PREDICTING HOUSE PRICES USING MACHINE LEARNING

PHASE 4: DEVELOPMENT PART 2

Continue building the house price prediction model by feature selection, model training and evaluation.

PROCEDURE:

FEATURE SELECTION:

There are several steps involved in feature selection. Some usually include steps are:

1. Identifying the target variable. The variable wants to predict or understand the analysis. It's the outcome or dependent variable.

2. Explore the data. It is an important step in building a predictive model. You can use data visualization and correlation analysis to identify features that are highly correlated with the target variable.

3. Remove redundant features. Analyse the correlation between different features and eliminate those that have high correlation with each other. This help to avoid multicollinearity and reduce noise in model.

4. Remove irrelevant features. If a feature has little or no impact on the target variable, it can be considered irrelevant and removed from the dataset.

#Checking for the missing values

Print(“missing values by column”)

Print(“-“*30)

Print(df.isna().sum())

Model Training:

Choose a machine learning algorithm:

For predicting house prices, we can consider using a regression algorithm like linear regression, decision tress regression, random forest regression, or lasso regression. These algorithms can help to analyse the relationship between different features and the house prices. To preprocess data and evaluate the performance of the model using appropriate metrics.

Machine learning models:

Linear Regression:

#Assuming you have feature matrix X and target variable y ready

#Split the data into training and testing sets

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2, rando m_state=42)

#Create an instance of the Linear Regression model

Model=LinearRegression()

#Fit the model to the training data

Model.fit(X_train,y_train)

#Make predictions on the testing data

Y_pred=model.predict(X_test)

#Evaluate the model using mean squared error

Mse=mean_squared_error(y_test,y_pred)

Print ("Mean squared Error:", mse)

To replace 'X' with feature matrix and 'y' with target variables.

Random Forest Regressor:

from sklearn.linear_model import RandomForestRegressor

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error

#Create an instance of the Ridge regression model

Model=RandomForestRegressor(estimator=100) #You can adjust the parameter to control the amount regularization

#Fit the model to the training data

Model.fit(X_train,y_train)

#Make predictions on the testing data

y_pred=model.predict(X_test)

#Evaluate the model using mean squared error

Mse=mean_squared_error(y_test,y_pred)

Print (“Mean Squared Error:”, mse)

Lasso Regression:

#Assuming you have feature matrix X and target variable y ready

#Split the data into training and testing sets

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2, rando m_state=42)

#Create an instance of the Linear Regression model

model=Lasso(alpha=0.5)

#Fit the model to the training data

model.fit(X_train, y_train)

#Make predictions on the testing data

y_pred=model.predict (X_test)

#Evaluate the model using mean squared error

Mse=mean_squared_error(y_test,y_pred)

Print (“Mean squared Error:”, mse)

Print (“Mean absolute error:”, mae)

Various features to perform model training:

1. Numerical features: These includes variables like square footage, number of bedrooms, number of bathrooms, and other quantitative attributed that can directly impact house prices.

2. Categorical features: These include variables like location, neighbourhood, property type, and other qualitative attributes that can influence house prices. It can encode categorical features using techniques like one-hot encoding or label encoding.

3. Temporal features: If you have access to data over time, you can extract features like year built, year of renovation, or seasonality patterns that can capture the temporal dynamics of house prices.

4. Interaction features: You can create interaction features by combining two or more existing features.

5. Derived features: These features derived from existing data through mathematical transformation or domain-specific knowledge.

6. External data: You can augment your dataset with external data sources, such as economic indicators, or demographic information that can provide additional insights into house prices.

Model evaluation:

To evaluate our model, we are going to use the explained_variance_score metric and the 'r2_score' metric methods which are provided by the scikit-learn package in python.

R-squared is a measurement of how well the dependent variable explains the variance of the independent variable. It is the most popular evaluation metric for regression models.

1. Split the data: Divide dataset into a training set and a testing set. The training set is used to train the model, while the testing set is used to evaluate its performance.

2. Choose an evaluation metric: Select an appropriate metric to measure the performance of our model. For regression tasks, common metrics include mean squared error (MSE), root mean squared error (RMSE), and R-squared. For classification tasks, metrics like accuracy, precision, recall, and F1 score are commonly used.

3.Train your model: Use the training set to train your machine learning model. This involves feeding the model with

the input features and corresponding target values, and allowing it to learn the underlying patterns in the data.

4.Evaluate your model: Once the model is trained, use the testing set to evaluate its performance. Make predictions on the test set using the trained model and compare them with the actual target values.

5.Calculate the evaluation metric: Apply the chosen evaluation metric to compare the predicted values with the actual values. This will give a quantitative measure of how well model is performing.

6.Iteate and improve: Based on the evaluation results, you can iterate and improve your model by adjusting hyperparameters, trying different algorithms, or performing feature engineering to enhance its performance.

There are a number of different metrics that can be used to evaluate the performance of a house price prediction model. Some of the most common metrics include:

• Mean squared error (MSE): This metric measures the average squared difference between the predicted and actual house prices.

• Mean Absolute Error (MAE): This metric calculates the average absolute difference between the predicted house prices and the actual prices. It gives you an idea of how far off, on average, predictions are from the true values.

• Root Mean Squared Error (RMSE): RMSE is similar to MAE but takes the square root of the average squared differences between the predicted and actual house prices.

• R-Squared Score: R-squared measures the proportion of the variance in the target variable (house prices) that can be explained by the model. It ranges from 0 to 1, with higher values indicating a better fit to the data.

Conclusion:

In conclusion, building a house price prediction model involves feature selection, model training, and model evaluation. By selecting relevant features, training the model using appropriate algorithms, and evaluating its performance using metrics like MAE, MSE, RMSE and R-squared, can create an effective model for predicting house prices. Each of these stages plays an indispensable role in crafting model that can provide meaningful insights and estimates for one of the most significant financial decisions individuals and businesses make real estate transactions.