Artificial Intelligence (UCS411)

LAB PROJECT SUBMISSION

Submitted to: Ms Swati Kumari

Submitted by:
1021033531,102283025,102153030,102153033

Problem Statement:

The target variable is the median house value for California districts, expressed in hundreds of thousands of dollars ($100,000).

Description of the problem:

This dataset was obtained from the StatLib repository. https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

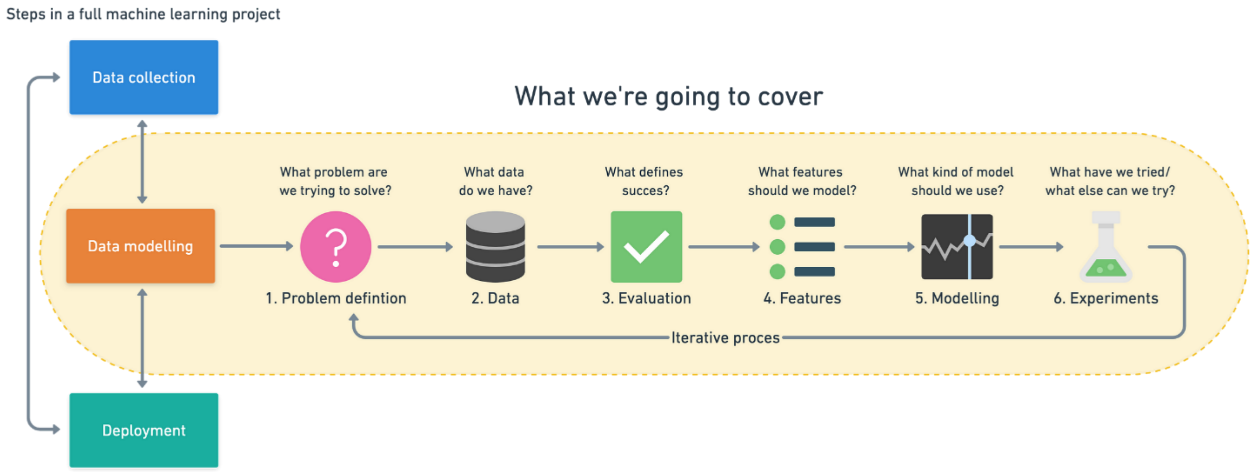
This dataset was derived from the 1990 U.S. census, using one row per census block group. A block group is the smallest geographical unit for which the U.S. Census Bureau publishes sample data (a block group typically has a population of 600 to 3,000 people).

A household is a group of people residing within a home. Since the average number of rooms and bedrooms in this dataset are provided per household, these columns may take surprisingly large values for block groups with few households and many empty houses, such as vacation resorts.

We strive to obtain the median house value for the California district, as median prices are a useful tool for understanding the price changes of properties that have transacted in a market. Furthermore, the median is more accurate than the average because it is less affected by a few unusually high or low sale prices.

Hence this helps property dealers and buyers alike in understanding the prices of houses in a particular area, and how they rise and fall, and their relation to the larger framework of property price growth.

Code and Explanation:



1. **Problem definition** — The given model is one of supervised learning, where we are going to use regression, since we have to find a numerical quantity.
2. **Data** — California Housing Dataset https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html
3. **Evaluation** — Evaluation using coefficient of determination(r^2),cross validation score and mean absolute error.
4. **Features** — 8 numeric, predictive attributes and the target
5. **Modelling** — We use both the Ridge Regressor and the RandomForestRegressor, with the latter having the highest score.
6. **Experimentation** — What else could we try? Does our deployed model do as we expected? How do the other steps change based on what we've found?

The above is the framework we rely on.
The following is how we approached the problem-

1) Firstly, import the dataset from the scikit library

```
In [65]: #Importing our Dataset, in this case, it is the California Housing Dataset
         # The target variable is the median house value for California districts, expressed in hundreds of thousands of dollars ($100,00
         # This dataset was derived from the 1990 U.S. census, using one row per census block group.
         # A block group is the smallest geographical unit for which the U.S. Census Bureau publishes sample data (a block group typicall
         # For more info see: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html

         from sklearn.datasets import fetch_california_housing
         housing = fetch_california_housing()
         housing
```

```
Out[65]: {'data': array([[   8.3252    ,   41.        ,    6.98412698, ...,    2.55555556,
                   37.88      , -122.23      ],
                [   8.3014    ,   21.        ,    6.23813708, ...,    2.10984183,
                   37.86      , -122.22      ],
                [   7.2574    ,   52.        ,    8.28813559, ...,    2.80225989,
                   37.85      , -122.24      ],
                ...,
                [   1.7       ,   17.        ,    5.20554273, ...,    2.3256351 ,
                   39.43      , -121.22      ],
                [   1.8672    ,   18.        ,    5.32951289, ...,    2.12320917,
                   39.43      , -121.32      ],
                [   2.3886    ,   16.        ,    5.25471698, ...,    2.61698113,
                   39.37      , -121.24      ]]),
          'target': array([4.526, 3.585, 3.521, ..., 0.923, 0.847, 0.894]),
          'frame': None,
          'target_names': ['MedHouseVal'],
          'feature_names': ['MedInc',
           'HouseAge',
           'AveRooms',
           'AveBedrms',
           'Population',
           'AveOccup',
           'Latitude',
           'Longitude'],
```

Next, we imported built-in modules –

Importing built-in modules

```
In [66]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import sklearn
         from sklearn.model_selection import train_test_split
         %matplotlib inline
```

A brief description of the modules imported is-

1)Pandas-pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

2)Numpy- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

3)matplotlib- Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

4)sklearn-scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms.

We then preprocessed the Dataset before building the Machine Learning model.

Data Preprocessing

In [67]:
```
housing_df = pd.DataFrame(housing["data"],columns=housing['feature_names'])
housing_df
```

Out[67]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25
...	...	...	...	...	...	...	...	...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24

20640 rows × 8 columns

We firstly took out the dataframe with the columns as feature_names from the larger dataset. We used the features inorder to predict the label, which is the median house value.

We then used the describe function in order to describe the various mathematical factors of the numerical columns, like count,std,max,min mean of the various features which might be useful later.

After that we used the info() function, which shows the datatypes of the various columns along with their indexes and their null criteria. This helped us in the future as we got to know that there is no missing data and that all the columns having floating point numerical values, hence we would not have to convert any column containing non-numerical values into numerical ones before making our model.

In [121]:
```
housing_df.describe() #describing the numerical columns of the data
```
Out[121]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	Median_House_Value
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070655	35.631861	-119.569704	2.068558
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386050	2.135952	2.003532	1.153956
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692308	32.540000	-124.350000	0.149990
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429741	33.930000	-121.800000	1.196000
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818116	34.260000	-118.490000	1.797000
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282261	37.710000	-118.010000	2.647250
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.333333	41.950000	-114.310000	5.000010

In [122]:
```
housing_df.info()#gives information about the datatypes of features along with their indexes
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   MedInc              20640 non-null  float64
 1   HouseAge            20640 non-null  float64
 2   AveRooms            20640 non-null  float64
 3   AveBedrms           20640 non-null  float64
 4   Population          20640 non-null  float64
 5   AveOccup            20640 non-null  float64
 6   Latitude            20640 non-null  float64
 7   Longitude           20640 non-null  float64
 8   Median_House_Value  20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
```

In [123]:
```
housing_df.corr()#Defines the correlation between two features as a function between 0 and 1/-1
```
Out[123]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	Median_House_Value
MedInc	1.000000	-0.119034	0.326895	-0.062040	0.004834	0.018766	-0.079809	-0.015176	0.688075
HouseAge	-0.119034	1.000000	-0.153277	-0.077747	-0.296244	0.013191	0.011173	-0.108197	0.105623
AveRooms	0.326895	-0.153277	1.000000	0.847621	-0.072213	-0.004852	0.106389	-0.027540	0.151948
AveBedrms	-0.062040	-0.077747	0.847621	1.000000	-0.066197	-0.006181	0.069721	0.013344	-0.046701
Population	0.004834	-0.296244	-0.072213	-0.066197	1.000000	0.069863	-0.108785	0.099773	-0.024650
AveOccup	0.018766	0.013191	-0.004852	-0.006181	0.069863	1.000000	0.002366	0.002476	-0.023737
Latitude	-0.079809	0.011173	0.106389	0.069721	-0.108785	0.002366	1.000000	-0.924664	-0.144160
Longitude	-0.015176	-0.108197	-0.027540	0.013344	0.099773	0.002476	-0.924664	1.000000	-0.045967
Median_House_Value	0.688075	0.105623	0.151948	-0.046701	-0.024650	-0.023737	-0.144160	-0.045967	1.000000

In [124]:
```
housing_df.mean()#Gives the mean of the values stored in all the numerical columns
```
Out[124]:
```
MedInc                   3.870671
HouseAge                28.639486
AveRooms                 5.429000
AveBedrms                1.096675
Population            1425.476744
AveOccup                 3.070655
Latitude                35.631861
Longitude             -119.569704
Median_House_Value       2.068558
dtype: float64
```

In [125]:
```
housing_df.head()#Gives a brief snapshot of the first five rows of the dataset
```
Out[125]:

MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	Median_House_Value

Next we used the corr() function which is used to determine the correlation between the columns. The correlation between median house value and other columns gives us a measure of which columns will be most useful while making our model. A value close to 1 or -1 indicates strong correlation between two columns.

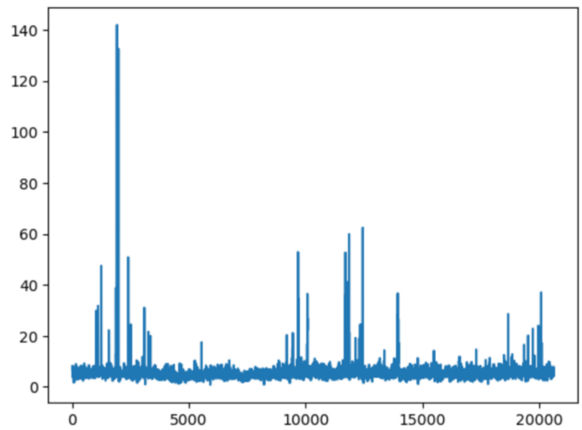
The mean function gives the mean of the columns having numerical datatype.

The head and the tail functions simply give us a snapshot of the data we are dealing with.

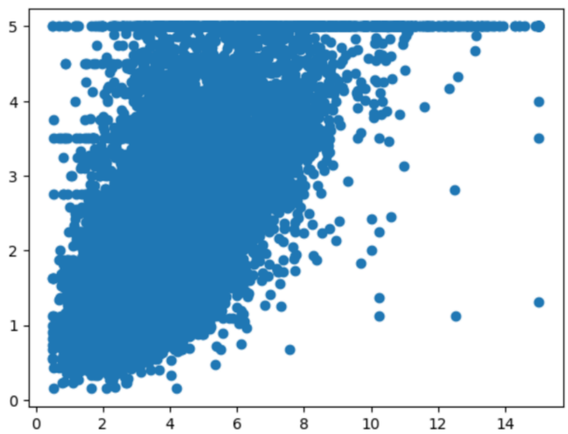
We then plotted some columns using the matplotlib module, along with their index. We then plotted some columns along the scatter plot, in order to find outliers in the data and check if we could remove some columns/reduce data in order to make our model more efficient.

Plotting some columns of the dataset intially

In [126]:
```
housing_df["AveRooms"].plot()#Gives the plot of the specified column along the index values
```
Out[126]: `<Axes: >`



In [127]:
```
plt.scatter(x=housing_df['MedInc'],y=housing_df['Median_House_Value'])
```
Out[127]: `<matplotlib.collections.PathCollection at 0x21bbc2007f0>`



We then made the column containing the Median House Values equivalent to the target column, since we have to find the Median House Values from the other 8 columns that we have.

```
In [70]: housing_df["Median_House_Value"] = housing["target"]#Setting the Median House Income values as the target, these are what we have
         housing_df.head()
```

Out[70]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	Median_House_Value
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

We see that the column gets added along the end of our dataset as a target variable.

Making the Machine Learning Model

We start off by making the machine learning model using three steps-

Three main things that we have to do-

1:Split the data into features and labels (usually 'X' and 'Y')

2.Filling (also called imputing) or disregarding missing values

3.Converting non-numerical values to numerical values(also called featuring coding)

Since there are no missing values in our dataset, and all the columns contain numerical floating type values, so steps 2 and 3 do not apply to our dataset.

We use 8 numeric and predictive values in order to compute the target.

We have initially used the Ridge Regressor in order to calculate the score, basically how good our predictions formed from the training data by the model when applied on the testing data are.
It gives a low value of 0.5758549611440126.

7.2.7. California Housing dataset

Data Set Characteristics:

Number of Instances:	20640
Number of Attributes:	8 numeric, predictive attributes and the target
Attribute Information:	• MedInc median income in block group • HouseAge median house age in block group • AveRooms average number of rooms per household • AveBedrms average number of bedrooms per household • Population block group population • AveOccup average number of household members • Latitude block group latitude • Longitude block group longitude
Missing Attribute Values:	None

Making the Machine Learning Model by Ridge Regressor

Three main things that we have to do-

1:Split the data into features and labels (usually 'X' and 'Y')

2.Filling (also called imputing) or disregarding missing values

3.Converting non-numerical values to numerical values(also called featuring coding)

In [109]:
```
#We have to split data into features and targets, so bascially "X" and "y"
#Setup random seed
np.random.seed(42)

#Create the data,basically the label
X = housing_df.drop("Median_House_Value",axis=1)
y = housing_df["Median_House_Value"]#Median house price is in $100,000

#Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2)

#Instantiate and fit the model
from sklearn.linear_model import Ridge
model = Ridge()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
model.score(X_test,y_test)
```

Out[109]:
```
0.5758549611440126
```

In [75]:
```
y_pred
```

Out[75]:
```
array([0.71923978, 1.76395141, 2.70909238, ..., 4.46864495, 1.18785499,
       2.00912494])
```

We then made a RandomForestRegressor model which came up with a score of 0.8, which was good, and we tried all other regressors, but they did not give a higher r^2 value, so we chose to stick with it.

Making the Machine Learning Model by RandomForestRegressor

```
In [120]: from sklearn.ensemble import RandomForestRegressor
          np.random.seed(42)
          from sklearn.svm import SVR
          sgd=SVR()

          X = housing_df.drop("Median_House_Value", axis=1)
          y = housing_df["Median_House_Value"]

          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)


          model = RandomForestRegressor(n_jobs=-1)
          model.fit(X_train,y_train)
          y_pred2=model.predict(X_test)
          model.score(X_test,y_test)

Out[120]: 0.8065734772187597

In [80]: y_pred2

Out[80]: array([0.49384  , 0.75494  , 4.9285964, ..., 4.8363785, 0.71782  ,
                1.67901  ])

In [88]: from sklearn.metrics import accuracy_score, r2_score
         r2_score(y_test,y_pred2)

Out[88]: 0.8065734772187598

In [ ]:
```

We then made predictions with our model, where we tried to find the output of a particular column within the test data, and then displayed the actual numeric values of the test data in an array. We then found that there was a mean absolute error of about 0.3.

Making Predictions with our Model

```
In [84]: from sklearn.ensemble import RandomForestRegressor

         np.random.seed(42)

         #Create the data
         X = housing_df.drop("Median_House_Value", axis=1)
         y = housing_df["Median_House_Value"]

         #Splitting the training and test sets
         X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2)

         #Creat model instance
         model = RandomForestRegressor(n_jobs=-1)

         #Lets fit the model to the data
         model.fit(X_train,y_train)

         #We are going to make predictions
         y_preds = model.predict(X_test)

In [85]: y_preds[:10]

Out[85]: array([0.49384  , 0.75494  , 4.9285964, 2.54316  , 2.33176  , 1.6525301,
                2.34323  , 1.66182  , 2.47489  , 4.8344779])

In [83]: np.array(y_test[:10])

Out[83]: array([0.477  , 0.458  , 5.00001, 2.186  , 2.78   , 1.587  , 1.982  ,
                1.575  , 3.4    , 4.466  ])
```

```
In [12]: #The average difference between the predicted values and the true values
         from sklearn.metrics import mean_absolute_error
         mean_absolute_error(y_test,y_preds)

Out[12]: 0.32659871732073664

In [13]: len(y_test)

Out[13]: 4128

In [14]: housing_df["Median_House_Value"]

Out[14]: 0        4.526
         1        3.585
         2        3.521
         3        3.413
         4        3.422
                  ...
         20635    0.781
         20636    0.771
         20637    0.923
         20638    0.847
         20639    0.894
         Name: Median_House_Value, Length: 20640, dtype: float64
```

We then evaluated our Machine Learning model along various parameters, mainly the coefficient of determination, cross validation score, and mean absolute error.

Evaluating our Machine Learning Model

There are three ways primarily to do so: 1:Estimator's built-in score method 2:The scoring parameter 3:Problem-specific metric functions

```
In [15]: from sklearn.ensemble import RandomForestRegressor

         X = housing_df.drop("Median_House_Value", axis=1)
         y = housing_df["Median_House_Value"]

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

         model = RandomForestRegressor()
         model.fit(X_train,y_train)
         model.score(X_test,y_test)#coefficient determination(r^2) is the main scoring paramter of Regressors.
         #It is the proportion of the dependence of the dependent variable(Median_House_Value) on the independent variables(The rest of t

Out[15]: 0.801891020387404

In [16]: model.score(X_train,y_train)

Out[16]: 0.9739540089728075
```

The value of the coefficient of determination comes about 0.80, which means that our predictions made by the model are about 80 percent accurate. The conclusion is that we can make an 80 percent accurate guess about the median house values from the 8 features we have used, using our machine learning model.

We then used the cross validation score in order to get a more holistic score for our model. It came out to be about 0.65, when the training and testing data were taken for 5 separate times, cv=5.

Evaluating using a scoring paramter

In [19]:
```
from sklearn.model_selection import cross_val_score

from sklearn.ensemble import RandomForestRegressor

X = housing_df.drop("Median_House_Value", axis=1)
y = housing_df["Median_House_Value"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

model = RandomForestRegressor()
model.fit(X_train,y_train)
model.score(X_test,y_test)
```

Out[19]:
```
0.8016052075802582
```

In [18]:
```
cross_val_score(model,X,y,cv=5)#Makes 5 different splits of the data, thus enabling us to get a better picture of our score
```

Out[18]:
```
array([0.51696995, 0.70143497, 0.74009095, 0.62805561, 0.68181405])
```

In [20]:
```
np.random.seed(42)

model_single_score = model.score(X_test,y_test)

model_cross_val_score = np.mean(cross_val_score(model,X,y,cv=5))

model_single_score, model_cross_val_score
```

Out[20]:
```
(0.8016052075802582, 0.6520824166120266)
```

We then calculated the mean absolute error(MAE), in order to find out how our model's predicted values were different from the true values.

It was indicative of how wrong our model could be, and it was then calculated that there was a deviation of 0.33 between the predicted and the actual test values on which the predictions were made.

Evaluating using MAE

In [ ]:
```
Mean Absolute Error

MAE is the average of the absolute differences between predictions and actual values

It is an indicator of how wrong our models are
```

In [35]:
```
from sklearn.metrics import mean_absolute_error

y_preds = model.predict(X_test)
mae = mean_absolute_error(y_test, y_preds)
mae
```

Out[35]:
```
0.3326664143653102
```

In [36]:
```
y_preds
```

Out[36]:
```
array([4.9886395, 3.0432   , 1.85535  , ..., 1.83101  , 4.9969499,
       2.1696101])
```

In [37]:
```
y_test
```

Out[37]:
```
18287    5.00001
5430     3.01900
7101     1.99700
16590    1.12500
5354     5.00001
          ...
18539    2.40500
13882    0.99300
1761     1.39100
18352    5.00001
10838    4.30900
Name: Median_House_Value, Length: 4128, dtype: float64
```

```
In [ ]: This means that on an average, our prediction deviates from the actual value by 0.33
```

```
In [41]: df = pd.DataFrame(data={"actual values":y_test,
                                  "predicted values":y_preds})
         df["difference"] = df["actual values"] - df["predicted values"]
         df.head(5)
```

Out[41]:

	actual values	predicted values	difference
18287	5.00001	4.988639	0.011371
5430	3.01900	3.043200	-0.024200
7101	1.99700	1.855350	0.141650
16590	1.12500	1.244570	-0.119570
5354	5.00001	2.746190	2.253820

```
In [43]: df["difference"].mean()#Includes negative sign hence there is deviation from MAE
Out[43]: -0.019569243338177847
```

```
In [44]: np.abs(df["difference"]).mean()#MAE using formulas and differences
Out[44]: 0.3326664143653102
```

Conclusion: We trained about 80% of the data and we tested our predictions on the remaining 20%, we used the RandomForestRegressor to get a score for the predictions made by our model, and we got a decent score of 0.8 after applying all improvement techniques. We evaluated our model based on various parameters like coefficient of determination,cross_validation score, mean absolute error etc. We successfully predicted about the Median House Values as the label from the other 8 columns we used as features.