DSBDA Practcial4 Boston

May 12, 2023

[1]:
```
#Import libraries and create alias for Pandas, Numpy and Matplotlibimport numpy␣
 ↪as np
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
#Import the Boston Housing dataset
from sklearn.datasets import load_boston
```

[5]:
```
boston = load_boston()
```

[6]:
```
#Initialize the data frame
data = pd.DataFrame(boston.data)
```

[7]:
```
#Add the feature names to the dataframe
data.columns = boston.feature_names
data.head()
```

[7]:

	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

[8]:
```
#Adding target variable to dataframe
data['PRICE'] = boston.target
```

[9]:
```
# Perform Data Preprocessing( Check for missing values)
data.isnull().sum()
```

[9]:
```
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
```

[10]:
```
#Split dependent variable and independent variables
x = data.drop(['PRICE'], axis = 1)
y = data['PRICE']
```

[11]:
```
#splitting data to training and testing dataset.
from sklearn.model_selection import train_test_split
```

[12]:
```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.
↪2,random_state = 0)
```

[13]:
```
#Use linear regression( Train the Machine ) to Create Model
import sklearn
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
model=lm.fit(xtrain, ytrain)
```

[14]:
```
#Predict the y_pred for all values of train_x and test_x
ytrain_pred = lm.predict(xtrain)
ytest_pred = lm.predict(xtest)
```

[15]:
```
#Evaluate the performance of Model for train_y and test_y
df=pd.DataFrame(ytrain_pred,ytrain)
df=pd.DataFrame(ytest_pred,ytest)
```

[16]:
```
#Calculate Mean Square Paper for train_y and test_y
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(ytest, ytest_pred)
print(mse)
```

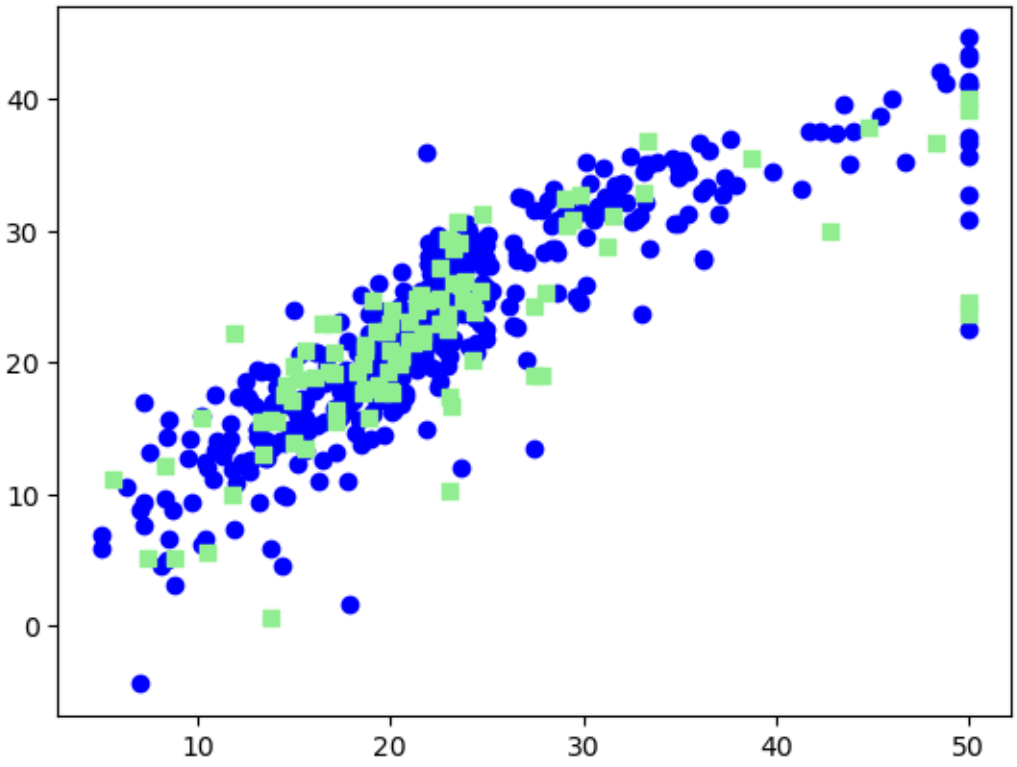
```
33.44897999767643
```

[17]:
```
mse = mean_squared_error(ytrain_pred,ytrain)
print(mse)
```

```
19.32647020358573
```

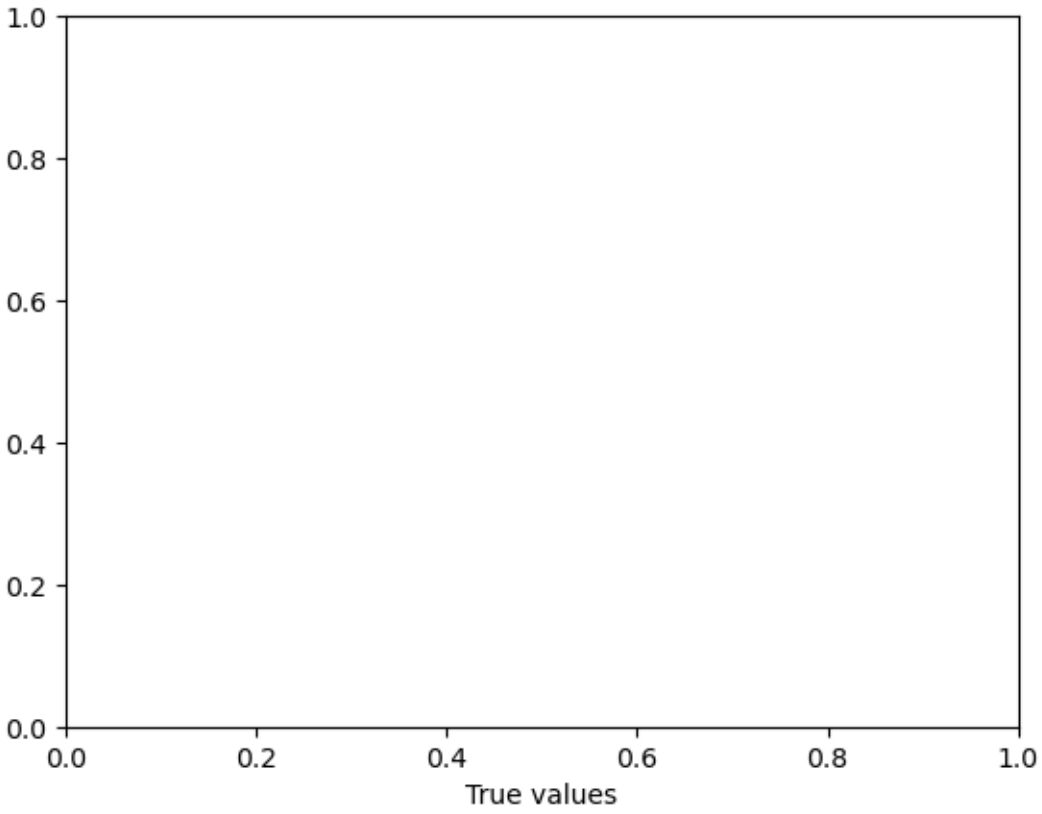
[18]:
```
#Plotting the linear regression model
plt.scatter(ytrain ,ytrain_pred,c='blue',marker='o',label='Training data')
plt.scatter(ytest,ytest_pred ,c='lightgreen',marker='s',label='Test data')
```

[18]: <matplotlib.collections.PathCollection at 0x2290a6ff760>



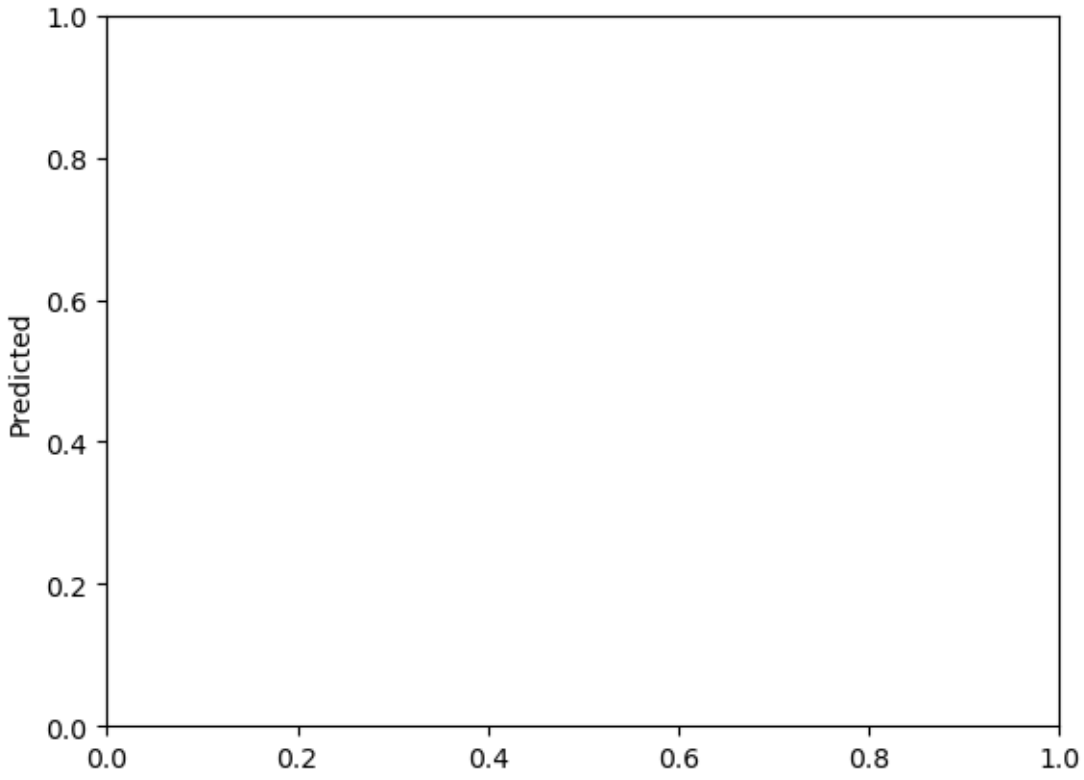
[19]:
```
plt.xlabel('True values')
```

[19]: Text(0.5, 0, 'True values')



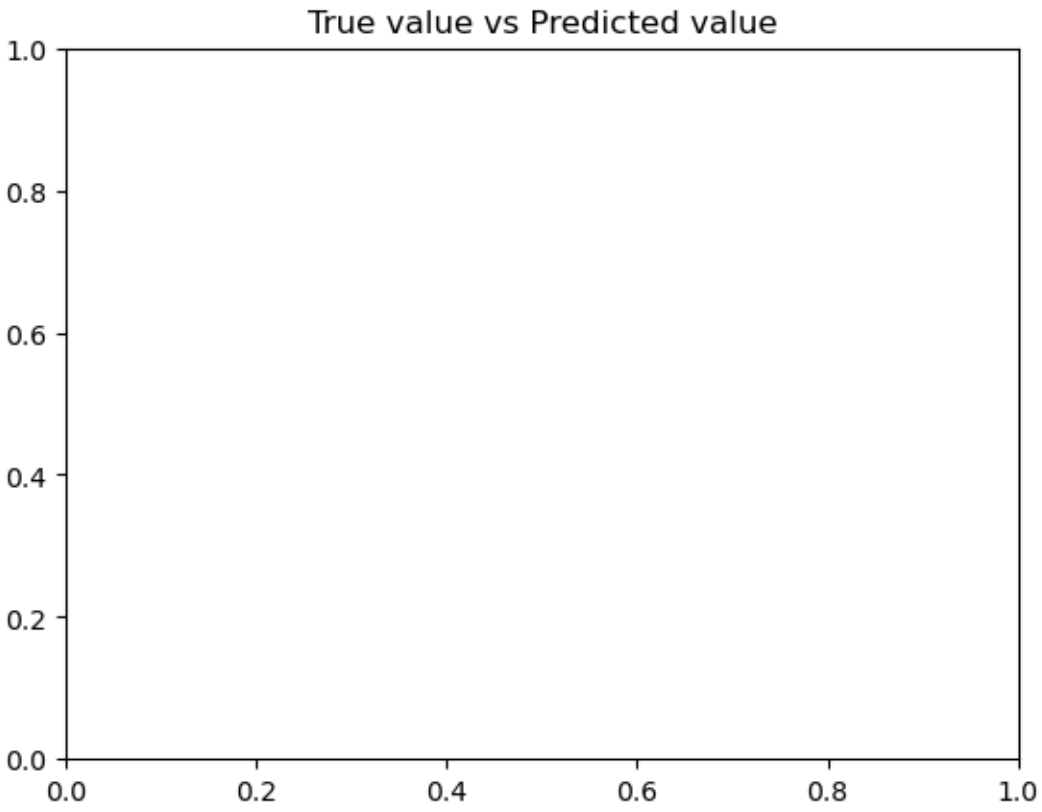
[20]:
```
plt.ylabel('Predicted')
```

[20]:
```
Text(0, 0.5, 'Predicted')
```



[21]:
```
plt.title("True value vs Predicted value")
```

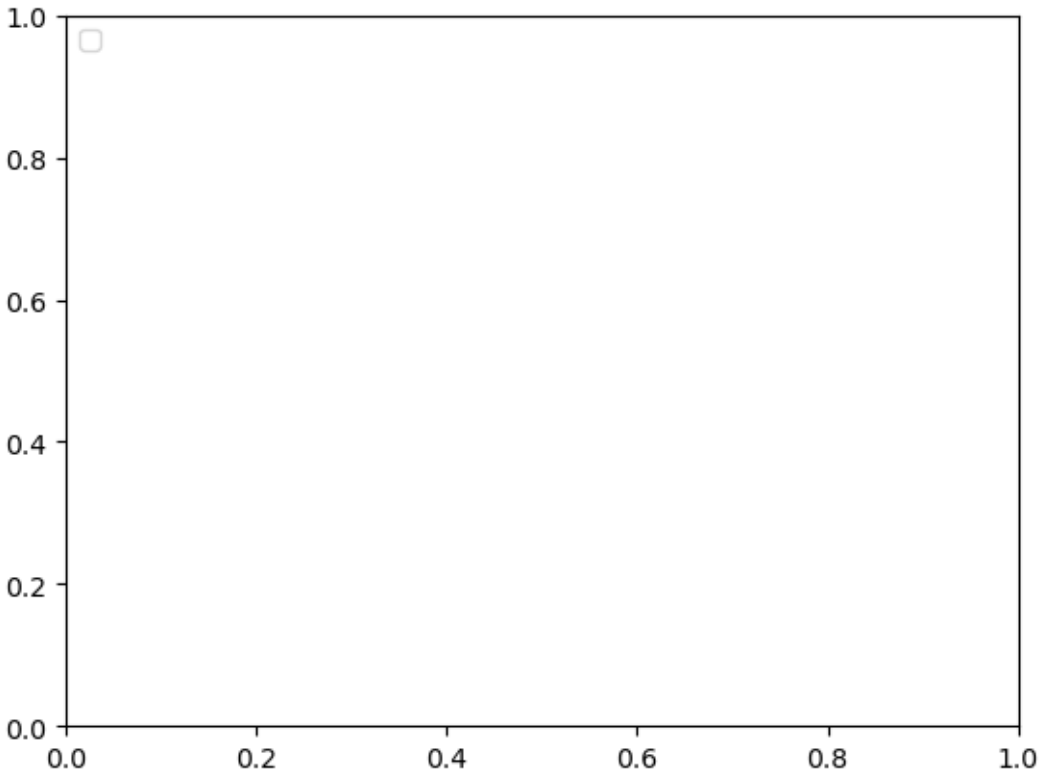
[21]: Text(0.5, 1.0, 'True value vs Predicted value')



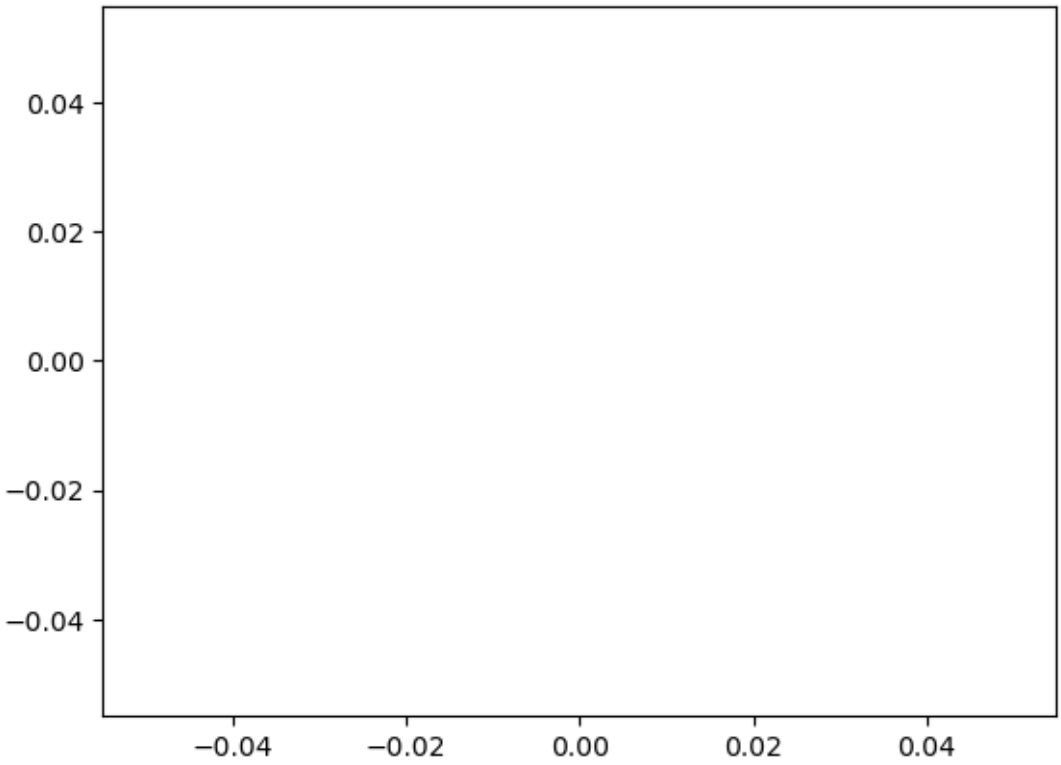
[22]:
```
plt.legend(loc= 'upper left')
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```

[22]: <matplotlib.legend.Legend at 0x2290a8197c0>



[23]:
```
#plt.hlines(y=0,xmin=0,xmax=50)
plt.plot()
plt.show()
```


0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04

[ ]:

[ ]: