SUPERIOR UNIVERSITY

Submitted By:

Name: **Um e Habiba**

Roll number **Su92-bsdsm-f23-040**

Section: **4A**

Task: **01**

Subject: **PAI(lab)**

Submitted to: **Sir Rasikh Ali**

https://www.kaggle.com/code/umehabibaakbarali/house-pricing-predicition

1. Environment Setup and Data Loading

- The notebook starts by importing necessary libraries such as pandas, numpy, and os.
- It lists the files available in the input directory using os.walk.
- The dataset (train.csv) is loaded into a pandas DataFrame (df).

2. Data Exploration

- The first few rows of the dataset are displayed using df.head(5) to get an initial look at the data.

3. Data Preprocessing

- **Handling Missing Values:**
  - Columns with more than 50% missing values are dropped.
  - Missing values in numerical columns are filled with the median, and missing values in categorical columns are filled with the string 'Unknown'.
- **Encoding Categorical Variables:**
  - Categorical columns are one-hot encoded using pd.get_dummies.

4. Model Training and Evaluation

- The target variable (SalePrice) is separated from the features.
- The data is split into training and validation sets using train_test_split.
- A RandomForestRegressor model is trained on the training data.
- The model's performance is evaluated on the validation set using metrics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R^2 score.

5. Submission Preparation

- The model is used to predict house prices on a test dataset (test.csv).
- The predictions are saved to a CSV file (submission.csv) for submission, typically to a competition platform like Kaggle.

6. Additional Model Training

- A simpler model is trained using a subset of features (LotArea, YearBuilt, 1stFlrSF, 2ndFlrSF, FullBath, BedroomAbvGr, TotRmsAbvGrd).
- This model is also evaluated using Mean Absolute Error (MAE), and predictions are saved to a submission file.

Libraries Used:

- **Pandas:** For data manipulation and analysis.
- **NumPy:** For numerical operations.
- **Scikit-learn:** For machine learning tasks, including model training, evaluation, and data splitting.
- **Matplotlib/Seaborn:** Although not explicitly used in this notebook, these libraries are often used for data visualization in similar projects.

```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import pandas as pd
import numpy as np
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/home-data-for-ml-course/sample_submission.csv
/kaggle/input/home-data-for-ml-course/sample_submission.csv.gz
/kaggle/input/home-data-for-ml-course/train.csv.gz
/kaggle/input/home-data-for-ml-course/data_description.txt
/kaggle/input/home-data-for-ml-course/test.csv.gz
/kaggle/input/home-data-for-ml-course/train.csv
/kaggle/input/home-data-for-ml-course/test.csv
```

```python
df=pd.read_csv('/kaggle/input/home-data-for-ml-course/train.csv')
```

Python

```python
df.head(5)
```

Python

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleConditio
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2008	WD	Norma
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	5	2007	WD	Norma
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	9	2008	WD	Norma
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2006	WD	Abnorm
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	12	2008	WD	Norma

5 rows × 81 columns

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import numpy as np
```

Python

Load dataset

[5]
```
file_path = '/kaggle/input/home-data-for-ml-course/train.csv'
df = pd.read_csv(file_path)
```

pre_processing

[6]
```
#Drop columns with more than 50% missing values
columns_to_drop = [col for col in df.columns if (df[col].isnull().sum() / df.shape[0]) * 100 > 50]
df.drop(columns=columns_to_drop, inplace=True)
```

fill missing values

[7]
```
#numerical columns

numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns
df[numerical_cols] = df[numerical_cols].fillna(df[numerical_cols].median())
# categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns
df[categorical_cols] = df[categorical_cols].fillna('Unknown')
```

```
#Encoded
df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
```

Test_Train Spliting

```
target = 'SalePrice'  # Target variable for this dataset
X = df.drop(columns=[target])
y = df[target]
```

```
assert X.select_dtypes(include=['object']).empty, "There are still non-numeric columns in X"
```

```
# Split data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
Rf = RandomForestRegressor(n_estimators=100, random_state=42)
Rf.fit(X_train, y_train)
```

```
RandomForestRegressor(random_state=42)
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

```
y_pred = Rf.predict(X_val)
rmse = np.sqrt(mean_squared_error(y_val, y_pred))
print(f"Root Mean Squared Error (RMSE): {rmse}")
mae = mean_absolute_error(y_val, y_pred)
r2 = r2_score(y_val, y_pred)
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"R^2 Score: {r2:.2f}")
```

```
Root Mean Squared Error (RMSE): 29311.513541784003
Mean Absolute Error (MAE): 17687.96
R^2 Score: 0.89
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
# Load the dataset
file_path = '../input/home-data-for-ml-course/train.csv'  # Adjust path as needed
data = pd.read_csv(file_path)
# Select target and features
y = data['SalePrice']
features = ['LotArea', 'YearBuilt', '1stFlrSF', '2ndFlrSF', 'FullBath', 'BedroomAbvGr', 'TotRmsAbvGrd']
X = data[features]
# Split data into training and validation datasets
X_train, X_valid, y_train, y_valid = train_test_split(X, y, train_size=0.8, test_size=0.2, random_state=0)

# Define the model
model = RandomForestRegressor(random_state=0)
# Fit the model
model.fit(X_train, y_train)

# Get predictions
preds = model.predict(X_valid)

# Calculate the mean absolute error
mae = mean_absolute_error(y_valid, preds)
print(f"Mean Absolute Error: {mae}")

# Prepare test data for submission
test_data = pd.read_csv('../input/home-data-for-ml-course/test.csv')  # Adjust path as needed
X_test = test_data[features]

test_preds = model.predict(X_test)
```

```
# Prepare test data for submission
test_data = pd.read_csv('../input/home-data-for-ml-course/test.csv')  # Adjust path as needed
X_test = test_data[features]

test_preds = model.predict(X_test)

# Create a DataFrame for submission
output = pd.DataFrame({'Id': test_data.Id, 'SalePrice': test_preds})
output.to_csv('submission.csv', index=False)
print("Submission file saved as submission.csv")
```

```
Mean Absolute Error: 23740.979228636657
Submission file saved as submission.csv
```