**PREDICTING HOUSE PRICE USING MACHINE LEARNING:**

**Phase-4 submission document**

**Project Title**: House Price Predictor

**Phase 4**: Development Part 2

**Topic**: Continue building the house price prediction model by feature engineering, model training, and evaluation.



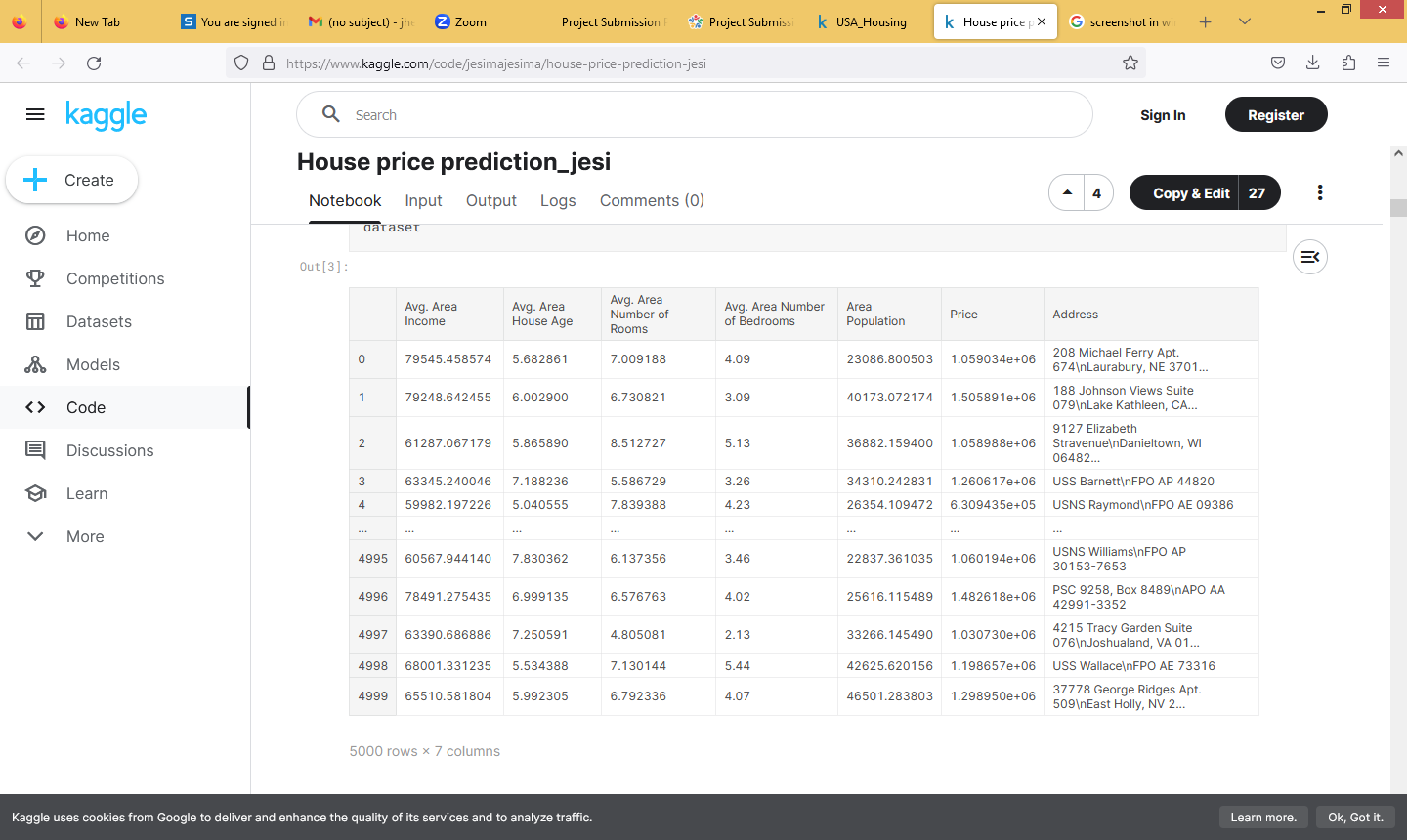
***House Price Prediction***

**Introduction:**

* The process of building a house price prediction model is a critical endeavor in the realm of real estate, finance, and property valuation. Accurately estimating the price of a house is essential for buyers, sellers, and investors to make informed decisions. In this comprehensive guide, we will continue to delve deeper into the construction of a robust house price prediction model by focusing on three fundamental components: feature selection, model training, and evaluation.
* Feature selection is the process of identifying and selecting the most relevant features from a dataset to improve the performance of a machine learning model. This is an important step in building a house price prediction model, as it can help to reduce over fitting and improve the generalization ability of the model.
* Model training is the process of feeding the selected features to a machine learning algorithm and allowing it to learn the relationship between the features and the target variable (i.e., house price). Once the model is trained, it can be used to predict the house prices of new houses, given their features.
* Model evaluation is the process of assessing theperformance of a trained machine learning model on a held-out test set.This is important to ensure that the model is generalizing well and that itis not overfitting the training data.

**Given data set:**

**Dataset Link:**[**https://www.kaggle.com/datasets/vedavyasv/usa-housing**](https://www.kaggle.com/datasets/vedavyasv/usa-housing)



**Feature Selection:**

We are selecting numerical features which have more than 0.50 or less than -0.50 correlation rate based on Pearson Correlation Method—which is the default value of parameter "method" in corr() function. As for selecting categorical features, I selected the categorical values which I believe have significant effect on the target variable such as Heating and MS Zoning.

**In [1]:**

important\_num\_cols = list(df.corr()["SalePrice"][(df.corr()["SalePrice"]>0.50) | (df.corr()["SalePrice"]<-0.50)].index)

cat\_cols = ["MSZoning", "Utilities","BldgType","Heating","KitchenQual"," SaleCondition","LandSlope"]

important\_cols = important\_num\_cols + cat\_cols

df = df[important\_cols]

**Checking for the missing values**

**In [2]:**

print("Missing Values by Column")

print("-"\*30)

print(df.isna().sum())

print("-"\*30)

print("TOTAL MISSING VALUES:",df.isna().sum().sum())

Missing Values by Column

------------------------------

OverallQual 0

YearBuilt 0

YearRemodAdd 0

TotalBsmtSF 0

1stFlrSF 0

GrLivArea 0

FullBath 0

TotRmsAbvGrd 0

GarageCars 0

GarageArea 0

SalePrice 0

MSZoning 0

Utilities 0

BldgType 0

Heating 0

KitchenQual 0

SaleCondition 0

LandSlope 0

dtype: int64

------------------------------

TOTAL MISSING VALUES: 0

**Model training:**

1. Choose a machine learning algorithm. There are a number of different machine learning algorithms that can be used for house price prediction, such as linear regression, ridge regression, lasso regression, decision trees, and random forests are Covered above.

**Machine Learning Models:**

**In [3]:**

models = pd.DataFrame(columns=["Model","MAE","MSE","RMSE","R2 Score","RMSE (Cross-Validation)"])

**Linear Regression:**

**In [4]:**

lin\_reg = LinearRegression()

lin\_reg.fit(X\_train, y\_train)

predictions = lin\_reg.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(lin\_reg)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "LinearRegression","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models = models.append(new\_row, ignore\_index=True)

**Out[4]:**

MAE: 23567.890565943395

MSE: 1414931404.6297863

RMSE: 37615.57396384889

R2 Score: 0.8155317822983865

-----------------------------

RMSE Cross-Validation: 36326.451444669496

**Ridge Regression:**

**In [5]:**

ridge = Ridge()ridge.fit(X\_train, y\_train)predictions = ridge.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(ridge)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "Ridge","MAE": mae, "MSE": mse, "RMSE": rmse,"R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}model s = models.append(new\_row, ignore\_index=True)

**Out[5]:**

MAE: 23435.50371200822

MSE: 1404264216.8595588

RMSE: 37473.513537691644

R2 Score: 0.8169224907874508

------------------------------

RMSE Cross-Validation: 35887.852791598336

**Lasso Regression:**

**In [6]:**

lasso = Lasso()lasso.fit(X\_train, y\_train)predictions = lasso.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(lasso)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "Lasso","MAE": mae, "MSE": mse, "RMSE": rmse,"R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}model s = models.append(new\_row, ignore\_index=True)

**Out[6]:**

MAE: 23560.45808027236

MSE: 1414337628.502095

RMSE: 37607.680445649596

R2 Score: 0.815609194407292

------------------------------

RMSE Cross-Validation: 35922.76936876075

**Elastic Net:**

**In [7]:**

elastic\_net = ElasticNet()elastic\_net.fit(X\_train, y\_train)predictions = elastic\_net.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(elastic\_net)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "ElasticNet","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models = models.append(new\_row, ignore\_index=True)

**Out[7]:**

MAE: 23792.743784996732

MSE: 1718445790.1371393

RMSE: 41454.14080809225

R2 Score: 0.775961837382229

------------------------------

RMSE Cross-Validation: 38449.00864609558

**Support Vector Machines:**

**In [8]:**

svr = SVR(C=100000)svr.fit(X\_train, y\_train)predictions = svr.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(svr)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "SVR","MAE": mae, "MSE": mse, "RMSE": rmse, " R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models = models.append(new\_row, ignore\_index=True)

**Out[9]:**

MAE: 17843.16228084976

MSE: 1132136370.3413317

RMSE: 33647.234215330864

R2 Score: 0.852400492526574

------------------------------

RMSE Cross-Validation: 30745.475239075837

**Random Forest Regressor:**

**In [9]:**

random\_forest = RandomForestRegressor(n\_estimators=100)random\_forest. fit(X\_train, y\_train)predictions = random\_forest.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(random\_forest)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "RandomForestRegressor","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models = models.append(new\_row, ignore\_index=True)

**Out[9]:**

MAE: 18115.11067351598

MSE: 1004422414.0219476

RMSE: 31692.623968708358

R2 Score: 0.869050886899595

------------------------------

RMSE Cross-Validation: 31138.863315259332

**XGBoost Regressor:**

**In [10]:**

xgb = XGBRegressor(n\_estimators=1000, learning\_rate=0.01)xgb.fit(X\_train, y\_train)predictions = xgb.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(xgb)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "XGBRegressor","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models = models.append(new\_row, ignore\_index=True)

**Out[10]:**

MAE: 17439.918396832192

MSE: 716579004.5214689

RMSE: 26768.993341578403

R2 Score: 0.9065777666861116

-----------------------------

RMSE Cross-Validation: 29698.84961808251

**Polynomial Regression (Degree=2)**

**In [11]:**

poly\_reg = PolynomialFeatures(degree=2)X\_train\_2d = poly\_reg.fit\_transform(X\_train)X\_test\_2d = poly\_reg.transform(X\_test)

lin\_reg = LinearRegression()lin\_reg.fit(X\_train\_2d, y\_train)predictions = lin\_reg.predict(X\_test\_2d)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(lin\_reg)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "Polynomial Regression (degree=2)","MAE": mae, " MSE": mse, "RMSE": rmse, "R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models = models.append(new\_row, ignore\_index=True)

**Out[11]:**

MAE: 2382228327828308.5

MSE: 1.5139911544182342e+32

RMSE: 1.230443478758059e+16

R2 Score: -1.9738289005226644e+22

------------------------------

RMSE Cross-Validation: 36326.451444669496

**Dividing Dataset in to features and target variable:**

**In [12]:**

X = dataset[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number ofRooms', 'Avg. Area Number of Bedrooms', 'Area Population']]

Y = dataset['Price']

2. **Split the data into training and test sets.** The training set will be used to train the model, and the test set will be used to evaluate the performance of the model.

**In [13]:**

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=101)

**In [14]:**

Y\_train.head()

**Out[14]:**

3413 1.305210e+06

1610 1.400961e+06

3459 1.048640e+06

4293 1.231157e+06

1039 1.391233e+06

Name: Price, dtype: float64

**In [15]:**

Y\_train.shape

**Out[15]:**

(4000,)

In [16]:

Y\_test.head()

**Out[16]:**

1718 1.251689e+06

2511 8.730483e+05

345 1.696978e+06

2521 1.063964e+06

54 9.487883e+05

Name: Price, dtype: float64

**In [17]:**

Y\_test.shape

**Out[17]:** (1000)

3. **Train the model on the training set.** This involves feeding the training data to the model and allowing it to learn the relationships between the features and the target variable.

4. **Evaluate the model on the test set**. This involves feeding the test data to the model and measuring how well it predicts the target variable.

**Evaluation of Predicted Data:**

**In [18]:**

plt.figure(figsize=(12,6))

plt.plot(np.arange(len(Y\_test)), Y\_test, label='Actual Trend')

plt.plot(np.arange(len(Y\_test)), Prediction5, label='Predicted Trend')

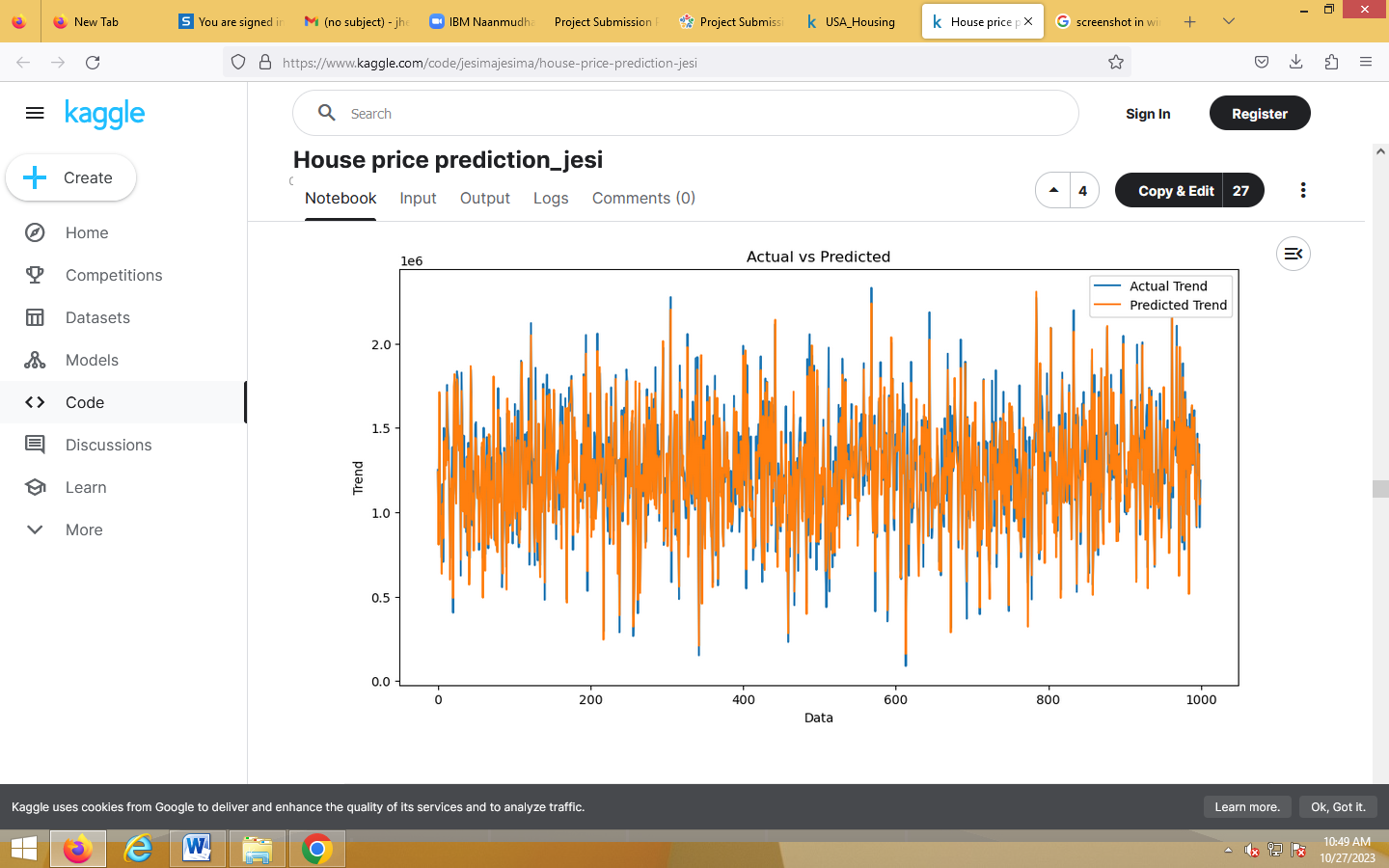
plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

**Out[18]:**Text(0.5, 1.0, 'Actual vs Predicted')

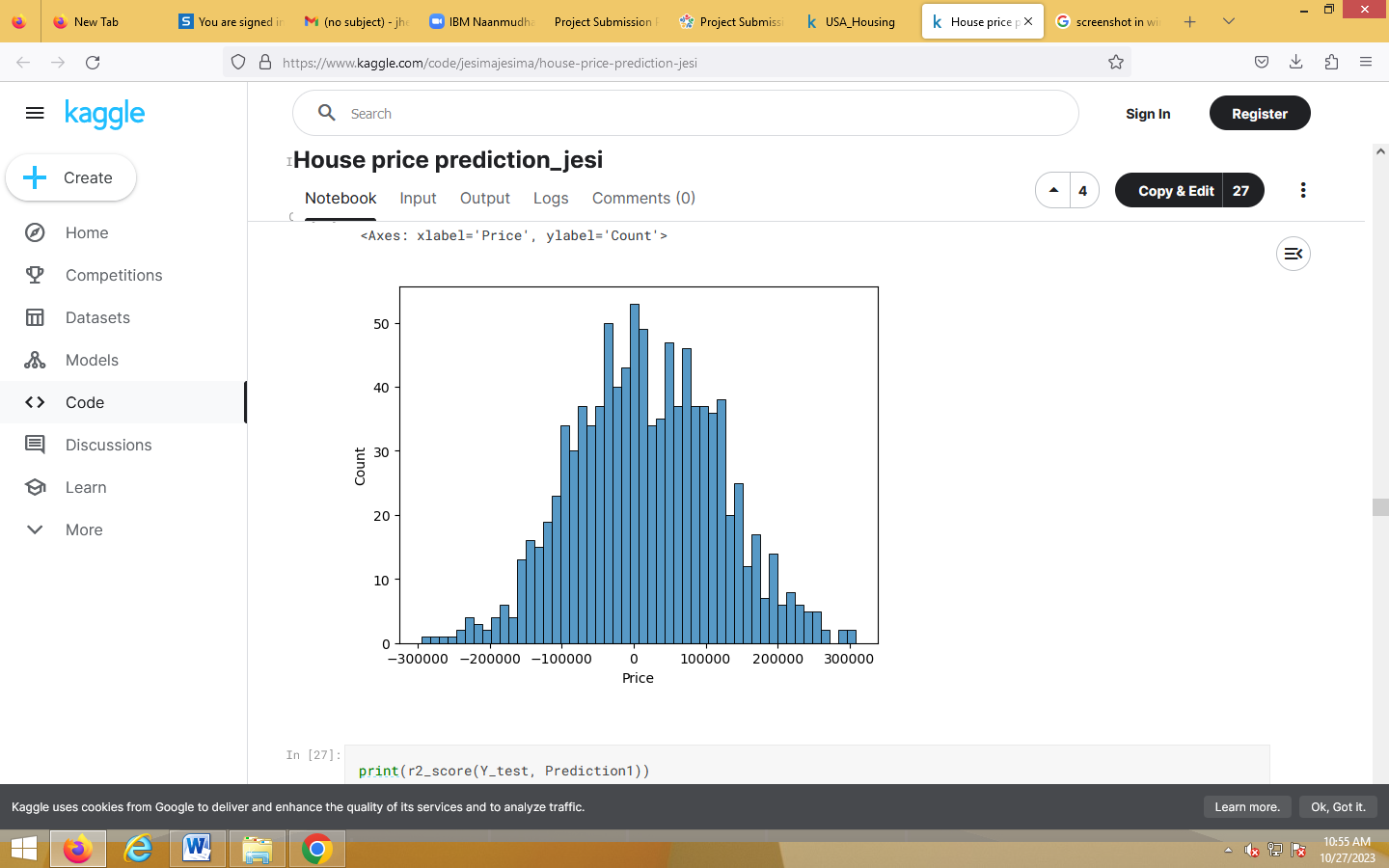


**In [19]:**

sns.histplot((Y\_test-Prediction4), bins=50)

**Out[19]:**

<Axes: xlabel='Price', ylabel='Count'>



**In [20]:**

print(r2\_score(Y\_test, Prediction2))

print(mean\_absolute\_error(Y\_test, Prediction2))

print(mean\_squared\_error(Y\_test, Prediction2))

**Out[20]:**

-0.0006222175925689744

286137.81086908665

128209033251.4034

**Model Comparison:**

The less the Root Mean Squared Error (RMSE), The better the model is.

**In [30]:**

models.sort\_values(by="RMSE (Cross-Validation)")

**Out[30]:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Model** | **MAE** | **MSE** | **RMSE** | **R2 Score** | **RMSE**  **(Cross-Validation)** |
| 6 | XGBRegressor | 1.743992e+04 | 7.165790e+08 | 2.676899e+04 | 9.065778e-01 | 29698.849618 |
| 4 | SVR | 1.784316  e+04 | 1.132136  e+09 | 3.364723  e+04 | 8.524005  e-01 | 30745.475239 |
| 5 | RandomForestRe  gressor | 1.811511  e+04 | 1.004422  e+09 | 3.169262  e+04 | 8.690509  e-01 | 31138.863315 |
| 1 | Ridge | 2.343550  e+04 | 1.404264  e+09 | 3.747351  e+04 | 8.169225  e-01 | 35887.852792 |
| 2 | Lasso | 2.356046  e+04 | 1.414338  e+09 | 3.760768  e+04 | 8.156092  e-01 | 35922.769369 |
| 0 | LinearRegression | 2.356789  e+04 | 1.414931  e+09 | 3.761557  e+04 | 8.155318  e-01 | 36326.451445 |
| 7 | Polynomial  Regression  (degree=2) | 2.382228  e+15 | 1.513991  e+32 | 1.230443  e+16 | -1.973829  e+22 | 36326.45  1445 |
| 3 | ElasticNet | 2.379274  e+04 | 1.718446  e+09 | 4.145414  e+04 | 7.759618  e-01 | 38449.00  8646 |

**In [31]:**

plt.figure(figsize=(12,8))

sns.barplot(x=models["Model"], y=models["RMSE (Cross-Validation)"])

plt.title("Models' RMSE Scores (Cross-Validated)", size=15)

plt.xticks(rotation=30, size=12)

plt.show()

