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 **3**
1.1 Motivation . . . 3
1.2 Objectives . . . 3
1.3 Report Structure . . . 3

2 Dataset Description **4**
2.1 Dataset Selection . . . 4
2.2 Data Preprocessing . . . 4
2.2.1 Data Quality Assessment . . . 4
2.2.2 Handling Missing or Noisy Data . . . 5
2.2.3 Complex Feature Extraction . . . 6
2.2.4 Boolean Feature Engineering . . . 7
2.2.5 Adding Geographic Information . . . 7
2.2.6 Derived Metrics . . . 9
2.2.7 Data Validation and Quality Metrics . . . 10
2.2.8 Technical Implementation . . . 10
2.2.9 Limitations and Future Improvements . . . 10
2.2.10 Feature Normalization . . . 12
2.2.11 Data Splitting . . . 12
2.2.12 Feature Engineering (Optional) . . . 12

3 Methodology **12**
3.1 Models Implemented . . . 12
3.1.1 Naïve Bayes . . . 12
3.1.2 Logistic Regression . . . 12
3.1.3 Softmax Regression (Optional) . . . 12
3.1.4 Decision Tree . . . 12
3.1.5 Random Forest . . . 12
3.1.6 Support Vector Machine . . . 12
3.2 Hyperparameter Tuning . . . 12

4 Results **12**
4.1 Model Performance . . . 12
4.2 Confusion Matrices . . . 12
4.3 ROC Curves and AUC . . . 12
4.4 Training vs. Validation Performance . . . 12
4.5 Computational Cost (Optional) . . . 12

5 Comparative Analysis **12**
5.1 Best Performing Models 12
5.2 Model Assumptions and Performance 12
5.3 Overfitting Trade-off 12
5.4 Visualizations 12
5.4.1 Learning Curves 12
5.4.2 Decision Boundaries 12
5.4.3 Feature Importance 12

6 Conclusions **13**
6.1 Summary of Findings 13
6.2 Key Takeaways 13
6.3 Limitations 13
6.4 Future Work 13

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 (`rome_rents_raw.csv`) are 13276, with 38 columns of information for each of them.

2.2 Data Preprocessing

This section outlines the preprocessing pipeline implemented to transform raw rental listings into a structured dataset suitable for analysis and modeling.

When we first looked at the raw data, we immediately noticed several issues typical of information scraped from real estate websites. The listings weren't standardized—which makes sense, since they were originally posted by different landlords with their own writing styles and conventions.

Numeric fields were particularly messy: prices included currency symbols and "/mese" suffixes, surface areas had "m^2" attached to them, and some values had strange encoding artifacts like `\x80` instead of the Euro symbol. Categorical information was scattered across Italian and sometimes mixed terminology, making it difficult to group similar entries together.

We also found interesting notation patterns that required careful handling. For instance, landlords often used expressions like "5+" or "3+" to indicate properties with many rooms or bathrooms, rather than listing the exact number. Floor information

proved especially tricky—descriptions ranged from straightforward numbers to Italian terms like "piano terra" (ground floor) or "seminterrato" (basement), sometimes combined with additional details about elevators or accessibility features.

Perhaps most challenging were the missing or ambiguous values scattered throughout the dataset. Critical fields like floor information and contract details were often incomplete or expressed in ways that needed interpretation rather than direct extraction.

The first step in the preprocessing pipeline involves standardizing the dataset structure. All column headers in the raw dataset, originally in Italian, are renamed to English to ensure consistency and readability throughout the analysis. For example, `prezzo` is renamed to `price`, `spese condominio` becomes `condo_fees`, `bagni` becomes `bathrooms`, and `quartiere` is mapped to `neighborhood`.

Following this translation, specific cleaning operations are performed on each column to handle missing values and remove noise.

2.2.1 Handling Missing or Noisy Data

Raw data from rental listings often contains inconsistencies, currency symbols, and non-numeric text in numeric fields. The following logic was applied to clean specific columns:

- **Price (`price`)**: The columns are cleaned by stripping non-numeric artifacts such as currency symbols (€) and monthly indicators (/mese). The resulting clean strings are converted into floating-point numbers.
- **Condo Fees (`condo_fees`):** This column often contains various text indicators for "zero costs" (e.g., "nessuna", "n.d.", "nil", "zero"). These text values are explicitly mapped to 0.0, while valid numeric entries are cleaned and parsed as floats.
- **Surface Area (`square_meters`):** The unit suffix (m^2) is removed from the values to isolate the numeric surface area, which is then converted to a float.
- **Deposit (`deposit`):** Similar to the price column, currency symbols and formatting characters are removed to convert the deposit amount into a numerical value.

Some columns contain mixed notations indicating capacity or size limits which require splitting into multiple features:

- **Rooms (`rooms`):** Listings often use notation like "5+" to denote large properties. This is handled by extracting the base integer (e.g., 5) for the main numeric column and creating a separate boolean flag (e.g., more_than_5_rooms) to capture the "greater than" information without corrupting the integer data type.

- **Bathrooms (`bathrooms`):** Similar to rooms, "3+" notations are parsed by extracting the integer 3 and creating a corresponding boolean flag (more_than_3_bathrooms).
- **Floor Level (`floor`):** The high variability in floor descriptions is normalized by mapping semantic Italian descriptions to integers. For instance, "piano terra" (ground floor) and "piano rialzato" (raised floor) are mapped to 0, while "seminterrato" (basement) is mapped to -1. Standard numeric floors (e.g., "1° piano") are parsed directly to their integer equivalents.

2.2.2 Complex Feature Extraction

Beyond the basic cleaning operations, several columns contained rich information that required more sophisticated parsing strategies to extract meaningful features from what initially appeared as unstructured text.

Consider the `floor` column: rather than simply extracting the floor number, we realized that the raw text often contained additional valuable information. For instance, descriptions like "3° piano con ascensore" or "piano terra senza barriere architettoniche" provide not just the floor level, but also accessibility features. We therefore implemented a parser that extracts multiple attributes from this field:

- Whether an elevator is present (`lift`)
- Accessibility for people with disabilities (`disabled_access`)
- Whether the property spans multiple floors (`multi_floor`)
- Whether it's located on the top floor of the building (`last_floor`)

Similarly, the `contract` field proved to be far more complex than anticipated. Italian rental contracts come in various forms with different legal implications, and landlords often specify minimum lease durations and renewal terms directly in their listings. Our parsing approach extracts:

- Minimum rental period (`rent_min_duration`), often expressed in formats like "3+2" meaning 3 years with a 2-year renewal option
- Renewal terms (`rent_renewal_duration`)
- Special contract categories such as student-specific leases, transitional contracts (typically shorter term), and controlled rent agreements ("canone concordato")

The `rooms_details` column presented another interesting challenge. Rather than a simple room count, listings typically describe the composition of the property—distinguishing between total rooms, actual bedrooms, and other spaces. We broke this down into:

- Total number of rooms (`room_total`)

- Number of bedrooms specifically (`bedrooms`)
- Count of other rooms like living rooms or studies (`other_rooms`)
- Kitchen type, which we further categorized since Italian listings distinguish between different kitchen configurations ("cucina abitabile," "cucinotto," etc.)

To ensure the model treats similar inputs identically, categorical and unstructured text features are transformed into consistent boolean or numeric formats. The `description` column, containing free-text descriptions of rental properties, is converted into a set of boolean features through keyword extraction. The text is scanned for high-value keywords indicating nearby amenities and property characteristics such as "metro" (subway access), "università" (university proximity), "ospedale" (hospital), "parco" (park), "luminoso" (bright), and "silenzioso" (quiet). Each keyword's presence in the description generates a corresponding binary indicator column, effectively transforming unstructured textual data into structured features that can be readily utilized by machine learning models. This approach captures semantic information about property location, accessibility, and desirable characteristics that would otherwise be lost in raw text format.

2.2.3 Boolean Feature Engineering

Multiple columns were transformed into comprehensive boolean feature sets:

Property Type: 22 distinct property type flags including condominium, villa types, luxury classifications, and ownership structures.

Heating Systems: 16 heating-related features covering system types (autonomous, centralized), distribution methods (radiators, floor), and energy sources (gas, solar, heat pump).

Air Conditioning: 6 features describing system configuration and capabilities.

Additional Amenities: 30+ binary features extracted from the `other_features` column, covering amenities from terraces to security systems.

2.2.4 Adding Geographic Information

Since only the neighborhood is provided, we decided to add some geographic information to help the model better understand the locations. For each neightborhood finded in the dataset, we added the following features: a boolean feature indicating if the location falls inside the GRA (Great Ring Road), a zone categorical feature indicating in which position of rome is collocated (center, north, south, east, west), and another categorical feature indicating the administrative zone (municipality). In this way we are able to give the model some more information about the position of the house in the city, which is a very important factor in determining the price of a rent. After

this step each raw has three more columns: Inside_GRA (boolean), zone (categorical), municipality (categorical).

- **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

2.2.5 Derived Metrics

Several composite metrics were calculated to enhance predictive power:

- `bathrooms_per_room`: Bathroom-to-room ratio as a luxury indicator
- Boolean flags for property size thresholds (`more_than_5_rooms`, `more_than_3_bathrooms`)
- Energy efficiency converted to both categorical and numerical representations

2.2.6 Data Validation and Quality Metrics

The cleaning pipeline includes validation checks and produces quality metrics:

- Currency symbol removal and numeric conversion success rates
- Geographic mapping coverage statistics
- Feature extraction completeness metrics
- Handling of edge cases (zero condo fees, special floor types)

Initial dataset: 13,276 records with 38 columns. The final cleaned dataset contains significantly more features due to the comprehensive one-hot encoding and boolean feature engineering applied during preprocessing.

2.2.7 Technical Implementation

The cleaning pipeline was implemented as a modular Python script with dedicated functions for each feature type. Key technical aspects include:

- Robust error handling with warning systems for conversion failures
- Support for both Italian and English terminology in text parsing
- Modular design allowing easy extension for additional features
- Preservation of original data semantics while ensuring machine-readability

2.2.8 Limitations and Future Improvements

- Geographic mapping depends on neighborhood name consistency
- Text parsing may miss nuanced descriptions or synonyms
- Some derived features may have sparse representation
- Temporal aspects (seasonality, market trends) not captured

2.2.9 Feature Normalization

2.2.10 Data Splitting

2.2.11 Feature Engineering (Optional)

3 Methodology

3.1 Models Implemented

3.1.1 Naïve Bayes

3.1.2 Logistic Regression

3.1.3 Softmax Regression (Optional)

3.1.4 Decision Tree

3.1.5 Random Forest

3.1.6 Support Vector Machine

3.2 Hyperparameter Tuning

4 Results

4.1 Model Performance

4.2 Confusion Matrices

4.3 ROC Curves and AUC

4.4 Training vs. Validation Performance

4.5 Computational Cost (Optional)

5 Comparative Analysis

5.1 Best Performing Models

5.2 Model Assumptions and Performance

5.3 Overfitting Trade-off

5.4 Visualizations

5.4.1 Learning Curves

5.4.2 Decision Boundaries

5.4.3 Feature Importance

6 Conclusions

6.1 Summary of Findings

6.2 Key Takeaways

6.3 Limitations

6.4 Future Work