ASSIGNMENT-2

Student Name: Bharath Kumar Nakka

Student Id: 700744145

Video link:

GIT Hub URL:

**1. Pandas**

1. Read the provided CSV file ‘data.csv’.

A screenshot of a computer

Description automatically generated with medium confidence

* Start by importing the pandas library, a powerful tool for manipulating and analysing data.
* Utilize the read\_csv() function provided by pandas to read the contents of a CSV file called data.csv.
* Assign the resulting DataFrame object to the variable df, enabling easy manipulation and analysis of the data within the file.

2. Show the basic statistical description about the data.

A screenshot of a computer

Description automatically generated

* Apply the describe() function to the pandas DataFrame object df, which provides descriptive statistics of the data.
* Store the output of the describe() function in a new pandas DataFrame object called DataDescription.
* Display the contents of DataDescription by utilizing the print() function to print it to the console.

3. Check if the data has null values.

a. Replace the null values with the mean.

A screenshot of a computer

Description automatically generated

* To determine if there are any null values in the DataFrame object df, the isnull() function is applied. It then employs the any() function with axis=1 to return the rows that contain null values.
* The rows containing null values are displayed in the console using the print() function.
* The null values in the Calories column are replaced with the mean value of the column. This is accomplished by utilizing the fillna() function with inplace=True.
* Once again, it checks for any remaining null values and prints the rows containing null values to the console.

4. Select at least two columns and aggregate the data using: min, max, count, mean

A screenshot of a computer

Description automatically generated with medium confidence

* Create a new DataFrame object by selecting two columns (Duration and Calories) from the original DataFrame object df.
* Apply the agg() function to the new DataFrame object, specifying four aggregate methods (min, max, count, and mean) for calculation.
* Store the output of the aggregation in a new DataFrame object named agg\_df.
* Display the contents of agg\_df by utilizing the print() function to print it to the console.

5. Filter the dataframe to select the rows with calories values between 500 and 1000.

A screenshot of a computer

Description automatically generated

* Use boolean indexing on the DataFrame object df to filter and select rows where the Calories column has a value between 500 and 1000.
* Assign the filtered data to a new DataFrame object named filtered\_df.
* Display the contents of filtered\_df by utilizing the print() function to print it to the console.

6. Filter the dataframe to select the rows with calories values > 500 and pulse < 100.

A screenshot of a computer

Description automatically generated with medium confidence

• Filters the DataFrame object df using boolean indexing to select rows where the Calories column has a value greater than 500 and the Pulse column has a value less than 100.

• Stores the filtered data in a new DataFrame object named filtered\_df.

• Prints the contents of filtered\_df to the console.

7. Create a new “df\_modified” dataframe that contains all the columns from df except for “Maxpulse”.

A screenshot of a computer

Description automatically generated with medium confidence

* Create a new DataFrame object called df\_modified, which includes all the columns from the original DataFrame object df except for the Maxpulse column.
* Use the drop() function with axis=1 to remove the Maxpulse column from the DataFrame object df.
* Assign the modified DataFrame object to the variable df\_modified.
* Display the contents of the modified DataFrame object df\_modified by utilizing the print() function to print it to the console.

8. Delete the “Maxpulse” column from the main df dataframe

A screenshot of a computer

Description automatically generated with medium confidence

* Remove the Maxpulse column from the DataFrame object df using the drop() function with axis=1 and inplace=True.
* Display the contents of the modified DataFrame object df by utilizing the print() function to print it to the console.

9. Convert the datatype of Calories column to int datatype.

A screenshot of a computer

Description automatically generated with medium confidence

* Convert the data type of the Calories column in the DataFrame object df from float to integer using the astype() function.
* Assign the modified DataFrame object back to the df variable to store the updated data.
* Display the contents of the modified DataFrame object df by utilizing the print() function to print it to the console.

10. Using pandas create a scatter plot for the two columns (Duration and Calories).

A screenshot of a computer

Description automatically generated

* Import the pandas library as pd and the matplotlib.pyplot library as plt.
* Load the contents of the CSV file named data.csv into a pandas DataFrame object named df using the read\_csv() function.
* Create a scatter plot of the Duration and Calories columns from the DataFrame object df using the plot() function. Specify kind='scatter', x='Duration', and y='Calories' as parameters.
* Display the plot using the show() function from the plt library.

**2. Scikit-learn**

1. Implement Naïve Bayes method using scikit-learn library.

a. Use the glass dataset available in Link also provided in your assignment.

b. Use train\_test\_split to create training and testing part.

2. Evaluate the model on testing part using score and classification\_report(y\_true, y\_pred)

A screenshot of a computer

Description automatically generated

* Import the necessary libraries, including pandas, train\_test\_split, GaussianNB, and classification\_report.
* Load the glass.csv dataset into a pandas DataFrame object named df.
* Split the dataset into training and testing parts using the train\_test\_split() function.
* Train the Naïve Bayes model using the GaussianNB() function.
* Evaluate the model by calculating the accuracy using the score() method and generating a classification report using the classification\_report() function.
* Print the accuracy and classification report to the console using the print() function.

1. Implement linear SVM method using scikit library

a. Use the glass dataset available in Link also provided in your assignment.

b. Use train\_test\_split to create training and testing part.

2. Evaluate the model on testing part using score and classification\_report(y\_true, y\_pred)

A screenshot of a computer

Description automatically generated

* Import the necessary libraries, including warnings, train\_test\_split, SVC, and classification\_report.
* Load the glass.csv dataset into a pandas DataFrame object named df.
* Split the dataset into training and testing parts using the train\_test\_split() function.
* Train the linear SVM model using SVC() with a linear kernel and a regularization parameter C=1.
* Evaluate the model by calculating the accuracy using the score() method and generating a classification report using the classification\_report() function.
* Print the accuracy and classification report to the console using the print() function.

**Do at least two visualizations to describe or show correlations in the Glass Dataset.**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Import the necessary libraries, including pandas, seaborn, and matplotlib.pyplot.
* Load the glass.csv dataset into a pandas DataFrame object named df.
* Create a bar plot of the glass type distribution using the countplot() function from seaborn.
* Create a scatter plot matrix of selected features using the pairplot() function from seaborn.
* Show the visualizations using the show() function from matplotlib.pyplot.

**Which algorithm you got better accuracy? Can you justify why?**

* The linear SVM model performs better than the Naïve Bayes model on the Glass dataset, with an accuracy of 0.6769 compared to 0.3077. This can be attributed to:
* Naïve Bayes' assumption of feature independence, which may not hold for the Glass dataset.
* SVM's ability to handle complex feature relationships and find an optimal decision boundary.
* The Glass dataset's class imbalance or overlapping classes, which SVM handles better.
* The choice of hyperparameters, train-test split, and other factors also affect model performance, but overall, linear SVM is a better choice for this dataset.