**PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE**

**ACADEMIC YEAR: 2021-22**

## **DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT**

**CLASS: T.E. SEMESTER: I**

**SUBJECT: DSBDAL**

|  |  |
| --- | --- |
| **ASSINGMENT NO.** | 4 |
| **TITLE** | Data Analytics I |
| **PROBLEM STATEMENT /DEFINITION** | Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (htt[ps://www.kaggle.com/c/boston](http://www.kaggle.com/c/boston-housing))-[housing).](http://www.kaggle.com/c/boston-housing)) The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset.  The objective is to predict the value of prices of the house using the given features. |
| **OBJECTIVE** | To create a linear regression model using a sample dataset |
| **OUTCOME** | After completion of this assignment the students will be able to:  create a linear regression model using a sample dataset |
| **S/W PACKAGES AND**  **HARDWARE APPARATUS USED** | Jupyter notebook, Pandas libraries, Windows/ Linux Operating System, I5 machines/ Laptops |
| **REFERENCES** | [www.kaggle.com](http://www.kaggle.com)  <https://www.geeksforgeeks.org/python-linear-regression-using-sklearn/> |
| **STEPS** | **Importing all the required libraries****Reading the dataset****Exploring the data scatter****Data cleaning****Training our model****Exploring our results****Working with a smaller dataset** |
| **INSTRUCTIONS FOR**  **WRITING JOURNAL** | 1. Date  2. Assignment no.  3. Problem definition  4. Learning objective  5. Learning Outcome  6. Concepts related Theory  7. Algorithm  8. Test cases  10. Conclusion/Analysis |

**Prerequisites:** Basic knowledge of DBMS

**Concepts related Theory and design of assignment using a sample dataset available here:**

<https://www.kaggle.com/sohier/calcofi?select=bottle.csv>

(Students are expected to go through the steps of linear regression applied to this sample dataset and then apply the same steps to the dataset given in the problem statement)

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 being used.

**Step 1: Importing all the required libraries**

**import** numpy as np

**import** pandas as pd

**import** seaborn as sns

**import** matplotlib.pyplot as plt

**from** sklearn **import** preprocessing, svm

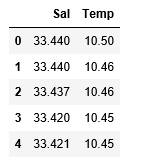
**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.linear\_model **import** LinearRegression

**Step 2: Reading the dataset**

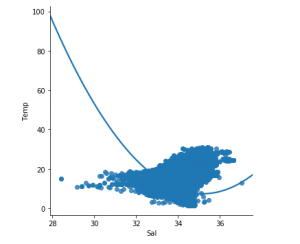
You can download the dataset [here.](https://www.kaggle.com/sohier/calcofi)

|  |
| --- |
| cd C:\Users\Dev\Desktop\Kaggle\Salinity    # Changing the file read location to the location of the dataset  df **=** pd.read\_csv('bottle.csv')  df\_binary **=** df[['Salnty', 'T\_degC']]    # Taking only the selected two attributes from the dataset  df\_binary.columns **=** ['Sal', 'Temp']    # Renaming the columns for easier writing of the code  df\_binary.head()    # Displaying only the 1st  rows along with the column names |



**Step 3: Exploring the data scatter**

|  |
| --- |
| sns.lmplot(x **=**"Sal", y **=**"Temp", data **=** df\_binary, order **=** 2, ci **=** None)    # Plotting the data scatter |



**Step 4: Data cleaning**

|  |
| --- |
| # Eliminating NaN or missing input numbers  df\_binary.fillna(method **=**'ffill', inplace **=** True) |

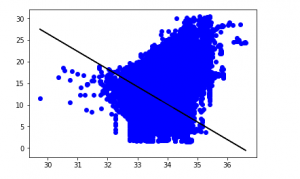
**Step 5: Training our model**

|  |
| --- |
| X **=** np.array(df\_binary['Sal']).reshape(**-**1, 1)  y **=** np.array(df\_binary['Temp']).reshape(**-**1, 1)    # Separating the data into independent and dependent variables  # Converting each dataframe into a numpy array  # since each dataframe contains only one column  df\_binary.dropna(inplace **=** True)    # Dropping any rows with Nan values  X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size **=** 0.25)    # Splitting the data into training and testing data  regr **=** LinearRegression()    regr.fit(X\_train, y\_train)  print(regr.score(X\_test, y\_test)) |

Lightbox

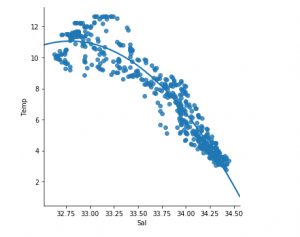
**Step 6: Exploring our results**

|  |
| --- |
| y\_pred **=** regr.predict(X\_test)  plt.scatter(X\_test, y\_test, color **=**'b')  plt.plot(X\_test, y\_pred, color **=**'k')    plt.show()  # Data scatter of predicted value |



**Step 7: Working with a smaller dataset**

|  |
| --- |
| df\_binary500 **=** df\_binary[:][:500]    # Selecting the 1st 500 rows of the data  sns.lmplot(x **=**"Sal", y **=**"Temp", data **=** df\_binary500,                                 order **=** 2, ci **=** None) |



We can already see that the first 500 rows follow a linear model. Continuing with the same steps as before.

|  |
| --- |
| df\_binary500.fillna(method **=**'ffill', inplace **=** True)    X **=** np.array(df\_binary500['Sal']).reshape(**-**1, 1)  y **=** np.array(df\_binary500['Temp']).reshape(**-**1, 1)    df\_binary500.dropna(inplace **=** True)  X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size **=** 0.25)    regr **=** LinearRegression()  regr.fit(X\_train, y\_train)  print(regr.score(X\_test, y\_test))  Lightbox |

y\_pred **=** regr.predict(X\_test)

plt.scatter(X\_test, y\_test, color **=**'b')

plt.plot(X\_test, y\_pred, color **=**'k')

plt.show()

