**Improving a Housing Rent Machine Learning Model’s Performance using Feature Engineering and Data Re-Encoding Techniques**

**Harlan Dy**

**3CSA – Advanced Intelligent Systems**

**Introduction:**

Housing in India has boomed in recent years due to the rise in the population size of the country. Many types of properties have been in showcase from maharajas and apartment buildings to condominiums and simple, primitive huts. Real-estate companies have tried to venture out into India to try and capitalize in the housing boom. One of the ways we can see this is in the rapid development of Machine Learning Models that can estimate the price of housing from current and historic data.

**Methodology:**

The dataset consists of 4,747 entries with the filename of “House\_Rent\_Dataset.csv”. It was divided into 10 categories which determines the features of each real estate property on various cities in India. Categories such as the Size of the area, Number of Bedrooms, Hall, and Kitchens, etc. For the Purposes of Experimentation, four out of eleven features of each of the houses were initially used in the training of the Regression Model. These Features are Size, Area Type, Furnishing Status, and City. The rationale behind using these features is their ease of re-encoding. In the case of Size, having a larger area for available for rent, can be beneficial for the tenant. Mainly due to business opportunities that could arise from having a large size. At the same time, it is also beneficial for the landlord, as they can charge more for their property. Meanwhile, for the other chosen features, it mainly boils down to the lack of outlier values they possess.

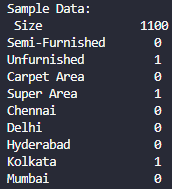
However, as the process came along, the chosen features above were either added, removed, or altered as seen in the section.

**Experiments:**

The first modification as seen above was change the features that are retained during the pre-processing step. Initially, for the first model, BHK, Bathroom count, Furnishing Status, and Rent were used as the features of the first model. Furnishing Status was re-encoded using One-Hot encoding in order to obtain the single bit representation of each furnishing status category. 20% of the data was set aside for testing and afterwards, it was standardized to normalize the distribution of the data. Using these features, we have achieved the following Mean Squared Error and Coefficient of determination r^2:



Afterwards, Forecasting was done with the following Data Point:



The results we’re unspectacular to say the least as seen below:



Despite this, we will use these values as a baseline for future experiments and tweaks to our model.

The first tweak done to the model was to change the features that will be used to build the model. As stated above: Size, Area Type, Furnishing Status, and City we’re used as the new features of the model. The only other change done was to also re-encode Area Type and City to One-Hot encoding. Besides that, no other tweaks we’re made in splitting the data or anything else. Despite that, we have achieved a somewhat significant improvement as seen below:

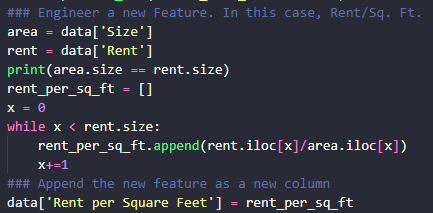


The mean squared error is still quite big, meaning our model still is not very accurate the data, but we we’re able to shave off a modest 793,573,577.45 over our loss, while improving the coefficient of determination by .20 points.



Afterwards, engineering a new feature was done. In this case, computing the value of the rent per square feet. As one of the things I have noticed was that the current rent values were not being considered by the model creation process. The rent per square feet is simply done by the ff. formula:

And was implemented with the ff. code snippet below:

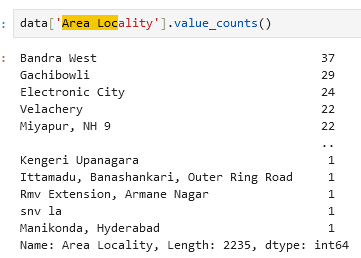


Where the Size and Rent columns were isolated from the dataset, and iterated over when doing the computations. The full prototype code is located in the file “FeatureEngineeringTest.ipynb”

Doing this reduced the mean squared error by a further 478,097,999.26 and increased the fit from 0.52 to 0.64. However, using the new model with revised weights barely improved the forecasted rent of the example:



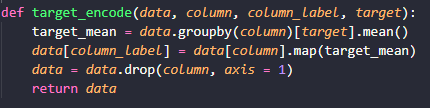
The next improvements done were to include two features: The Floor where the Property is located, and the Specific Part of the City it is located. Unlike the prior features, both of these have a large number of categories that make it hard for us to re encode the data using one-hot encoding as seen below:



As re-encoding it to One-hot encoding will make the table too large for us to traverse. As each category of the area locality will have its own column. The same thing could also be said with the floor number the house is located, it makes the table too large to be traversed and explored easily, and it wastes too much memory, as some localities are sparsely populated with viable housing.

One way to solve this was to re-encode these two columns using Target Encoding, where the values are replaced with the average of the target value (Actual Rent) with respect to the its relationship with the selected feature.

In other words, it takes the average target value of the rows that showcase the same features that are selected. In our case from the data above, there are 37 houses that are located in Bandra West. We take the average rent of all those houses and we replace the column with the average rent in that area. Unfortunately, Target Encoding can result in overfitting, as the actual rent value can “leak” when doing target encoding. But due to the time constraints of this exercise, I wasn’t able to optimize the Target encoding. This was the function used in order to re-encode the data:





Lastly, I added two more features to the model: BHK, or Bedrooms, Hallways, and Kitchens, and the Number of Bathrooms.

There was also another failed experiment where I tried to include the date the House Listing was posted, where the date value was re-encoded using the seconds elapsed from the Unix Epoch from: 1/1/1970. Which is a common way to encode time to encode time on Unix, and Unix-like systems. But sadly, it didn’t really affect the performance too much. Once again, the full prototype code is located in the file “FeatureEngineeringTest.ipynb”

**Results and Analysis:**

In order to evaluate the accuracy of the model within the time-constraints provided, we need to not only input the testing data, but also to the training data into the model, in order to measure its performance. That way, we can see if there is some overfitting between one dataset or another. The code is located at Cell 10 of the Final Notebook named: “IS2-LabExercise1.ipynb”. The two metrics used were the MSE or Mean Squared error for our loss function, and the Coefficient of determination R^2, which is the value ranging from 0 to 1 that tells us how accurately the model can predict results. The training set is split with a ratio of 20/80, or 20% for the data set, and 80% for the training set. With a random\_state value of 42. It is trained using Scikit-Learn’s Linear Regression Function.

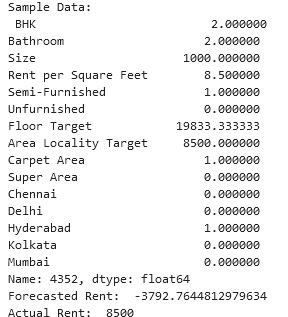


This is the MSE and R^2 values of the model when tested against the test set. After all the changes were made, we ended up with a score of 1,297,028,467.39. Ideally, we’d want the MSE to be smaller, but my educated guess is that due to the number of weights present in the model from the features we used, the number is unlikely to go near zero. But it is still a huge improvement nonetheless from our baseline of 2,693,550,407.36. Our R^2 meanwhile has more than doubled, from a baseline of 0.32 to 0.67.



Meanwhile this is our MSE and R^2 when the model is tested against the training set. Unfortunately, the MSE is almost a billion higher than the test set, but the R^2 is around the same. From my observation, the model doesn’t seem to overfit itself on one data set or another. But I can definitely say now that more improvements could be made if time-constraints were improved.

Doing a random Qualitative Evaluation on the current iteration of the model, there was one data-point that had a negative forecast value, Index #4352.



There could be more, but I was unable to check given the time constraints.

**Conclusion and Recommendations:**

After a lot of trial and experiments, were able to reduce the error produced by the model and improve its performance by approximately 2x. The final Features included in the model in determining the value of the House of Rent are:

* BHK: Number of Bedrooms, Hall, Kitchen.
* Size: Size of the Properties in Square Feet.
* Floor: Which floor the property is situated in and the Total Number of Floors of the building.
* Area Type: Size of the properties calculated on either Super Area or Carpet Area or Build Area.
* Area Locality: Locality of the properties.
* City: City where the properties are Located.
* Furnishing Status: Furnishing Status of the properties. Whether it is Furnished, Semi-Furnished or Unfurnished.
* Bathroom: Number of Bathrooms.

We were able to use other Encoding techniques such as One-hot encoding, in order to represent nominal data in a way that doesn’t confuse the Model with possible non-existent relationships, and Target Encoding, which nets us the average rent with respect to a selected feature.

Not only that, when testing the model using both the training and test datasets derived from the master set using a 20/80 split, the machine was able to generalize its predictions against the two. And not overfitting itself on one model or the other. In the future, in order to truly the gauge the performance of this model, a brand-new dataset should be used using data not derived from the current training and testing sets. This should allow us how well it performs on real world conditions and brand-new data.

For future researchers of this problem, other parameters could be adjusted such as how the data is split, and the seed used from the value of 42. Future researchers should explore more ways to represent certain features like where which Floor the property is situated. Others include the use of Regularization techniques and possibly include smoothing in the data done in Target Encoding. Another is the use of graphs to present how well the model performs during testing instead of just relying on MSE and R^2 metrics. Lastly, more ways of pre-processing the data should be considered in the future, if time-constraints were to allow it.

To view the full code repository, including the raw python code and the notebooks used to explore the data, visited the following link: https://github.com/dyharlan/IS2-LabExercise1