HOUSE PRICE PREDICTION

Sadhana Das, Bhawan

Guided by: KIIT Robotics Society

School of Electronics and School of Computer Science
Kalinga Institute of Industrial Technology
1930037@kiit.ac.in , 2129022@kiit.ac.in
robotics.society@kiit.ac.in

Abstract-

In this project, the power of a model is developed where its performance is checked, along with its predictive power is trained and tested on data collected from houses. A model like this would be very handy for a real estate agent who could make use of the information provided daily. The regression algorithms used are linear regression and Random Forest, the study is attempted to analyze the correlation between variables to determine the most important factors that affect house prices. The accuracy of the prediction is evaluated by checking the root square and root mean square error scores of the training model. The test is performed after applying the required pre-processing methods and splitting the data into two parts. But, one part will be used in the training and the other in the test phase. We have also presented a binning strategy that improved the accuracy of the models.

Index Terms-

Random Forest, Linear Regression, Mean Squared Error, Machine Learning, House Prediction

I. INTRODUCTION

Machine learning is a subfield of Artificial Intelligence (AI) that works with algorithms and technologies to extract useful information from data. Machine learning methods are appropriate in big data since attempting to manually process vast volumes of data would be impossible without the support of machines. Machine learning in computer science attempts to solve problems algorithmically rather than purely mathematically. Therefore, it is based on creating algorithms that permit the machine to learn. However, there are two general groups in machine learning which are supervised and unsupervised. Supervised is where the program gets trained on a pre-determined set to be able to predict when new data is given. Unsupervised is where the program tries to find the relationship and the hidden pattern between the data [1].

The performance will be measured upon predicting house prices since the prediction in many regression algorithms relies not only on a specific feature but on an unknown number of attributes that result in the value to be predicted. House prices depend on an individual house specification. Houses have a variant number of features that may not have the same cost due to their location. For instance, a big house may have a higher price if it is located in a desirable rich area than being placed in a poor neighbourhood. The data used in the experiment will be handled by using a combination of pre-processing methods to improve the prediction accuracy.

I.I Aim and Purpose

The No Free Lunch Theorem state that algorithms perform differently when they are used under the same circumstances [2]. This study aims to analyze the accuracy of predicting house prices when using Linear, and Random Forest regression algorithms Thus, the purpose of this study is to deepen the knowledge in regression methods in machine learning.

In addition, the given datasets should be processed to enhance performance, which is accomplished by identifying the necessary features by applying one of the selection methods 2 to eliminate the unwanted variables since each house has its unique features that help to estimate its price. These features may or may not be shared with all houses, which means they do not have the same influence on the house pricing resulting in inaccurate output.

II. STUDY OF SIMILAR PROJECTS OR TECHNOLOGY\ LITERATURE REVIEW

Most of the literature study is based on articles with full text online, open access articles and peer-reviewed publications from search engine Google Scholar, and the search websites; the Research Gate publications instead of textbooks and chapters of books. The literature study endeavors to construct a robust basis on regression techniques, regularization, and artificial neural network in machine learning and on how it can precisely be applied to house prices prediction. The literature study gives an overview of the articles that are related to this study, the feature engineering methods that have been used in this study. As well as evaluation metrics that is used to measure the performance of the algorithms. In addition, the factors that have been used in the Kaggle dataset.

III. BASIC CONCEPTS/ TECHNOLOGY USED

1. Linear Regression

   The model representation is established using the notation, x for input variables also called input variables and y for output or target variables. A pair (x, y) is called a training example and the dataset that has the list of training examples is a training set.

   Function h: X→Y; such that h(x) is a good predictor for corresponding value of y, function 'h' is called a hypothesis. When the target variable that we're trying to predict is continuous, such as in housing example. It's called learning problem or a regression problem.

   It is prone to many problems such as multicollinearity, noises, and overfitting, which effect on the prediction accuracy. Regularized regression plays a significant part in Multiple Linear Regression because it helps to reduce variance at the cost of introducing some bias, avoid the overfitting problem and solve ordinary least squares (OLS) problems. There are two types of regularization techniques L1 norm (least absolute deviations) and L2 norm (least squares). L1 and L2 have different cost functions regarding model complexity [3].

2. Random Forest Regression

   A Random Forest is an ensemble technique qualified for performing classification and regression tasks with the help of multiple decision trees and a method called Bootstrap Aggregation known as Bagging [4].

Decision Trees are used in classification and regression tasks, where the model (tree) is formed of nodes and branches. The tree starts with a root node, while the internal nodes correspond to an input attribute. The nodes that do not have children are called leaves, where each leaf performs the prediction of the output variable [5].

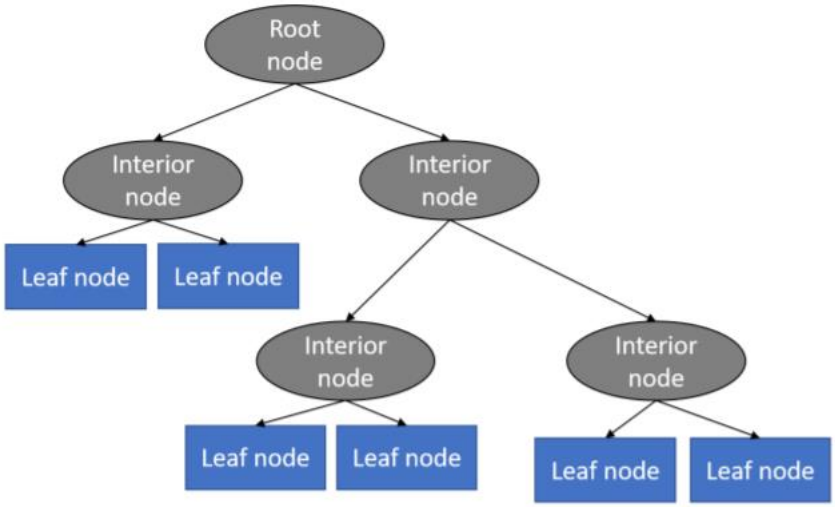


Figure 1. Decision Tree

A Decision Tree can be defined as a model [6]:

$$\varphi = X \mapsto Y$$

Where any node t represents a subspace $Xt \subseteq X$ of the input space and internal nodes t are labelled with a split st taken from a set of questions Q. However, to determine the best separation in Decision Trees, the Impurity equation of dividing the nodes should be taken into consideration, which is defined as:

$$\Delta i(s,t) = i(t) - pLi(t_L) - pRi(t_R)$$

Where $s \in Q$, t_L and tR are left and right nodes, respectively. pL And pR are the proportion of learning samples from $\mathscr{L}t$ going to t_L and t_R respectively.

Random Forest is a model that constructs an ensemble predictor by averaging over a collection of decision trees. Therefore, it is called a forest, and there are two reasons for calling it random. The first reason is growing trees with a random independent bootstrap sample of the data. The second reason is splitting the nodes with arbitrary subsets of features [7]. However, using the bootstrapped sample and considering only a subset of the variables at each step results in a wide variety of trees. The variety is what makes Random Forest more effective than individual Decision Tree.

One advantage is that its computational speed, especially when dealing with large data dimensions.

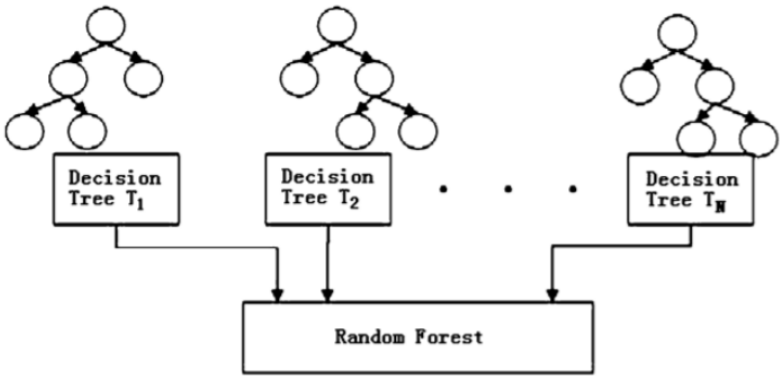


Figure 2. Random Forests

IV. PROJECT COMPONENTS

The algorithms used in this study have different properties that will be used during the implementation. The experiment is done with the IDE Google Colab using Python as a programming language. However, in all algorithms, the data is split into four variables, namely, x_train, x_test, y_train, and y_test, by using train_test_split class from the library sklearn.model_selection. In addition, in all algorithms, the train_test_split class takes as parameters the independent variables, which is the data, the dependent variable.

V. PROPOSED MODEL / ARCHITECTURE / METHODOLOGY/ MODEL TOOL

About the Algorithms used :

1. Linear Regression
2. 2. Random forest Regressor

Machine learning Packages:

Numpy, Pandas, Seaborn, Matplotlib.pyplot and sklearn

The following libraries are imported:
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings

For Model Fitting:

- Multiple linear: Multiple linear is implemented using the Linear Regression from the library sklearn.linear_model. This library takes only the independent variables and dependent variable as parameters.
- Random Forest: Random forest is implemented using the sklearn.ensemble.RandomForestRegressor library. This library takes several parameters to set up the model properties. The model consists of 1200 tree where the max depth of the tree is set to 60.

Data Collection:

I got the Dataset from Kaggle. This Dataset consist several features such as Crime Rate, and Tax and so on. You can download the dataset from Kaggle in csv file format.

```
df=pd.read_csv("Boston.csv")
df.drop(columns=['Unnamed: 0'], axis=0, inplace=True)
df.head()
```

Out[142…

	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

VI. IMPLEMENTATION AND RESULTS

Data Processing:
The dataset already cleaned when we download from the Kaggle. Still for verification we can check for number of null or missing values in the dataset. As well as we need to understand shape of the dataset.

In [145…

```
# check for null values
df.isnull().sum()
```

Out[145…

```
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
```

The target variable is the last one which is called medv.

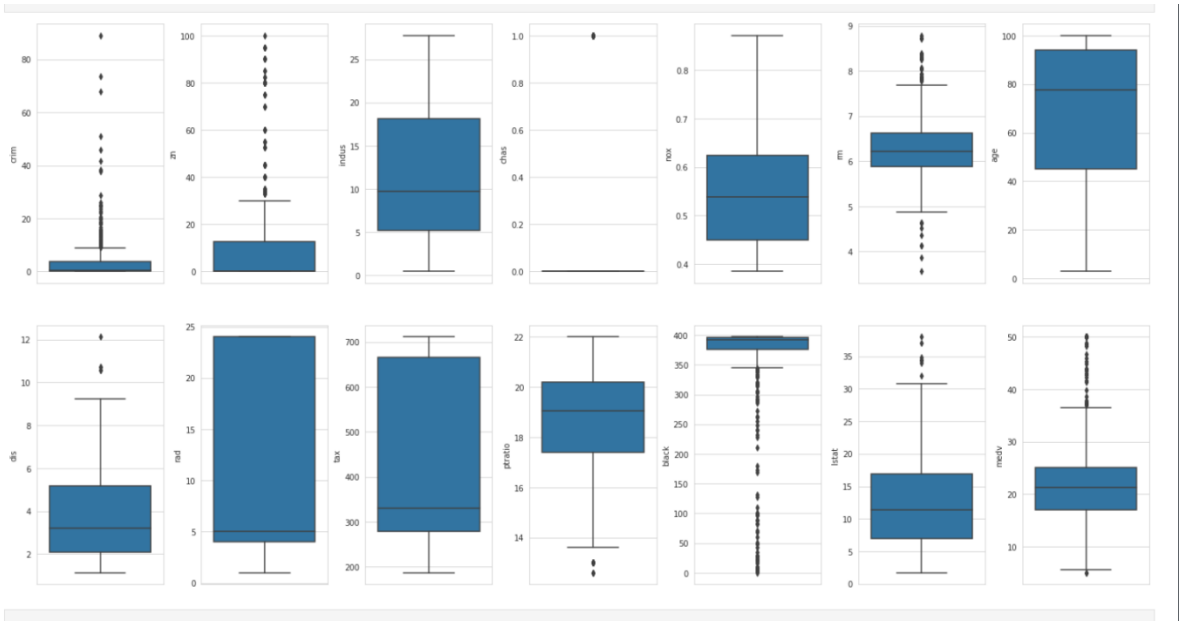
Exploratory Data Analysis:

In statistics, exploratory data analysis (EDA) is an approach to analysing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modelling or hypothesis testing task.

In [152…

```
# 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)
```



.

In [144…

```
# datatype info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64
 9   tax      506 non-null    int64
 10  ptratio  506 non-null    float64
 11  black    506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

In [143…

```
# statistical info
df.describe()
```

Out[143…

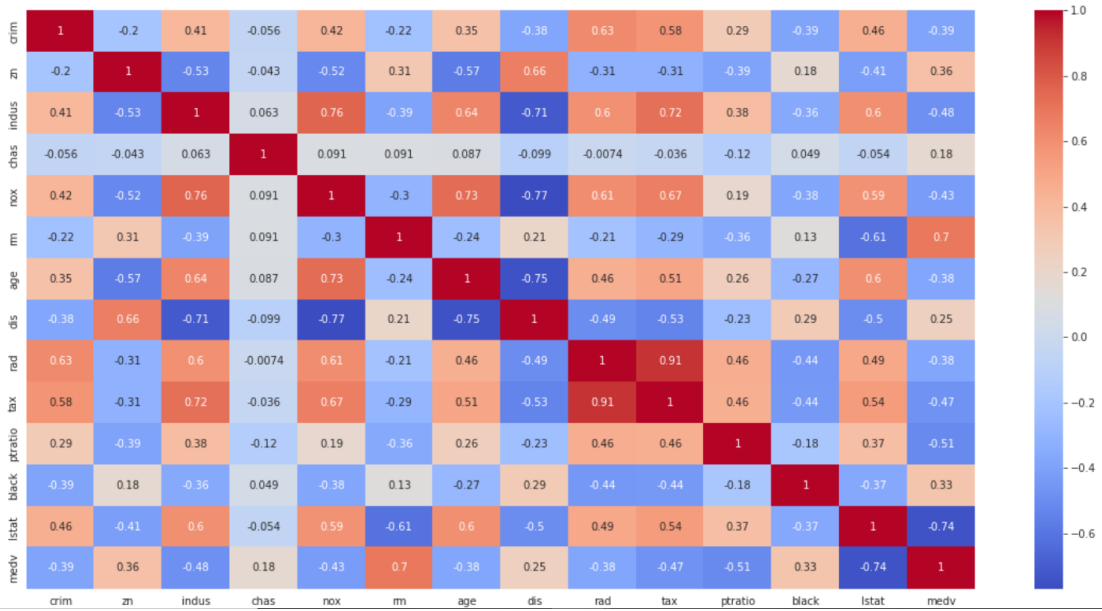
	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.653063	22.532806
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.141062	9.197104
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000	5.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.950000	17.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.360000	21.200000
75%	3.677082	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955000	25.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970000	50.000000

Correlation Matrix

First Understanding the correlation of features between target and other features

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

Out[159… <matplotlib.axes._subplots.AxesSubplot at 0x7efe6fbc7fd0>

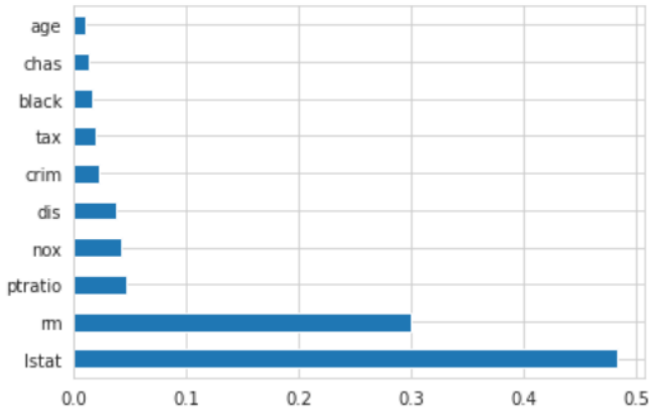


Feature Selection:

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

In [169…

```
# Plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```



Model Fitting:

Linear Regression-

1. Train test split

In [172…
```
#values Assigning
x= df.iloc[:,0:13]
y= df.iloc[:,-1]
```

In [173…
```
from sklearn.model_selection import train_test_split
x_train,x_text,y_train,y_test = train_test_split(x,y,test_size=0.20,random_state=0)
```

2. Train Accuracy Score Prediction

In [174…
```
from sklearn.linear_model import LinearRegression
model= LinearRegression()
model.fit(x_train,y_train)
```

Out[174…
```
LinearRegression()
```

In [175…
```
y_pred = model.predict(x_train)
```

In [176…
```
print("Training Accuracy:",model.score (x_train,y_train)*100)
```

```
Training Accuracy: 77.30135569264233
```

3. Model Prediction

In [177…
```
print ("Testing Accuracy:" ,model.score(x_text,y_test)*100)
```

```
Testing Accuracy: 58.922238491825155
```

In [178…
```
from sklearn.metrics import mean_squared_error, r2_score
```

In [179…
```
print ("Model Accuracy:", r2_score(y,model.predict(x))*100)
```

```
Model Accuracy: 73.73440319905036
```

4. Model Visualization

In [180…

```
plt.scatter(y_train,y_pred)
plt.xlabel("Prices")
plt.ylabel ("Predicted prices")
plt.title("Prices vs Predicted prices")
plt.show()
```

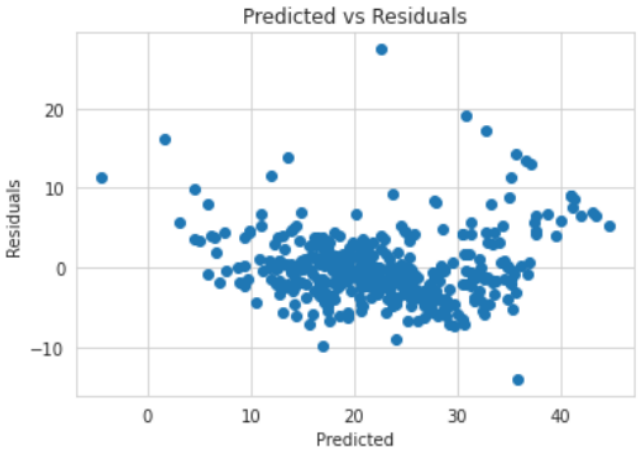
5. Residuals values

In [181…

```
#checking residuals
plt.scatter(y_pred,y_train-y_pred)
plt.ylabel("Residuals")
plt.xlabel ("Predicted")
plt.title("Predicted vs Residuals")
plt.show
```

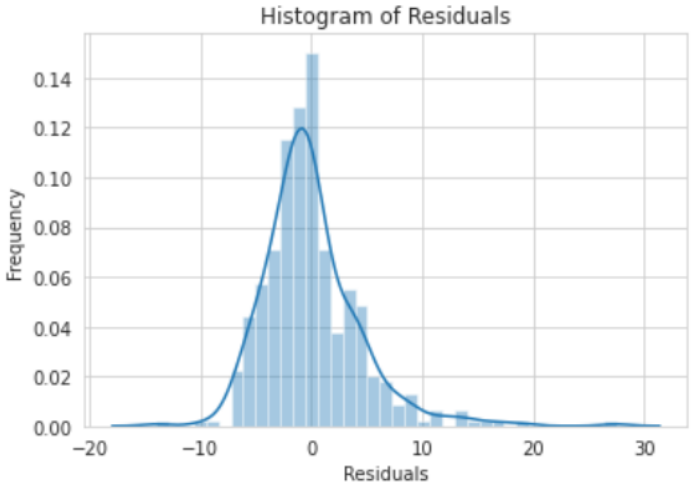
Out[181…

```
<function matplotlib.pyplot.show>
```



6. Checking Normality of Errors

In [182…
```
#checking normality of errors
sns.distplot(y_train-y_pred)
plt.title("Histogram of Residuals")
plt.xlabel("Residuals")
plt.ylabel("Frequency")
plt.show()
```



Random Forest Regressor

1. Values Assigning

In [183…
```
X = df.iloc[:,[-1,5,10,4,9]]
y = df.iloc[:,[-1]]
```

In [184…
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20,random_state=0)
```

2. Model Fitting

In [185…
```
from sklearn.ensemble import RandomForestRegressor
reg = RandomForestRegressor()
reg.fit(X_train,y_train)
```

Out[185… RandomForestRegressor()

3. Prediction Scores

In [186…
```
y_pred = reg.predict(X_train)
```

In [187…
```
print("Training Accuracy:",reg.score(X_train,y_train)*100)
```

Training Accuracy: 99.99199635164813

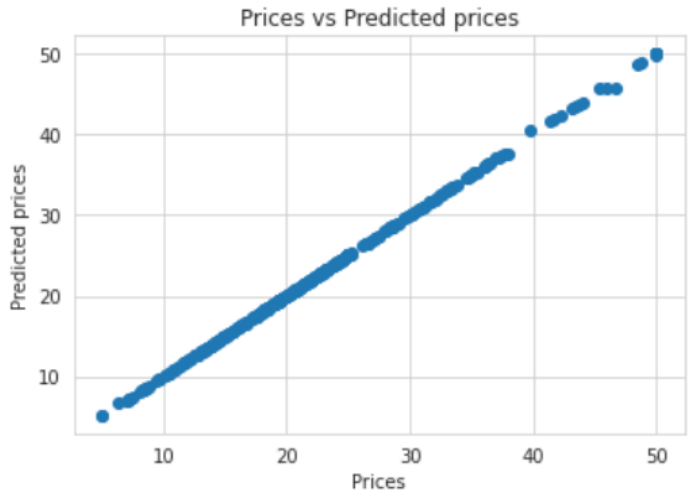
In [188…
```
print("Testing Accuracy:",reg.score(X_test,y_test)*100)
```

Testing Accuracy: 99.98085752134241

4. Visualization

In [189…

```
# Visualizing the differences between actual prices and predicted values
plt.scatter(y_train, y_pred)
plt.xlabel("Prices")
plt.ylabel("Predicted prices")
plt.title("Prices vs Predicted prices")
plt.show()
```



VII. PERFOMANCE ANALYSIS

Linear Regression

Model Score: 73.73% Accuracy

Training Accuracy: 77.30% Accuracy

Testing Accuracy: 58.92% Accuracy

Random Forest Regressor

Training Accuracy: 99.99% Accuracy

Testing Accuracy: 99.98% Accuracy

VIII. SOCIETAL IMPACT AND FUTURE SCOPE

Future work on this study could be divided into seven main areas to improve the result even further. This can be done by:

- The used pre-processing methods do help in the prediction accuracy. However, experimenting with different combinations of pre-processing methods to achieve better prediction accuracy.

- Make use of the available features and if they could be combined as binning features has shown that the data got improved.

- Training the datasets with different regression methods such as Elastic net regression that combines both L1 and L2 norms. In order to expand the comparison and check the performance.

- The correlation has shown the association in the local data. Thus, attempting to enhance the local data is required to make rich with features that vary and can provide a strong correlation relationship.

- For Random Forest Classification/Regression, besides the depth, we might need to examine further variations to optimize this algorithm, such as considering the splits of nodes, the requirements of leaf nodes, etc.

IX. CONCLUSION

It can be seen from the evaluation of three models that Random Forest Regressor performed better than Linear Regression. Hence, Random Forest Regressor proves to be more useful.

REFERENCES

[1] Annina S, Mahima SD, Ramesh B. An Overview of Machine Learning and its Applications. International Journal of Electrical Sciences & Engineering (IJESE). 2015 January; I(1): 22-24.

[2] David HW, William GM. No Free Lunch Theorems for Optimisation. IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. 1997 April; I(1): 67-82

[3] Peter JB, Bo L. Regularization in Statistics. Sociedad de Estad´ıstica e Investigaci´on Operativa. 2006; XV(2): 271-344.

[4] Couronné R, Probst P, Boulesteix A. Random forest versus logistic regression. a largescale benchmark experiment. BMC bioinformatics. 2018 December: p. 270. 12. Arnaiz-González Á,

[5] Díez-Pastor J, García-Osorio C, Rodríguez J. Random feature weights for regression trees. Progress in Artificial Intelligence. 2016 May: p. 91-103

[6] Louppe G. Understanding random forests: From theory to practice. arXiv preprint arXiv:1407.7502. 2014 July.

[7]Ben Ishak A. Variable selection using support vector regression and random forests: A comparative study. Intelligent Data Analysis. 2016 January: p. 83-104.

Appendix A:

These features used:

1. **CRIM** per capital crime rate by town

2. **ZN** proportion of residential land zoned for lots over 25,000 ft

3. **INDUS** proportion of non-retail business acres per town

4. **CHAS** Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

5. **NOX** nitric oxides concentration (parts per 10 million)

6. **RM** average number of rooms per dwelling

7. **AGE** proportion of owner-occupied units built prior to 1940

8. **DIS** weighted distances to five Boston employment centres

9. **RAD** index of accessibility to radial highways

10. **TAX** full-value property-tax rate per 10,000 USD

11. **PTRATIO** pupil-teacher ratio by town

12. **Black** $1000(Bk — 0.63)^2$ where Bk is the proportion of blacks by town

13. **LSTAT** % lower status of the population

14. **MEDV** Median value of owner-occupied homes in $1000s

Appendix B:

The dataset collected from Kaggle is imported in the Google Colab for the implementation of house price prediction model, and for the same algorithms are used such as Linear and random Forest Regression. The accuracy of each model is calculated and compared.

Appendix C:

The link to GitHub repository to access the source code: https://github.com/SadhanaDas/Boston-housing-Project.git