Frida Nicole Polanco Dominguez

ML2 – MBD APR24 A1

Individual Assignment

Technical Annex

The present document describes data cleaning & EDA, model training and results for a logistic regression problem. Taking a dataset from Kaggle that describes properties in certain districts in Madrid we aim to predict if the monthly rent for the property will be higher than 850 EUR or not. This way, users can evaluate that their properties are correctly priced in Madrid's real estate market. The data was provided by real estate website idealista and can be found following this link. The original variables we had are as follows:

1. ***Address***: which includes the address in Spanish
2. ***Price***: In Euros €
3. ***Rooms***: Number of rooms in the houses, for Studios the values are NaN
4. ***Squared meters***: the squared meters of the renting house
5. ***Floor***: The floor in which the housing is
6. ***Link***: The hyperlink of the referencing house
7. ***Summary***: A brief description in Spanish of the house, including information such as the floor
8. ***District***: Malasaña, Moncloa, Lavapies, Chueca, Chamartin and La Latina.
9. ***Type***: Flat, Studio, Duplex and Attic
10. ***Pool***: True or False
11. ***Furniture***: True or False
12. ***Exterior***: True or False
13. ***Elevator***: True or False

Data Cleaning & Exploratory Data Analysis

To ensure an optimized dataset was used to train our classification machine learning model, multiple actions for data cleaning and EDA were taken. Below a brief executive summary of everything done.

Feature selection

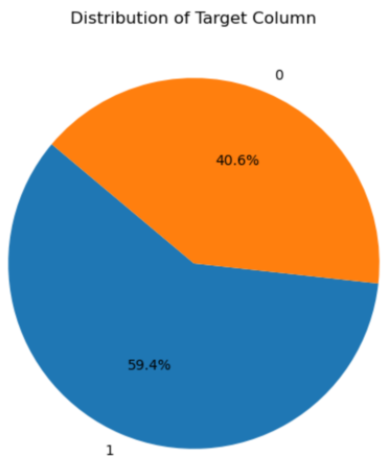
From the dataset there are three variables that will not help our model such as *link*, *summary* & *address* so these were dropped.

Nulls

The dataset was mostly complete, only Rooms column had missing values. However, our documentation specified that studios were filled with Nan values in rooms; so, we addressed this by replacing them with '1'.

Target variable

Our target variable, price, is continuous. However, our problem is classification, we created a target column by splitting our properties into two classes depending on the monthly price rent. **The main goal is to identify which properties can be valued at 850 EUR or more** monthly rent and which cannot be based on the market trends. Before moving forward, we checked that the distribution in our Target variable was balanced for a better fitted model.



With our target variable ready, we were able to drop column price as keeping it would be redundant.

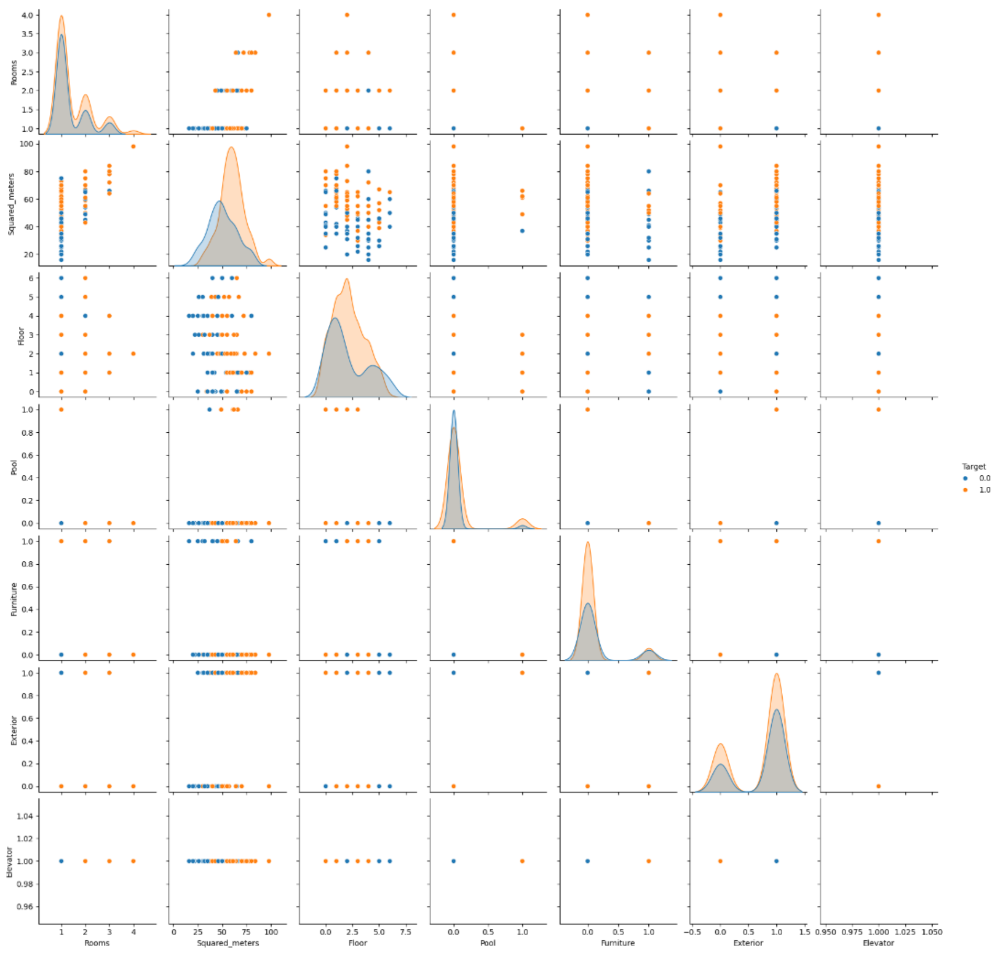
Datatypes

Each column was reviewed to understand if the datatype it has was correct or not. Multiple of our categorial variables were booleans; so, we changed the datatype to have them as binary and reduce the number of variables we will have to OneHotEncode further in the process. The variables that were affected by this change were *Pool*, *Furniture*, *Exterior*, *Elevator*, *Target*, and *Squared_meter*.

Data Visualization

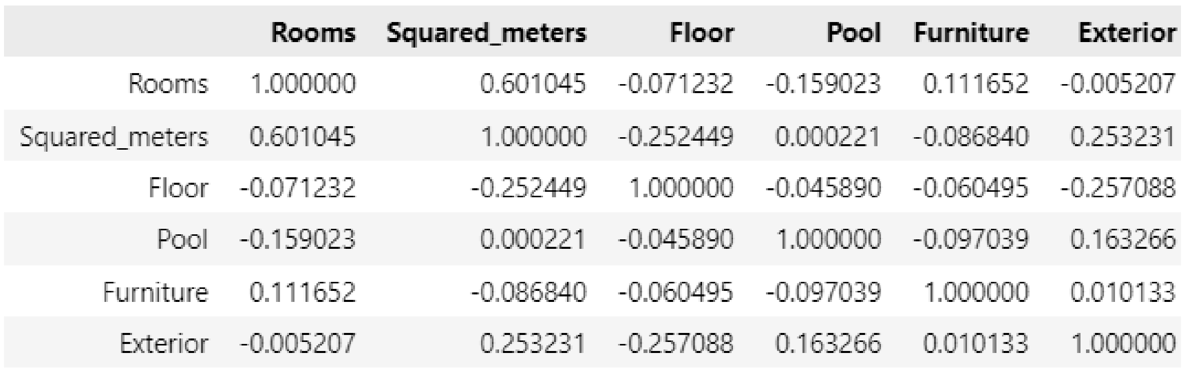
When creating a pair plot with seaborn we could observe the relationships between features including density plots along the diagonals. Additionally, we used hue to divide points between the two target classes to fully understand the distribution. From the below graphs key observations are:

- *Rooms* and *square_meters* have a clear distinction between each target class making them excellent features for classification Additionally, we can see a positive correlation between rooms and squared meters which makes sense with our industry knowledge. Normally, the more rooms, the bigger the apartment.
- *Floor* variable is distinctly distributed between target classes that can be observed in its density plot.
- *Pool* and *Elevator* seem to be more relevant than *Furniture* and *Exterior.*



Pairplot of variables

Checking correlations between our variables, the only high variable we could find is between squared meters and Rooms with 0.6. **However, the correlation factor does not surpass our threshold of |0.66| so we will keep both for the analysis**. Additionally, with the business context of the real estate industry we consider both variables are necessary as prospective clients will consider total square footage of the property and how many rooms it has make a final decision.

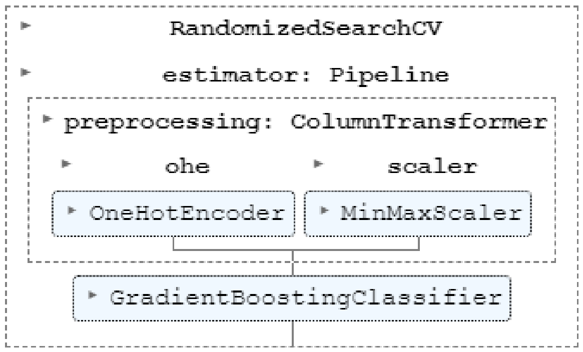


	Rooms	Squared_meters	Floor	Pool	Furniture	Exterior
Rooms	1.000000	0.601045	-0.071232	-0.159023	0.111652	-0.005207
Squared_meters	0.601045	1.000000	-0.252449	0.000221	-0.086840	0.253231
Floor	-0.071232	-0.252449	1.000000	-0.045890	-0.060495	-0.257088
Pool	-0.159023	0.000221	-0.045890	1.000000	-0.097039	0.163266
Furniture	0.111652	-0.086840	-0.060495	-0.097039	1.000000	0.010133
Exterior	-0.005207	0.253231	-0.257088	0.163266	0.010133	1.000000

Building the model

For this model we decided to run a **gradient boosting method to optimize our f1 score.** A summary of steps and decisions taken can be found below:

1. Split data into X_train, X_test, y_train, y_test with a train set of 70% and 30% for test.
2. Split our features into categorical and numerical. Although columns like *Furniture*, *Exterior*, *Elevator* & *Pool* are categorical; since they are already binary, they were considered as numerical columns to avoid encoding them. Therefore, our variables division resulted in:
    a. Categorical: *Floor*, *Distric*, *Type.*
    b. Numerical: *Rooms, Squared_meters, Pool, Furniture, Exterior, Elevator.*
3. The **first step of our pipeline is our data *preprocessing***. Using a column transformer where we encoded (using **OHE**) the categorical columns & scaled our numerical columns (using **MinMaxScaler**). We decided to go with MinMaxScaler to not affect our already binary columns.
4. We created **a first pipeline including our preprocessing steps and a gradient boosting classifier.** GBC was our chosen model for this problem.
5. To include **hyperparameter tunning** we created another 'black box' pipeline where we would iterate over multiple parameters to find the best combination. Within this big pipeline we would include our smaller pipeline (step 4) that has our preprocessing and GBC.
    a. Our parameter included multiple options for number of estimators (how many decision trees to try) and max depth (depth of the decision tree). This way by trying multiple options we will get the best feasible option. Considering our sample size is not huge, we kept the number of estimators low to, once again, avoid overfitting.
    b. We also included a **cross-validation** component of **5 k-folds**.
    c. It is also worthy of mentioning that our sample size complies with the recommended rule of n < k*30 for classification models.
6. Lastly, the model was fitted with our train data.



Final pipeline fitted.

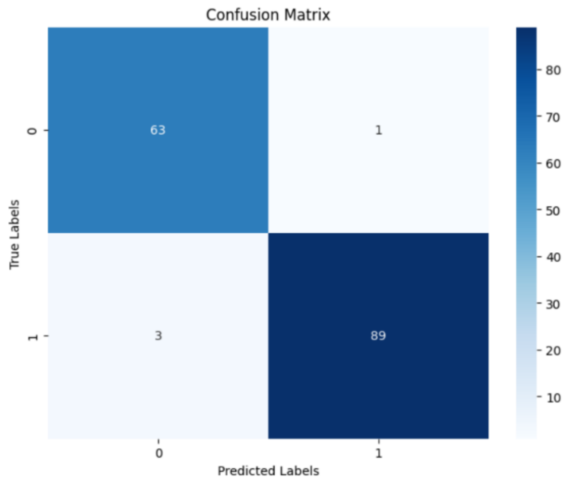
Evaluation

Once the model was trained, we used the prediction function to get predictions and evaluate the results. A comparison of the train & test results is showed further down.

As we can observe, our Train metrics resulted extremely high with precision and ROC being 1.0, in other words, a perfect score. Although we want high metrics, such high metrics could indicate that our model is overfitted and can't predict well on new data. Therefore, we need to investigate our test metrics to confirm or deny this hypothesis. If a model is overfitted, we would see very high scores on train metrics and comparably low scores on our test metrics.

When observing our Test metrics, we can see they are very high indicating that our model is great predicting. Moreover, our most important metric is the confusion matrix which we calculated afterwards. Our confusion matrix shows very few false values which also confirms the model predicts correctly.

Train	Test
Train Precision: 1.0 Train Recall: 0.977 Train F1: 0.988 Train ROC_AUC: 1.0	Test Precision: 0.989 Test Recall: 0.967 Test F1: 0.978 Test ROC_AUC: 0.989



Although it'd be great to consider our model finalized and move forward with the implementation; we can't. Our metrics clearly portray that predicting whether an apartment's rent is above/below our threshold is very easy for the model. This is likely due to the small dataset we have and that the dataset includes only certain districts in Madrid which might be more likely to have high rents. We would need to ensure that we have more volume and variety of data to have a better understanding of rent trends in Madrid. Otherwise, if we implement this model to our users, they will likely get great results if their properties are like our dataset, but inaccurate if they live in different conditions.

For this reason, we will move forwards with our webapp creation to have a demo of how the user facing interface would look once we have the final model. This ML model prototype will serve as a proof of concept of the feasibility of the model. Additionally, once the web app is complete, we can track user interaction to measure if the market is interested in the app before investing more time and money into developing it.

Saving the Model

Using joblib we serialized our best model into a sklearn pipeline we could use in our streamlit app for predicting with user input data. Before saving it, we reviewed the selected parameters which were n__estimators = 40 and max_depth = 5. A streamlit app was deployed to make this model accessible to our user.

To access the web app please go to: https://rentclassifier-fnpd.streamlit.app/