ZAHIDE GULSAH DURMAZ

26 JUNE 2024

HOUSE PRICE PREDICTION ML PROJECT

BACKGROUND

Dataset: You can download the dataset from **this** link.

Notebook: You can find the colab notebook file from **this** link.

No	Column Name	Description
1	Id	To count the records.
2	MSSubClass	Identifies the type of dwelling involved in the sale.
3	MSZoning	Identifies the general zoning classification of the sale.
4	LotArea	Lot size in square feet.
5	LotConfig	Configuration of the lot.
6	BldgType	Type of dwelling.
7	OverallCond	Rates the overall condition of the house.
8	YearBuilt	Original construction year.
9	YearRemodAdd	Remodel date (same as construction date if no remodeling or additions).
10	Exterior1st	Exterior covering on house.
11	BsmtFinSF2	Type 2 finished square feet.
12	TotalBsmtSF	Total square feet of basement area.
13	SalePrice	To be predicted.

- The dataset contains 2,919 house prices and associated predictors.
- It includes 13 explanatory variables that describe various aspects of residential homes.
- Using advanced regression techniques, the goal is to predict the final price of each home.

LIBRARIES

Pandas: To load the Dataframe

Matplotlib: To visualize the data features i.e. barplot

Seaborn: To see the correlation between features using heatmap

```
[ ] # Step 1: Mount Google Drive
    from google.colab import drive
    drive.mount('/content/drive/')

    # Step 2: Import the necessary libraries
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns

    # Step 3: Load the dataset from Google Drive
    file_path = '/content/drive/MyDrive/REDI_FINAL_DURMAZ/HousePricePrediction.xlsx'
    dataset = pd.read_excel(file_path)

    # Step 4: Print the first 5 records of the dataset
    print(dataset.head(5))
```

```
Mounted at /content/drive/
   Id  MSSubClass MSZoning  LotArea LotConfig BldgType  OverallCond  \
0   0          60       RL     8450    Inside     1Fam            5
1   1          20       RL     9600       FR2     1Fam            8
2   2          60       RL    11250    Inside     1Fam            5
3   3          70       RL     9550    Corner     1Fam            5
4   4          60       RL    14260       FR2     1Fam            5

   YearBuilt  YearRemodAdd Exterior1st  BsmtFinSF2  TotalBsmtSF  SalePrice
0       2003          2003     VinylSd         0.0        856.0   208500.0
1       1976          1976     MetalSd         0.0       1262.0   181500.0
2       2001          2002     VinylSd         0.0        920.0   223500.0
3       1915          1970     Wd Sdng         0.0        756.0   140000.0
4       2000          2000     VinylSd         0.0       1145.0   250000.0
```

DATA PREPROCESSING

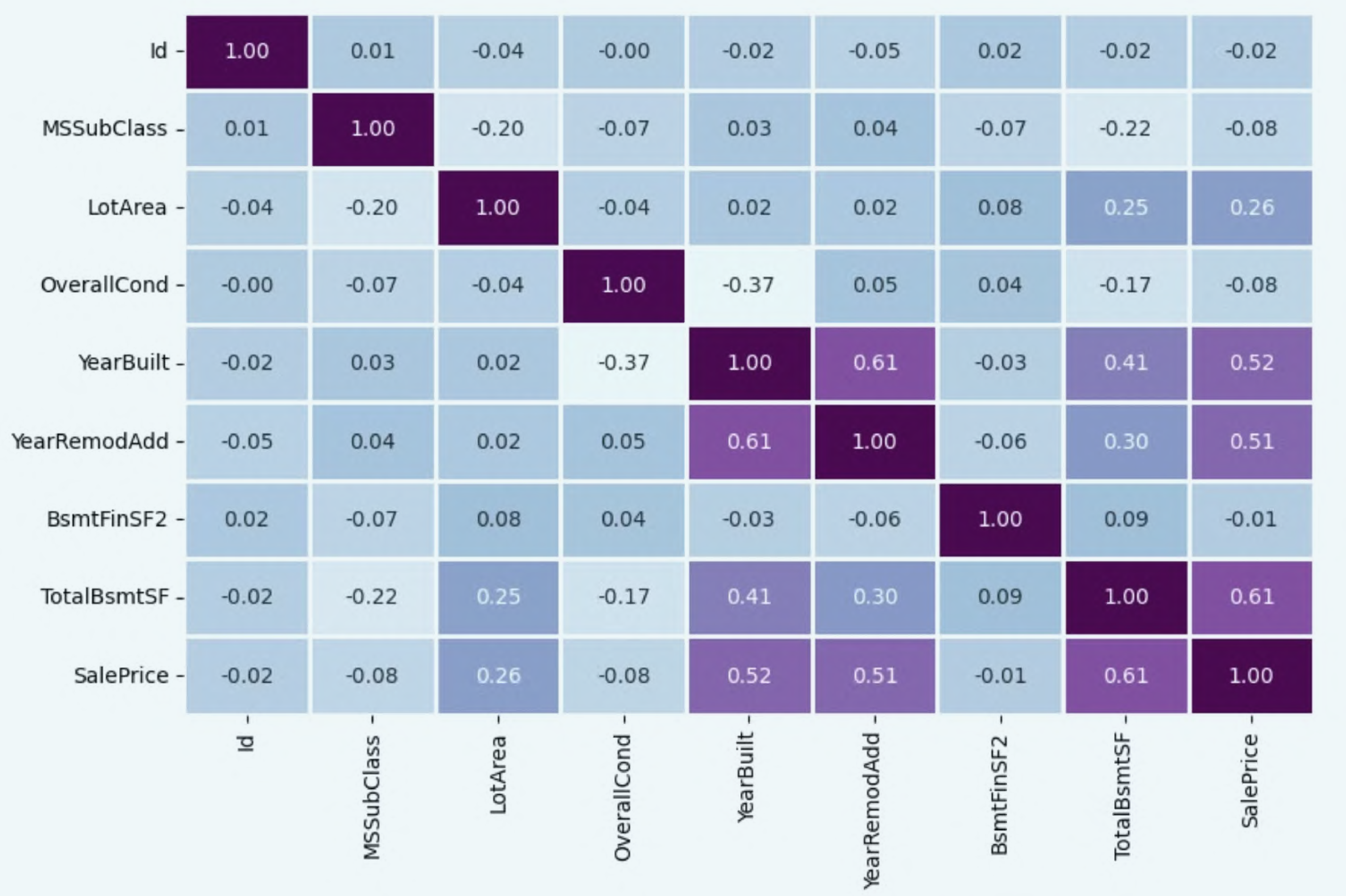
Categorizing the features depending on their datatype (int, float, object) and then calculate the number of them

```
[ ] obj_ = (dataset.dtypes == 'object')
    # print (obj_)
    # object_col = list(obj_)
    # print (object_col)
    # object_col2 = list(obj_[obj_])
    # print (object_col2)
    object_cols = list(obj_[obj_].index)
    # print (object_cols)
    print("Categorical variables:",len(object_cols))

    int_ = (dataset.dtypes == 'int')
    num_cols = list(int_[int_].index)
    print("Integer variables:",len(num_cols))

    fl_ = (dataset.dtypes == 'float')
    fl_cols = list(fl_[fl_].index)
    print("Float variables:",len(fl_cols))
```

```
Categorical variables: 4
Integer variables: 6
Float variables: 3
```

EXPLORATORY DATA ANALYSIS

Heatmap using seaborn library.



Categorical columns (object type) cannot be included in the correlation matrix because correlation is only meaningful for numeric data. Thus, we need to exclude the categorical columns from our dataset before computing the correlation matrix.

DATA CLEANING

As in our dataset, there are some columns that are not important and irrelevant for the model training. So, we can drop that column before training. There are 2 approaches to dealing with empty/null values

- We can easily delete the column/row (if the feature or record is not much important).
- Filling the empty slots with mean/mode/0/NA/etc. (depending on the dataset requirement). As Id Column will not be participating in any prediction. So we can Drop it.

```
[ ] dataset['SalePrice'] = dataset['SalePrice'].fillna(
    dataset['SalePrice'].mean())
```

```
missing_values= dataset.isnull().sum()
print (missing_values)
```

```
Id              0
MSSubClass      0
MSZoning        4
LotArea         0
LotConfig       0
BldgType        0
OverallCond     0
YearBuilt       0
YearRemodAdd    0
Exterior1st     1
BsmtFinSF2      1
TotalBsmtSF     1
SalePrice       0
dtype: int64
```

Drop records with null values (as the empty records are very less).

```
[ ] new_dataset = dataset.dropna()
```

```
[ ] missing_values= dataset.isnull().sum()
    print ("dataset")
    print (missing_values)
    print ('\n')

    missing_values_new_dataset= new_dataset.isnull().sum()
    print ("new_dataset")
    print (missing_values_new_dataset)
    print ('\n')
```

SPLITTING DATASET INTO TRAINING AND TESTING

X and Y splitting (i.e. Y is the SalePrice column and the rest of the other columns are X)

```
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split

X = df_final.drop(['SalePrice'], axis=1)
Y = df_final['SalePrice']

# Split the training set into
# training and validation set
X_train, X_valid, Y_train, Y_valid = train_test_split(
  X, Y, train_size=0.8, test_size=0.2, random_state=0)
```

MODEL AND ACCURACY

SVM – Support vector Machine

```
[ ] from sklearn import svm
    from sklearn.svm import SVC
    from sklearn.metrics import mean_absolute_percentage_error

    model_SVR = svm.SVR()
    model_SVR.fit(X_train,Y_train)
    Y_pred = model_SVR.predict(X_valid)

    print(mean_absolute_percentage_error(Y_valid, Y_pred))

    0.18704778826125987
```

Random Forest Regression

```
[ ] from sklearn.ensemble import RandomForestRegressor

    model_RFR = RandomForestRegressor(n_estimators=10)
    model_RFR.fit(X_train, Y_train)
    Y_pred = model_RFR.predict(X_valid)

    mean_absolute_percentage_error(Y_valid, Y_pred)

    0.08354500868585663
```

Linear Regression

```
[ ] from sklearn.linear_model import LinearRegression

    model_LR = LinearRegression()
    model_LR.fit(X_train, Y_train)
    Y_pred = model_LR.predict(X_valid)

    print(mean_absolute_percentage_error(Y_valid, Y_pred))

    0.18633155158087458
```

CONCLUSION

Clearly, SVM model is giving better accuracy as the mean absolute error is the least among all the other regressor models i.e. 0.18 approx. To get much better results ensemble learning techniques like Bagging and Boosting can also be used.

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)