LAB MANUAL OF

DS using Python Lab

Lab Code: ITL605

Class: TE Information Technology

Semester: VI (Rev-2019 ‘C’ Scheme)

Ms Archana S/Manisha P
Lab Incharges

H.O.D
(Dr. Pradip Mane)

Prerequisite: Basics of Python programming and Database management system.

Hardware & Software Requirements:

Hardware Requirements	Software Requirements
PC i3 processor and above	Python Libraries

Lab /syllabus:

Sr. No.	Module	Detailed Content	Hours	LO Mapping
1	I	i. Introduction, Benefits and uses of data science ii. Data Science tasks iii. Introduction to Pandas iv. Data preparation: Data cleansing, Data transformation, Combine/Merge /Join data, Data loading & preprocessing with pandas v. Data aggregation vi. Querying data in Pandas vii. Statistics with Pandas Data Frames viii. Working with categorical and text data ix. Data Indexing and Selection x. Handling Missing Data	04	LO1
2	II	i. Visualization with Matplotlib and Seaborn ii. Plotting Line Plots, Bar Plots, Histograms Density Plots, Paths, 3Dplot, Stream plot, Logarithmic plots, Pie chart, Scatter Plots and Image visualization using Matplotlib iii. Plotting scatter plot, box plot, Violin plot, swarm plot, Heatmap, Bar Plot using seaborn iv. Introduction to scikit-learn and SciPy v. Statistics using python: Linear algebra, Eigen value, Eigen Vector, Determinant, Singular Value Decomposition, Integration, Correlation, Central Tendency, Variability, Hypothesis testing, Anova, z- test, t-test and chi-square test.	04	LO2
3	III	i. What is Machine Learning? ii. Applications of Machine Learning; iii. Introduction to Supervised Learning iv. Overview of Regression v. Support Vector Machine vi. Classification algorithms	05	LO3

4	IV	i. Introduction to Unsupervised Learning ii. Overview of Clustering iii. Decision Trees iv. Random Forests v. Association	05	LO4
5	V	i. Introduction to Apache Spark ii. Architecture of Apache Spark iii. Modes and components iv. Basics of PySpark	04	LO5
6	VI	i. Understanding the different data science phases used in selected case study ii. Implementation of Machine learning algorithm for selected case study	04	LO1,LO6

Textbooks:

1. Jake VanderPlas, “Python Data Science Handbook”, O’Reilly publication
2. Frank Kane, “Hands-On Data Science and Python Machine Learning”, packt publication
3. M.T. Savaliya, R.K. Maurya, G.M.Magar, “Programming with Python”, 2nd Edition, Sybgen Learning.

References:

1. Armando Fandango, “Python Data Analysis”, Second Edition, Packt publication.
2. Alberto Boschetti, Luca Massaron, “Python Data Science Essentials Second Edition”, Packt Publishing
3. Davy Cielen, Arno D. B. Meysman, Mohamed Ali, “Introducing Data Science”, Manning Publications.

List of Experiments

Sr. No	Title	LO
1	Data preparation using NumPy and Pandas a. Derive an index field and add it to the data set. b. Obtain a listing of all records that are outliers according to the any field.	LO1
2	Data Visualization / Exploratory Data Analysis for the selected data set using Matplotlib and Seaborn a. Create a bar graph, contingency table using any 2 variables. b. Create a normalized histogram.	LO2
3	Data Modelling : a. Identify the total number of records in the training data set. b. Validate your partition by performing a two-sample Z-test	LO2
4	Implementation of Statistical Hypothesis Test using Scipy and Sci-kit learn 1. Pearson's Correlation Coefficient 2. Spearman's Rank Correlation 3. Kendall's Rank Correlation 4. Chi-Squared Test	LO2
5	Regression Analysis a. Perform Logistic Regression to find out relation between variables. b. Apply regression Model techniques to predict the data on above dataset	LO3
6	Classification modeling a. Choose classifier for classification problem. b. Evaluate the performance of classifier.	LO3
7	Clustering a. Clustering algorithms for unsupervised classification. b. Plot the cluster data	LO4
8	Develop a recommendation system by Applying any machine learning techniques and using available data set	LO4
9	Exploratory data analysis using Apache Spark and Pandas	LO5
10	Batch and streamed Data Analysis using Spark.	LO5
11	Implementation of Mini project based on a case study using Data science and Machine learning. 1) Understand the different data science phases used in selected case study. 2) Implementation of Machine learning algorithm for selected case study.	LO1-LO6

EXPERIMENT NO: 1

Aim: Data preparation using NumPy and Pandas

a. Derive an index field and add it to the data set.
b. Obtain a listing of all records that are outliers according to the any field.

NumPy:

- NumPy is a short form for Numerical Python
- It is the fundamental package for scientific computing in Python.
- It is a Python library that provides a multidimensional array object, various derived objects and an assortment of routines for fast operations on arrays

Features of NumPy

- ➢ NumPy has various features including these important ones:
- ➢ A powerful N-dimensional array object
- ➢ Sophisticated (broadcasting) functions
- ➢ Tools for integrating C/C++ and Fortran code
- ➢ Useful linear algebra, Fourier transform, and random number capabilities

Pandas:

- Pandas is a powerful and versatile library that simplifies tasks of data manipulation in Python .
- Pandas is built on top of the NumPy library and is particularly well-suited for working with tabular data, such as spreadsheets or SQL tables.
- Its versatility and ease of use make it an essential tool for data analysts, scientists, and engineers working with structured data in Python.

Features of Pandas:

- ➢ Efficient data handling
- ➢ Flexibility
- ➢ Easy integration with other libraries
- ➢ Wide adoption and support
- ➢ Readability
- ➢ Handling diverse data sources

Seaborn:

- Seaborn is a Python data visualization library based on matplotlib.
- Seaborn is a library for making statistical graphics in Python.
- It builds on top of matplotlib and integrates closely with Pandas data structures. Seaborn helps you explore and understand your data.

Features:

- ➢ Built in themes for styling matplotlib graphics
- ➢ Visualizing univariate and bivariate data
- ➢ Fitting in and visualizing linear regression models
- ➢ Plotting statistical time series data
- ➢ Seaborn works well with NumPy and Pandas data structures

- It comes with built in themes for styling Matplotlib graphics

Boxplot in seaborn:

- Box plots provide a quick visual summary of the variability of values in a dataset.
- They show the median, upper and lower quartiles, minimum and maximum values, and any outliers in the dataset.
- Outliers can reveal mistakes or unusual occurrences in data.

A box plot consists of 5 things.

- Minimum
- First Quartile or 25%
- Median (Second Quartile) or 50%
- Third Quartile or 75%
- Maximum

a. <u>Derive an index field and add it to the data set.</u>

1. Import the Pandas and NumPy:

```
In [1]: import pandas as pd

In [2]: import numpy as np
```

2. **Read:**to read data stored as a csv file into a Pandas DataFrame.

```
In [3]: df = pd.read_csv('dr.csv')

In [5]: print(df.to_string())
```

```
          Date  Facebook  Pinterest  Twitter  St
umbleUpon  YouTube  Instagram  Tumblr  reddit
VKontakte  LinkedIn  Google+  Digg  MySpace  F
ark  NowPublic  iWiW  news.ycombinator.com  De
licious  orkut  Odnoklassniki  Vimeo  Other
0    2009-04     20.16        0.00     6.86
36.79     0.00       0.00     0.00     8.98
0.00       0.00     0.00   6.70    14.81  0.22
0.04  0.29                    0.08        0.49
1.75            0.00    0.00   2.83
1    2009-05     24.30        0.00     9.95
33.78     0.00       0.00     0.00     7.62
0.00       0.28     0.00   7.34     8.95  0.44
```

3. Head: returns a specified number of rows, string from the top. This method returns the first 5 rows if a number is not specified.

In [6]:
```
df.head()
```

Out[6]:

	Date	Facebook	Pinterest	Twitter	StumbleUpon	YouTube	Instagram
0	2009-04	20.16	0.0	6.86	36.79	0.0	0.0
1	2009-05	24.30	0.0	9.95	33.78	0.0	0.0
2	2009-06	26.48	0.0	10.56	29.65	0.0	0.0
3	2009-07	29.10	0.0	10.35	33.55	0.0	0.0
4	2009-08	34.25	0.0	11.15	29.01	0.0	0.0

4. Tail: returns a specified number of last rows. This method returns the last 5 rows if a number is not specified.

In [7]:
```
df.tail()
```

Out[7]:

	Date	Facebook	Pinterest	Twitter	StumbleUpon	YouTube	Insta
173	2023-09	65.24	7.28	8.75	0.00	4.00	
174	2023-10	65.15	8.47	8.75	0.01	4.43	
175	2023-11	63.56	10.02	8.49	0.00	5.02	
176	2023-12	64.79	9.97	7.75	0.00	5.88	
177	2009-03	0.00	0.00	0.00	0.00	0.00	

5. **Loc**: It is used for label indexing and can access multiple columns

```
In [10]: df.loc[3:5]
```

Out[10]:

	Date	Facebook	Pinterest	Twitter	StumbleUpon	YouTube	Instagram	T
3	2009-07	29.10	0.0	10.35	33.55	0.00	0.0	
4	2009-08	34.25	0.0	11.15	29.01	0.00	0.0	
5	2009-09	39.89	0.0	10.96	27.31	0.23	0.0	

6. **Describe:** It returns description of the data in the DataFrame

```
In [13]: df.describe()
```

Out[13]:

	Facebook	Pinterest	Twitter	StumbleUpon	YouTube	Instagram
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000
mean	69.210056	7.308427	7.266742	5.176854	3.767753	2.178315
std	12.722400	4.672744	2.550880	8.729947	2.674020	3.522951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	64.677500	5.105000	5.517500	0.010000	1.405000	0.000000
50%	68.935000	7.320000	6.840000	0.385000	3.845000	0.000000
75%	75.527500	10.685000	8.750000	5.737500	5.055000	2.400000
max	87.830000	16.960000	15.480000	36.790000	11.040000	14.320000

7. **Axes:** It is used to access the group of rows and columns labels of the given DataFrame.

```
In [15]: df.axes

Out[15]: [RangeIndex(start=0, stop=178, step=1),
          Index(['Date', 'Facebook', 'Pinterest',
                 'Instagram', 'Tumblr', 'reddit',
                 'Digg', 'MySpace', 'Fark', 'NowP
                 'Delicious', 'orkut', 'Odnoklass
                dtype='object')]
```

8. **Shape:** It enables us to obtain the shape of a DataFrame

```
In [16]: df.shape

Out[16]: (178, 23)
```

9. **Info:** It prints information about the DataFrame.

```
In [17]: df.info

Out[17]: <bound method DataFrame.info of          Date  Facebook  Pinterest
         0     2009-04     20.16     0.00    6.86      36.79     0.00
         1     2009-05     24.30     0.00    9.95      33.78     0.00
         2     2009-06     26.48     0.00   10.56      29.65     0.00
         3     2009-07     29.10     0.00   10.35      33.55     0.00
         4     2009-08     34.25     0.00   11.15      29.01     0.00
         ..        ...       ...      ...     ...        ...      ...
         173   2023-09     65.24     7.28    8.75       0.00     4.00
         174   2023-10     65.15     8.47    8.75       0.01     4.43
         175   2023-11     63.56    10.02    8.49       0.00     5.02
         176   2023-12     64.79     9.97    7.75       0.00     5.88
         177   2009-03      0.00     0.00    0.00       0.00     0.00
```

10. **Isnull:** Detect missing values for an array-like object.

```
In [18]: df.isnull()
```

Out[18]:

	Date	Facebook	Pinterest	Twitter	StumbleUpon	YouTube	Instagram
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False

11. **Isnull().sum():** Calling the sum() method on the isnull() series returns the count of True values which actually corresponds to the number of NaN values.

```
In [19]: df.isnull().sum()

Out[19]: Date            0
         Facebook        0
         Pinterest       0
         Twitter         0
         StumbleUpon     0
         YouTube         0
         Instagram       0
```

12. **Iloc:** It returns a view of the selected rows and columns from a Pandas DataFrame.

```
In [23]: df.iloc[2:5]
Out[23]:
```

	Date	Facebook	Pinterest	Twitter	StumbleUpon	YouTube	Instagram
2	2009-06	26.48	0.0	10.56	29.65	0.0	0.0
3	2009-07	29.10	0.0	10.35	33.55	0.0	0.0
4	2009-08	34.25	0.0	11.15	29.01	0.0	0.0

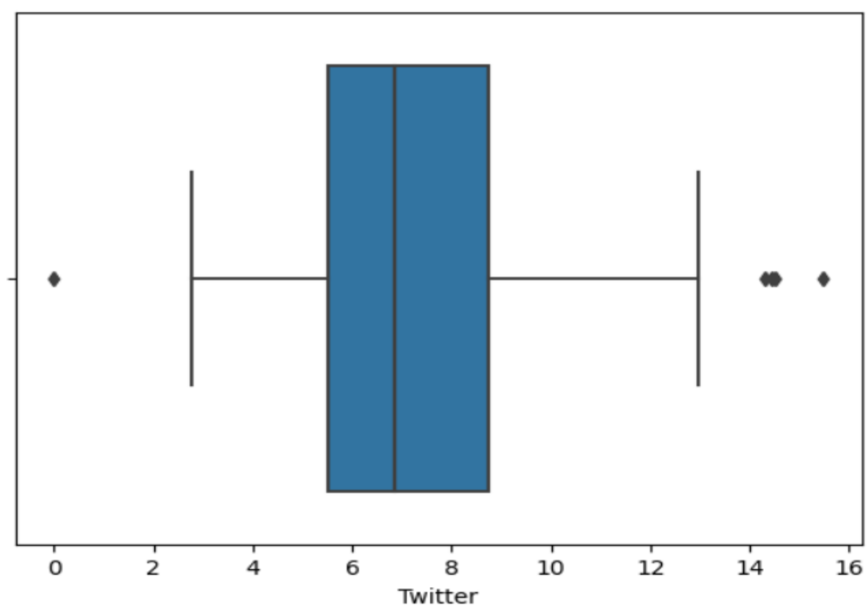
b. <u>Obtain a listing of all records that are outliers according to the any field.</u>

1. Import seaborn:

```
In [24]: import seaborn as sb
```

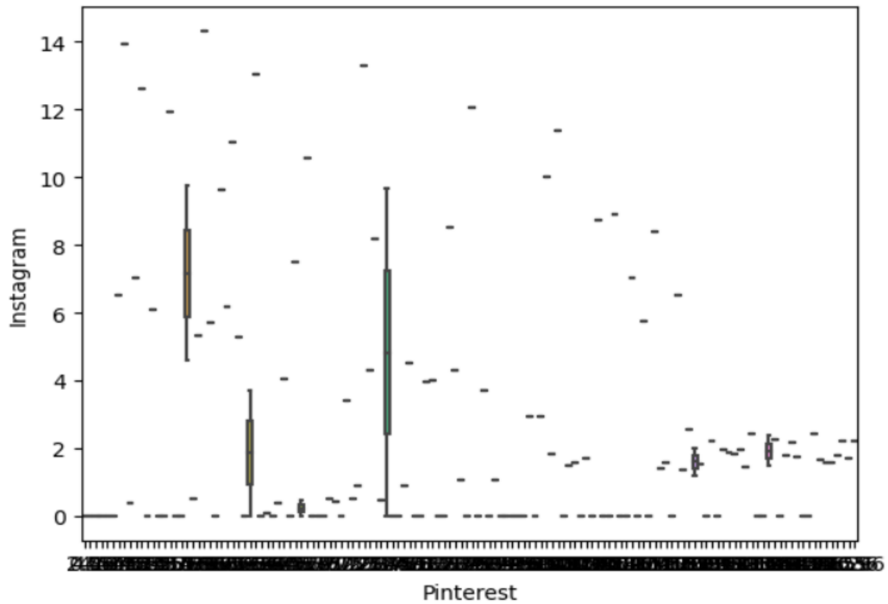
2. Draw a single Horizontal boxplot:

```
In [25]: sb.boxplot(x=df["Twitter"])
Out[25]: <Axes: xlabel='Twitter'>
```



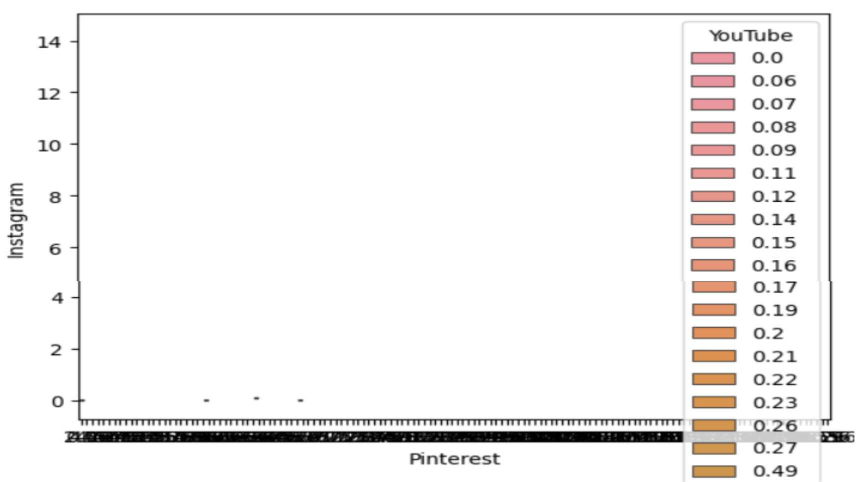
3. Group by Categorical variables:

```
In [31]: sb.boxplot(data=df, x="Pinterest", y="Instagram")
Out[31]: <Axes: xlabel='Pinterest', ylabel='Instagram'>
```



4. Draw a vertical boxplot with nested grouping by 2 variables:

```
In [33]: sb.boxplot(data=df, x="Pinterest", y="Instagram",hue="YouTube")
Out[33]: <Axes: xlabel='Pinterest', ylabel='Instagram'>
```



Conclusion: Hence, we have successfully implemented data preparation using NumPy and Pandas.

EXPERIMENT NO: 2

Aim: Data Visualization/Exploratory data Analysis for the selected data set using Matplotlib and Seaborn

- **a.** Create a bar graph, contingency table using any 2 variables.
- **b.** Create a normalized histogram

Data Visualization:

- Data visualization is the representation of information and data using charts, graphs, maps, and other visual tools.
- These visualizations allow us to easily understand any patterns, trends, or outliers in a data set.

Seaborn:

- Seaborn is a Python data visualization library based on matplotlib.
- Seaborn is a library for making statistical graphics in Python.
- It builds on top of matplotlib and integrates closely with Pandas data structures. Seaborn helps you explore and understand your data.

Features:

- ➢ Built in themes for styling matplotlib graphics
- ➢ Visualizing univariate and bivariate data
- ➢ Fitting in and visualizing linear regression models
- ➢ Plotting statistical time series data
- ➢ Seaborn works well with NumPy and Pandas data structures
- ➢ It comes with built in themes for styling Matplotlib graphics

Matplotlib:

- Matplotlib is extremely powerful because it allows users to create numerous and diverse plot types.
- It can be used in variety of user interfaces such as IPhython shells, Python scripts, Jupyter notebooks, as well as web applications and GUI toolkits.
- It has support for LaTeX-formatted labels and texts.

Import the NumPy, Pandas and Matplotlib libraries:

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
```

Open the dataset and read the data:

```
df1 = pd.read_csv("C:/Users/DELL/Downloads/tested.csv")
```

```
df1.head()
```

Python

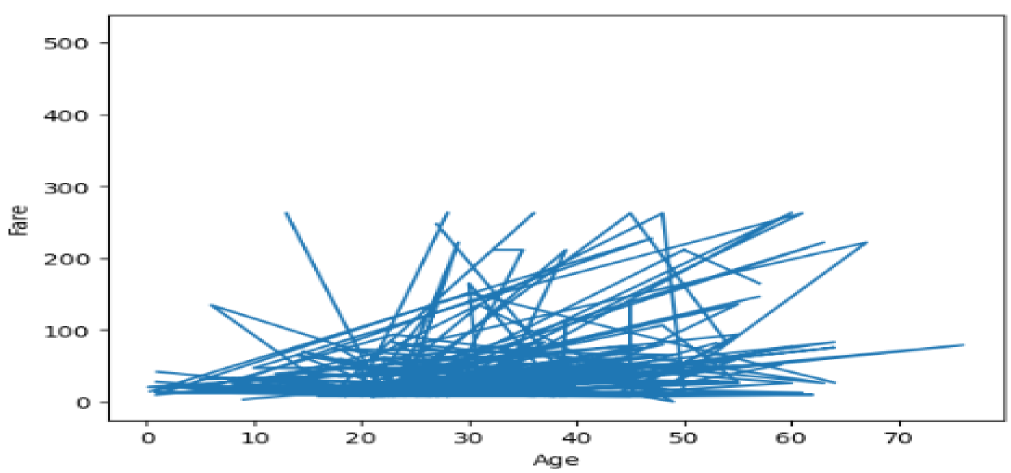
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

Line plots:

- Line plots are drawn by joining straight lines connecting data points where the x-axis and y-axis values intersect.
- Line plots are the simplest form of representing data. In Matplotlib, the plot() function represents this.

```
x = df1.Age
y = df1.Fare
plt.xlabel("Age")
plt.ylabel("Fare")

plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x22ebc30d750>]
```

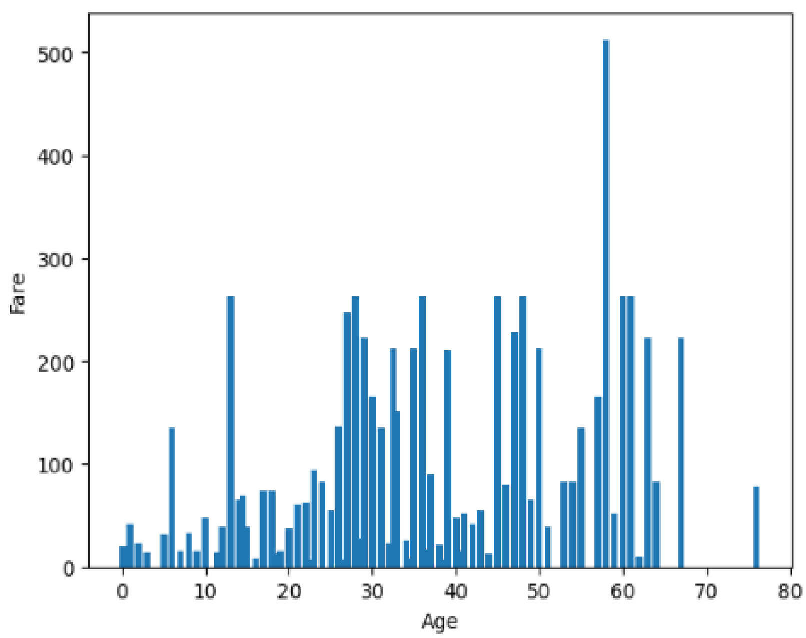


Bar Plot:

- The bar plots are vertical/horizontal rectangular graphs that show data comparison where you can gauge the changes over a period represented in another axis (mostly the X-axis).
- we use the bar() or bar() function to represent it.

```
x = df1.Age
y = df1.Fare
plt.xlabel("Age")
plt.ylabel("Fare")
plt.bar(x,y)
```

```
<BarContainer object of 418 artists>
```

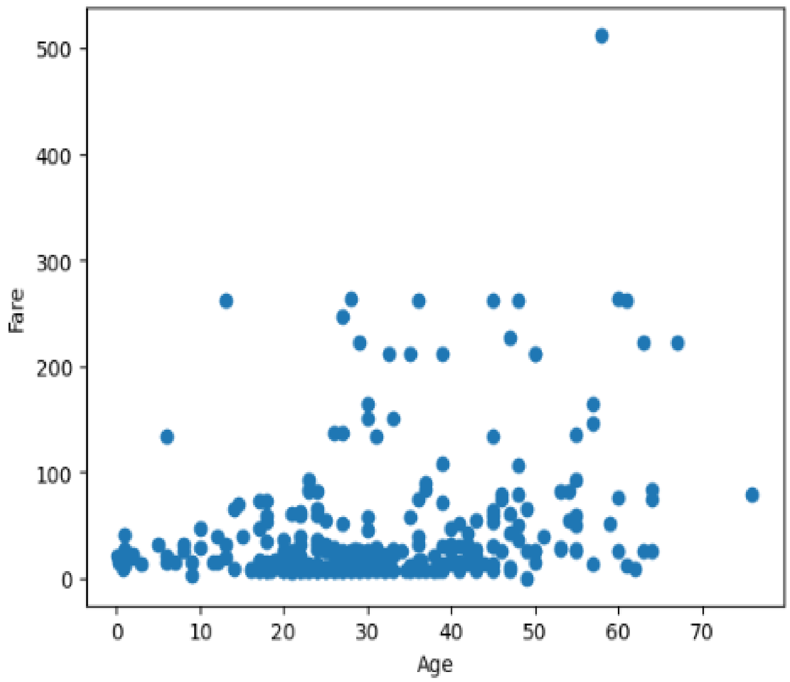


Scatter plot:

- We can implement the scatter plots while comparing various data variables to determine the connection between dependent and independent variables.
- The data gets expressed as a collection of points clustered together meaningfully.
- Here each value has one variable (x) determining the relationship with the other (Y).

```
plt.xlabel("Age")
plt.ylabel("Fare")
x = df1.Age
y = df1.Fare
plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x22ebd5c8350>
```

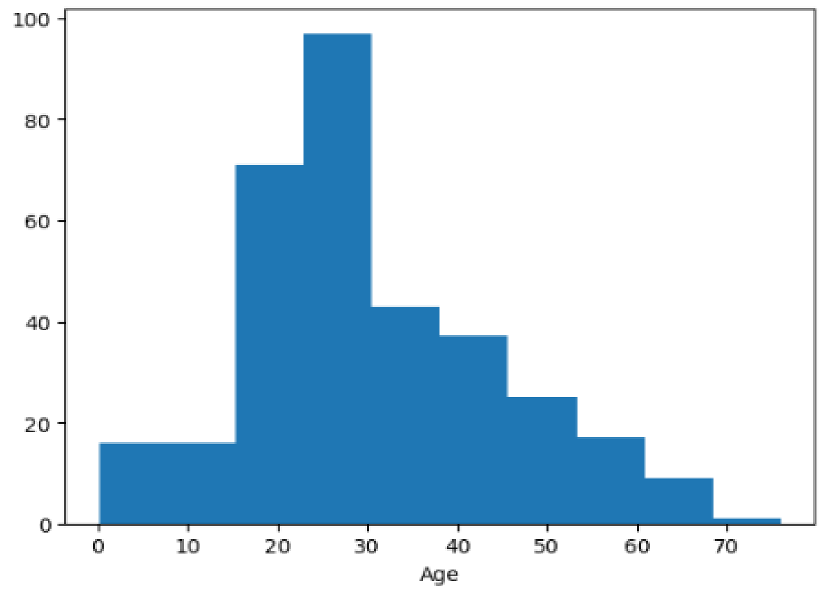


Histogram plot:

- We can use a histogram plot when the data remains distributed, whereas we can use a bar graph to compare two entities.
- Both histogram and bar plot look alike but are used in different scenarios. In Matplotlib, the hist() function represents this

```
plt.xlabel("Age")
plt.hist(x)
```

```
(array([16., 16., 71., 97., 43., 37., 25., 17.,  9.,  1.]),
 array([ 0.17 ,  7.753, 15.336, 22.919, 30.502, 38.085, 45.668
        60.834, 68.417, 76.   ]),
 <BarContainer object of 10 artists>)
```



Conclusion:

Hence, we have successfully implemented Data Visualization/Exploratory data Analysis for the selected data set using Matplotlib and Seaborn.

EXPERIMENT NO: 3

Aim: Data Modeling:

a. Identify the total number of records in the training data set.
b. Validate your partition by performing a two-sample z-test.

Data Modeling:

- The act of deciding how to break up your application data into multiple tables and establishing the relationships between the tables is called data modeling.

Sample Z-test:

- The one-sample z-test is used to test whether the mean of a population is greater than, less than, or not equal to a specific value.
- Because the standard normal distribution is used to calculate critical values for the test, this test is often called the one-sample z-test.
- The z-test assumes that the population standard deviation is known.

sklearn.model_selection:

- **sklearn.model_selection** is a module within the scikit-learn library (also known as sklearn) in Python.
- This module provides tools for working with the model selection process in machine learning, including techniques for splitting datasets, cross-validation, and hyperparameter tuning.

train_test_split:

- In Python, train_test_split is a function provided by the scikit-learn library (sklearn) in the sklearn.model_selection module.
- This function is commonly used for splitting a dataset into two subsets: one for training a machine learning model and the other for testing or validating the model.
- The purpose of this split is to evaluate how well the model performs on new, unseen data.

Import Libraries:

- **Pandas:** used for data manipulation and anlysis
- **Matplotlib.pyplot:** used for data visualization
- **Train_text_split from sklearn.model_selection:** used to split the dataset into training and test sets.

```
In [2]: #import library
        import pandas as pd
        import matplotlib.pyplot as plt
        from sklearn.model_selection import train_test_split
```

Load the titanic dataset:

```
In [3]: # Load the Titanic dataset
        titanic_data = pd.read_csv('C:/Users/squir/Downloads/tested.csv')
```

Display the fist few rows of the dataset:

```
In [4]: # Display the first few rows of the dataset to understand its structure
        print(titanic_data.head())
```

```
   PassengerId  Survived  Pclass  \
0          892         0       3
1          893         1       3
2          894         0       2
3          895         0       3
4          896         1       3

                                           Name     Sex   Age  SibSp  Parch
\
0                              Kelly, Mr. James    male  34.5      0      0
1              Wilkes, Mrs. James (Ellen Needs)  female  47.0      1      0
2                     Myles, Mr. Thomas Francis    male  62.0      0      0
3                              Wirz, Mr. Albert    male  27.0      0      0
4  Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female  22.0      1      1

    Ticket     Fare Cabin Embarked
0   330911   7.8292   NaN        Q
1   363272   7.0000   NaN        S
2   240276   9.6875   NaN        Q
3   315154   8.6625   NaN        S
4  3101298  12.2875   NaN        S
```

Define features x and target variables y:

- X: contains the features by dropping the survived column
- Y: contains the target variable which is survived

```
In [5]: # Define features (X) and target variable (y)
        X = titanic_data.drop('Survived', axis=1)
        y = titanic_data['Survived']
```

Split data into Training and Test sets:

- test_size: It specifies proportion of the dataset to include in the test split.
- random_state: It sets a seed for reproducibility

```
In [6]: # Split the data into training and test sets (75% training, 25% test)
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ranc
```

Check proportions and visualize using bar graph:

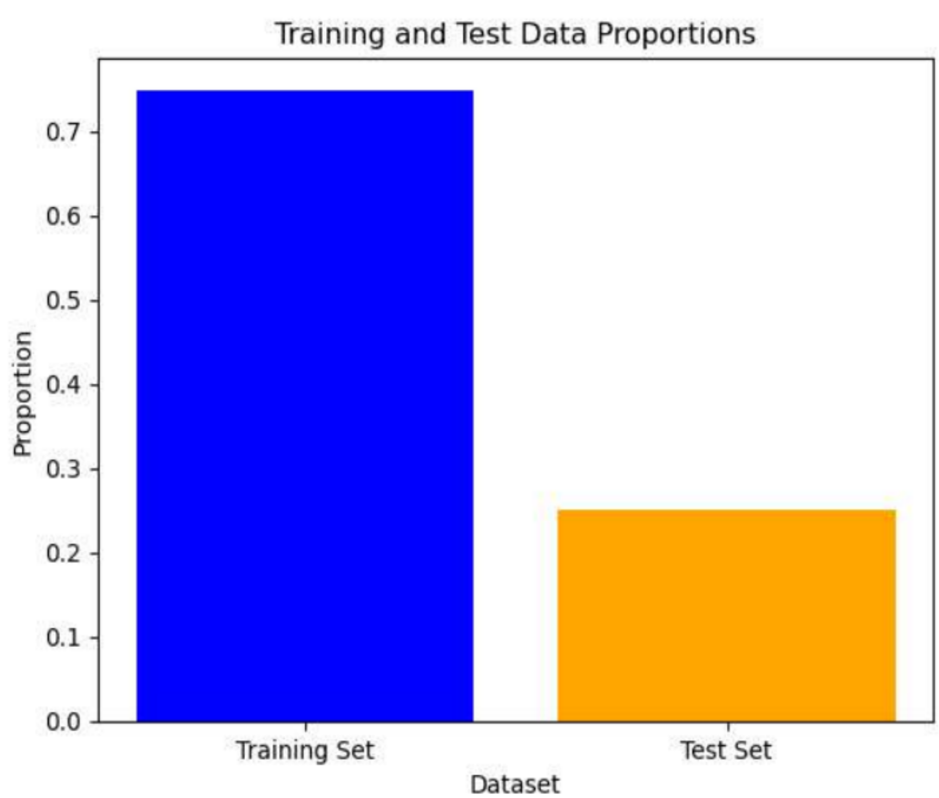
- Proportions: It id a list containing the calculated proportions
- Labels: It is containing labels for the bar graph.

In [7]:
```
# Check the proportions and visualize using a bar graph
proportions = [len(X_train) / len(titanic_data), len(X_test) / len(titanic_dat
labels = ['Training Set', 'Test Set']
```

Plot the bar Graph:

- Plt.bar: Creates the bar graph.
- Plt.xlable, plt.ylabel. plt.title: Adds labels and title to the graph.
- Plt.show(): Display the graph

In [8]:
```
# Plotting the bar graph
plt.bar(labels, proportions, color=['blue', 'orange'])
plt.xlabel('Dataset')
plt.ylabel('Proportion')
plt.title('Training and Test Data Proportions')
plt.show()
```



Display the total no. of records in the training set:

```
In [9]: # Display the total number of records in the training set
        num_records_training_set = X_train.shape[0]
        print("Total number of records in the training data set:", num_records_trainir

        Total number of records in the training data set: 313
```

Scipy:

- It is an open-source library for mathematics, science, and engineering.
- SciPy builds on the capabilities of NumPy and provides additional functionality for a wide range of scientific computing tasks.
- It includes modules for optimization, signal and image processing, linear algebra, integration, interpolation, statistical functions, and more.

Stats in python:

- The term "stats" is often used to refer to statistical functions and methods available in various libraries, particularly in the context of data analysis and scientific computing.
- The most common library for statistical functions in Python is the **scipy.stats** module within the SciPy library.

Import library:

```
In [10]: #Importing Library
         from scipy import stats
```

Calculate the mean of the column:

```
In [13]: # Calculate the mean of the 'Survived' column for training and test sets
         mean_survived_train = y_train.mean()
         mean_survived_test = y_test.mean()
```

Perform two sample t-test:

- The two-sample t-test (also known as the independent samples t-test) is a method used to test whether the unknown population means of two groups are equal or not.

```
In [14]: # Perform a two-sample t-test
         t_stat, p_value = stats.ttest_ind(y_train, y_test)
```

Display mean and t-test results:

```
In [16]: # Display the means and t-test results
print("Mean of 'Survived' in Training Set:", mean_survived_train)
print("Mean of 'Survived' in Test Set:", mean_survived_test)
print("\nT-test results:")
print("T-statistic:", t_stat)
print("P-value:", p_value)
```

```
Mean of 'Survived' in Training Set: 0.3706070287539936
Mean of 'Survived' in Test Set: 0.34285714285714286

T-test results:
T-statistic: 0.5104439211062601
P-value: 0.610011234153832
```

Calculate mean of column for training and test sets:

```
In [17]: # Calculate the mean of the 'Survived' column for training and test sets
mean_survived_train = round(y_train.mean())
mean_survived_test = round(y_test.mean())

# Perform a two-sample t-test
t_stat, p_value = stats.ttest_ind(y_train, y_test)

# Display the means and t-test results as integers
print("Mean of 'Survived' in Training Set:", int(mean_survived_train))
print("Mean of 'Survived' in Test Set:", int(mean_survived_test))
print("\nT-test results:")
print("T-statistic:", t_stat)
print("P-value:", p_value)
```

```
Mean of 'Survived' in Training Set: 0
Mean of 'Survived' in Test Set: 0

T-test results:
T-statistic: 0.5104439211062601
P-value: 0.610011234153832
```

Conclusion:

Hence, we have successfully studied &implemented Data Modeling and tests.

EXPERIMENT NO.. 4

Aim: Implementation of Statistical Hypothesis Test using Scipy and Sci-kit learn
1. Pearson’s Correlation Coefficient
2. Spearman’s Rank Correlation
3. Kendall’s Rank Correlation
4. Chi-Squared Test

Theory:
The Pearson's Correlation Coefficient is also known as the *Pearson Product-Moment Correlation Coefficient*. It is a measure of the linear relationship between two random variables - **X** and **Y**. Mathematically, if (**σXY**) is the covariance between **X** and **Y**, and (**σX**) is the standard deviation of **X**, then the Pearson's correlation coefficient **ρ** is given by:

Program:

```
importnumpy  np
importmatplotlib.pyplot as plt
```

We'll use the same values from the manual example from before. Let's store that into `x_simple` and compute the correlation matrix:

```
x_simple = np.array([-, -, , , ])
y_simple = np.array([, , , , ])
my_rho = np.corrcoef(x_simple, y_simple)

print(my_rho)
```

The following is the output correlation matrix. Note the ones on the diagonals, indicating that the correlation coefficient of a variable with itself is one:

```
[[ 1.  -0.7]
[-0.7  1. ]]
```

Output:

```
In [2]: # Importing Libraries

import pandas as pd
from scipy.stats import shapiro, normaltest, anderson, pearsonr, spearmanr, kendalltau, chi2_contingency
from statsmodels.tsa.stattools import adfuller, kpss
from scipy.stats import ttest_ind, ttest_rel, f_oneway, mannwhitneyu, wilcoxon, kruskal, friedmanchisquare

In [13]: # Load Titanic dataset

titanic_df= pd.read_csv('C:/Users/squir/Downloads/tested.csv')
```

In [13]:
```python
# Load Titanic dataset

titanic_df= pd.read_csv('C:/Users/squir/Downloads/tested.csv')
```

In [14]:
```python
titanic_df.head()
```

Out[14]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

In [15]:
```python
# Drop irrelevant columns
titanic_df = titanic_df.drop(['Name', 'Ticket', 'Cabin', 'Embarked'], axis=1)
```

In [17]:
```python
# Handle missing values
titanic_df['Age'].fillna(titanic_df['Age'].median(), inplace=True)
titanic_df['Fare'].fillna(titanic_df['Fare'].median(), inplace=True)
```

In [18]:
```python
# Convert categorical variables to numerical
titanic_df['Sex'] = titanic_df['Sex'].map({'male': 0, 'female': 1})
```

In [19]:
```python
# Create a new variable 'Survived' to represent survival as 1 and not survival as 0
titanic_df['Survived'] = titanic_df['Survived'].astype(int)
```

In [20]:
```python
# Normality Tests
# Shapiro-Wilk Test
stat, p_value = shapiro(titanic_df['Age'])
print(f"Shapiro-Wilk Test: Statistics={stat}, p-value={p_value}")

# D'Agostino's K^2 Test
stat, p_value = normaltest(titanic_df['Age'])
print(f"D'Agostino's K^2 Test: Statistics={stat}, p-value={p_value}")

# Anderson-Darling Test
result = anderson(titanic_df['Age'])
print(f"Anderson-Darling Test: Statistic={result.statistic}, Critical Values={result.critical_values}, Significance Level={result
```

```
Shapiro-Wilk Test: Statistics=0.9353150129318237, p-value=1.7022099875474428e-12
D'Agostino's K^2 Test: Statistics=34.81555255420222, p-value=2.7535871142178257e-08
Anderson-Darling Test: Statistic=12.460808507225352, Critical Values=[0.571 0.65  0.78  0.909 1.082], Significance Level=[15.
10.   5.   2.5  1. ]
```

In [21]:
```python
# Correlation Tests
# Pearson's Correlation Coefficient
correlation, p_value = pearsonr(titanic_df['Age'], titanic_df['Fare'])
print(f"Pearson's Correlation Coefficient: Correlation={correlation}, p-value={p_value}")

# Spearman's Rank Correlation
correlation, p_value = spearmanr(titanic_df['Age'], titanic_df['Fare'])
print(f"Spearman's Rank Correlation: Correlation={correlation}, p-value={p_value}")

# Kendall's Rank Correlation
correlation, p_value = kendalltau(titanic_df['Age'], titanic_df['Fare'])
print(f"Kendall's Rank Correlation: Correlation={correlation}, p-value={p_value}")

# Chi-Squared Test
contingency_table = pd.crosstab(titanic_df['Survived'], titanic_df['Sex'])
chi2_stat, p_value, dof, expected = chi2_contingency(contingency_table)
print(f"Chi-Squared Test: Chi2 Statistic={chi2_stat}, p-value={p_value}, Degrees of Freedom={dof}")
```

```
Pearson's Correlation Coefficient: Correlation=0.34235685018571027, p-value=6.147154025484477e-13
Spearman's Rank Correlation: Correlation=0.27724790736124283, p-value=8.177127214177605e-09
Kendall's Rank Correlation: Correlation=0.18843374022157644, p-value=2.7225756670044467e-08
Chi-Squared Test: Chi2 Statistic=413.6897405343716, p-value=5.767311139789629e-92, Degrees of Freedom=1
```

```
In [23]: # Parametric Statistical Hypothesis Tests
         # Student's t-test
         stat, p_value = ttest_ind(titanic_df['Age'], titanic_df['Fare'])
         print(f"Student's t-test: t-statistic={stat}, p-value={p_value}")

         # Paired Student's t-test
         stat, p_value = ttest_rel(titanic_df['Age'], titanic_df['Fare'])
         print(f"Paired Student's t-test: t-statistic={stat}, p-value={p_value}")

         # Analysis of Variance Test (ANOVA)
         result = f_oneway(titanic_df['Age'], titanic_df['Fare'])
         print(f"Analysis of Variance Test (ANOVA): F-statistic={result.statistic}, p-value={result.pvalue}")

         Student's t-test: t-statistic=-2.133594291454706, p-value=0.033167229221356787
         Paired Student's t-test: t-statistic=-2.3116048159532463, p-value=0.021286109068868662
         Analysis of Variance Test (ANOVA): F-statistic=4.552224600528116, p-value=0.03316722922137293
```


```
Student's t-test: t-statistic=-2.133594291454706, p-value=0.033167229221356787
Paired Student's t-test: t-statistic=-2.3116048159532463, p-value=0.021286109068868662
Analysis of Variance Test (ANOVA): F-statistic=4.552224600528116, p-value=0.03316722922137293
```


```
In [25]: # Nonparametric Statistical Hypothesis Tests
         # Mann-Whitney U Test
         stat, p_value = mannwhitneyu(titanic_df['Age'], titanic_df['Fare'])
         print(f"Mann-Whitney U Test: U-statistic={stat}, p-value={p_value}")

         # Wilcoxon Signed-Rank Test
         stat, p_value = wilcoxon(titanic_df['Age'], titanic_df['Fare'])
         print(f"Wilcoxon Signed-Rank Test: W-statistic={stat}, p-value={p_value}")

         # Kruskal-Wallis H Test
         stat, p_value = kruskal(titanic_df['Age'], titanic_df['Fare'])
         print(f"Kruskal-Wallis H Test: H-statistic={stat}, p-value={p_value}")

         # Friedman Test
         stat, p_value = friedmanchisquare(titanic_df['Age'], titanic_df['Fare'], titanic_df['Survived'])
         print(f"Friedman Test: Chi2-statistic={stat}, p-value={p_value}")

         Mann-Whitney U Test: U-statistic=116738.0, p-value=3.695066672640353e-17
         Wilcoxon Signed-Rank Test: W-statistic=33145.0, p-value=2.275273810723764e-05
         Kruskal-Wallis H Test: H-statistic=70.93572317835647, p-value=3.690546910949811e-17
         Friedman Test: Chi2-statistic=646.7070828331326, p-value=3.709721088881058e-141
```

Conclusion: Thus we have successfully implemented Statistical Hypothesis Test using Scipy and Sci-kit learn

EXPERIMENT NO. 5

Aim: Regression Analysis

a. Perform Logistic Regression to find out relation between variables.

b. Apply regression Model techniques to predict the data on above dataset

Theory: Logistic Regression:

Logistic regression (LR) is a statistical method similar to linear regression since LR finds an equation that predicts an outcome for a binary variable, *Y*, from one or more response variables, *X*. However, unlike linear regression the response variables can be categorical *or* continuous, as the model does not strictly require continuous data. To predict group membership, LR uses the log odds ratio rather than probabilities and an iterative **maximum likelihood** method rather than a least squares to fit the final model. This means the researcher has more freedom when using LR and the method may be more appropriate for nonnormally distributed data or when the samples have unequal covariance matrices. Logistic regression assumes independence among variables, which is not always met in morphoscopic datasets

Logistic regression is a linear classifier, so you'll use a linear function $f(\mathbf{x}) = b_0 + b_1x_1 + \cdots + b_rx_r$, also called the **logit**. The variables b_0, b_1, …, b_r are the **estimators** of the regression coefficients, which are also called the **predicted weights** or just **coefficients**.

The logistic regression function $p(\mathbf{x})$ is the sigmoid function of $f(\mathbf{x})$: $p(\mathbf{x}) = 1 / (1 + \exp(-f(\mathbf{x}))$. As such, it's often close to either 0 or 1. The function $p(\mathbf{x})$ is often interpreted as the predicted probability that the output for a given **x** is equal to 1. Therefore, $1 - p(x)$ is the probability that the output is 0.

Logistic regression determines the best predicted weights b_0, b_1, …, b_r such that the function $p(\mathbf{x})$ is as close as possible to all actual responses y_i, $i = 1, \ldots, n$, where n is the number of observations. The process of calculating the best weights using available observations is called **model training** or **fitting**.

Logistic regression is a fundamental classification technique. It belongs to the group of **linear classifiers** and is somewhat similar to polynomial and **linear regression**. Logistic regression is fast and relatively uncomplicated, and it's convenient for you to interpret the results. Although it's essentially a method for binary classification, it can also be applied to multiclass problems.

Python

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
```

GetData

```
x = np.arange(10).reshape(-1, 1)
y = np.array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
```

```
>>> x
```

```
array([[0],
       [1],
       [2],
       [3],
       [4],
       [5],
       [6],
       [7],
       [8],
       [9]])
>>> y
array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
```

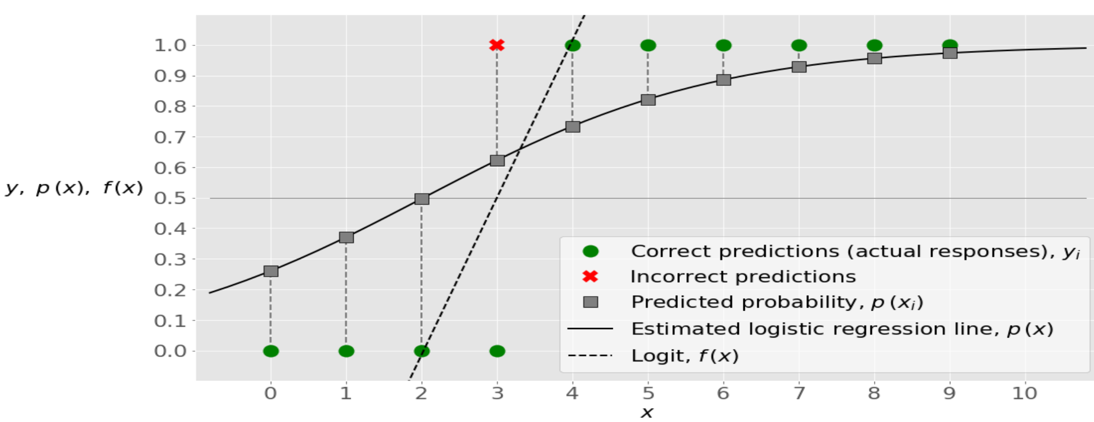
Model Training:

```
model = LogisticRegression(solver='liblinear', random_state=0)

model.fit(x, y)
```

String Representation:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=0, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)

model = LogisticRegression(solver='liblinear', random_state=0).fit(x, y)
```

Output:



Conclusion: Hence we performed logistic regression to find out relation between 2 variables.

EXPERIMENT NO. 6

Aim: Classification modelling
a. Choose classifier for classification problem.
b. Evaluate the performance of classifier.

Theory:

Classification is when the feature to be predicted contains categories of values. Each of these categories is considered as a class into which the predicted value falls and hence has its name, classification

As stated earlier, classification is when the feature to be predicted contains categories of values. Each of these categories is considered as a class into which the predicted value falls. Classification algorithms include:

- Naive Bayes
- Logistic regression
- K-nearest neighbors
- (Kernel) SVM
- Decision tree
- Ensemble learning

Naive Bayes

Naive Bayes applies the Bayes' theorem to calculate the probability of a data point belonging to a particular class. Given the probability of certain related values, the formula to calculate the probability of an event *B*, given event *A* to occur is calculated as follows.

P(B|A) = (P(A|B) * P(B) / P(A))

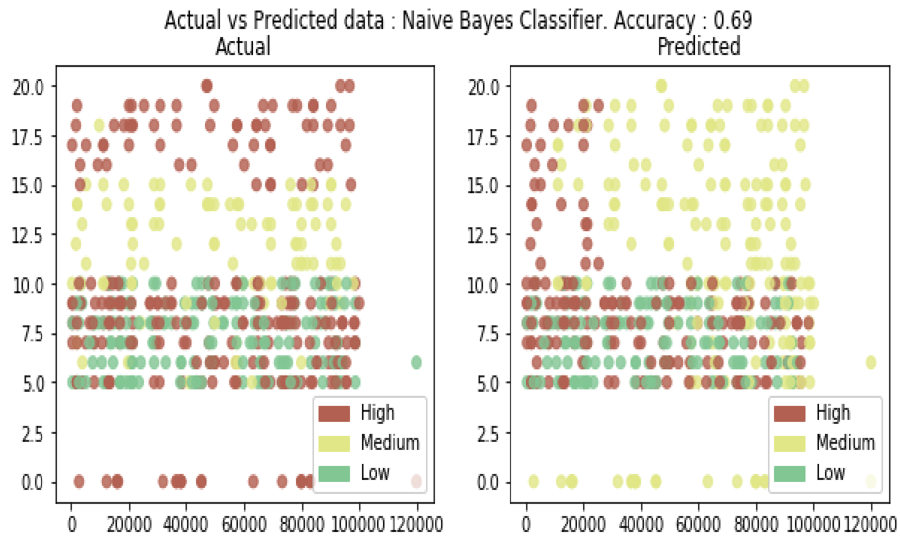
This theory is considered naive, because it assumes that there is no dependency between any of the input features. Even with this not true or naive assumption, the Naive Bayes algorithm has been proven to perform really well in certain use cases like spam filters.

In [18]:
```
from sklearn.naive_bayes import MultinomialNB

model_name = 'Naive Bayes Classifier'

nbClassifier = MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)

nb_model = Pipeline(steps=[('preprocessor', preprocessorForFeatures),('classifier', nbClassifier)])

nb_model.fit(X_train,y_train)

y_pred_nb= nb_model.predict(X_test)
```

While analyzing the predicted output list, we see that the accuracy of the model is at 69%. A comparative chart between the actual and predicted values is also shown.

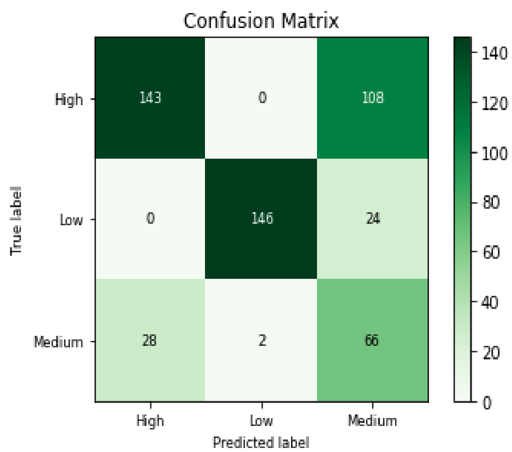
Output:

In [19]:
```
y_test = label_encoder.transform(y_test)
#y_pred_nb = label_encoder.transform(y_pred_nb)
two_d_compare(y_test,y_pred_nb,model_name)
```



In [17]:
```
y_test = label_encoder.inverse_transform(y_test)
y_pred_nb = label_encoder.inverse_transform(y_pred_nb)
model_metrics(y_test,y_pred_nb)
```

```
Decoded values of Churnrisk after applying inverse of label encoder : ['High' 'Low' 'Medium']
```



Conclusion: Thus we have studied and implemented classification modelling.

EXPERIMENT NO. 7

Aim: Clustering

a. Clustering algorithms for unsupervised classification.

b. Plot the cluster data

Theory: Clustering

Cluster analysis, or clustering, is an unsupervised machine learning task.

It involves automatically discovering natural grouping in data. Unlike supervised learning (like predictive modeling), clustering algorithms only interpret the input data and find natural groups or clusters in feature space.

A cluster is often an area of density in the feature space where examples from the domain (observations or rows of data) are closer to the cluster than other clusters. The cluster may have a center (the centroid) that is a sample or a point feature space and may have a boundary or extent.

Clustering can be helpful as a data analysis activity in order to learn more about the problem domain, so-called pattern discovery or knowledge discovery.

For example:

- The phylogenetic tree could be considered the result of a manual clustering analysis.
- Separating normal data from outliers or anomalies may be considered a clustering problem.
- Separating clusters based on their natural behavior is a clustering problem, referred to as market segmentation.

Clustering can also be useful as a type of feature engineering, where existing and new examples can be mapped and labeled as belonging to one of the identified clusters in the data.

Clustering Algorithms

here are many types of clustering algorithms.

Many algorithms use similarity or distance measures between examples in the feature space in an effort to discover dense regions of observations. As such, it is often good practice to scale data prior to using clustering algorithms.

Some clustering algorithms require you to specify or guess at the number of clusters to discover in the data, whereas others require the specification of some minimum distance between observations in which examples may be considered "*close*" or "*connected*."

As such, cluster analysis is an iterative process where subjective evaluation of the identified clusters is fed back into changes to algorithm configuration until a desired or appropriate result is achieved.

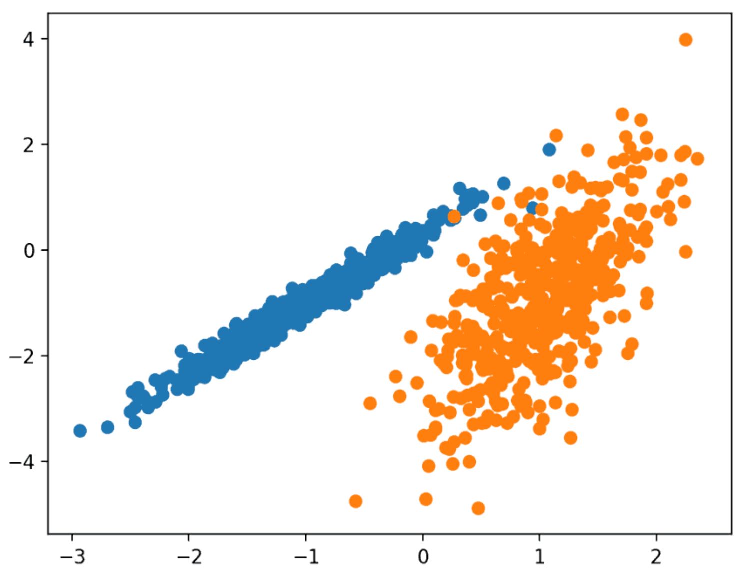
CODE:

```
# synthetic classification dataset
from numpy import where
from sklearn.datasets import make_classification
from matplotlib import pyplot
# define dataset
```

```
X, y = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0,
n_clusters_per_class=1, random_state=4)
# create scatter plot for samples from each class
for class_value in range(2):
    # get row indexes for samples with this class
    row_ix = where(y == class_value)
    # create scatter of these samples
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()
```

Running the example creates the synthetic clustering dataset, then creates a scatter plot of the input data with points colored by class label (idealized clusters).
We can clearly see two distinct groups of data in two dimensions and the hope would be that an automatic clustering algorithm can detect these groupings.

Output:



Conclusion: Hence we studied and implemented clustering algorithm for unsupervised data.

EXPERIMENT NO. 8

Aim: Using any machine learning techniques using available data set to develop a recommendation system

Theory:

Recommendation systems are computer programs that suggest recommendations to users depending on a variety of criteria.
These systems estimate the most likely product that consumers will buy and that they will be interested in. Netflix, Amazon, and other companies use recommender systems to help their users find the right product or movie for them.

There are 3 types of recommendation systems.

1. **Demographic Filtering:** The recommendations are the same for every user. They are generalized, not personalized. These types of systems are behind sections like “Top Trending”.

2. **Content-based Filtering:** These suggest recommendations based on the item metadata (movie, product, song, etc). Here, the main idea is if a user likes an item, then the user will also like items similar to it.

3. **Collaboration-based Filtering:** These systems make recommendations by grouping the users with similar interests. For this system, metadata of the item is not required.

In this project, we are building a Content-based recommendation engine for movies.

Perform Exploratory Data Analysis (EDA) on the data

The dataset contains two CSV files, credits, and movies. The credits file contains all the metadata information about the movie and the movie file contains the information like name and id of the movie, budget, languages in the movie that has been released, etc.

Program:

```
import pandas as pd
path = "./Desktop/TechVidvan/movie_recommendation"
credits_df = pd.read_csv(path + "/tmdb_credits.csv")
movies_df = pd.read_csv(path + "/tmdb_movies.csv")

movies_df.head()
```

Output:

Out[3]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_com
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "In Film Partne
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt Pictures", "id":
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond's past sends him o...	107.376788	[{"name": "C Pictures",
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	The Dark Knight Rises	Following the death of District Attorney Harve...	112.312950	[{"name": "Le Pictures", "id": 9
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	en	John Carter	John Carter is a war-weary, former military ca...	43.926995	[{"name": "Walt Pictures",

Build the Movie Recommender System

The accuracy of predictions made by the recommendation system can be personalized using the "plot/description" of the movie.

```
from ast import literal_eval
features = ["cast", "crew", "keywords", "genres"]
for feature in features:
movies_df[feature] = movies_df[feature].apply(literal_eval)
movies_df[features].head(10)
```

Out[16]:

	cast	crew	keywords	genres
0	[{'cast_id': 242, 'character': 'Jake Sully', '...	[{'credit_id': '52fe48009251416c750aca23', 'de...	[{'id': 1463, 'name': 'culture clash'}, {'id':...	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...
1	[{'cast_id': 4, 'character': 'Captain Jack Spa...	[{'credit_id': '52fe4232c3a36847f800b579', 'de...	[{'id': 270, 'name': 'ocean'}, {'id': 726, 'na...	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...
2	[{'cast_id': 1, 'character': 'James Bond', 'cr...	[{'credit_id': '54805967c3a36829b5002c41', 'de...	[{'id': 470, 'name': 'spy'}, {'id': 818, 'name...	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...
3	[{'cast_id': 2, 'character': 'Bruce Wayne / Ba...	[{'credit_id': '52fe4781c3a36847f81398c3', 'de...	[{'id': 849, 'name': 'dc comics'}, {'id': 853,...	[{'id': 28, 'name': 'Action'}, {'id': 80, 'nam...
4	[{'cast_id': 5, 'character': 'John Carter', 'c...	[{'credit_id': '52fe479ac3a36847f813eaa3', 'de...	[{'id': 818, 'name': 'based on novel'}, {'id':...	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...
5	[{'cast_id': 30, 'character': 'Peter Parker / ...	[{'credit_id': '52fe4252c3a36847f80151a5', 'de...	[{'id': 851, 'name': 'dual identity'}, {'id': ...	[{'id': 14, 'name': 'Fantasy'}, {'id': 28, 'na...
6	[{'cast_id': 34, 'character': 'Flynn Rider (vo...	[{'credit_id': '52fe46db9251416c91062101', 'de...	[{'id': 1562, 'name': 'hostage'}, {'id': 2343,...	[{'id': 16, 'name': 'Animation'}, {'id': 10751...
7	[{'cast_id': 76, 'character': 'Tony Stark / Ir...	[{'credit_id': '55d5f7d4c3a3683e7e0016eb', 'de...	[{'id': 8828, 'name': 'marvel comic'}, {'id': ...	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...
8	[{'cast_id': 3, 'character': 'Harry Potter', '...	[{'credit_id': '52fe4273c3a36847f801fab1', 'de...	[{'id': 616, 'name': 'witch'}, {'id': 2343, 'n...	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...
9	[{'cast_id': 18, 'character': 'Bruce Wayne / B...	[{'credit_id': '553bf23692514135c8002886', 'de...	[{'id': 849, 'name': 'dc comics'}, {'id': 7002...	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...

Get recommendations for the movies

The get_recommendations() function takes the title of the movie and the similarity function as input. It follows the below steps to make recommendations.

```
def get_recommendations(title, cosine_sim=cosine_sim):
idx = indices[title]
similarity_scores = list(enumerate(cosine_sim[idx]))
similarity_scores= sorted(similarity_scores, key=lambda x: x[1], reverse=True)
similarity_scores= sim_scores[1:11]
# (a, b) where a is id of movie, b is similarity_scores
movies_indices = [ind[0] for ind in similarity_scores]
movies = movies_df["title"].iloc[movies_indices]
return movies
print("################ Content Based System #############")
print("Recommendations for The Dark Knight Rises")
print(get_recommendations("The Dark Knight Rises", cosine_sim2))
print()
print("Recommendations for Avengers")
print(get_recommendations("The Avengers", cosine_sim2))
```

```
################ Content Based System #############
Recommendations for The Dark Knight Rises
65              The Dark Knight
119               Batman Begins
4638    Amidst the Devil's Wings
1196                The Prestige
3073           Romeo Is Bleeding
3326              Black November
1503                      Takers
1986                      Faster
303                     Catwoman
747               Gangster Squad
Name: title, dtype: object

Recommendations for Avengers
7               Avengers: Age of Ultron
26           Captain America: Civil War
79                           Iron Man 2
169   Captain America: The First Avenger
174                 The Incredible Hulk
85  Captain America: The Winter Soldier
31                           Iron Man 3
33                X-Men: The Last Stand
68                             Iron Man
94              Guardians of the Galaxy
Name: title, dtype: object
```

C**onclusion :** Hence we studied and implemented Recommendation system for movies using Machine Learning.

EXPERIMENT NO. 9

Aim: Exploratory data analysis using Apache Spark and Pandas

Theory:

We're thrilled to announce that the pandas API will be part of the upcoming Apache Spark™ 3.2 release. pandas is a powerful, flexible library and has grown rapidly to become one of the standard data science libraries. Now pandas users will be able to leverage the pandas API on their existing Spark clusters.

A few years ago, we launched Koalas, an open source project that implements the pandas DataFrame API on top of Spark, which became widely adopted among data scientists. Recently, Koalas was officially merged into PySpark by SPIP: Support pandas API layer on PySpark as part of Project Zen (see also Project Zen: Making Data Science Easier in PySpark from Data + AI Summit 2021).

pandas users will be able scale their workloads with one simple line change in the upcoming Spark 3.2 release:

here are two kinds of variables, continuous and categorical. Each of them has different EDA requirements:

Continuous variables EDA list:

- missing values
- statistic values: mean, min, max, stddev, quantiles
- binning & distribution
- correlation

Now first, Let's load the data. The data I used is from a Kaggle competition, Santander Customer Transaction Prediction.

```
# It's always best to manually write the Schema, I am lazy heredf = (spark.read
.option("inferSchema","true")                .option("header","true")
 .csv("/FileStore/tables/train.csv"))
```

EDA for continuous variables

The built-in function describe() is extremely helpful. It computes count, mean, stddev, min and max for the selected variables. For example:

```
df.select('var_0').describe().show()
```

```
1  df.select('var_0').describe().show()

▸ (1) Spark Jobs
+-------+------------------+
|summary|             var_0|
+-------+------------------+
|  count|            200000|
|   mean|10.679914251999943|
| stddev|3.0400508706688196|
|    min|            0.4084|
|    max|            20.315|
+-------+------------------+
```

However, when you calculate statistic values for multiple variables, this data frame showed will not be neat to check.

Remember that we can use Pandas to do calculations before. We can simply use it to display the result. Here, the describe() function which is built in the spark data frame has done the statistic values calculation. The computed summary table is not large in size. So we can use pandas to display it.

```
df.select('var_0','var_1','var_2','var_3','var_4','var_5','var_6','var_7','var_8','var_9','var_10','var_11','var_12','var_13','var_14').describe().toPandas()
```

```
1 df.select('var_0','var_1','var_2','var_3','var_4','var_5','var_6','var_7','var_8','var_9','var_10','var_11','var_12','var_13','var_14').describe().toPandas()
```

(1) Spark Jobs

Out[13]:

	summary	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	var_9
0	count	200000	200000	200000	200000	200000	200000	200000	200000	200000	200000
1	mean	10.679914251999943	-1.6276216894999918	10.715191851000073	6.796529156999958	11.078333240499944	-5.0653174935	5.408948681500002	16.545849889500087	0.2841618500000177	7.567236361499978
2	stddev	3.0400508706688196	4.050044189955	2.640894191799898	2.0433190163597277	1.623149533936842	7.863266683476755	0.8666072662169012	3.418075578937141	3.332633536717578	1.2350699252999378
3	min	0.4084	-15.0434	2.1171	-0.0402	5.0748	-32.5626	2.3473	5.3497	-10.5055	3.9705
4	max	20.315	10.3768	19.353	13.1883	16.6714	17.2516	8.4477	27.6918	10.1513	11.1506

Get the quantiles:

```
quantile = df.approxQuantile(['var_0'], [0.25, 0.5, 0.75], 0)
quantile_25 = quantile[0][0]
quantile_50 = quantile[0][1]
quantile_75 = quantile[0][2]
print('quantile_25: '+str(quantile_25))
print('quantile_50: '+str(quantile_50))
print('quantile_75: '+str(quantile_75))'''
quantile_25: 8.4537
quantile_50: 10.5247
quantile_75: 12.7582
'''
```

Check the missings:

Introduce two functions to do the filter

```
# where
df.where(col("var_0").isNull()).count()
# filter
df.filter(col("var_0").isNull()).count()
```

Conclusion : Hence we implemented EDA using Apache Spark and pandas

Experiment No. 10

Aim: Batch and Streamed Data Analysis using Spark

Theory:

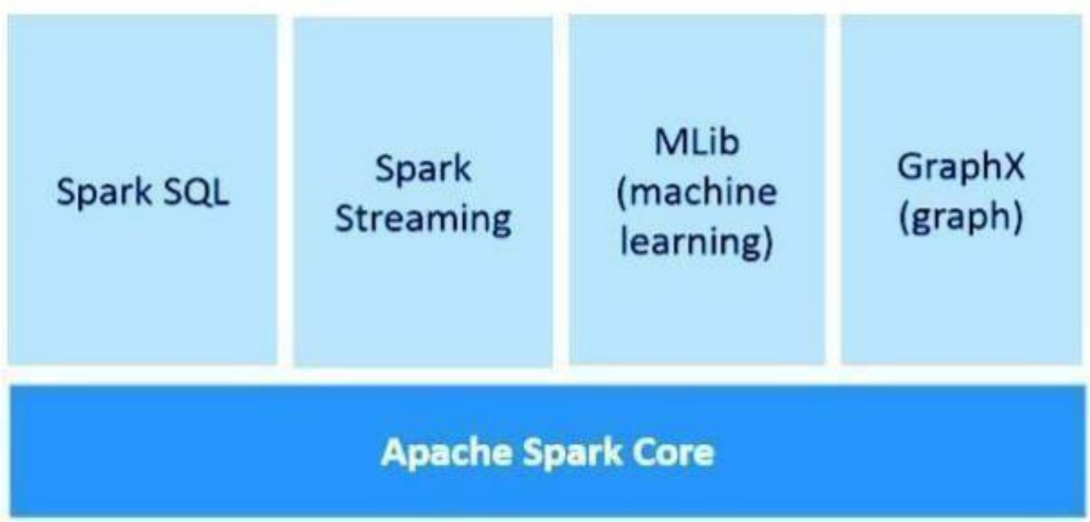
Data Streaming

Data streaming is a way of collecting data continuously in real-time from multiple data sources in the form of data streams. DataStream can be thought of as a table that is continuously being appended.

Data streaming is essential for handling massive amounts of live data. Such data can be from a variety of sources like online transactions, log files, sensors, in-game player activities, etc.

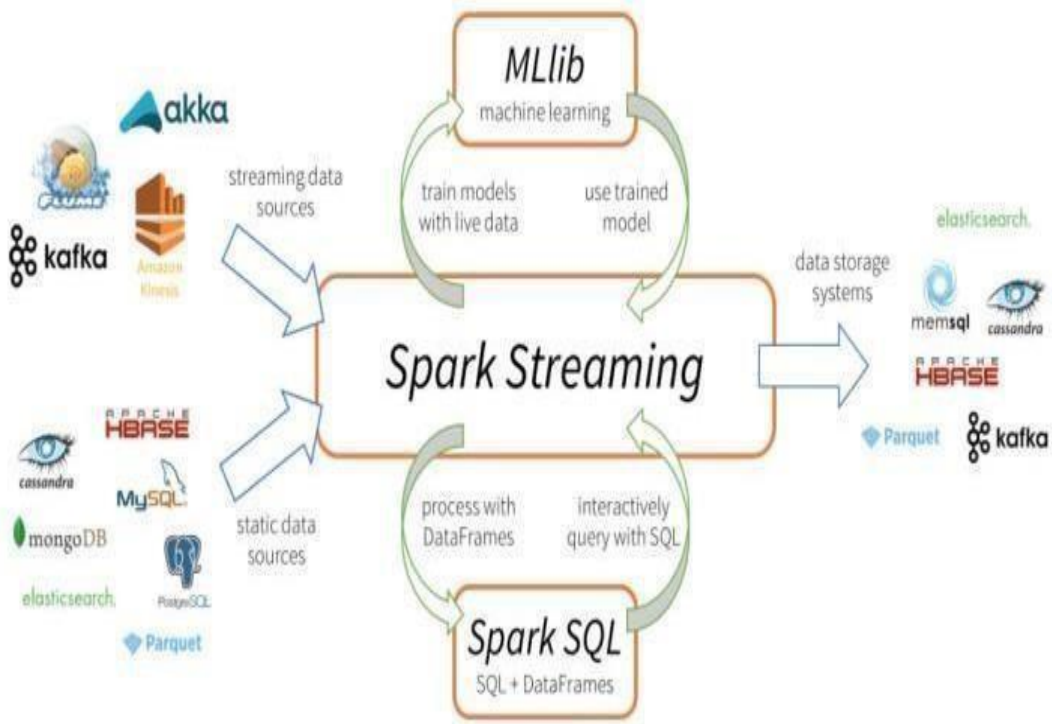
There are various real-time data streaming techniques like Apache Kafka, Spark Streaming, Apache Flume etc. In this post, we will discuss data streaming using Spark Streaming

Spark Streaming



Spark Streaming is an integral part of Spark core API to perform real-time data analytics. It allows us to build a scalable, high- throughput, and fault-tolerant streaming application of live data streams.

Spark Streaming supports the processing of real-time data from various input sources and storing the processed data to various output sinks.



SparkStreaminghas3majorcomponentsasshownintheaboveimage.

- Input data sources: Streaming data sources (like Kafka, Flume, Kinesis, etc.), staticdatasources(likeMySQL,MongoDB,Cassandra,etc.),TCPsockets,Twitter,etc.
- Spark Streaming engine: To process incoming data using various built-in functions, complex algorithms. Also, we can query live streams, apply machine learning using Spark SQL and ML lib respectively.
- Output Sinks: processed data can be stored to file systems, databases (relational and NoSQL), live dashboards etc.

Such unique data processing capabilities, input and output formatting make Spark Streaming more attractive, which further leads to rapid adoption.

Advantages of Spark Streaming

- Unified streaming framework for all data processing tasks (including machine learning, graph processing, SQL operations) on live data streams.
- Dynamic load balancing and better resource management by efficiently balancing the workload across the workers and launching the task in parallel.
- Deeply integrated with advanced processing libraries like Spark SQL, MLlib, GraphX.

Faster recovery from failures by re-launching the failed tasks in parallel on other free nodes.

Code:

```
importre
fromnltk.corpusimportstopwords
stop_words =
set(stopwords.words('english'))from
nltk.stem import
WordNetLemmatizer lemmatizer
=WordNetLemmatizer()
def
        clean_text(
        sentence):
        sentence
        =sentence.l
        ower()
sentence = re.sub("s+"," ", sentence)sentence = re.sub("W"," ", sentence)sentence =
        re.sub(r"httpS+", "",sentence)
        sentence = ' '.join(word for word in sentence.split() if word not
        instop_words)sentence=[lemmatizer.lemmatize(token,"v")for token in
sentence.split()]
        sentence = " ".join(sentence)returnsentence.strip()cleaned_text =
        text_data.map(lambda line:
        clean_text(line))cleaned_text.pprint()strc.start()strc.awaitTermination()
```

Conclusion : Made Spark Streaming, its benefits in real-time data streaming, and a sample application(using TCP sockets) to receive the live data streams and process them as per the requirement

Experiment No. 11

Aim: Implementation of Mini project based on following case study using Data science and Machine learning

Content: DSPS Mini Project: Each group assigned one new case study for this;

A project report must be prepared outlining the following steps:

a)Problem definition, Identifying which data science algorithm is needed

b) Identify and use a standard dataset available for the problem. Some links for machine learning datasets are: KAGGLE , UCI Machine Learning Repository, etc.

c) Implement the machine learning algorithm of choice.

d) Interpret and visualize the results.

e) Provide clearly the comparative analysis of the output.