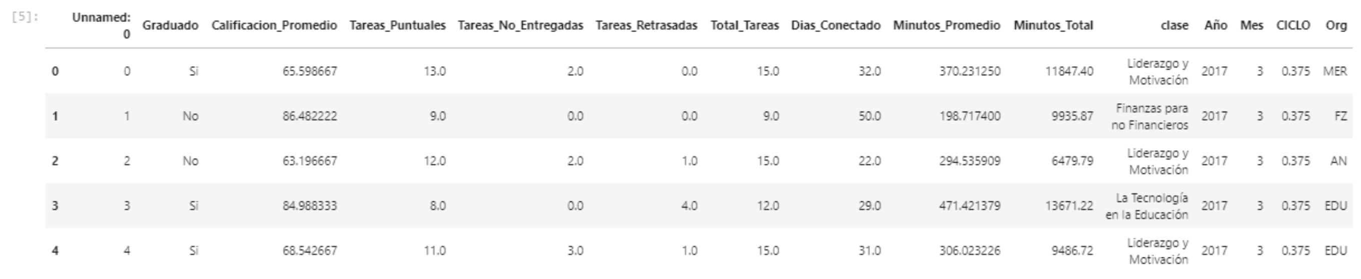
Objective

To create a predictive model capable of identifying students with high probability of failing.

The Data

A starting python notebook for data science can be found at https://github.com/maratonadev-la/desafio-6-2020-anahuac and in there, we can find the data sources.

The data comes in many files, and after some preprocessing we get the following dataset:



[5]:

	Unnamed: 0	Graduado	Calificacion_Promedio	Tareas_Puntuales	Tareas_No_Entregadas	Tareas_Retrasadas	Total_Tareas	Dias_Conectado	Minutos_Promedio	Minutos_Total	clase	Año	Mes	CICLO	Org
0	0	Si	65.598667	13.0	2.0	0.0	15.0	32.0	370.231250	11847.40	Liderazgo y Motivación	2017	3	0.375	MER
1	1	No	86.482222	9.0	0.0	0.0	9.0	50.0	198.717400	9935.87	Finanzas para no Financieros	2017	3	0.375	FZ
2	2	No	63.196667	12.0	2.0	1.0	15.0	22.0	294.535909	6479.79	Liderazgo y Motivación	2017	3	0.375	AN
3	3	Si	84.988333	8.0	0.0	4.0	12.0	29.0	471.421379	13671.22	La Tecnología en la Educación	2017	3	0.375	EDU
4	4	Si	68.542667	11.0	3.0	1.0	15.0	31.0	306.023226	9486.72	Liderazgo y Motivación	2017	3	0.375	EDU

Columns:

- 'Unnamed:0', id
- 'Calificacion_Promedio', avg grade
- 'Tareas_Puntuales', how many hw delivered in time
- 'Tareas_No_Entregadas', how many hw not delivered
- 'Tareas_Retrasadas', how many hw delivered but not in time
- 'Total_Tareas', sum of prev 3 cols
- 'Dias_Conectado', how many days student attends the school online

- 'Minutos_Promedio', avg time per online connection
- 'Minutos_Total', days_connected * minutes
- 'clase', name of their class taken
- 'Año', year (between 2017, 2019)
- 'Mes', between 1-8
- 'CICLO', an aggregate including date
- 'Org': what organization does the class belong to.

Our target column will be "Graduado".

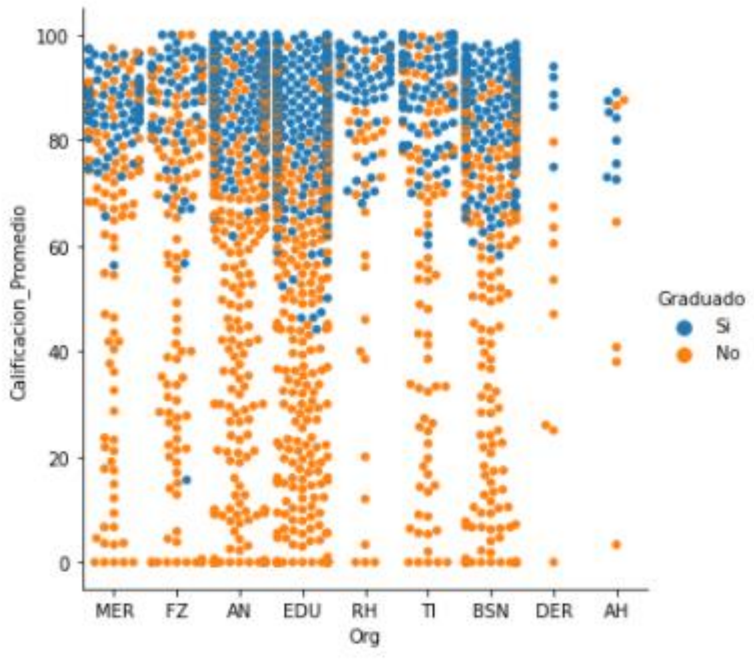
We can outright eliminate some of the columns:

- "Unnamed: 0", because it's an id column
- Tareas_No_entregadas: it's the result of other columns computations
- Tareas_Retrasadas: same as previous one
- Clase: Data is only 2500 rows and there are over 190 unique classes, so the data for each will be minor and we can probably not get much out of it.

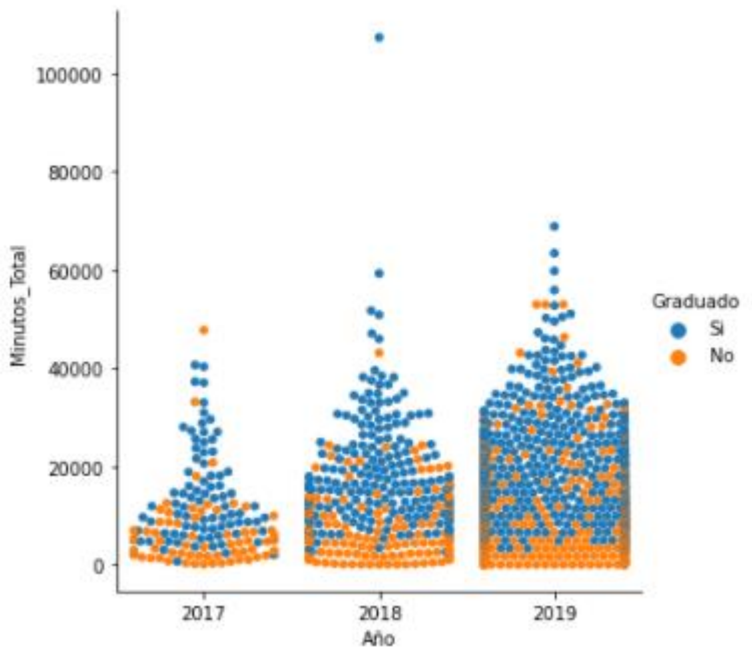
<u>Data Exploration</u>

We want to derive some insight into how the data behaves.

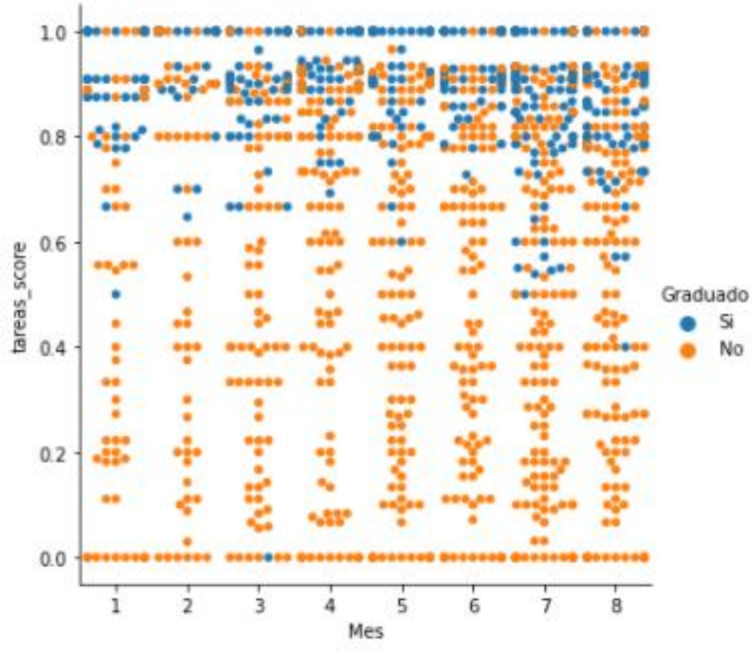
# Plot Grades vs Organization vs Graduado



# Plot time vs year vs Graduado



Plotting hw score and the month of course taken



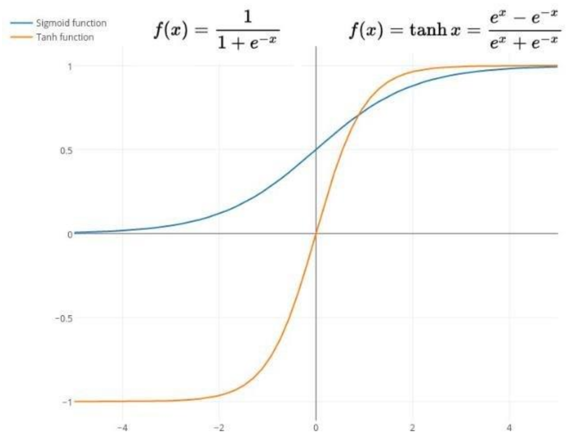
Some interesting points:

- The score affects a lot whether they graduate or no, but it’s far from the only factor
- Month and Year data describes data that represents the target in a not so linear way
- Many variables are continuous, so not a lot of problem there.

Logistic Regression Model From Scratch

I wrote an LR model from scratch using numpy, which is just the regular old Linear Regression model paired with an activation function.

Sigmoid function:



Forward propagation is the way our model learns its parameters, there are some things that need to be calculated:

A = sigmoid(mx + b), adds non linearity to our predictions

$$A = \sigma(w^T X + b) = (a^{(1)}, a^{(2)}, \ldots, a^{(m-1)}, a^{(m)})$$

Cross Entropy Loss J

$$J = -\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log(a^{(i)}) + (1 - y^{(i)}) \log(1 - a^{(i)})$$

Prediction:

$$\hat{Y} = A = \sigma(w^T X + b)$$

Back propagation is done with gradient descent, taking the derivatives of the values used for forward propagation, with the aim to optimize our parameters and reach a global minima for our loss function.

dW: gradient for W parameter.

db: gradient for b parameter.

$$\frac{\partial J}{\partial w} = \frac{1}{m} X(A - Y)^T$$

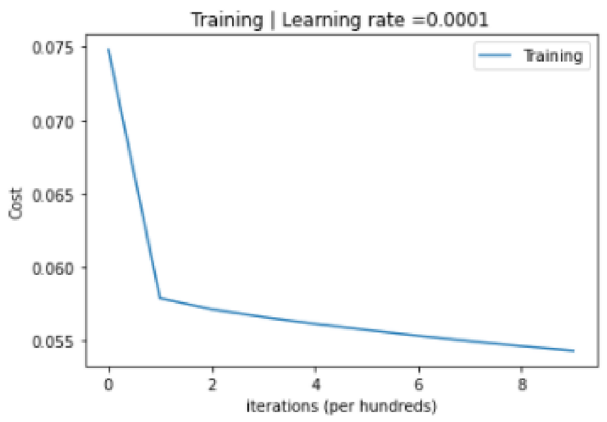
$$\frac{\partial J}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} (a^{(i)} - y^{(i)})$$

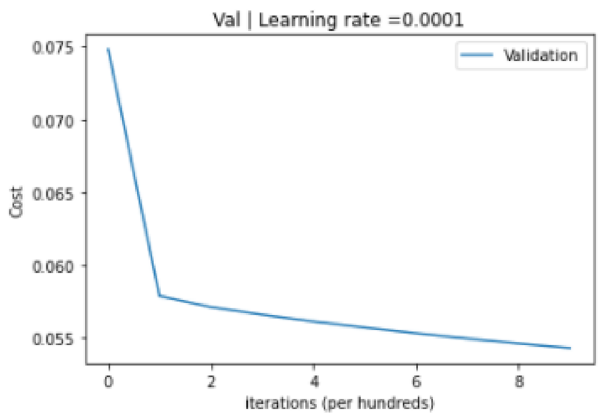
<u>Training the model:</u>

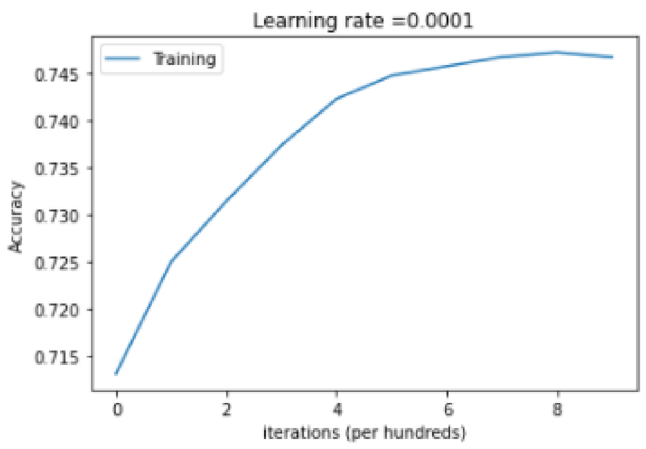
Hyper parameter tuning was done to get the most performance out of our model. The parameters tuned were the learning rate and the number of epochs, results:

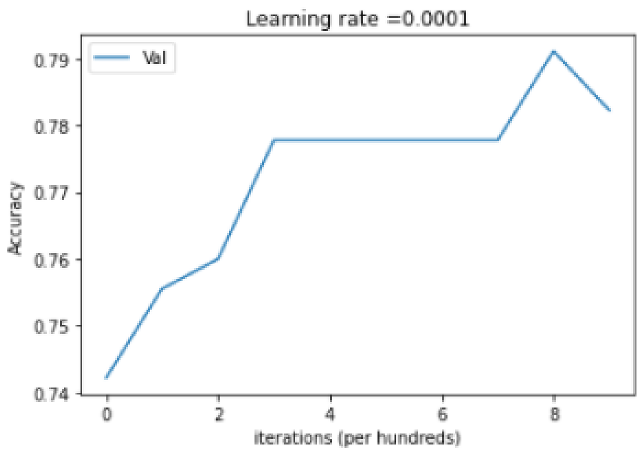
	Learning Rate	Epochs	Train Cost	Val Cost	Train Acc	Val Acc
3	0.00010	1000.0	0.054316	0.054316	0.746667	0.782222
5	0.00001	1000.0	0.058023	0.058023	0.723951	0.751111
4	0.00001	100.0	0.076789	0.076789	0.713086	0.742222
2	0.00010	100.0	0.074817	0.074817	0.713086	0.742222
0	0.00100	100.0	0.062030	0.062030	0.713086	0.742222
1	0.00100	1000.0	0.504981	0.504981	0.685432	0.666667

Plotting the best model:





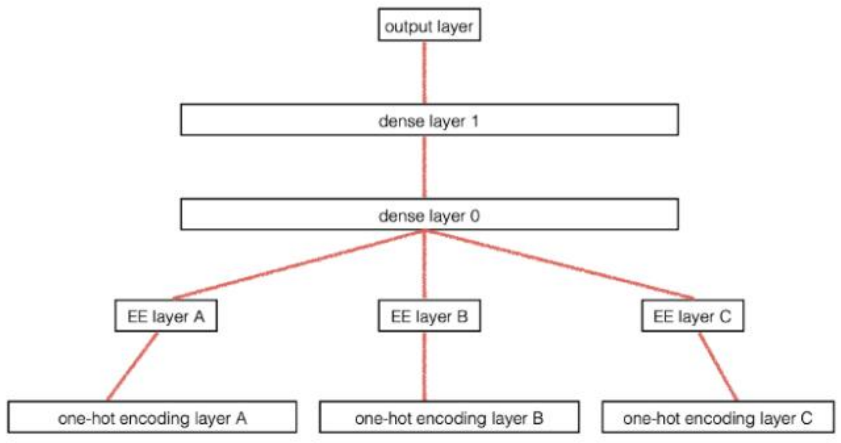




Some insights: the model is fitting well to the training data and the validation metrics may infer that its also fitting well for the whole data, although the accuracy plot seems weird.

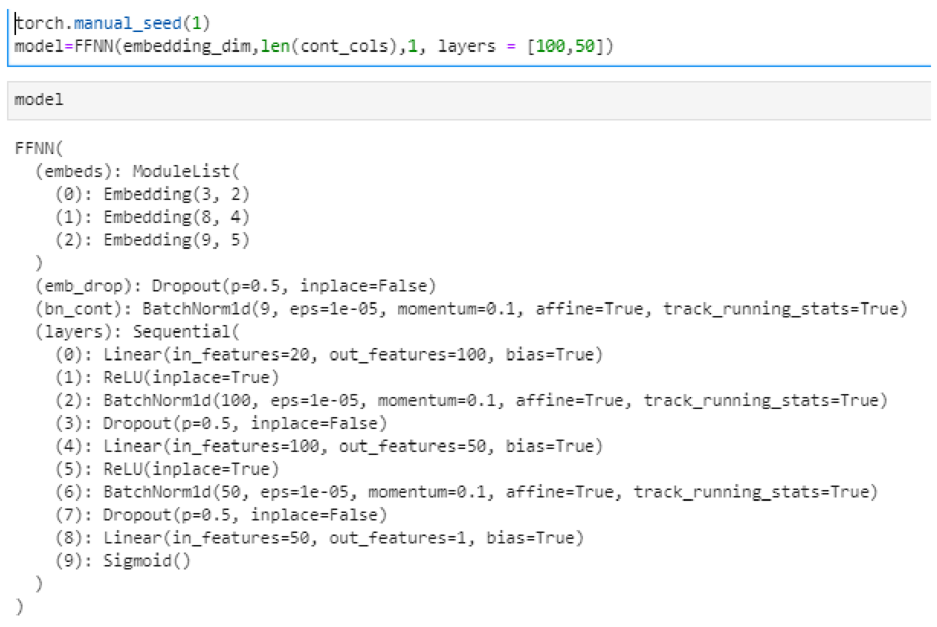
Deep Learning Model with Pytorch

It is well known in industry that deep learning is better suited for tasks like image recognition and NLP than to tabular data because its hard to interpret nn parameters, tabular data dimensions are way less than images, for example, and only a few features are almost always responsible for the majority of the accuracy score. A useful technique used for tackling this problem is “Categorical embedding”:



It’s the same idea of nlp using words for embedding layers but now for categorical features, and we can use them for our model instead of the sparse and inefficient one hot encoding and label binarizers.

A sample model looks like this:



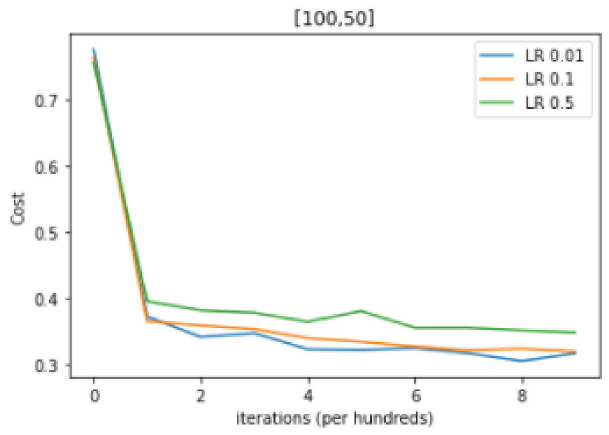
```
torch.manual_seed(1)
model=FFNN(embedding_dim,len(cont_cols),1, layers = [100,50])
```

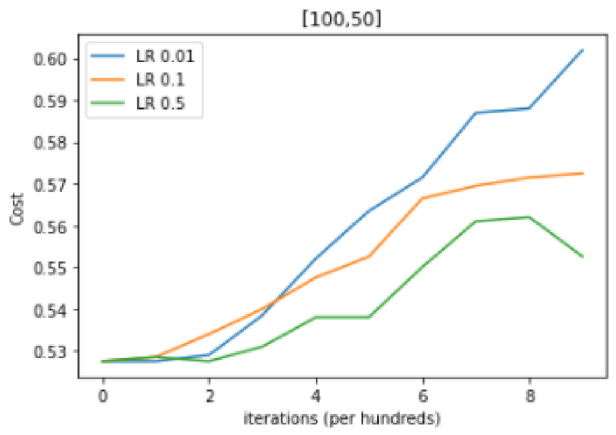
```
model
```

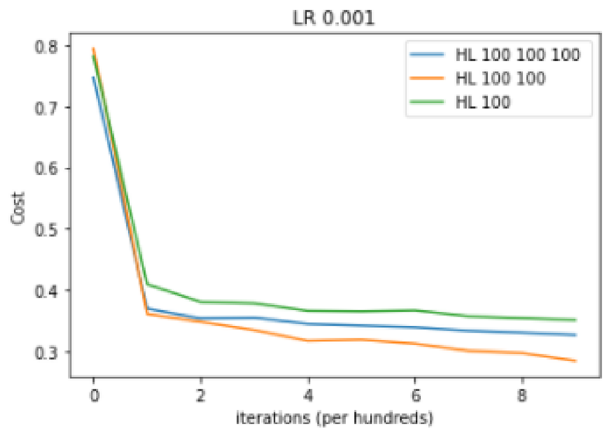
```
FFNN(
  (embeds): ModuleList(
    (0): Embedding(3, 2)
    (1): Embedding(8, 4)
    (2): Embedding(9, 5)
  )
  (emb_drop): Dropout(p=0.5, inplace=False)
  (bn_cont): BatchNorm1d(9, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (layers): Sequential(
    (0): Linear(in_features=20, out_features=100, bias=True)
    (1): ReLU(inplace=True)
    (2): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=100, out_features=50, bias=True)
    (5): ReLU(inplace=True)
    (6): BatchNorm1d(50, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (7): Dropout(p=0.5, inplace=False)
    (8): Linear(in_features=50, out_features=1, bias=True)
    (9): Sigmoid()
  )
)
```

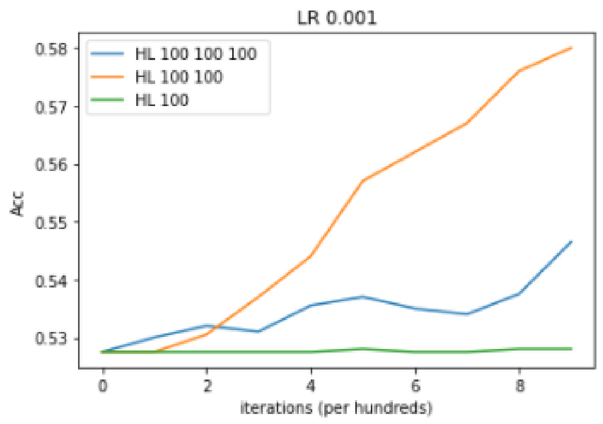
For this model, hyperparameter tuning was also done, in the variables learning rate and hidden layer sizes:

Results:







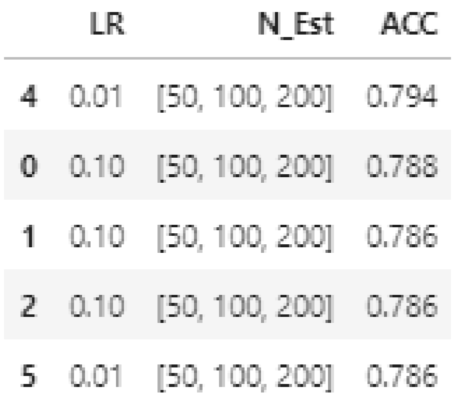


As we can see in this case, the addition of embedding layers do not seem to add a performance boost, more over, the added complexity of the model seems to be counterproductive.

Comparison of best models

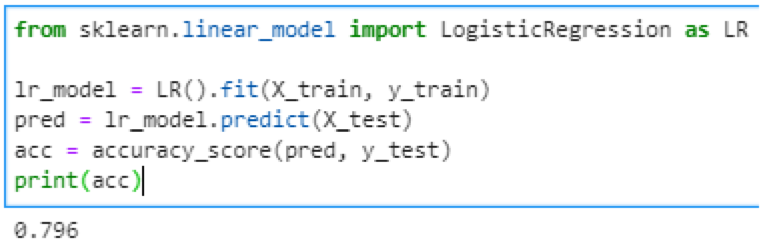
Just for the sake of completeness, I decided to perform accuracy evaluation on an XGB model, which is now a favorite for tabular data and a LogisticRegression from scikit

For the first model, tuning was done for the params learning rate and number of estimators:



	LR	N_Est	ACC
4	0.01	[50, 100, 200]	0.794
0	0.10	[50, 100, 200]	0.788
1	0.10	[50, 100, 200]	0.786
2	0.10	[50, 100, 200]	0.786
5	0.01	[50, 100, 200]	0.786

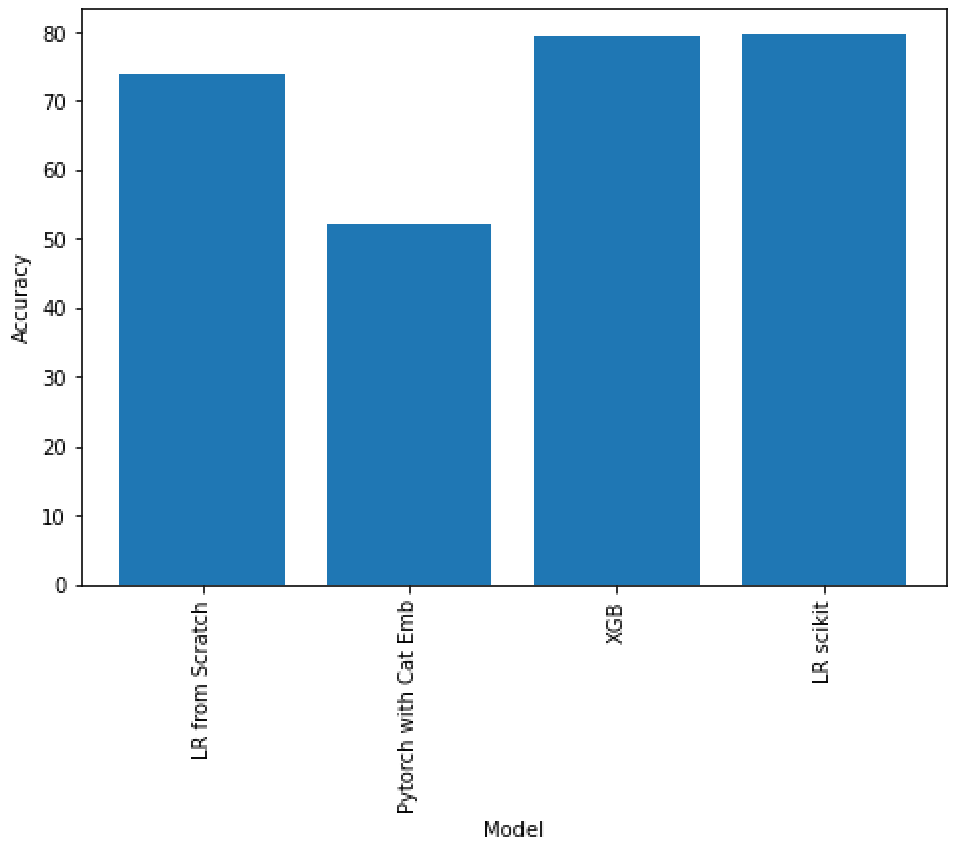
For the second one, pure training was done:



```
from sklearn.linear_model import LogisticRegression as LR

lr_model = LR().fit(X_train, y_train)
pred = lr_model.predict(X_test)
acc = accuracy_score(pred, y_test)
print(acc)
```

```
0.796
```



Even though Cat Embedding offer new ways to work with tabular data for dnn, it doesn't do miracles and we should always try different models

Sources

https://towardsdatascience.com/the-unreasonable-ineffectiveness-of-deep-learning-on-tabular-data-fd784ea29c33

https://towardsdatascience.com/how-to-gain-state-of-the-art-result-on-tabular-data-with-deep-learning-and-embedding-layers-d1eb6b83c52c

https://www.fast.ai/2018/04/29/categorical-embeddings/

https://github.com/maratonadev-la/desafio-6-2020-anahuac

https://arxiv.org/pdf/1604.06737.pdf