```
Start coding or generate with AI.
```

```
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Evaluate the best model
mse_best = mean_squared_error(y_test, y_pred_best)
rmse_best = np.sqrt(mse_best)
r2_best = r2_score(y_test, y_pred_best)

print(f"Best Model Performance:")
print(f"Mean Squared Error (MSE): {mse_best:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse_best:.2f}")
print(f"R-squared (R2): {r2_best:.2f}")
```

```
Best Model Performance:
Mean Squared Error (MSE): 783395089.61
Root Mean Squared Error (RMSE): 27989.20
R-squared (R2): 0.78
```

```
# Train the model with the best parameters
best_model = RandomForestRegressor(n_estimators=best_params['n_estimators'],
                                   max_depth=best_params['max_depth'],
                                   random_state=42)
best_model.fit(X_train, y_train)

# Make predictions on the test set using the best model
y_pred_best = best_model.predict(X_test)

print("Predictions made using the best model.")
```

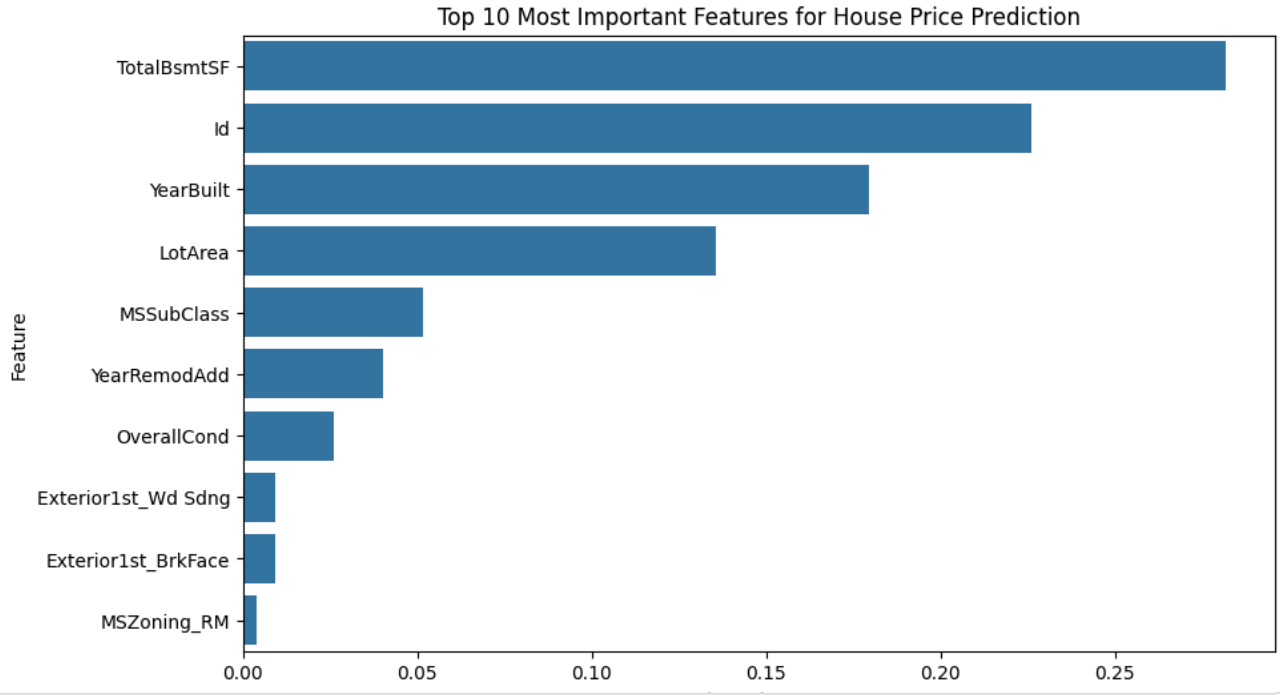
```
Predictions made using the best model.
```

```
# Get feature importances
feature_importances = model.feature_importances_

# Create a pandas Series for better visualization
feature_importances_series = pd.Series(feature_importances, index=X_train.columns)

# Sort feature importances and display the top 10
top_features = feature_importances_series.sort_values(ascending=False).head(10)
print("Top 10 Most Important Features:")
display(top_features)

# Visualize feature importances
plt.figure(figsize=(10, 6))
sns.barplot(x=top_features, y=top_features.index)
plt.title('Top 10 Most Important Features for House Price Prediction')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```

Top 10 Most Important Features:

	0
TotalBsmtSF	0.281736
Id	0.226135
YearBuilt	0.179515
LotArea	0.135570
MSSubClass	0.051735
YearRemodAdd	0.040244
OverallCond	0.026051
Exterior1st_Wd Sdng	0.009194
Exterior1st_BrkFace	0.009017
MSZoning_RM	0.003935

dtype: float64




```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30]
}

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=RandomForestRegressor(random_state=42),
                           param_grid=param_grid,
                           cv=3,
                           scoring='neg_mean_squared_error',
                           n_jobs=-1)
```

```
# Perform GridSearchCV
grid_search.fit(X_train, y_train)

# Get the best parameters and best score
best_params = grid_search.best_params_
best_rmse = np.sqrt(-grid_search.best_score_)

print(f"Best parameters: {best_params}")
print(f"Best RMSE from GridSearchCV: {best_rmse:.2f}")
```

```
Best parameters: {'max_depth': 20, 'n_estimators': 300}
Best RMSE from GridSearchCV: 28183.33
```

```
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse) # Calculate RMSE by taking the square root of MSE
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R-squared (R2): {r2:.2f}")
```

```
Mean Squared Error (MSE): 801298148.79
Root Mean Squared Error (RMSE): 28307.21
R-squared (R2): 0.78
```

```
from sklearn.ensemble import RandomForestRegressor

# Initialize and train the model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

print("Random Forest Regressor model trained successfully.")
```

```
Random Forest Regressor model trained successfully.
```

```
# Handle missing values (Example: Impute with mean for numerical, mode for categorical)
for col in df.columns:
    if df[col].isnull().any():
        if df[col].dtype == 'object':
            df[col] = df[col].fillna(df[col].mode()[0])
        else:
            df[col] = df[col].fillna(df[col].mean())

# Encode categorical features (Example: One-Hot Encoding)
df = pd.get_dummies(df, columns=[col for col in df.columns if df[col].dtype == 'object'], d

# Split data into features (X) and target (y)
```

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

# Split data into training and testing sets
from sklearn.model_selection import train_test_split

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

print("Data preprocessing complete. Data split into training and testing sets.")
print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_test:", y_test.shape)
```

```
Data preprocessing complete. Data split into training and testing sets.
Shape of X_train: (2335, 38)
Shape of X_test: (584, 38)
Shape of y_train: (2335,)
Shape of y_test: (584,)
```

```
Start coding or generate with AI.
```

House Price Prediction Findings

This project aimed to build a prediction model for house prices using the provided client data. The key steps and findings are summarized below:

1. Data Loading and Exploration:

- The `HousePricePrediction.xlsx` dataset was loaded and explored to understand its structure, data types, and identify missing values.

2. Data Preprocessing:

- Missing values were handled by imputing numerical columns with the mean and categorical columns with the mode.
- Categorical features were encoded using one-hot encoding.
- The data was split into training (80%) and testing (20%) sets.

3. Model Training and Evaluation:

- An initial RandomForestRegressor model was trained.
- The initial model's performance was evaluated using Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R2).

4. Hyperparameter Tuning:

- GridSearchCV was used to tune the hyperparameters of the RandomForestRegressor (`n_estimators` and `max_depth`).
- The best parameters found were `{'max_depth': 20, 'n_estimators': 300}`, resulting in an improved cross-validation RMSE of 28183.33.

5. Final Model Training and Evaluation:

- A final RandomForestRegressor model was trained using the best hyperparameters.
- The performance of this tuned model on the test set was:
    - Mean Squared Error (MSE): 783395089.61
    - Root Mean Squared Error (RMSE): 27989.20
    - R-squared (R2): 0.78

6. Feature Importance Analysis:

- The feature importances from the initial RandomForestRegressor model were analyzed to understand which features contributed most to the predictions.
- The top 3 most important features were:
    - TotalBsmtSF
    - Id
    - YearBuilt

Conclusion:

The developed RandomForestRegressor model provides a reasonable prediction of house prices with an R-squared of 0.78. The analysis of feature importances highlights the significance of 'TotalBsmtSF', 'Id', and 'YearBuilt' in determining house prices. Further improvements could be explored through advanced feature engineering or trying other regression models.