

# HOME VALUE PREDICTION

## REPORT ON AI MODEL

*Submitted in fulfillment for the Internal*

*Assessment of*

*Artificial Intelligence Lab (BITE308P)*

*in*

**B.Tech. – Information Technology**

*by*

**REPAKULA YASODA RUSHITHA (21BIT0433)**  
**MADHURANTHAKAM S KAVYA SREE (21BIT0483)**  
**AADITYA V MENON(21BIT0454)**  
**SNEHA VEJJU (21BIT0647)**  
**ISHIKA KHANDELWAL (21BIT0657)**

*Under the guidance of*

**Dr. S. Hemalatha**

**SCORE**



**VIT<sup>®</sup>**  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

**School of Computer Science Engineering &  
Information Systems**

Winter Semester 2023-24

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	2
<b>1.</b>	<b>Model Description &amp; Details</b>	2
<b>2.</b>	<b>Model design</b>	3
	<b>2.1 Architecture Diagram / Flow Diagram / Flowchart / ...</b>	3
<b>3.</b>	<b>Software Requirement Specifications</b>	4
<b>4.</b>	<b>Experimental Results &amp; Discussion</b>	4
	<b>4.1.Source code</b>	5
	<b>4.2.Screenshots with caption</b>	7

### ABSTRACT :

In the pursuit of buying a new house, individuals often encounter challenges such as fraud, negotiations, and thorough research of local areas. To mitigate these issues, a machine learning-based model is developed for house price prediction. This model is trained on a comprehensive dataset containing 13 features, including information such as lot size, construction year, and sale price. The dataset undergoes data preprocessing, exploratory data analysis (EDA), and data cleaning before being used for model training and evaluation. Various regression models, including Support Vector Machine (SVM), Random Forest, and Linear Regression, are employed to predict house prices accurately.

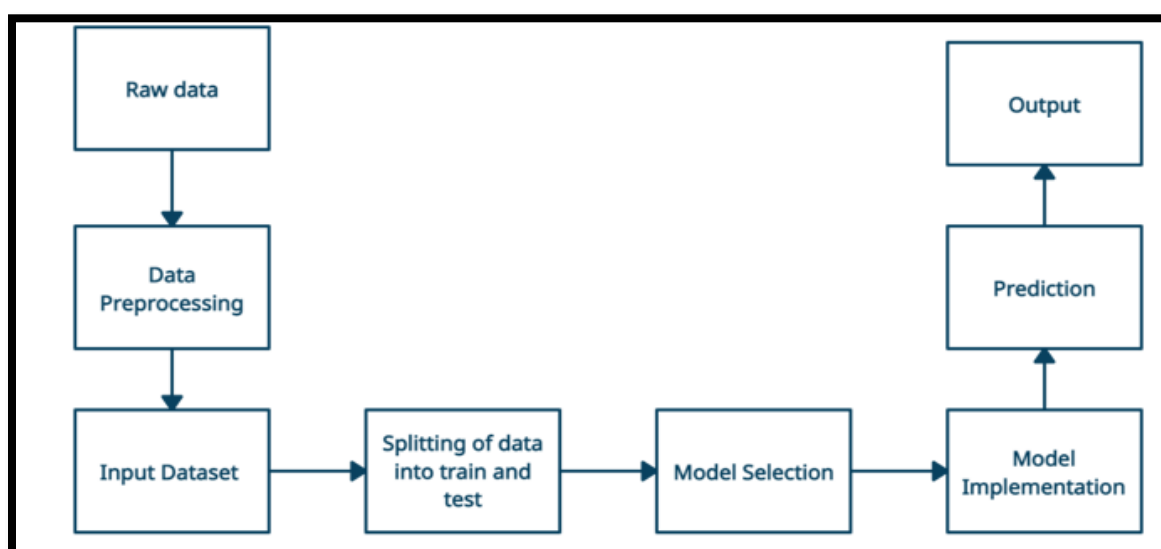
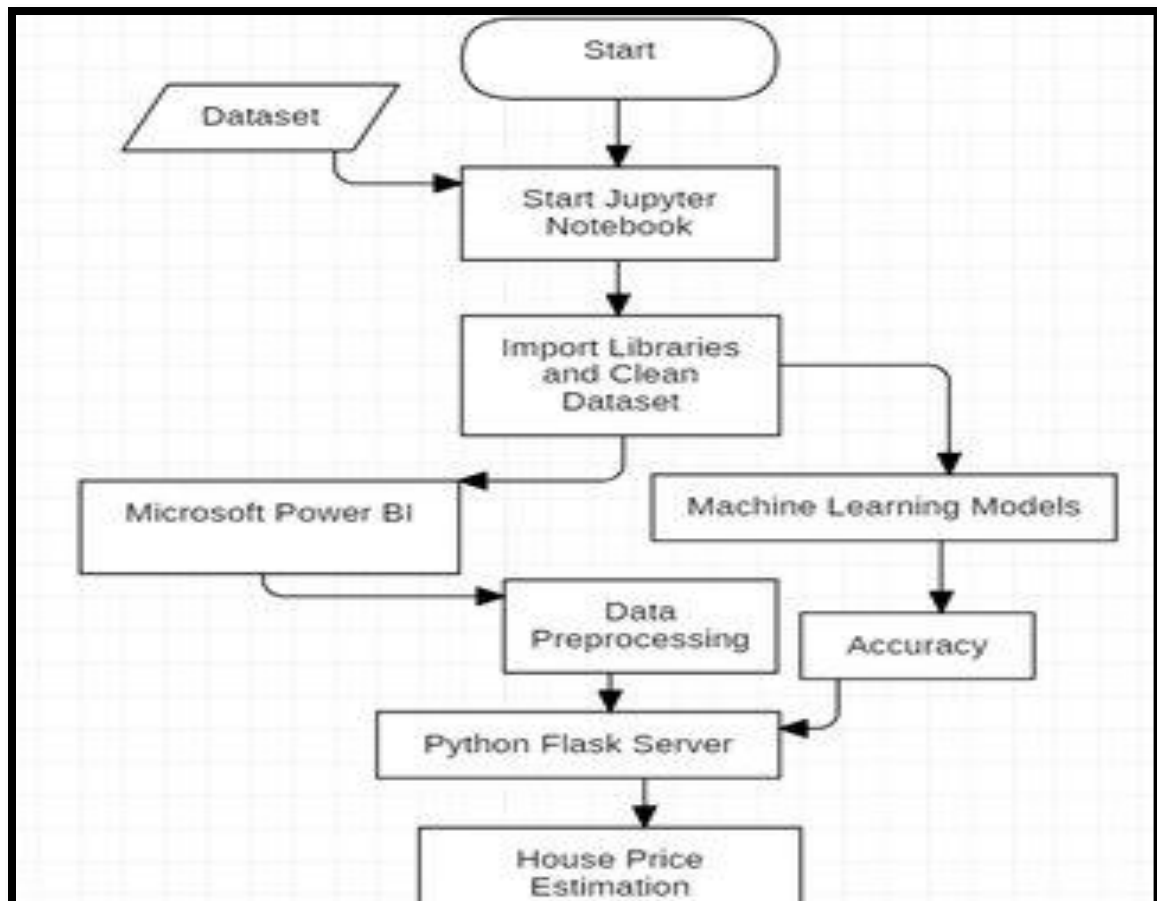
### MODEL DESCRIPTION & DETAILS :

1. **Data Preparation:** The dataset is loaded using Pandas, and its dimensions are examined using the shape method to understand its structure.
2. **Data Preprocessing:** Features are categorized based on their data types, and the number of categorical, integer, and float variables is determined.
3. **Exploratory Data Analysis (EDA):** EDA techniques, such as heatmaps and barplots, are utilized to visualize correlations between features and analyze the distribution of categorical variables.
4. **Data Cleaning:** Irrelevant columns are dropped, and missing values are handled by either deleting rows or filling empty slots with appropriate values.
5. **OneHotEncoder:** Categorical features are encoded using OneHotEncoder to convert them into binary vectors.
6. **Splitting Dataset:** The dataset is split into training and testing sets to prepare for model training.

7. **Model Training and Evaluation:** Support Vector Machine, Random Forest, and Linear Regression models are trained on the training set and evaluated using mean absolute percentage error (MAPE) to measure their accuracy.

## MODEL DESIGN

### 1) ARCHITECTURE DIAGRAM/ FLOW DIAGRAM/ FLOW CHART :



## **SOFTWARE REQUIREMENTS AND SPECIFICATIONS :**

### **1. Programming Language: Python**

- Python is used for data analysis, preprocessing, model training, and evaluation.

### **2. Libraries and Frameworks:**

- Pandas: Used for data manipulation and analysis.
- Matplotlib: Utilized for data visualization, particularly for creating barplots and heatmaps.
- Seaborn: Used for advanced data visualization and generating correlation heatmaps.
- Scikit-learn: Provides machine learning algorithms and tools for model training and evaluation.
- CatBoost: Utilized for training the CatBoostRegressor model, an efficient gradient boosting library.

### **3. Dataset:**

- House Price Prediction Dataset: Contains 13 features including :
  - Id: To count the records.
  - MSSubClass: Identifies the type of dwelling involved in the sale.
  - MSZoning: Identifies the general zoning classification of the sale.
  - LotArea: Lot size in square feet.
  - LotConfig: Configuration of the lot
  - BldgType: Type of dwelling
  - OverallCond: Rates the overall condition of the house
  - YearBuilt: Original construction year
  - YearRemodAdd: Remodel date (same as construction date if no remodeling).
  - Exterior1st: Exterior covering on house
  - BsmtFinSF2: Type 2 finished square feet.
  - TotalBsmtSF: Total square feet of basement area
  - SalePrice: To be predicted

### **4. Integrated Development Environment (IDE):**

- Any Python-compatible IDE can be used for coding and development, such as:
  - Jupyter Notebook
  - PyCharm
  - Visual Studio Code
  - Spyder
  - Sublime Text

### **5. Operating System:**

- The project can be developed and executed on various operating systems including Windows, macOS, and Linux.

### **6. Internet Connection:**

- Required for downloading datasets, libraries, and accessing online documentation and resources.

## **EXPERIMENTAL RESULTS & DISCUSSION**

### **1. SOURCE CODE :**

#### **i. Importing Data:**

```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

dataset = pd.read_excel("HousePricePrediction.xlsx")

# Printing first 5 records of the dataset
print(dataset.head(5))

```

## ii. Data Processing:

```

obj = (dataset.dtypes == 'object')
object_cols = list(obj[obj].index)
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))

```

## iii. EDA:

```

plt.figure(figsize=(12, 6))
sns.heatmap(dataset.corr(),
             cmap = 'BrBG',
             fmt = '.2f',
             linewidths = 2,
             annot = True)
plt.title('correlation Heatmap')
plt.show()

```

## iv. Catagorical Features:

```

unique_values = []
for col in object_cols:
    unique_values.append(dataset[col].unique().size)
plt.figure(figsize=(10,6))
plt.title('No. Unique values of Categorical Features')
plt.xticks(rotation=90)
sns.barplot(x=object_cols,y=unique_values)

```

## v. Actual cout of each category:

```

plt.figure(figsize=(18, 36))
plt.title('Categorical Features: Distribution')
plt.xticks(rotation=90)

```

```
index = 1
```

```
for col in object_cols:
    y = dataset[col].value_counts()
    plt.subplot(11, 4, index)
    plt.xticks(rotation=90)
    sns.barplot(x=list(y.index), y=y)
    index += 1
```

**vi. Data Cleaning or Replacing empty value with their mean value:**

```
dataset.drop(['Id'],
              axis=1,
              inplace=True)
OR
dataset['SalePrice'] = dataset['SalePrice'].fillna(
dataset['SalePrice'].mean())
```

**vii. Checking features with null values:**

```
new_dataset.isnull().sum()
```

**viii. OneHotEncoder:**

```
from sklearn.preprocessing import OneHotEncoder

s = (new_dataset.dtypes == 'object')
object_cols = list(s[s].index)
print("Categorical variables:")
print(object_cols)
print('No. of. categorical features: ',
      len(object_cols))
```

**ix. Splitting Dataset:**

```
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)
```

**x. Model & Accuracy :  
SVM:**

```
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))

```

### 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)

```

### Linear Regression:

```

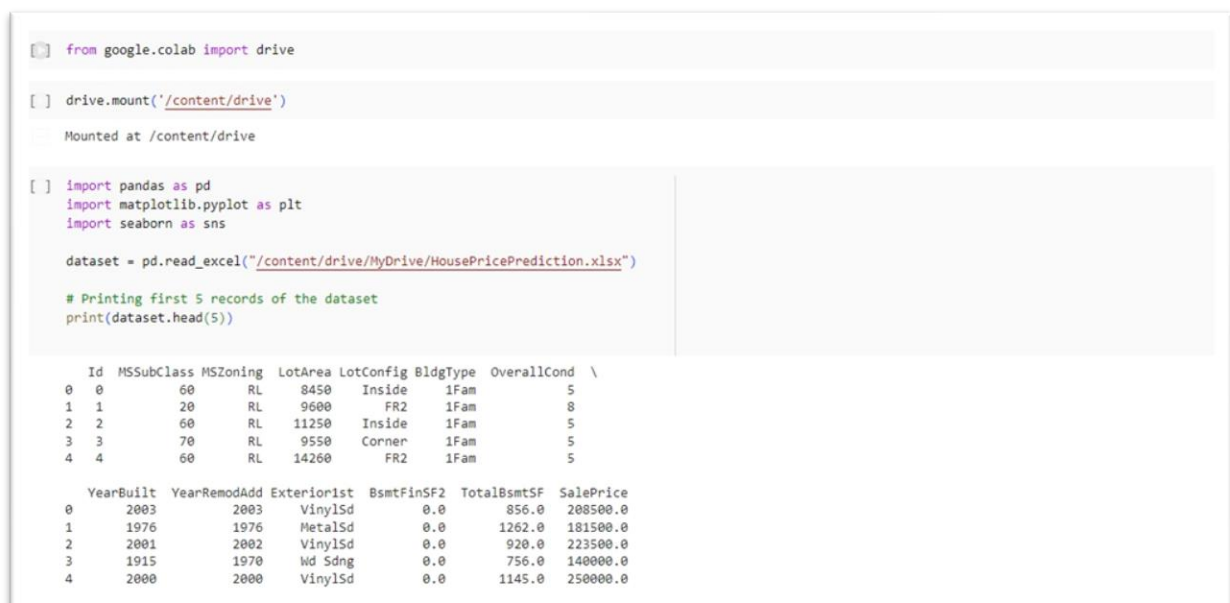
from sklearn.linear_model import LinearRegression

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))

```

## 2. SCREENSHOTS WITH CAPTIONS



```

from google.colab import drive

drive.mount('/content/drive')

Mounted at /content/drive

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

dataset = pd.read_excel("/content/drive/MyDrive/HousePricePrediction.xlsx")

# Printing first 5 records of the dataset
print(dataset.head(5))

```

	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 stored in Excel sheet and printing 5 records of the datasets.

```
[ ]
dataset.shape

(2919, 13)

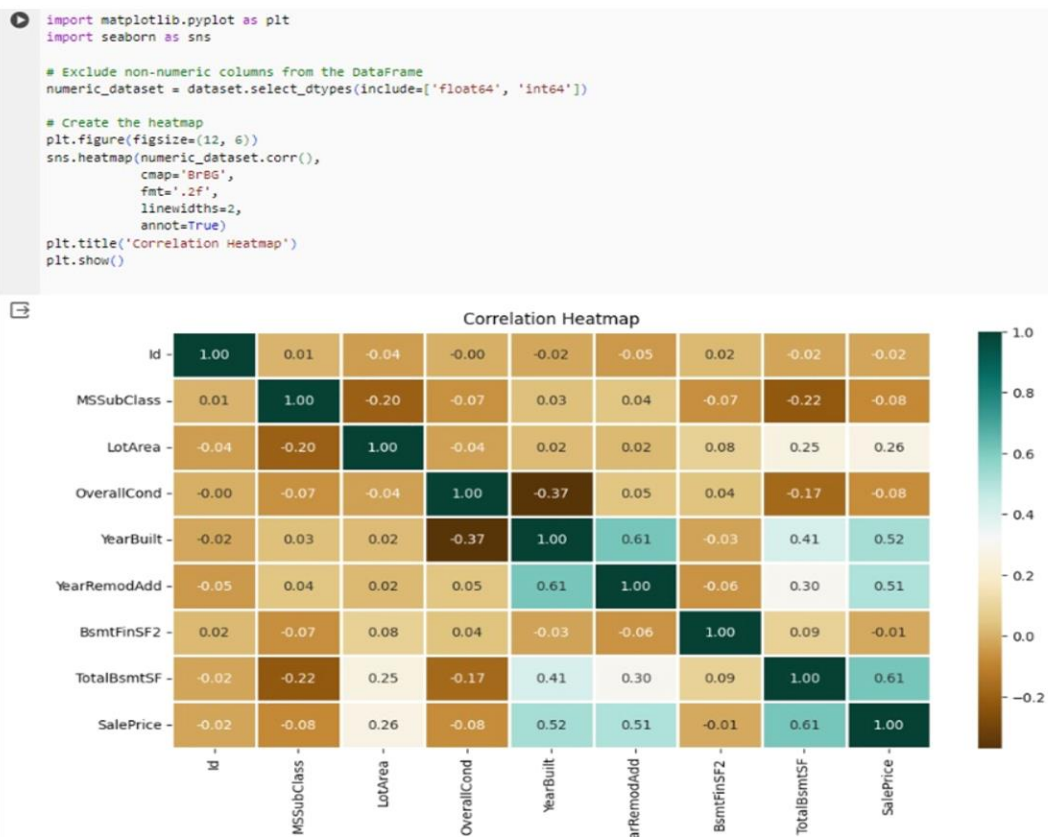
obj = (dataset.dtypes == 'object')
object_cols = list(obj[obj].index)
print("Categorical variables:",len(object_cols))

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))

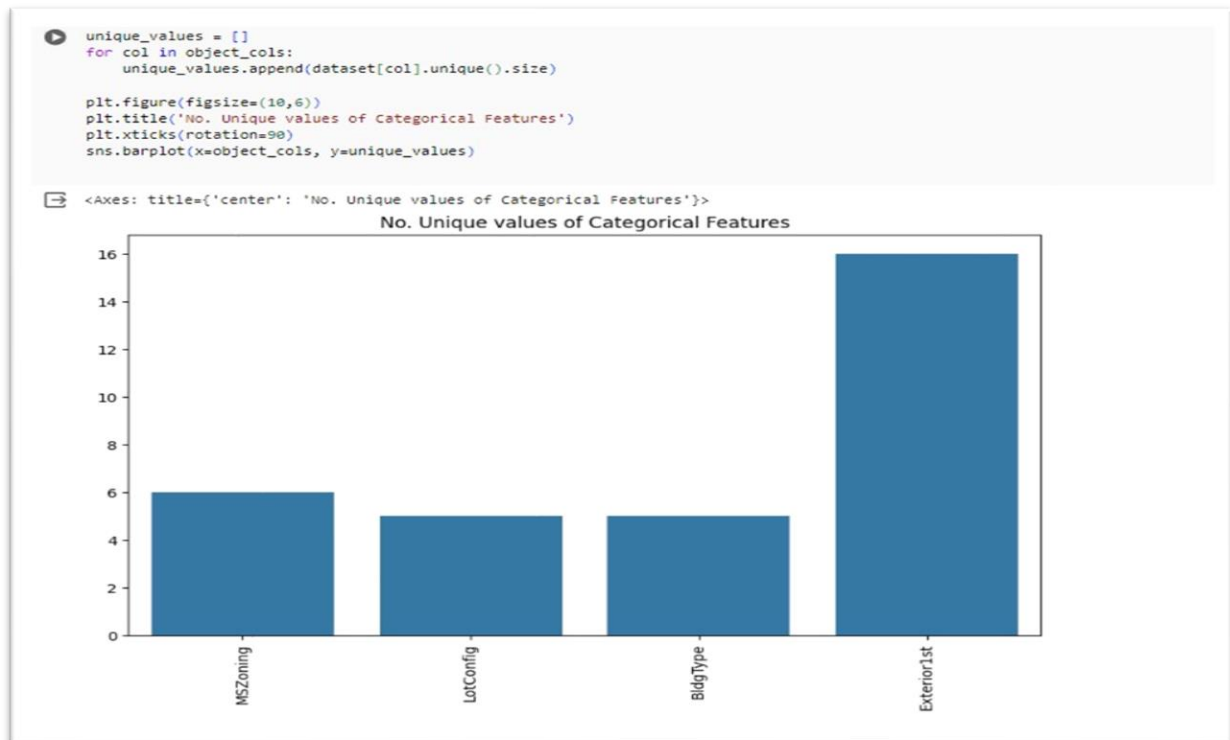
➡ Categorical variables: 4
Integer variables: 6
Float variables: 3
```

As we have imported the data. So shape method will show us the dimension of the dataset. We categorize the features depending on their datatype (int, float, object) and then calculate the number of them.

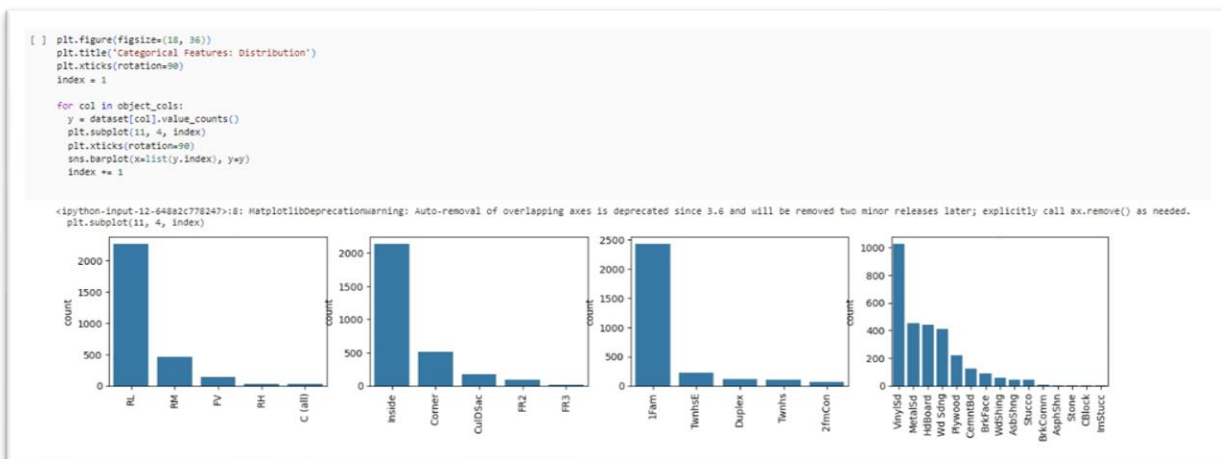




EDA refers to the deep analysis of data so as to discover different patterns and spot anomalies. Before making inferences from data it is essential to examine all your variables.



To analyze the different categorical features. Let's draw the barplot. The plot shows that Exterior1st has around 16 unique categories and other features have around 6 unique categories.



To find out the actual count of each category we can plot the bargraph of each four features separately.

```

dataset.drop(['Id'],
             axis=1,
             inplace=True)

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

[ ] new_dataset = dataset.dropna()

[ ] new_dataset.isnull().sum()

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

```

Data Cleaning is the way to improvise the data or remove incorrect, corrupted or irrelevant data.

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.

Replacing SalePrice empty values with their mean values to make the data distribution symmetric.

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

Checking features which have null values in the new dataframe (if there are still any).

```

[ ] from sklearn.preprocessing import OneHotEncoder

s = (new_dataset.dtypes == 'object')
object_cols = list(s[s].index)
print("Categorical variables:")
print(object_cols)
print('No. of. categorical features: ',
      len(object_cols))

Categorical variables:
['MSZoning', 'LotConfig', 'BldgType', 'Exterior1st']
No. of. categorical features: 4

```

One hot Encoding is the best way to convert categorical data into binary vectors. This maps the values to integer values. By using OneHotEncoder, we can easily convert object data into

int. So for that, firstly we have to collect all the features which have the object datatype. To do so, we will make a loop.

```
[ ] OH_encoder = OneHotEncoder(sparse=False)
OH_cols = pd.DataFrame(OH_encoder.fit_transform(new_dataset[object_cols]))
OH_cols.index = new_dataset.index
OH_cols.columns = OH_encoder.get_feature_names_out(input_features=object_cols) # Use get_feature_names_out()
df_final = new_dataset.drop(object_cols, axis=1)
df_final = pd.concat([df_final, OH_cols], axis=1)

/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to
warnings.warn(

[ ] 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)
```

Then once we have a list of all the features. We can apply OneHotEncoding to the whole list. X and Y splitting (i.e. Y is the SalePrice column and the rest of the other columns are X)

```
✓ [28] from sklearn import svm
    Os 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))

0.1870512931870423
```

```
✓ [29] from sklearn.ensemble import RandomForestRegressor
    Os

    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)

0.1899216673214028
```

```
✓ [30] from sklearn.linear_model import LinearRegression
    Os

    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))

0.18741683841599854
```

1. SVM can be used for both regression and classification model. It finds the hyperplane in the n-dimensional plane.
2. Random Forest is an ensemble technique that uses multiple of decision trees and can be used for both regression and classification tasks.
3. Linear Regression predicts the final output-dependent value based on the given independent features. Like, here we have to predict SalePrice depending on features like MSSubClass, YearBuilt, BldgType, Exterior1st etc.

```
✓ [33] from catboost import CatBoostRegressor
      cb_model = CatBoostRegressor()
      cb_model.fit(X_train, Y_train)
      preds = cb_model.predict(X_valid)

      cb_r2_score=r2_score(Y_valid, preds)
      cb_r2_score
```

```
943: learn: 24895.9359012 total: 4.83s remaining: 287ms
944: learn: 24885.8891683 total: 4.84s remaining: 281ms
945: learn: 24869.3289658 total: 4.84s remaining: 276ms
946: learn: 24857.4844584 total: 4.84s remaining: 271ms
947: learn: 24850.1350142 total: 4.85s remaining: 266ms
948: learn: 24831.5426380 total: 4.86s remaining: 261ms
949: learn: 24820.4710430 total: 4.87s remaining: 256ms
950: learn: 24805.8606697 total: 4.87s remaining: 251ms
951: learn: 24785.2701657 total: 4.88s remaining: 246ms
```

```
992: learn: 24502.7457525 total: 5.13s remaining: 36.2ms
993: learn: 24490.1166911 total: 5.14s remaining: 31ms
994: learn: 24488.5023143 total: 5.14s remaining: 25.8ms
995: learn: 24468.9731701 total: 5.15s remaining: 20.7ms
996: learn: 24459.0497225 total: 5.16s remaining: 15.5ms
997: learn: 24458.7284279 total: 5.16s remaining: 10.3ms
998: learn: 24448.9981948 total: 5.17s remaining: 5.17ms
999: learn: 24440.7422679 total: 5.17s remaining: 0us
0.38351169878113034
```

CatBoost is a machine learning algorithm implemented by Yandex and is open-source. It is simple to interface with deep learning frameworks such as Apple's Core ML and Google's TensorFlow. Performance, ease-of-use, and robustness are the main advantages of the CatBoost library.

## RESULTS:

- SVM achieved the best performance with a MAPE of 0.18.
- CatBoost came in second with a MAPE of 0.19.
- Random Forest and Linear Regression had similar performance with a MAPE around 0.18-0.19.

## CONCLUSION:

- The Support Vector Machine (SVM) model demonstrates the best accuracy among the regression models, with a mean absolute percentage error (MAPE) of approximately 18%.

