| |
|---|
| Experiment No. 1 |
| Analyze the Boston Housing dataset and Apply appropriate Regression Technique |
| Date of Performance: |
| Date of Submission: |

Aim: Analyze the Boston Housing dataset and Apply appropriate Regression Technique.

Objective: Ability to perform various feature engineering tasks, apply linear regression on the given dataset and minimize the error.

Theory:

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.
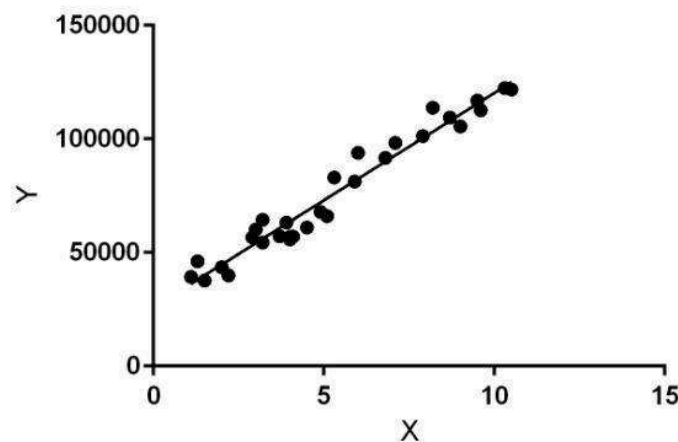
Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Dataset:

The Boston Housing Dataset

The Boston Housing Dataset is derived from information collected by the U.S. Census Service concerning housing in the area of Boston MA. The following describes the dataset columns:

CRIM - per capita crime rate by town

ZN - proportion of residential land zoned for lots over 25,000 sq.ft.

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

CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)

NOX - nitric oxides concentration (parts per 10 million)

RM - average number of rooms per dwelling

AGE - proportion of owner-occupied units built prior to 1940

DIS - weighted distances to five Boston employment centres

RAD - index of accessibility to radial highways

TAX - full-value property-tax rate per $10,000

PTRATIO - pupil-teacher ratio by town

B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town

LSTAT - % lower status of the population

MEDV - Median value of owner-occupied homes in $1000's

Code & Output:

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

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

CSL701: Machine Learning Lab

```python
from scipy import stats
```

```python
Boston=pd.read_csv('Boston.csv')
```

```python
Boston.head()
```

	ID	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
0	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
1	6	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21
2	8	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.90	19.15
3	9	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	386.63	29.93
4	10	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.10

```python
Boston.info()
```

```python
Boston.describe()
```

```
<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  b        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
```

CSL701: Machine Learning Lab

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptra...
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.0000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.4555
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.1649
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.6000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.4000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.0500
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.2000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.0000

```python
missing_values = Boston.isna().sum()
print(missing_values)
```

```
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
medv        0
dtype: int64
```

```python
na_columns = ['crim', 'zn', 'indus', 'chas', 'age', 'lstat']
Boston[na_columns] = Boston[na_columns].fillna(Boston.mean())
print(Boston)
```

CSL701: Machine Learning Lab

```
        crim    zn  indus  chas    nox      rm   age      dis  rad  tax  \
0    0.00632  18.0   2.31     0  0.538   6.575  65.2   4.0900    1  296
1    0.02731   0.0   7.07     0  0.469   6.421  78.9   4.9671    2  242
2    0.02729   0.0   7.07     0  0.469   7.185  61.1   4.9671    2  242
3    0.03237   0.0   2.18     0  0.458   6.998  45.8   6.0622    3  222
4    0.06905   0.0   2.18     0  0.458   7.147  54.2   6.0622    3  222
..       ...   ...    ...   ...    ...     ...   ...      ...  ...  ...
501  0.06263   0.0  11.93     0  0.573   6.593  69.1   2.4786    1  273
502  0.04527   0.0  11.93     0  0.573   6.120  76.7   2.2875    1  273
503  0.06076   0.0  11.93     0  0.573   6.976  91.0   2.1675    1  273
504  0.10959   0.0  11.93     0  0.573   6.794  89.3   2.3889    1  273
505  0.04741   0.0  11.93     0  0.573   6.030  80.8   2.5050    1  273

     ptratio       b  lstat  medv
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       17.8  392.83   4.03  34.7
3       18.7  394.63   2.94  33.4
4       18.7  396.90   5.33  36.2
..       ...     ...    ...   ...
501     21.0  391.99   9.67  22.4
502     21.0  396.90   9.08  20.6
503     21.0  396.90   5.64  23.9
504     21.0  393.45   6.48  22.0
505     21.0  396.90   7.88  11.9

[506 rows x 14 columns]
```

```python
target = Boston['medv']
print(target)
```

```
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: medv, Length: 506, dtype: float64
```

```python
X = Boston.drop(["medv"], axis=1)
y = Boston["medv"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)

model = LinearRegression()
```
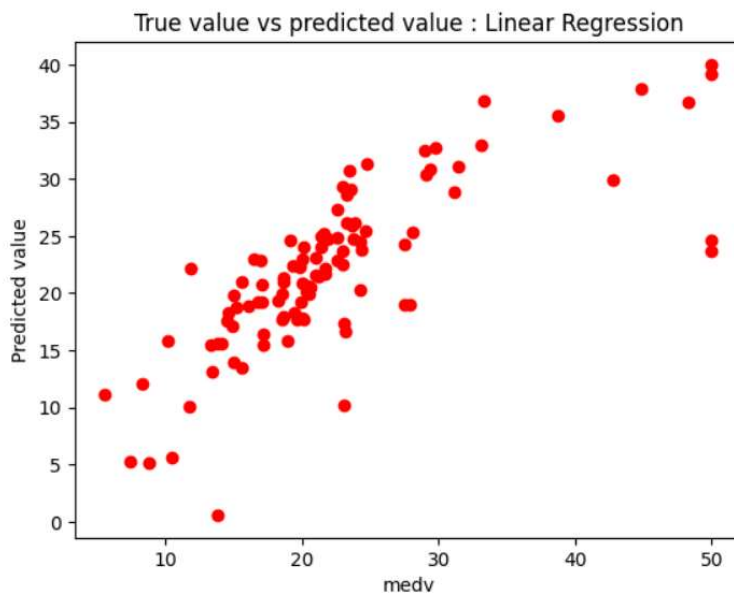
CSL701: Machine Learning Lab

```
model.fit(X_train ,y_train)

LinearRegression()

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)

plt.scatter(y_test, y_pred, c = 'red')
plt.xlabel("medv")
plt.ylabel("Predicted value")
plt.title("True value vs predicted value : Linear Regression")
plt.show()
```
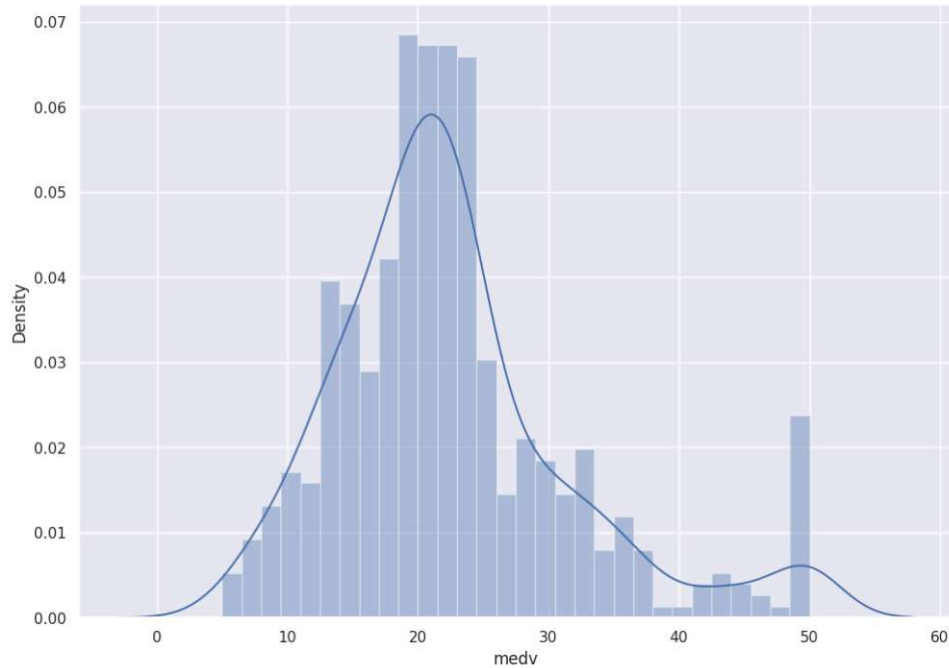


```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.distplot(boston['medv'], bins=30)
plt.show()
```

CSL701: Machine Learning Lab

```
correlation_matrix = boston.corr().round(2)
# annot = True to print the values inside the square
sns.heatmap(data=correlation_matrix, annot=True)
```
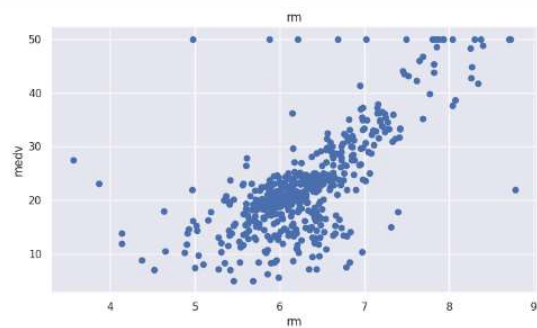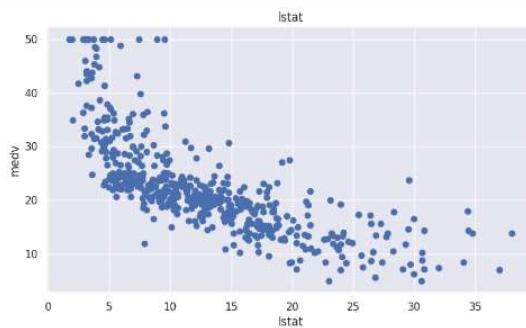

CSL701: Machine Learning Lab

```python
plt.figure(figsize=(20, 5))

features = ['lstat', 'rm']
target = boston['medv']

for i, col in enumerate(features):
    plt.subplot(1, len(features) , i+1)
    x = boston[col]
    y = target
    plt.scatter(x, y, marker='o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('medv')
```



```python
X   =   pd.DataFrame(np.c_[boston['lstat'],   boston['rm']],   columns   =
['lstat','rm'])
Y = boston['medv']

from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,
random_state=5)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)

(404, 2)
(102, 2)
```

CSL701: Machine Learning Lab

```
(404,)
(102,)

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)


LinearRegression()


# model evaluation for training set
from sklearn.metrics import r2_score
y_train_predict = lin_model.predict(X_train)
rmse = (np.sqrt(mean_squared_error(Y_train, y_train_predict)))
r2 = r2_score(Y_train, y_train_predict)


print("The model performance for training set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print("\n")


# model evaluation for testing set
y_test_predict = lin_model.predict(X_test)
rmse = (np.sqrt(mean_squared_error(Y_test, y_test_predict)))
r2 = r2_score(Y_test, y_test_predict)


print("The model performance for testing set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
```

CSL701: Machine Learning Lab

```
print('R2 score is {}'.format(r2))
```

```
The model performance for training set
--------------------------------------
RMSE is 5.637129335071195
R2 score is 0.6300745149331701


The model performance for testing set
--------------------------------------
RMSE is 5.137400784702911
R2 score is 0.6628996975186952
```

Conclusion:

The Mean Squared Error (MSE) quantifies the average squared difference between the actual and predicted values. A lower MSE signifies that the model's predictions are closer to the actual values. After calculating the MSE, it should be interpreted in relation to the range and distribution of the target variable (medv). If the MSE is small relative to the variance of medv, it indicates that the model fits the data well. Conversely, a large MSE suggests that the model may not be effectively capturing the underlying patterns in the data.

CSL701: Machine Learning Lab