Name: PADAM SINGH

Registration no: ADIT22AP00545

Date: 09 Nov. 23

Module: CORE MODULE 5

Course Name: IBM-ADIT

NSTI Name: NSTI MUMBAI

**Practical Aim:** Calculate the count, average and standard deviation from the following salary data in MS Excel using appropriate excel function.

Requirement/Tools:

Hardware: Computer/Laptop

Software: MS Excel

**Solution:**

**Follow these steps:**

1. Go to the Search bar, type excel and click on open.

|  |
| --- |
|  |

1. Create a Salary sheet which have First name, Last name and Salary.

|  |
| --- |
|  |

1. We have Salary in Doller. Now convert the number into the Doller currency. Go to the number and select the currency in English (United Kingdom).

|  |
| --- |
|  |

1. Now Calculate the **count**, **average** and **standard deviation.**

**Output:**

**Count**

|  |
| --- |
|  |

**Average**

|  |
| --- |
|  |

**Standard Deviation**

|  |
| --- |
|  |

**Practical Aim:** Implement KNN algorithm using sklearn on the given IRIS dataset and display the accuracy and classification report.

Download the dataset form the Kaggle.

Requirement/Tools:

Hardware: Computer/Laptop

Software: Anaconda, Jupyter Notebook and google collab

**Solution:**

**Follow these steps:**

1. Open the browser, search Kaggle and search iris dataset.

|  |
| --- |
|  |

1. Now go to the search bar and type jupyter notebook and create a new folder I have a folder which name is Core Module 5 FinalExam.
2. Now open this folder and create a new file which name is Question 1 iris dataset.

|  |
| --- |
|  |

1. We need to import the libraries.

|  |
| --- |
| # import python libraries  import numpy as np  import pandas as pd  import seaborn as sns |

1. Read the csv file with the help of pandas.

|  |
| --- |
| # import and read the iris.csv file  df=pd.read\_csv('iris.csv')  df |

**Output:**

|  |
| --- |
|  |

1. Finding the head.

|  |
| --- |
| df.head() |

**Output:**

|  |
| --- |
|  |

1. Finding the tail.

|  |
| --- |
| df.tail() |

**Output:**

|  |
| --- |
|  |

1. Finding the shape. It is showing how many rows and columns into the dataset.

|  |
| --- |
| df.shape |

**Output:**

|  |
| --- |
|  |

1. Finding the info. It is showing all information about the dataset.

|  |
| --- |
| df.info() |

**Output:**

|  |
| --- |
|  |

1. Finding the describe. It is showing lots of information like what is mean, std, min and etc.

|  |
| --- |
| df.describe() |

**Output:**

|  |
| --- |
|  |

1. Finding the columns into the dataset.

|  |
| --- |
| df.columns |

**Output:**

|  |
| --- |
|  |

1. **Feature Selection** in this part we are selecting columns and for create a better model.

|  |
| --- |
| f\_var=['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']  X=df[f\_var]  y=df['Species']  X.shape, y.shape |

**Output:**

|  |
| --- |
|  |

1. **Data Scaling**

|  |
| --- |
| # Standard scalar  from sklearn.preprocessing import StandardScaler  scaler\_data = StandardScaler()  X\_scaled\_data = scaler\_data.fit\_transform(X) |

1. **Data Spliting**.

|  |
| --- |
| # Importing libraries and data splitting into test and train  from sklearn.model\_selection import train\_test\_split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled\_data, y, test\_size=0.25, stratify=y)  X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape |

**Output:**

|  |
| --- |
|  |

1. **Model Implementation** with KNN algorithm.

|  |
| --- |
| from sklearn.neighbors import KNeighborsClassifier  from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix  # Create a K-Nearest Neighbors classifier  KNN\_Clf = KNeighborsClassifier(n\_neighbors=3) # You can specify the number of neighbors (K) here  # Train the KNN classifier on your training data  KNN\_Clf.fit(X\_train, y\_train)  y\_train\_predictions = KNN\_Clf.predict(X\_train)  y\_test\_predictions = KNN\_Clf.predict(X\_test)  training\_data\_accuracy = accuracy\_score(y\_train\_predictions, y\_train)  print('Accuracy score on the training data: %.2f' % training\_data\_accuracy)  test\_data\_accuracy = accuracy\_score(y\_test\_predictions, y\_test)  print('Accuracy score on the test data: %.2f' % test\_data\_accuracy)  # Display classification report and confusion matrix  print('\nClassification Report:\n', classification\_report(y\_test\_predictions, y\_test))  print('\nConfusion Matrix:\n', confusion\_matrix(y\_test\_predictions, y\_test)) |

**Output:**

|  |
| --- |
|  |