Forecasting House Prices Using Smart Regression Techniques

In [ ]:
```
# 1 & 2. Upload and Load the Dataset
from google.colab import files
import pandas as pd

uploaded = files.upload()  # Upload 'sample_house_prices.csv'
df = pd.read_csv("sample_house_prices.csv")
print("Data loaded successfully!")
df.head()
```

Upload widget is only available when the cell has been executed in the current browser session. P rerun this cell to enable.

```
Saving sample_house_prices.csv to sample_house_prices.csv
Data loaded successfully!
```

Out[ ]:

	LotArea	OverallQual	YearBuilt	TotRmsAbvGrd	GarageCars	FullBath	SalePrice
0	7732	3	1953	8	2	2	267712
1	14845	2	1980	8	1	1	450090
2	8264	5	1962	6	0	3	271880
3	9859	7	1908	6	1	1	139208
4	14225	9	1961	4	1	1	111761

In [ ]:
```
# 3. Data Exploration
print(df.info())
print(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   LotArea       100 non-null    int64
 1   OverallQual   100 non-null    int64
 2   YearBuilt     100 non-null    int64
 3   TotRmsAbvGrd  100 non-null    int64
 4   GarageCars    100 non-null    int64
 5   FullBath      100 non-null    int64
 6   SalePrice     100 non-null    int64
dtypes: int64(7)
memory usage: 5.6 KB
None
            LotArea  OverallQual    YearBuilt  TotRmsAbvGrd  GarageCars  \
count    100.000000   100.000000   100.000000    100.000000  100.000000
mean   10177.920000     5.390000  1961.840000      5.470000    1.410000
std     2760.741738     2.651072    33.081387      2.293843    1.146757
min     5537.000000     1.000000  1900.000000      2.000000    0.000000
25%     7741.750000     3.000000  1934.500000      4.000000    0.000000
50%    10028.000000     5.000000  1963.500000      6.000000    1.000000
75%    12387.750000     8.000000  1991.000000      7.000000    2.250000
max    14893.000000     9.000000  2017.000000      9.000000    3.000000

         FullBath      SalePrice
count  100.00000     100.000000
mean     2.12000  270794.020000
std      0.80754  135362.162819
min      1.00000   51221.000000
25%      1.00000  145913.000000
50%      2.00000  269796.000000
75%      3.00000  385745.500000
max      3.00000  495678.000000
```

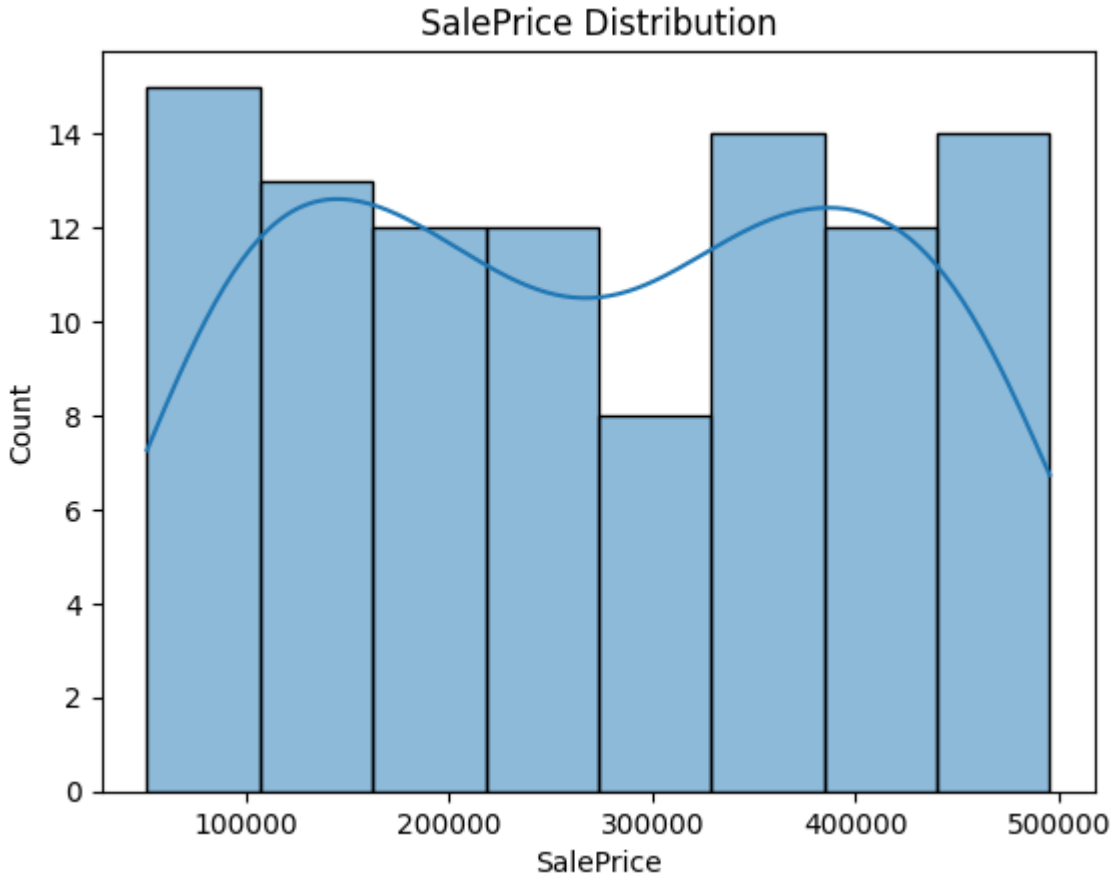
In [ ]:
```
# 4. Check for Missing Values and Duplicates
print("Missing values:\n", df.isnull().sum())
print("Duplicate rows:", df.duplicated().sum())
```

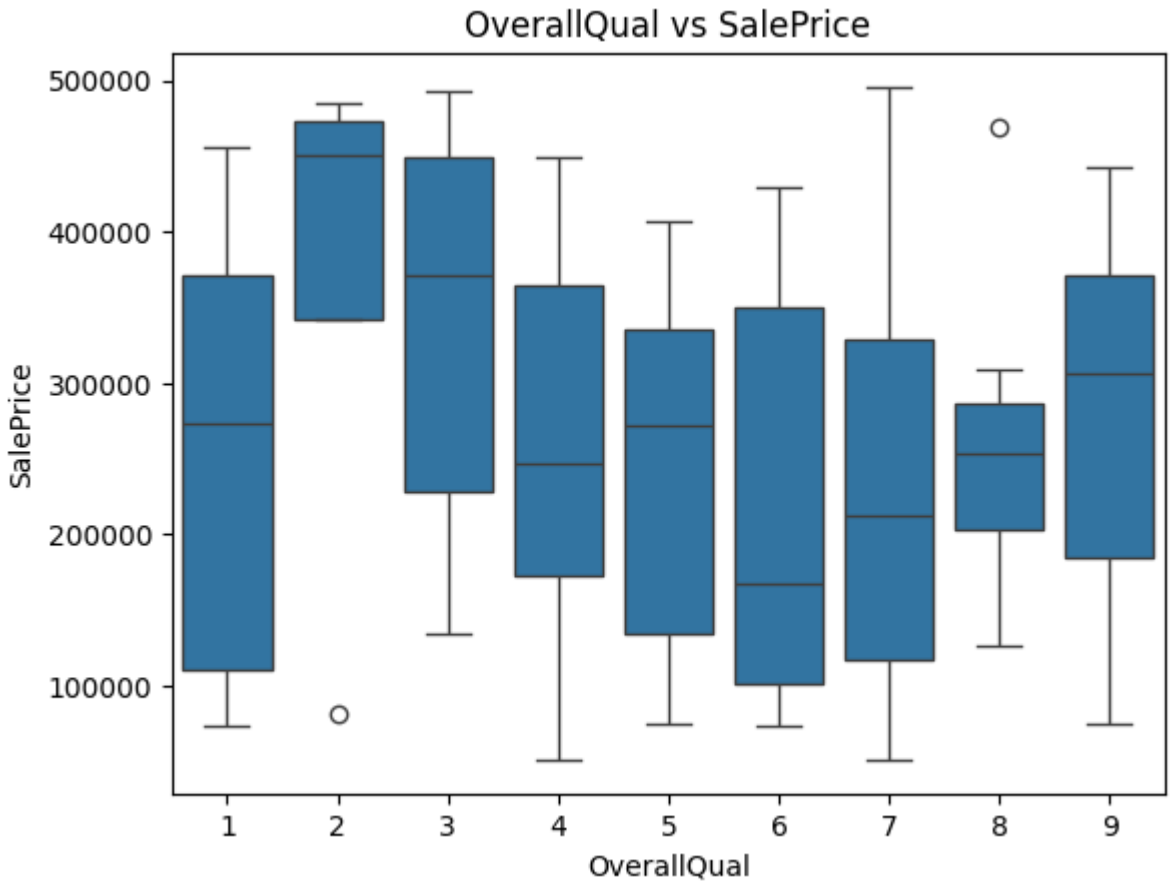
```
Missing values:
 LotArea         0
OverallQual     0
YearBuilt       0
TotRmsAbvGrd    0
GarageCars      0
FullBath        0
SalePrice       0
dtype: int64
Duplicate rows: 0
```

In [ ]:
```
# 5. Visualize a Few Features
import matplotlib.pyplot as plt
import seaborn as sns

sns.histplot(df['SalePrice'], kde=True)
plt.title("SalePrice Distribution")
plt.show()

sns.boxplot(x=df['OverallQual'], y=df['SalePrice'])
plt.title("OverallQual vs SalePrice")
plt.show()
```





In [ ]:
```
# 6. Identify Target and Features
X = df.drop("SalePrice", axis=1)
y = df["SalePrice"]
```

In [ ]:
```
# 7 & 8. Convert Categorical Columns to Numerical and One-Hot Encode
categorical_cols = X.select_dtypes(include=['object']).columns
if not categorical_cols.empty:
    X = pd.get_dummies(X, columns=categorical_cols, drop_first=True)
```

In [ ]:
```
# 9. Feature Scaling
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [ ]:
```
# 10. 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
```

In [ ]:
```
# 11. Model Building
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

Out[ ]:
```
▾LinearRegression
LinearRegression()
```

In [ ]:
```
# 12. Evaluation
from sklearn.metrics import mean_squared_error, r2_score

y_pred = model.predict(X_test)
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
Mean Squared Error: 19118292320.297195
R² Score: -0.08970262913123639
```

In [ ]:
```
# 13–15. Make Predictions, Convert Input to DataFrame, Predict Final Price
def predict_house_price(LotArea, OverallQual, YearBuilt, TotalBsmtSF, GrLi
                        GarageCars, GarageArea, FullBath, BedroomAbvGr, Ki
    input_data = pd.DataFrame([[LotArea, OverallQual, YearBuilt, TotalBsmt
                                GarageCars, GarageArea, FullBath, BedroomA
                               columns=X.columns)
    input_scaled = scaler.transform(input_data)
    prediction = model.predict(input_scaled)[0]
    return round(prediction, 2)
```

In [ ]:
```
# Prediction Function (for sample_house_prices.csv)
def predict_house_price(LotArea, OverallQual, YearBuilt, TotRmsAbvGrd, Ga
    input_data = pd.DataFrame([[LotArea, OverallQual, YearBuilt, TotRmsAb
                               columns=['LotArea', 'OverallQual', 'YearBui

    input_scaled = scaler.transform(input_data)
    prediction = model.predict(input_scaled)
    return prediction[0]

# Example Prediction
predicted_price = predict_house_price(8000, 7, 2005, 6, 2, 2)
print("Predicted House Price: ${:,.2f}".format(predicted_price))
```

```
Predicted House Price: $307,317.35
```