Boston House Price Prediction

Content

1. Importing libraries
2. Data pre-processing
3. Correlation and Heat map
4. Linear regression model
5. Random forest regressor model
6. Conclusion
7. Result

Importing Libraries

For our project to get working
Importing the pandas library for working with data frames and performing data manipulation and analysis tasks

- Importing the numpy library for working with arrays and performing numerical computations
- Importing the sklearn library for machine learning in Python, including tools and algorithms for tasks such as classification, regression, clustering, and dimensionality reduction
- Importing the matplotlib library for creating visualizations of data, such as plots and charts
- Importing the seaborn library for creating statistical graphics and visualizations
- Using the %matplotlib inline magic command to display plots and charts in the Jupyter notebook

Finally, the code is using the load_boston function from sklearn's datasets module to load the Boston house price dataset into the variable boston. This dataset contains information about the prices of houses in the Boston area, along with various features that may be related to the house prices, such as the number of rooms, the crime rate, and the distance to employment centers.

What is Data Pre Processing

Data Preprocessing is **a technique that is used to convert the raw data into a clean data set**.

Steps to be followed during data pre processing

- ❑ *Importing the dataset*
- ❑ *Initializing the dataframe*
- ❑ *See head of the dataset*
- ❑ *adding the feature names to the dataframe*
- ❑ *Adding target variable to dataframe*
- ❑ *Check the shape of dataframe*
- ❑ *Identifying the unique number of values in the dataset*
- ❑ *Check for missing values*
- ❑ *See rows with missing values*
- ❑ *Viewing the data statistics*
- ❑ *Finding out the correlation between the features*

❑ *Importing the dataset*

```
from sklearn.datasets import load_boston
boston = load_boston()
```

❑ *Initializing the dataframe*

❑ *See head of the dataset*

```
data.head()
```

❑ *adding the feature names to the dataframe*

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

- *Adding target variable to dataframe*

  PRICE
- *Check the shape of dataframe*
- *Identifying the unique number of values in the dataset*
- *Check for missing values*
- *See rows with missing values*
- *Viewing the data statistics*
- *Finding out the correlation*

  between the features

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
PRICE      0
dtype: int64
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS
count	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
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500

Correlation and heatmap

- In this code, the corr variable is being assigned the result of the corr method applied to the data dataframe. This calculates the pairwise correlations between the columns in the dataframe.
- The shape attribute of the corr variable is then printed to the console. This will output the number of rows and columns in the corr dataframe.
- The sort_values method is then applied to the 'PRICE' column of the corr dataframe, and the result is printed to the console. The ascending parameter is set to False, which sorts the values in descending order. This will print the correlations between the 'PRICE' column and the other columns in the dataframe, sorted by the magnitude of the correlation.

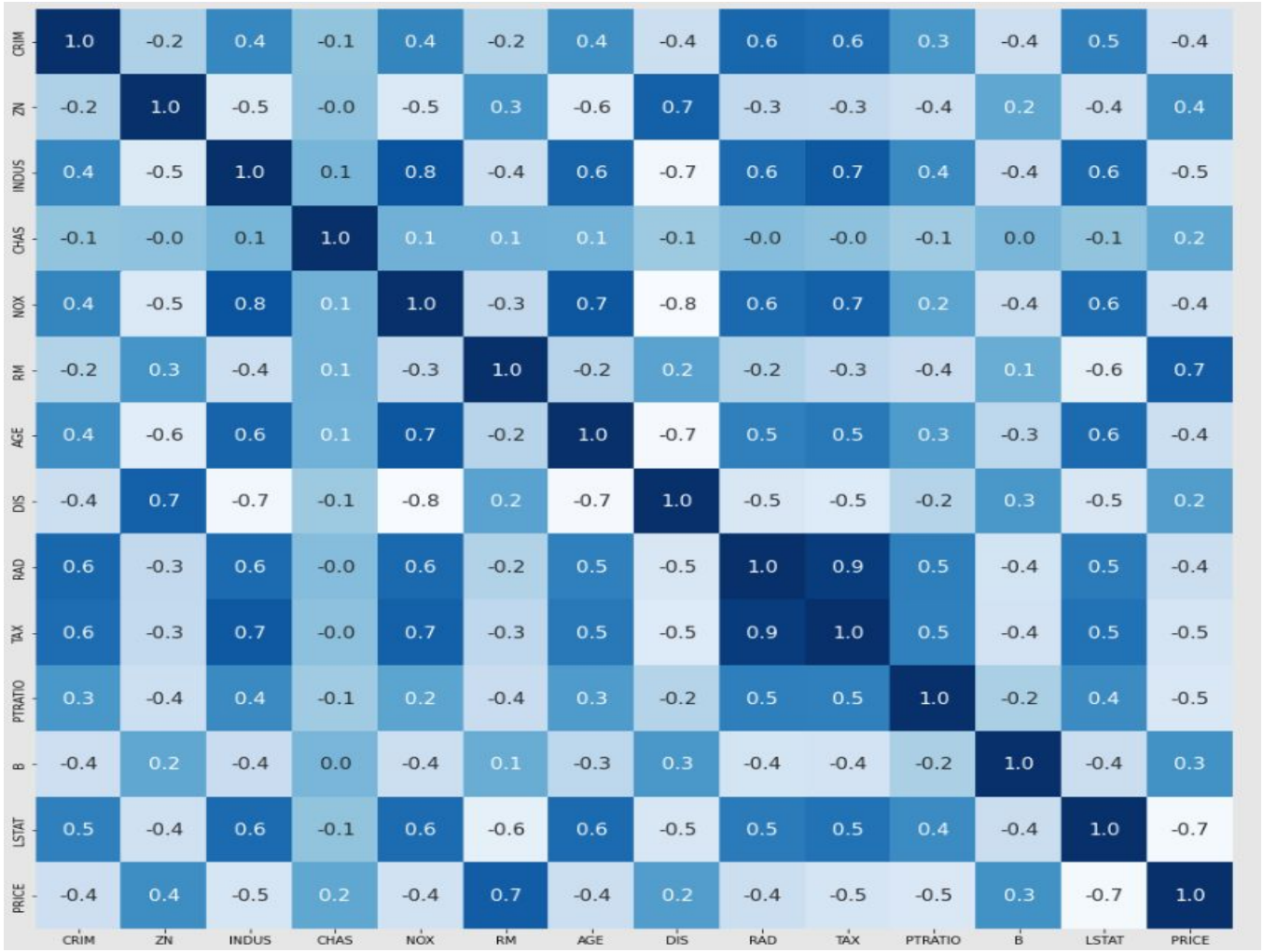
```
[ ]  corr = data.corr()
     corr.shape

     (14, 14)

     print(corr['PRICE'].sort_values(ascending= False))

     PRICE      1.000000
     RM         0.695360
     ZN         0.360445
     B          0.333461
     DIS        0.249929
     CHAS       0.175260
     AGE       -0.376955
     RAD       -0.381626
     CRIM      -0.388305
     NOX       -0.427321
     TAX       -0.468536
     INDUS     -0.483725
     PTRATIO   -0.507787
     LSTAT     -0.737663
     Name: PRICE, dtype: float64
```

This was how we represented the correlation.
(*a heat map*).



Linear regression Model

- A linear regression model is used to predict a dependent variable based on one or more independent variables
- The model fits a linear equation to the data that best approximates the relationship between the variables
- The equation for the model is y = b0 + b1*x, where y is the dependent variable, x is the independent variable, b0 is the intercept, and b1 is the slope
- The model is used to make predictions about the dependent variable based on new values of the independent variable
- To build the model, a line (the "regression line") is fit to the data that minimizes the distance between the points and the line (the "residuals")

Prediction From Train data

```
[ ]  # Import library for Linear Regression
     from sklearn.linear_model import LinearRegression

     # Create a Linear regressor
     lm = LinearRegression()

     # Train the model using the training sets
     lm.fit(X_train, y_train)
```

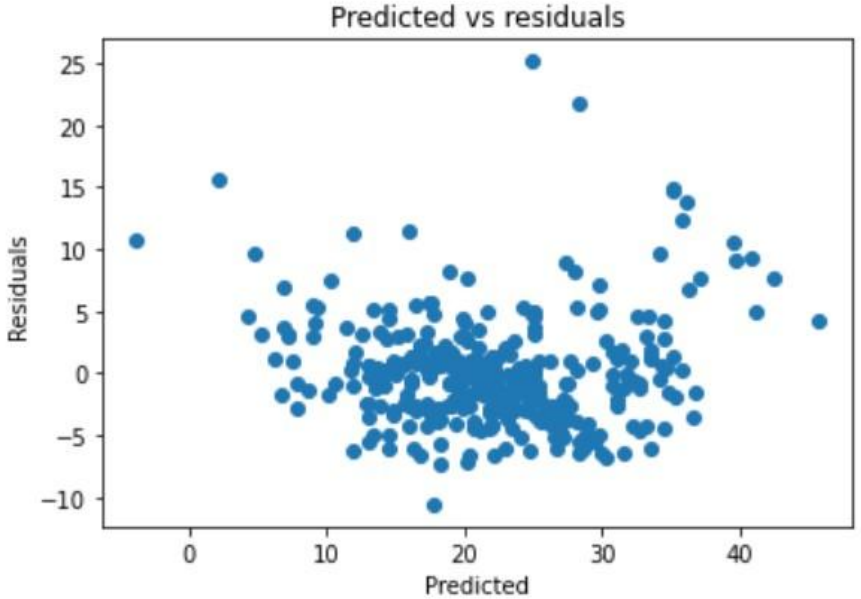
```
LinearRegression()
```

```
[ ]  # Value of y intercept
     lm.intercept_
```

```
36.35704137659508
```

```
R^2: 0.7465991966746854
Adjusted R^2: 0.736910342429894
MAE: 3.0898610949711305
MSE: 19.073688703469035
RMSE: 4.367343437774162
```

Visualising the difference between Actual and predicted Values



Predictions of Test data

```
R^2: 0.7121818377409193
Adjusted R^2: 0.6850685326005711
MAE: 3.859005592370744
MSE: 30.053993307124 16
RMSE: 5.482152251362977
```

Disadvantages of Linear regression model

There are several problems that can arise when using linear regression models:

1. Linear regression assumes that the relationship between the input features and the output is linear, which may not always be the case. This can lead to poor model performance if the data has a non-linear relationship.
2. Linear regression is sensitive to outliers, as they can have a significant impact on the model's coefficients. This can lead to a poor fit and inaccurate predictions.
3. Linear regression models can have high variance, meaning that small changes in the training data can lead to large changes in the model's coefficients. This can make the model less stable and harder to interpret.

Random forest models can help to overcome some of these problems by using an ensemble of decision trees rather than a single linear model. Decision trees can model non-linear relationships and are less sensitive to outliers, and the use of an ensemble can help to reduce variance and improve stability.

Random Forest Regression model

- A random forest regressor is a machine learning model used for regression tasks, which involve predicting a continuous output value given an input.
- It is an ensemble model made up of a collection of decision trees, and the final prediction is made by averaging or voting the predictions of the individual trees.
- Each decision tree in a random forest model is trained on a randomly selected subset of the data and uses randomly chosen features at each step.
- This helps to reduce overfitting and improve the generalizability of the model.
- To make a prediction, the model passes the input data through each decision tree and the final prediction is the average or majority vote of the individual tree predictions.
- Random forest models are popular because they are easy to train and tune, perform well on a wide range of tasks, and can handle missing values and large numbers of categorical features.

Working

- In this code, a random forest regressor model is being trained on the input data X_train and the target variable y_train. The fit method is used to train the model on the training data.
- Once the model is trained, it is used to make predictions on the training data itself (X_train) using the predict method. The predicted values are stored in the variable y_pred.
- Finally, the model's performance is evaluated using a number of different metrics:r2_score, mean_absolute_error, Mean_squared_error, root_mean_squared_error.
- These metrics are all calculated using the y_train true values and the y_pred predicted values, and the results are printed to the console.

```
[ ] # Import Random Forest Regressor
    from sklearn.ensemble import RandomForestRegressor

    # Create a Random Forest Regressor
    reg = RandomForestRegressor()

    # Train the model using the training sets
    reg.fit(X_train, y_train)
```

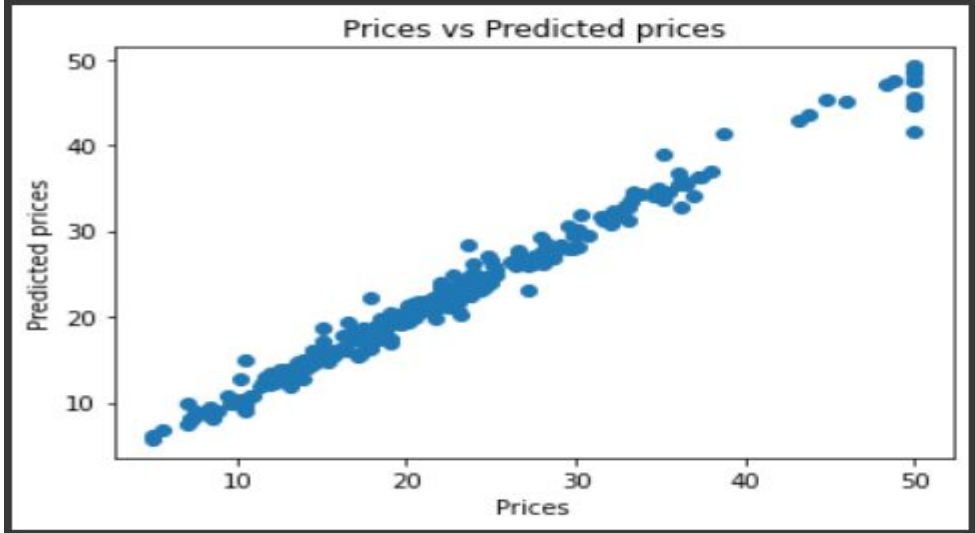
```
RandomForestRegressor()
```

```
[ ] # Model prediction on train data
    y_pred = reg.predict(X_train)
    # Model Evaluation
    print('R^2:',metrics.r2_score(y_train, y_pred))
    print('Adjusted R^2:',1 - (1-metrics.r2_score(y_train, y_pred))*(len(y_train)-1)/(len(y_train)-X_train.shape[1]-1))
    print('MAE:',metrics.mean_absolute_error(y_train, y_pred))
    print('MSE:',metrics.mean_squared_error(y_train, y_pred))
    print('RMSE:',np.sqrt(metrics.mean_squared_error(y_train, y_pred)))
```

```
R^2: 0.9790656090594555
Adjusted R^2: 0.9782651764646699
MAE: 0.8364661016949151
MSE: 1.5757489745762716
RMSE: 1.2552884029482116
```

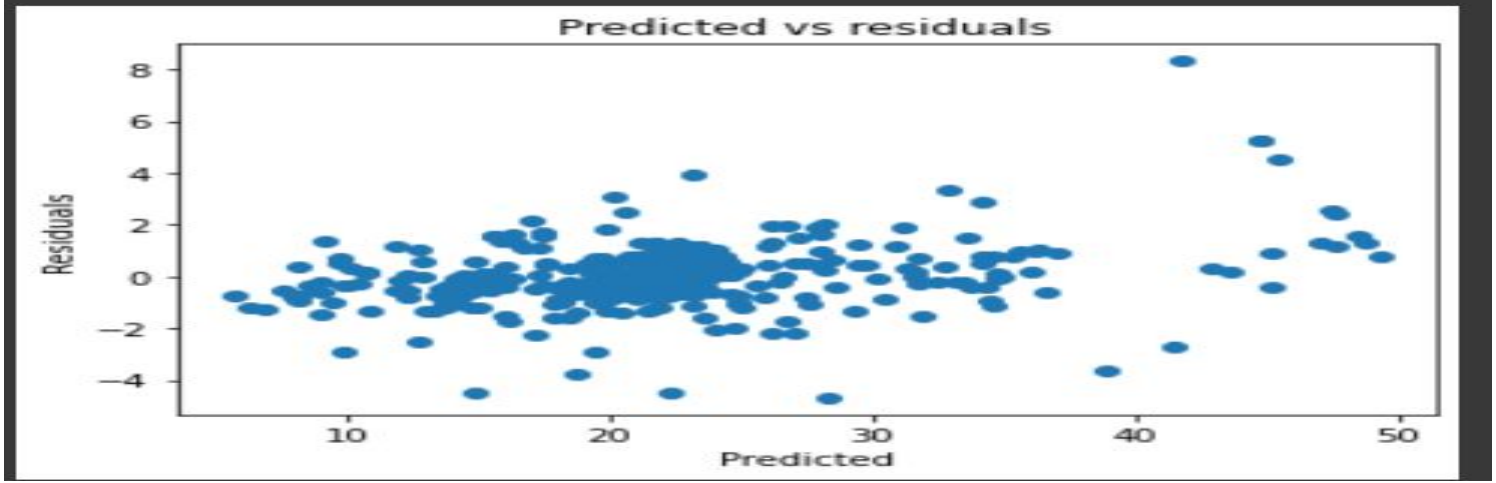
- This code is creating a scatter plot that visualizes the relationship between the true values of the target variable (y_train) and the predicted values (y_pred) produced by the model. T
- he x-axis of the plot represents the true values, and the y-axis represents the predicted values.

```
# 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()
```



- This code is creating a scatter plot that visualizes the residuals of the model, which are the differences between the true values of the target variable (y_train) and the predicted values (y_pred).
- The x-axis of the plot represents the predicted values, and the y-axis represents the residuals.

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



- In this code, the trained random forest regressor model is being used to make predictions on the test data (X_test). The predicted values are stored in the variable y_test_pred.
- Once the predictions have been made, the model's performance is evaluated using a number of different

```
# Predicting Test data with the model
y_test_pred = reg.predict(X_test)
# Model Evaluation
acc_rf = metrics.r2_score(y_test, y_test_pred)
print('R^2:', acc_rf)
print('Adjusted R^2:',1 - (1-metrics.r2_score(y_test, y_test_pred))*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1))
print('MAE:',metrics.mean_absolute_error(y_test, y_test_pred))
print('MSE:',metrics.mean_squared_error(y_test, y_test_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
R^2: 0.8347586887519151
Adjusted R^2: 0.819192478272023
MAE: 2.496124999999999
MSE: 17.25450966447369
RMSE: 4.153854795785921
```

Conclusion

- How to preprocess and clean data to prepare it for use in a machine learning model.
- How to select and use appropriate machine learning algorithms and evaluation metrics for a given task.
- How to train, fine-tune, and evaluate a machine learning model on a dataset.
- How to interpret the results of a machine learning model and use them to make decisions or predictions.
- How to communicate the results of a machine learning project effectively to others.

References

1. https://towardsdatascience.com/random-forest-regression-5f605132d19d
2. https://www.javatpoint.com/linear-regression-in-machine-learning
3. https://www.javatpoint.com/data-preprocessing-machine-learning
4. https://medium.com/analytics-vidhya/linear-regression-and-random-forest-33d4297a186a

Thank You!!