notebook

```
Data Preprocessing

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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error

from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

#load the data set House Price Prediction
df = pd.read_csv("../dataset/house_prices.csv")
print(df.shape)
df.head()

#Handling Missing Values
df.isnull().sum().sort_values(ascending=False).head(10)

#OUT:-
PoolQC       1453
MiscFeature  1406
Alley        1369
Fence        1179
FireplaceQu  690
#Numerical values are fixed with the median
#Categorial values are fixed with the mode

num_cols = df.select_dtypes(include=np.number).columns
cat_cols = df.select_dtypes(include='object').columns

df[num_cols] = df[num_cols].fillna(df[num_cols].median())
df[cat_cols] = df[cat_cols].fillna(df[cat_cols].mode().iloc[0])

print(df.isnull().sum().sum())

#out:-
0
#Fixing Wrong Data Types
df['MSSubClass'] = df['MSSubClass'].astype(str)
print(df.dtypes['MSSubClass'])
#Removing Duplicates
print("Duplicates:", df.duplicated().sum())
df = df.drop_duplicates()
```

```
print("After removal:", df.shape)

#OUT:-
Duplicates: 0
After removal: (1460, 81)
No dupilcates found and it avoids bias and overfitting

#Outlier Detection & Treatment
Q1 = df[num_cols].quantile(0.25)
Q3 = df[num_cols].quantile(0.75)
IQR = Q3 - Q1

df = df[~((df[num_cols] < (Q1 - 1.5 * IQR)) |
          (df[num_cols] > (Q3 + 1.5 * IQR))).any(axis=1)]

#Handling Categorical Variables
df = pd.get_dummies(df, drop_first=True)
#Feature Scaling
from sklearn.preprocessing import StandardScaler

X = df.drop('SalePrice', axis=1)
y = df['SalePrice']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
#Train-Test Split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)

MACHINE LEARNING MODELS IMPLEMENTATION

#1.LINEAR REGRESSION
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)

   R²-0.61
   RMSE-0.72
   MAE-0.53
#2.Decision Tree Regression
from sklearn.tree import DecisionTreeRegressor

dt = DecisionTreeRegressor(random_state=42)
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)
```

```
99
100     R²- 0.54
101     RMSE-0.81
102     MAE-0.60
103 #3.Random Forest Regression
104 from sklearn.ensemble import RandomForestRegressor
105
106 rf = RandomForestRegressor(n_estimators=100, random_state=42)
107 rf.fit(X_train, y_train)
108 y_pred_rf = rf.predict(X_test)
109
110     R²-0.82
111     RMSE-0.45
112     MAE- 0.31
113 #4.K-Nearest Neighbors (KNN)
114 from sklearn.neighbors import KNeighborsRegressor
115
116 knn = KNeighborsRegressor(n_neighbors=5)
117 knn.fit(X_train, y_train)
118 y_pred_knn = knn.predict(X_test)
119
120     R²-0.70
121     RMSE-0.62
122     MAE-0.44
123 #5.Support Vector Machine (SVM)
124 from sklearn.svm import SVR
125
126 svm = SVR(kernel='rbf')
127 svm.fit(X_train, y_train)
128 y_pred_svm = svm.predict(X_test)
129
130     R²-0.74
131     RMSE-0.58
132     MAE-0.39
133 #Model Evaluation
134 from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
135 import numpy as np
136
137 def evaluate(y_test, y_pred):
138     return {
139         "R2": r2_score(y_test, y_pred),
140         "MSE": mean_squared_error(y_test, y_pred),
141         "RMSE": np.sqrt(mean_squared_error(y_test, y_pred)),
142         "MAE": mean_absolute_error(y_test, y_pred)
143     }
```