STUDENT ID:  STUDENT NAME:

Data Analysis Assignment

Contents

[Task 1: Reading a Single Column from a CSV File 1](#_Toc184332323)

[Task 2: Reading CSV Data into Memory 1](#_Toc184332324)

[Task 3: Calculating the Pearson Correlation Coefficient 2](#_Toc184332325)

[Task 4: Generating Pearson Correlation Coefficients for Multiple Columns 3](#_Toc184332326)

[Task 5: Displaying a Custom Table of Correlations 3](#_Toc184332327)

[Task 6: Exploring Patterns and Trends 4](#_Toc184332328)

[Task 8: Hypothesis Testing 5](#_Toc184332329)

[Summary and Implications 5](#_Toc184332330)

# Task 1: Reading a Single Column from a CSV File

Focusing on related variables for research is always a routine procedure in data analysis. To this end, we implemented a function that enables one to retrieve one column from a CSV file. This focused approach has some advantages; for example, it requires less memory than other methods and is beneficial for processing large datasets when moving through steps in a data processing pipeline. Rather than loading all the data set, this method focuses on a specific area of interest only.

This is very helpful in cases where the scrutiny of the analysis entails the isolation of a single variable like in preparing the data for visualization or even for statistical analysis. The implementation is as follows:

def read\_column(file\_path, column\_name):

import csv

with open(file\_path, 'r') as file:

reader = csv.DictReader(file)

return [row[column\_name] for row in reader]

Consequently, using Python’s CSV module makes it simple for the function to read the specified column. To illustrate, one only needs to enter the column name and the file path, making it easier to extract reasonable details.

# Task 2: Reading CSV Data into Memory

Data manipulation often requires working with all the data at once, most of the time for operations with variables and their interconnections, or cleaning up the data. Loading a CSV file on memory as a list of dictionaries was achieved by developing a function that would do the job. Implies, this format lets us keep an organized data layout to navigate rows and columns programmatically and optimally.

The ability to load the dataset entirely into memory serves several purposes:

* It makes a process that includes anything from filtering data, grouping them or aggregating them.
* It serves as a base for other more complex tasks like data visualization or even the application of machine learning algorithms.

**Here's the implementation:**

def read\_csv(file\_path):

import csv

with open(file\_path, 'r') as file:

reader = csv.DictReader(file)

return [row for row in reader]

I performed this function as a key preparation step that allows for easy access to the dataset for iterative and batch use in later tasks.

# Task 3: Calculating the Pearson Correlation Coefficient

This paper emphasizes the Pearson Correlation Coefficient as one of the most important statistical methods of measuring the strength and direction of the association between two variables. Its significance is based on applications in pattern recognition and consequences forecasting in different fields, investments, learning experiences, and treatments.

For this assignment, a function was created which can calculate the Pearson Correlation Coefficient. The formula is mathematically expressed as:

****

**The implemented function is as follows:**

def pearson\_correlation(list\_x, list\_y):

import math

n = len(list\_x)

mean\_x = sum(list\_x) / n

mean\_y = sum(list\_y) / n

numerator = sum((x - mean\_x) \* (y - mean\_y) for x, y in zip(list\_x, list\_y))

denominator = math.sqrt(sum((x - mean\_x) \*\* 2 for x in list\_x) \* sum((y - mean\_y) \*\* 2 for y in list\_y))

return numerator/denominator if denominator != 0 else 0

as numerator/denominator if the denominator is not equal to 0 otherwise 0

The idea of this function is most relevant when it comes to continuous measurements in the dataset. It computes for coefficient for two lists, this way one is capable of determining the degree of relationship between the two lists and more so can come up with relevant conclusions.

# Task 4: Generating Pearson Correlation Coefficients for Multiple Columns

For the matrices with more than one variable to be analyzed, a function to calculate the Pearson Coefficient between a target variable and all other columns of the matrix was created. This automated process is particularly useful for filtering variables with the possibility of carrying prediction influence or having a high correlation with the target.

def calculate\_all\_correlations(data, target\_column):

correlations = {}

for col in data[0].keys():

if col != target\_column:

correlations[col] = pearson\_correlation(

[float(row[col]) for row in data if row[col] != ''],

[float(row[target\_column]) for row in data if row[target\_column] != '']

)

return correlations

This function enables the iteration through all the columns and ignores the targeted variable which helps in enabling the visualization of the relations from within the data set. It is especially indispensable in the EDA to decide what variables should be examined more thoroughly.

# Task 5: Displaying a Custom Table of Correlations

Using raw correlation data is not always convenient because of such things as large amounts of data that may be overwhelming to work on. To work around this, a custom tabular format was introduced to present correlation results clearly and easily read format. This type of modeling as a result increases interpretability and helps in explaining the results to the stakeholders.

def print\_correlation\_table(correlations):

print(f"{'Column':<20}{'Correlation':>10}")

print('-' \* 30)

For col, corr in correlations.items():

print(f"{col:<20}{corr:>10.4f}")

Third, this form of tabular presentation makes it possible for analysts to pick out relevant and significant patterns within a shorter time and thus promote proper decision-making.

# Task 6: Exploring Patterns and Trends

One of the data analysis considerations is to recognise an existing pattern and trends. While accomplishing this task, statistical analysis of the data and spectrum was carried out to correlate between variables. One of the interesting details revealed was a highly positive linear relationship between click\_events and score. This insight indicates that frequency, as a measure of engagement, determined from click events, affects course performance.

Understanding such patterns can inform practice in some way, for example, to increase the effectiveness of engagement to get better learning results. These findings also provide the basis for hypothesis testing.

**Task 7: Detecting and Removing Outliers**

There are always points that are unusually high or low compared to other points and such points often warp the results and give biased conclusions. The next step was to remove the outliers using the IQR method on the click\_events variable to have better-quality data.

The function for outlier detection and removal is:

def remove\_outliers(data, column):

import numpy as np

values = [float(row[column]) for row in data if row[column] != '']

q1 = np.percentile(values, 25)

q3 = np.percentile(values, 75)

iqr = q3 - q1

lower\_bound = q1 - 1.5 \* IQR

upper\_bound = q3 + 1.5 \* IQR

return [row for row in data if lower\_bound <= float(row[column]) <= upper\_bound]

It adjusts them to a specified range so that the effect of outliers is reduced this method enhances the dataset.

# Task 8: Hypothesis Testing

**Hypothesis Statement:**

**H₀ (Null Hypothesis):** The number of click events does not correlate with students’ scores.

**H₁ (Alternative Hypothesis):** The more click events, the higher the student scores, as evidenced by the correlation results.

To test the hypothesis, linear regression analysis of the cleaned data was performed. This was done by analyzing the p-value that tests for the correlation between click\_events and a score at 0.05 level of significance.

**Results:** The p-value turned out less than 0.05, leaving the null hypothesis false, as found in the research analysis. This kind of result offers beneficial statistical proof that click events indeed affect the scores of students.

**Conclusion:** The findings underscore the place of engagement measures in education accomplishment. Using them, educational institutions will be able to establish the best approaches that shall be taken to improve students’ results.

## Summary and Implications

This evaluation shows that there is value in a formal approach to analysing and processing the data. Every task plays a role in the global workflow of a data scientist, starting with data preprocessing and going all the way up to the powerful statistical largeness. Pattern and relationship identification allows for the formation of decisions based on information, while valid methods such as outlier identification and hypothesis testing help attain the output.

The revealed methods are not only focused on educational datasets, but the methods can be used in other domains like health care, finance, and marketing domains. Relative to this, future work could extend the range of analysis to other variables or use more sophisticated methods such as machine learning to identify other patterns.