**1. Pandas**

A screenshot of a computer program

Description automatically generated with low confidence

The code imports required libraries for working with tabular data and implementing algorithms, including numpy, pandas, seaborn, matplotlib, scipy, and scikit-learn. It also suppresses warnings. The provided CSV file 'data.csv' is read using pandas into a dataframe called 'df', and the first few rows of the dataframe are displayed using the 'head()' method.

A screenshot of a computer

Description automatically generated with medium confidence

The code calls the 'describe()' method on the dataframe 'df', which returns basic statistical information about the data such as count, mean, standard deviation, minimum value, maximum value, and quartile values for each numerical column in the dataframe.

A screenshot of a computer

Description automatically generated with medium confidence

The code first calls the 'isnull().any()' method on the dataframe 'df', which checks if there are any null values in the dataframe and returns a boolean value for each column indicating if it contains null values. Then, it fills the null values with the mean using the 'fillna()' method and the 'mean()' function, and checks again for null values using 'isnull().any()'.

Finally, the code selects two columns ('Maxpulse' and 'Calories') and aggregates the data using the 'agg()' method. The aggregation operations used are 'min', 'max', 'count', and 'mean'. The resulting dataframe shows the minimum, maximum, count, and mean values for each selected column.

A screenshot of a computer

Description automatically generated with medium confidence

The code filters the dataframe 'df' using the 'loc[]' method to select rows where the 'Calories' column has values between 500 and 1000 (exclusive). The resulting dataframe contains only the rows that satisfy the condition.

A screenshot of a computer

Description automatically generated

The code filters the dataframe 'df' to select rows where the 'Calories' column has values greater than 500 and the 'Pulse' column has values less than 100, using the 'loc[]' method. The resulting dataframe contains only the rows that satisfy the condition.

Then, the code creates a new dataframe 'df\_modified' by selecting all columns from the original dataframe 'df' except for 'Maxpulse'. This is done by indexing the dataframe with a list of the desired column names. The resulting dataframe contains only the columns 'Duration', 'Pulse', and 'Calories', and is assigned to the variable 'df\_modified'.

A screenshot of a computer

Description automatically generated with medium confidence

The code deletes the 'Maxpulse' column from the dataframe 'df' using the 'del' statement.

Then, the code converts the datatype of the 'Calories' column from float to integer using the 'astype()' method and the numpy 'int64' datatype. The resulting dataframe 'df' contains the modified 'Calories' column with integer datatype.

Finally, the 'dtypes' attribute is used to display the datatypes of all columns in the dataframe 'df'.

A screen shot of a computer screen

Description automatically generated with low confidence

The code creates a scatter plot of the dataframe 'df' with the 'Duration' column as the x-axis and the 'Calories' column as the y-axis, using the 'plot.scatter()' method. Each point in the scatter plot represents the relationship between the duration of exercise and the corresponding number of calories burned. The points are colored blue.

**2. Scikit-learn**

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

This code loads the 'glass.csv' dataset into a Pandas dataframe and performs correlation analysis on the dataset. It then defines the features and target variables for classification, splits the dataset into training and validation sets using the train\_test\_split method from scikit-learn. The code then trains and tests two classifiers - Gaussian Naive Bayes and Linear Support Vector Classifier - on the dataset and prints out a classification report, confusion matrix, and accuracy score for each classifier.

**Conclusion:**

The code above is performing two types of classification analysis: Naive Bayes and Linear SVM. Both methods are being applied on a dataset of glass compositions to predict the type of glass. After splitting the data into training and validation sets, the classifiers are trained on the training set and then used to make predictions on the validation set.

The accuracy score is then calculated for both classifiers. The Naive Bayes classifier performed better with an accuracy score of 0.837, while the Linear SVM had an accuracy score of 0.674.

The reason for Naive Bayes performing better could be attributed to the probabilistic nature of the algorithm. It is well-suited for problems involving probabilities, while the Linear SVM relies on linear regression concepts. However, the performance of Linear SVM can be improved with larger amounts of data for training and testing.

Therefore, based on this dataset and the amount of data available, Naive Bayes is the better classifier for predicting the type of glass.

GitHub URL:

<https://github.com/nxk78010/prog_assign2.git>