Bansilal Ramnath Agarwal Charitable Trust's

VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE-48

Department of Information Technology

ITUA40201: DATA SCIENCE AND ANALYTICS
Assignment-2

Shreyas Shripad Kulkarni

BTech A Division

Roll No.: 431048

PRN: 22010443

AIM: Predictive Modelling Exercise

OBJECTIVE: Build a predictive model using machine learning algorithms and evaluate its performance.

TASKS:

1) Select a dataset (e.g., customer churn, housing prices).
2) Perform data preprocessing, including handling missing values and categorical variables.
3) Split the dataset into training and testing sets.
4) Train a predictive model (e.g., regression, classification) using suitable algorithms (e.g., linear regression, logistic regression, decision trees).
5) Evaluate the model & performance using appropriate metrics (e.g., accuracy, mean squared error). Fine-tune the model by adjusting parameters or using ensemble methods. Compare and interpret the results.

THEORY:

In this assignment I have used the housing price prediction dataset.

We will be first load the dataset. After cleaning the dataset and preprocessing the dataset we will form a correlation matrix.

After encoding the dataset we will train the predictive model after splitting the dataset into train and testing datasets.

Once the model is trained, we will evaluate the model by calculating the training and testing score & training, testing accuracy along with R-Squared, Mean Average Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE).

We will use the regularization (Ridge regression) to fine tune our model.

- Φ Correlation Matrix: A correlation matrix is a table or matrix that displays the correlation coefficients between many variables. A correlation matrix provides a concise overview of the relationships between variables, with values ranging from -1 to 1 indicating the strength and direction of linear correlations.
- Φ Encoding: Encoding is essential in machine learning as it transforms data into a format suitable for algorithms, such as converting categorical variables into numerical representations.
- Φ Training Accuracy: Training accuracy assesses how well a model fits the training data.
- Φ Testing Accuracy: Testing accuracy measures its performance on unseen data, crucial for evaluating generalization.
- Φ R-Squared: R-squared quantifies the goodness of fit in regression models, representing the proportion of variance in the dependent variable explained by independent variables.
- Φ Regularization: Regularization is a technique used in machine learning and statistical modelling to prevent overfitting and improve the generalization performance of a model.

There are 2 types of regularization:

- Δ Lasso Regression (L1 Regularization): Lasso Regression is also a type of regularization linear model. It also adds a penalty term to the cost function. *The main aim of Lasso Regression is to reduce the features and hence can be used for Feature Selection.*

∆ <u>Ridge Regression (L2 Regularization):</u> Ridge regression is a type of regularized regression model. This means it is a variation of the standard linear regression model that includes a regularized term in the cost function. The purpose of this is to prevent Overfitting.

IMPLEMENTATION:

In this assignment we will be using the standard dataset (Housing).

Step 1: Load Dataset.

431048_DSA_Assg2_Shreyas_

Code | Share | anaconda-panel-2023.05-py310

```
[1]: import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
```

```
[3]: df=pd.read_csv('Housing.csv')
```

```
[4]: df.head()
```

[4]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnished
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnished
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished

```
[43]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             545 non-null    int64
 1   area              545 non-null    int64
 2   bedrooms          545 non-null    int64
 3   bathrooms         545 non-null    int64
 4   stories           545 non-null    int64
 5   mainroad          545 non-null    int64
 6   guestroom         545 non-null    int64
 7   basement          545 non-null    int64
 8   hotwaterheating   545 non-null    int64
 9   airconditioning   545 non-null    int64
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    int64
 12  furnishingstatus  545 non-null    int64
dtypes: int64(13)
memory usage: 55.5 KB
```

```
[12]: df.dtypes
```

```
[12]: price                 int64
      area                  int64
      bedrooms              int64
      bathrooms             int64
      stories               int64
      mainroad             object
      guestroom            object
      basement             object
      hotwaterheating      object
      airconditioning      object
      parking               int64
      prefarea             object
      furnishingstatus     object
      dtype: object
```

```
[57]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             545 non-null    int64
 1   area              545 non-null    int64
 2   bedrooms          545 non-null    int64
 3   bathrooms         545 non-null    int64
 4   stories           545 non-null    int64
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

Step 2: Clean Dataset

```
[61]: df.duplicated().sum() #Returns Duplicate rows

[61]: 0
```

`df.isnull() .sum()` is the count of missing values in the corresponding column

```
[6]: df.isnull() .sum()

[6]: price               0
     area                0
     bedrooms            0
     bathrooms           0
     stories             0
     mainroad            0
     guestroom           0
     basement            0
     hotwaterheating     0
     airconditioning     0
     parking             0
     prefarea            0
     furnishingstatus    0
```

No missing values or duplicate values which means that our dataset is clean

Step 3: Form the Correlation matrix

[63]:
```
#Correlation Matrix
corr=df.corr()
print(corr)
```

```
              price      area  bedrooms  bathrooms   stories   parking
price      1.000000  0.535997  0.366494   0.517545  0.420712  0.384394
area       0.535997  1.000000  0.151858   0.193820  0.083996  0.352980
bedrooms   0.366494  0.151858  1.000000   0.373930  0.408564  0.139270
bathrooms  0.517545  0.193820  0.373930   1.000000  0.326165  0.177496
stories    0.420712  0.083996  0.408564   0.326165  1.000000  0.045547
parking    0.384394  0.352980  0.139270   0.177496  0.045547  1.000000
```

Step 4 : Encoding Categorical Data

[14]:
```
from sklearn.preprocessing import LabelEncoder
```

[17]:
```
le = LabelEncoder()
```

[69]:
```
#Defining a list of categorical columns to be encoded & encoding them
categorical_data = ['mainroad','guestroom','basement','hotwaterheating','airconditioning','prefarea','furnishingstatus']
for i in categorical_data:
    df[i] = le.fit_transform(df[i])
```

[21]:
```
df.head()
```

[21]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	1	0	0	0	1	2	1	0
1	12250000	8960	4	4	4	1	0	0	0	1	3	0	0
2	12250000	9960	3	2	2	1	0	1	0	0	2	1	1
3	12215000	7500	4	2	2	1	0	1	0	1	3	1	0
4	11410000	7420	4	1	2	1	1	1	0	1	2	0	0

[23]:
```
x=df.drop('price',axis=1) #x will contain all columns except the price column
y=df.price # y will have only the price column
```

[25]:
```
print(x.head())
```

```
   area  bedrooms  bathrooms  stories  mainroad  guestroom  basement  \
0  7420         4          2        3         1          0         0
1  8960         4          4        4         1          0         0
2  9960         3          2        2         1          0         1
3  7500         4          2        2         1          0         1
4  7420         4          1        2         1          1         1

   hotwaterheating  airconditioning  parking  prefarea  furnishingstatus
0                0                1        2         1                 0
1                0                1        3         0                 0
2                0                0        2         1                 1
3                0                1        3         1                 0
4                0                1        2         0                 0
```

[27]:
```
print(y.head())
```

```
0    13300000
1    12250000
2    12250000
3    12215000
4    11410000
Name: price, dtype: int64
```

Step 5: Split the dataset

We will split the data in 7:3 Ratio i.e 20% of the dataset will be training data & 30% data will be test data.

```
[29]: from sklearn.model_selection import train_test_split
      from sklearn.metrics import mean_absolute_error, mean_squared_error,confusion_matrix,r2_score
```

```
[31]: #Split data in 70:30 ratio
      x_train, x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state = 5)
```

Step 6: Train the predictive model

```
[33]: from sklearn.linear_model import LinearRegression
```

```
[35]: model = LinearRegression()
      model.fit(x_train,y_train)
```

```
[35]: ▾ LinearRegression
      LinearRegression()
```

Step 7: Evaluating the model

Training & Testing score

```
[48]:  print("The training score is,",model.score(x_train,y_train),end='\n')
        print("The testing score is,",model.score(x_test,y_test))
```

```
The training score is, 0.6722721620878297
The testing score is, 0.6701127297811891
```

Training & Testing Accuracy

```
[60]: training_score = model.score(x_train, y_train)*100
      training_accuracy = "{:.2f}".format(training_score)
      print("Training Accuracy in % =",training_accuracy,end='\n')

      testing_score = model.score(x_test, y_test)*100
      testing_accuracy = "{:.2f}".format(testing_score)
      print("Testing Accuracy in % =",testing_accuracy,end='\n')
```

```
Training Accuracy in % = 67.23
Testing Accuracy in % = 67.01
```

```
[87]: import numpy as np

y_pred=model.predict(x_test)

mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error (MAE):", mae)

mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)

rmse = np.sqrt(mse)
print("Root Mean Squared Error (RMSE):", rmse)

r_squared = r2_score(y_test, y_pred)
print("R-squared:", r_squared)
```

```
Mean Absolute Error (MAE): 721569.6538611307
Mean Squared Error (MSE): 827865989822.3118
Root Mean Squared Error (RMSE): 909871.4138944644
R-squared: 0.6701127297811891
```

Step 8: Fine Tuning

We will use the regularization (Ridge regression) to fine tune our model.

```
[140]: #fine tuning
#Regularization (Ridge Regression)
from sklearn.linear_model import Ridge
ridge_model = Ridge(alpha=0.13)  # WE can adjust the alpha value
ridge_model.fit(x_train_scaled, y_train)

mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error (MAE):", mae)

mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)

rmse = np.sqrt(mse)
print("Root Mean Squared Error (RMSE):", rmse)

r_squared = r2_score(y_test, y_pred)
print("R-squared:", r_squared)
```

```
Mean Absolute Error (MAE): 699698.2978645586
Mean Squared Error (MSE): 767377026262.3
Root Mean Squared Error (RMSE): 876000.5857659571
R-squared: 0.6589825037347056
```

We can clearly see that the MAE,MSE & RMS has been reduced which indicates the model is finely tuned and is fine that the previous model

<u>CONCLUSION</u>: We have learnt, understood and performed predictive modelling on housing prediction dataset.

GitHub Repository