**Paper Title (Something Interesting)**

**Find a relevant dataset from Kaggle.**

**Multiple Linear Regression using R and Python**

Name, [student@bellarmine.edu](mailto:student@bellarmine.edu)

Name, [student@bellarmine.edu](mailto:student@bellarmine.edu)

**ABSTRACT**

This project focuses on predicting housing process using multiple linear regression models. The dataset includes various features such as area, bedrooms, bathrooms, stories, air conditioning, and 6 other relevant variables. The target variable for the regression model is price. The objective of this model is to determine which factors have the most significant impact on housing prices and to build an accurate prediction for housing prices.

1. **INTRODUCTION**

This data set contains 13 different variables – price, area, bedrooms, bathrooms, stories, mainroad, guestroom basement, hotwaterheating, airconditioning, parking, prefarea, and furnishingstatus. These variables contain no null values and have 545 entries. We chose this dataset because it contained a clear variable to predict (price) and it was large enough to build a reliable prediction model on. This project is built both with python and R to gain knowledge in both fields.

1. **BACKGROUND**
   1. *Data Set Description*

The “Housing Prices” dataset was collected by M Yasser H who wanted to emphasize the complexities of the housing market. The objective of this study was to analyze and predict housing prices based on certain factors like house area, bedrooms, furnished, nearness to mainroad, etc. The dataset is small, yet its complexity arises due to the fact that it has strong multicollinearity. This dataset was built and gathered from two outside sources.

* 1. *Machine Learning Model*

Multiple Linear Regression is a statistical method used to analyze the relationship between a dependent variable and multiple independent variables. In this example, the dependent variable is ‘price’ and the independent variables are all of the other features. MLR aims to identify how the independent variables affect the dependent variable. The model works by identifying the coefficients for each independent variable, which determine the magnitude and direction of its impact on the corresponding dependent variable. The coefficients are measured through a process called least squares, which minimizes the sum of squared errors between the predicted and actual values of the dependent variable.

1. **EXPLORATORY ANALYSIS**

This dataset contains 575 samples with 13 columns with various data types.

**Table 1: Data Types**

|  |  |
| --- | --- |
| *Variable Name* | *Data Type* |
| V1 price | Int64 |
| V2 area | Int64 |
| V3 bedrooms | Int64 |
| V4 bathrooms | Int64 |
| V5 stories | Int64 |
| V6 mainroad | Object |
| V7 basement | Object |
| V8 hotwaterheating | Object |
| V9 airconditioning | Object |
| V10 parking | Int64 |
| V11 prefarea | Object |
| V12 guestroom | Object |
| V13 furnishingstatus | Object |

1. **METHODS**
   1. *Data Preparation*

The data contained no null values, so we did not need to impute any missing values. We then split the data into dependent and independent variables on order to split them up into training and test sets later. The next thing we did was create dummy variables for every object that is listed above. This was done with pandas get\_dummies function.

* 1. *Experimental Design*

Table X: Experiment Parameters

|  |  |
| --- | --- |
| **Experiment Number** | **Parameters** |
| 1 | All ten (10) raw features with 80/20 split for train and test  **Python Results**  Accuracy – 91%  MSE – 45720769.76  RMSE 6761.71  **R Results**  R-Squared – 0.9114  MSE – 45616484.32  RSME – 6753.997 |
| 2 | All ten (10) raw features with a 70/30 split for train and test  **Python Results**  Accuracy – 91%  MSE – 45560816.27  RMSE 6749.88  **R Results**  R-Squared – 0.9114  MSE – 45616484.32  RSME – 6753.997 |
| 3 | All ten (10) raw features with 60/40 split for train and test  **Python Results**  Accuracy – 91%  MSE – 45536092.63  RMSE 6748.04  **R Results**  R-Squared – 0.9114  MSE – 45616484.32  RSME – 6753.997 |

* 1. *Tools Used*

The following tool were used for this analysis: Python running in an Anaconda environment and running with Python 3.9.12, Pandas for working with the data, sci-kit-learn to create our MLR model and give us metrics on the model. An R environment using the tidyverse and caTools libraries.

1. **RESULTS**
   1. *Mean square Error and R-Square calculation*

MSE is the mean of the ((actual datapoints minus the predicted datapoints) squared). MSE result gives an average of how far each datapoint is from the regression line. The larger the MSE the less that the regression line fits the dataset, while the closer to zero the MSE the more that the regression line follows the dataset.

R squared is 1 – (sum or squares of residuals divided by the total sum of squares). R-squared indicates how much variation in the dependent variable is explained by an independent variable in the linear regression model. The higher the R-squared value the more that the dependent variable is explained by independent variables, which we saw in that all but two categories were significant in predicting the dependent variable.

* 1. *Discussion of Results*

The best classification for Python was the 70/30 split because that was our lowest MSE, meaning that it was the closest to having the regression line hitting or getting close to most of the data points. The worst for python was the 80/20 split with the highest MSE. In R all the splits produced the same MSE, RMSE, and R-Sqrd values.

* 1. *Problems Encountered*

One problem was that the dataset was too large to implement the model on. We dropped a lot of values after we encoded the variables so that it made the data easier to work with, especially the “no” values. With the R program there was also a problem running the validation model.

* 1. *Limitations of Implementation*

Because the model is linear, it can create a line the best fits the dataset points but there is a possibility that the dataset is a non-linear function. I model that could work better is the non-linear regression analysis because the regression line follows the dataset closer to produce a curved line.

* 1. *Improvements/Future Work*

In future work, I would like to see if the non-linear model would produce a better regression line. I would also like to be able to use different data sets that can produce more accurate predictions.

1. **CONCLUSION**

While there were issues with the size of the program and validating the model, that linear regression model slightly worked for this dataset. The best slip for python was the 70/30 and for R the results we the same with each split. The model showed that the independent variables did explain the dependent variable. Yet the linear regression line was not the best for following the data points, this could be because of how large the dataset is or that a non-linear model would follow the dataset better. In conclusion this model needs some more work and may not be the best match for this dataset.

**REFERENCES**

List any websites, books, articles, etc. that you found useful while you worked on this project. It is not necessary to cite the references in the paper unless you specifically mention it in the text.