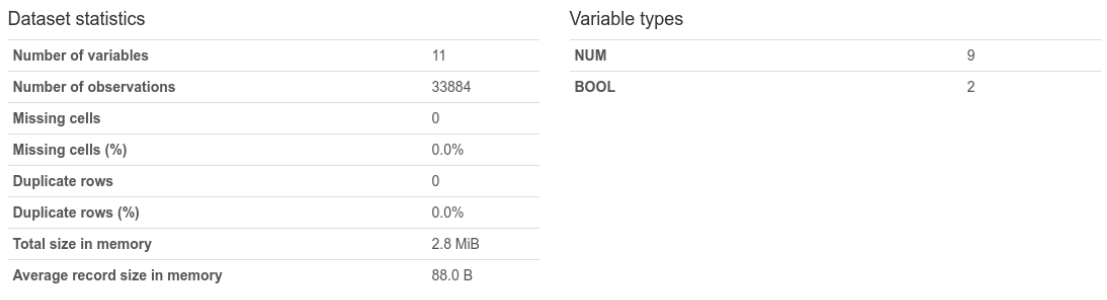
Relazione Assignment 1

Gianmaria Balducci

31 October 2020

Data Preprocessing

Goal: the prediction of the price of a private room/entire apartment using a neural network, from the dataset provided: *Airbnb apartments in New York City in 2019.*



Dataset statistics

Number of variables	11
Number of observations	33884
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	2.8 MiB
Average record size in memory	88.0 B

Variable types

NUM	9
BOOL	2

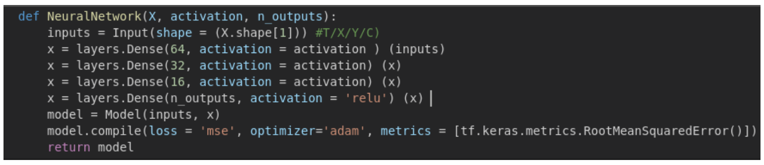
Figura 1: Overview on the dataset

The dataset provided is already divided in trainset, testset and the target. Train and test are complete, there is no *null* value in any variable, and there aren't duplicate rows.

Types of the variable's dataset are all float/integer, there isn't any categorical variable, therefore it no need to factorize any variable. Variable called *Unnamed: 0* is probably an error in phase of reading the csv file, it's a kind of index and it is deletd from the train, test and the target. In the traget variable *price* some prices reach values equal to 10000, 9999 so the row in transet that are related to this value are deleted. In *Latitude* and *Longitude* there are some values greater than +90 (latitude) and less than -180 (longitude) all these value are a thousand times larger, so i've decided to divide these value by 1000 in order to restore the real values. The trainset and the testset are standardized with *StandardScaler* from *sklearn* while for the target variable i decide to not standardize it and not even to normalize it in order to keep real value for regression task.

Model Creation

I split 10% of the trainset and the target in order to create a validation set.
The structure of the model is:



```
def NeuralNetwork(X, activation, n_outputs):
    inputs = Input(shape = (X.shape[1])) #T/X/Y/C)
    x = layers.Dense(64, activation = activation ) (inputs)
    x = layers.Dense(32, activation = activation) (x)
    x = layers.Dense(16, activation = activation) (x)
    x = layers.Dense(n_outputs, activation = 'relu') (x)
    model = Model(inputs, x)
    model.compile(loss = 'mse', optimizer='adam', metrics = [tf.keras.metrics.RootMeanSquaredError()])
    return model
```

Figura 2: Neural Network

It has 3 hidden layers consisting of 64, 32 and 16 units, the input layer contain units as many as the labels of the trainset and the output layer contain only one units as our target. I've tried to increase the number of the hidden layers but the result was worse. I've tried also to increase or decrease the number of units for each layer but this change does not affect the value of root mean squared error and the loss function.

Metric is root mean squred error related to the loss function *mse* and the optimizer *adam* in this model reach better performance that *sgd*.
I choose *ReLU* function as the output function because its range is (0, +inf) that is perfect for the range value of my target that is (0,+ 8500).
I tried different activation function in the hidden layers :

act_func = ['relu', 'elu', 'leaky-relu', 'selu', 'gelu', 'swish']

The best performance is given by *SeLU* function with **RMSE = 147.67** that is chosen for the final model.

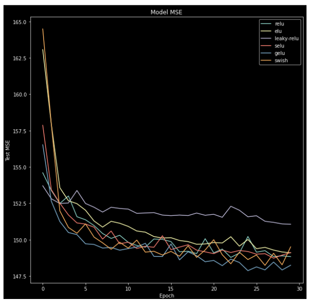
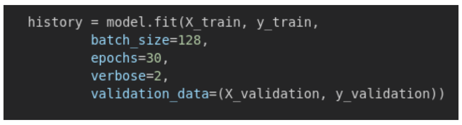


Figura 3: Model structure

The batchsize and the number of epochs are balanced for prevent to increase the variance.



```
history = model.fit(X_train, y_train,
          batch_size=128,
          epochs=30,
          verbose=2,
          validation_data=(X_validation, y_validation))
```

Figura 4: Model structure

If the number of epochs are less then 20 the **RMSE** of the training of the model increase, while increasing the number of epochs to 30/40/50 the error does not decrease the error and the training is slower. The validation error is less than the training error and it decreas every epochs. Therefore with unseen data the model behavior gets better but the error is still not very small.

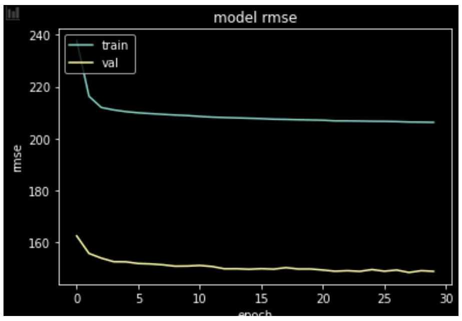


Figura 5: Training and Validation