Alma Mater Studiorum · University of Bologna

Machine Learning Project Report: Restaurant Revenue Prediction

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Machine Learning

Tiberio Marras (tiberio.marras@studio.unibo.it)

October 25, 2022

Contents

1 Introduction

1.1 Task description

The task of the project is to create a reliable mathematical model able to predict the revenue of 100.000 restaurants.
The model is trained using the *location*, the *age*, the *kind of city* where is located, the *type* of restaurant and three categories of obfuscated data (*demographic* data, *real estate* data and *commercial* data).
The model to find is a *regressor* which, given the features, must predict the eventual revenue of the restaurant.
The target doesn't strictly represent the real revenue but a number which relates to it.
The metric used to compare the results is *Root Mean Squared Error* (RMSE) over the 100.000 predictions.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

1.2 Background

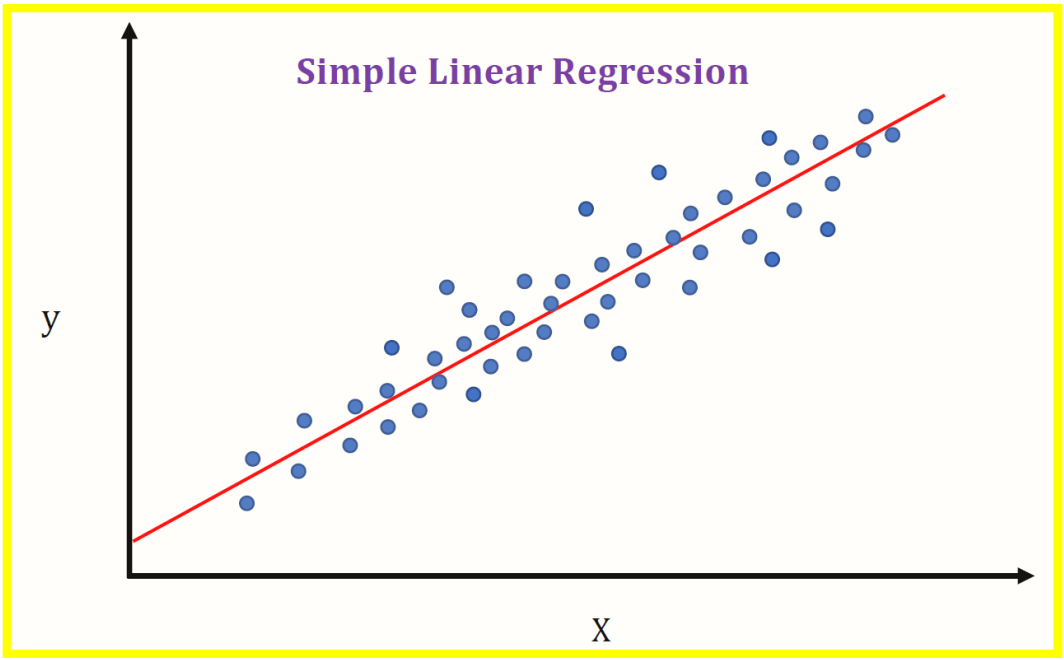
The regression is an approach for modelling the relationship between a scalar response and one or more explanatory variables.
The case of one explanatory variable is called *simple regression*; for more than one, the process is called *multiple regression*.
The relationships between the dipendent and the indipendent variables are modeled using predictor functions whose unknown model parameters are estimated from the data.
In machine learning there are several kinds of regressors such as *linear regressor*, *Ridge regressor*, *Lasso regressor* and others which we will see in the following sections.
Solving regression problems is one of the most common applications for machine learning models, especially in *supervised machine learning*. Algorithms are trained to understand the relationship between independent variables and an outcome or dependent variable. The model can then be leveraged to predict the outcome of new and unseen input data, or to fill a gap in missing data.
Common use for machine learning regression models include:

- Forecasting continuous outcomes like house prices, stock prices, or sales.
- Predicting the success of future retail sales or marketing campaigns to ensure resources are used effectively.
- Predicting customer or user trends, such as on streaming services or ecommerce websites.
- Analysing datasets to establish the relationships between variables and an output.
- Predicting interest rates or stock prices from a variety of factors.
- Creating time series visualisations.

Some of the most common regression techniques in machine learning can be grouped into the following types of regression analysis:

- Simple Linear Regression
- Multiple linear regression
- Logistic regression

To follow a graph representing the *Simple Linear Regression*.



2 The dataset

The dataset for the training is composed of 137 records representing restaurants.
Each row has 42 columns for the features and 1 column for the target.
The test set is composed of 100.000 records representing restaurant for which to predict the revenue. The test set doesn't contain the target variable.

2.1 Composition

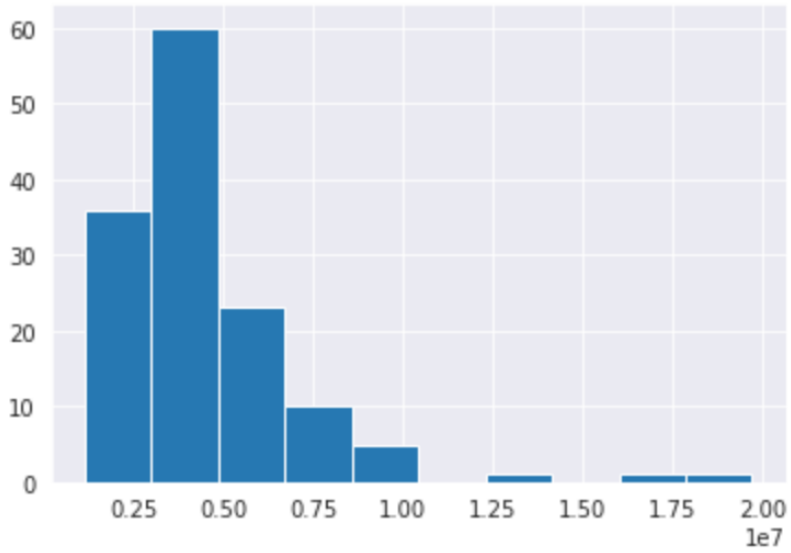
The features for the training are:

- *Id* - the id of the row
- *Open date* - opening date for a restaurant
- *City* - the city of the restaurant
- *City group* - the type of city. Big cities or others.
- *Type* - type of the restaurant. FC: Food Court, IL: Inline, DT: Drive Thru, MB: Mobile
- *P1, P2 - P37* - obfuscated data representing three categories: Demographic data are gathered from third party providers with GIS systems. These include population in any given area, age and gender distribution, development scales. Real estate data mainly relate to the m2 of the location, front facade of the location, car park availability. Commercial data mainly include the existence of points of interest including schools, banks, other QSR operators.

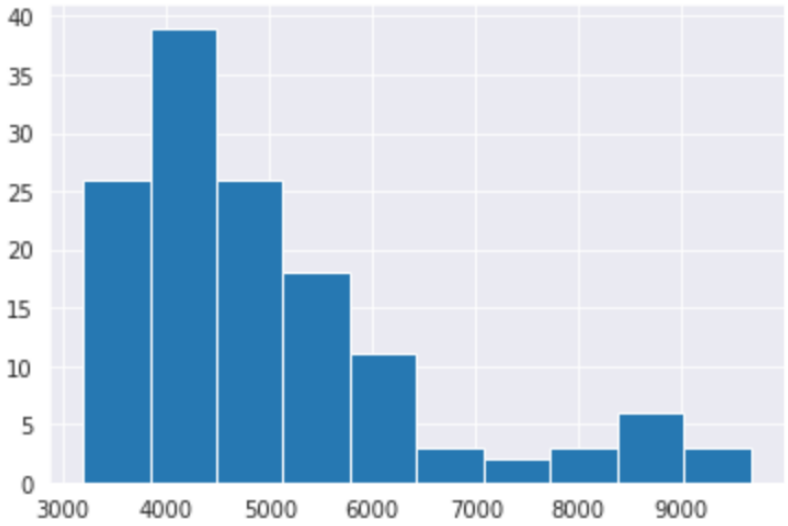
The target is the transformed revenue, since it is transformed it doesn't represent real dollar value.

2.2 Inspect the dataset

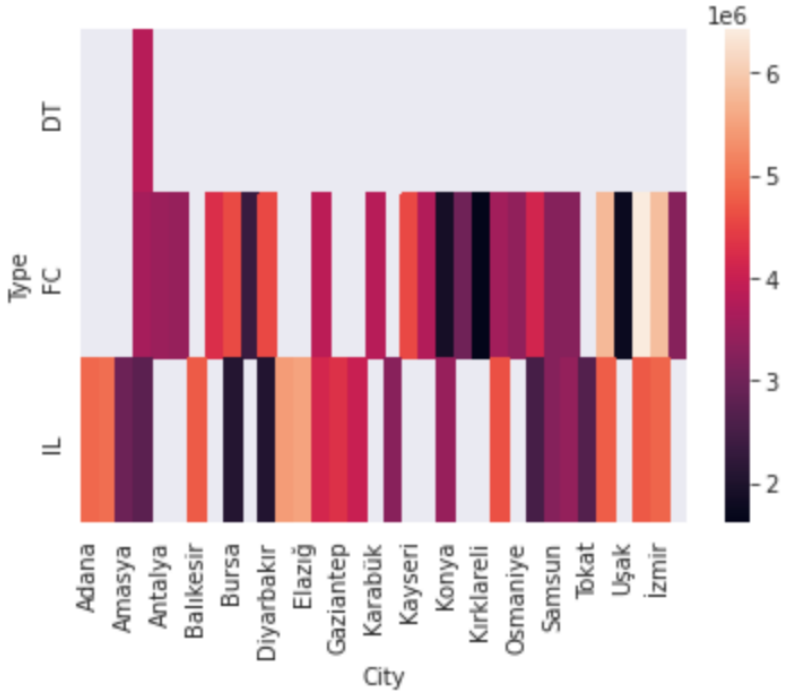
First I inspect the revenue.



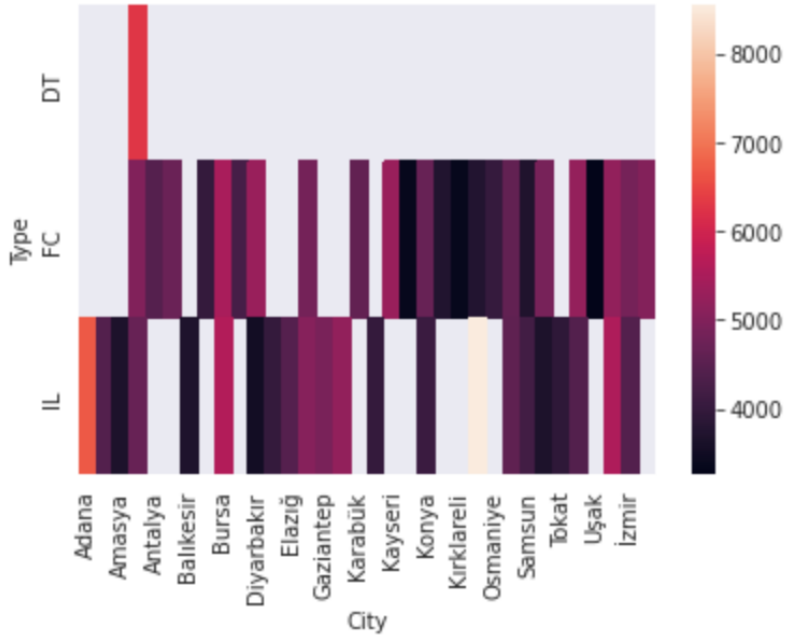
We can say that the majority of restaurants has *low* revenue. This can be caused by many factors. I also created a new feature which computes the number of days since the opening date in order to evaluate the age of the restaurant.



We can say that the majority of restaurants has a *young* age. This is because statistically it is more likely for a restaurant to fail after a certain amount of time.

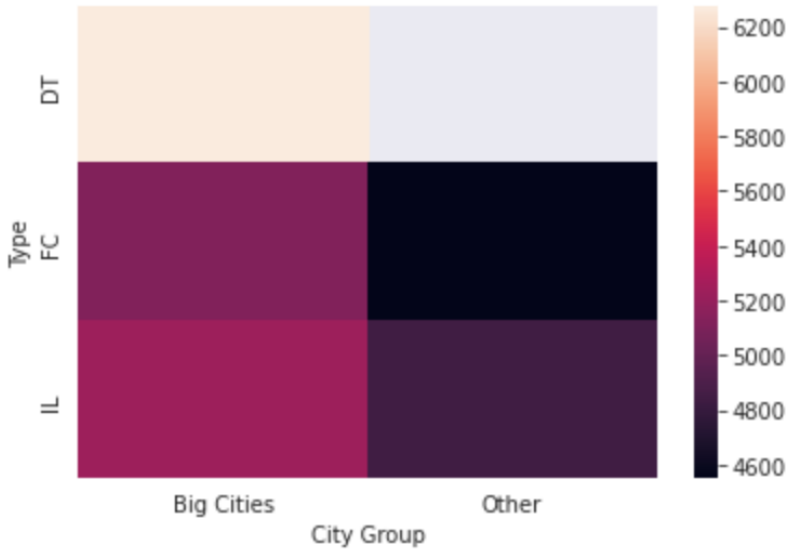


In this map the color of the cells is the revenue, we can also say that DT restaurants are not good at all.



In this map the color of the cells is the open date. As we already said the majority of restaurants

has a young age, and this doesn't make much difference between IL and FC restaurants.



In this map the color of the cells is the open date. As we can see restaurants in big cities tend to last more than restaurants in other cities, this is probably for the bigger number of customers which is easier to reach in big cities.

In this map the color of the cells is the open date. We can say that FC and IL restaurants tend to have bigger revenues in big cities than in other cities.

2.3 Preprocessing

I performed two different preprocessings for the classic regressors and the neural network.

2.3.1 Classic regressors

For the classic regressors I used `SimpleImputer` to fill the missing values in the dataset using the *median* as descriptive statistic.
Then I standardized numerical values using the `StandardScaler`.
I decided to perform standardization in order to make the models converge faster and reduce the computation time of training.
For the *city group* I used the `OrdinalEncoder` which encodes the categorical values as integers.
In this case I decided to use `OrdinalEncoder` because of the fact that the categories are only two.
For the *type* of restaurant I used `OneHotEncoder` which encodes the categorical values ad one-hot encoded arrays.
In this case, differently from the *city group*, the number of categories is greater than 2, this is why I decided to perform one-hot encoding.
In order to reduce the number of features I used *PCA* with a number of components equal to 0.99.
This way the number of features became 27.

2.3.2 Neural network

For the neural network I performed the same preprocessing steps except for the `StandardScaler` for the numerical features.
In this case I used a `MinMaxScaler` instead, which scales the feature using the min - max range.
I decided not to use standardization in this case in order not to change the distribution of data.
I used *PCA* to reduce the number of features with the neural network as well, but this time with number of components equal to 0.95.
This way I reduced the number of features to 13.

3 Methods

To find the best regression model I performed a `GridSearchCV` over various regressors (included the neural network) and I compared the results.
For the classic regressors I tried with various combinations of parameters the following ones:

- *Support Vector Regressor* (I tuned on Kernel, Epsilon, Gamma and C)
- *AdaBoost Regressor* (I tuned on n_estimators, learning_rate and random_state)
- *KNeighbors Regressor* (I tuned on n_neightbors, weights and metric)
- *SGD Regressor* (I tuned on Alpha, max_iter, loss and Penalty)
- *Ridge* (I tuned on Alpha)
- *Lasso* (I tuned on Alpha)

During the loop on `GridSearchCV` I create 2 dictionaries with the best parameters and the best score for each regressor.
Using these dictionaries the program can compare each result in order to find the best regressor available.
In the case of the neural network I created a function `create_model()` which takes various parameters for the creation of the model.
The parameters I tuned on are *Batch_size*, *Epochs*, *Learning_rate*, *Momentum*, *Dropout_rate*. I tried various architectures changing the number of layers, the number of hidden nodes and the activation functions.
The activation function is the classic *Relu* for the hidden layers and *Sigmoid* for the output layer.
After tuning on the neural network I compared the result of `GridSearchCV` on the neural network with the one of the best regressor found.
For the cross-validation of `GridSearchCV` I used 5 folds and I used *Negative Root Mean Square Error* as scoring to compare the results.

4 The environment

The environment I used is a *Python 3* notebook with *Colab*. I created a directory in my *Google Drive* with the project and a subdirectory with the dataset. I used the *Drive* library of *Colab* to mount the path to the directory in order to fetch the training and the test datasets to load on a dataframe.
The output file with the results on the test set is located on the same subdirectory of the dataset.
I used the free version of *Colab* with standard *RAM* and *Disk*.
The *Colab* environment has 2 processors *Intel(R) Xeon(R)* with 1 core each.
Each processor has a *frequency clock* of 2200.218 MHz and a *cache size* of 56320 Kb.
To follow the charasteristics of one processor on *Colab*.

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 79
model name      : Intel(R) Xeon(R) CPU @ 2.20GHz
stepping        : 0
microcode       : 0x1
cpu MHz         : 2200.218
cache size      : 56320 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 c
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs taa
bogomips        : 4400.43
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

As a first step I had to perform `!pip install scikeras` since the library wasn't available by default on *Colab.*

5 Results

The best model resulted to be *KNeighbors Regressor* with a *Negative Mean Squared Error* of -5795686580515.175. I obtained this result with the following combination of parameters:

- *Metric*: "Manhattan"
- *N_neightbors*: 15
- *Weigths*: "Distance"

To follow the result score for each regressor (with GridSearchCV).

Support Vector Regressor	AdaBoost Regressor	KNeighbors Regressor	SGD Regressor	Ridge	Lasso	Neural Network
-2544211.7854674375	-2458315.471066861	-2344581.3373125563	-2453311.810404105	-3077972.9178781994	-3170407.2124743764	-26364459747764.65

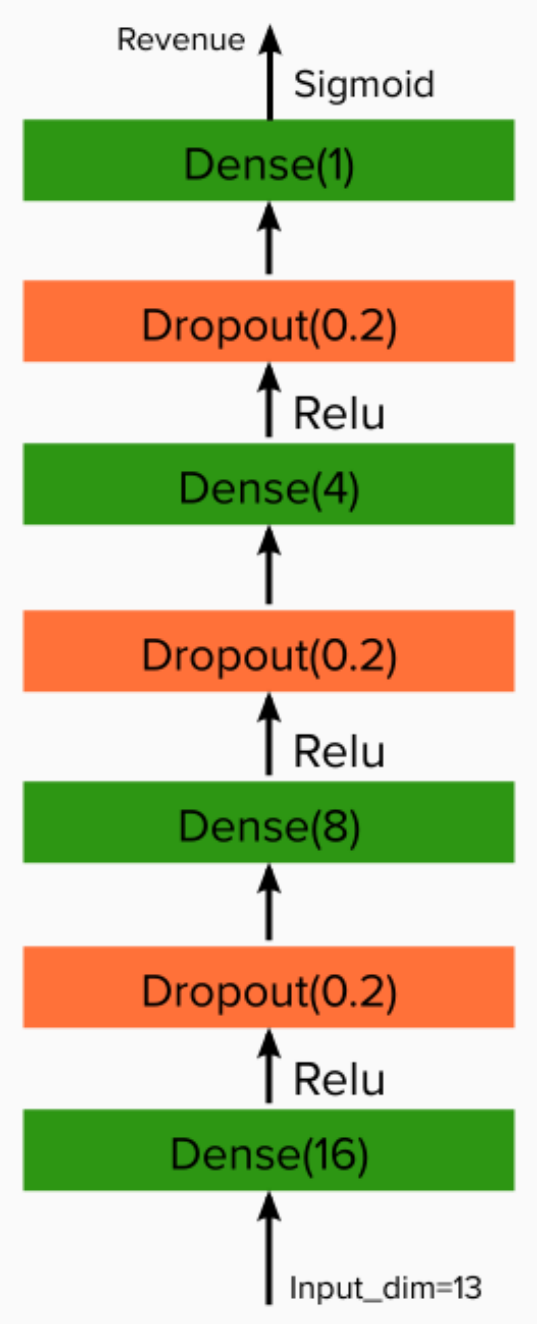
To follow the best parameters for each regressor.

Support Vector Regressor	{'C': 1000, 'epsilon': 1, 'gamma': 0.001, 'kernel': 'linear'}
AdaBoost Regressor	{'learning_rate': 0.1, 'n_estimators': 500, 'random_state': 2}
SGD Regressor	{'alpha': 1.0, 'loss': 'squared_error', 'max_iter': 10000, 'penalty': 'l2'}
Ridge	{'alpha': 1.0}
Lasso	{'alpha': 1.0}

For the neural network I obtained the best result with the following combination of parameters:

- *Batch_size*: 8
- *Epochs*: 60
- *Learning_rate*: 0.01
- *Momentum*: 0.2
- *Dropout_rate*: 0.2

The best architecture I found counts 4 *dense layers* (plus the *input layer*) with 16, 8, 4 and 1 *nodes* and 3 *dropout layers* with a dropout rate of 0.2.
To follow the architecture of the neural network.



5.1 Conclusion

To conclude the classic regressors seem to perform quite good compared to the *neural network*, which has the 5th best result.
Only *Ridge* and *Lasso* performed worse resulting respectively in 6th and 7th position.
A possible explanation for this is the fact that the training set is really small compared to the test set, and since neural networks are hungry of data this could really influence the results.
Actually a thing to point out is that the mean *Negative Root Mean Squared Error* computed on all the results is still high once all the models are trained, and this again is probably caused by the small dimension of the training set.
However this seems the best approach to tackle this kind of problem, also because the use of `GridSearchCV` makes the result score more reliable. `GridSearchCV` works even better with a bigger dataset, either with the same number of folds or increasing it.