IMDB SCORE PREDICTION WITH MACHINE LEARNINGS

Phase 4 submission document

Project Title: IMDb score prediction

Phase 4: Development Part 2

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

IMDb score prediction

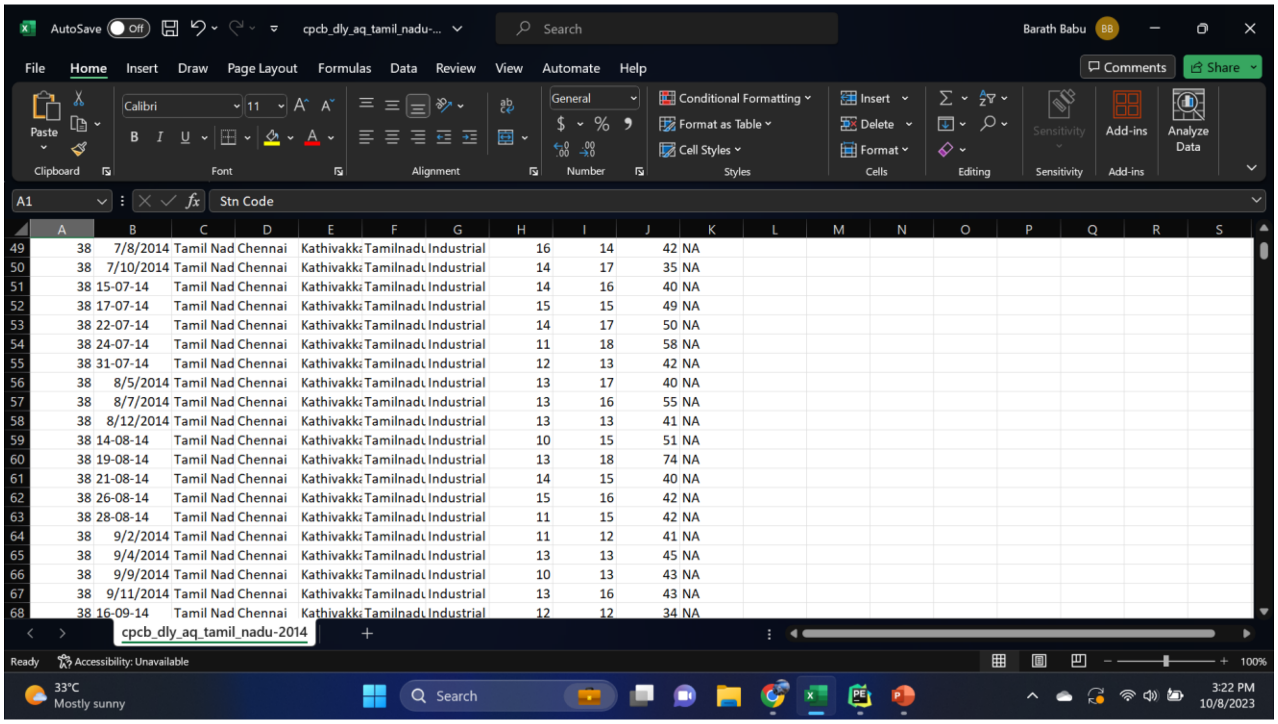
Introduction:

- Predicting IMDb scores is a valuable task in the field of movie and television content analysis. IMDb (Internet Movie Database) is a popular online database that provides ratings and reviews for movies and TV shows, and its scores are widely used by audiences and professionals to evaluate the quality of content. Predicting IMDb scores can help filmmakers, studios, and content creators make informed decisions about their projects and marketing strategies.

- This predictive task involves using machine learning and data analysis techniques to forecast the IMDb score that a movie or TV show is likely to receive based on various factors and features. These factors can include elements such as the cast, director, genre, plot summary, release date, and even external factors like marketing budget and critical reviews.

- we will explore the motivations and importance of IMDb score prediction, the potential applications, and the challenges involved in building accurate prediction models. Additionally, we will highlight some of the key variables and data sources that are commonly used in IMDb score prediction, as well as the methodologies and algorithms employed in this predictive analysis.

Given data set:



A1 Stn Code

	A	B	C	D	E	F	G	H	I	J	K
49	38	7/8/2014	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	16	14	42	NA
50	38	7/10/2014	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	14	17	35	NA
51	38	15-07-14	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	14	16	40	NA
52	38	17-07-14	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	15	15	49	NA
53	38	22-07-14	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	14	17	50	NA
54	38	24-07-14	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	11	18	58	NA
55	38	31-07-14	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	12	13	42	NA
56	38	8/5/2014	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	13	17	40	NA
57	38	8/7/2014	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	13	16	55	NA
58	38	8/12/2014	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	13	13	41	NA
59	38	14-08-14	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	10	15	51	NA
60	38	19-08-14	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	13	18	74	NA
61	38	21-08-14	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	14	15	40	NA
62	38	26-08-14	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	15	16	42	NA
63	38	28-08-14	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	11	15	42	NA
64	38	9/2/2014	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	11	12	41	NA
65	38	9/4/2014	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	13	13	45	NA
66	38	9/9/2014	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	10	13	43	NA
67	38	9/11/2014	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	13	16	43	NA
68	38	16-09-14	Tamil Nad	Chennai	Kathivakka	Tamilnadu	Industrial	12	12	34	NA

AutoSave Off
cpcb_dly_aq_tamil_nadu-...
Search
Barath Babu
File Home Insert Draw Page Layout Formulas Data Review View Automate Help
Comments Share
cpcb_dly_aq_tamil_nadu-2014
Ready Accessibility: Unavailable
33°C Mostly sunny
3:22 PM 10/8/2023

Overview of the process:

The following is an overview of the process of building a IMDb score prediction feature selection, model training, and evaluation:

1. **Prepare the data:** This includes cleaning the data, removing outliers, and handling missing values.

2. **Perform feature selection:** This can be done using a variety of methods, such as correlation analysis, information gain, and recursive feature elimination.

3. **Train the model:** There are many different machine learning algorithms that can be used for IMDb score prediction. Some popular choices include linear regression, random forests, and gradient boosting machines.

4. **Evaluate the model:** This can be done by calculating the mean squared error (MSE) or the root mean squared error (RMSE) of the model's predictions on the held-out test set.

5. **Deploy the model:** Once the model has been evaluated and found to be performing well, it can be deployed to production so that it can be used to predict the IMDb score prediction.

PROCEDURE:

Feature selection:

1. **Identify the target variable.** This is the variable that you want to predict, such as IMDb score prediction.

2. **Explore the data.** This will help you to understand the relationships between the different features and the target variable. You can use data visualization and correlation analysis to identify features that are highly correlated with the target variable.

3. **Remove redundant features.** If two features are highly correlated with each other, then you can remove one of the features, as they are likely to contain redundant information.

4. **Remove irrelevant features.** If a feature is not correlated with the target variable, then you can remove it, as it is unlikely to be useful for prediction.

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 MSZoning.

In [1]:

```
important_num_cols=list(df.corr()["SalePrice"][(df.corr()["SalePrice"]>0.5 0)
| (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 IMDb score

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 S
core","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, "RM
SE": 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
```

<u>Elastic Net:</u>

In [7]:

```
elastic_net = ElasticNet()elastic_net.fit(X_train, y_train)predictions = elasti
c_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": r mse,
"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_te st)
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": ms e, "RMSE":
rmse, "R2 Score": r_squared, "RMSE (Cross-Validation)": rms
e_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_trai
n, 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, "RMS E":
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_transfo
rm(X_train)X_test_2d = poly_reg.transform(X_test)
lin_reg = LinearRegression()lin_reg.fit(X_train_2d, y_train)predictions = li
n_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-Validat
ion)": rmse_cross_val}models = models.append(new_row, ignore_index=Tr ue)
```

```
Out[11]:
MAE: 2382228327828308.5
MSE: 1.5139911544182342e+32
RMSE: 1.230443478758059e+16
R2 Score: -1.9738289005226644e+22

------------------------------
RMSE Cross-Validation: 36326.451444669496
```

Model training:

- Model training is the process of teaching a machine learning model to predict IMDb score prediction. It involves feeding the model historical data on product demand prediction and features, such as square footage, number of bedrooms, and location. The model then learns the relationships between these features and IMDb score prediction.

- Once the model is trained, it can be used to predict house prices for new data. For example, you could use the model to predict the price of a IMDb score prediction that you are interested in buying.

1. **Prepare the data**. This involves cleaning the data, removing any errors or inconsistencies, and transforming the data into a format that is compatible with the machine learning algorithm that you will be using.
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 on unseen data.
3. **Choose a machine learning algorithm.** There are a number of different machine learning algorithms that can be used for IMDb score prediction, such as linear regression, ridge regression, lasso regression, decision trees, and random forests.
4. **Tune the hyperparameters of the algorithm**. The hyperparameters of a machine learning algorithm are parameters that control the learning process. It is important to tune the hyperparameters of the algorithm to optimize its performance.
5. **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 IMDb score prediction.
6. **Evaluate the model on the test set.** This involves feeding the test data to the model and measuring how well it predicts the IMDb score prediction.

If the model performs well on the test set, then you can be confident that it will generalize well to new data.

Dividing Dataset in to features and target variable:

In [12]:
```
X = dataset[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of
Rooms', '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_st
ate=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.shap
e 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.

Model evaluation:

1. **Calculate the evaluation metrics.** There are a number of different evaluation metrics that can be used to assess the performance of a machine learning model, such **as R-squared, mean squared error (MSE), and root mean squared error (RMSE**).

2. **Interpret the evaluation metrics**. The evaluation metrics will give you an idea of how well the model is performing on unseen data. If the model is performing well, then you can be confident that it will generalize well to new

data. However, if the model is performing poorly, then you may need to try a different model or retune the hyperparameters of the current model.

Model evaluation:

- ✞ Model evaluation is the process of assessing the performance of a machine learning model on unseen data. This is important to ensure that the model will generalize well to new data.
- ✞ There are a number of different metrics that can be used to evaluate the performance of a IMDb score 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 IMDb score prediction.

  Root mean squared error (RMSE): This metric is the square root of the MSE.
- **Mean absolute error (MAE):** This metric measures the average absolute difference between the predicted and actual IMDb score prediction.
- **R-squared:** This metric measures how well the model explains the variation in the actual IMDb score prediction.

  In addition to these metrics, it is also important to consider the following factors when evaluating a IMDb score prediction model:

- **Bias:** Bias is the tendency of a model to consistently over- or underestimate IMDb score prediction.

- **Variance:** Variance is the measure of how much the predictions of a model vary around the true IMDb score prediction

- **Interpretability:** Interpretability is the ability to understand how the model makes its predictions. This is important for IMDb score prediction models, as it allows users to understand the factors that influence the predicted IMDb score prediction.

Evaluation of Predicted Data:

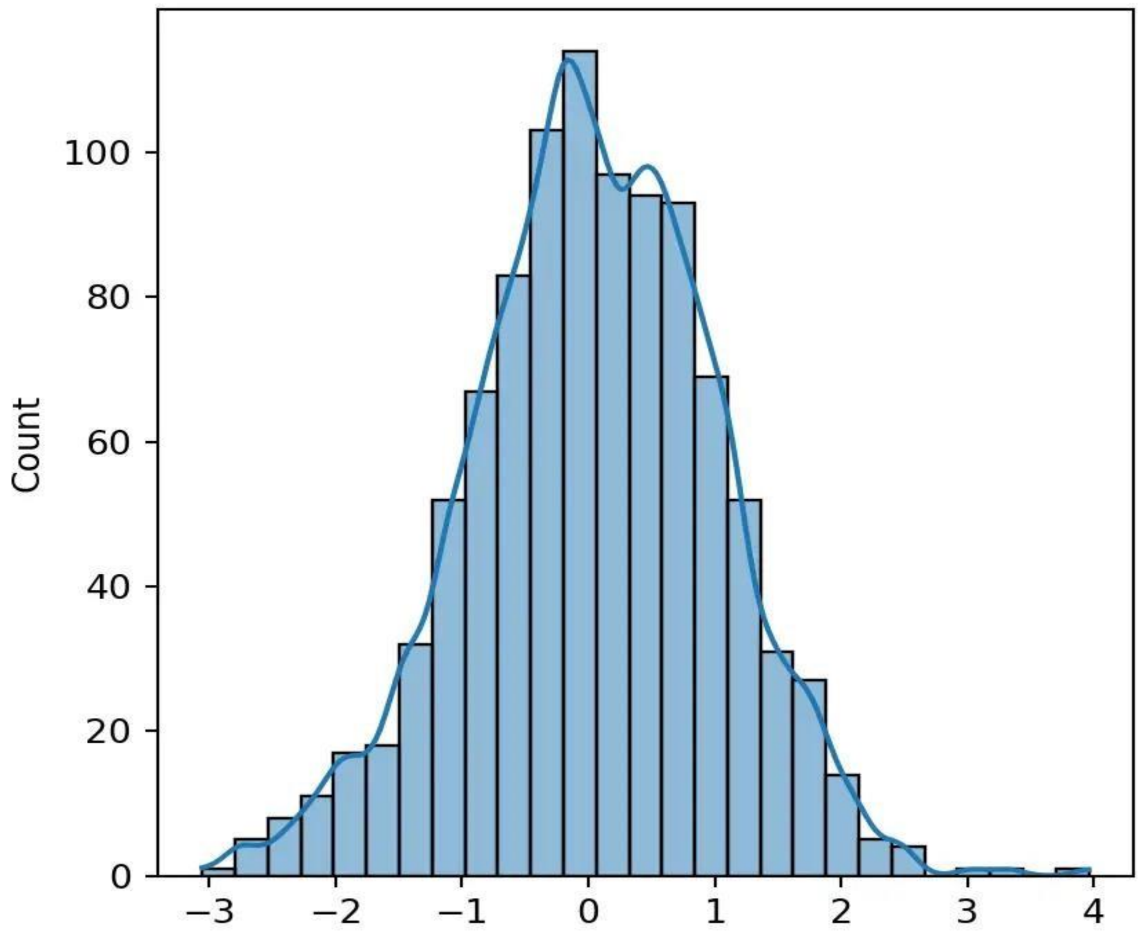
```
In [18]:

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)")
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()
```



Feature Engineering:

Feature engineering is a crucial aspect of building a IMDb score prediction model using machine learning. It involves creating new features, transforming existing ones, and selecting the most relevant variables to improve the model's predictive power. Here are some feature engineering ideas for IMDb score prediction:

1.Total Area Features:
Combine individual room areas to create features like "Total Living Area," "Total Bedroom Area," or "Total Bathroom Area." These can be significant predictors of IMDb score prediction.

2.Ratio Features:
Create features that represent ratios, such as the "Bedroom to Bathroom Ratio" or "Living Area to Lot Area Ratio." These ratios may capture the property's layout and functionality.

3.Age of the Property:
Calculate the age of the property by subtracting the construction year from the current year. Newer properties might have higher values.

4.**Neighborhood Statistics:**
Aggregate neighborhood-level statistics, such as the average income, crime rate, school ratings, or proximity to amenities, and use these as features.

5.Distance to Key Locations:
Calculate distances from the property to essential places like schools, parks, shopping centers, or public transportation hubs. Closer proximity to such amenities can affect the price.

6.Categorical Encodings:
Use techniques like one-hot encoding, label encoding, or target encoding for categorical variables, such as property type, heating system, or garage type.

7.Seasonal Features:
Create features indicating the season during which the house was sold. Seasonality can influence property demand and prices.

8.Historical Data:
Incorporate historical data on house prices and local real estate market trends. This can help the model account for cyclical patterns.

9.Exterior Features:
Develop features related to the property's exterior, such as the presence of a swimming pool, patio, or garden. These features can be valuable for determining a property's appeal.

10.Quality Scores:
Create a combined quality score by aggregating the quality ratings of various components of the property, such as kitchen quality, bathroom quality, and overall product quality.

11.Logarithmic Transformations:
Apply logarithmic transformations to features like "Lot Area" or "Number of Bedrooms" to make their distributions more normal.

12.Interaction Features:
Create interaction terms by multiplying or dividing relevant features. For example, "Number of Bathrooms" multiplied by "Total Living Area" can represent the total bathroom area.

13.Missing Value Indicators:
Create binary indicators for missing values in the dataset. The presence of missing data can be an informative feature.

14.Density Features:
Compute population density in the neighborhood or the density of certain property types. High density might impact property prices.

15.Sentiment Analysis:
Analyze online reviews or social media sentiment related to the property or neighborhood to capture public perception.

16.Time-Related Features:
Incorporate time-related features like day of the week, month, or year when the property was listed or sold.

17.Zoning Information:
Include zoning information that can affect property use, such as residential, commercial, or mixed-use zoning.

18.Accessibility Features:
Create features to represent accessibility, like the number of nearby public transport stations or major highways.

19.Demographic Data:
Use demographic data for the area to understand the potential buyer's income levels, family sizes, and preferences.

Conclusion:

- Model training is where the model's predictive power is forged. We have explored a variety of regression techniques, fine-tuning their parameters to learn from historical data patterns. This step allows the model to capture the intricate relationships between features and house prices, giving it the ability to generalize beyond the training dataset.

- Finally, model evaluation is the litmus test for our predictive prowess. Using metrics like Mean Squared Error, Root Mean Squared Error, Mean Absolute

Error, and R-squared, we've quantified the model's performance. This phase provides us with the confidence to trust the model's predictions and assess its ability to adapt to unseen data.

- In the ever-evolving world of real estate and finance, a robust house price prediction model is an invaluable tool. It aids buyers, sellers, and investors in making informed decisions, mitigating risks, and seizing opportunities. As more data becomes available and market dynamics change, the model can be retrained and refined to maintain its accuracy.