from google. colab import files uploaded = files . upload()

|  |
| --- |
| Choose Files |

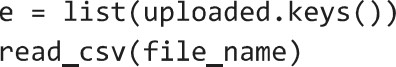
No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

qnvinø 1-4011<i nø I-4n11<i nø (l h

Double-click (or enter) to edit

import pandas as pd import numpy as np

# Load Dataset

# Access the uploaded file from the dictionary returned by files. upload() file name = [0] # Assuming only one file was uploaded df = pd .

# Initial Inspection print(df. shape) print(df. info())

# Handle missing values df. dropna (axis=0)

# Drop duplicates if any df. drop\_duplicates (inp1ace=True)

# Drop irrelevant columns if necessary (customize)

# df.drop( [ ' ld ' ], axis=l, inp1ace=True)

(545, 13)

(class ' pandas. core. frame. DataFrame' > Rangelndex: 545 entries, e to 544 Data columns (total 13 columns) :

 Column Non-Null Count Dtype



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | price | 545 | non-null | int64 |
| 1 | area | 545 | non-null | int64 |
| 2 | bedrooms | 545 | non-null | int64 |
| 3 | bathrooms | 545 | non-null | int64 |
| 4 | stories | 545 | non-null | int64 |
| 5 | mainroad | 545 | non-null | object |
| 6 | guestroom | 545 | non-null | object |
| 7 | basement | 545 | non-null | object |
| 8 | hotwaterheating | 545 | non-null | object |
| 9 | airconditioning | 545 | non-null | object |
| 10 | parking | 545 | non-null | int64 |
| 11 | prefarea | 545 | non-null | object |
| 12 | furnishingstatus | 545 | non-null | object |

dtypes: int64(6), object(7) memory usage: 55.5+ KB

None

import seaborn as sns import matplotlib. pyplot as plt

# Distribution of target

# Changed ' saleprice' to 'price' to match the actual column name in the DataFrame sns . histplot (df[ price ' ] , kde=True) plt.tit1e("House Price Distribution") plt. show()

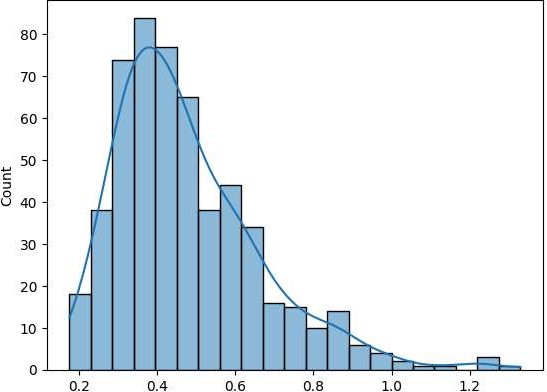
# Correlation heatmap plt . figure (figsize=(12, 10)) corr = df. select\_dtypes (include=np. number) . corr() sns. heatmap(corr, annot=True, fmt=' .2f cmap= ' coolwarm' ) plt.tit1e("Feature Correlation Matrix") plt. show()

# Boxplot: Price vs Overall Quality

# Changed 'SalePrice' to 'price' to match the actual column name in the DataFrame # Assuming the correct column name is 'OverallQua1ity' if 'OverallQua1ity• in df.columns:

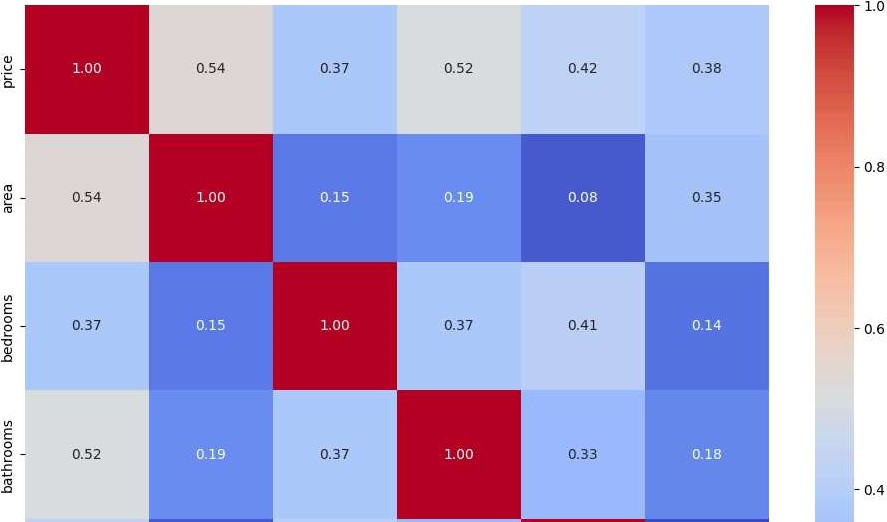
sns.boxplot(x= 'OverallQua1ity' , y= ' price' , data=df) plt.tit1e("Sa1e Price vs Overall Quality") # Updated title to reflect the corrected column name plt.show()

# House Price Distribution



price le7

# Feature Correlation Matrix



# Import Libraries import pandas as pd import numpy as np from sklearn.model\_selection import train\_test\_split from sklearn. preprocessing import StandardSca1er from sklearn. linear\_model import LinearRegression, Ridge, Lasso from sklearn. ensemble import RandomForestRegressor, GradientBoostingRegressor from metrics import mean\_squared\_error, r2\_score

# Load Dataset df — csv") # Ensure this file is in your working directory

# Binary Conversion (yes/no to I/e) binary\_cols — 'mainroad' , guestroom' , 'basement' , • hotwaterheating• , ' airconditioning' , ' prefarea df[binary\_ cols] x: 'yes' : I, 'no' :

# One-hot Encoding for ' furnishingstatus' df — pd. get\_dummies (df, columnsz[ • furnishingstatus'], drop\_firstzTrue)

# Features and Target X axis-I) y

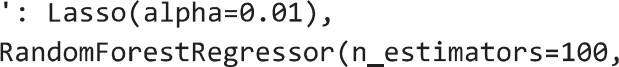
# Train-Test Split

X\_train, X\_test, y\_train, y\_test = train\_test\_sp1it(X, y,  . 2, random\_state=42)

# Feature Scaling scaler = StandardSca1er()

X train scaled = scaler.fit\_transform(X\_train) X test scaled = scaler . transform (X\_test)

# Define Models models

' Linear Regression' : LinearRegression() , 'Ridge Regression' Ridge (alpha=l.e), ' Lasso Regression' .

'Random Forest' :random\_state=42),

'Gradient Boosting' : GradientBoostingRegressor(n\_estimators=100,random\_state=42)

# Model Training and Evaluation results = for name, model in models . items() : model. fit (X\_train\_scaled, y\_train) predictions = model . predict (X\_test \_scaled) rmse = np . sq rt (mean \_squared\_error (y\_test , predictions)) r2 = r2\_score(y\_test, predictions) results [name] = { 'RMSE' : rmse, 'R2 Score' : r2}

# Display Results results\_df = pd.DataFrame(resu1ts) . T print("Mode1 Evaluation Metrics: ") sort\_values  Score" , ascending=Fa1se))

Model Evaluation Metrics:

RMSE R2 Score

Gradient Boosting 1.299761e+06 e. 665772 Linear Regression 1.324507e+06 e. 652924 Lasso Regression 1.324507e+€6 e. 652924 Ridge Regression 1.324703e+€6 e. 652821

Random Forest 1.400694e+06 e. 611848

import pandas as pd import numpy as np import matplotlib. pyplot as plt import seaborn as sns from sklearn. metrics import mean\_squared\_error, mean\_absolute\_error, r2 score

# Assuming models have been trained and stored like:

# models = {"Linear Regression" : model 1,

# and predictions have been made on X\_test \_ scaled

evaluation results

for name, model in models. items() : y\_pred = model. predict(x\_test\_scaled) rmse = np.  y\_pred)) mae = mean\_absolute\_error(y\_test, y\_pred) r2\_score(y\_test, y\_pred)

evaluation \_ results [name]

* RMSE' : rmse,

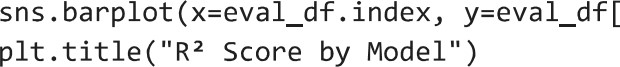
'MAE' : mae,

* R 2 Score'

# Convert results to DataFrame eval\_df = pd .  . T print ( " Model Evaluation Summary: " ) print(eval\_df. 'R 2 Score' , ascending-False))

Optional: Visualization

# Plot R 2 Score plt.figure(figsize=(le, 6))

"R 2 Score" ] , palette=" Blues\_d " )

plt.y1abe1("R 2 Score") plt . xticks ( rotation=45 )

plt . tight\_layout ( ) plt. show()

# Plot RMSE plt.figure(figsize=(1Ø, 6)) sns.barplot index, y=eval\_df["RMSE " ] , palette="Oranges\_d") plt.tit1e("RMSE by Model") plt.y1abe1("Root Mean Squared Error") plt . xticks ( rotation=45 ) plt . tight\_layout ( ) plt. show()

 Model Evaluation Summary :

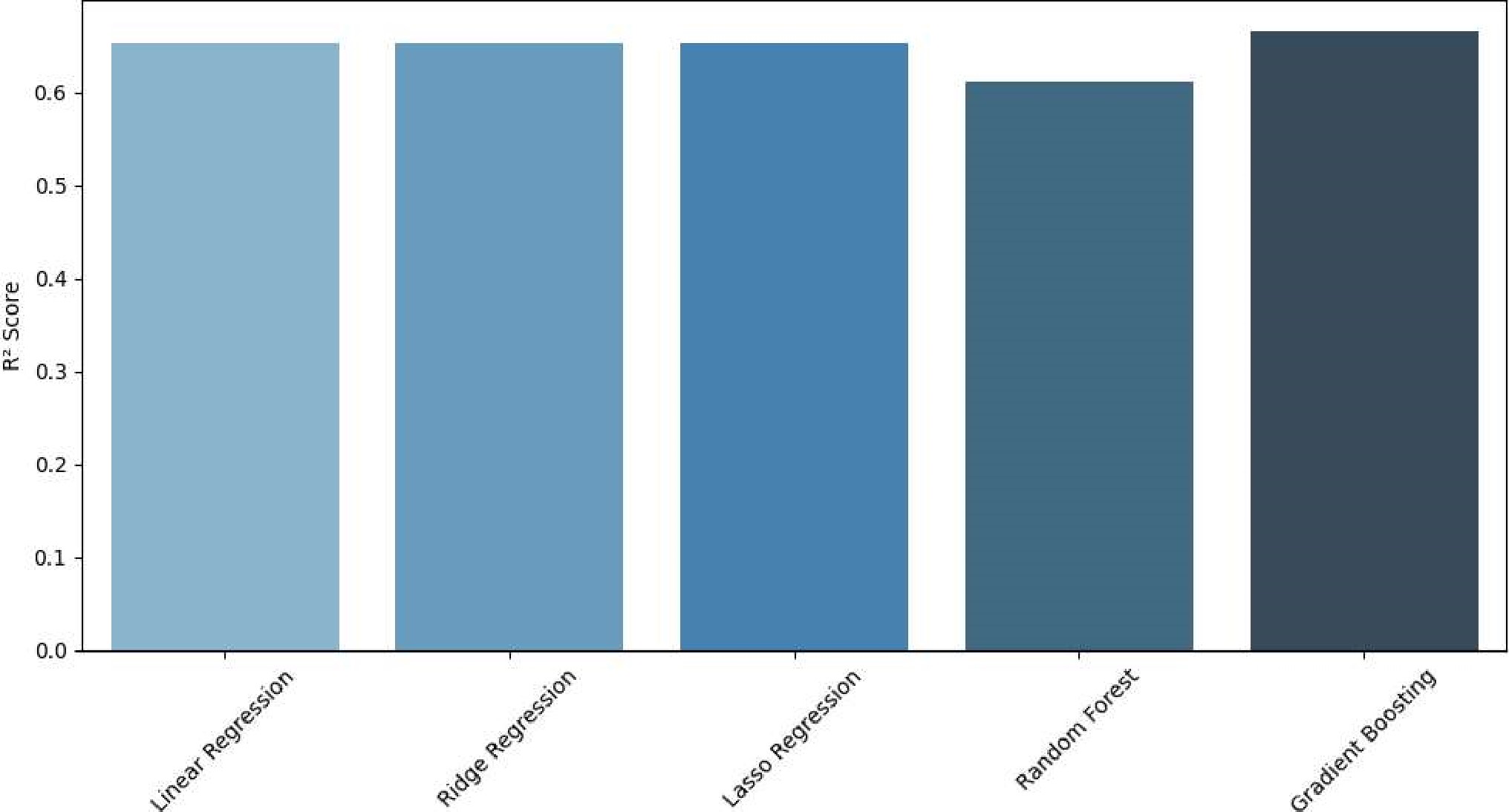
|  |  |  |  |
| --- | --- | --- | --- |
|  | RMSE | MAE | R 2 Score |
| Gradient Boosting | 1.299761e+06 | 9 .605788e+05 | e. 665772 |
| Linear Regression | 1.324507e+06 | 9 .700434e+05 | e. 652924 |
| Lasso Regression | 1.324507e+€6 | 9 .700434e+05 | e. 652924 |
| Ridge Regression | 1.324703e+06 | 9 .698579e+05 | e. 652821 |
| Random Forest | 1.400694e+Ø6 | 1 .022197e+06 | e. 611848 |

: 34 : FutureWarning:

Passing • palette\* without assigning is deprecated and will be removed in v0.14.0. Assign the x variable to hue\* and set leg

sns .  y=eva1\_df["R 2 Score"], palette="B1ues\_d")

# R2 Score by Model

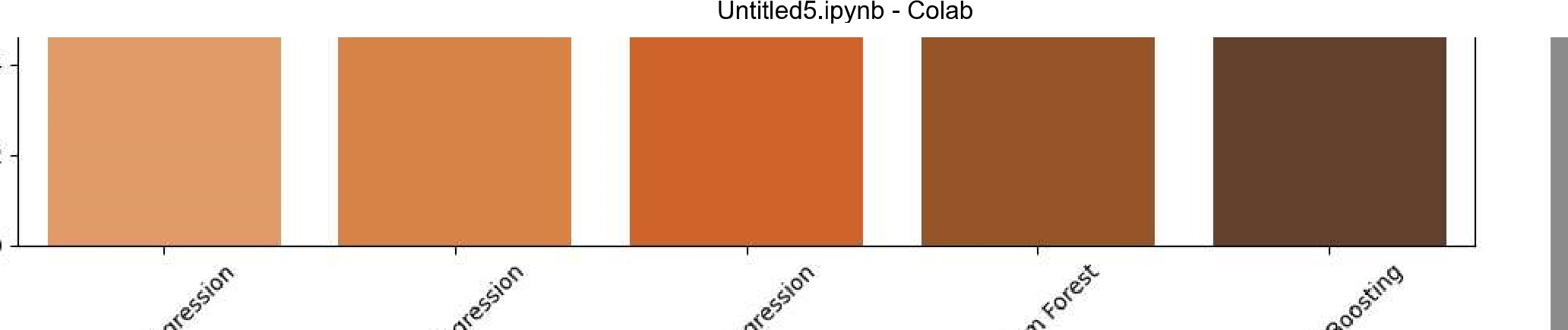


None

<ipython-input-8-31cae9603e61>:43: FutureWarning:

Passing palette' without assigning & hue is deprecated and will be removed in v0.14.0. Assign the x variable to hue and set leg

sns . "RMSE"], palette="Oranges\_d") le6 RMSE by Model

0.4

0.2

0.0