MINOR PROJECT REPORT

Boston House Price Prediction
Using Machine Learning

Abstract

The phenomenon of the falling or rising of the house prices has attracted interest from the researcher as well as many other interested parties. There have been many previous research works that used various regression techniques to address the question of the changes in house prices. This work considers the issue of changing house prices as a classification problem and applies machine learning techniques to predict whether house prices will rise or fall. This work applies various feature selection techniques such as variance influence factor, Information value, principal component analysis, and data transformation techniques such as outlier and missing value treatment as well as box-cox transformation techniques. The performance of the machine learning techniques is measured by the four parameters of accuracy, precision, specificity, and sensitivity. The work considers two discrete values 0 and 1 as respective classes. If the value of the class is 0 then we consider that the price of the house has decreased and if the value of the class is 1 then we consider that the price of the house has increased.

Introduction

For this project, we have taken the dataset of Boston houses, which is the fastest-growing city in the US. The growth of this city has already suppressed the economic growth rate of its own country. The city is the economic and cultural anchor of a substantially larger metropolitan area known as Greater Boston, a metropolitan statistical area (MSA) home to a census-estimated 4.8 million people in 2016 and ranking as the tenth-largest MSA in the country. A broader combined

statistical area (CSA), generally corresponding to the commuting area and including Providence, Rhode Island, is home to some 8.2 million people, making it the sixth-most populous in the United States. All these factors have caused the migration of people from other states to Boston but due to this, the cost-of-living increases, and normal people find it difficult to manage their household. This makes the decisions harder and to cope up with this we used various optimal regression techniques to forecast the actual cost of a property using large data of buy and sales to analyze the property prices that can help us make better decisions. We took the Kaggle dataset, which is an open-source dataset, easy to use, and reliable. For our house price prediction, a dataset was chosen which contains the data of 13320 records and 9 features to train our model. The platform that we used to conduct this research is python and Jupiter notebook. It is a comprehensive environment for interactive and exploratory computing it also supports parallel computing which helps in faster execution. In this paper we took the varies parameters into consideration and put these parameters to various models like Decision Tree Regression, XGBoost, Support Vector Regression (SVR), and Random Forest to predict the value of a house and compared their output with each other based on their errors matrices like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and accuracy.

Data Description

In this project, we will develop and evaluate the performance and the predictive power of a model trained and tested on data collected from houses in Boston's suburbs. Once we get a good fit, we will use this model to predict the monetary value of a house located in the Boston area. A model like this would be very valuable for a real estate agent who could make use of the information provided on a daily basis. The dataset used in this project comes from the UCI Machine Learning Repository. This data was collected in 1978 and each of the 506 entries represents aggregate information about 14 features of homes from various suburbs located in Boston.

This is an overview of the original dataset, with its original feature

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	11.9

506 rows × 14 columns

The features can be summarized as follows:

- CRIM: This is the per capita crime rate by town
- ZN: This is the proportion of residential land zoned for lots larger than 25,000 sq. Ft.
- INDUS: This is the proportion of non-retail business acres per town.
- CHAS: This is the Charles River dummy variable (this is equal to 1 if tract bounds river; 0 otherwise)
- NOX: This is the concentration of the nitric oxide (parts per 10 million)
- RM: This is the average number of rooms per dwelling
- AGE: This is the proportion of owner-occupied units built prior to 1940
- DIS: This is the weighted distances to five Boston employment center's
- RAD: This is the index of accessibility to radial highways
- TAX: This is the full-value property-tax rate per $10,000
- PTRATIO: This is the pupil-teacher ratio by town
- B: This is calculated as 1000(Bk — 0.63) ², where Bk is the proportion of people of African American descent by town
- LSTAT: This is the percentage lower status of the population
- MEDV: This is the median value of owner-occupied homes in $1000s

Approach

In this my approach to the solution of this project is first we will be doing the visualization of the data evaluation matrices as well such as precision. Understanding the given dataset and helps clean up the given dataset. It gives you a clear picture of the features and the relationships between them. Providing guidelines for essential variables and leaving behind/removing non-essential variables. Handling Missing values or human error.
Identifying outliers. EDA process would be maximizing insights of a dataset.

Visualization

First Importing all the important Libraries/Modules

```
# Importing Libraries

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

Loading the Dataset and Dropping unnecessary columns

```
# Loading the Datasets

test = pd.read_csv('Boston_test.csv')
train = pd.read_csv('Boston_train.csv')
test.drop(columns=['Unnamed: 0'], axis=0, inplace=True)
train.drop(columns=['Unnamed: 0'], axis=0, inplace=True)
```

```
train.describe()
```

	crim	zn	indus	chas	nox	rm	age	
count	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.00
mean	0.401659	15.327635	8.435670	0.076923	0.510737	6.403900	60.817949	4.42
std	0.641716	25.605040	6.088947	0.266850	0.102256	0.676424	28.393094	1.96
min	0.006320	0.000000	0.460000	0.000000	0.385000	4.903000	2.900000	1.32
25%	0.057845	0.000000	4.025000	0.000000	0.437450	5.949500	36.150000	2.76
50%	0.132620	0.000000	6.200000	0.000000	0.493000	6.266000	62.000000	4.09
75%	0.404865	22.000000	10.010000	0.000000	0.544000	6.733000	88.450000	5.87
max	4.097400	100.000000	25.650000	1.000000	0.871000	8.725000	100.000000	9.22

Preprocessing

Now to get a count of all of the columns that contain empty (NaN, NAN, na) values

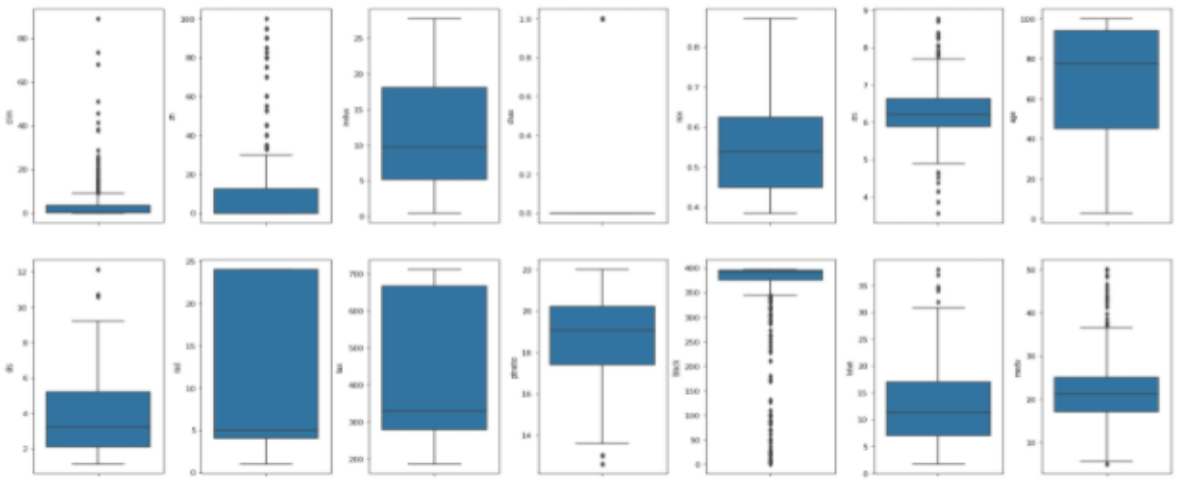
```
df.isnull().sum()
```

```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
black      0
lstat      0
medv       0
dtype: int64
```

Exploratory Data Analysis (EDA)

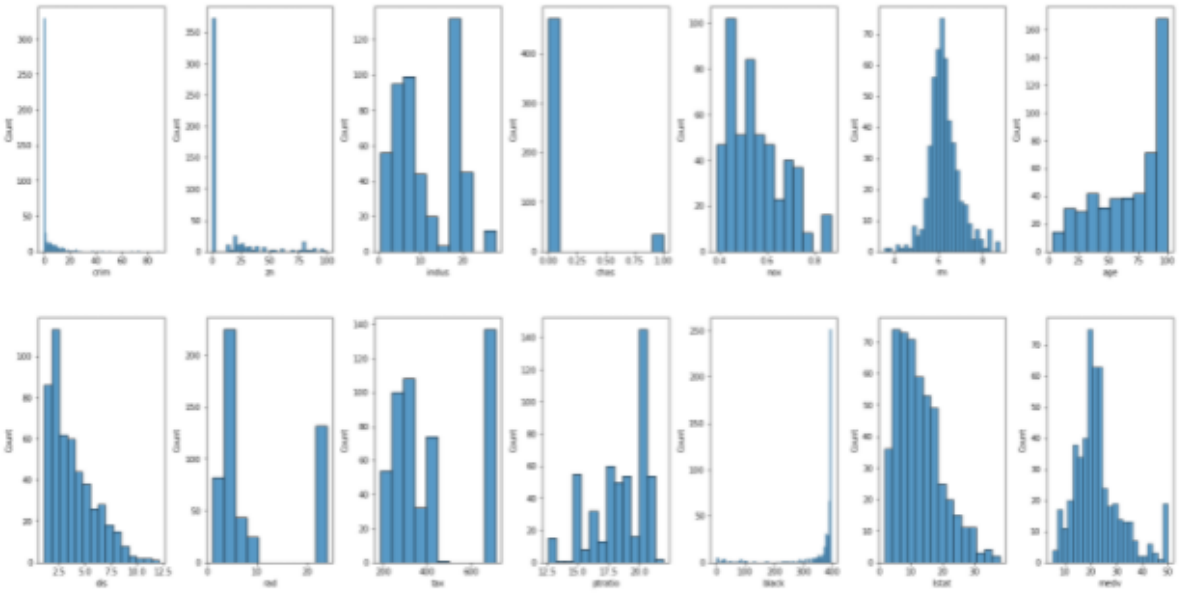
Creating box plots

```
# create box plots
fig, ax = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))
index = 0
ax = ax.flatten()
for col, value in df.items():
    sns.boxplot(y=col, data=df, ax=ax[index])
    index += 1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```



Creating dist plots

```
# create dist plot
fig, ax = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))
index = 0
ax = ax.flatten()
for col, value in df.items():
    sns.histplot(value, ax=ax[index])
    index += 1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```



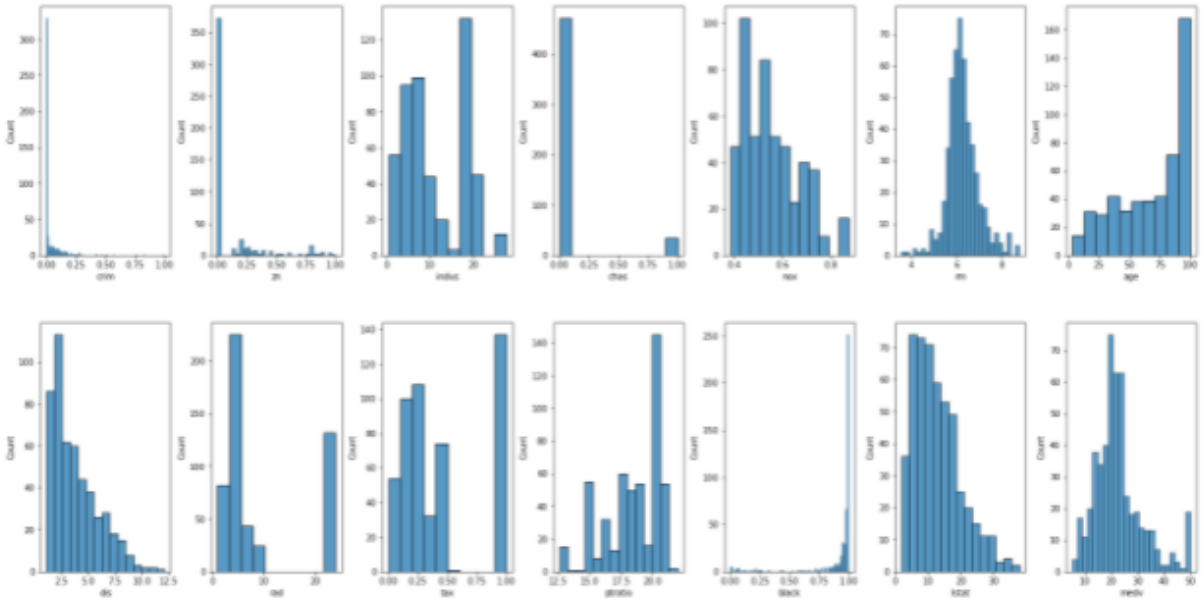
Min-Max Normalization

Finding minimum and maximum of the following columns: [crim, zn, tax, black]

```
#Min-Max Normalization

cols = ['crim', 'zn', 'tax', 'black']
for col in cols:
    minimum = min(df[col])
    maximum = max(df[col])
    df[col] = (df[col] - minimum) / (maximum - minimum)
```

```
fig, ax = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))
index = 0
ax = ax.flatten()
for col, value in df.items():
    sns.histplot(value, ax=ax[index])
    index += 1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```



Now applying Standardization to the previous columns, so that the visualization becomes simple

```
# Standardization
from sklearn import preprocessing
scalar = preprocessing.StandardScaler()

scaled_cols = scalar.fit_transform(df[cols])
scaled_cols = pd.DataFrame(scaled_cols, columns=cols)
scaled_cols.head()
```

	crim	zn	tax	black
0	-0.419782	0.284830	-0.666608	0.441052
1	-0.417339	-0.487722	-0.987329	0.441052
2	-0.417342	-0.487722	-0.987329	0.396427
3	-0.416750	-0.487722	-1.106115	0.416163
4	-0.412482	-0.487722	-1.106115	0.441052

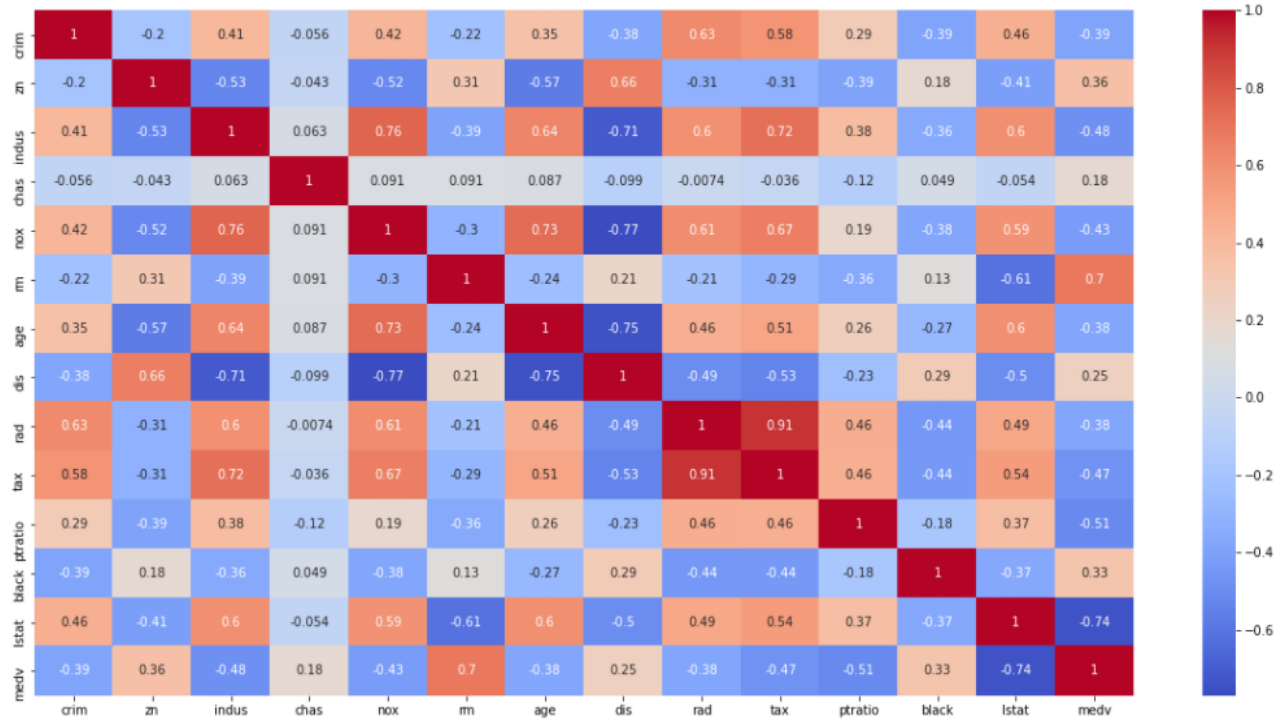
Correlation Matrix

Visualizing the correlation by creating a heat map

```
#Correlation matrix

corr = df.corr()
plt.figure(figsize=(20, 10))
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

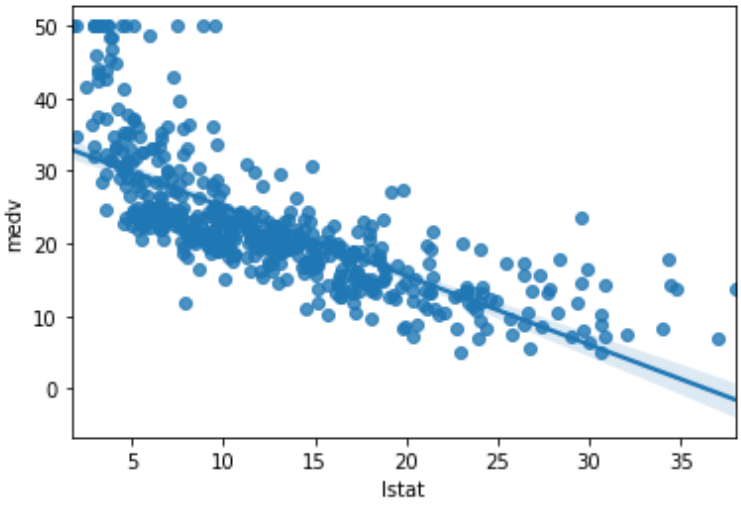
```
<AxesSubplot:>
```



From the above heatmap, you can see lstat and rm is highly correlating with the target
So using regression plot to show the relation between lstat, rm and medv

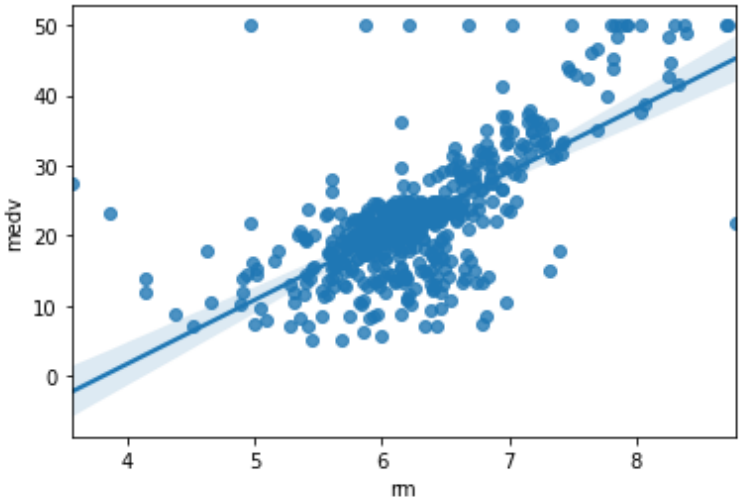
```
sns.regplot(y=df['medv'], x=df['lstat'])
```

```
<AxesSubplot:xlabel='lstat', ylabel='medv'>
```




```
sns.regplot(y=df['medv'], x=df['rm'])
```

```
<AxesSubplot:xlabel='rm', ylabel='medv'>
```



Splitting the dataset

Input split

```
# Input split
X = df.drop(columns=['medv', 'rad'], axis=1)
y = df['medv']
```

Model Training

Here we will be training our model by cross-validation

Form sklearn.model_selection

```
# Model Training

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
def train(model, X, y):
    x_train, x_test, y_train, y_test = train_test_split(X, y, random_state=42)
    model.fit(x_train, y_train)

    pred = model.predict(x_test)
    cv_score = cross_val_score(model, X, y, scoring='neg_mean_squared_error', cv
    cv_score = np.abs(np.mean(cv_score))

    print("model report")
    print("MSE:",mean_squared_error(y_test, pred))
    print('cv score:', cv_score)
```

ALGORITHMS:

Linear Regression

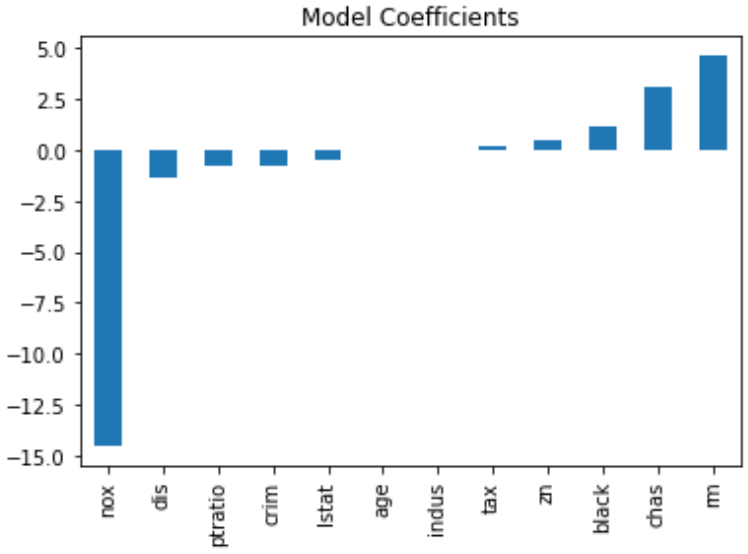
And then Importing Linear Regression modules from sklearn.linear_model and the training the model(X,y)

We will get the report

```
from sklearn.linear_model import LinearRegression
model = LinearRegression(normalize=True)
train(model, X, y)
coef = pd.Series(model.coef_, X.columns).sort_values()
coef.plot(kind='bar', title='Model Coefficients')
```

```
model report
MSE: 23.87100506736489
cv score: 35.58136621076922
```

```
<AxesSubplot:title={'center':'Model Coefficients'}>
```



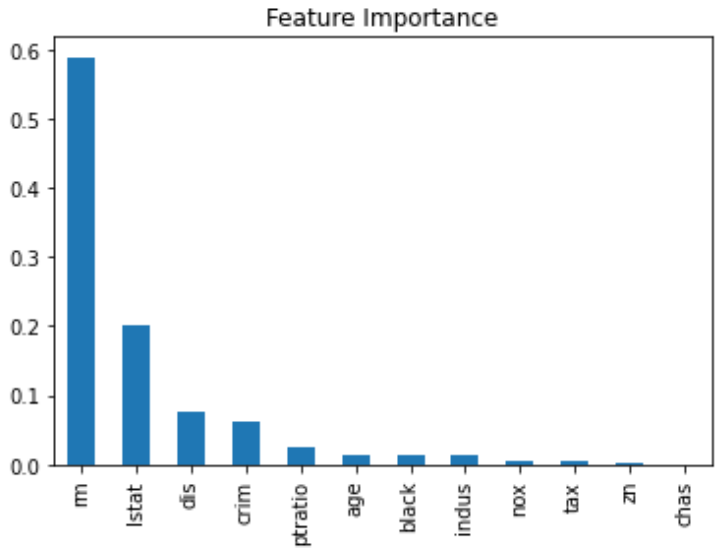
Decision Tree Regressor

Importing DecisionTreeRegressor modules from sklearn.tree and training the model(X,y)
We will get the report

```
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor()
train(model, X, y)
coef = pd.Series(model.feature_importances_, X.columns).sort_values(ascending=Fa
coef.plot(kind='bar', title='Feature Importance')
```

```
model report
MSE: 11.293385826771654
cv score: 38.119730926033775
```

```
<AxesSubplot:title={'center':'Feature Importance'}>
```



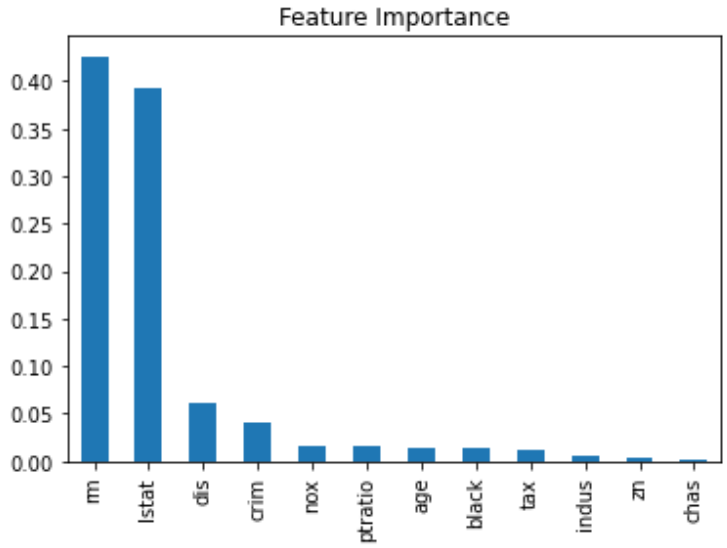
Random Forest Regressor

Importing RandomForestRegressor modules from sklearn.ensemble and training the model(X,y)
We will get the report

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()
train(model, X, y)
coef = pd.Series(model.feature_importances_, X.columns).sort_values(ascending=Fa
coef.plot(kind='bar', title='Feature Importance')
```

```
model report
MSE: 9.877787984251965
cv score: 20.511559817627635

<AxesSubplot:title={'center':'Feature Importance'}>
```



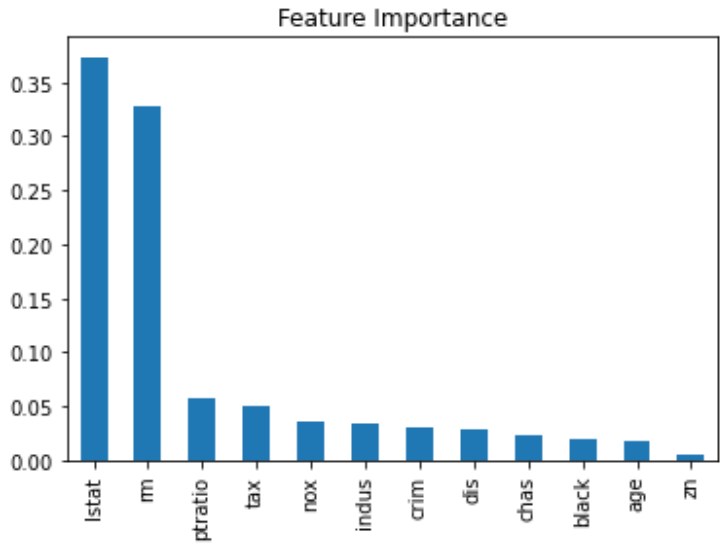
Extra Trees Regressor

Importing ExtraTreesRegressor modules from sklearn.ensemble and training the model(X,y)
We will get the report

```
from sklearn.ensemble import ExtraTreesRegressor
model = ExtraTreesRegressor()
train(model, X, y)
coef = pd.Series(model.feature_importances_, X.columns).sort_values(ascending=Fa
coef.plot(kind='bar', title='Feature Importance')
```

```
model report
MSE: 10.998954921259841
cv score: 19.55976844216655
```

```
<AxesSubplot:title={'center':'Feature Importance'}>
```



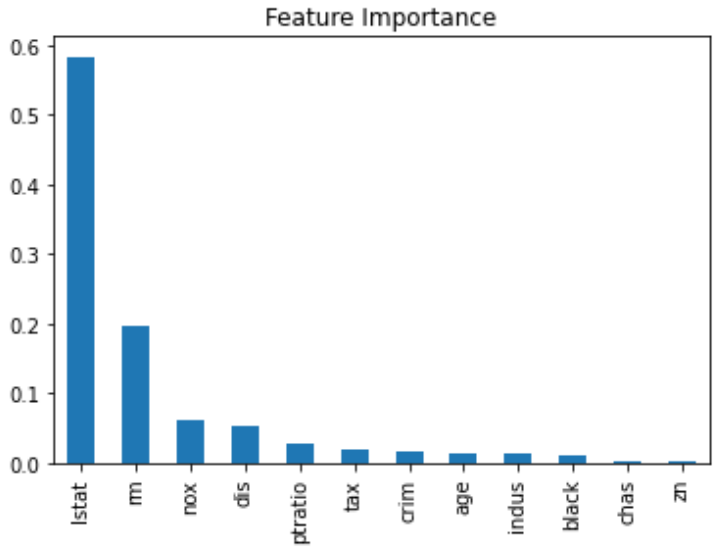
XG Boost Prediction

Importing ExtraTreesRegressor modules from sklearn.ensemble and training the model(X,y)
We will get the report

```
import xgboost as xgb
model = xgb.XGBRegressor()
train(model, X, y)
coef = pd.Series(model.feature_importances_, X.columns).sort_values(ascending=Fa
coef.plot(kind='bar', title='Feature Importance')
```

```
model report
MSE: 10.229776363874551
cv score: 18.766198044819188
```

```
<AxesSubplot:title={'center':'Feature Importance'}>
```



On comparing various algorithms, I found that on test data XG boost performed the best with an accuracy of 63% while on training data decision tree performed the best with an accuracy of 99%.

Prediction of Linear Regression model

Predicting the train and test data model score using R squarred metrics

```
# R squarred metrics

from sklearn import metrics
score_train = metrics.r2_score(y_train, training_data_prediction)
```

```
print(score_train)
```

```
0.7149187097459346
```

```
# Prediction for Test Data

test_data_prediction = lr.predict(x_test)
```

```
# R squarred metrics

score_test = metrics.r2_score(y_test, test_data_prediction)
print(score_test)
```

```
0.7789181825925791
```

```
print('Score_for_Train:', score_train)
print('Score_for_Test:', score_test)
```

```
Score_for_Train: 0.7149187097459346
Score_for_Test: 0.7789181825925791
```

Conclusion & Future Scope

In this study, we used various machine learning algorithms to guess the house prices in Bangalore, the fastest-growing city in Asia. All the methods were described step by step then we took our dataset as an input and applied the five models and generated a CSV folder that had the results of our prediction. Henceforth the presentation of each model was compared based on Features. We found that Decision Tree Regression overfits our data with an accuracy of 99% followed by XG Boost with an overall accuracy of 63% We concluded that the result was in our favour and can help people shortly. But we need a larger dataset and more machine learning algorithms to get even better results with better accuracy.

REFERENCES

https://ieeexplore.ieee.org/document/9362864:

[1] Jingyi Mu, Fang Wu, and Aihua Zhang “Housing Value Forecasting Based on Machine Learning Methods” 2014.

[2] Darshil Shah, Harshad Rajput and Jay Chheda, “House Price Prediction Using Machine Learning and RPA”, Volume 7 Issue 3 (2020).

[3] Thuraiya Mohd, Suraya Masrom and Noraini Johari “Machine Learning Housing Price Prediction In Petaling Jaya, Selangor, Malaysia”, Volume 8 Issue 2 (2019).

[4] G. Naga Satish, Ch. V. Raghavendran, M.D.Sugnana Rao, Ch.Srinivasulu, “House Price Prediction Using Machine Learning”, Volume 8 Issue 9 (2019).

[5] S. S. Gavankar and S. D. Sawarkar, "Eager decision tree," 2017 2nd International Conference for Convergence in Technology (I2CT), Mumbai, 2017, pp. 837-840, DOI: 10.1109/I2CT.2017.8226246

[6] T. Qunzhu, Z. Rui, Y. Yufei, Z. Chengyao, and L. Zhijun, "Improvement of Random Forest Cascade Regression Algorithm and Its Application in Fatigue Detection," 2019 IEEE 2nd International Conference on Electronics Technology (ICET), Chengdu, China, 2019, pp. 499-503, DOI: 10.1109/ELTECH.2019.8839317

[7] Tianqi Chen, Carlos Guestrin XGBoost: A Scalable Tree Boosting System 9 Mar 2016

[8] Kavitha S, Varuna S, and Ramya R, "A comparative analysis on linear regression and support vector regression," 2016 Online International Conference on Green Engineering and Technologies (IC-GET), Coimbatore, 2016, pp. 1-5, DOI: 10.1109/GET.2016.7916627.