HOUSE PRICE Prediction

## **Introduction:**

This dataset contains Real Estate listings in the US broken by State and City.  
Data was collected via web scraping using python libraries. The dataset has one CSV file with 12 columns. The dataset includes information on housing prices, as well as its associated attributes and location.

## **Motivation:**

House price predictions are meant to assist customers who are planning to buy a home by allowing them to know the price range in the future so that they may properly arrange their finances. House price projections are also useful for property investors who want to know the trajectory of housing prices in a specific area. We will be building models to predict house prices in USA using data which consist of metrics such as price, acre\_lot, house\_size and other parameters.

## **Aim:**

* Predicting the price of house based off its location and features.
* Figuring out a trend between the house prices.
* Classifying house prices state wise.

## **Data Cleaning:**

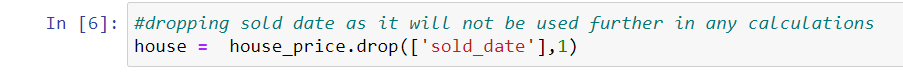
* There are 12 columns and 402159 rows totally.
* Initial percentage of missing values: 12.37%
* Number of missing values per columns

Table

Description automatically generated

It can be noticed that more than half the missing values come from the sold date.

* Dropping sold date as it will not be used in any further calculations.



* After dropping sold date, the percentage of missing values is: 7.504%

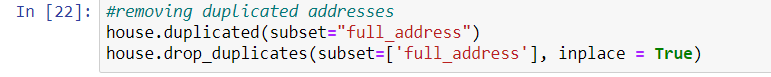
Graphical user interface, text, application

Description automatically generated

* Dropping duplicate columns:

Text

Description automatically generated



After dropping the duplicated full address rows, the shape of the dataset is:

Graphical user interface, text, application, chat or text message

Description automatically generated

* Handling missing values:

To handle the numerical missing values for number of beds, baths, acre\_lot and house\_size, we took the city wise average to fill in these values.

Graphical user interface, application, Word

Description automatically generated

It can be noticed that there are still some missing values in the dataset. So, a state wise average was taken to fill up the rest of the missing values.

Graphical user interface, application, Word

Description automatically generated

The rest of the missing values were dropped using the *dropna* command.

Graphical user interface, text

Description automatically generated

Since number of beds and number of bathrooms cannot be a decimal value, we round up the beds and baths columns. We also limit the dataset to a maximum of 50 bedrooms and 30 bathrooms.

Graphical user interface, text, application

Description automatically generated

The dataset shape comparison before cleaning and after cleaning.

|  |  |  |
| --- | --- | --- |
|  | Before Cleaning | After Cleaning |
| Shape | (402159, 12) | (36387, 11) |

**Data Preparation Activities:**

So, the data obtained contains features of various dimensions and scales altogether. Different scales of the data features affect the modeling of a dataset adversely. It leads to a biased outcome of predictions in terms of misclassification error and accuracy rates. Thus, it is necessary to Scale the data prior to modeling.

Standardization is a scaling technique wherein it makes the data scale-free by converting the statistical distribution of the data into the below format:

* mean – 0 (zero)
* standard deviation – 1

Using standard scaler to scale the numerical variables.

A picture containing timeline

Description automatically generated

According to the above syntax, we initially create an object of the StandardScaler() function. Further, we use fit\_transform() along with the assigned object to transform the data and standardize it.

* Label Encoding refers to converting the labels into a numeric form to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning. Using label encoder to normalize categorical columns:

Text

Description automatically generated

Label encoder is used to transform non-numerical labels to numerical labels. We apply Label Encoding on the house dataset on the target columns which are city and state.

## **Visualization Techniques:**

* Descriptive Summary Statistics of the numerical columns:

Table

Description automatically generated

* Number of houses in each state. Scatter plot between number of beds and price.

**Chart, bar chart

Description automatically generated Chart, scatter chart

Description automatically generated**

Massachusetts has highest number of houses in the dataset.

There is no real pattern that can be noticed from the scatter plot. Most high prices are in between 3 and 10 beds.

Scatter plot between house size and price Relation between the number of bathrooms and price

**A picture containing shape

Description automatically generated Chart, scatter chart

Description automatically generated**

There seems to be one house in Puerto Rico that has a very large house size which has created a bias in the plot.

No real pattern between the price and bath plots. Most of the high prices seem to be in between houses with 5 to 15 bathrooms.

* Count of bathrooms histogram Count of bedrooms histogram:

**Shape

Description automatically generated** A picture containing graphical user interface

Description automatically generated

The maximum number of houses have 2 bathrooms.

The maximum number of bedrooms have 3 bedrooms.

* Box plot of beds and baths: Box plot of house size and acre lot size

Chart

Description automatically generated with medium confidence Chart

Description automatically generated

* Box plot of house size Box plot of size of plot

**Chart

Description automatically generated Chart

Description automatically generated**

* Relationship between size of the plot and size of the house:

**Chart

Description automatically generated Chart

Description automatically generated**

Box plot of price per state:

New Jersey, Long Island and Georgia have the lowest house prices. Puerto Rico and Massachusetts have highest prices.

## **Regression Analysis:**

The test size taken is 0.3 and random\_state = 42.

* **Linear Regression:**

Linear regression is a linear model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x). Here the target variable we have taken in price (y) and we use beds, baths, house\_size, city, state, acre\_lot.

|  |  |  |
| --- | --- | --- |
|  | Mean Square Error | R-Squared |
| Linear Regression | 0.594 | 0.230 |

A picture containing table

Description automatically generated

The value of 𝑏₀ is approximately 0.056. This illustrates that our model predicts the response 0.056 when 𝑥1=x2=x3=x4=x5=x6=0.

Once we have a satisfactory model, then we can use it for predictions with either existing or new data. To obtain the predicted response, we use. predict()

Chart, scatter chart

Description automatically generated

The plot shows a linear regression line that has a low 𝑅². This is likely to be an underfitting model. **Underfitting** occurs when a model can’t accurately capture the dependencies among data, usually because of its own simplicity. It often yields a low 𝑅² with known data and bad generalization capabilities when applied with new data.

The model has a test accuracy of 23%.

* **Polynomial Regression:**

Polynomial regression also a type of linear regression is often used to make predictions using polynomial powers of the independent variables. There’s only one extra step: we need to transform the array of inputs to include nonlinear terms such as 𝑥².

With .fit\_transform(), you’re fitting and transforming the input array in one statement. This method also takes the input array and effectively does the same thing as fit() and transform() called in that order. It also returns the modified array.

We can see although polynomial degree being 2 is not very reasonable. However, we could keep on putting values in and test. But sklearn has a far smarter way of doing this.

|  |  |  |
| --- | --- | --- |
|  | Mean Square Error | R-Squared |
| Polynomial Regression | 9.0619 | -10.7227 |

Grid Search for Hyperparameter Tuning

Here we choose degrees from 3 to 6 for the experiment. This grid search can be used with any other model, and you can have as many parameters as possible. In this scenario, we are trying to find the set of parameters that minimizes the mean squared error.

Polynomial Regression Hyper parameter Tunning:

Different degrees of 3,4,5,6 is run.

|  |  |  |
| --- | --- | --- |
|  | Mean Square Error | R-Squared |
| Polynomial Hyper-Tune | 16120.69 | -20853.11 |

Best degree selected is 3.

The model is severely inaccurate as the R-squared values are negative. The model makes no sense at all given these data. R2 compares the fit of the chosen model with that of a horizontal straight line (the null hypothesis). If the chosen model fits worse than a horizontal line, then R2 is negative. R2 is negative only when the chosen model does not follow the trend of the data, so fits worse than a horizontal line.

**Bottom line:** A negative R2 is not a mathematical impossibility or the sign of a computer bug. It simply means that the chosen model (with its constraints) fits the data poorly.

* **Ridge Regression:**

Ridge regression is a method of estimating the coefficients of multiple-regression models in scenarios where the independent variables are highly correlated. **Ridge Regression** is a popular type of regularized linear regression that includes an L2 penalty. This has the effect of shrinking the coefficients for those input variables that do not contribute much to the prediction task.

For an alpha = 1:

|  |  |  |
| --- | --- | --- |
|  | Mean Square Error | R-Squared |
| Ridge Regression | 0.5946 | 0.2307 |

We do not know that the default hyperparameter of alpha=1.0 is appropriate for our dataset

Instead, it is good practice to test a suite of different configurations and discover what works best for our dataset. One approach would be to grid search alpha values on a log scale and discover what works best for a dataset.

Graphical user interface, text, application, email

Description automatically generated

Using the [GridSearchCV](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html) class with a grid of values we have defined.

Ridge Regression Hyperparameter Tuning:

For alpha = [0.001, 0.01, 0.1, 10,100,1000,10000,1000000]:

|  |  |  |
| --- | --- | --- |
|  | Mean Square Error | R-Squared |
| Ridge Regression | 0.5946 | 0.21505 |

Best alpha selected is 10000.

The ridge model comes when alpha = 1 with a test accuracy of 23.07%

* **Lasso Regression:**

**Lasso Regression** is a popular type of regularized linear regression that includes an L1 penalty. This has the effect of shrinking the coefficients for those input variables that do not contribute much to the prediction task. This penalty allows some coefficient values to go to the value of zero, allowing input variables to be effectively removed from the model, providing a type of automatic feature selection.

For alpha = 1:

|  |  |  |
| --- | --- | --- |
|  | Mean Square Error | R-Squared |
| Lasso Regression | 0.7716 | 0.0017 |

Lasso Regression Hyper parameter tunning:

For alpha = [0.001, 0.01, 0.1, 10,100]:

|  |  |  |
| --- | --- | --- |
|  | Mean Square Error | R-Squared |
| Lasso Hyper-Tune | 0.600879 | 0.2226 |

Best alpha = 0.1. This gives a test accuracy of 22.6%.

The different regression analysis performed can summarized from the table below:

Table

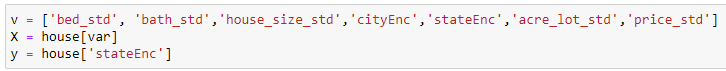
Description automatically generated

The best model to use for predictions is the ridge regression model when alpha = 1. This is because it has the highest R2 value and lowest mean squared error value. The linear regression model also gives similar results.

## **Classification Techniques:**

* **Logistic Regression:**

For Logistic regression in this case, we use multinomial logistic regression. Multinomial logistic regression is used to classify the output labels to more than 2 classes. Here the states are classified. There are 11 classes in total.



Logistic Regression can be configured for multinomial logistic regression by setting the multi\_class argument to “ovr” (One Vs Rest algorithm) and the “solver” argument to a solver that supports multinomial logistic regression such as “liblinear”. The algorithm chooses one class and put all the other classes into a second virtual class and run the binary logistic regression on it. It repeats this procedure for all the classes in the dataset. So, we end up with binary classifiers designed to recognize each class in dataset.

The initial classification and confusion matrix can be found in *APPENDIX-A.*

This model can classify with 61% accuracy.

* *Logistic Regression Hyperparameter tuning:*

For hyperparameter tuning, penalty functions l1 and l2 were applied and C = [0.001,0.01,0.1,1,10]

Text, Word

Description automatically generated

The classification and confusion matrix can be found in *APPENDIX-B.*

* The best estimator across ALL searched params:

LogisticRegression(C=10, multi\_class='ovr', penalty='l1', solver='liblinear')

* The best score across ALL searched params:

0.629

* The best parameters across ALL searched params:

{'C': 10, 'penalty': 'l1'}

* The best test accuracy ALL searched params:

0.634

* **K-Nearest Neighbors:**

KNN assumes that similar things are in proximity of each other. So, if a datapoint is near to another datapoint, it assumes that they belong to similar classes. For the initial model we take a k value of 10. The classification report is shown below:

The classification and confusion matrix can be found in *APPENDIX-C.*

The model has an accuracy of 0.95.

* *K-nearest Neighbors Hyperparameter Tuning:*

For hyperparameter tuning we take a range of k values from 2 to 14. Text

Description automatically generated

The classification and confusion matrix can be found in *APPENDIX-D.*

* Results from Grid Search:
* The best estimator across ALL searched params:

KNeighborsClassifier(n\_neighbors=2)

* The best score across ALL searched params:

0.9816782704287285

* The best parameters across ALL searched params:

{'n\_neighbors': 2}

The best test accuracy ALL searched params: 0.985

* **Decision Tree:**

Decision Tree is a supervised machine learning algorithm where all the decisions were made based on some conditions. The decision tree has a root node and leaf nodes extended from the root node.

Here for the initial model a max\_depth =3, and the gini criterion is selected. Gini is selected as it is a more efficient than entropy.

The classification report and confusion matrix are shown in Appendix – F

The accuracy is 79.4%.

* *Decision Tree Hyperparameter Tuning:*
* The parameters that are hyper tuned are ‘max\_depth’ = [1,2,3,4] and ‘max\_leaf\_nodes’ = [2,3,4,5,6,7,8,9,10].
* The best estimator across ALL searched params:

DecisionTreeClassifier(max\_depth=4, max\_leaf\_nodes=7, random\_state=42)

* The best score across ALL searched params:

0.8713

* The best parameters across ALL searched params:

{'max\_depth': 4, 'max\_leaf\_nodes': 7}

* The best test accuracy ALL searched params:

0.8657

The accuracy is 86.57%.

The classification report and confusion matrix are shown in Appendix – F

* **Classification Model Selection:**

The accuracy of all the classification models is shown in the table below:

Diagram

Description automatically generated with low confidence

Chart, bar chart

Description automatically generated

The best model selected is KNN with a k value of 2. This is selected as it has the highest accuracy value of 98.55%.

* Conclusion:

The regression models had very low accuracy. The reasons this could have happened:

1. Inconsistencies in the data,
2. Probably need to use a deep learning algorithm to solve the inconsistencies.
3. Data could have not been from similar time periods hence the disparity in prices.

Amongst the classification models, KNN seems to classify the best with a k value of 2.

* Suggestions:

1. Data prediction could be better if there was uniformity within the data. (E.g., Price could be updated to the latest date)
2. Information about the house like living room area, backyard space, bedroom size and bathroom size could be added.

APPENDIX – A:

Classification and Confusion Matrix of Logistic Regression:

Table

Description automatically generated

APPENDIX – B:

Classification and Confusion Matrix of Logistic Hyperparameter Tunning Regression:

Table

Description automatically generated

APPENDIX – C:

Classification Report and Confusion Matrix of KNN:

Table

Description automatically generated

APPENDIX-D

Classification Report and Confusion Matrix of KNN grid model:

A picture containing table

Description automatically generated

APPENDIX-E:

Classification Report and Confusion Matrix of Decision Tree:

A picture containing table

Description automatically generated

APPENDIX – F:

Classification Report and Confusion Matrix of Decision Tree Hyperparameter Tuning:

Table

Description automatically generated