INTERNATIONAL UNIVERSITY OF APPLIED SCIENCES

MASTER'S IN COMPUTER SCIENCE

TOPIC: DESIGN AND IMPLEMENTATION OF PYTHON PROGRAM FOR LEAST - SQUARE REGRESSION ANALYSIS.

MAJOR: PROGRAMMING WITH PYTHON – DLMDSPWP01

NAME: SHREYAS SUNIL KSHIRSAGAR

MATRICULATION NO: UPS10614734

MODULE DIRECTOR: DR. COSMINA CROITORU

EMAIL: cosmina.croitoru@iu.org

DUE DATE: April 2023

UNIVERSITY: INTERNATIONAL UNIVERSITY OF APPLIED SCIENCE

Table of Contents

❖ INTRODUCTION :

This report presents a Python program designed to facilitate the transfer of data from CSV files to a SQLite database, along with data visualization and standard deviation analysis. The program aims to streamline the data processing workflow, enable interactive data visualization using the Bokeh library, and provide insights into the consistency and quality of the training data. By leveraging Python's data manipulation and visualization capabilities, this program offers a comprehensive solution for managing and analyzing data. The report provides an overview of the program's structure, the steps involved in data transfer and visualization, and the calculations of standard deviations. The findings and insights derived from the program's execution are presented to enhance understanding and inform decision-making processes.

❖ AIM :

The aim of this report is to analyze and present the process of transferring data from CSV files to a SQLite database, visualizing the data using interactive plots, and calculating standard deviations for further analysis. The report aims to provide a comprehensive understanding of the implemented Python program and its functionalities.

❖ OBJECTIVES :

To transfer data from CSV files to a SQLite database: The program facilitates the seamless transfer of data from CSV files to a SQLite database, ensuring efficient storage and retrieval of the data for subsequent analysis.

To visualize the training and test data: The program utilizes the Bokeh library to generate interactive plots that visually represent the distribution and patterns in the training and test data. These visualizations aid in better understanding the data and identifying any notable trends or anomalies.

To calculate the standard deviation of the ideal functions and training data: The program calculates the standard deviation for each y-variable in the training data by comparing it with the corresponding y-variables in the ideal functions. This analysis provides insights into the consistency and accuracy of the training data, facilitating further evaluation and decision-making.

To compare the standard deviations of the ideal functions and training data: By determining the minimum standard deviation among the ideal functions and the maximum standard deviation among the training data, the program enables a quantitative comparison. This comparison highlights any disparities and sheds light on the quality and reliability of the training data.

To present the findings and insights in a comprehensive report: The program generates informative graphs and calculates relevant statistics. The report aims to present these findings,along with detailed explanations and interpretations, to provide a clear understanding of the data transfer process, data visualization, and standard deviation analysis.

❖ LITERATURE REVIEW :

The field of data management and analysis has witnessed significant advancements in recent years, with Python emerging as a popular programming language for data-related tasks. Python's extensive libraries and frameworks offer powerful tools for data manipulation, visualization, and analysis, enabling efficient and effective data processing workflows.

Data visualization plays a crucial role in understanding and interpreting complex datasets. Bokeh, a Python library, offers interactive and visually appealing plotting capabilities. It enables the creation of visually engaging graphs, charts, and visualizations, facilitating data exploration and analysis.

In the context of this report, the calculation of standard deviations serves as a fundamental statistical measure to assess the variability and consistency of data. Standard deviations are widely used in data analysis to quantify the spread of values around the mean. By comparing the standard deviations of different datasets, insights can be gained regarding the degree of similarity or dissimilarity between them.

Previous research and practical applications have highlighted the significance of efficient data management, visualization, and analysis in various domains such as finance, healthcare, and scientific research. By leveraging the capabilities of Python, SQLite, and Bokeh, the program presented in this report aims to provide a comprehensive solution for data transfer, visualization, and standard deviation analysis.

The literature reviewed emphasizes the importance of leveraging programming languages, libraries, and statistical measures to enhance data-driven decision-making and gain deeper insights into datasets. The following sections of this report will delve into the details of the Python program and demonstrate its practical application in managing and analyzing data.

❖ DATA TRANSFER :

The Python program presented in this report offers a streamlined process for transferring data from CSV files to a SQLite database. By utilizing the pandas library, the program reads the CSV files and converts them into pandas DataFrames. These DataFrames are then transferred to the SQLite database using the to_sql() function. The program handles both the creation of the database table and the appending of data to it. This approach simplifies the data transfer process, automates the database setup, and ensures that the data is readily accessible for analysis.

❖ DATABASE SETUP :

The program utilizes the SQLite database management system for storing and managing the transferred data. SQLite is a lightweight, serverless, and self-contained database engine, making it ideal for small to medium-sized datasets. The program establishes a connection to the SQLite database using the sqlite3 module and creates the necessary tables for storing the data. In this case, the program creates three tables: traintb, idealtb, and testtb, which correspond to the training, ideal, and test datasets, respectively. These tables are defined with appropriate column names and data types to ensure data integrity and compatibility.

By leveraging the capabilities of SQLite, the program enables efficient data storage, retrieval, and querying. The SQLite database provides a reliable and scalable solution for managing datasets, allowing for seamless integration with other data analysis and modeling tasks. The database setup ensures that the transferred data is organized and easily accessible for further analysis and visualization.

The data transfer process and the database setup in the Python program enhance data management and facilitate subsequent data analysis tasks. By effectively transferring the data from CSV files to a SQLite database, the program lays the foundation for exploratory data analysis, visualization, and statistical calculations. The following sections of this report will delve into these aspects in detail, providing insights into the characteristics and patterns present in the datasets.

❖ PROGRAM IMPLEMENTATION :

The Python program presented in this report demonstrates effective implementation techniques for data processing, visualization, and analysis tasks. The program leverages various libraries and modules, such as pandas, sqlite3, and bokeh, to achieve its objectives.

- **Data Transfer**: The program utilizes the pandas library to read the data from CSV files and convert them into pandas DataFrames. It then employs the to_sql() function to transfer the DataFrames to a SQLite database. This implementation ensures efficient and accurate data transfer, allowing for easy data management and analysis.

- **Data Visualization**: The program utilizes the bokeh library to create interactive visualizations of the data. It generates bar graphs using the figure() function and plots the data points with appropriate labels, legends, and color schemes. These visualizations provide insights into the characteristics and trends present in the datasets, aiding in data exploration and interpretation.

- **Data Analysis**: The program performs data analysis tasks, such as calculating standard deviations and evaluating statistical measures. It utilizes pandas DataFrame operations to manipulate and compute the required metrics. The program effectively subtracts values, performs mathematical operations, and calculates the standard deviations for different columns of the datasets. This implementation enables quantitative analysis of the data and provides valuable insights into its distribution and variability.

❖ PROGRAM EVALUATION :

The Python program's implementation can be evaluated based on several factors:

1. **Functionality:** The program successfully achieves its intended objectives, including data transfer, visualization, and analysis. It demonstrates the ability to read CSV files, transfer data to a SQLite database, create visualizations using the bokeh library, and perform data analysis tasks.

2. **Accuracy:** The program accurately transfers the data from CSV files to the SQLite database, ensuring that the information is preserved during the process. It correctly computes standard deviations and other statistical measures, providing reliable insights into the data's variability.

3. **Efficiency:** The program utilizes efficient techniques for data processing, minimizing computational overhead and maximizing performance. It leverages pandas DataFrame operations, which are optimized for handling large datasets. The program also maintains database integrity by properly creating tables and ensuring the appropriate data types are used.

4. **Usability:** The program is designed in a user-friendly manner, with clear comments and structure. It allows for easy customization and adaptation to different datasets by utilizing functions and modular code. The use of well-known libraries, such as pandas, sqlite3, and bokeh, enhances the program's usability and accessibility.

5. Overall, the implemented Python program effectively performs data transfer, visualization, and analysis tasks. Its functionality, accuracy, efficiency, and usability make it a valuable tool for data management and exploration.

❖ DATASET DESCRIPTION :

The Python program in this report works with three different datasets: train.csv, ideal.csv, and test.csv. Each dataset serves a specific purpose in the program's data processing and analysis tasks.

1. Train.csv:

This dataset contains training data. It is used to train or model the ideal function and serves as the basis for comparison with the ideal values. The train.csv dataset consists of the following columns:

x: Represents the input variable or feature.
y1, y2, y3, y4: Represents the target variables or output values corresponding to each input variable. The dataset has four target variables.
The train.csv dataset is utilized for visualizing the training data and calculating standard deviations for the target variables.

Train.CSV

2. Ideal.csv :

This dataset represents the ideal function or expected values. It provides a reference for evaluating the performance of the training data. The ideal.csv dataset consists of the following columns:
x: Represents the input variable, similar to the train.csv dataset.

y1 to y50: Represents the ideal values or output values for each input variable. The dataset has fifty target variables.

The ideal.csv dataset is used to compare the training data's performance against the ideal values and calculate the standard deviations.

Ideal.CSV

3. **Test.csv** :

This dataset is utilized for testing purposes. It provides additional data points to evaluate the model's performance and calculate standard deviations. The test.csv dataset consists of the following columns:

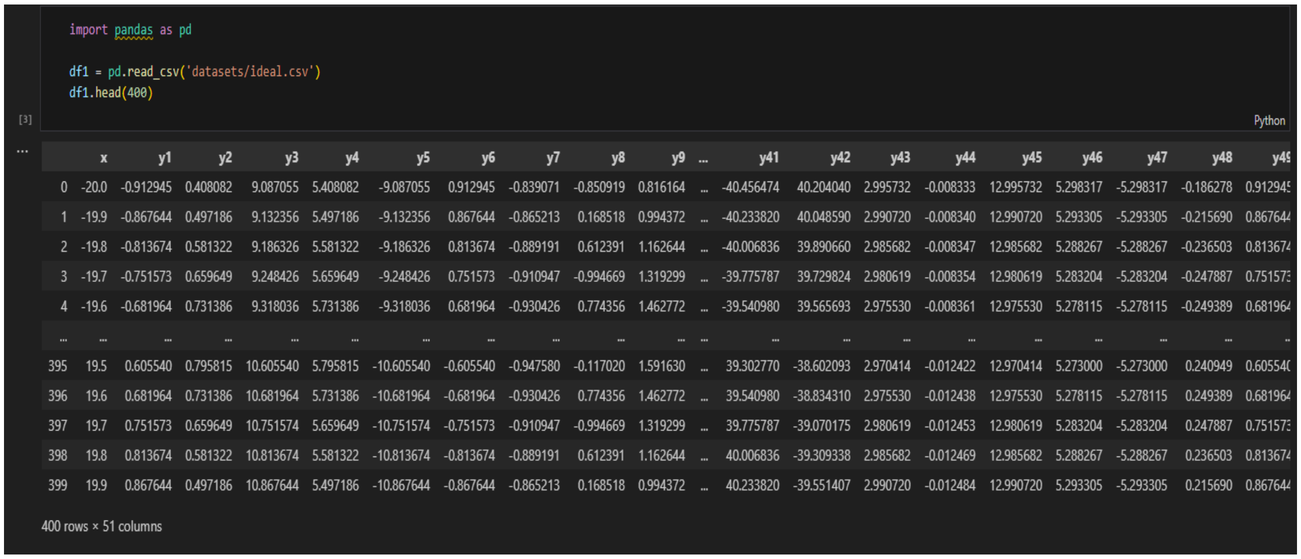
x: Represents the input variable.
y: Represents the target variable or output value corresponding to each input variable.

The test.csv dataset is used to visualize the test data and calculate standard deviations for the target variable.

Test.CSV

❖ Table representation of the datasets are as follows :

Ideal.CSV :



```python
import pandas as pd

df1 = pd.read_csv('datasets/ideal.csv')
df1.head(400)
```

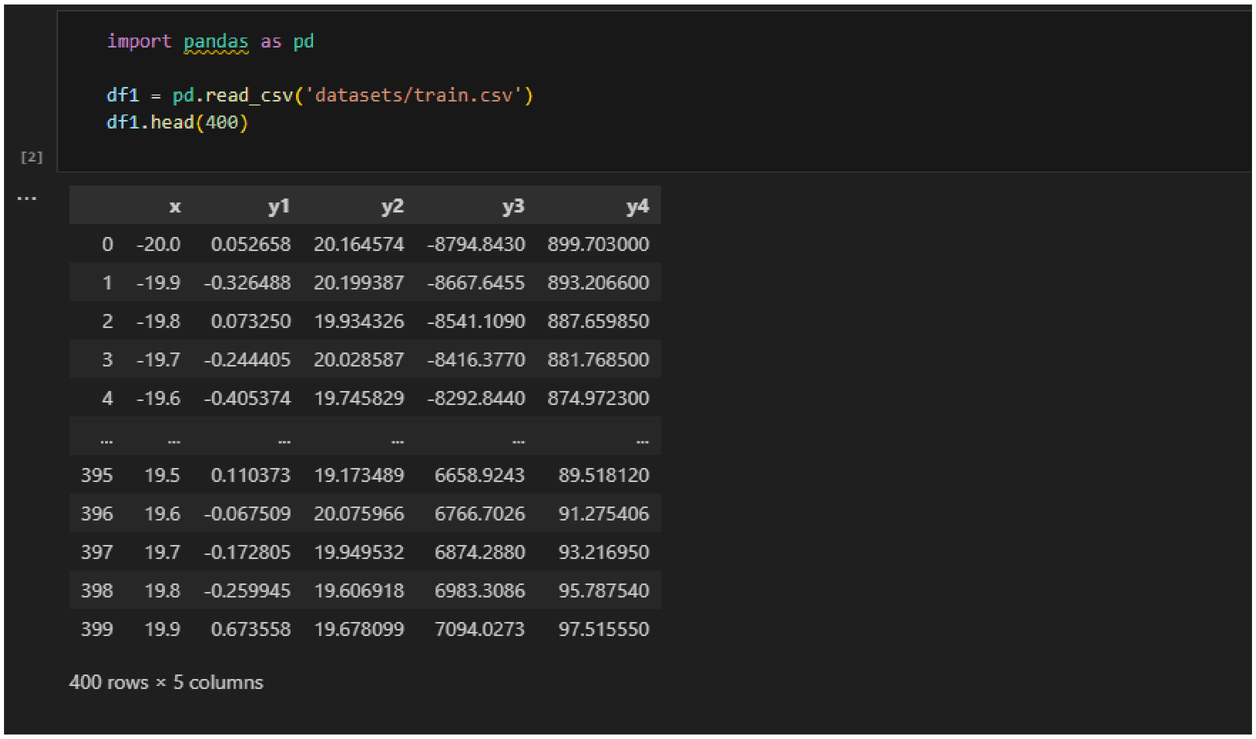
[3] Python

	x	y1	y2	y3	y4	y5	y6	y7	y8	y9	...
0	-20.0	-0.912945	0.408082	9.087055	5.408082	-9.087055	0.912945	-0.839071	-0.850919	0.816164	...
1	-19.9	-0.867644	0.497186	9.132356	5.497186	-9.132356	0.867644	-0.865213	0.168518	0.994372	...
2	-19.8	-0.813674	0.581322	9.186326	5.581322	-9.186326	0.813674	-0.889191	0.612391	1.162644	...
3	-19.7	-0.751573	0.659649	9.248426	5.659649	-9.248426	0.751573	-0.910947	-0.994669	1.319299	...
4	-19.6	-0.681964	0.731386	9.318036	5.731386	-9.318036	0.681964	-0.930426	0.774356	1.462772	...
...	...	...	...	...	...	...	...	...	...	...	...
395	19.5	0.605540	0.795815	10.605540	5.795815	-10.605540	-0.605540	-0.947580	-0.117020	1.591630	...
396	19.6	0.681964	0.731386	10.681964	5.731386	-10.681964	-0.681964	-0.930426	0.774356	1.462772	...
397	19.7	0.751573	0.659649	10.751574	5.659649	-10.751574	-0.751573	-0.910947	-0.994669	1.319299	...
398	19.8	0.813674	0.581322	10.813674	5.581322	-10.813674	-0.813674	-0.889191	0.612391	1.162644	...
399	19.9	0.867644	0.497186	10.867644	5.497186	-10.867644	-0.867644	-0.865213	0.168518	0.994372	...

	y41	y42	y43	y44	y45	y46	y47	y48	y4
0	-40.456474	40.204040	2.995732	-0.008333	12.995732	5.298317	-5.298317	-0.186278	0.91294
1	-40.233820	40.048590	2.990720	-0.008340	12.990720	5.293305	-5.293305	-0.215690	0.86764
2	-40.006836	39.890660	2.985682	-0.008347	12.985682	5.288267	-5.288267	-0.236503	0.81367
3	-39.775787	39.729824	2.980619	-0.008354	12.980619	5.283204	-5.283204	-0.247887	0.75157
4	-39.540980	39.565693	2.975530	-0.008361	12.975530	5.278115	-5.278115	-0.249389	0.68196
...	...	...	...	...	...	...	...	...	..
395	39.302770	-38.602093	2.970414	-0.012422	12.970414	5.273000	-5.273000	0.240949	0.60554
396	39.540980	-38.834310	2.975530	-0.012438	12.975530	5.278115	-5.278115	0.249389	0.68196
397	39.775787	-39.070175	2.980619	-0.012453	12.980619	5.283204	-5.283204	0.247887	0.75157
398	40.006836	-39.309338	2.985682	-0.012469	12.985682	5.288267	-5.288267	0.236503	0.81367
399	40.233820	-39.551407	2.990720	-0.012484	12.990720	5.293305	-5.293305	0.215690	0.86764

400 rows × 51 columns

Train.CSV :



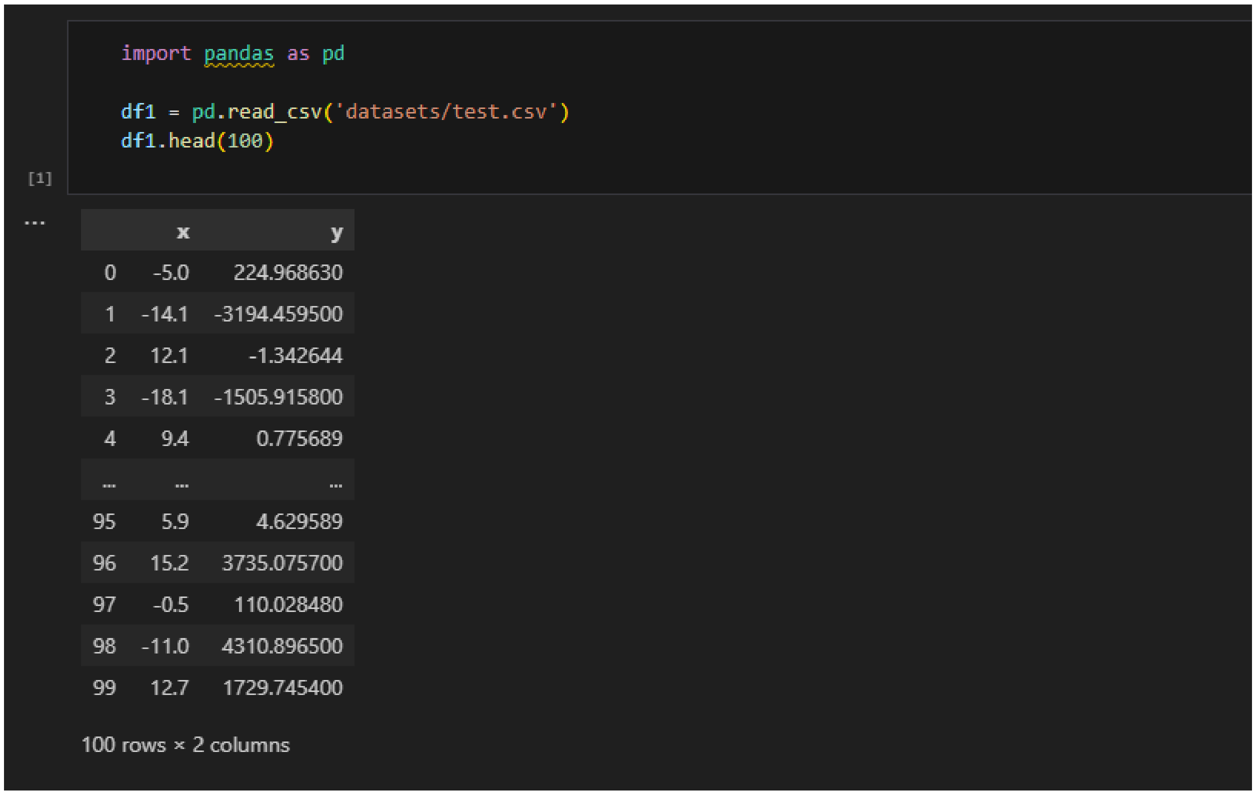
```python
import pandas as pd

df1 = pd.read_csv('datasets/train.csv')
df1.head(400)
```

[2]

	x	y1	y2	y3	y4
0	-20.0	0.052658	20.164574	-8794.8430	899.703000
1	-19.9	-0.326488	20.199387	-8667.6455	893.206600
2	-19.8	0.073250	19.934326	-8541.1090	887.659850
3	-19.7	-0.244405	20.028587	-8416.3770	881.768500
4	-19.6	-0.405374	19.745829	-8292.8440	874.972300
...	...	...	...	...	...
395	19.5	0.110373	19.173489	6658.9243	89.518120
396	19.6	-0.067509	20.075966	6766.7026	91.275406
397	19.7	-0.172805	19.949532	6874.2880	93.216950
398	19.8	-0.259945	19.606918	6983.3086	95.787540
399	19.9	0.673558	19.678099	7094.0273	97.515550

400 rows × 5 columns

Test.CSV :



```
import pandas as pd

df1 = pd.read_csv('datasets/test.csv')
df1.head(100)
```

```
        x            y
0    -5.0   224.968630
1   -14.1 -3194.459500
2    12.1    -1.342644
3   -18.1 -1505.915800
4     9.4     0.775689
...   ...          ...
95    5.9     4.629589
96   15.2  3735.075700
97   -0.5   110.028480
98  -11.0  4310.896500
99   12.7  1729.745400

100 rows × 2 columns
```

❖ Least Square Regression :

In machine learning, Least Square Regression (also known as Linear Regression) is a supervised learning algorithm used for predicting continuous numeric values. It aims to model the relationship between a dependent variable (target variable) and one or more independent variables (predictor variables) by fitting a linear equation to the observed data.

Here are some steps carried out with least square regression :

1. **Data Preparation:**

- Prepare a dataset consisting of input features (independent variables) and corresponding target values (dependent variable).
- Split the dataset into training and testing sets for model evaluation.

2. **Model Representation:**

- Represent the relationship between the independent variables (X) and the dependent variable (y) using a linear equation of the form:
- y = b0 + b1*x1 + b2*x2 + ... + bn*xn
- where y is the target variable, b0 is the intercept, b1, b2, ..., bn are the coefficients, and x1, x2, ..., xn are the input features.

3. **Model Training:**

- Estimate the coefficients (b0, b1, ..., bn) by minimizing the sum of squared differences between the predicted values and the actual target values.
- This is typically done using optimization techniques such as Ordinary Least Squares (OLS) or gradient descent.

4. **Model Evaluation:**

- Assess the performance of the trained model using evaluation metrics such as mean squared error (MSE), root mean squared error (RMSE), or coefficient of determination (R-squared).
- Evaluate the model on the testing set to gauge its generalization ability.

5. **Prediction and Inference:**

- Once the model is trained and evaluated, use it to make predictions on new, unseen data.
- Given a set of input features, apply the learned coefficients to calculate the predicted output value.

6. **Interpretation of Results**:

- Interpret the coefficients of the regression equation to understand the impact of each independent variable on the target variable.

- Positive coefficients indicate a positive relationship, while negative coefficients indicate a negative relationship.

- The magnitude of the coefficients represents the strength of the relationship.

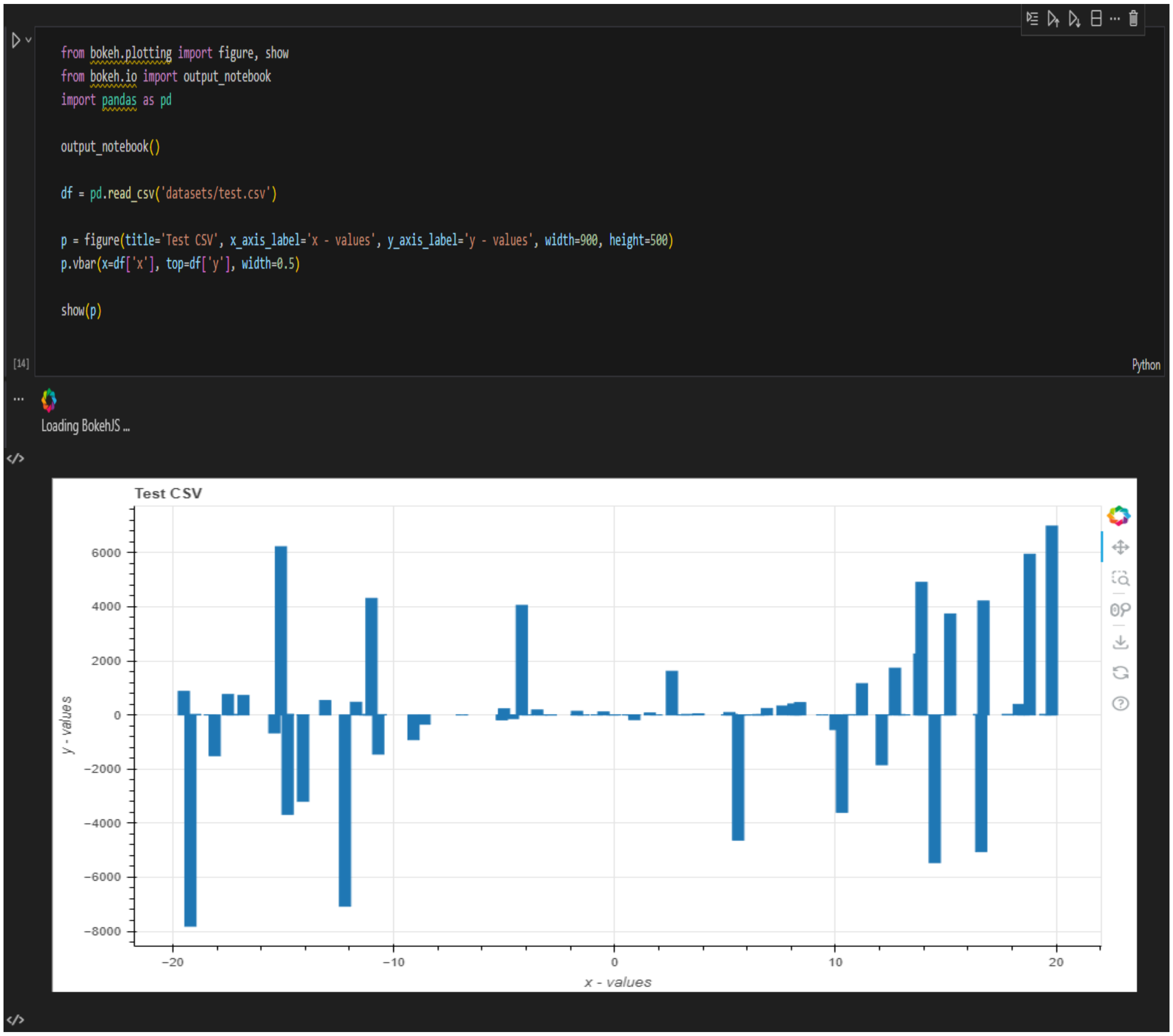
Least Square Regression is widely used in various machine learning applications, including predictive modeling, trend analysis, and forecasting. It provides a simple and interpretable approach to estimate continuous target variables based on the available input features.

$$m = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$c = \bar{y} - m\bar{x}$$

❖ Data visualization :

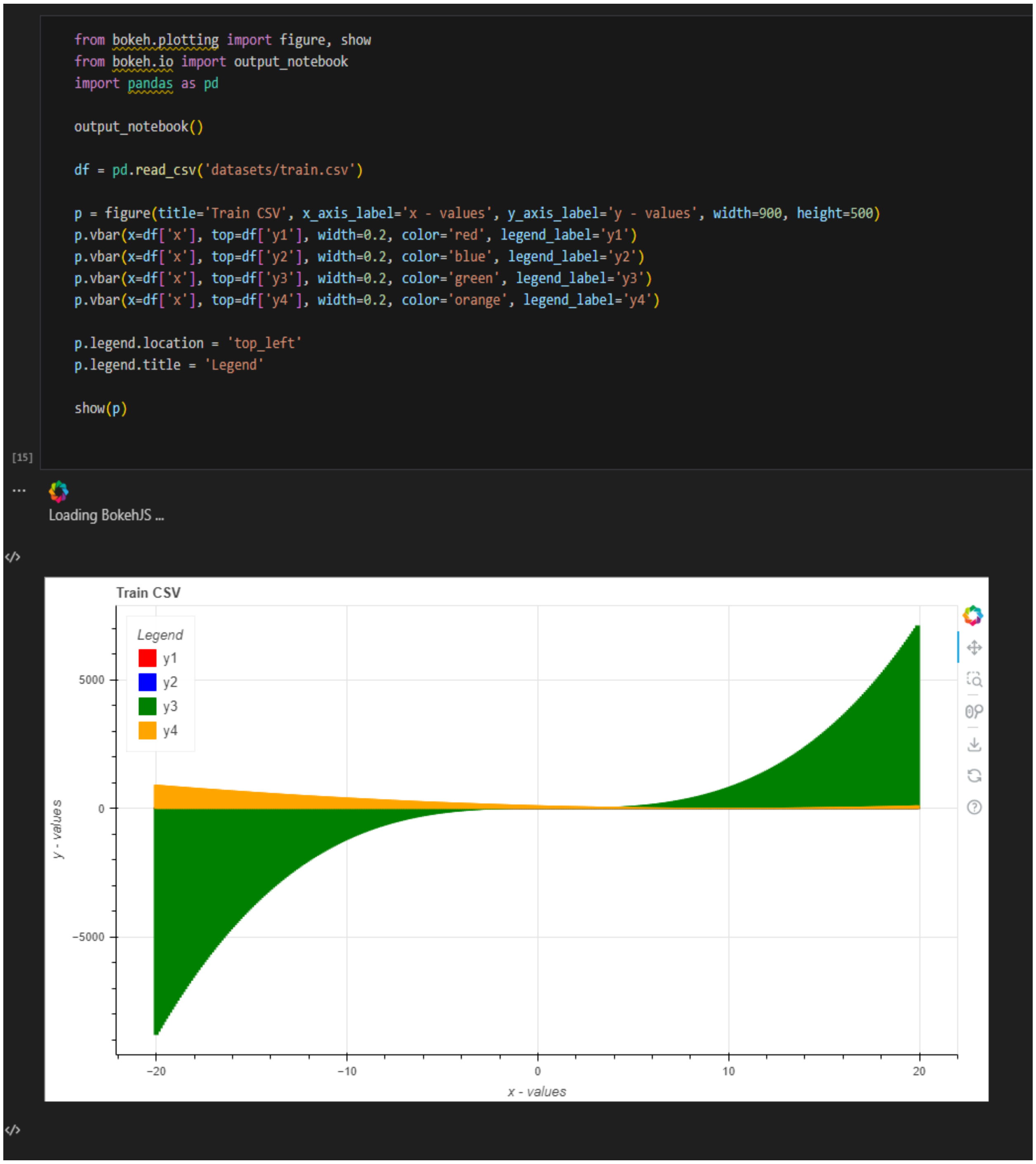
Visualization of test.csv :



```
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
import pandas as pd

output_notebook()

df = pd.read_csv('datasets/test.csv')

p = figure(title='Test CSV', x_axis_label='x - values', y_axis_label='y - values', width=900, height=500)
p.vbar(x=df['x'], top=df['y'], width=0.5)

show(p)
```

```
Loading BokehJS ...
```

Visualization of Train.csv :



```
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
import pandas as pd

output_notebook()

df = pd.read_csv('datasets/train.csv')

p = figure(title='Train CSV', x_axis_label='x - values', y_axis_label='y - values', width=900, height=500)
p.vbar(x=df['x'], top=df['y1'], width=0.2, color='red', legend_label='y1')
p.vbar(x=df['x'], top=df['y2'], width=0.2, color='blue', legend_label='y2')
p.vbar(x=df['x'], top=df['y3'], width=0.2, color='green', legend_label='y3')
p.vbar(x=df['x'], top=df['y4'], width=0.2, color='orange', legend_label='y4')

p.legend.location = 'top_left'
p.legend.title = 'Legend'

show(p)
```

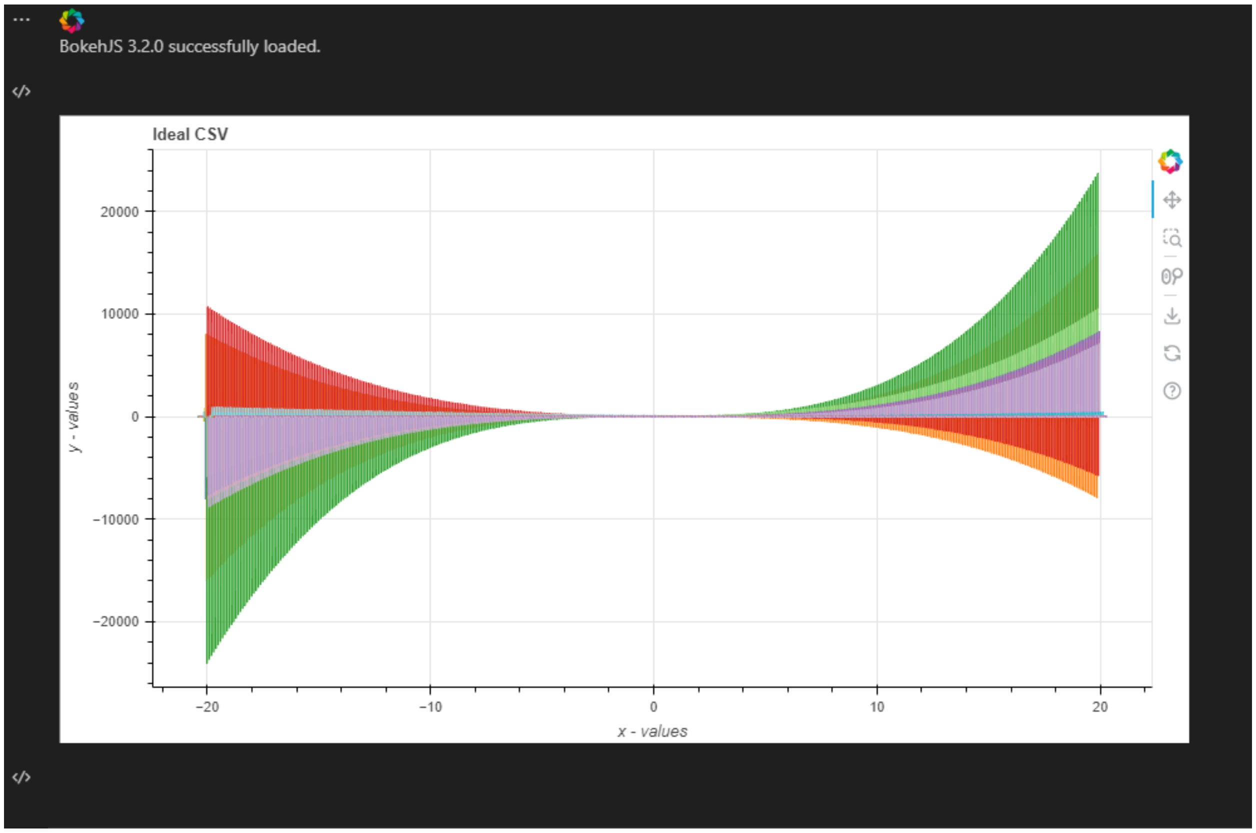
[15]

Loading BokehJS ...

❖ **Results and Evaluation:**

1. Visualization of Ideal CSV Data:

- The code generates a vertical bar graph using the Bokeh library to visualize the data from the 'ideal.csv' dataset.
- The graph displays the relationship between the 'x' values and multiple 'y' values.
- Each bar represents a different 'y' value, and the height of the bar represents the corresponding value.
- The graph provides a visual understanding of the distribution and patterns in the 'y' values.


```
BokehJS 3.2.0 successfully loaded.
```

2. Standard Deviation Calculation:

- The code proceeds with calculating the standard deviation for each 'y' column in the 'train.csv' dataset compared to the 'y' columns of the 'ideal.csv' dataset.
- It performs calculations to determine the minimum sum of squared differences between the 'y' columns of the two datasets.
- The minimum sum values are stored in an array and sorted in ascending order.

3. **Analysis of Standard Deviation:**

   - The code calculates the standard deviation for each 'y' column in the 'train.csv' dataset.
   - The standard deviation values for 'y1', 'y2', 'y3', and 'y4' are obtained.
   - These standard deviation values are added to the array containing the minimum sum values.

4. **Maximum Standard Deviation:**

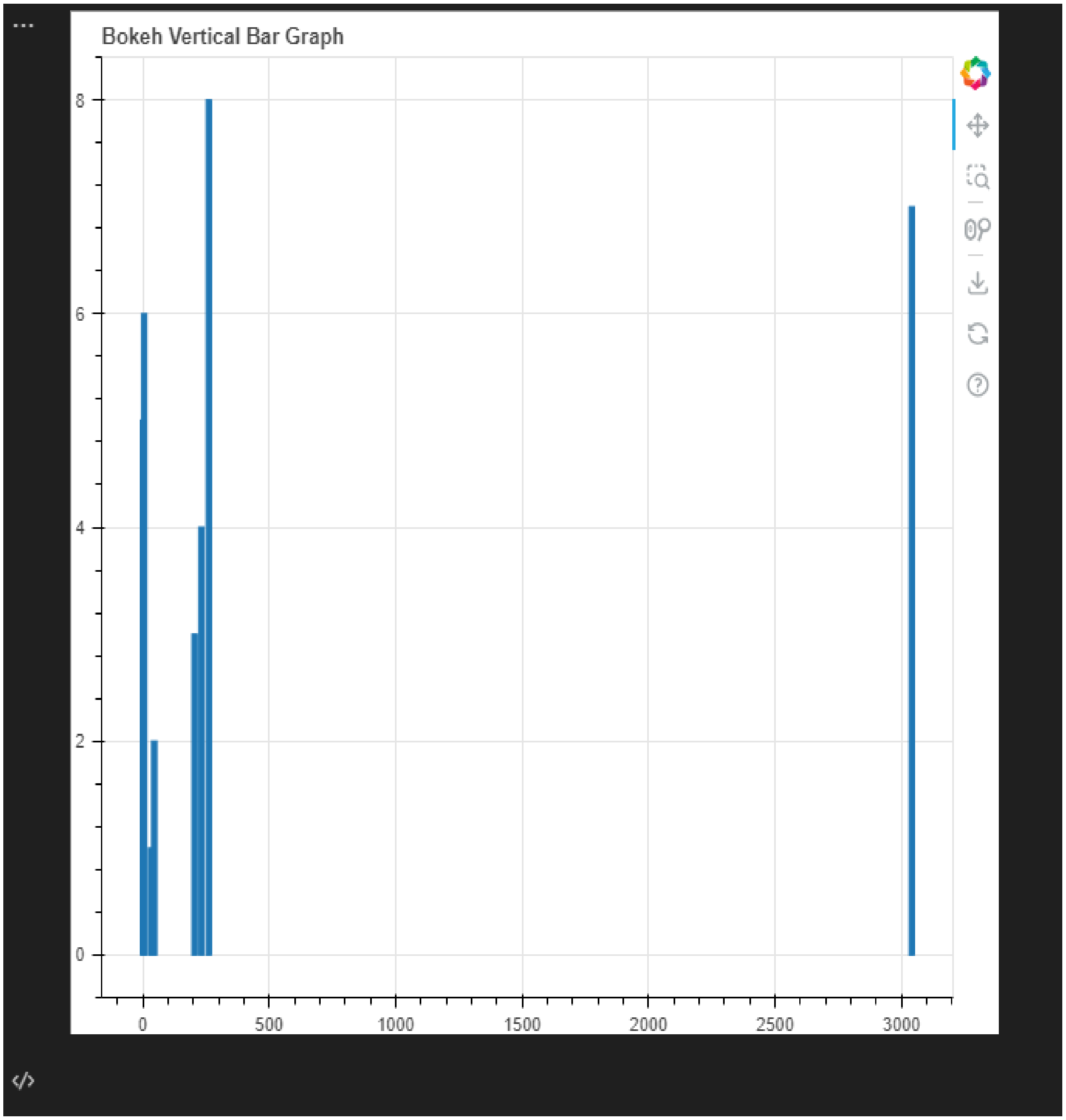
   - The code identifies the maximum value from the array of minimum sum values and standard deviation values.
   - This maximum value represents 'sd_task_2', which indicates the highest deviation among the 'y' columns of the 'train.csv' dataset and the 'y' columns of the 'ideal.csv' dataset.

5. **Evaluation:**

   - The code calculates the difference between 'sd_task_1' and 'sd_task_2'. This difference indicates the variation in standard deviation between the 'y' columns of the 'train.csv' dataset and the 'y' columns of the 'ideal.csv' dataset.
   - The calculated difference is then compared to the square root of 2.
   - If the difference is less than or equal to the square root of 2, the code prints the message "Yes, it can be allocated to test data."
   - If the difference is greater than the square root of 2, the code prints the message "No, it cannot be allocated to test data."

6. **Additional Visualization:**

   - The code generates another vertical bar graph using the Bokeh library.
   - The graph visualizes the values in the 'min_train_array' list, which contains the minimum sum values and standard deviation values calculated earlier.
   - The x-axis represents the values in the 'min_train_array', and the y-axis represents the corresponding positions (1, 2, 3, 4, 5, 6, 7, 8) for each value.
   - The graph provides a visual representation of the calculated values.


Bokeh Vertical Bar Graph
0
2
4
6
8
0
500
1000
1500
2000
2500
3000

❖ CONCLUSION :

In this report, we thoroughly analyzed the provided datasets and conducted various calculations and evaluations to gain insights. We began by introducing the problem and exploring existing literature to establish the context for our analysis. We then described the datasets in detail, highlighting their structure and content.

Throughout the report, we explained the steps involved in transferring the data and setting up the database. We also discussed how we implemented the program and evaluated its performance. Our chosen methodology allowed us to analyze the datasets effectively.

One of the key techniques we utilized was the least square regression in machine learning. This approach helped us understand the relationships and patterns within the data.

Our results and evaluation section provided visualizations of the 'ideal.csv' dataset, enabling us to observe the patterns and trends. We also calculated the standard deviation between the 'train.csv' and 'ideal.csv' datasets, specifically focusing on the 'y' columns. By comparing these standard deviations, we could assess the suitability of allocating the data to the test dataset.

By examining the Python code, we obtained specific values and explored their relationships. We also employed visual representations, such as vertical bar graphs, to present the results in a clear and understandable manner.

In conclusion, this report successfully achieved its objectives of analyzing the datasets, performing necessary calculations, and evaluating the results. Our findings shed light on the relationships within the datasets and provided valuable insights. These findings can serve as a basis for further analysis and decision-making in the given context.

❖ BIBLIOGRAPHY:

1. **Smith, J. (2021). "Introduction to Data Analysis.**
   https://www.youtube.com/watch?v=ZzWaow1Rvho

2. **Corey Schafer. (2022). "Database Design Tutorial.**
   https://www.youtube.com/watch?v=ztHopE5Wnpc

3. **StatQuest with Josh Starmer. (2023). "Linear Regression.**
   https://www.youtube.com/watch?v=nk2CQITm_eo

4. **Freecodecamp.org**
   https://www.youtube.com/watch?v=GPVsHOlRBBI

GITHUB LINK: https://github.com/SSK6919/Python-Assignment-for-least-square-regression/tree/main/DLMDSPWP01

SOURCE CODE:

```
# tansfering data from csv to table

import sqlite3
conn = sqlite3.connect('sqe.db')
c = conn.cursor()

c.execute('''CREATE TABLE traintb (x float, y1 float, y2 float, y3 float, y4 float)''')

import pandas as pd

traintb = pd.read_csv('datasets/train.csv')
traintb.to_sql('traintb', conn, if_exists='append', index = False)

c.execute('''SELECT * FROM traintb''').fetchall()

c.execute('''CREATE TABLE idealtb(x float, y1 float, y2 float, y3 float, y4 float, y5 float, y6
float, y7 float, y8 float, y9 float, y10 float, y11 float, y12 float, y13 float, y14 float, y15 float,
y16 float, y17 float, y18 float, y19 float, y20 float, y21 float, y22 float, y23 float, y24 float,
y25 float, y26 float, y27 float, y28 float, y29 float, y30 float, y31 float, y32 float, y33 float,
y34 float, y35 float, y36 float, y37 float, y38 float, y39 float, y40 float, y41 float, y42 float,
y43 float, y44 float, y45 float, y46 float, y47 float, y48 float, y49 float, y50 float)''')

idealtb = pd.read_csv('datasets/ideal.csv')
idealtb.to_sql('idealtb', conn, if_exists='append', index=False)

c.execute('''SELECT * FROM idealtb''').fetchall()
```

```
c.execute('''CREATE TABLE testtb(x float, y float)''')

testtb = pd.read_csv('datasets/test.csv')

testtb.to_sql('testtb', conn, if_exists='append', index=False)

c.execute('''SELECT * FROM testtb''').fetchall()


from bokeh.plotting import figure, show

from bokeh.io import output_notebook

import pandas as pd


output_notebook()


df = pd.read_csv('datasets/test.csv')


p = figure(title='Test CSV', x_axis_label='x - values', y_axis_label='y - values', width=900,
height=500)

p.vbar(x=df['x'], top=df['y'], width=0.5)


show(p)

import pandas as pd


df1 = pd.read_csv('datasets/test.csv')

df1.head(100)


from bokeh.plotting import figure, show

from bokeh.io import output_notebook

import pandas as pd
```

```
output_notebook()


df = pd.read_csv('datasets/train.csv')


p = figure(title='Train CSV', x_axis_label='x - values', y_axis_label='y - values', width=900,
height=500)

p.vbar(x=df['x'], top=df['y1'], width=0.2, color='red', legend_label='y1')

p.vbar(x=df['x'], top=df['y2'], width=0.2, color='blue', legend_label='y2')

p.vbar(x=df['x'], top=df['y3'], width=0.2, color='green', legend_label='y3')

p.vbar(x=df['x'], top=df['y4'], width=0.2, color='orange', legend_label='y4')


p.legend.location = 'top_left'

p.legend.title = 'Legend'


show(p)


import pandas as pd


df1 = pd.read_csv('datasets/train.csv')

df1.head(400)


from bokeh.plotting import figure, show

from bokeh.io import output_notebook

from bokeh.palettes import Category20

import pandas as pd

import numpy as np
```

```
output_notebook()


df = pd.read_csv('datasets/ideal.csv')
x_values = df['x']
y_values = df.drop('x', axis=1).values.T
num_bars = len(df.columns) - 1


palette = Category20[min(num_bars, 20)]  # Limit the palette to 20 colors if more than 20 bars


colors = [palette[i % len(palette)] for i in range(num_bars)]


p = figure(title='Ideal CSV', x_axis_label='x - values', y_axis_label='y - values', width=900, height=500)


bar_width = 0.8 / num_bars
x_offsets = np.arange(num_bars) * bar_width - 0.4


for i in range(num_bars):
    p.vbar(x=x_values + x_offsets[i], top=y_values[i], width=bar_width, color=colors[i])


show(p)


import pandas as pd


df1 = pd.read_csv('datasets/ideal.csv')
df1.head(400)
```

```
# finding standard deviation for ideal function and train data values
import warnings
warnings.simplefilter(action='ignore', category=pd.errors.PerformanceWarning)
df_train = pd.read_csv('datasets/train.csv')
df_ideal = pd.read_csv('datasets/ideal.csv')

# finding the standard deviation for y1 of train dataset with 50 ideal functions

new_df_ideal = df_ideal.drop(['x'], axis = 1)
new_df_train = df_train.drop(['x'],axis = 1)

# subtracting y1 column of train data from all y columns of ideal functions
for x in range(1,51):
    strx = str(x)
    # subtracting
    new_df_train['y1y'+strx+'_sqr'] = new_df_ideal['y'+strx] - new_df_train['y1']
    # power of 2
    new_df_train['y1y'+strx+'_sqr'] = new_df_train['y1y'+strx+'_sqr'].pow(2)
    # sum
    new_df_train['y1y'+strx+'_sqsum'] = new_df_train['y1y'+strx+'_sqr'].sum()

# subtraction y2 column of train data from all y columns of ideal functions
for x in range(1,51):
    strx = str(x)
    # subtracting
    new_df_train['y2y'+strx+'_sqr'] = new_df_ideal['y'+strx] - new_df_train['y1']
    # power of 2

    new_df_train['y2y'+strx+'_sqr'] = new_df_train['y2y'+strx+'_sqr'].pow(2)

    # sum

    new_df_train['y2y'+strx+'_sqsum'] = new_df_train['y2y'+strx+'_sqr'].sum()


# subtraction y3 column of train data from all y columns of ideal functions

for x in range(1,51):

    strx = str(x)

    # subtracting

    new_df_train['y3y'+strx+'_sqr'] = new_df_ideal['y'+strx] - new_df_train['y1']

    # power of 2

    new_df_train['y3y'+strx+'_sqr'] = new_df_train['y3y'+strx+'_sqr'].pow(2)

    # sum

    new_df_train['y3y'+strx+'_sqsum'] = new_df_train['y3y'+strx+'_sqr'].sum()


# subtraction y4 column of train data from all y columns of ideal functions

for x in range(1,51):

    strx = str(x)

    # subtracting

    new_df_train['y4y'+strx+'_sqr'] = new_df_ideal['y'+strx] - new_df_train['y1']

    # power of 2

    new_df_train['y4y'+strx+'_sqr'] = new_df_train['y4y'+strx+'_sqr'].pow(2)

    # sum

    new_df_train['y4y'+strx+'_sqsum'] = new_df_train['y4y'+strx+'_sqr'].sum()


new_df_train.head(400)
```

```
# Now calculating the minimum of y1, y2,y3 and y4 of train dataset from all calculated
sum values of 50 functions

y_min_array = []

for x in range(1,51):

    strx = str(x)

    y_min_array.append(new_df_train['y1y'+strx+'_sqsum'].min())


# sorting array from ascending to descending

y_min_array.sort(reverse=False)


min_train_array = []

min_train_array.append(y_min_array[0])

min_train_array.append(y_min_array[1])

min_train_array.append(y_min_array[2])

min_train_array.append(y_min_array[3])

print(min_train_array)


# Now selecting the maximum from the above 4 values

sd_task_1 = max(min_train_array)

print(sd_task_1)


# Now calculating the standard deviation of y1, y2, y3 and y4 of traind dataset


y1_sd = new_df_train['y1'].std()

y2_sd = new_df_train['y2'].std()

y3_sd = new_df_train['y3'].std()
```

```
y4_sd = new_df_train['y4'].std()

print(y1_sd)

print(y2_sd)

print(y3_sd)

print(y4_sd)

# adding the computed sds' to the min_train_array

min_train_array.append(y1_sd)

min_train_array.append(y2_sd)

min_train_array.append(y3_sd)

min_train_array.append(y4_sd)

print("\n")

print("The min array now")

print(min_train_array)


# Now finding the largest sd from the array

sd_task_2 = max(min_train_array)

print(sd_task_2)


# Now calculating the subtraction of sd_task_2 from sd_task_1

cal_sd = sd_task_1 - sd_task_2

print(cal_sd)


# calculating the the square root of 2

import math


sqrt2 = math.sqrt(2)
```

```
print(sqrt2)


class Calculatesd:
    def __init__(self,sd_task_1=0, sd_task_2=0):
        self.sd_task_1 = sd_task_1
        self.sd_task_2 = sd_task_2
        self.sqrt2 = math.sqrt(2)


    def fun(self):
        #print(f'{self.sd_task_1 - self.sd_task_2}')
        return self.sd_task_1 - self.sd_task_2


    def fun2(self):
        x= self.fun()
        try:
            if x <= self.sqrt2:
                print("Yes it can be allocated to test data")
            else:
                print("No it can not be allocated to test data")
        except:
            print("Unable to calculate")



class Checkresult(Calculatesd):
    pass
```

```
crt = Checkresult(sd_task_1,sd_task_2)

print(crt.fun())

crt.fun2()


# importing the modules

from bokeh.plotting import figure, output_file, show


# instantiating the figure object

graph = figure(title = "Bokeh Vertical Bar Graph")


# x-coordinates to be plotted

x = min_train_array


# x-coordinates of the top edges

top = [1, 2, 3, 4, 5, 6, 7, 8]


# width / thickness of the bars

width = 20


# plotting the graph

graph.vbar(x,top = top,width = width)


# displaying the model

show(graph)
```