SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | Approved by AICTE

HCL INTERNSHIP

DOMAIN: MACHINE LEARNING

PROJECT TITLE:
House Price Prediction

NAME: G. GRISHMITA

REG NO: 39110337

ROLL NO: 19S115701

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

HOUSE PRICE PREDICTION USING MACHINE LEARNING

Introduction:

Thousands of houses are sold every day. There are some questions every buyer asks himself like: What is the actual price that this house deserves? Am I paying a fair price? In this paper, a machine learning model is proposed to predict a house price based on data related to the house (its size, the year it was built in, etc.).The dataset "AmesHousing" was taken from Kaggle.

Problem statement:

As people believe that the house price depend upon:

1. The square foot area
2. Neighbourhood
3. The no. of bedrooms

But it depends upon many factors such as:

1. No. of storeys
2. Area outside the house
3. Rooms on one floor

Objective:

The main objectives of this study are as follows:

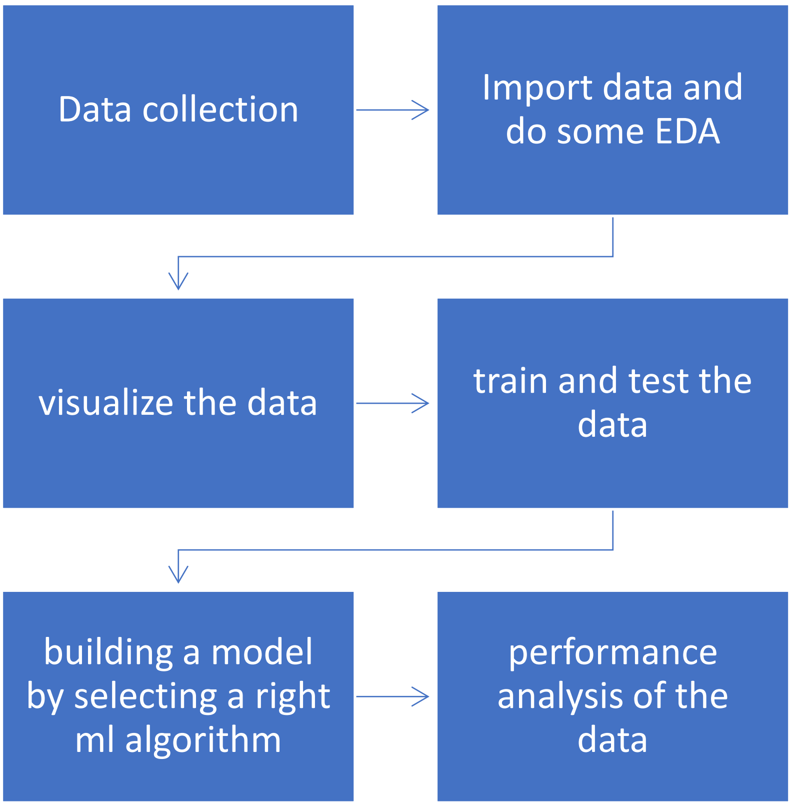
- To apply data pre-processing and preparation techniques in order to obtain clean data
- To build machine learning models able to predict house price based on house features
- To analyze and compare models performance in order to choose the best model

Steps Involved:

1. Importing the required packages into our python environment

2. Importing the house price data and do some EDA on it
3. Data Visualization on the house price data
4. Feature Selection & Data Split
5. Modelling the data using the algorithms
6. Evaluating the built model using the evaluation metrics

System Architecture



Source code

```
!pip install termcolor

# IMPORTING PACKAGES


import pandas as pd # data processing

import numpy as np # working with arrays

import matplotlib.pyplot as plt # visualization

import seaborn as sb # visualization

from termcolor import colored as cl # text customization


from sklearn.model_selection import train_test_split # data split


from sklearn.linear_model import LinearRegression # LR algorithm

from sklearn.linear_model import Ridge # Ridge algorithm

from sklearn.linear_model import Lasso # Lasso algorithm

from sklearn.linear_model import BayesianRidge # Bayesian algorithm

from sklearn.linear_model import ElasticNet # ElasticNet algorithm


from sklearn.metrics import explained_variance_score as evs # evaluation
metric

from sklearn.metrics import r2_score as r2 # evaluation metric


sb.set_style('whitegrid') # plot style

plt.rcParams['figure.figsize'] = (20, 10) # plot size

# IMPORTING DATA
```

```
df = pd.read_csv('./House_Data.csv')

df.set_index('Id', inplace = True)


df.head(5)

df.dropna(inplace = True)


print(cl(df.isnull().sum(), attrs = ['bold']))

df.describe()

print(cl(df.dtypes, attrs = ['bold']))

df['MasVnrArea'] = pd.to_numeric(df['MasVnrArea'], errors = 'coerce')

df['MasVnrArea'] = df['MasVnrArea'].astype('int64')


print(cl(df.dtypes, attrs = ['bold']))

# Data visualization

# Heat Map

sb.heatmap(df.corr(), annot = True, cmap = 'magma')


plt.savefig('heatmap.png')

plt.show()

#Scatter plot

def scatter_df(y_var):

    scatter_df = df.drop(y_var, axis = 1)

    i = df.columns


    plot1 = sb.scatterplot(i[0], y_var, data = df, color = 'orange', edgecolor =
'b', s = 150)

    plt.title('{} / Sale Price'.format(i[0]), fontsize = 16)
```

```
    plt.xlabel('{}'.format(i[0]), fontsize = 14)

    plt.ylabel('Sale Price', fontsize = 14)

    plt.xticks(fontsize = 12)

    plt.yticks(fontsize = 12)

    plt.savefig('scatter1.png')

    plt.show()



    plot2 = sb.scatterplot(i[1], y_var, data = df, color = 'yellow', edgecolor =
'b', s = 150)

    plt.title('{} / Sale Price'.format(i[1]), fontsize = 16)

    plt.xlabel('{}'.format(i[1]), fontsize = 14)

    plt.ylabel('Sale Price', fontsize = 14)

    plt.xticks(fontsize = 12)

    plt.yticks(fontsize = 12)

    plt.savefig('scatter2.png')

    plt.show()



    plot3 = sb.scatterplot(i[2], y_var, data = df, color = 'aquamarine',
edgecolor = 'b', s = 150)

    plt.title('{} / Sale Price'.format(i[2]), fontsize = 16)

    plt.xlabel('{}'.format(i[2]), fontsize = 14)

    plt.ylabel('Sale Price', fontsize = 14)

    plt.xticks(fontsize = 12)

    plt.yticks(fontsize = 12)

    plt.savefig('scatter3.png')

    plt.show()
```

```
    plot4 = sb.scatterplot(i[3], y_var, data = df, color = 'deepskyblue', edgecolor = 'b', s = 150)

    plt.title('{} / Sale Price'.format(i[3]), fontsize = 16)

    plt.xlabel('{}'.format(i[3]), fontsize = 14)

    plt.ylabel('Sale Price', fontsize = 14)

    plt.xticks(fontsize = 12)

    plt.yticks(fontsize = 12)

    plt.savefig('scatter4.png')

    plt.show()



    plot5 = sb.scatterplot(i[4], y_var, data = df, color = 'crimson', edgecolor = 'white', s = 150)

    plt.title('{} / Sale Price'.format(i[4]), fontsize = 16)

    plt.xlabel('{}'.format(i[4]), fontsize = 14)

    plt.ylabel('Sale Price', fontsize = 14)

    plt.xticks(fontsize = 12)

    plt.yticks(fontsize = 12)

    plt.savefig('scatter5.png')

    plt.show()



    plot6 = sb.scatterplot(i[5], y_var, data = df, color = 'darkviolet', edgecolor = 'white', s = 150)

    plt.title('{} / Sale Price'.format(i[5]), fontsize = 16)

    plt.xlabel('{}'.format(i[5]), fontsize = 14)

    plt.ylabel('Sale Price', fontsize = 14)

    plt.xticks(fontsize = 12)

    plt.yticks(fontsize = 12)

    plt.savefig('scatter6.png')
```

```
    plt.show()



    plot7 = sb.scatterplot(i[6], y_var, data = df, color = 'khaki', edgecolor =
'b', s = 150)

    plt.title('{} / Sale Price'.format(i[6]), fontsize = 16)

    plt.xlabel('{}'.format(i[6]), fontsize = 14)

    plt.ylabel('Sale Price', fontsize = 14)

    plt.xticks(fontsize = 12)

    plt.yticks(fontsize = 12)

    plt.savefig('scatter7.png')

    plt.show()



    plot8 = sb.scatterplot(i[7], y_var, data = df, color = 'gold', edgecolor = 'b',
s = 150)

    plt.title('{} / Sale Price'.format(i[7]), fontsize = 16)

    plt.xlabel('{}'.format(i[7]), fontsize = 14)

    plt.ylabel('Sale Price', fontsize = 14)

    plt.xticks(fontsize = 12)

    plt.yticks(fontsize = 12)

    plt.savefig('scatter8.png')

    plt.show()



    plot9 = sb.scatterplot(i[8], y_var, data = df, color = 'r', edgecolor = 'b', s =
150)

    plt.title('{} / Sale Price'.format(i[8]), fontsize = 16)

    plt.xlabel('{}'.format(i[8]), fontsize = 14)

    plt.ylabel('Sale Price', fontsize = 14)

    plt.xticks(fontsize = 12)
```

```
    plt.yticks(fontsize = 12)

    plt.savefig('scatter9.png')

    plt.show()


    plot10 = sb.scatterplot(i[9], y_var, data = df, color = 'deeppink', edgecolor
= 'b', s = 150)

    plt.title('{} / Sale Price'.format(i[9]), fontsize = 16)

    plt.xlabel('{}'.format(i[9]), fontsize = 14)

    plt.ylabel('Sale Price', fontsize = 14)

    plt.xticks(fontsize = 12)

    plt.yticks(fontsize = 12)

    plt.savefig('scatter10.png')

    plt.show()


scatter_df('SalePrice')

# Distribution Plot

sb.distplot(df['SalePrice'], color = 'r')

plt.title('Sale Price Distribution', fontsize = 16)

plt.xlabel('Sale Price', fontsize = 14)

plt.ylabel('Frequency', fontsize = 14)

plt.xticks(fontsize = 12)

plt.yticks(fontsize = 12)


plt.savefig('distplot.png')

plt.show()

# FEATURE SELECTION & DATA SPLIT
```

```
X_var = df[['LotArea', 'MasVnrArea', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF',
'2ndFlrSF', 'GrLivArea', 'GarageArea', 'WoodDeckSF',
'OpenPorchSF']].values

y_var = df['SalePrice'].values



X_train, X_test, y_train, y_test = train_test_split(X_var, y_var, test_size =
0.2, random_state = 0)



print(cl('X_train samples : ', attrs = ['bold']), X_train[0:5])

print(cl('X_test samples : ', attrs = ['bold']), X_test[0:5])

print(cl('y_train samples : ', attrs = ['bold']), y_train[0:5])

print(cl('y_test samples : ', attrs = ['bold']), y_test[0:5])

X_train

# MODELING



# 1. Lr



lr = LinearRegression()

lr.fit(X_train, y_train)

lr_yhat = lr.predict(X_test)



# 2. Ridge



ridge = Ridge(alpha = 0.5)

ridge.fit(X_train, y_train)

ridge_yhat = ridge.predict(X_test)



# 3. Lasso
```

```
lasso = Lasso(alpha = 0.01)

lasso.fit(X_train, y_train)

lasso_yhat = lasso.predict(X_test)


# 4. Bayesian


bayesian = BayesianRidge()

bayesian.fit(X_train, y_train)

bayesian_yhat = bayesian.predict(X_test)


# 5. ElasticNet


en = ElasticNet(alpha = 0.01)

en.fit(X_train, y_train)

en_yhat = en.predict(X_test)

# 1. Explained Variance Score


print(cl('EXPLAINED VARIANCE SCORE:', attrs = ['bold']))

print('---------------------------------------------------------------------------')

print(cl('Explained Variance Score of LR model is {}'.format(evs(y_test,
lr_yhat)), attrs = ['bold']))

print('---------------------------------------------------------------------------')

print(cl('Explained Variance Score of Ridge model is {}'.format(evs(y_test,
ridge_yhat)), attrs = ['bold']))

print('---------------------------------------------------------------------------')

print(cl('Explained Variance Score of Lasso model is {}'.format(evs(y_test,
lasso_yhat)), attrs = ['bold']))
```

```
print('--------------------------------------------------------------------------')

print(cl('Explained Variance Score of Bayesian model is {}'.format(evs(y_test, bayesian_yhat)), attrs = ['bold']))

print('--------------------------------------------------------------------------')

print(cl('Explained Variance Score of ElasticNet is {}'.format(evs(y_test, en_yhat)), attrs = ['bold']))

print('--------------------------------------------------------------------------')

# 2. R-squared



print(cl('R-SQUARED:', attrs = ['bold']))

print('--------------------------------------------------------------------------')

print(cl('R-Squared of LR model is {}'.format(r2(y_test, lr_yhat)), attrs = ['bold']))

print('--------------------------------------------------------------------------')

print(cl('R-Squared of Ridge model is {}'.format(r2(y_test, ridge_yhat)), attrs = ['bold']))

print('--------------------------------------------------------------------------')

print(cl('R-Squared of Lasso model is {}'.format(r2(y_test, lasso_yhat)), attrs = ['bold']))

print('--------------------------------------------------------------------------')

print(cl('R-Squared of Bayesian model is {}'.format(r2(y_test, bayesian_yhat)), attrs = ['bold']))

print('--------------------------------------------------------------------------')

print(cl('R-Squared of ElasticNet is {}'.format(r2(y_test, en_yhat)), attrs = ['bold']))

print('--------------------------------------------------------------------------')

import pickle

pickle.dump(lr,open('houseprice.pkl','wb'))
```