

# EECS1516 Assignment 1: Data-Driven Apartment Evaluations

Due: February 27, 9 p.m.

## 1 Introduction

*This assignment is adapted from an original project developed for CS 61A at UC Berkeley. We thank the original authors and the City of Toronto for making the RentSafeTO data publicly available.*

In this assignment, you will analyze apartment building evaluations using data from the RentSafeTO program. RentSafeTO officers inspect apartment buildings every two years and assign ratings across several categories, including COSMETIC, MODERATE RISK, HIGH RISK, and OVERALL. Your goal will be to build tools that:

- Represent apartment buildings and their evaluations
- Extract useful statistics from buildings
- Predict overall building ratings using cosmetic ratings using both **heuristic-based** and **data-driven** prediction methods.

The `code.zip` archive contains starter code and a dataset that has been drawn from the city of Toronto's RentSafeTO database. You will modify the following files:

- `abstractions.py`
- `classifiers.py`

## 2 Phase 1: Apartment Building Abstraction (20%)

First, Complete the `ApartmentBuilding` data abstraction in the file `abstractions.py`. The constructor has the form:

```
def __init__(self, type: str, ward: int, year: int, reviews:
               list[Review]) -> None
```

The parameters are:

- `type`: a string one of TCHC, PRIVATE, or SOCIAL HOUSING
- `ward`: an integer ward number

- **year**: the year the building was constructed
- **reviews**: a list of **Review** objects for a given building

You must then implement the following class methods. Types of parameters and outputs are in the comments in your starter code.

- **num\_ratings**: to return the number of reviews for a given building.
- **apartment\_min\_rating**: to return the **minimum** rating of the building among all its reviews.
- **apartment\_overall\_rating**: to return the **overall** rating of the building.
- **apartment\_cosmetic\_rating**: to return the **cosmetic** rating of the building.

## 3 Phase 2: Making Predictions

Next, you will predict an apartment's overall ratings using the cosmetic score for that apartment. First, you will use a simple similarity-based approach to make predictions. Then, you will implement a prediction strategy that uses linear regression. Both implementations will be written in `classifiers.py`.

### 3.1 Nearest Neighbour Prediction (20%)

The constructor for the **NearestNeighbour** class has been written for you. This constructor binds example data (or training data) to a class attribute called **\_training\_data**. You will use this attribute to make predictions of overall apartment building scores. To do this, implement the following method for the **NearestNeighbour** class:

```
predict(apartment: Apartment)
```

This function should:

- Accept an input apartment whose overall rating is to be predicted
- Compare the input's **COSMETIC** rating to each **\_training\_data** apartment
- Locate the **\_training\_data** apartment whose **COSMETIC** rating is **closest** to the input apartment's **COSMETIC** rating.
- Return the **OVERALL** rating for that building. This will be your **prediction** as to the **OVERALL** rating of the input apartment.

Note that in the case of ties, you may return the overall rating of **any tied building**.

Then, in `classifiers.py` and in the **NearestNeighbour** class, implement:

```
make_predictions(apartments: list[ApartmentBuilding])
```

This function should:

- Accept an **list** of apartments.
- Predict the overall rating for each apartment in the list based on its **COSMETIC** score.
- Return a dictionary whose keys are apartments in the input list and whose values are predicted ratings.

### 3.2 Training a Regression Model (20%)

Implementing **least-squares linear regression** will require a little more work; first we must train a model and then we can use it to make predictions. To start, implement the following in the **LinearRegression** class of *classifiers.py*:

```
train(training_set: list[Apartment])
```

This will compute the best fit parameters  $a$  and  $b$  for the model:

$$y = a + bx$$

Where  $x$  is a building's **COSMETIC** rating and  $y$  it's **overall** rating. To solve for  $a$  and  $b$ , you will need to compute the following sums of squares using the apartment information in your **training\_set**:

$$\begin{aligned} S_{xx} &= \sum_{i=1}^m (x_i - \text{mean}(x))^2 \\ S_{yy} &= \sum_{i=1}^m (y_i - \text{mean}(y))^2 \\ S_{xy} &= \sum_{i=1}^m (x_i - \text{mean}(x)) * (y_i - \text{mean}(y)) \end{aligned}$$

Here,  $x_i$  denotes the **COSMETIC** rating for the  $i$ th building in the training set and  $y_i$  is its overall rating;  $m$  is size of the training set. Parameters  $a$  and  $b$  can then be calculated as follows:

$$\begin{aligned} b &= S_{xy} / S_{xx} \\ a &= \text{mean}(y) - b * \text{mean}(x) \end{aligned}$$

Before you are done, you are also asked to calculate  $r^2$ , which reflects how **well** your model fits the training data:

$$r^2 = S_{xy}^2 / (S_{xx} * S_{yy})$$

### 3.3 Least Squares Prediction (20%)

We are now ready to implement the class method **predict**. This will accept an apartment building and return a predicted overall rating using your model. Note that this method may only be called **after** you have completed training.

In addition, in *classifiers.py* and in the **LinearRegression** class, you are asked to implement:

```
make_predictions(apartments: list[ApartmentBuilding])
```

This function:

- Accepts an input **list** of apartments.
- Predicts an overall rating for each apartment based on its **COSMETIC** score and using the model you have trained.
- Returns a dictionary; keys are apartments in the input list and values are predicted ratings.

## 4 Phase 3: Evaluating Predictions (20%)

Only one more function to implement! This one is:

```
calculate_mse(predictions: list[float], true_values: list[float]).
```

This function accepts lists of predicted and corresponding ground truth ratings and it returns the mean squared error (MSE) of the predictions. Calculate MSE as follows:

$$MSE = SSE/n$$

Here,  $n$  is the number of predictions that you have made and  $SSE$  is the sum of squared errors across all predictions. You can calculate the  $SSE$  as follows:

$$SSE = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

In this equation:

- $y_i$  is the **ground truth** rating of the  $i$ -th apartment.
- $\hat{y}_i$  is your **predicted** rating for the  $i$ -th apartment.
- $n$  is the **total number** of predictions made.

## 5 Testing

Throughout this project, you should be testing the correctness of your code. We have provided some tests and will mark your assignment with additional tests. Our tests are not complete; you are expected to test your own code as well!

---

Congratulations! You've now implemented some basic strategies to predict ratings of apartments based on their cosmetic scores. And you did this in an object oriented way!

**Have fun and GOOD LUCK!!**