Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

Fall 2020

Artificial Intelligence Lab

CSE 4108

Report on
Machine Learning Project

Submitted To

Md. Siam Ansary
Lecturer
Dept. of CSE, AUST

Submitted By

Abrar Rafid Noor
17.02.04.059
Lab Group: B1

Date of Submission: 9 September, 2021

Contents

Report on Machine Learning Project

Abrar Rafid Noor

September 2021

1 Brief Description of the Problem

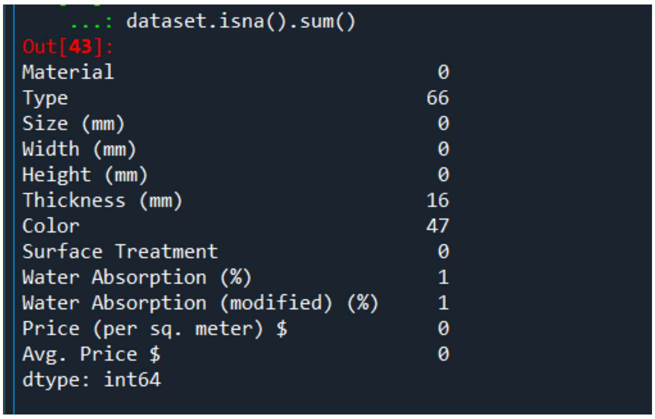
A dataset need to be developed on a particular topic and apply ML models and need to do comparative analysis.
The topic selected is “Ceramic Tiles Price Prediction”. It is a regression problem. Here, we will predict prices of ceramic tiles from a prepared dataset with multiple machine learning models (used for regression) and compare their performance .

2 Brief Description of the Dataset

The dataset’s name is ”Ceramic Tiles Dataset”. It has been made from Foshan Urban Ceramics Co Ltd’s website on ceramic tiles. It consists of total 12 features (columns) including both dependent and independent variables and 100 cases/samples (rows), consisting of both categorical and numerical values as well as missing values.

3 Data Preprocessing

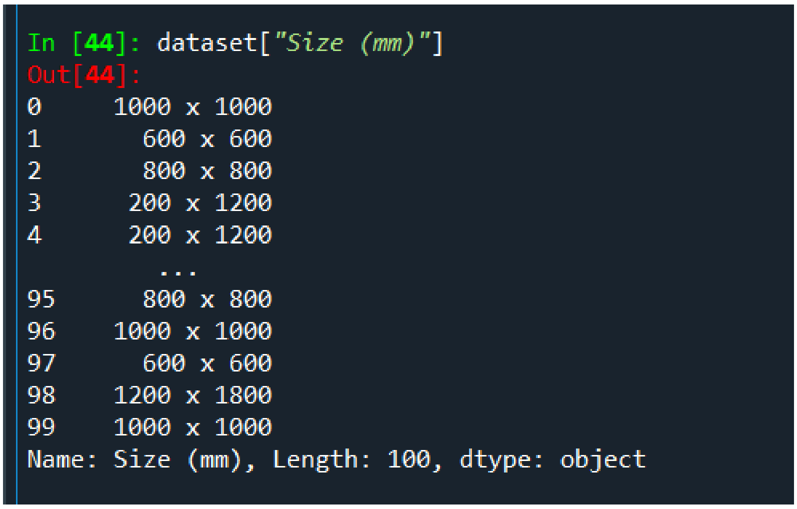
As mentioned in the previous section, the dataset contains missing values, which is not suitable for our training models. Here we can see attributes “Type” and “Color” has 66 and 47 missing values out of 100. Which is not preferable.



```
   ...: dataset.isna().sum()
Out[43]:
Material                           0
Type                              66
Size (mm)                          0
Width (mm)                         0
Height (mm)                        0
Thickness (mm)                    16
Color                             47
Surface Treatment                  0
Water Absorption (%)               1
Water Absorption (modified) (%)    1
Price (per sq. meter) $            0
Avg. Price $                       0
dtype: int64
```

Figure 1: Number of missing values per features

And attribute “Size (mm)” has each tile’s size in a format: “width x height”, but width and height already have their own separate column. So, column “Size (mm)” is also not necessary.



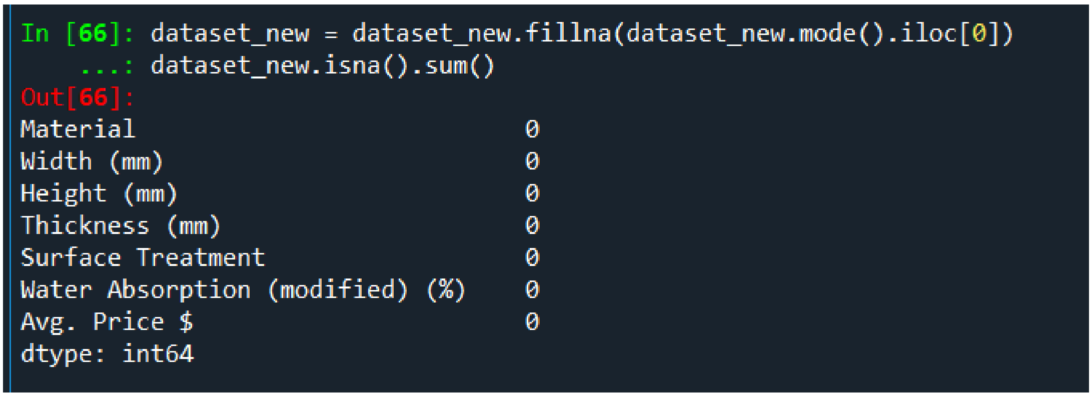
```
In [44]: dataset["Size (mm)"]
Out[44]:
0     1000 x 1000
1       600 x 600
2       800 x 800
3      200 x 1200
4      200 x 1200
         ...
95      800 x 800
96    1000 x 1000
97      600 x 600
98    1200 x 1800
99    1000 x 1000
Name: Size (mm), Length: 100, dtype: object
```

Figure 2: ”Size” feature’s values

For these reasons, a new dataframe has been created where these three columns- “Type”, “Color” and “Size (mm)” have been excluded.

```
dataset_new = dataset.iloc[:, [0,3,4,5,7,9,11]]
```

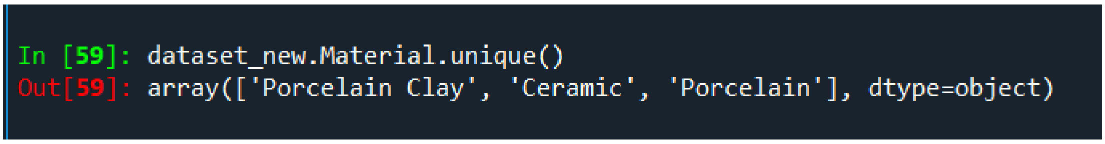
After that the attributes, who had fewer missing values - have been replaced by the most frequent occurring value of that attribute. And with this, missing values problem of the dataset has been dealt.



```
In [66]: dataset_new = dataset_new.fillna(dataset_new.mode().iloc[0])
    ...: dataset_new.isna().sum()
Out[66]:
Material                          0
Width (mm)                        0
Height (mm)                       0
Thickness (mm)                    0
Surface Treatment                 0
Water Absorption (modified) (%)   0
Avg. Price $                      0
dtype: int64
```

Figure 3: Number of missing values per features after filling na values

In the “Material” attribute, among the unique values “Porcelain Clay” and “Porcelain” mean the same material.



```
In [59]: dataset_new.Material.unique()
Out[59]: array(['Porcelain Clay', 'Ceramic', 'Porcelain'], dtype=object)
```

Figure 4: ”Material” feature’s unique values

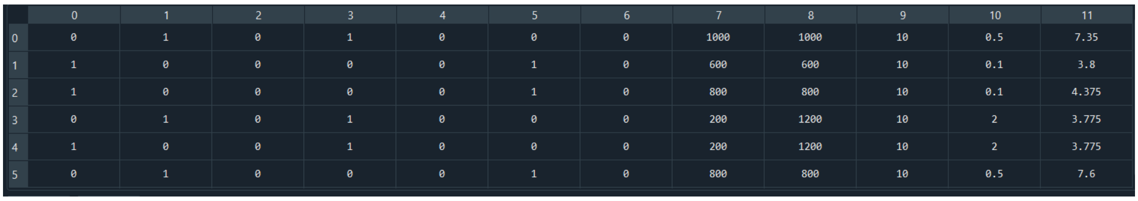
So, “Porcelain Clay” has been replaced with “Porcelain”.

```
dataset_new['Material'] = dataset_new['Material'].replace(['Porcelain Clay']
,['Porcelain'])
```

Amongst the features, “Material” and “Surface Treatment” are of categorical values. So, they have been encoded to numerical value using ColumnTransformer and OneHotEncoder.

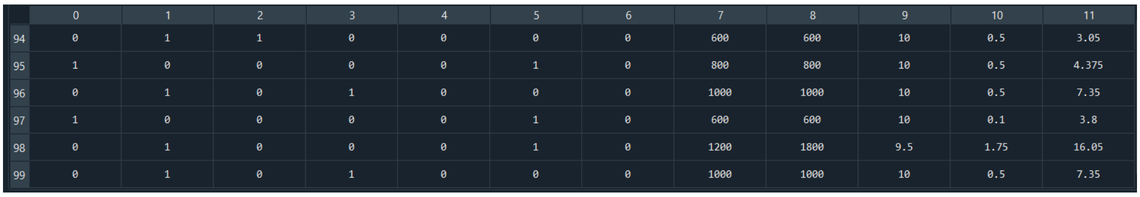
```
ct = ColumnTransformer([('Ceramic Tiles Dataset', OneHotEncoder(), [0, 4])],
remainder = 'passthrough')
dataset_new = ct.fit_transform(dataset_new)
```

Now our dataset consists of numerical values only.



	0	1	2	3	4	5	6	7	8	9	10	11
0	0	1	0	1	0	0	0	1000	1000	10	0.5	7.35
1	1	0	0	0	0	1	0	600	600	10	0.1	3.8
2	1	0	0	0	0	1	0	800	800	10	0.1	4.375
3	0	1	0	1	0	0	0	200	1200	10	2	3.775
4	1	0	0	1	0	0	0	200	1200	10	2	3.775
5	0	1	0	0	0	1	0	800	800	10	0.5	7.6

Figure 5: Cases 1 to 6 of the dataset



	0	1	2	3	4	5	6	7	8	9	10	11
94	0	1	1	0	0	0	0	600	600	10	0.5	3.05
95	1	0	0	0	0	1	0	800	800	10	0.5	4.375
96	0	1	0	1	0	0	0	1000	1000	10	0.5	7.35
97	1	0	0	0	0	1	0	600	600	10	0.1	3.8
98	0	1	0	0	0	1	0	1200	1800	9.5	1.75	16.05
99	0	1	0	1	0	0	0	1000	1000	10	0.5	7.35

Figure 6: Cases 95 to 100 of the dataset

Plotting heatmap to correlate the features.

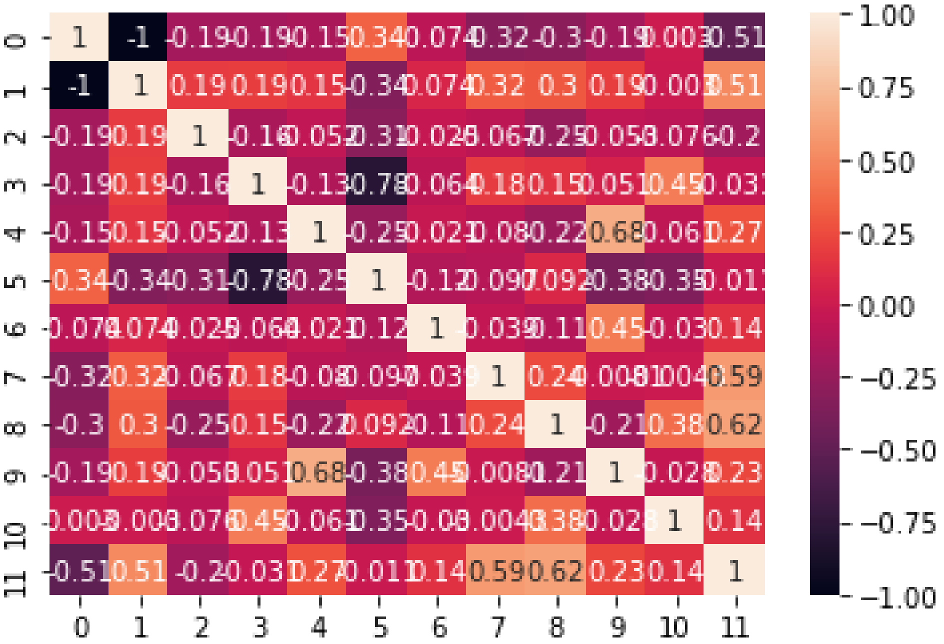


Figure 7: Heatmap of features correlation

Column 0 and 2 is showing negative correlation with our target variable (column 11), so these attributes should be avoided.

4 Description of the Models Used

For this project four models have been used. They are:

- Decision Tree
- Random Forrest
- Bayesian Ridge
- Support Vector Regression

4.1 Decision Tree

Decision trees are constructed via an algorithmic approach that identifies ways to split a dataset based on different conditions. It is a non-parametric supervised learning method used for both classification and regression tasks.

4.2 Random Forest

Random forest is a supervised learning algorithm. The ”forest” it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result. In a brief, random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

4.3 Bayesian Ridge

Bayesian regression allows a natural mechanism to survive insufficient data or poorly distributed data by formulating linear regression using probability distributors rather than point estimates. The output or response ‘y’ is assumed to drawn from a probability distribution rather than estimated as a single value.

4.4 Support Vector Regression

Support Vector Regression (SVR) is a supervised learning algorithm that is used to predict discrete values. Support Vector Regression uses the same principle as the Support Vector Machines. The basic idea behind SVR is to find the best fit line. In SVR, the best fit line is the hyperplane that has the maximum number of points.

5 Comparison of the Performance Scores Between Models

For the evaluation of our models, the performance metrics selected are:

- Mean Absolute Error (MAE): Calculates the absolute difference between actual and predicted values. And then sums all the errors and divide them by the total number of observations.
- Mean Squared Error (MSE): Finds the squared difference between actual and predicted value. It also sums all the errors and divide them by the total number of observations.
- Root Mean Squared Error (RMSE): Square root of mean squared error.
- R Squared (R2 Score): Tells the performance of the model, not the loss in an absolute sense that how many wells did the model perform. Basically R squared calculates how much regression line is better than a mean line.

The respected models' performance scores in accordance with their evaluation metrics is given below:

Model Name	MAE	MSE	RMSE	R2 Score
Decision Tree	0.347	1.320	0.935	0.783
Random Forest	0.410	1.162	0.963	0.831
Bayesian Ridge	1.050	2.302	1.447	0.654
Support Vector Regression	0.923	3.265	1.672	0.693

6 Discussion

We cross validated, trained multiple models on our prepared dataset and measured their performances.

In case of Mean Absolute Error (MAE), Decision Tree and Random Forrest performed very well (0.347 and 0.410). Bayesian Ridge and Support Vector Regression performed decent as well. The same pattern can be seen for Mean Squared Error (MSE) too where Decision Tree and Random Forrest outdid Bayesian Ridge and SVR. In case of Root Mean Squared Error (RMSE), again, Decision Tree and Random Forrest outplayed Bayesian Ridge and SVR. Similarly, according to our last performance metric, R2 Score, Decision Tree and Random Forrest's score is better than those of Bayesian Ridge and Support Vector Regression's. For example, the best R squared score came from Random Forrest model: 0.831 which means it is capable to explain almost 83.1 % of the variance of our data.

After analysing our models' performances, all of them showed good performances in general. And also this conclusion can be drawn that Decision Tree and Random Forrest worked as better models (in all cases) for our Ceramic Tiles Dataset in terms of predicting the tiles' prices.