Group A

Assignment No:

10

Contents for Theory:

1. **Seaborn Library Basics**
2. **Know your Data**
3. **Finding patterns of data.**
4. **Data Visualization III Download the Iris flower dataset or any other dataset into a DataFrame. (e.g., https://archive.ics.uci.edu/ml/datasets/Iris). Scan the dataset and give the inference as:**

   1. List down the features and their types (e.g., numeric, nominal) available in the dataset.

   2. Create a histogram for each feature in the dataset to illustrate the feature distributions.

   3. Create a boxplot for each feature in the dataset. 4. Compare distributions and identify outliers

   4. Compare distributions and identify outliers

Theory:

Data Visualisation plays a very important role in Data mining. Various data scientists spent their time exploring data through visualisation. To accelerate this process we need to have a well-documentation of all the plots.

Even plenty of resources can't be transformed into valuable goods without planning and architecture

1. **Seaborn Library Basics**

Seaborn is a Python data visualisation library based on matplotlib. It provides ahigh-level interface for drawing attractive and informative statistical graphics.

For the installation of Seaborn, you may run any of the following in your command line.

```
pip install seaborn
conda install seaborn
```

To import seaborn you can run the following command.

```
import seaborn as sns
```

2. **Know your data**

The dataset that we are going to use to draw our plots will be the Titanic dataset, which is downloaded by default with the Seaborn library. All you have to do is use the load_dataset function and pass it the name of the dataset.

Let's see what the Titanic dataset looks like. Execute the following script:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


dataset = sns.load_dataset('iris')

dataset.head()
```

3. **Finding patterns of data.**

Patterns of data can be find out with the help of different types of plots

Types of plots are:

A. **Distribution Plots**

a. Dist-Plot

b. Joint Plot

d. Rug Plot

B. Categorical Plots

a. Bar Plot

b. Count Plot

c. Box Plot

d. Violin Plot

C. Advanced Plots

a. Strip Plot

b. Swarm Plot

D. Matrix Plots

a. Heat Map

b. Cluster Map

A. Distribution Plots:

These plots help us to visualise the distribution of data. We can use these plots to understand the mean, median, range, variance, deviation, etc of the data.

a. Distplot

- Dist plot gives us the histogram of the selected continuous variable.
- It is an example of a univariate analysis.
- We can change the number of bins i.e. number of vertical bars in a histogram

b. Joint Plot

- It is the combination of the distplot of two variables.
- It is an example of bivariate analysis.
- We additionally obtain a scatter plot between the variables to reflect their linear relationship. We can customise the scatter plot into a hexagonal plot, where, the more the colour intensity, the more will be the number of observations.

c. The Rug Plot

a. The rugplot() is used to draw small bars along the x-axis for each point in the dataset. To plot a rug plot, you need to pass the name of the column. Let's plot a rug plot for fare.

These are some of the most commonly used distribution plots offered by the Python's Seaborn Library. Let's see some of the categorical plots in the Seaborn library.

2. Categorical Plots

Categorical plots, as the name suggests, are normally used to plot categorical data. The categorical plots plot the values in the categorical column against another categorical column or a numeric column. Let's see some of the most commonly used categorical data.

b. The Bar Plot

The barplot() is used to display the mean value for each value in a categorical column, against a numeric column. The first parameter is the categorical column, the second parameter is the numeric column while the third parameter is the dataset.

From the output, you can clearly see that the average age of male passengers is just less than 40 while the average age of female passengers is around 33.

In addition to finding the average, the bar plot can also be used to calculate other aggregate values for each category. To do so, you need to pass the aggregate function to the estimator. For instance, you can calculate the standard deviation for the age of each gender as follows:

c. The Count Plot

The count plot is similar to the bar plot, however it displays the count of the categories in a specific column. For instance, if we want to count the number of males and women passenger we can do so using count plot as follows:

d. The Box Plot

The box plot is used to display the distribution of the categorical data in the form of quartiles. The centre of the box shows the median value. The value from the lower whisker to the bottom of the box shows the first quartile. From the bottom of the box to the middle of the box lies the second quartile. From the middle of the box to the top of the box lies the third quartile and finally from the top of the box to the top whisker lies the last quartile.

e. The Violin Plot

The violin plot is similar to the box plot, however, the violin plot allows us to display all the components that actually correspond to the data point. The violinplot() function is used to plot the violin plot. Like the box plot, the first parameter is the categorical column, the second

parameter is the numeric column while the third parameter is the dataset.

Now you can see a lot of information on the violin plot. For instance, if you look at the bottom of the violin plot for the males who survived (left-orange), you can see that it is thicker than the bottom of the violin plot for the males who didn't survive (left-blue). This means that the number of young male passengers who survived is greater than the number of young male passengers who did not survive

Advanced Plots:

a. The Strip Plot

The strip plot draws a scatter plot where one of the variables is categorical. We have seen scatter plots in the joint plot and the pair plot sections where we had two numeric variables. The strip plot is different in a way that one of the variables is categorical in this case, and for each category in the categorical variable, you will see a scatter plot with respect to the numeric column.

b. The Swarm Plot

The swarm plot is a combination of the strip and the violin plots. In the swarm plots, the points are adjusted in such a way that they don't overlap. Let's plot a swarm plot for the distribution of age against gender. The swarmplot() function is used to plot the violin plot. Like the box plot, the first parameter is the categorical column, the second parameter is the numeric column while the third parameter is the dataset. Look at the following script:

You can clearly see that the above plot contains scattered data points like the strip plot and the data points are not overlapping. Rather they are arranged to give a view similar to that of a violin plot.

1. Matrix Plots

Matrix plots are the type of plots that show data in the form of rows and columns. Heat maps are the prime examples of matrix plots.

a. Heat Maps

Heat maps are normally used to plot correlation between numeric columns in the form of a matrix. It is important to mention here that to draw matrix plots, you need to have meaningful information on rows as well as columns. Let's plot the first five rows of the Titanic dataset to see if both the rows and column headers have meaningful information. Execute the following script:

From the output, you can see that the column headers contain useful information such as passengers surviving, their age, fare etc. However the row headers only contain indexes 0, 1, 2, etc. To plot matrix plots, we need useful information on both columns and row headers. One way to do this is to call the corr() method on the dataset. The corr() function returns the correlation between all the numeric columns of the dataset. Execute the following script:

Now to create a heat map with these correlation values, you need to call the heatmap() function and pass it your correlation dataframe.

From the output, it can be seen that what heatmap essentially does is that it plots a box for every combination of rows and column value. The colour of the box depends upon the gradient. For instance, in the above image if there is a high correlation between two features, the corresponding cell or the box is white, on the other hand if there is no correlation, the corresponding cell remains black.

The correlation values can also be plotted on the heatmap by passing True for the annot parameter. Execute the following script to see this in action:

b. **Cluster Map:**

   In addition to the heat map, another commonly used matrix plot is the cluster map. The cluster map basically uses Hierarchical Clustering to cluster the rows and columns of the matrix.

Conclusion-

Seaborn is an advanced data visualisation library built on top of Matplotlib library. In this assignment, we looked at how we can draw distributional and categorical plots using the Seaborn library. We have seen how to plot matrix plots in Seaborn. We also saw how to change plot styles and use grid functions to manipulate subplots.

Assignment Questions

1. **List out different types of plot to find patterns of data**
2. **Explain when you will use distribution plots and when you will use categorical plots.**
3. **Write the conclusion from the following swarm plot (consider titanic dataset)**

4. **Which parameter is used to add another categorical variable to the violin plot, Explain with syntax and example.**

Group B
Assignment 11

PROBLEM STATEMENT:

Create databases and tables, insert small amounts of data, and run simple queries using Impala

OBJECTIVE:

- To Learn and understand the steps to create databases and tables in Impala, insert data into these tables, and execute simple queries.
- To learn and understand how to use Impala to analyze data.

Theory:

Impalas

- A tool which we use to overcome the slowness of Hive Queries is what we call Impala.
- Syntactically Impala queries run very faster than Hive Queries even after they are more or less same as Hive Queries.
- It offers high-performance, low-latency SQL queries.
- Impala is the best option while we are dealing with medium-sized datasets and we expect the real-time response from our queries. However, make sure Impala is available only in Hadoop distribution.
- Since MapReduce store intermediate results in the file system, Impala is not built on MapReduce. Hence, it is very slow for real-time query processing.

In addition, Impala has its own execution engine. Basically, that stores the intermediate results in In-memory. Therefore, when compared to other tools which use MapReduce its query execution is very fast.

Some Key Points

- It offers high-performance, low-latency SQL queries.
- Moreover, to share databases and tables between both Impala and Hive it integrates very well with the Hive Metastore.
- Also, it is Compatible with HiveQL Syntax

Impala is an open-source SQL query engine allows you to query data stored in Apache Hadoop clusters. It's a fast, distributed, and highly scalable system that can run complex

queries on large datasets in near real-time. In this manual, we will guide you through the process of creating databases and tables, inserting small amounts of data, and running simple queries using Impala.

Impala uses a distributed architecture, which means that data is stored across multiple nodes in a cluster. Impala also uses a query engine that allows you to write SQL queries that are executed in parallel across the nodes in the cluster. This allows for fast query execution, even on large datasets.

Impala uses a metadata store to keep track of the databases and tables that are created. The metadata store is used to store information about the location of the data, the schema of the tables, and other metadata. Impala supports a wide range of data formats, including Parquet, Avro, and RCFile.

Data Types:

BIGINT :- This datatype stores numerical values and the range of this data type is -9223372036854775808 to 9223372036854775807. This datatype is used in create table and alter table statements.

BOOLEAN:-This data type stores only true or false values and it is used in the column definition of create table statement.

CHAR:- This data type is a fixed length storage, it is padded with spaces, you can store up to the maximum length of 255.

DECIMAL:- This data type is used to store decimal values and it is used in create table and alter table statements.

DOUBLE:- This data type is used to store the floating point values in the range of positive or negative 4.94065645841246544e-324d -1.79769313486231570e+308.

FLOAT :- This data type is used to store single precision floating value datatypes in the range of positive or negative 1.40129846432481707e-45 .. 3.40282346638528860e+38.

Example:

To create a database in Impala, you can use the following SQL command:

```
CREATE DATABASE my_database;
```

This will create a new database called "my_database". To create a table within this database, you can use the following SQL command:

```
CREATE TABLE my_table ( id INT,   name STRING,  age INT );
```

This will create a new table called "my_table" with three columns: id, name, and age. To insert data into this table, you can use the following SQL command:

INSERT INTO my_table VALUES (1, 'John', 25), (2, 'Jane', 30), (3, 'Bob', 40);

This will insert three rows into the table, each with a unique id, name, and age. To run a simple query on this table, you can use the following SQL command:

SELECT * FROM my_table;

This will return all the rows in the table. You can also use more complex queries, such as aggregations and joins, to analyze the data in the table.

Conclusion:

Impala is a powerful SQL query engine that can be used to analyze large datasets stored in Apache Hadoop clusters. We have seen how to create databases and tables, insert data, and execute simple queries in Impala.

Group B

Assignment No: 12

Problem Statemen:

Write a simple program in SCALA using Apache Spark framework

Theory:

- Scala is an acronym for “Scalable Language”.
- It is a general-purpose programming language designed for the programmers who want to write programs in a concise, elegant, and typesafe way.
- Scala enables programmers to be more productive. Scala is developed as an object-oriented and functional programming language.
- If you write a code in Scala, you will see that the style is similar to a scripting language.
- Even though Scala is a new language, it has gained enough users and has a wide community support. It is one of the most user-friendly languages.

About Scala

- Scala is pure Object-Oriented programming language
- Scala is an object-oriented programming language.
- Everything in Scala is an object and any operations you perform is a method call.
- Scala, allow you to add new operations to existing classes with the help of implicit classes.
- One of the advantages of Scala is that it makes it very easy to interact with Java code.
- You can also write a Java code inside Scala class.
- The Scala supports advanced component architectures through classes and traits.

Scala is a functional language

- Scala is a programming language that has implemented major functional programming concepts.
- In Functional programming, every computation is treated as a mathematical function which avoids states and mutable data.
- The functional programming exhibits following characteristics:
- Power and flexibility
- Simplicity
- Suitable for parallel processing
- Scala is not a pure functional language. Haskell is an example of a pure functional language.

Scala is a compiler based language (and not interpreted)

- Scala is a compiler based language which makes Scala execution very fast if you compare it with Python (which is an interpreted language).

- The compiler in Scala works in similar fashion as Java compiler. It gets the source code and generates Java byte-code that can be executed independently on any standard JVM (Java Virtual Machine).
- There are more important points about Scala which I have not covered. Some of them are:
- Scala has thread based executors
- Scala is statically typed language
- Scala can execute Java code
- You can do concurrent and Synchronized processing in Scala
- Scala is JVM based languages

Installing Scala

- Scala can be installed in any Unix or windows-based system. Below are the steps to install for Ubuntu (14.04) for scala version 2.11.7.
- I am showing the steps for installing Scala (2.11.7) with Java version 7. It is necessary to install Java before installing Scala. You can also install latest version of Scala(2.12.1) as well. Step 0: Open the terminal Step 1: Install Java
- $ sudo apt-add-repository ppa:webupd8team/java $ sudo apt-get update $ sudo apt-get install oracle-java7installer
- If you are asked to accept Java license terms, click on "Yes" and proceed. Once finished, let us check whether Java has installed successfully or not. To check the Java version and installation, you can type:
- $ java -version
- Step 2: Once Java is installed, we need to install Scala
- $ cd ~/Downloads $ wget http://www.scala-lang.org/files/archive/scala-2.11.7.deb $ sudo dpkg -i scala2.11.7.deb $ scala –version

Scala Basics Terms

- Object: An entity that has state and behavior is known as an object. For example: table, person, car etc.
- Class: A class can be defined as a blueprint or a template for creating different objects which defines its properties and behavior.
- Method: It is a behavior of a class. A class can contain one or more than one method. For example: deposit can be considered a method of bank class.
- Closure: Closure is any function that closes over the environment in which it's defined. A closure returns value depends on the value of one or more variables which is declared outside this closure.
- Traits: Traits are used to define object types by specifying the signature of the supported methods. It is like interface in java.

Variable declaration in Scala

- In Scala, you can declare a variable using 'var' or 'val' keyword. The decision is based on whether it is a constant or a variable.
- If you use 'var' keyword, you define a variable as mutable variable. On the other hand, if you use 'val', you define it as immutable. Let's first declare a variable using "var" and then using "val".

Declare using var

var Var1 : String = "Ankit"

- In the above Scala statement, you declare a mutable variable called “Var1” which takes a string value. You can also write the above statement without specifying the type of variable. Scala will automatically identify it. For example:

  var Var1 = “Joshi"

Operations on Variables

- You can perform various operations on variables. There are various kinds of operators defined in Scala. For example: Arithmetic Operators, Relational Operators, Logical Operators, Bitwise Operators, Assignment Operators.
- Lets see “+” , “==” operators on two variables ‘Var4’, “Var5”. But, before that, let us first assign values to “Var4” and “Var5”.

  scala> var Var4 = 2 Output: Var4: Int = 2 scala> var Var5 = 3

  Output: Var5: Int = 3

  Now, let us apply some operations using operators in Scala.

  Apply ‘+’ operator Var4+Var5

  Output: res1: Int = 5 Apply “==” operator

  Var4==Var5 Output: res2: Boolean = false

- In Scala, if-else expression is used for conditional statements.
- YouThe ifcan write-oneelse expression in Scalaor more conditions inside “if”.
- Let’s declare a variable called “Var3” with a value 1 and then compare “Var3” using if-else expression.

- In the above snippet, the condition evaluates to True and hence True will be printed in the output.

Iteration in Scala

Like most languages, Scala also has a FOR-loop which is the most widely used method for iteration. It has a simple syntax too.

- Declare a simple function in Scala and call it by passing value
- You can define a function in Scala using “def” keyword. Let’s define a function called
- “mul2” which will take a number and multiply it by 10.
- You need to define the return type of function, if a function not returning any value you should use the “Unit” keyword.
- In the below example, the function returns an integer value. Let’s define the function

“mul2”:

def mul2(m: Int): Int = m * 10

Output: mul2: (m: Int)Int

Example:

Here is an example of a simple Scala program that prints "Hello, World!" to the console:

```
object HelloWorld {
  def main(args: Array[String]) {
    println("Hello, World!")
  }
}
```

In this program, we define an object called "HelloWorld". Inside the object, we define a method called "main" that takes an array of strings as an argument. The method simply prints the string "Hello, World!" to the console using the println() method.

To compile and run this program on Ubuntu, you will need to follow these steps:

Install the Scala compiler by running the following command in the terminal:

sudo apt-get install scala

Save the above code as "HelloWorld.scala" in a directory of your choice.

Open a terminal in the directory where you saved the file and run the following command to compile the code:

scalac HelloWorld.scala

This will generate a class file called "HelloWorld.class" in the same directory.

Run the program by entering the following command in the terminal:

scala HelloWorld

This will execute the "main" method of the "HelloWorld" object and print "Hello, World!" to the console.

Assignment Questions

1. Write a prime number code.
2. rite Factorial code.