HOUSE PRICE PREDICTION

Group Members:

Shruti Kiran, Asansol Engineering College, 10800220016
Priya Jha, Asansol Engineering College, 10800220020
Ankita, Asansol Engineering College, 10800220011
Ishika Prabhakar, Asansol Engineering College, 10800220063

Table of Contents

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my faculty Prof. Arnab Chakraborty for his exemplary guidance, monitoring, and constant encouragement throughout the course of this project. The blessing, help and guidancegiven by him/her time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Shruti Kiran
Priya Jha
Ankita
Ishika Prabhakar

Project Objective

In this project we have a shortened 'House Price Prediction' Dataset from Kaggle. In this dataset, the target attribute is the house rent. So, in this project we need to do classification based on the attributes present in our dataset and predict whether the house rent suites the buyer or not.

In recent times, finding the ideal housing option according to budget and preference is such a hustle. The cost of house rent depends on many factors such as; the house size, number of bedrooms, locality, number of bathrooms, halls and kitchen, furnishing status, and a lot more. With the use of appropriate machine learning algorithms, real estate owners can find the ideal house according to customers budgets and preferences with ease.
This project is an end-to-end implementation of a project where the goal is to predict house rent prices for various different cities. This project uses a machine learning model in the backend to predict the rent prices in various cities using the inputs given to it. This project uses a XGBoost Regression model, Linear Regression Model, Decision Tree Regression model.
The models were evaluated on the basis of two metrics

1. R^2score
2. Mean Absolute Error

The basic procedures implemented to achieve the goals of this research are:

1. Data collection
2. Data cleaning and exploration
3. Feature encoding
4. Train-test split validation
5. Feature scaling
6. Modeling
7. Model evaluation

Project Scope

The broad scope of 'House Price Prediction' project is given below:

• The given dataset has attributes based on which the availability of houses according to the buyer or the renter can be predicted.

• It is a useful project as the Classifier models can be used to quickly determine the availability status of houses in a large datasets.

• Various real estate company can use these models and modify them according to their needs to use in their House Price Predicition application. This will reduce the manual labour and time invested.

• Customers who intend to owe a house can use these trained models to check whether the houses according to their preference is available or not at a determinined budget. The trained models would be required to be implemented in a platform or interface easily accessible as well as with an easy GUI.

• The dataset given to us is a shortened form of the original dataset from Kaggle. So, the results might have some mismatch with the real-world applications. But that can be avoided if the models are trained accordingly.

Data Description

Source of data

Kaggle. The given dataset is a shortened version of the original dataset in Kaggle.

MSSubClass: Identifies the type of dwelling involved in the sale.

20	1-STORY 1946 & NEWER ALL STYLES
30	1-STORY 1945 & OLDER
40	1-STORY W/FINISHED ATTIC ALL AGES
45	1-1/2 STORY - UNFINISHED ALL AGES
50	1-1/2 STORY FINISHED ALL AGES
60	2-STORY 1946 & NEWER
70	2-STORY 1945 & OLDER
75	2-1/2 STORY ALL AGES
80	SPLIT OR MULTI-LEVEL
85	SPLIT FOYER
90	DUPLEX - ALL STYLES AND AGES
120	1-STORY PUD (Planned Unit Development) - 1946 & NEWER
150	1-1/2 STORY PUD - ALL AGES
160	2-STORY PUD - 1946 & NEWER
180	PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
190	2 FAMILY CONVERSION - ALL STYLES AND AGES

MSZoning: Identifies the general zoning classification of the sale.

A	Agriculture
C	Commercial
FV	Floating Village Residential
I	Industrial
RH	Residential High Density
RL	Residential Low Density
RP	Residential Low Density Park
RM	Residential Medium Density

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

Grvl	Gravel
Pave	Paved

Alley: Type of alley access to property

Grvl	Gravel
Pave	Paved
NA	No alley access

LotShape: General shape of property

Reg	Regular
IR1	Slightly irregular
IR2	Moderately Irregular
IR3	Irregular

LandContour: Flatness of the property

Lvl	Near Flat/Level
Bnk	Banked - Quick and significant rise from street grade to building
HLS	Hillside - Significant slope from side to side
Low	Depression

Utilities: Type of utilities available

AllPub	All public Utilities (E,G,W,& S)
NoSewr	Electricity, Gas, and Water (Septic Tank)
NoSeWa	Electricity and Gas Only
ELO	Electricity only

LotConfig: Lot configuration

Inside	Inside lot
Corner	Corner lot
CulDSac	Cul-de-sac
FR2	Frontage on 2 sides of property
FR3	Frontage on 3 sides of property

LandSlope: Slope of property

Gtl	Gentle slope
Mod	Moderate Slope
Sev	Severe Slope

Neighborhood: Physical locations within Ames city limits

Blmngtn	Bloomington Heights
Blueste	Bluestem
BrDale	Briardale
BrkSide	Brookside
ClearCr	Clear Creek
CollgCr	College Creek
Crawfor	Crawford
Edwards	Edwards
Gilbert	Gilbert
IDOTRR	Iowa DOT and Rail Road
MeadowV	Meadow Village
Mitchel	Mitchell
Names	North Ames
NoRidge	Northridge
NPkVill	Northpark Villa
NridgHt	Northridge Heights
NWAmes	Northwest Ames
OldTown	Old Town
SWISU	South & West of Iowa State University
Sawyer	Sawyer
SawyerW	Sawyer West
Somerst	Somerset
StoneBr	Stone Brook
Timber	Timberland
Veenker	Veenker

Condition1: Proximity to various conditions

Artery	Adjacent to arterial street
Feedr	Adjacent to feeder street
Norm	Normal
RRNn	Within 200' of North-South Railroad
RRAn	Adjacent to North-South Railroad
PosN	Near positive off-site feature--park, greenbelt, etc.
PosA	Adjacent to postive off-site feature
RRNe	Within 200' of East-West Railroad
RRAe	Adjacent to East-West Railroad

Condition2: Proximity to various conditions (if more than one is present)

Artery	Adjacent to arterial street
Feedr	Adjacent to feeder street
Norm	Normal
RRNn	Within 200' of North-South Railroad
RRAn	Adjacent to North-South Railroad

PosN Near positive off-site feature--park, greenbelt, etc.
PosA Adjacent to postive off-site feature
RRNe Within 200' of East-West Railroad
RRAe Adjacent to East-West Railroad

BldgType: Type of dwelling

1Fam Single-family Detached
2FmCon Two-family Conversion; originally built as one-family dwelling
Duplx Duplex
TwnhsE Townhouse End Unit
TwnhsI Townhouse Inside Unit

HouseStyle: Style of dwelling

1Story One story
1.5Fin One and one-half story: 2nd level finished
1.5Unf One and one-half story: 2nd level unfinished
2Story Two story
2.5Fin Two and one-half story: 2nd level finished
2.5Unf Two and one-half story: 2nd level unfinished
SFoyer Split Foyer
SLvl Split Level

OverallQual: Rates the overall material and finish of the house

10 Very Excellent
9 Excellent
8 Very Good
7 Good
6 Above Average
5 Average
4 Below Average
3 Fair
2 Poor
1 Very Poor

OverallCond: Rates the overall condition of the house

10 Very Excellent
9 Excellent
8 Very Good
7 Good
6 Above Average
5 Average
4 Below Average
3 Fair
2 Poor

1 Very Poor

YearBuilt: Original construction date

YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

RoofStyle: Type of roof

Flat Flat
Gable Gable
Gambrel Gabrel (Barn)
Hip Hip
Mansard Mansard
Shed Shed

RoofMatl: Roof material

ClyTile Clay or Tile
CompShg Standard (Composite) Shingle
Membran Membrane
Metal Metal
Roll Roll
Tar&Grv Gravel & Tar
WdShake Wood Shakes
WdShngl Wood Shingles

Exterior1st: Exterior covering on house

AsbShng Asbestos Shingles
AsphShn Asphalt Shingles
BrkComm Brick Common
BrkFace Brick Face
CBlock Cinder Block
CemntBd Cement Board
HdBoard Hard Board
ImStucc Imitation Stucco
MetalSd Metal Siding
Other Other
Plywood Plywood
PreCast PreCast
Stone Stone
Stucco Stucco
VinylSd Vinyl Siding
Wd Sdng Wood Siding
WdShing Wood Shingles

Exterior2nd: Exterior covering on house (if more than one material)

AsbShng Asbestos Shingles
AsphShn Asphalt Shingles
BrkComm Brick Common
BrkFace Brick Face
CBlock Cinder Block
CemntBd Cement Board
HdBoard Hard Board
ImStucc Imitation Stucco
MetalSd Metal Siding
Other Other
Plywood Plywood
PreCast PreCast
Stone Stone
Stucco Stucco
VinylSd Vinyl Siding
Wd Sdng Wood Siding
WdShing Wood Shingles

MasVnrType: Masonry veneer type

BrkCmn Brick Common
BrkFace Brick Face
CBlock Cinder Block
None None
Stone Stone

MasVnrArea: Masonry veneer area in square feet

ExterQual: Evaluates the quality of the material on the exterior

Ex Excellent
Gd Good
TA Average/Typical
FaFair
Po Poor

ExterCond: Evaluates the present condition of the material on the exterior

Ex Excellent
Gd Good
TA Average/Typical
FaFair
Po Poor

Foundation: Type of foundation

BrkTil Brick & Tile

CBlock Cinder Block
PConc Poured Contrete
Slab Slab
Stone Stone
Wood Wood

BsmtQual: Evaluates the height of the basement

Ex Excellent (100+ inches)
Gd Good (90-99 inches)
TA Typical (80-89 inches)
FaFair (70-79 inches)
Po Poor (<70 inches
NA No Basement

BsmtCond: Evaluates the general condition of the basement

Ex Excellent
Gd Good
TA Typical - slight dampness allowed
FaFair - dampness or some cracking or settling
Po Poor - Severe cracking, settling, or wetness
NA No Basement

BsmtExposure: Refers to walkout or garden level walls

Gd Good Exposure
Av Average Exposure (split levels or foyers typically score average or above)
Mn Mimimum Exposure
No No Exposure
NA No Basement

BsmtFinType1: Rating of basement finished area

GLQ Good Living Quarters
ALQ Average Living Quarters
BLQ Below Average Living Quarters
Rec Average Rec Room
LwQ Low Quality
Unf Unfinshed
NA No Basement

BsmtFinSF1: Type 1 finished square feet

BsmtFinType2: Rating of basement finished area (if multiple types)

GLQ Good Living Quarters

ALQ Average Living Quarters
BLQ Below Average Living Quarters
Rec Average Rec Room
LwQ Low Quality
Unf Unfinshed
NA No Basement

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

Floor Floor Furnace
GasA Gas forced warm air furnace
GasW Gas hot water or steam heat
Grav Gravity furnace
OthW Hot water or steam heat other than gas
Wall Wall furnace

HeatingQC: Heating quality and condition

Ex Excellent
Gd Good
TA Average/Typical
FaFair
Po Poor

CentralAir: Central air conditioning

N No
Y Yes

Electrical: Electrical system

SBrkr Standard Circuit Breakers & Romex
FuseA Fuse Box over 60 AMP and all Romex wiring (Average)
FuseF 60 AMP Fuse Box and mostly Romex wiring (Fair)
FuseP 60 AMP Fuse Box and mostly knob & tube wiring (poor)
Mix Mixed

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality

Ex	Excellent
Gd	Good
TA	Typical/Average
FaFair	
Po	Poor

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

Functional: Home functionality (Assume typical unless deductions are warranted)

Typ	Typical Functionality
Min1	Minor Deductions 1
Min2	Minor Deductions 2
Mod	Moderate Deductions
Maj1	Major Deductions 1
Maj2	Major Deductions 2
Sev	Severely Damaged
Sal	Salvage only

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

Ex	Excellent - Exceptional Masonry Fireplace
Gd	Good - Masonry Fireplace in main level
TA	Average - Prefabricated Fireplace in main living area or Masonry Fireplace in basement
FaFair	Prefabricated Fireplace in basement
Po	Poor - Ben Franklin Stove
NA	No Fireplace

GarageType: Garage location

2Types More than one type of garage
Attchd Attached to home
Basment Basement Garage
BuiltIn Built-In (Garage part of house - typically has room above garage)
CarPort Car Port
Detchd Detached from home
NA No Garage

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

Fin Finished
RFn Rough Finished
Unf Unfinished
NA No Garage

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

Ex Excellent
Gd Good
TA Typical/Average
FaFair
Po Poor
NA No Garage

GarageCond: Garage condition

Ex Excellent
Gd Good
TA Typical/Average
FaFair
Po Poor
NA No Garage

PavedDrive: Paved driveway

Y Paved
P Partial Pavement
N Dirt/Gravel

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

Ex	Excellent
Gd	Good
TA	Average/Typical
FaFair	
NA	No Pool

Fence: Fence quality

GdPrv	Good Privacy
MnPrv	Minimum Privacy
GdWo	Good Wood
MnWw	Minimum Wood/Wire
NA	No Fence

MiscFeature: Miscellaneous feature not covered in other categories

Elev	Elevator
Gar2	2nd Garage (if not described in garage section)
Othr	Other
Shed	Shed (over 100 SF)
TenC	Tennis Court
NA	None

MiscVal: $Value of miscellaneous feature

MoSold: Month Sold (MM)

YrSold: Year Sold (YYYY)

SaleType: Type of sale

WD	Warranty Deed - Conventional
CWD	Warranty Deed - Cash

VWD Warranty Deed - VA Loan
New Home just constructed and sold
COD Court Officer Deed/Estate
Con Contract 15% Down payment regular terms
ConLw Contract Low Down payment and low interest
ConLI Contract Low Interest
ConLD Contract Low Down
Oth Other

SaleCondition: Condition of sale

Normal Normal Sale
Abnorml Abnormal Sale - trade, foreclosure, short sale
AdjLandAdjoining Land Purchase
Alloca Allocation - two linked properties with separate deeds, typically condo with a garage unit
Family Sale between family members
Partial Home was not completed when last assessed (associated with New Homes)

Model Building

The models tried out in this project are

1. Linear Regression
2. Decision Tree Regression
3. Random Forest Regression
4. Bagging and Boosting
5. SVM model Building
6. XGBoost Regression

Linear Regression :-

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:

Mathematically, we can represent a linear regression as:

$y= a0+a1x+ \varepsilon$

Here,

Y= Dependent Variable (Target Variable)

X= Independent Variable (predictor Variable)

a0= intercept of the line (Gives an additional degree of freedom)

a1 = Linear regression coefficient (scale factor to each input value).

ε = random error

The values for x and y variables are training datasets for Linear Regression model representation.

Types of Linear Regression

Linear regression can be further divided into two types of the algorithm:

Simple Linear Regression:

If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.

Multiple Linear regression:

If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

Linear Regression Line

A linear line showing the relationship between the dependent and independent variables is called a regression line. A regression line can show two types of relationship:

Positive Linear Relationship:

If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.

Negative Linear Relationship:

If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.

Random Forest Regression :-

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.

The trees run in parallel with no interaction amongst them. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees. To get a better understanding of the Random Forest algorithm, let's walk through the steps:

Pick at random k data points from the training set.

Build a decision tree associated to these k data points.

Choose the number N of trees you want to build and repeat steps 1 and 2.

For a new data point, make each one of your N-tree trees predict the value of y for the data point in question and assign the new data point to the average across all of the predicted y values.

A Random Forest Regression model is powerful and accurate. It usually performs great on many problems, including features with non-linear relationships. Disadvantages, however, include the following: there is no interpretability, overfitting may easily occur, we must choose the number of trees to include in the model.

Let's see Random Forest Regression in action!

Now that we have a basic understanding of how the Random Forest Regression model works, we can assess its performance on a real-world dataset. Similar to my previous posts, I will be using data on House Sales in King County, USA.

After importing the libraries, importing the dataset, addressing null values, and dropping any necessary columns, we are ready to create our Random Forest Regression model!

Step 1: Identify your dependent (y) and independent variables (X)

Step 2: Split the dataset into the Training set and Test set

Step 3: Training the Random Forest Regression model on the whole dataset

Step 4: Predicting the Test set results

XGBoost :-

XGBoost is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction. XGBoost stands for "Extreme Gradient Boosting" and it has become one of the most popular and widely used machine learning algorithms due to its ability to handle large datasets and

its ability to achieve state-of-the-art performance in many machine learning tasks such as classification and regression.

One of the key features of XGBoost is its efficient handling of missing values, which allows it to handle real-world data with missing values without requiring significant pre-processing. Additionally, XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time.

XGBoost can be used in a variety of applications, including Kaggle competitions, recommendation systems, and click-through rate prediction, among others. It is also highly customizable and allows for fine-tuning of various model parameters to optimize performance.

XgBoost stands for Extreme Gradient Boosting, which was proposed by the researchers at the University of Washington. It is a library written in C++ which optimizes the training for Gradient Boosting.

Mathematics behind XgBoost

Before beginning with mathematics about Gradient Boosting, Here's a simple example of a CART that classifies whether someone will like a hypothetical computer game X. The example of tree is below:

The prediction scores of each individual decision tree then sum up to get If you look at the example, an important fact is that the two trees try to complement each other. Mathematically, we can write our model in the form

$$\hat{y}{i} = \sum{k=1}^{K} f_k(x_i), f_k \epsilon \mathcal{F} \\$$

where,

K is the number of trees

f is the functional space of F

F is the set of possible CARTs.

The objective function for the above model is given by:

$$obj(\theta) = \sum_{i}^{n} l(y_{i}, \hat{y}{i}) + \sum{k=1}^K \Omega(f_{k}) \\$$

where,

first term is the loss function and the second is the regularization parameter.

Now, Instead of learning the tree all at once which makes the optimization harder, we apply the additive stretegy, minimize the loss what we have learned and add a new tree which can be summarised below:

$$\hat{y}i^{(0)} =0\\ \hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)\\ \hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i)= \hat{y}_i^{(1)} + f_2(x_i)\\ ....\\ \hat{y}_i^{(t)} = \sum{k=1}^t f_k(x_i)= \hat{y}_i^{(t-1)} + f_t(x_i)$$

SVM Model Building :-

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane.

Types of SVM

SVM can be of two types:

Linear SVM:

Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

Non-linear SVM:

Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane:

There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

How does SVM works?

Linear SVM:

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue.

So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes.

Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a hyperplane. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as margin. And the goal of SVM is to maximize this margin. The hyperplane with maximum margin is called the optimal hyperplane.

Non-Linear SVM:

If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line.

So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$z=x2 +y2$

By adding the third dimension, the sample space will become as below image:

So now, SVM will divide the datasets into classes in the following way.

Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1.

Hence we get a circumference of radius 1 in case of non-linear data.

Decision Tree Regression :-

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.

Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.

The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies

Root Node:

Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

Leaf Node:

Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

Splitting:

Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

Branch/Sub Tree:

A tree formed by splitting the tree.

Pruning:

Pruning is the process of removing the unwanted branches from the tree.

Parent/Child node:

The root node of the tree is called the parent node, and other nodes are called the child nodes.

How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer).

Advantages

It is simple to understand as it follows the same process which a human follow while making any decision in real-life.

It can be very useful for solving decision-related problems.

It helps to think about all the possible outcomes for a problem.

There is less requirement of data cleaning compared to other algorithms.

Disadvantages

The decision tree contains lots of layers, which makes it complex.

It may have an overfitting issue, which can be resolved using the Random Forest algorithm.

For more class labels, the computational complexity of the decision tree may increase.

Code

```
importnumpyasn
                p
                    import pandas as pd
                    import matplotlib.pyplot as plt
                    import seaborn as sns


                    # In[157]:


                    train = pd.read_csv('train.csv')
                    test = pd.read_csv('test.csv')

                    print("Shape of train: ", train.shape)
                    print("Shape of test: ", test.shape)


                    # In[158]:


                    train.head(10)


                    # In[159]:


                    test.head(10)


                    # In[160]:


                    df = pd.concat((train, test))
                    temp_df = df
                    print("Shape of df: ", df.shape)
```

In[161]:

df.head(6)

In[162]:

df.tail(6)

In[163]:

pd.set_option("display.max_columns", 2000)
pd.set_option("display.max_rows", 85)

In[164]:

df.head(6)

In[165]:

df.tail(6)

In[166]:

df.info()

In[167]:

```
df.describe()

# In[168]:

df.select_dtypes(include=['int64', 'float64']).columns

# In[169]:

df.select_dtypes(include=['object']).columns

# In[170]:

df = df.set_index("Id")

# In[171]:

plt.figure(figsize=(16,9))
sns.heatmap(df.isnull())

# In[172]:

null_percent = df.isnull().sum()/df.shape[0]*100
null_percent

# In[173]:

train["SalePrice"].describe()
```

```
# In[174]:


plt.figure(figsize=(10,8))
bar = sns.distplot(train["SalePrice"])
bar.legend(["Skewness:
{:.2f}".format(train['SalePrice'].skew())])


# In[175]:


plt.figure(figsize=(25,25))
ax = sns.heatmap(train.corr(), cmap = "coolwarm",
annot=True, linewidth=2)

bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)


# In[176]:


hig_corr = train.corr()
hig_corr_features =
hig_corr.index[abs(hig_corr["SalePrice"]) >= 0.5]
hig_corr_features


# In[177]:


plt.figure(figsize=(10,8))
ax = sns.heatmap(train[hig_corr_features].corr(), cmap =
"coolwarm", annot=True, linewidth=3)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)


# In[178]:
```

```
plt.figure(figsize=(16,9))
for i in range(len(hig_corr_features)):
    if i <= 9:
        plt.subplot(3,4,i+1)
        plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
        sns.regplot(data=train, x = hig_corr_features[i], y =
'SalePrice')


# In[179]:


missing_col = df.columns[df.isnull().any()]
missing_col


# In[180]:


bsmt_col = ['BsmtCond', 'BsmtExposure', 'BsmtFinSF1',
'BsmtFinSF2', 'BsmtFinType1',
       'BsmtFinType2', 'BsmtFullBath', 'BsmtHalfBath',
'BsmtQual', 'BsmtUnfSF', 'TotalBsmtSF']
bsmt_feat = df[bsmt_col]
bsmt_feat


# In[181]:


bsmt_feat.info()


# In[182]:


bsmt_feat.isnull().sum()


# In[183]:
```

```
bsmt_feat = bsmt_feat[bsmt_feat.isnull().any(axis=1)]
bsmt_feat
```

```
# In[184]:
```

```
bsmt_feat_all_nan = bsmt_feat[(bsmt_feat.isnull() |
bsmt_feat.isin([0])).all(1)]
bsmt_feat_all_nan
```

```
# In[185]:
```

```
bsmt_feat_all_nan.shape
```

```
# In[186]:
```

```
qual = list(df.loc[:, df.dtypes == 'object'].columns.values)
qual
```

```
# In[187]:
```

```
bsmt_feat =
bsmt_feat[bsmt_feat.isin([np.nan]).any(axis=1)]
bsmt_feat
```

```
# In[188]:
```

```
bsmt_feat.shape
```

In[189]:

```
print(df['BsmtFinSF2'].max())
print(df['BsmtFinSF2'].min())
```

In[190]:

```
pd.cut(range(0,1526),5) # create a bucket
```

In[191]:

```
df_slice = df[(df['BsmtFinSF2'] >= 305) &
(df['BsmtFinSF2'] <= 610)]
df_slice
```

In[192]:

```
bsmt_feat.at[333,'BsmtFinType2'] =
df_slice['BsmtFinType2'].mode()[0]
```

In[193]:

```
bsmt_feat
```

In[194]:

```
df.update(bsmt_feat)
```

In[195]:

```
bsmt_feat.isnull().sum()
```

In[196]:

```
df.columns[df.isnull().any()]
```

In[197]:

```
garage_col = ['GarageArea', 'GarageCars', 'GarageCond',
'GarageFinish', 'GarageQual', 'GarageType', 'GarageYrBlt',]
garage_feat = df[garage_col]
garage_feat = garage_feat[garage_feat.isnull().any(axis=1)]
garage_feat
```

In[198]:

```
garage_feat.shape
```

In[199]:

```
garage_feat_all_nan = garage_feat[(garage_feat.isnull() |
garage_feat.isin([0])).all(1)]
garage_feat_all_nan.shape
```

In[200]:

```
garage_feat_all_nan = garage_feat[(garage_feat.isnull() |
garage_feat.isin([0])).all(1)]
garage_feat_all_nan.shape


# In[201]:


garage_feat = garage_feat[garage_feat.isnull().any(axis=1)]
garage_feat


# In[202]:


garage_feat.isnull().any()


# In[203]:


df.update(garage_feat)


# In[204]:


df.columns[df.isnull().any()]


# In[205]:


df['Electrical'] =
df['Electrical'].fillna(df['Electrical'].mode()[0])
df['Exterior1st'] =
df['Exterior1st'].fillna(df['Exterior1st'].mode()[0])
df['Exterior2nd'] =
df['Exterior2nd'].fillna(df['Exterior2nd'].mode()[0])
df['Functional'] =
df['Functional'].fillna(df['Functional'].mode()[0])
```

```
df['KitchenQual'] =
df['KitchenQual'].fillna(df['KitchenQual'].mode()[0])
df['MSZoning'] =
df['MSZoning'].fillna(df['MSZoning'].mode()[0])
df['SaleType'] =
df['SaleType'].fillna(df['SaleType'].mode()[0])
df['Utilities'] = df['Utilities'].fillna(df['Utilities'].mode()[0])
df['MasVnrType'] =
df['MasVnrType'].fillna(df['MasVnrType'].mode()[0])


# In[206]:


df.columns[df.isnull().any()]


# In[207]:


df[df['MasVnrArea'].isnull() ==
True]['MasVnrType'].unique()


# In[208]:


df.loc[(df['MasVnrType'] == 'None') &
(df['MasVnrArea'].isnull() == True), 'MasVnrArea'] = 0


# In[209]:


df.isnull().sum()/df.shape[0] * 100


# In[210]:


lotconfig = ['Corner', 'Inside', 'CulDSac', 'FR2', 'FR3']
```

```
for i in lotconfig:
    df['LotFrontage'] =
pd.np.where((df['LotFrontage'].isnull() == True) &
(df['LotConfig'] == i) , df[df['LotConfig'] == i]
['LotFrontage'].mean(), df['LotFrontage'])


# In[211]:


df.isnull().sum()


# In[212]:


df.columns


# In[213]:


feat_dtype_convert = ['MSSubClass', 'YearBuilt',
'YearRemodAdd', 'GarageYrBlt', 'YrSold']
for i in feat_dtype_convert:
    df[i] = df[i].astype(str)


# In[214]:


df['MoSold'].unique()


# In[215]:


import calendar
df['MoSold'] = df['MoSold'].apply(lambda x :
calendar.month_abbr[x])
```

In[216]:

```
df['MoSold'].unique()
```

In[217]:

```
quan = list(df.loc[:, df.dtypes != 'object'].columns.values)
```

In[218]:

```
quan
```

In[219]:

```
len(quan)
```

In[220]:

```
obj_feat = list(df.loc[:, df.dtypes ==
'object'].columns.values)
obj_feat
```

In[221]:

```
from pandas.api.types import CategoricalDtype
df['BsmtCond'] =
df['BsmtCond'].astype(CategoricalDtype(categories=['NA',
'Po', 'Fa', 'TA', 'Gd', 'Ex'], ordered = True)).cat.codes
```

In[222]:

```
df['BsmtCond'].unique()
```

In[223]:

```
df['BsmtExposure'] =
df['BsmtExposure'].astype(CategoricalDtype(categories=['
NA', 'Mn', 'Av', 'Gd'], ordered = True)).cat.codes
```

In[224]:

```
df['BsmtExposure'].unique()
```

In[225]:

```
df['BsmtFinType1'] =
df['BsmtFinType1'].astype(CategoricalDtype(categories=['
NA', 'Unf', 'LwQ', 'Rec', 'BLQ','ALQ', 'GLQ'], ordered =
True)).cat.codes
df['BsmtFinType2'] =
df['BsmtFinType2'].astype(CategoricalDtype(categories=['
NA', 'Unf', 'LwQ', 'Rec', 'BLQ','ALQ', 'GLQ'], ordered =
True)).cat.codes
df['BsmtQual'] =
df['BsmtQual'].astype(CategoricalDtype(categories=['NA',
'Po', 'Fa', 'TA', 'Gd', 'Ex'], ordered = True)).cat.codes
df['ExterQual'] =
df['ExterQual'].astype(CategoricalDtype(categories=['Po',
'Fa', 'TA', 'Gd', 'Ex'], ordered = True)).cat.codes
```

```
df['ExterCond'] =
df['ExterCond'].astype(CategoricalDtype(categories=['Po',
'Fa', 'TA', 'Gd', 'Ex'], ordered = True)).cat.codes
df['Functional'] =
df['Functional'].astype(CategoricalDtype(categories=['Sal',
'Sev', 'Maj2', 'Maj1', 'Mod','Min2','Min1', 'Typ'], ordered =
True)).cat.codes
df['GarageCond'] =
df['GarageCond'].astype(CategoricalDtype(categories=['N
A', 'Po', 'Fa', 'TA', 'Gd', 'Ex'], ordered = True)).cat.codes
df['GarageQual'] =
df['GarageQual'].astype(CategoricalDtype(categories=['NA
', 'Po', 'Fa', 'TA', 'Gd', 'Ex'], ordered = True)).cat.codes
df['GarageFinish'] =
df['GarageFinish'].astype(CategoricalDtype(categories=['N
A', 'Unf', 'RFn', 'Fin'], ordered = True)).cat.codes
df['HeatingQC'] =
df['HeatingQC'].astype(CategoricalDtype(categories=['Po',
'Fa', 'TA', 'Gd', 'Ex'], ordered = True)).cat.codes
df['KitchenQual'] =
df['KitchenQual'].astype(CategoricalDtype(categories=['P
o', 'Fa', 'TA', 'Gd', 'Ex'], ordered = True)).cat.codes
df['PavedDrive'] =
df['PavedDrive'].astype(CategoricalDtype(categories=['N',
'P', 'Y'], ordered = True)).cat.codes
df['Utilities'] =
df['Utilities'].astype(CategoricalDtype(categories=['ELO',
'NASeWa', 'NASeWr', 'AllPub'], ordered = True)).cat.codes


# In[226]:


df['Utilities'].unique()


# In[227]:


skewed_features = ['1stFlrSF',
 '2ndFlrSF',
```

```
'3SsnPorch',
'BedroomAbvGr',
'BsmtFinSF1',
'BsmtFinSF2',
'BsmtFullBath',
'BsmtHalfBath',
'BsmtUnfSF',
'EnclosedPorch',
'Fireplaces',
'FullBath',
'GarageArea',
'GarageCars',
'GrLivArea',
'HalfBath',
'KitchenAbvGr',
'LotArea',
'LotFrontage',
'LowQualFinSF',
'MasVnrArea',
'MiscVal',
'OpenPorchSF',
'PoolArea',
'ScreenPorch',
'TotRmsAbvGrd',
'TotalBsmtSF',
'WoodDeckSF']


# In[228]:


quan == skewed_features


# In[229]:


plt.figure(figsize=(25,20))
for i in range(len(skewed_features)):
    if i <= 28:
        plt.subplot(7,4,i+1)
```

```
        plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
        ax = sns.distplot(df[skewed_features[i]])
        ax.legend(["Skewness:
{:.2f}".format(df[skewed_features[i]].skew())], fontsize =
'xx-large')


# In[230]:


df_back = df


# In[231]:


for i in skewed_features:
    df[i] = np.log(df[i] + 1)



# In[232]:


plt.figure(figsize=(25,20))
for i in range(len(skewed_features)):
    if i <= 28:
        plt.subplot(7,4,i+1)
        plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
        ax = sns.distplot(df[skewed_features[i]])
        ax.legend(["Skewness:
{:.2f}".format(df[skewed_features[i]].skew())], fontsize =
'xx-large')


# In[233]:


SalePrice = np.log(train['SalePrice'] + 1)
```

In[234]:

```
obj_feat = list(df.loc[:,df.dtypes ==
'object'].columns.values)
len(obj_feat)
```

In[235]:

```
dummy_drop = []
clean_df = df
for i in obj_feat:
   dummy_drop += [i + '_' + str(df[i].unique()[-1])]

df = pd.get_dummies(df, columns = obj_feat)
df = df.drop(dummy_drop, axis = 1)
```

In[236]:

```
df.shape
```

In[237]:

```
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
scaler.fit(df)
df = scaler.transform(df)
```

```
# # model building
#
```

In[238]:

```
train_len = len(train)


# In[239]:


X_train = df[:train_len]
X_test = df[train_len:]
y_train = SalePrice

print(X_train.shape)
print(X_test.shape)
print(len(y_train))


# In[240]:


from sklearn.model_selection import KFold,
cross_val_score
from sklearn.metrics import make_scorer, r2_score

def test_model(model, X_train=X_train, y_train=y_train):
    cv = KFold(n_splits = 3, shuffle=True, random_state =
45)
    r2 = make_scorer(r2_score)
    r2_val_score = cross_val_score(model, X_train, y_train,
cv=cv, scoring = r2)
    score = [r2_val_score.mean()]
    return score


#
#
# # Linear regression

# In[241]:


import sklearn.linear_model as linear_model
LR = linear_model.LinearRegression()
```

```
test_model(LR)


# In[242]:


cross_validation = cross_val_score(estimator = LR, X =
X_train, y = y_train, cv = 10)
print("Cross validation accuracy of LR model = ",
cross_validation)
print("\nCross validation mean accuracy of LR model = ",
cross_validation.mean())


# In[243]:


rdg = linear_model.Ridge()
test_model(rdg)


# In[244]:


lasso = linear_model.Lasso(alpha=1e-4)
test_model(lasso)


# In[245]:


from sklearn.svm import SVR
svr_reg = SVR(kernel='rbf')
test_model(svr_reg)


# # DECISION TREE

# In[246]:
```

```
from sklearn.tree import DecisionTreeRegressor
dt_reg = DecisionTreeRegressor(random_state=21)
test_model(dt_reg)
```

```
# In[257]:
```

```
from sklearn.ensemble import RandomForestRegressor
rf_reg = RandomForestRegressor(n_estimators = 1000,
random_state=51)
test_model(rf_reg)
```

```
# # Bagging & Boosting¶
#
```

```
# In[259]:
```

```
from sklearn.ensemble import BaggingRegressor,
GradientBoostingRegressor
br_reg = BaggingRegressor(n_estimators=1000,
random_state=51)
gbr_reg = GradientBoostingRegressor(n_estimators=1000,
learning_rate=0.1, loss='ls', random_state=51)
```

```
# In[261]:
```

```
test_model(br_reg)
```

```
# In[262]:
```

```
get_ipython().system('pip install xgboost')
```

In[263]:

```
import xgboost
xgb_reg = xgboost.XGBRegressor(bbooster='gbtree',
random_state=51)
test_model(xgb_reg)
```

In[264]:

```
svr_reg.fit(X_train,y_train)
y_pred = np.exp(svr_reg.predict(X_test)).round(2)
```

In[269]:

```
y_pred
```

In[270]:

```
submit_test1 =
pd.concat([test['Id'],pd.DataFrame(y_pred)], axis=1)
submit_test1.columns=['Id', 'SalePrice']
```

In[271]:

```
submit_test1
```

In[272]:

```
submit_test1.to_csv('sample_submission.csv', index=False
)


# In[274]:


from sklearn.model_selection import
RandomizedSearchCV, GridSearchCV
params = {'kernel': ['rbf'],
        'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
        'C': [0.1, 1, 10, 100, 1000],
        'epsilon': [1, 0.2, 0.1, 0.01, 0.001, 0.0001]}
rand_search = RandomizedSearchCV(svr_reg,
param_distributions=params, n_jobs=-1, cv=11)
rand_search.fit(X_train, y_train)
rand_search.cv


# In[ ]:


svr_reg= SVR(C=100, cache_size=200, coef0=0.0,
degree=3, epsilon=0.01, gamma=0.0001,
   kernel='rbf', max_iter=-1, shrinking=True, tol=0.001,
verbose=False)
test_model(svr_reg)


# In[275]:


svr_reg.fit(X_train,y_train)
y_pred = np.exp(svr_reg.predict(X_test)).round(2)


# In[ ]:


y_pred
```

In[ ]:

```
submit_test3 =
pd.concat([test['Id'],pd.DataFrame(y_pred)], axis=1)
submit_test3.columns=['Id', 'SalePrice']
```

In[277]:

```
submit_test3.to_csv('sample_submission.csv', index=False)
submit_test3
```

XGBoost Parameter Tuning¶
#

In[278]:

```
xgb2_reg=xgboost.XGBRegressor(n_estimators= 899,
 mon_child_weight= 2,
 max_depth= 4,
 learning_rate= 0.05,
 booster= 'gbtree')

test_model(xgb2_reg)
```

In[279]:

```
xgb2_reg.fit(X_train,y_train)
y_pred_xgb_rs=xgb2_reg.predict(X_test)
```

In[280]:

```
np.exp(y_pred_xgb_rs).round(2)


# In[282]:


y_pred_xgb_rs =
np.exp(xgb2_reg.predict(X_test)).round(2)
xgb_rs_solution = pd.concat([test['Id'],
pd.DataFrame(y_pred_xgb_rs)], axis=1)
xgb_rs_solution.columns=['Id', 'SalePrice']
xgb_rs_solution.to_csv('sample_submission.csv',
index=False)


# In[283]:


xgb_rs_solution


# # Selection To Improve Accuracy

# In[284]:


plt.figure(figsize=(9,16))
corr_feat_series =
pd.Series.sort_values(train.corrwith(train.SalePrice))
sns.barplot(x=corr_feat_series, y=corr_feat_series.index,
orient='h')


# In[285]:


df_back1 = df_back


# In[286]:
```

```
df_back1.to_csv('df_for_feature_engineering.csv',
index=False)


# In[287]:


list(corr_feat_series.index)


# # Feature Selection

# In[288]:


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# In[289]:


df = pd.read_csv('df_for_feature_engineering.csv')
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
df


# # Drop Feature
#

# In[290]:


df = df.drop(['YrSold',
 'LowQualFinSF',
 'MiscVal',
 'BsmtHalfBath',
```

```
 'BsmtFinSF2',
 '3SsnPorch',
 'MoSold'],axis=1)


# In[291]:


quan = list(df.loc[:,df.dtypes != 'object'].columns.values)
quan


# In[292]:


skewd_feat = ['1stFlrSF',
 '2ndFlrSF',
 'BedroomAbvGr',
 'BsmtFinSF1',
 'BsmtFullBath',
 'BsmtUnfSF',
 'EnclosedPorch',
 'Fireplaces',
 'FullBath',
 'GarageArea',
 'GarageCars',
 'GrLivArea',
 'HalfBath',
 'KitchenAbvGr',
 'LotArea',
 'LotFrontage',
 'MasVnrArea',
 'OpenPorchSF',
 'PoolArea',
 'ScreenPorch',
 'TotRmsAbvGrd',
 'TotalBsmtSF',
 'WoodDeckSF']


# In[293]:
```

```
for i in skewd_feat:
    df[i] = np.log(df[i] + 1)

SalePrice = np.log(train['SalePrice'] + 1)


# In[294]:


df


# In[295]:


obj_feat = list(df.loc[:, df.dtypes ==
'object'].columns.values)
print(len(obj_feat))

obj_feat


# In[296]:


dummy_drop = []
for i in obj_feat:
    dummy_drop += [i + '_' + str(df[i].unique()[-1])]

df = pd.get_dummies(df, columns = obj_feat)
df = df.drop(dummy_drop, axis = 1)


# In[297]:


df.shape
```

In[298]:

```
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
scaler.fit(df)
df = scaler.transform(df)
```

Model Bulding
#

In[299]:

```
train_len = len(train)
X_train = df[:train_len]
X_test = df[train_len:]
y_train = SalePrice

print("Shape of X_train: ", len(X_train))
print("Shape of X_test: ", len(X_test))
print("Shape of y_train: ", len(y_train))
```

In[300]:

```
from sklearn.model_selection import KFold,
cross_val_score
from sklearn.metrics import make_scorer, r2_score

def test_model(model, X_train=X_train, y_train=y_train):
    cv = KFold(n_splits = 3, shuffle=True, random_state =
45)
    r2 = make_scorer(r2_score)
    r2_val_score = cross_val_score(model, X_train, y_train,
cv=cv, scoring = r2)
    score = [r2_val_score.mean()]
    return score
```

linear model

In[301]:

```
import sklearn.linear_model as linear_model
LR = linear_model.LinearRegression()
test_model(LR)
```

In[302]:

```
rdg = linear_model.Ridge()
test_model(rdg)
```

In[303]:

```
lasso = linear_model.Lasso(alpha=1e-4)
test_model(lasso)
```

In[304]:

```
from sklearn.svm import SVR
svr = SVR(kernel='rbf')
test_model(svr)
```

Svm Hyper Parameter Tuning
#
#

In[305]:

```
from sklearn.model_selection import
RandomizedSearchCV, GridSearchCV
params = {'kernel': ['rbf'],
        'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
        'C': [0.1, 1, 10, 100, 1000],
        'epsilon': [1, 0.2, 0.1, 0.01, 0.001, 0.0001]}
rand_search = RandomizedSearchCV(svr_reg,
param_distributions=params, n_jobs=-1, cv=11)
rand_search.fit(X_train, y_train)
rand_search.best_score_


# In[306]:


rand_search.best_estimator_


# In[308]:


svr_reg1=SVR(C=10, cache_size=200, coef0=0.0, degree=3,
epsilon=0.1, gamma=0.001,
   kernel='rbf', max_iter=-1, shrinking=True, tol=0.001,
verbose=False)
test_model(svr_reg1)


# In[309]:


svr_reg= SVR(C=100, cache_size=200, coef0=0.0,
degree=3, epsilon=0.01, gamma=0.0001,
   kernel='rbf', max_iter=-1, shrinking=True, tol=0.001,
verbose=False)
test_model(svr_reg)


# # XGBoost
#
#
```

In[310]:

```
import xgboost
xgb_reg = xgboost.XGBRegressor(bbooster='gbtree',
random_state=51)
test_model(xgb_reg)
```

In[311]:

```
xgb2_reg=xgboost.XGBRegressor(n_estimators= 899,
 mon_child_weight= 2,
 max_depth= 4,
 learning_rate= 0.05,
 booster= 'gbtree')

test_model(xgb2_reg)
```

In[329]:

```
xgb2_reg.fit(X_train,y_train)
y_pred = np.exp(xgb2_reg.predict(X_test)).round(2)
submit_test = pd.concat([test['Id'],pd.DataFrame(y_pred)],
axis=1)
submit_test.columns=['Id', 'SalePrice']
submit_test.to_csv('sample_submission.csv', index=False)
submit_test
```

In[327]:

```
svr_reg.fit(X_train,y_train)
y_pred = np.exp(svr_reg.predict(X_test)).round(2)
submit_test = pd.concat([test['Id'],pd.DataFrame(y_pred)],
axis=1)
```

```
submit_test.columns=['Id', 'SalePrice']
submit_test.to_csv('sample_submission.csv', index=False)
submit_test


# # Model Save
#

# In[324]:


import pickle

pickle.dump(svr_reg,
open('model_house_price_prediction.csv', 'wb'))
model_house_price_prediction =
pickle.load(open('model_house_price_prediction.csv',
'rb'))
model_house_price_prediction.predict(X_test)


# In[323]:


test_model(model_house_price_prediction.csv)
```

Future Scope of Improvements

Our model had a low RMSE score, but there is still room for improvement. In a real world scenario, we can use such a model to predict house prices. This model should check for new data, once in a month, and incorporate them to expand the dataset and produce better results.

We can try out other dimensionality reduction techniques like Univariate Feature Selection and Recursive feature elimination in the initial stages.

We can try out other advanced regression techniques, like Random Forest and Bayesian Ridge Algorithm, for prediction. Since the data is highly correlated, we should also try Elastic Net regression technique.

Certificate

This is to certify that Ms. Shruti Kiran of Asansol Engineering College, registration number: 10800220016, has successfully completed a project on House Rent Prediction using Machine Learning with Python under the guidance of Mr. Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty
Asansol Engineering College

Department of Computer Applications

Asansol Engineering College

Kalyanpur, Sen Raleigh Road, Asansol – 713304

CERTIFICATE

This is to certify that **Ms. Shruti Kiran** department of Computer Application of Asansol Engineering College registration number: 201080100210068 has successfully completed a project on **"House price prediction"** using machine learning with python under the guidance of Prof. Arnab Chakraborty.

Signature of
The Project Guide
Name: Dr./ Mr.....
Designation: Assistant Professor

Recommendation & Signature of
The Principal
Name: Dr. P.P. Bhattacharya

Recommendation & Signature of
The Head of Department
Name: Dr. Anup Kumar Mukhopadhyay

Recommendation & Signature of
Internal / External Examiners

Certificate

This is to certify that Ms. Priya Jha of Asansol Engineering College, registration number: 10800220020, has successfully completed a project on House Rent Prediction using Machine Learning with Python under the guidance of Mr. Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty
Asansol Engineering College

Department of Computer Applications

Asansol Engineering College

Kalyanpur, Sen Raleigh Road, Asansol – 713304

CERTIFICATE

This is to certify that **Ms. Priya Jha** department of Computer Application of Asansol Engineering College registration number: 201080100210064 has successfully completed a project on **"House price prediction"** using machine learning with python under the guidance of Prof. Arnab Chakraborty.

Signature of
The Project Guide
Name: Dr./ Mr.....
Designation: Assistant Professor

Recommendation & Signature of
The Principal
Name: Dr. P.P. Bhattacharya

Recommendation & Signature of
The Head of Department
Name: Dr. Anup Kumar Mukhopadhyay

Recommendation & Signature of
Internal / External Examiners

Certificate

This is to certify that Ms. Ankita of Asansol Engineering College, registration number: 10800220011, has successfully completed a project on House Rent Prediction using Machine Learning with Python under the guidance of Mr. Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty
Asansol Engineering College

Department of Computer Applications

Asansol Engineering College

Kalyanpur, Sen Raleigh Road, Asansol – 713304

CERTIFICATE

This is to certify that **Ms. Ankita** department of Computer Application of Asansol Engineering College registration number: 201080100210073 has successfully completed a project on **"House price prediction"** using machine learning with python under the guidance of Prof. Arnab Chakraborty.

Signature of
The Project Guide
Name: Dr./ Mr.....
Designation: Assistant Professor

Recommendation & Signature of
The Principal
Name: D r. P.P. Bhattacharya

Recommendation & Signature of
The Head of Department
Name: Dr. Anup Kumar Mukhopadhyay

Recommendation & Signature of
Internal / External Examiners

Certificate

This is to certify that Ms. Ishika Prabhakar of Asansol Engineering College, registration number: 10800220063, has successfully completed a project on House Rent Prediction using Machine Learning with Python under the guidance of Mr. Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty
Asansol Engineering College

Department of Computer Applications

Asansol Engineering College

Kalyanpur, Sen Raleigh Road, Asansol – 713304

CERTIFICATE

This is to certify that **Ms. Ishika Prabhakar** department of Computer Application of Asansol Engineering College registration number: 201080100210021 has successfully completed a project on **"House price prediction"** using machine learning with python under the guidance of Prof. Arnab Chakraborty.

Signature of
The Project Guide
Name: Dr./ Mr.....
Designation: Assistant Professor

Recommendation & Signature of
The Principal
Name: D r. P.P. Bhattacharya

Recommendation & Signature of
The Head of Department
Name: Dr. Anup Kumar Mukhopadhyay

Recommendation & Signature of
Internal / External Examiners