School of Computer Science Engineering and Technology
Assignment-05

Course- B.Tech	**Type-** Core
Course Code-	**Course Name-** Statistical Machine Learning
Year- 2023-2024	**Semester-** odd
Date– 14-08-2023	**Batch-** AIML-B, D

1 Implement Kernel Density Estimation for Feature Space

Part 1 – Import the required Python, Pandas, Matplotlib, Seaborn packages

Kernel Density Estimation plots] [1]. These KDE plots replace every single observation with a Gaussian (Normal) distribution centered around that value. KDE is a technique that let's you create a smooth curve given a set of data.

Step 1 –The kernel density estimator for given distribution

:

$$\hat{f}(x;h) := \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right). \tag{1}$$

The KDE algorithm takes a parameter, bandwidth, that affects how "smooth" the resulting curve is.

1. Create a dataset using a random `random.rand(100)`
2. Create a rugplot for the dataset
3. Set-up x-axis for the plot with $x_{min} - 2$, x_{max}+2
4. Distribute among 100 equal points from x_min to x_max

Step 2 –Bandwidth calculation for estimating Kernel Density using the formula

:

$$h := \left(\frac{4\sigma^{-0.5}}{3n}\right)^{0.2} \approx 1.06\hat{\sigma}n^{0.2} \tag{2}$$

[1] `http://en.wikipedia.org/wiki/Kernel\_density\_estimation\#Practical_estimation_of_the_bandwidth`

1. Create a empty kernel list
2. Plot the basis function
3. Create a kernel for each point and append to list
4. Scale for plotting using the kernel

Step 3 –Generate the Kernel Density Plot Estimate by sum of basis function

1. Plot the sum of basis function
2. Plot using seaborn rugplot `rugplot`
3. Generate the kernel density basis function plot.

2 Implement the L1 and L2 Regularization using Boston Housing Dataset

Part 1 – Import the required Python, Pandas, Matplotlib, Seaborn packages

Step 1 – Load the Boston Housing Dataset

The two most commonly used regularizors are as follows:

1. **L1 regularization: Least Absolute Shrinkage and Selection Operator** This adds a term to the scoring function that is proportional to the sum of all absolute weight values. In other words, it is based on the L1 norm of the weight vector (also known as the rectilinear distance, snake distance, or Manhattan distance). The resulting algorithm is also known as **LASSO Regression**.
2. **L2 regularization:** This adds a term to the scoring function that is proportional to the sum of all squared weight values. In other words, it is based on the L2 norm of the weight vector (also known as the Euclidean distance). Since the L2 norm involves a squaring operation, it punishes strong outliers in the weight vector much harder than the L1 norm. The resulting algorithm is also known as **Ridge Regression**.

Step 2 –Train the Model

1. Implement the linear Ridge regression using `linear_model.Ridge()`
2. Fit the ridge regression
3. Compute the mean squared error
4. Generate the Ridge Regression score for the dataset using `ridgereg.score()`

Step 3 – Test the Model

1. Predict the Ridge Regression using `ridgereg.predict()`
2. Compute the Mean Squared Error
3. Generate a plot between *Predicted and Ground Truth Values*
4. Compute the R^2 and Mean Square Error using Predicted and Ground Truth Values