

Presentación: José YAYA

## # ML\_Real\_estate\_price\_predictions

### ## Project description

This repository contains the development of the **"ML\_Real estate price predictions"** project to predict real estate prices using an API requested to Idealista

Search API lets us integrate property information published on Idealista into their site or app.

<https://developers.idealista.com/access-request>

### ### Use case:

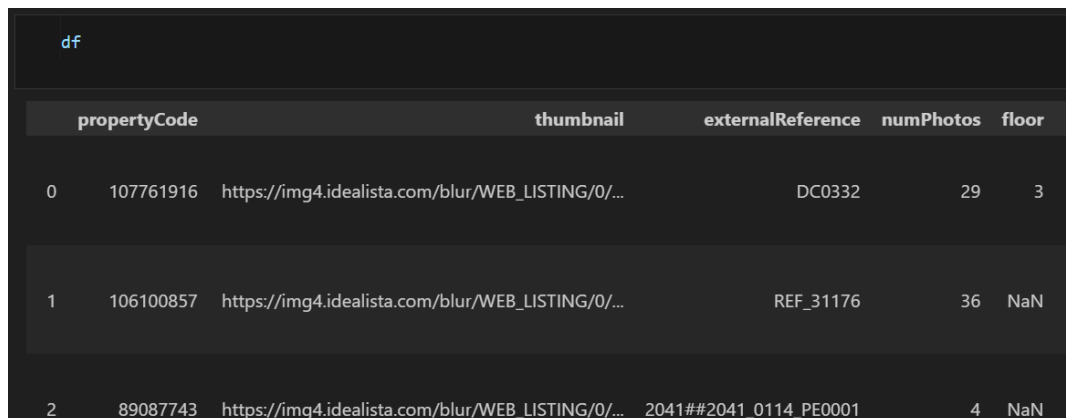
The main objective is to analyze whether sales prices represent a favourable purchasing opportunity in **Asturias**. This will allow potential buyers to make more informed decisions in the real estate market.

The prediction will be made using ML, extracting data using an Idealista API, where monthly searches can be performed and a model trained to calculate updated prices.

## Project: ML\_Real\_estate\_price\_predictions

### 0. Extracting data using the Idealista API, defining functions, parameters, URL, API search, performing pagination, total number of pages, and visualization of the dataset

# Saving the results in a DataFrame



	propertyCode	thumbnail	externalReference	numPhotos	floor
0	107761916	<a href="https://img4.idealista.com/blur/WEB_LISTING/0/...">https://img4.idealista.com/blur/WEB_LISTING/0/...</a>	DC0332	29	3
1	106100857	<a href="https://img4.idealista.com/blur/WEB_LISTING/0/...">https://img4.idealista.com/blur/WEB_LISTING/0/...</a>	REF_31176	36	NaN
2	89087743	<a href="https://img4.idealista.com/blur/WEB_LISTING/0/...">https://img4.idealista.com/blur/WEB_LISTING/0/...</a>	2041##2041_0114_PE0001	4	NaN

# Creating a copy of the DataFrame 'df' and storing it in the variable 'df\_total'.

```
df_total = df.copy()
```

### ### 1. Saving the Dataset

```
df_total.to_csv('C:/Users/argpe/REPO_PRUEBA_2/ML_Real_estate_price_predictions/src/data_sample/idealista_astur1.csv')
```

### 2. The function ``data_report(df)`` generates a comprehensive report on a DataFrame, offering valuable insights into its columns. This report serves as a crucial tool for our analysis,

data cleaning, and overall understanding of the dataset, helping us make informed decisions as we work with the data.

```
def data_report(df):
    '''Function generates a detailed report on a DataFrame'''
    # Creating a DataFrame containing column names
    cols = pd.DataFrame(df.columns.values, columns=["COL_N"])

    # Getting the data type of each column
    types = pd.DataFrame(df.dtypes.values, columns=["DATA_TYPE"])

    # Missing values:
    # Calculating the percentage of null values (missing values) for each column
    # percent_missing = round(df.isnull().sum() * 100 / len(df), 2) --> (*)
    percent_missing = round(df.isnull().mean() * 100, 2) # changed to the line above (*). It's more concise
    percent_missing_df = pd.DataFrame(percent_missing.values, columns=["MISSINGS (%)"])

    # Counting the number of unique values in each column
    unicos = pd.DataFrame(df.nunique().values, columns=["UNIQUE_VALUES"])

    # Calculating the cardinality percentage (proportion of unique values relative to the total number of rows)
    percent_cardin = round(unicos['UNIQUE_VALUES']*100/len(df), 2)
    percent_cardin_df = pd.DataFrame(percent_cardin.values, columns=["CARDIN (%)"])

    # Concatenating the results
    # concatenado = pd.concat([cols, types, percent_missing_df, unicos, percent_cardin_df], axis=1, sort=False)
    concatenado = pd.concat([cols, types, percent_missing_df, unicos, percent_cardin_df], axis=1) # 'sort=False' has b

    # Setting the column (COL_N) as an index of the resulting DataFrame
    concatenado.set_index('COL_N', drop=True, inplace=True)

    # Returning the resulting transposed DataFrame
    return concatenado.T
```

### ### 3. Working with the dataset 'df\_total'

# Accessing the dataset

df\_total =

pd.read\_csv('C:/Users/argpe/REPO\_PRUEBA\_2/ML\_Real\_estate\_price\_predictions/src/data\_sample/idealista\_astur1.csv')

df\_total.head()

	Unnamed: 0.1	Unnamed: 0	propertyCode	thumbnail	externalReference	numPhotos	floor
0	0	0	107761916	https://img4.idealista.com/blur/WEB_LISTING/0/...	DC0332	29	3
1	1	1	106100857	https://img4.idealista.com/blur/WEB_LISTING/0/...	REF_31176	36	NaN
2	2	2	89087743	https://img4.idealista.com/blur/WEB_LISTING/0/...	2041##2041_0114_PE0001	4	NaN
3	3	3	107683548	https://img4.idealista.com/blur/WEB_LISTING/0/...	00663	58	1

Performing some tasks such as df\_total.shape, len(df\_total.columns), df\_total.columns, Obtaining the number of apartments in Asturias (equal to 1119), Visualizing the data frame, Getting the data types, Calculating the percentage of missing values, among others.

### ### 4. SPLIT TRAIN/TEST

Splitting training and test sets (80-20%). Obtaining a table of average costs per square meter by district, and generating a bar chart and compare prices per square meter (m2) across the different districts in Asturias.

```
# Splitting training and test sets (80-20%) -> División de conjuntos de entrenamiento y prueba (80-20%)
X_train, X_test, y_train, y_test = train_test_split(apartments_asturias.drop('price', axis=1),
                                                    apartments_asturias['price'],
                                                    test_size=0.2,
                                                    random_state=42)
```

```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(895, 45)
(895,)
(224, 45)
(224,)
```

### ### 5. miniEDA

- Describing some columns, which will not be necessary for the analysis.
- Generating a horizontal box plot, allowing the visualization of the median, quartiles, and any outliers in the data.

Analysing the columns and defining a list called 'one\_hot', 'escalar', 'booleanos', among others.

- Calculating the relative distribution (proportion) of each category in the 'propertyType' column of the X\_train dataset, and defining a list 'escalar'.
- Visualizing the distribution of the 'size' column to filter out excessively high values. Finally, defining a list called 'escalar'
- Visualizing the values of the 'exterior' column, and defining a list called 'booleanos'
- Visualizing the contents of the 'rooms' column, seeing which is the largest apartment, and defining a list 'escalar'.
- Visualizing the contents of the 'bathrooms', and defining a list 'escalar'
- Visualizing the values "latitude, and longitude", and defining a list 'escalar'. Likewise, we generate a scatter plot with the coordinates, and create a new dataset 'df\_price\_coordinate (price, latitude and longitude)'
- Analyzing and visualizing the 'status' column, and defining a list 'one\_hot'
- Analysing the 'parkingSpace' column to create two new columns: (1) hasParkingSpace: 1 if there is parking available (True), and 0 if there is not (False). (2) IncludedInPrice: 1 if parking is included in the price (True), and 0 if it is not (False).
- Analyzing the 'detailedType' column
- Analyzing the 'floor' column to perform the following:
  - Replace 'bj' and 'en' with 0 (zero)
  - Replace 'ss' with '-1'
  - Replace 'NaN', the 'chalets' with 0 (zero)
  - Replace 'NaN', the 'flats' with the median
- Analyzing the 'district' column, and defining a list called 'booleanos'

Executing each task of the analysis performed above (miniEDA), and grouping based on the analysis performed previously.

- Visualizing the one-hot, scalar, and Boolean lists
- Standardizing the features
- Visualizing, and save X\_train

- Applying transformations to X\_test, and save **X\_test**.
- Verifying dimensions of both the X\_train and X\_test matrices

```
# Saving the X_train dataset
X_train.to_csv('C:/Users/argpe/REPO_PRUEBA_2/ML_Real_estate_price_predictions/src/data_sample/X_train.csv')
```

```
# Saving the X_test dataset
X_test.to_csv('C:/Users/argpe/REPO_PRUEBA_2/ML_Real_estate_price_predictions/src/data_sample/X_test.csv')
```

### ### 6. Model Training

Several models were trained, such as **RandomForestRegressor**, **XGBoost**, **ADABOOST**, **GradientBoosting**, **LGBM**, and **CatBoostRegressor** using **GridSearchCV**.

DataFrame was created, which summarizing the performance of different regression models in terms of mean absolute error (**MAE**) and root mean square error (**RMSE**) for both X\_train and X\_test.

#### X\_train:

	TRAIN MODELS	Mean_Absolute_error	Root_Mean_Squared_error
0	RandomForestRegressor	4130.749335	14187.329612
1	XGboost	3937.557378	10097.869228
2	ADABOOST	32340.087100	39538.877393
3	GradientBoosting	1109.055589	1590.257085
4	LGBM	12930.064417	70666.107770
5	CatBoost	1650.618062	2138.157343

In X\_train, the **GradientBoosting model** has the lowest MAE (1109.06), suggesting that it is the most accurate model among those evaluated on this dataset.

#### X\_test:

	TEST MODELS	Mean_Absolute_error	Root_Mean_Squared_error
0	RandomForestRegressor	4845.375070	15768.527180
1	XGboost	12045.745745	79095.986554
2	ADABOOST	30453.499198	36985.201982
3	GradientBoosting	1850.222612	5965.661787
4	LGBM	9935.323181	49675.470452
5	CatBoost	5331.470102	12358.232893

In X\_test, the **GradientBoosting model** appears to be the best in terms of accuracy according to these metrics, followed by RandomForestRegressor.

### ### 7. Optimizing hyperparameters

#### #### 7.1. Optimizing hyperparameters for **\*\*GradientBoosting\*\*** model using **\*\*GridSearchCV\*\***

```

'''
The hyperparameters to consider for optimization are:
- n_estimators: Number of trees in the model. [100, 200, 300]
- learning_rate: Learning rate, which controls how well each tree adjusts to the errors of previous trees. [0.01, 0.05, 0.1]
- max_depth: Maximum depth of each tree. [3, 5, 7]
- min_samples_split: Minimum number of samples needed to split a node. [2, 5, 10]
- min_samples_leaf: Minimum number of samples needed to form a leaf. [1, 2, 4]
- subsample: Proportion of samples to use to train each tree.[0.8, 1.0]
'''

```

Results:

```

RMSE TEST: 13623.931448665084
MAE TEST: 1367.5256983108227

```

Although the RMSE (13623) and MAE (1367) results are higher than those of the previous evaluation (RMSE: 5965, and MAE:1850), the RMSE value is still relatively high. This could be due to several reasons: the model might not be flexible enough, indicating possible underfitting; the hyperparameters might be too constrained, which could limit the model's ability to improve; or the dataset might contain noise or irrelevant features that affect performance.

## #### 7.2. Optimizing hyperparameters for **RandomForestRegressor** model using **GridSearchCV**

```

'''
The hyperparameters to consider for optimization are:
- n_estimators: Number of trees in the forest. [100, 200, 300]
- max_features: Number of features to consider when searching for the best split. ['auto', 'sqrt']
- max_depth: Maximum depth of the tree. [None, 10, 20, 30]
- min_samples_split: Minimum number of samples needed to split a node. [2, 5, 10]
- min_samples_leaf: Minimum number of samples that must be present at a leaf node. [1, 2, 4]
'''

```

Results:

```

RMSE TEST: 82086.124875275
MAE TEST: 40854.359700504916

```

The results of RandomForestRegressor are significantly worse than those of GradientBoosting, indicating potential issues with model configuration or data quality. This difference could be caused by overfitting of RandomForest, especially with many trees and high depth, which affects its performance on the test set. It is also possible that the hyperparameters are not optimized correctly, thus limiting the model's potential.

## ### 8. CONCLUSIONS:

Both models have room for improvement, but the performance of RandomForestRegressor suggests more serious issues that need to be addressed. It is important to perform data analysis and adjust the modelling approach and hyperparameters to optimize performance. I might also consider exploring other algorithms or modelling techniques that may be better suited to my dataset.

### ### 9. SUGGESTIONS FOR IMPROVING THE MODELS

#### 9.1. Suggestions for improving the **\*\*GradientBoosting\*\*** model:

I should probably increase the number of trees and the maximum depth in the hyperparameter search, try different regularization strategies, such as increasing the `min_samples_split` or `min_samples_leaf` parameter, perform feature selection or feature engineering to improve data quality, and consider using ensemble techniques with other models or creating model stacking.

#### 9.2. Suggestions for improving the **\*\*RandomForestRegressor\*\*** model:

Like the other model, expand the search ranges for hyperparameters in `GridSearchCV`, try using `'max_features'` as a percentage of features instead of a fixed number and see if that improves performance, perform cross-validation to ensure the model generalizes well and isn't overfitting, explore dimensionality reduction techniques (such as PCA, if there are many features), and review the data to detect and address outliers that may be influencing the model.