House Prices Prediction

Course: Data Science for Business I

Instructor: Pekka Malo, Antti Suominen (TA), Laroslav Kriuchkov (TA)

Presenters: Naiyue Zhu, Baining Zhang, Jie Zhou, Tinh Ta (Group 10)

What factors in this database affect housing price?

Aim of Analysis

1. Create an effective price prediction model

2. Validate the model's prediction accuracy

3. Identify the important home price attributes which feed the model's predictive power.

The study was conducted to provide valuable insights into the complex factors influencing housing prices, and the benefits are wide-ranging, including improved decision-making for all stakeholders and the potential to inform more effective housing policies and development strategies.

The study on factors affecting housing prices was conducted for several reasons:

1. Understanding local housing market
2. Knowledge about what factors will affect housing price in this market
3. Implication of economies and policies
4. For home buyers and sellers have better understanding about whether the price is reasonable or not

Who benefits from the results

1. Home buyers and sellers
2. Urban planners
3. Potential investors

Bottom line of this study

1. A list of the most significant factors affecting housing prices, such as units of area, garage, construction years, etc.
2. An analysis of how these factors interact and their relative importance.
3. Recommendations or insights on how different stakeholders can leverage this information to their advantage.

How can it be implemented?

1. Home buyers and sellers: by using this results, they can make better decisions about prices and negotiation.
2. Urban planner: using results to update or perfect policies. For factors that should have adjustments, planners could bring up solutions such as taxes.
3. Potential investors: they can incorporate this information into their strategies. For example, they may target specific neighborhoods or property types based on the factors revealed in the study.

Content

Introduction to Dataset

Methods for Prediction

Regression Models

Results of Analysis

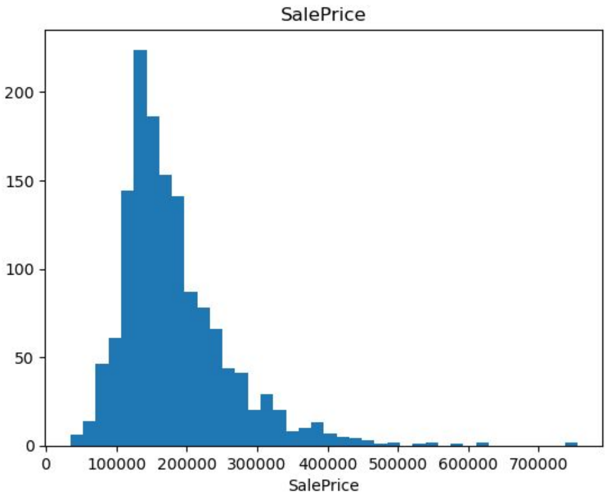
Conclusion and Application

Introduction to dataset

Description of dataset

This dataset contains 74 explanatory variables related to residential home sales prices in Ames, Iowa.

Overall salesprice distribution shows house prices in this dataset are generally concentrated at around 1,50000 USD



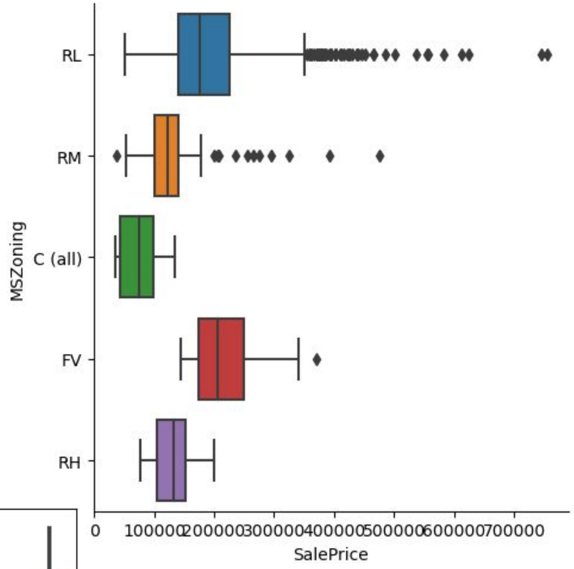

```
df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 1459 entries, 0 to 1459
Data columns (total 74 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MSSubClass     1459 non-null   int64
 1   MSZoning       1459 non-null   object
 2   LotFrontage    1459 non-null   float64
 3   LotArea        1459 non-null   int64
 4   Street         1459 non-null   object
 5   Alley          1459 non-null   object
 6   LotShape       1459 non-null   object
 7   LandContour    1459 non-null   object
 8   Utilities      1459 non-null   object
 9   LotConfig      1459 non-null   object
 10  LandSlope      1459 non-null   object
 11  Neighborhood   1459 non-null   object
 12  Condition1     1459 non-null   object
 13  Condition2     1459 non-null   object
 14  BldgType       1459 non-null   object
 15  HouseStyle     1459 non-null   object
 16  OverallQual    1459 non-null   int64
 17  OverallCond    1459 non-null   int64
 18  YearBuilt      1459 non-null   int64
 19  RoofStyle      1459 non-null   object
 20  RoofMatl       1459 non-null   object
 21  Exterior1st    1459 non-null   object
 22  Exterior2nd    1459 non-null   object
 23  ExterQual      1459 non-null   object
4   ExterCond      1460 non-null   object
5   Foundation     1460 non-null   object
6   BsmtQual       1460 non-null   object
7   BsmtCond       1460 non-null   object
8   BsmtExposure   1460 non-null   object
9   BsmtFinType1   1460 non-null   object
0   BsmtFinSF1     1460 non-null   int64
1   BsmtFinType2   1460 non-null   object
2   BsmtFinSF2     1460 non-null   int64
3   BsmtUnfSF      1460 non-null   int64
4   TotalBsmtSF    1460 non-null   int64
5   Heating        1460 non-null   object
6   HeatingQC      1460 non-null   object
7   CentralAir     1460 non-null   object
8   Electrical     1459 non-null   object
9   1stFlrSF       1460 non-null   int64
0   2ndFlrSF       1460 non-null   int64
1   LowQualFinSF   1460 non-null   int64
2   GrLivArea      1460 non-null   int64
3   BsmtFullBath   1460 non-null   int64
4   BsmtHalfBath   1460 non-null   int64
5   FullBath       1460 non-null   int64
6   HalfBath       1460 non-null   int64
7   BedroomAbvGr   1460 non-null   int64
8   KitchenAbvGr   1460 non-null   int64
9   KitchenQual    1460 non-null   object
0   TotRmsAbvGrd   1460 non-null   int64
1   Functional     1460 non-null   object
2   Fireplaces     1460 non-null   int64
3   FireplaceQu    1460 non-null   object
4   GarageType     1460 non-null   object
5   GarageFinish   1460 non-null   object
6   GarageCars     1460 non-null   int64
7   GarageArea     1460 non-null   int64
58  GarageQual     1460 non-null   object
59  GarageCond     1460 non-null   object
60  PavedDrive     1460 non-null   object
61  WoodDeckSF     1460 non-null   int64
62  OpenPorchSF    1460 non-null   int64
63  EnclosedPorch  1460 non-null   int64
64  3SsnPorch      1460 non-null   int64
65  ScreenPorch    1460 non-null   int64
66  PoolArea       1460 non-null   int64
67  PoolQC         1460 non-null   object
68  Fence          1460 non-null   object
69  MiscFeature    1460 non-null   object
70  MiscVal        1460 non-null   int64
71  SaleType       1460 non-null   object
72  SaleCondition  1460 non-null   object
73  SalePrice      1460 non-null   int64
dtypes: float64(1), int64(31), object(42)
memory usage: 844.2+ KB
```

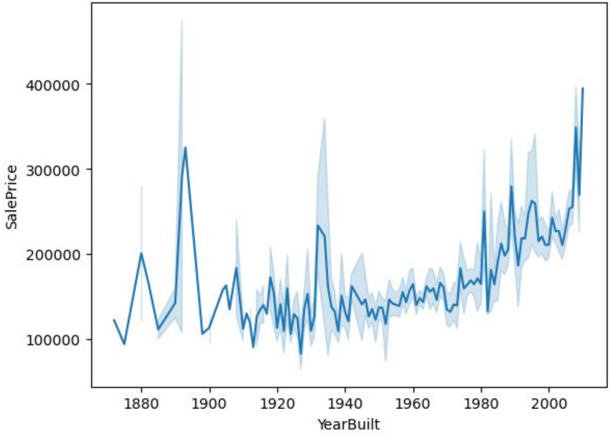
Descriptive Analysis

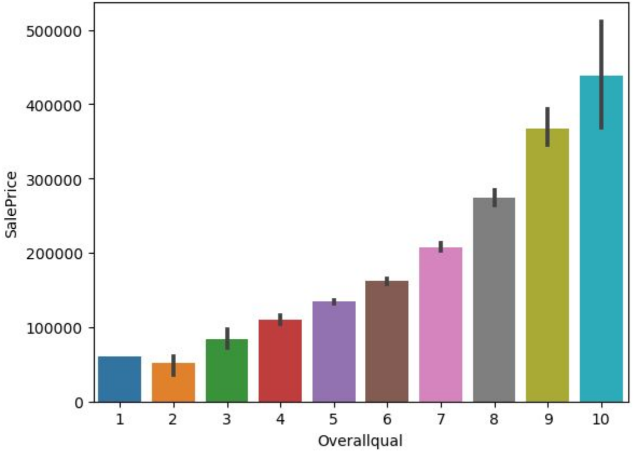
In descriptive analysis, selecting common features widely believed to influence house prices and examining their simple distributions provides an initial insight into the data's key factors.

Salesprice distribution of by housing types



Salesprice distribution by construction year



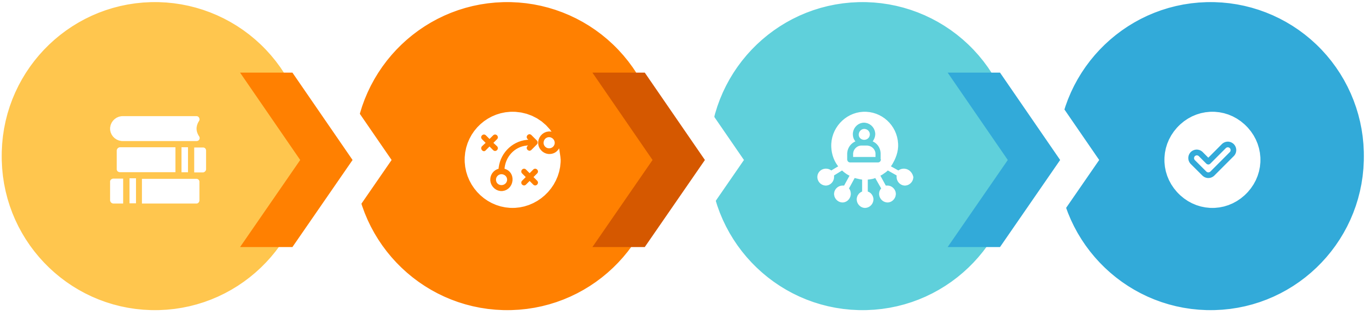


Salesprice distribution by house quality

Introduction to dataset

Data cleaning

Remove unnecessary variables	Change, remove, and fill in empty values	One-hot encoding non-ordinal variables	Ordinal variables encoding



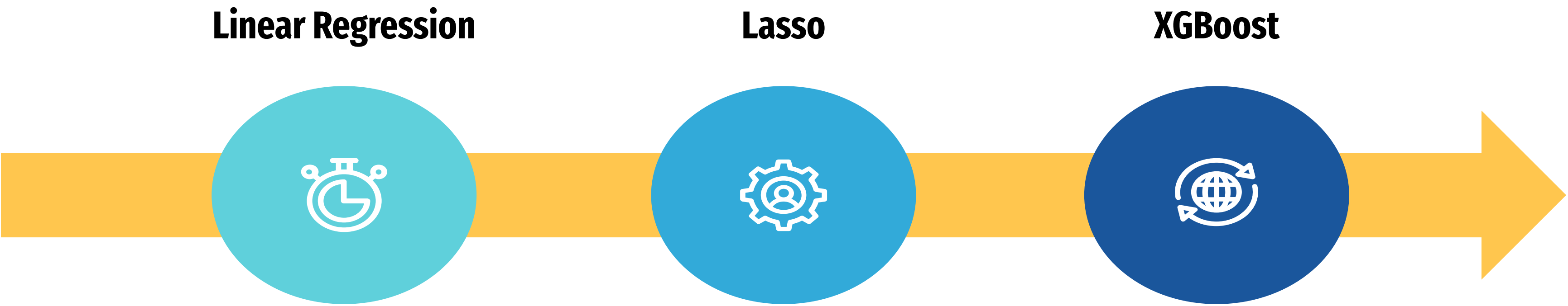
Remove unnecessary variables	Change, remove, and fill in empty values	One-hot encoding non-ordinal variables	Ordinal variables encoding
Remove variables like: ID, Month and Year sold, Remodeling year	Change values that are mistaken as empty, fill in actual empty value and index out unnecessary ones	Use one-hot encoding to create dummy variables for machine learning analysis	Assign numeric values for ordinal variables for machine learning analysis

Removed variables

- **GarageYrBlt**: Garage year built is removed because in case of there is no garage built in the house, the numeric variables representing the such variable would be confusing for machine learning so we remove it. We still keep other Garage comparison criterias such as quality and replace no garage with ‘0’.
- **YrSold** & **MoSold**: The Year and Month of selling the house would not be applicable in this analysis because the data does not track the changes in house prices over the years (from built to sold), so the time of sell would not be relevant in this analysis.
- **MasVnrType** & **MasVnrArea**: We decided to remove these two variables because after filling values, these two variables have two much missing values, which after indexing out they reduce the number of rows from 1460 rows to only around 500 rows. With more than 280 variables after ordinal and one-hot-encoding, we decided that would not be sufficient values for machine learning and remove them.
- **YearRemodAdd**: After comparison, there is not much different between the Year built variable and Year remodel variable, so we decided to remove it.
- **Id**: Id variable is unnecessary for the analysis and it does not contribute to predictive modelling.

Methods and Results for Prediction

Models



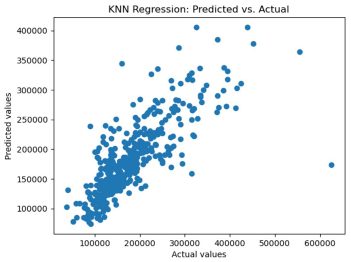
To process the data, models such as support vector machine, k nearest neighbors, decision tree regressor are conducted. However, there results are not ideal as expected, so Linear Regression, Lasso, and XGBoost are chosen to conduct analysis. Pictures from the bottom are the results of three models that do not fit this data. Model score, R squared, and Mean squared error are used to compare predictive capabilities of these models.

```
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 1340235360.0649302
```

```
print(svm.score(X_test, y_test))
```

```
0.7769112998518982
```




```
[18]: r2 = knn_regressor.score(X_test, y_test)
      print("R-squared Score:", r2)
```

```
R-squared Score: 0.628062408954114
```

```
[67]: X_train_dtr = X_train[selected_features_dtr]
      X_test_dtr = X_test[selected_features_dtr]

[69]: df_dtr = DecisionTreeRegressor().fit(X_train_dtr, y_train)

[70]: y_pred_dtr = df_dtr.predict(X_test_dtr)
      mean_squared_error(y_pred_dtr, y_test)

[70]: 2574673332.079977

[71]: print(df_dtr.score(X_test_dtr,y_test))

0.5062284446863035
```

Linear Regression vs Lasso Regression vs Xgboost

Why Regression models？

Results of the house prices problems are continuous or real values.

Why Linear Regression？

Linear regression is chosen as the foundational model due to its simplicity and basic nature, serving as a valuable baseline for subsequent analyses and model comparisons.

Why Lasso Regression？

Lasso,an extension of linear regression, is employed to enhance model complexity while preventing overfitting by introducing regularization. By adding a penalty term that encourages sparsity in the coefficient estimates, Lasso aids in variable selection and promotes a more robust model for intricate relationships, making it a natural progression from the simplicity of linear regression in certain scenarios.

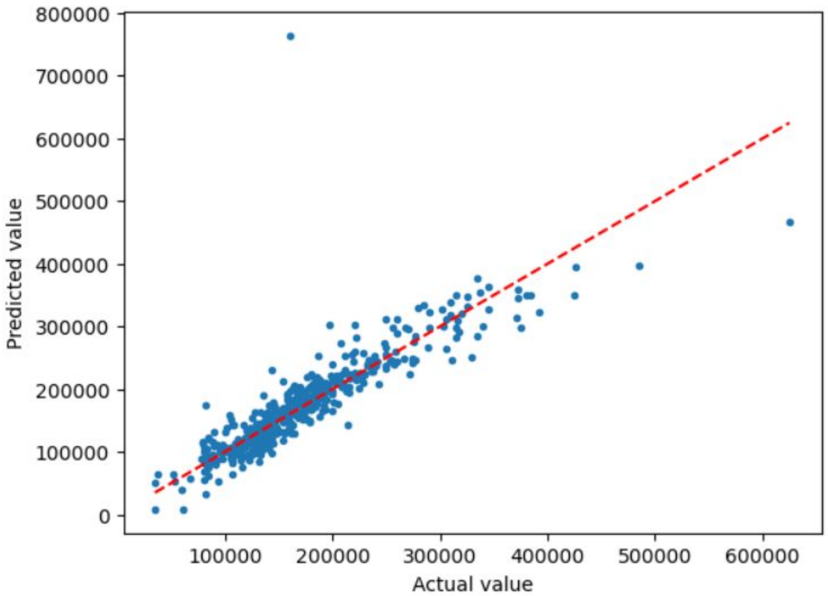
Why XGBoost？

It has a number of hyperparameters that can be tuned to improve model performance, including the learning rate, depth of the trees, and regularization parameters.

Results

Results

Linear Regression with forward selection 70%



Basic linear regression:

```
y_pred = lr.predict(X_test)
print('Mean squared error (MSE) score: ', mean_squared_error(y_pred, y_test))
print('R squared (R2) score: ',r2_score(y_pred,y_test))
print('Model score: ', lr.score(X_test,y_test))
```

```
Mean squared error (MSE) score:  1807614527.4507494
R squared (R2) score:  0.7167736392126516
Model score:  0.6533351918839606
```

Forward selection:

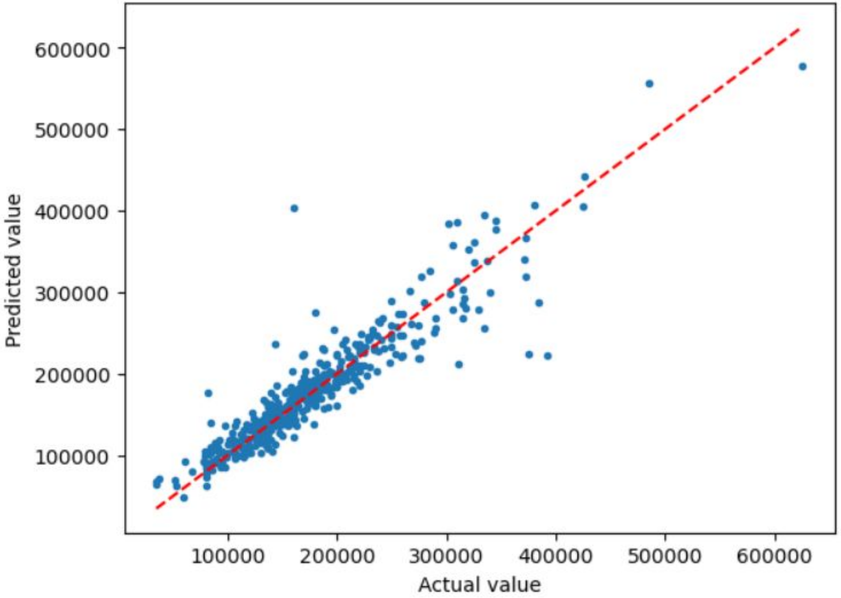
```
y_pred_sfs = lr_sfs.predict(X_test_sfs)
print('Mean squared error (MSE) score: ', mean_squared_error(y_pred_sfs, y_test))
print('R squared (R2) score: ',r2_score(y_pred_sfs,y_test))
print('Model score: ', lr_sfs.score(X_test_sfs, y_test))
```

```
Mean squared error (MSE) score:  1529827848.5579665
R squared (R2) score:  0.737569059559478
Model score:  0.7066091970842658
```

Results

Lasso Regression

65.7%



Basic Lasso:

```
y_pred_original = lasso_original.predict(X_test_standardized)
print('Mean squared error (MSE) score: ', mean_squared_error(y_pred_original, y_test))
print('R squared (R2) score: ',r2_score(y_pred_original,y_test))
print('Model score: ', lasso_original.score(X_test_standardized, y_test))
```

```
Mean squared error (MSE) score:  1785415743.2378323
R squared (R2) score:  0.7192186279654738
Model score:  0.6575924807874931
```

After feature selection:

```
y_pred = lasso.predict(X_test_standardized)
print('Mean squared error (MSE) score: ', mean_squared_error(y_pred, y_test))
print('R squared (R2) score: ',r2_score(y_pred,y_test))
print('Model score: ', lasso_original.score(X_test_standardized, y_test))
```

```
Mean squared error (MSE) score:  1747195969.4903564
R squared (R2) score:  0.7231035187368924
Model score:  0.6575924807874931
```

Results

XGBoost

85.2%

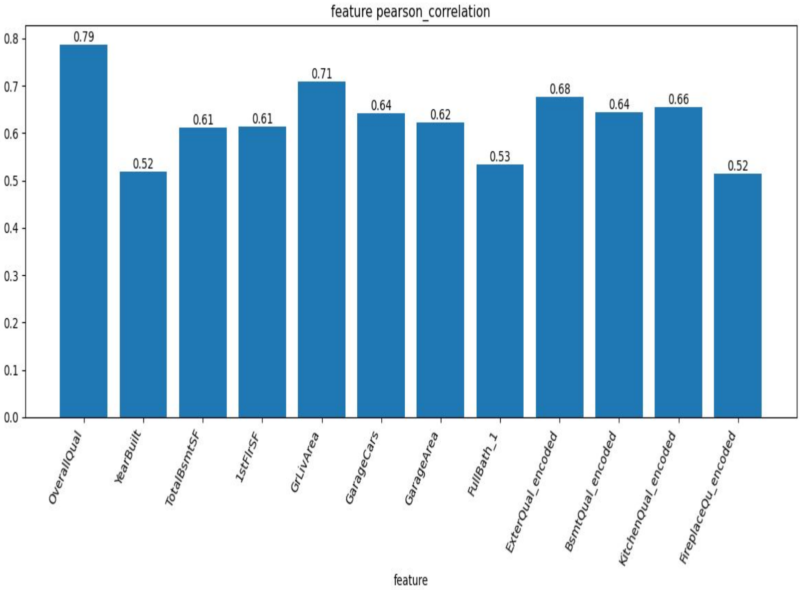
10-Fold Cross Validation

Grid search

```
# Make predictions on the test data
y_pred = model_xgb2.predict(X_test)
# Get R_2 score of XGBoost Regressor
score = model_xgb2.score(X_test,y_test)
print("Model score:",score)
# Calculate Mean Squared Error (MAE) for regression
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error: %.2f" % mse)
#Calculate R2 score
r2 = r2_score(y_pred,y_test)
print("R2 score:", r2)
```

```
Model score: 0.8522767196373697
Mean Squared Error: 770273593.90
R2 score: 0.8458909093473441
```

Conclusion



1、XGboost shows the best performance among three models (R_2 = 0.865,MSE = 702705261).Using 10-Fold Cross Validation and Grid search we found the best hyperparameters to get a good fitting performance.

2、Ensure all variables are correctly coded as continuous, address missing data, outliers, and multicollinearity, consider normalization, and validate assumptions before applying regression models for accurate and robust analysis.

3、It's essential to continuously update and validate the model with new data. Housing markets are dynamic and subject to changing economic, social, and regulatory factors, so regular model maintenance and retraining are crucial to ensure its ongoing accuracy and relevance for decision-making in the real estate sector and related fields.

4、Identifying the most important home prices attributes(like overall quality, built year, area, garage of houses) we could understand what goes into house prices predicting and make an informed decision.

Applications

Applications

Real Estate Investment

Real estate investors can use this predictive model to assess the potential risk and return on their investments. They can estimate future housing prices in a specific location, helping them make informed decisions on property acquisition, development, or resale.

Actual Values

Home buyers and sellers can use the model to estimate the current and future value of their properties. This information is valuable when deciding on a listing price or negotiating a sale.

Mortgage Lending

Mortgage lenders can leverage this predictive model to assess the value of properties and make more accurate lending decisions. It can help determine the loan-to-value ratio and the associated risks.

Urban planning

Urban planners can use the model to anticipate housing demand in specific regions, which can guide decisions on infrastructure development, zoning regulations, and the allocation of resources for public services.

Economic forecasting

Housing prices are closely tied to the economic health of an area. The model can serve as an indicator of economic trends. A rising housing market can signal economic growth, while a declining one might indicate economic challenges.

Housing policy

Housing policy makers can use the model to evaluate the impact of various policy changes on housing affordability. It can guide decisions on where to invest in infrastructure, affordable housing projects, and transportation systems.

Thanks for watching!