ASSIGNMENT 1 MLT

House Price Prediction using Linear Regression

Introduction

The objective of this assignment is to develop a linear regression model to predict house prices using a dataset from Zillow's Zestimate. The dataset includes features such as square footage, number of bedrooms and bathrooms, location, and other attributes of the houses. The model is evaluated using metrics like Mean Squared Error (MSE) and R-squared (R²).

Step 1: Data Loading and Inspection

First, we load the dataset and perform a preliminary inspection to understand the data structure. We examine the first few rows, data types, and basic statistics of each feature.

Code:

import pandas as pd

# Load the training and test data

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

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

# Display the first few rows and info about the data

print(train\_df.head())

print(train\_df.info())

print(train\_df.describe())

Step 2: Handling Missing Values

To address missing values, we use median imputation for numerical columns and the most frequent value for categorical columns. This helps ensure that the model can be trained effectively without biases introduced by missing data.

Code:

from sklearn.impute import SimpleImputer

# Identify columns with missing values

missing\_values = X.isnull().sum().sort\_values(ascending=False)

print(missing\_values[missing\_values > 0]) # Display columns with missing values

Step 3: Encoding Categorical Data

Categorical features are identified and converted into numerical values using one-hot encoding. This step is necessary to ensure the regression model can interpret the categorical variables.

Code:

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

# Preprocessing for categorical data

categorical\_transformer = Pipeline(steps=[

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

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

])

Step 4: Splitting the Data

The data is split into training and validation sets. The training set is used to fit the model, and the validation set is used to evaluate the model’s performance before applying it to the test data.

Code:

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

# Split the data into training and validation sets

X\_train, X\_valid, y\_train, y\_valid = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

Step 5: Training the Linear Regression Model

A linear regression model is trained using the pipeline, which preprocesses the data and fits the model. This step involves combining both numerical and categorical preprocessing.

Code:

from sklearn.linear\_model import LinearRegression

# Define the model

model = LinearRegression()

# Create a pipeline to preprocess the data and fit the model

pipeline = Pipeline(steps=[('preprocessor', preprocessor),

('model', model)])

# Train the model

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

Step 6: Model Evaluation

The model is evaluated on the validation set using Mean Squared Error (MSE) and R-squared (R²) metrics. These metrics provide insights into the model’s accuracy and its ability to explain the variance in house prices.

Code:

from sklearn.metrics import mean\_squared\_error, r2\_score

# Make predictions on the validation set

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

# Evaluate the model

mse = mean\_squared\_error(y\_valid, y\_pred)

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

print(f"Mean Squared Error (MSE): {mse}")

print(f"R-squared (R²): {r2}")

Conclusion

In this assignment, we developed a linear regression model to predict house prices based on various features. The model was evaluated using MSE and R², achieving moderate accuracy. Improvements can be explored through feature engineering, advanced models, or hyperparameter tuning.