# Predicting AirBnB listing prices: a Machine Learning approach

## Objective

The primary objective of this project is to forecast Airbnb listings prices across various locations in Italy.
More precisely:

- Rome
- Milan
- Venice
- Florence
- Bologna
- Naples
- Bergamo

By leveraging different modeling techniques, we aim to provide a tool to observe the performance of various prediction models to accomplish the regression task at hand.

## Data Source

Data for this project has been sourced from various datasets extracted from the "Inside Airbnb" platform.
Each dataset includes thousands of listings, with each listing containing essential features such as price, location, amenities, host information, etc,which are crucial for building predictive models.

## Methodology

To achieve the project's goal, we merged the various dataset, in order to enrich the pool of data and increase both the training and the test set size.
Several machine learning regression models are employed and evaluated, each aiming to capture the characteristics of the data.

## Model Evaluation

The effectiveness of each model has been assessed some performance metrics, including:

- explained variance score, which measures the proportion of variance in the target variable that can be explained by the model;
- Mean Absolute Error (MAE), which provides an average of the absolute differences between predicted and actual prices;

- Mean Absolute Percentage Error (MAPE), expressing prediction accuracy as a percentage, facilitating comparison across different scales;
- Mean Squared Error (MSE), which calculates the average of the squares of the errors, emphasizing larger differences in predictions;

The project goal is to deliver a tool to observe forecast capabilities of different models and understand which is better suited to accomplish the task in a more precise manner.

# Project structure

The project source code is structured into several files and scripts, each containing distinct elements that are organized as follows.

## Src folder

The src folder contains some custom modules, written by us to automate processes that are repeatedly executed in the code. These modules are specifically aimed at containing custom Sklearn class transformers, custom Sklearn function transformers and some more general classes used to handle dataset importing and various tasks.
The custom Sklearn function transformers were created using the FuctionTransformer class from the preprocessing module in Sklearn and are used to handle more easily simple feature transformation processes that still cannot be handled with the default Sklearn transformers.
All the class Sklearn transformers were generated using the BaseEstimator and the TransformerMixin classes, which enabled us to inherit base functions for Estimator and Transformer creation, respectively.

## Data folder

### Mappings

The mappings folder contains several files, mostly json files that are used to remap several categorical features into features with lower cardinality of values. Specifically, the baths.json and the neighbourhoods_levels.json were used for this purpose. The host_locations.json contains the remapping of latitude and longitude into cities; the strategic_locations.json contains the locations of some meaningful places into the Venice city.
The create_map.py script was used to generate the html map of Listings, embedded in the Streamlit App; moreover the it.json file contains the coordinates needed to plot the polygons used for the creation of the map.

### Other files in data folder

Data folder contains the testing_df.csv dataset, which is the final dataset used by each training process as the input dataset for the Sklearn pipelines. The all_cities folder contains the data source files that were downloaded from the InsideAirBnB website. The city_data folder contains the csv dataset used for data visualization purposes.

# Jupyter notebooks

## Dataset creation

This jupyter notebook is used to create the first version of the final dataset, which is then used as input for the training pipelines. This notebook contains all the initial transformations, raw features engineering and some exploratory steps that were needed in order to analyse the data.

## Models

The project includes the training of five different ML models, all based on the Sklearn library. Specifically:

- Multi Layer Perceptron regressor
- Random Forest regressor
- K-Nearest Neighbor regressor
- Linear Support Vector regressor
- Nu Support Vector regressor

These models were chosen among the ones provided by Sklearn because they are better suited to handle regression tasks on a dataset with a high number of features (roughly 60 features).
Each notebook contains an initial setup section, where some variables are defined in order to isolate different feature categories. Moreover, some other data transformation steps are included, like text features handling. Then in each notebook we defined a custom Sklearn pipeline that contains several data transformation steps to be executed each time the training or prediction is carried out. Inside each Pipeline we used several kind of transformers:

- base sklearn transformers;
- Custom sklearn transformers;
- feature_engine transformers;

Where *feature_engine* is a sklearn-compatible python library that provides a wider range of sklearn transformer. These transformers can be included in base Sklearn Pipelines to handle more complex transformation tasks (like dates subtraction, frequency encoding and more complex imputation strategies).
The last step of each Pipeline is constituted by a different Sklearn model.
As a last step for the model training process, we decided to implement a Grid Search process for every Model, to isolate the best combination of hyperparameters to be used to optimize for some error measures. Once the grid search is executed and the best combination of hyperparameters is discovered, the best estimator is used to finally train the model on the train dataset.
At the end, the trained pipeline is dumped and saved in a file, in order to be used when needed (specifically during the prediction process in the Streamlit App).

## From saved models

This jupyter notebook contains the prediction steps carried out with the use of the test set.

This is a notebook version of what's in the "Models and prediction" page in the StreamlitApp: for each model, it contains some metrics and visuals to better grasp the functionality of the model and its effectiveness on executing the prediction.

## Pickle folder

This folder contains the dumped file versions of the five models, saved as pkl binaries with the use of the joblib library.
The process of model saving is particularly useful once the trained model needs to be recalled for prediction by other applications or scripts. Saving trained models as binaries increases model portability and enabled us to then cache the models in the Streamlit App, thus increasing the speed of price predictions.

## App folder

### Homepage script

This script contains the source for the Streamlit App homepage, in which a general overview of the project is included.

### Custom folder

This folder contains a module used for some custom settings of the Streamlit App.

### Pages folder

This folder contains three scripts, each one contains the source code for the respective App page.

- The map script contains the code for embedding the HTML map (generated with the create_map.py script contained in the mappings folder) inside the Streamlit app. The HTML code has been wrapped inside a compatible Streamlit object, in order to facilitate the user navigation in the map;
- The Models and Predictions script contains the source code for them Models and Predictions page of the App. Inside this page the user can choose among the five models and, by pressing the button in the page, the App returns
    - The Pipeline used for the prediction, that is the pipeline with the best estimator;
    - Some performance metrics, specifically the Explained Variance, the Mean Absolute Error, Mean Absolute Percentage Error, Median Absolute Error, Mean Squared Error and R squared;
    - Some visuals showing the performance of the model in a visual manner. For the Multi Layer Perceptron regressor the App returns also the Loss Curve graph, showing how the Loss function has been optimized throughout the epochs.
- Predict Custom Listings page, which gives the opportunity to the user to
    - Insert manually some data for a custom listing to then be used in the prediction process;
    - Draw a random observation from the test set and execute a prediction on that single observation using the chosen model.

The App enables the user to visualize in a quantitative manner the difference between the actual listing value and the predicted value (by looking at the cards). Moreover, a qualitative comparison between models is provided with the use of a True vs Predicted graph, where each prediction is plotted, given that predictions executed with different models are plotted in different colors. This visual can help to grasp the goodness of the model in *keeping the predictions close to the 45 degrees line*, that is the Perfect Prediction scenario.

## What to run

In order to reproduce from scratch the project, it's suggested to create a python environment with the use of Virtualenv and to install the required libraries and dependencies contained in requirements.txt.
Given this preparatory step, having the data source files at hand and the custom mappings, the user can safely jump to running the Streamlit app by navigating to the project root folder and running the following command:

*python -m streamlit run app/Homepage.py*

This command will spin up a local Streamlit Server, safely accessible from any web browser by searching for the following URL

*http://localhost:8501*

Overall, the user can reproduce any result by running the jupyter notebooks in alphabetical order. Therefore starting from the 000_dataset_creation.ipynb and so on. The execution of the notebooks will save (and overwrite) each intermediate dataset and each pkl object provided by this initial submission.
For the sake of simplicity, the user is provided with the trained models (inside the pickle folder) and can safely jump to the prediction process directly. Therefore, by running the predictions within the Streamlit App - or with the 020_from_saved_models.ipynb notebook - the results should come up in a faster manner.