SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER, CONTROL, AND MANAGEMENT
ENGINEERING ANTONIO RUBERTI

Comparative Study of Classification Algorithms on Rome Rent Prices Dataset

MACHINE LEARNING

Instructor:
Federico Fusco
Fabio Patrizi

Students:
Antonio Turco
1986183
Damiano Spadaccini
1986173

Academic Year 2025/2026

Contents

1 Introduction

1.1 Motivation

1.2 Objectives

The goal of this project is to develop an understanding of foundational supervised learning algorithms by implementing, analyzing, and comparing multiple models on a real-world dataset. Specifically, we aim to:

- Implement and train various classification algorithms
- Evaluate model performance using multiple metrics
- Compare and contrast different approaches
- Analyze the trade-offs between model complexity and performance

1.3 Report Structure

During this assignment, the SEMMA methodology will be followed:

> According to (Moine et al., 2011), the SEMMA methodology has five basic phases: Sample, Explore, Modify, Model and Assess. From (Olson and Delen, 2008) SEMMA facilitates the statistical exploration, the visualization techniques and the selection and transforming of the relevant variables in prediction, also can model the variables for prediction processes and later validate the precision of the model.

This report is organized as follows:

- Section 2: Description of the dataset and preprocessing steps
- Section 3: Methodology and model descriptions
- Section 4: Experimental results and evaluation
- Section 5: Comparative analysis and discussion
- Section 6: Conclusions and future work

2 Dataset Description

2.1 Dataset Selection

The dataset is taken from Kaggle and was uploaded 2 years ago by user Tommaso Ramella. It contains data scraped from the website Immobiliare.it about housing announcements in Italy in the year 2023. The author provides the public with two different datasets: the first is about rentals and the second contains purchase listings. Two different versions of each are available, a raw one, consisting of the original data, and a clean one. On the page, the author states the intention to provide a notebook with the methodology applied to parse and clean the data, but we were not able to find the complete version of this notebook (a part of the process can be found in the data publisher's GitHub folder) and as such we assume it was never released. While at the start we were inclined to just use the clean version for rents, in the end we decided against it and tried to gain more insight into the data by doing the work ourselves and trying to get as many features as we could.

The raw dataset contains around 126000 entries. Due to the number of elements in the original set, it was decided to apply machine learning techniques to a smaller subset of interest; that is, the rents in the city of Rome, a sore spot for students and a field with very practical ramifications and consequences. Our objective was to create a way for someone to get a basic understanding of the price a house should have given its characteristics, to avoid being fooled.

The records in the dataset related to Rome are 13276, with 47 columns of information for each of them; however some of those are very sparse and do not contain values for the majority of the rows. By sampling the data, it was possible to quickly determine some important underlying structure that could cause problems if not addressed properly: most importantly, we were able to find fields that contained null values (where to this invalid value could not be quickly associated some meaning) and columns with extraordinary outliers. Particularly, real estate listings for commercial properties (for example, offices) created some significant outliers in the price distribution, with some listings being orders of magnitude more expensive than the average rent for a house.

2.2 Data Preprocessing

Most features are categorical, and a small subset is instead numerical; most notably, "prezzo" (price), "stanze" (rooms), and "m2" (square meters). The price feature has been chosen for regression tasks: as such, we will not be using it to do any kind of feature augmentation, in order to avoid data leakage. Our hypothesis is that some features will have a very strong impact: for example, the surface area, the floor or whether the house is an apartment or not. Initially there was skepticism

about the importance of the "quartiere" feature, but in the end we were able to see some interesting results. However, some of this information is not represented as machine-readable data or explicitly codified in the raw database and it was necessary to process it in order to get a dataframe where it was possible to apply the models provided by our Machine Learning framework of choice, scikit-learn.

- **prezzo** → prezzo (float)
  - Removed "/mese" suffix and special characters ("€" symbol)
  - Converted to numeric values
- **stanze** → rooms, more_than_5_rooms
  - Converted to integer; "5+" becomes 5 with flag
  - Boolean indicator for properties with 5+ rooms
- **m2** → m2 (float)
  - Removed "m^2" unit suffix
  - Converted to numeric
- **bagni** → bathrooms, more_than_3_bathrooms, bathrooms_per_locali
  - "3+" becomes 3 with boolean flag
  - Ratio calculated: bathrooms / total_rooms
- **piano** → floor, ascensore, accesso_disabili, piano_rialzato, multi_floor, totale_piani_edificio, ultimo_piano
  - Extracted floor number (ground=0, basement=-1)
  - Boolean flags for elevator, disability access, top floor status
- **contratto** → affitto, affitto_libero, affitto_concordato, affitto_transitorio, affitto_studenti, affitto_riscatto, immobile_a_reddito, affitto_durata_minima, affitto_durata_rinnovo
  - Multiple boolean columns for contract types
  - Regex extraction for lease duration (e.g., "3+2" format)
- **locali** → totale_locali, camere_da_letto, altri_locali, tipo_cucina, campo_da_tennis
  - Parsed room counts and kitchen type
  - One-hot encoded kitchen types (cucina_ columns)
- **Posti Auto** → garage_box, esterno, parcheggio_comune, box_privato, has_garage_box, has_esterno, has_parcheggio_comune, has_box_privato

  - Regex parsing for parking types
  - Boolean indicators for presence of each type

- **stato** → stato_condition, stato_renovation + one-hot encoded
  - Split "Condition / Renovation" format
  - Created stato_condition_ and stato_renovation_ columns
- **tipologia** → 22 boolean columns (appartamento, attico, villa_unifamiliare, etc.)
  - One-hot encoded property types
- **disponibilità** → disponibilita (boolean)
  - True if contains "libero" (available)
- **anno di costruzione** → anno_di_costruzione (int)
  - Converted year to integer
- **riscaldamento** → 16 boolean columns (riscaldamento_autonomo, riscaldamento_metano, etc.)
  - Parsed heating system type and energy source
- **Climatizzatore** → 6 boolean columns (climatizzatore_autonomo, climatizzatore_freddo, etc.)
  - System type and cooling/heating capabilities
- **Efficienza energetica** → efficienza_classe, efficienza_classe_numerica, efficienza_consumo_kwh
  - Extracted energy class (A–G) and consumption (kWh/m^2/year)
- **altre caratteristiche** → 31 boolean columns (terrazza, ascensore, piscina, etc.)
  - Feature detection via string matching
- **description** → 11 columns (metro, stazione, universita, ospedale, parco, doccia, vasca, luminoso, silenzioso, guardaroba, mercato)
  - Binary indicators for nearby amenities mentioned in text
- **quartiere** → One-hot encoded (quartiere_ columns)
  - Neighborhood categories
- **spese condominio** → spese_condominio (float)

  - Handled “N.D.”, “nessuna”, etc. as 0.0
  - Converted to numeric

- **cauzione** → cauzione (float)
  - Removed currency symbols and converted to numeric

6 Conclusions

6.1 Summary of Findings

6.2 Key Takeaways

6.3 Limitations

6.4 Future Work