**Lab 1**

**Explanation:**

This code implements a machine learning pipeline to predict house prices using the Random Forest Regressor. It begins by loading training (train.csv) and test (test.csv) datasets using pandas. The target variable (SalePrice) is separated from the features, and the Id column is dropped since it’s not useful for predictions. The data is then split into numerical and categorical features. For numerical data, missing values are imputed with the mean, and the values are standardized using StandardScaler. For categorical data, missing values are filled with the most frequent category, and OneHotEncoder is used to convert categorical variables into numerical format. A ColumnTransformer applies these preprocessing steps to the dataset. A RandomForestRegressor is used as the predictive model, and the entire workflow is organized using a Pipeline for streamlined processing. The dataset is split into training and validation sets, and the model is trained on the training set. The predictions are evaluated using Root Mean Squared Error (RMSE) and R² Score. Finally, the trained model makes predictions on the test dataset, and the results are saved in a submission.csv file. A histogram of the target variable (SalePrice) is also plotted to visualize its distribution. This structured approach ensures efficient data preprocessing, feature transformation, and model training, making the prediction process scalable and reproducible.

**Dataset:**

<https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data>

**Task: Kaggle Competition: House Price Prediction**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.impute import SimpleImputer

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error

import numpy as np

import matplotlib.pyplot as plt

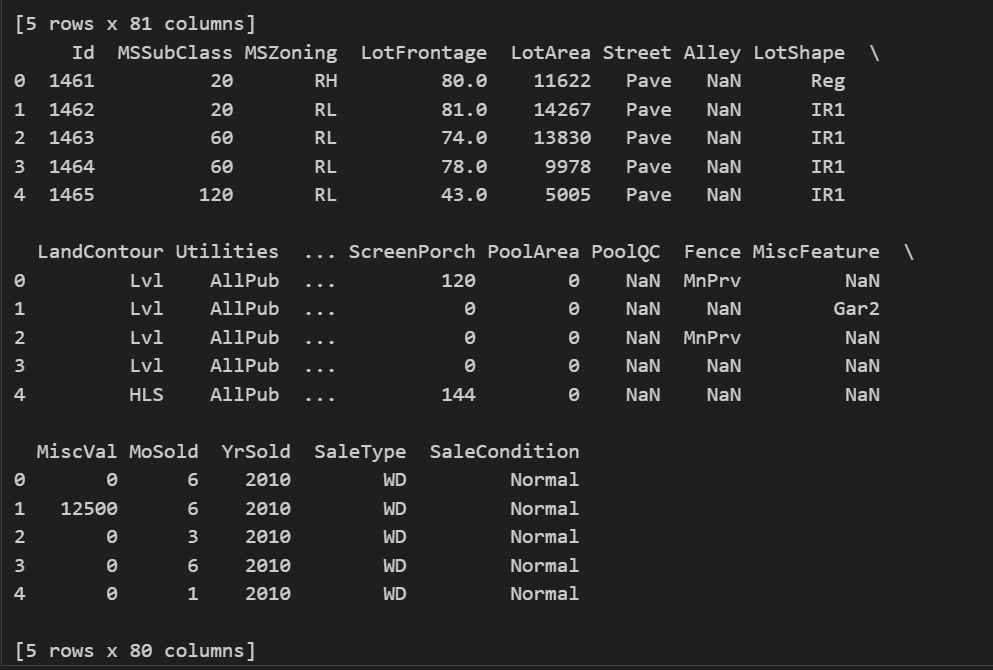
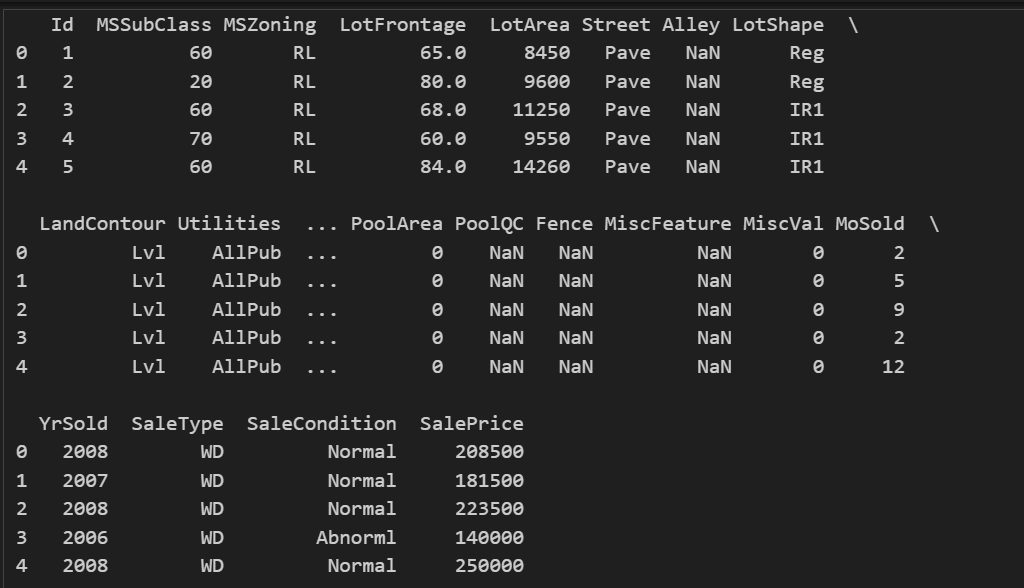
import seaborn as sns

train\_data = pd.read\_csv('train.csv')

test\_data = pd.read\_csv('test.csv')

print(train\_data.head())

print(test\_data.head())



X = train\_data.drop(columns=['SalePrice', 'Id'])

y = train\_data['SalePrice']

X\_test = test\_data.drop(columns=['Id'])

categorical\_cols = X.select\_dtypes(include=['object']).columns

numerical\_cols = X.select\_dtypes(exclude=['object']).columns

numerical\_transformer = Pipeline([

('imputer', SimpleImputer(strategy='mean')),

('scaler', StandardScaler())

])

categorical\_transformer = Pipeline([

('imputer', SimpleImputer(strategy='most\_frequent')),

('onehot', OneHotEncoder(handle\_unknown='ignore'))

])

preprocessor = ColumnTransformer([

('num', numerical\_transformer, numerical\_cols),

('cat', categorical\_transformer, categorical\_cols)

])

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

pipeline = Pipeline([

('preprocessor', preprocessor),

('model', model)

])

X\_train, X\_valid, y\_train, y\_valid = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

pipeline.fit(X\_train, y\_train)

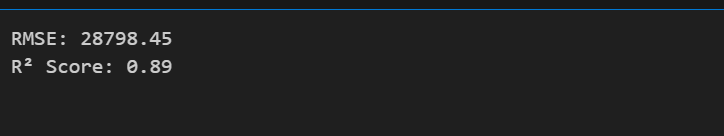
y\_pred = pipeline.predict(X\_valid)

rmse = np.sqrt(mean\_squared\_error(y\_valid, y\_pred))

r2 = r2\_score(y\_valid, y\_pred)

print(f"RMSE: {rmse:.2f}")

print(f"R² Score: {r2:.2f}")

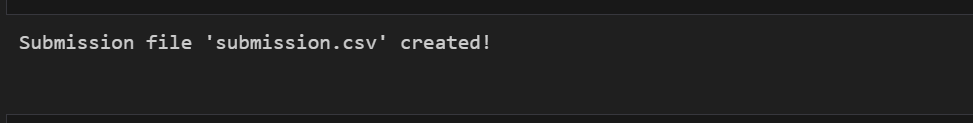


test\_preds = pipeline.predict(X\_test)

submission = pd.DataFrame({'Id': test\_data['Id'], 'SalePrice': test\_preds})

submission.to\_csv('submission.csv', index=False)

print("Submission file 'submission.csv' created!")



plt.figure(figsize=(8, 6))

sns.histplot(y, kde=True, color='blue')

plt.title("SalePrice Distribution")

plt.xlabel("SalePrice")

plt.ylabel("Frequency")

plt.show()

