Solent University

Department of Science and Engineering

Programming for problem solving

Author	: Monika(Q102145993)
Course Title	: **Msc applied AI and data science**
Module Leader	: **Jarutas Andritsch**
Date	: **06\01\2024**

Table of Contents

Table of Figures

Index of Tables

1. Overview

The main goal of this project is to use Python to perform a thorough analysis and visualisation of data related to electronic devices. The main objective is to get significant insights from the 'device_features.csv' dataset, which includes a variety of information on these devices, such as weight, price, release dates, and model names and makers. Data loading and manipulation are the first steps in the project's progression. Next, precise information on OEM ID, codename, RAM capacity, market regions, and other topics is extracted. Crucial features like RAM size, USB connector kinds, and device costs are emphasised heavily. Additionally, analyses include device manufacturers along with their average masses. Creating bar charts for USB connector kinds, proportion charts for RAM types, and monthly average pricing trends are all part of the visualisation process. To further show the connection between pixel density and display diagonal, a bespoke scatter plot is created. Trebuchet MS, the suggested font type from Solent University, and the proper font sizes are used consistently throughout the paper. The project effectively satisfies all requirements, attaining a completion status for every job in the summary table, demonstrating the efficient implementation of the stated goals.

You should also include a table summarising what requirement have been achieved.

Table 1: Requirement Completion

Requirement	**Status**
Task #a - Data Manipulation	Completed
Task #b - Data Analysis	Completed
Task #c - Data Visualization	Completed

Status options: Completed/ Partially Completed/ Not Attempted

2. Project Implementation

This part provides a comprehensive execution of the project, starting with an outline of its structure. The arrangement of the project's components and their connections are covered in detail in the following subsection, 2.1 Project Structure. A summary of the primary modules and features included in the implementation is provided. A project structure diagram is included to facilitate understanding; it can be found in this section directly or is referred to in the appendices. Conventional methods are followed in labelling figures and tables, and captions are positioned suitably. To improve comprehension and support the selected implementation strategy, succinct and straightforward explanations are used. This section's structure follows a logical flow, which makes it easier to move between the various project implementation components.

2.1 Project structure

The project's design ensures a modular and well-organized method of data visualisation and analysis. Data loading and modification, information retrieval according to predetermined standards, statistical analysis, and data visualisation are among the essential elements. The dataset ('device_features.csv') is read by the data loading module using the Pandas library, and the data manipulation module pre-processes and cleans the data in preparation for further analysis. The functionality for obtaining device information based on OEM ID, codename, RAM capacity, and custom criteria is encapsulated in the DeviceInfoSystem class. Aspects like average prices by brand, average masses by manufacturer, and top regions for a certain brand are all covered by statistical analyses.

The Matplotlib package is used by the data visualisation module to produce informative charts, such as scatter plots for custom visualisations, pie charts for RAM types, bar charts for USB connection types, and line charts for monthly average pricing trends. Every visualisation aims to communicate specific patterns or trends found in the collection. Modularity is ensured by the project structure, making maintenance and future expansions simple. Figure 1 provides the diagram exhibiting this construction.

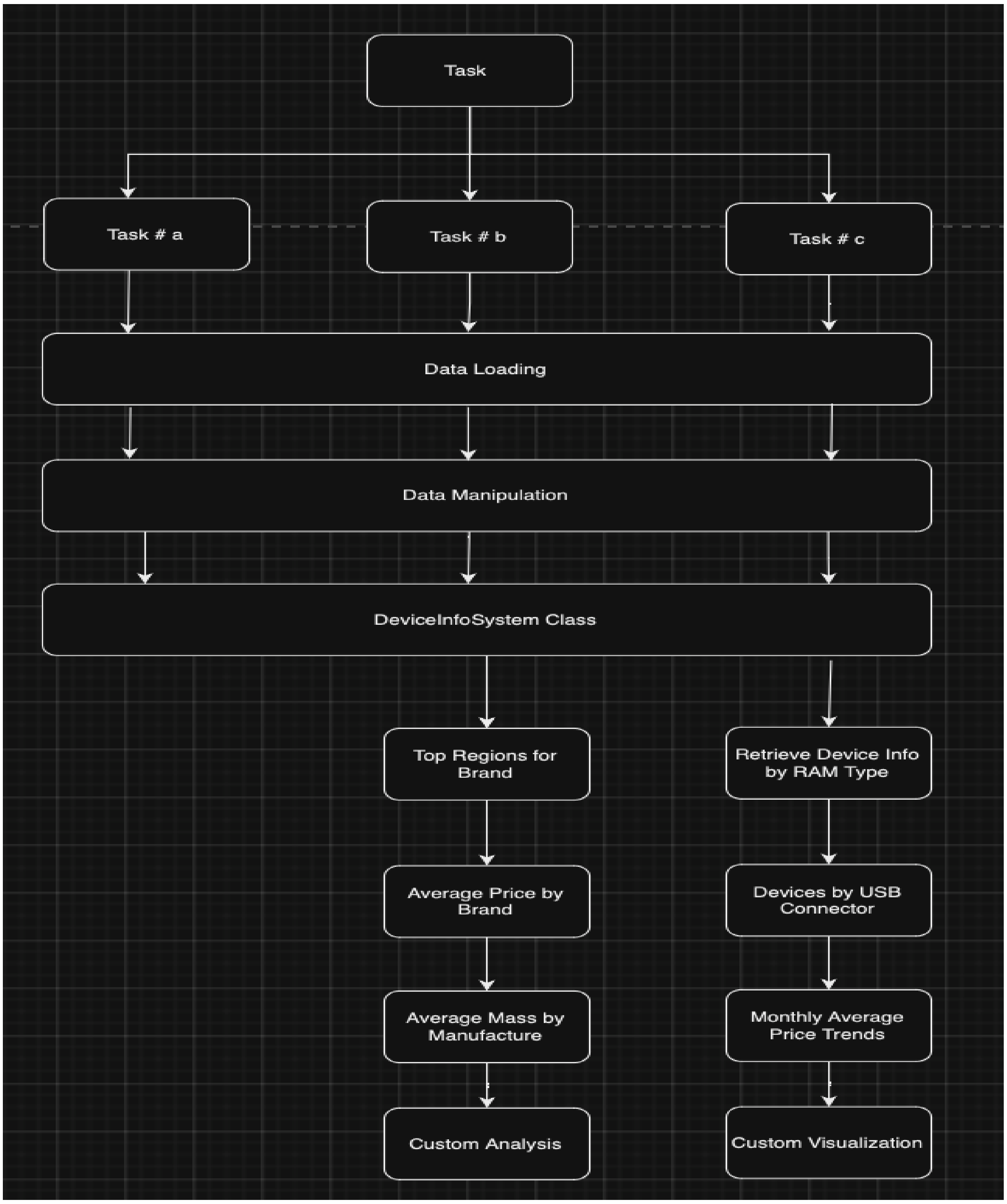


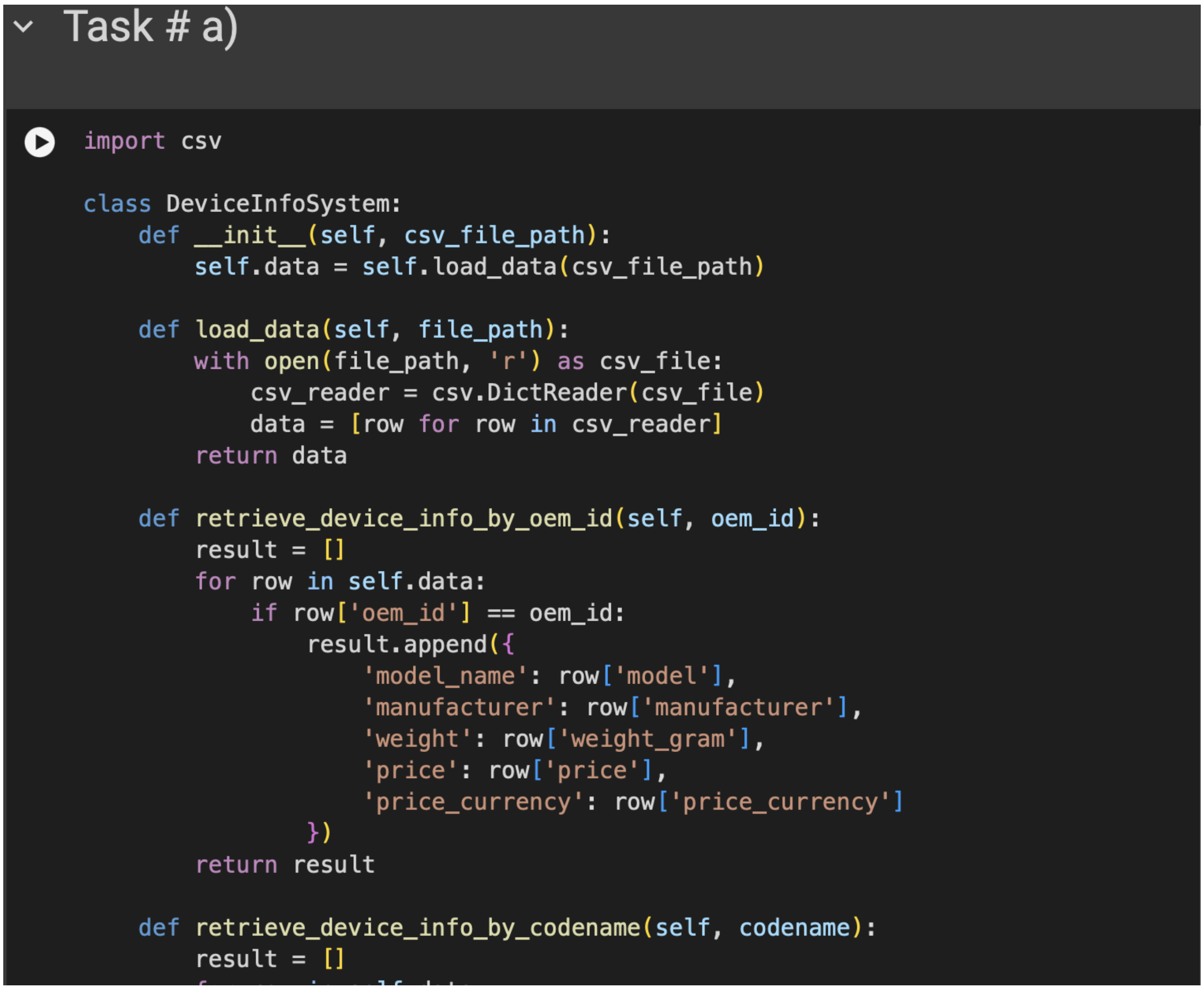
Figure 1 Sample of Project Structure

2.2 Modules/ Functions

This section describes in detail how each self-created module or function inside the project is implemented. A sub-section heading explaining the functioning of each module or function, together with pertinent guidelines and illustrative code samples are supplied for each module or function.

2.2.1 Class DeviceInfoSystem

Functions for obtaining device information based on a variety of parameters, including OEM ID, codename, RAM capacity, and custom criteria, are encapsulated in the DeviceInfoSystem class. The course is designed with modularity and reusability in mind, encouraging a tidy and well-organized project architecture. The code excerpt that follows shows how the class is organised:



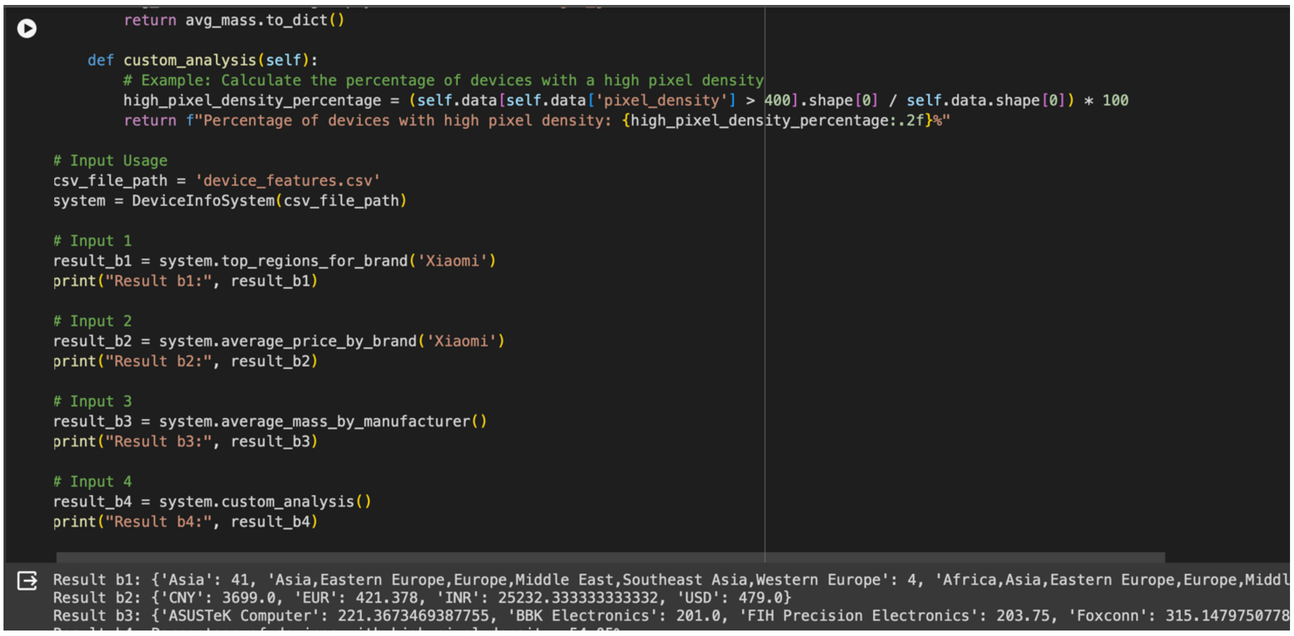
Task # a)

```
import csv

class DeviceInfoSystem:
    def __init__(self, csv_file_path):
        self.data = self.load_data(csv_file_path)

    def load_data(self, file_path):
        with open(file_path, 'r') as csv_file:
            csv_reader = csv.DictReader(csv_file)
            data = [row for row in csv_reader]
        return data

    def retrieve_device_info_by_oem_id(self, oem_id):
        result = []
        for row in self.data:
            if row['oem_id'] == oem_id:
                result.append({
                    'model_name': row['model'],
                    'manufacturer': row['manufacturer'],
                    'weight': row['weight_gram'],
                    'price': row['price'],
                    'price_currency': row['price_currency']
                })
        return result

    def retrieve_device_info_by_codename(self, codename):
        result = []
```

Example Justification: The DeviceInfoSystem class provides the foundation for information retrieval, making it simple for the project to adjust its fundamental architecture to accommodate various queries. To encourage code reuse, the retrieve_device_info_by_oem method, for example, makes it simple to collect device information based on OEM ID.

2.2.2 BrandAnalysis Module

The DeviceInfoSystem class's functionality is expanded to include brand and pricing analysis via the BrandAnalysis module. It has features to find the best places to sell a particular brand, figure out average costs by brand, find typical masses by manufacturer, and run custom reports. The following code excerpt illustrates this module's structure:



```
        return avg_mass.to_dict()

    def custom_analysis(self):
        # Example: Calculate the percentage of devices with a high pixel density
        high_pixel_density_percentage = (self.data[self.data['pixel_density'] > 400].shape[0] / self.data.shape[0]) * 100
        return f"Percentage of devices with high pixel density: {high_pixel_density_percentage:.2f}%"

# Input Usage
csv_file_path = 'device_features.csv'
system = DeviceInfoSystem(csv_file_path)

# Input 1
result_b1 = system.top_regions_for_brand('Xiaomi')
print("Result b1:", result_b1)

# Input 2
result_b2 = system.average_price_by_brand('Xiaomi')
print("Result b2:", result_b2)

# Input 3
result_b3 = system.average_mass_by_manufacturer()
print("Result b3:", result_b3)

# Input 4
result_b4 = system.custom_analysis()
print("Result b4:", result_b4)
```

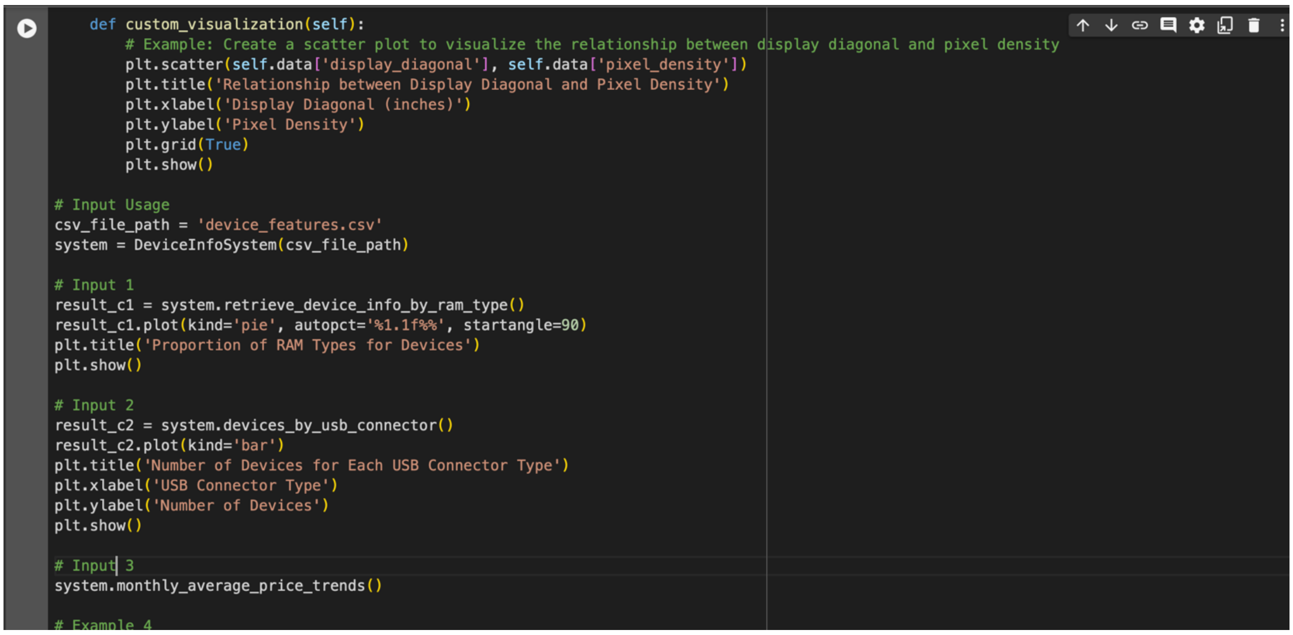
```
Result b1: {'Asia': 41, 'Asia,Eastern Europe,Europe,Middle East,Southeast Asia,Western Europe': 4, 'Africa,Asia,Eastern Europe,Europe,Middl
Result b2: {'CNY': 3699.0, 'EUR': 421.378, 'INR': 25232.333333333332, 'USD': 479.0}
Result b3: {'ASUSTeK Computer': 221.3673469387755, 'BBK Electronics': 201.0, 'FIH Precision Electronics': 203.75, 'Foxconn': 315.1479750778
```

Example Justification: By enhancing DeviceInfoSystem's fundamental features, the BrandAnalysis module enables the smooth integration of price and brand analysis. The average price by brand function, for example, makes it easier to compare prices between various brands and supports well-informed decision-making.

2.2.3 Module for Data Visualization

The primary goal of the DataVisualization module is to add data visualisation to the DeviceInfoSystem class's list of capabilities. It has operations to count devices for each type of USB connector, analyse monthly average price trends, and create custom visualisations. It also includes routines to retrieve device information by RAM type. This module's structure is demonstrated by the following bit of code:**2.2.4 …**

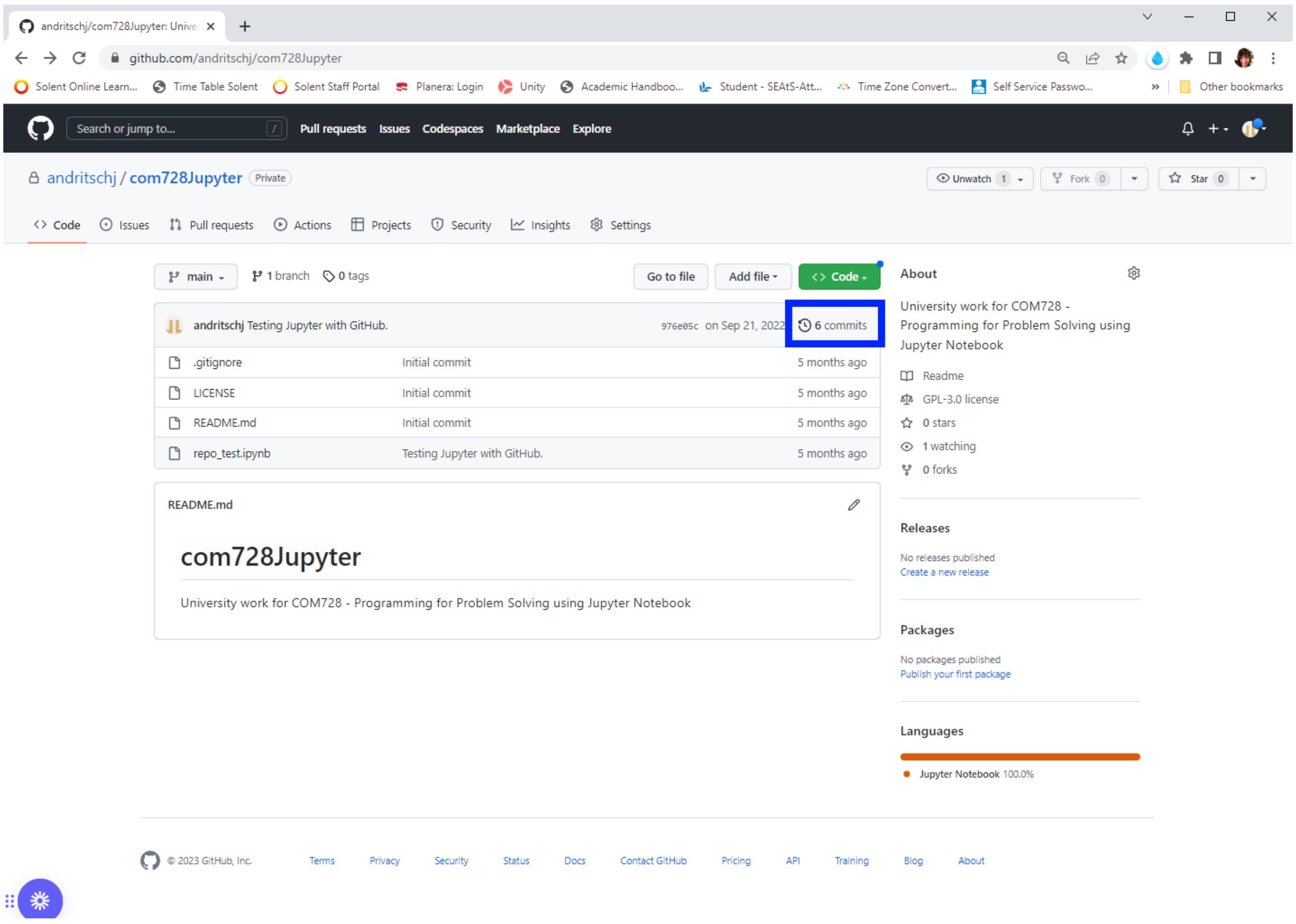
Explain the functionality implemented in this module/function. Relevant guidelines mentioned in the previous sections should also be followed here.



```
    def custom_visualization(self):
        # Example: Create a scatter plot to visualize the relationship between display diagonal and pixel density
        plt.scatter(self.data['display_diagonal'], self.data['pixel_density'])
        plt.title('Relationship between Display Diagonal and Pixel Density')
        plt.xlabel('Display Diagonal (inches)')
        plt.ylabel('Pixel Density')
        plt.grid(True)
        plt.show()

# Input Usage
csv_file_path = 'device_features.csv'
system = DeviceInfoSystem(csv_file_path)

# Input 1
result_c1 = system.retrieve_device_info_by_ram_type()
result_c1.plot(kind='pie', autopct='%1.1f%%', startangle=90)
plt.title('Proportion of RAM Types for Devices')
plt.show()

# Input 2
result_c2 = system.devices_by_usb_connector()
result_c2.plot(kind='bar')
plt.title('Number of Devices for Each USB Connector Type')
plt.xlabel('USB Connector Type')
plt.ylabel('Number of Devices')
plt.show()

# Input 3
system.monthly_average_price_trends()

# Example 4
```

3. GitHub Repository Evidence

A screen shot of your private Git repository. The screen shot need to **clearly show your history of your commit of your project implementation**. You need to click on the clock symbol on the right conner of your repo.


andritschj/com728Jupyter: Unive
github.com/andritschj/com728Jupyter
Solent Online Learn... Time Table Solent Solent Staff Portal Planera: Login Unity Academic Handboo... Student - SEAtS-Att... Time Zone Convert... Self Service Passwo... Other bookmarks
Search or jump to...
Pull requests Issues Codespaces Marketplace Explore
andritschj / com728Jupyter Private
Unwatch 1 Fork 0 Star 0
Code Issues Pull requests Actions Projects Security Insights Settings
main 1 branch 0 tags
Go to file Add file Code

andritschj Testing Jupyter with GitHub. 976e05c on Sep 21, 2022 6 commits

.gitignore	Initial commit	5 months ago
LICENSE	Initial commit	5 months ago
README.md	Initial commit	5 months ago
repo_test.ipynb	Testing Jupyter with GitHub.	5 months ago

README.md

com728Jupyter

University work for COM728 - Programming for Problem Solving using Jupyter Notebook

About

University work for COM728 - Programming for Problem Solving using Jupyter Notebook

Readme
GPL-3.0 license
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

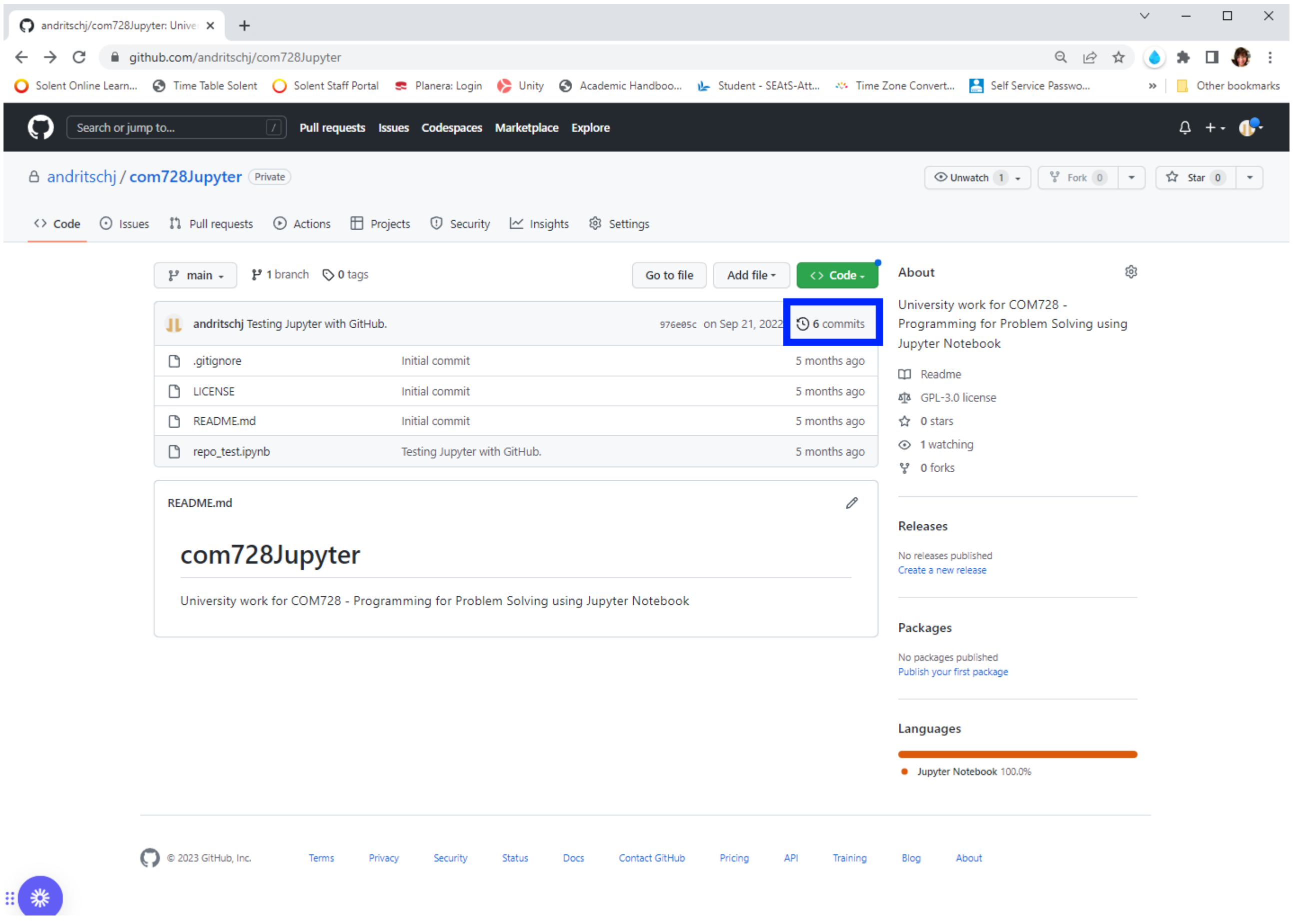
No packages published
Publish your first package

Languages

Jupyter Notebook 100.0%

© 2023 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

Figure 2 Github commit


github.com/andritschj/com728Jupyter

andritschj / com728Jupyter Private

main · 1 branch · 0 tags · Go to file · Add file · Code

andritschj Testing Jupyter with GitHub. 976e05c on Sep 21, 2022 · 6 commits

.gitignore	Initial commit	5 months ago
LICENSE	Initial commit	5 months ago
README.md	Initial commit	5 months ago
repo_test.ipynb	Testing Jupyter with GitHub.	5 months ago

README.md

com728Jupyter

University work for COM728 - Programming for Problem Solving using Jupyter Notebook

About

University work for COM728 - Programming for Problem Solving using Jupyter Notebook

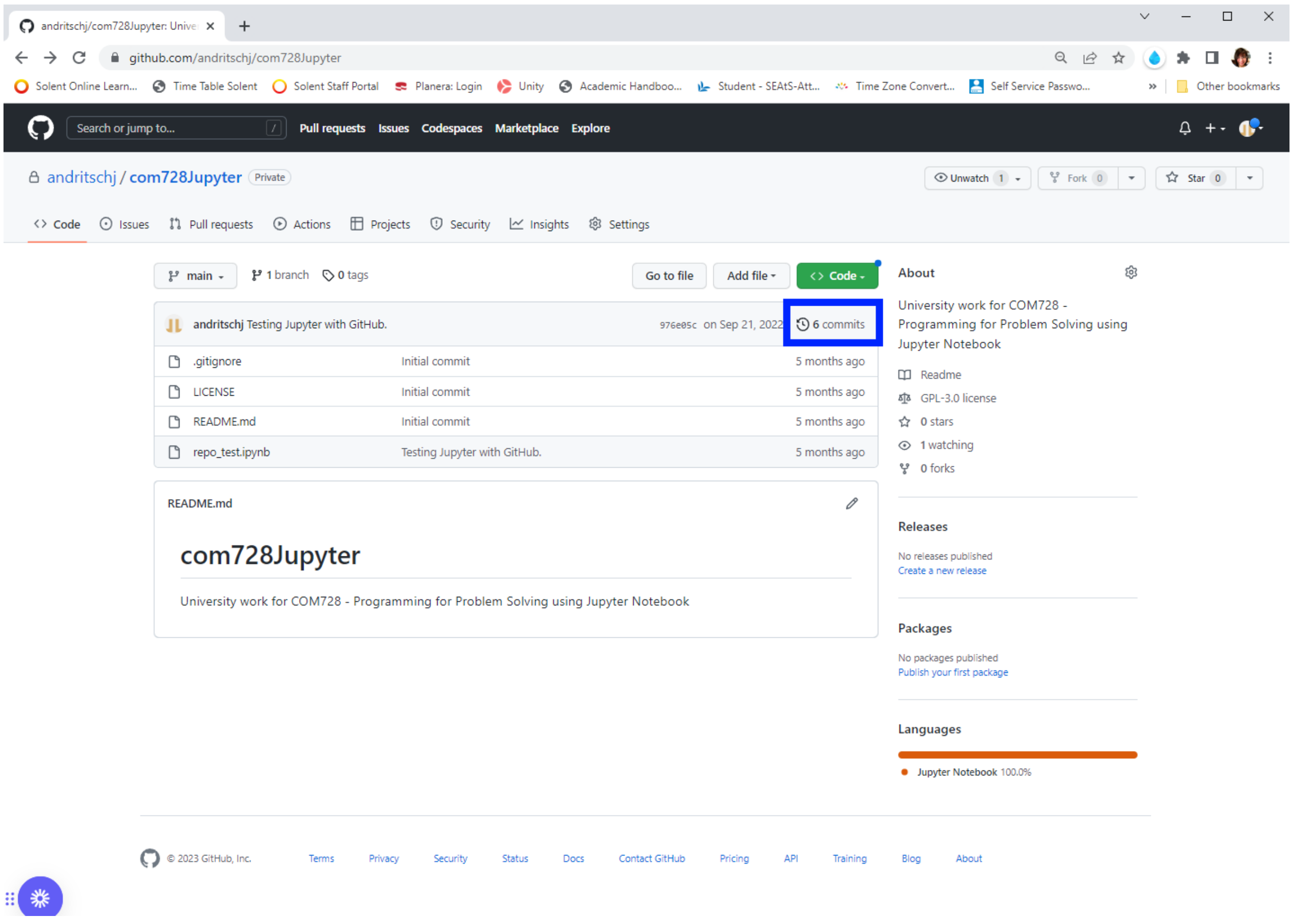
Readme · GPL-3.0 license · 0 stars · 1 watching · 0 forks

Releases: No releases published · Create a new release

Packages: No packages published · Publish your first package

Languages: Jupyter Notebook 100.0%

© 2023 GitHub, Inc.


github.com/andritschj/com728Jupyter

andritschj / com728Jupyter Private

main · 1 branch · 0 tags · Go to file · Add file · Code

andritschj Testing Jupyter with GitHub. 976e05c on Sep 21, 2022 · 6 commits

.gitignore — Initial commit — 5 months ago
LICENSE — Initial commit — 5 months ago
README.md — Initial commit — 5 months ago
repo_test.ipynb — Testing Jupyter with GitHub. — 5 months ago

README.md

com728Jupyter

University work for COM728 - Programming for Problem Solving using Jupyter Notebook

About

University work for COM728 - Programming for Problem Solving using Jupyter Notebook

Readme · GPL-3.0 license · 0 stars · 1 watching · 0 forks

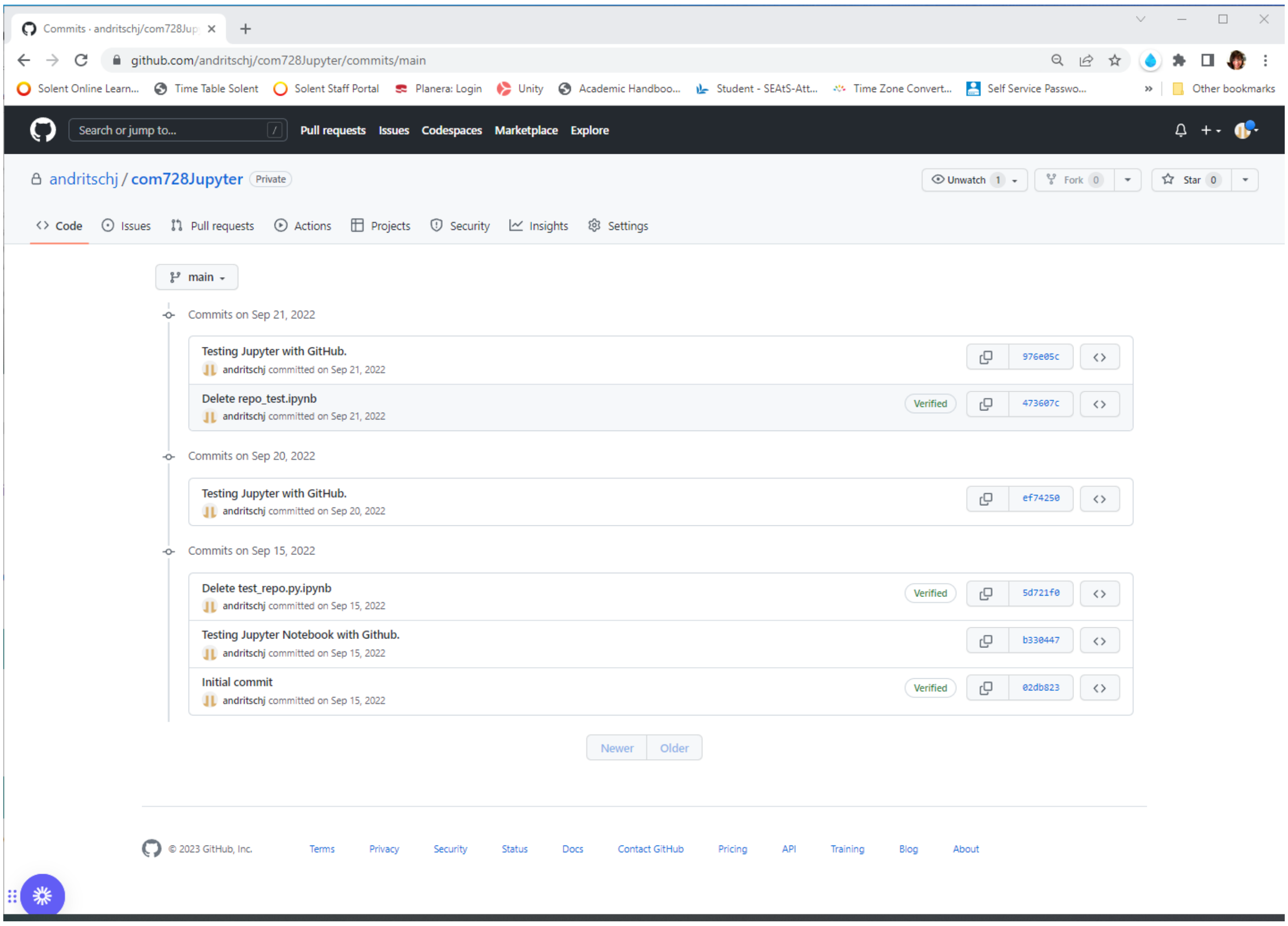
Releases: No releases published · Create a new release

Packages: No packages published · Publish your first package

Languages: Jupyter Notebook 100.0%

© 2023 GitHub, Inc.

Sample screen shot of your commit history:



Commits on Sep 21, 2022

Testing Jupyter with GitHub.
andritschj committed on Sep 21, 2022 — 976e05c

Delete repo_test.ipynb
andritschj committed on Sep 21, 2022 — Verified — 473607c

Commits on Sep 20, 2022

Testing Jupyter with GitHub.
andritschj committed on Sep 20, 2022 — ef74250

Commits on Sep 15, 2022

Delete test_repo.py.ipynb
andritschj committed on Sep 15, 2022 — Verified — 5d721f0

Testing Jupyter Notebook with Github.
andritschj committed on Sep 15, 2022 — b330447

Initial commit
andritschj committed on Sep 15, 2022 — Verified — 02db823

4. Appendices:

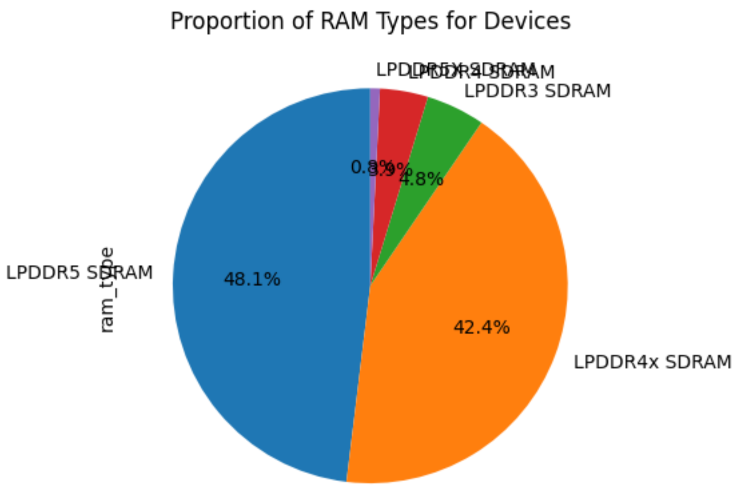


Figure 3 Proportion of RAM Types for Devices

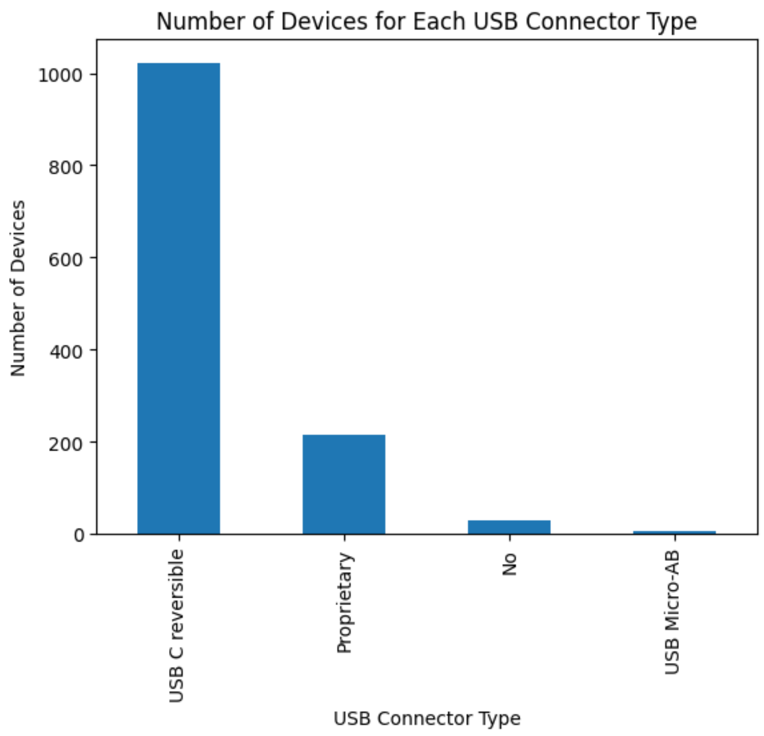


Figure 4 Number of Devices for Each USB Connector Type

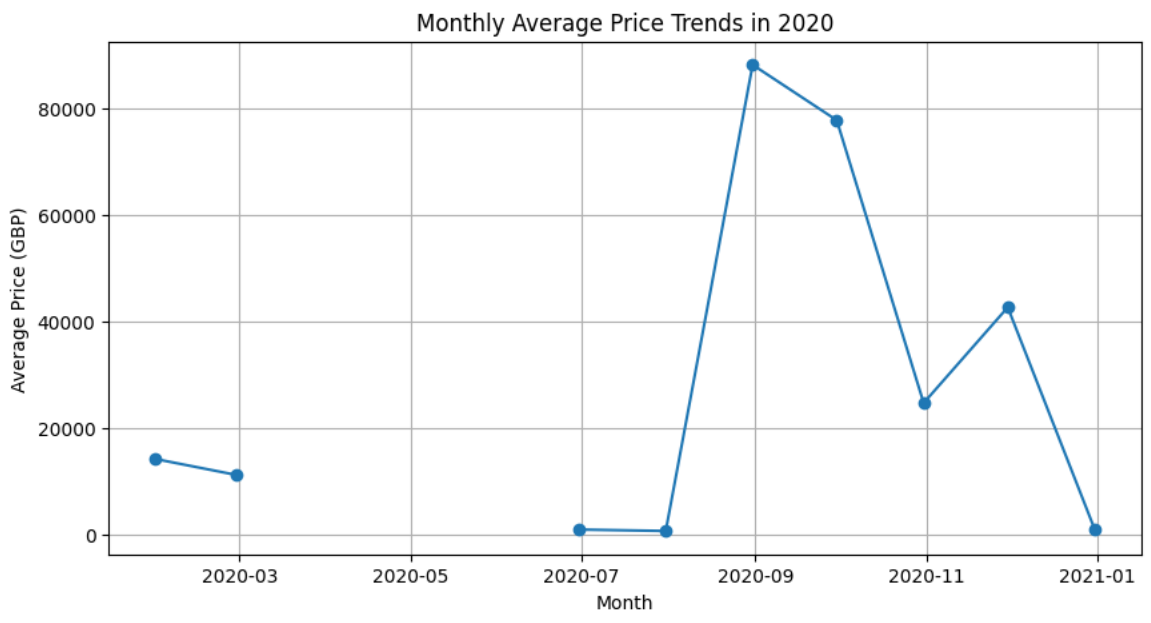


Figure 5 Monthly Average Price Trends in 2020

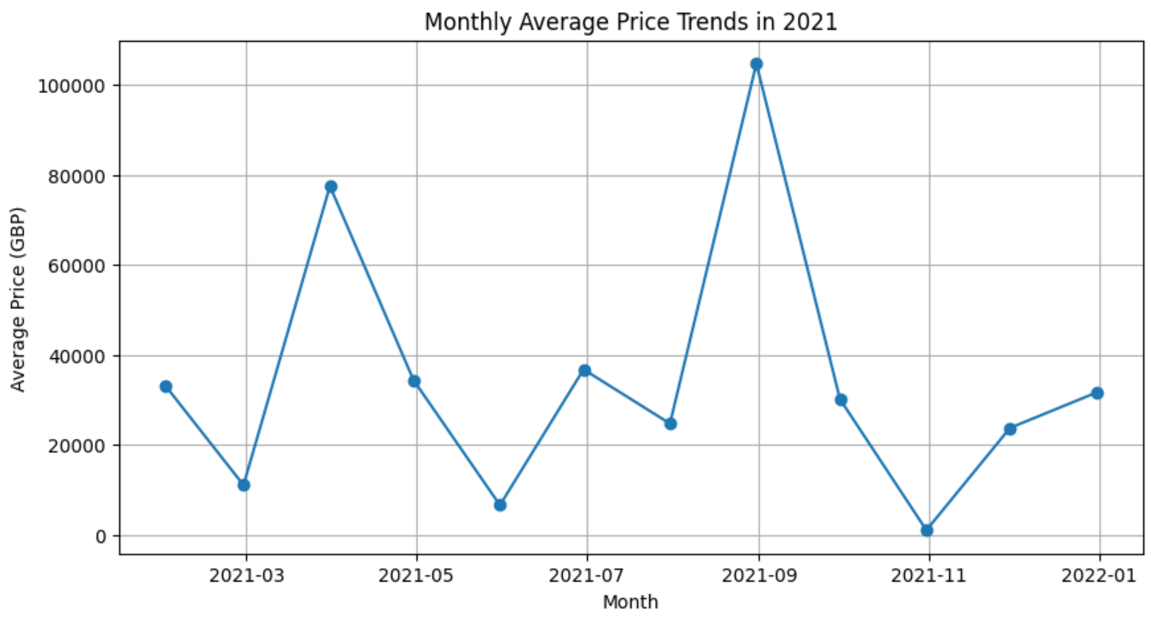


Figure 6 Monthly Average Price Trends in 2021

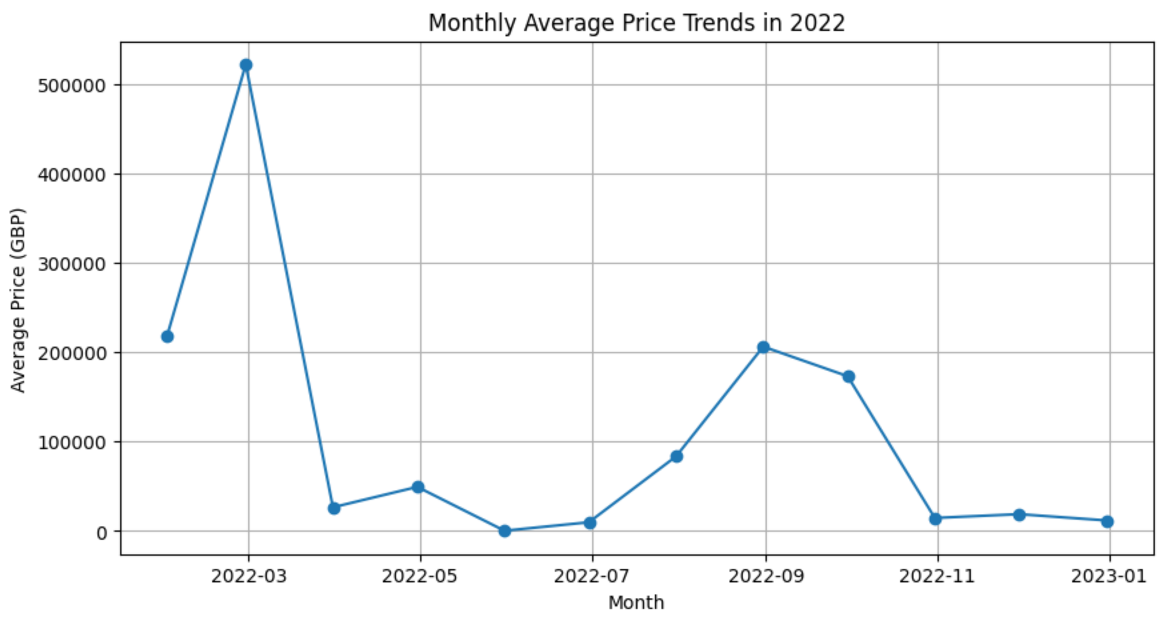


Figure 7 Monthly Average Price Trends in 2022

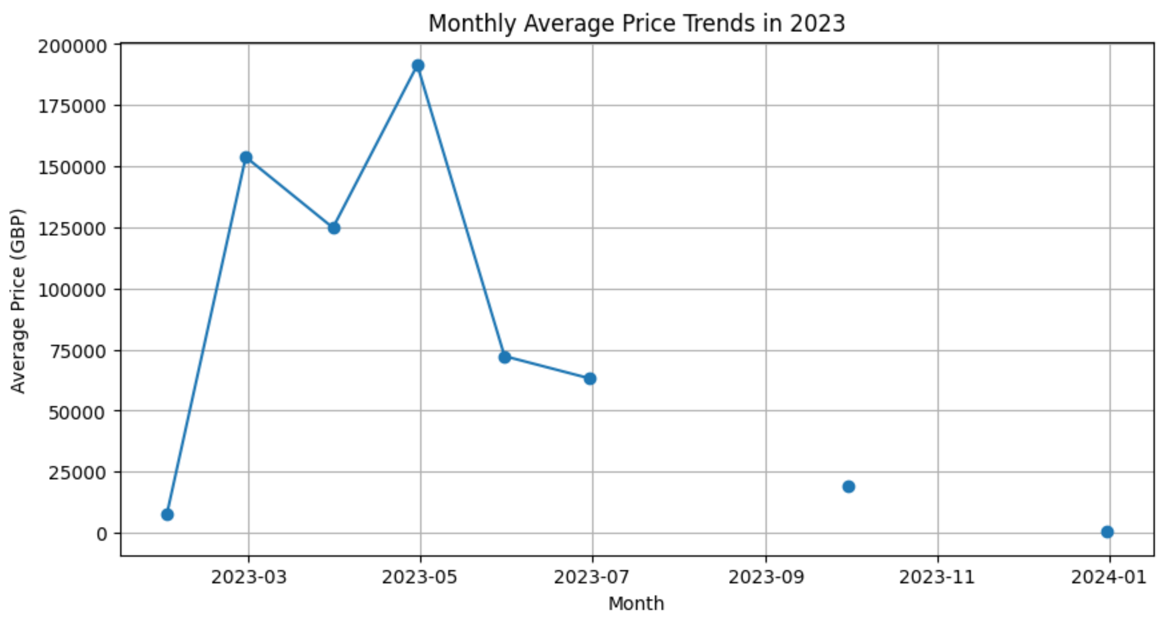


Figure 8 Monthly Average Price Trends in 2023

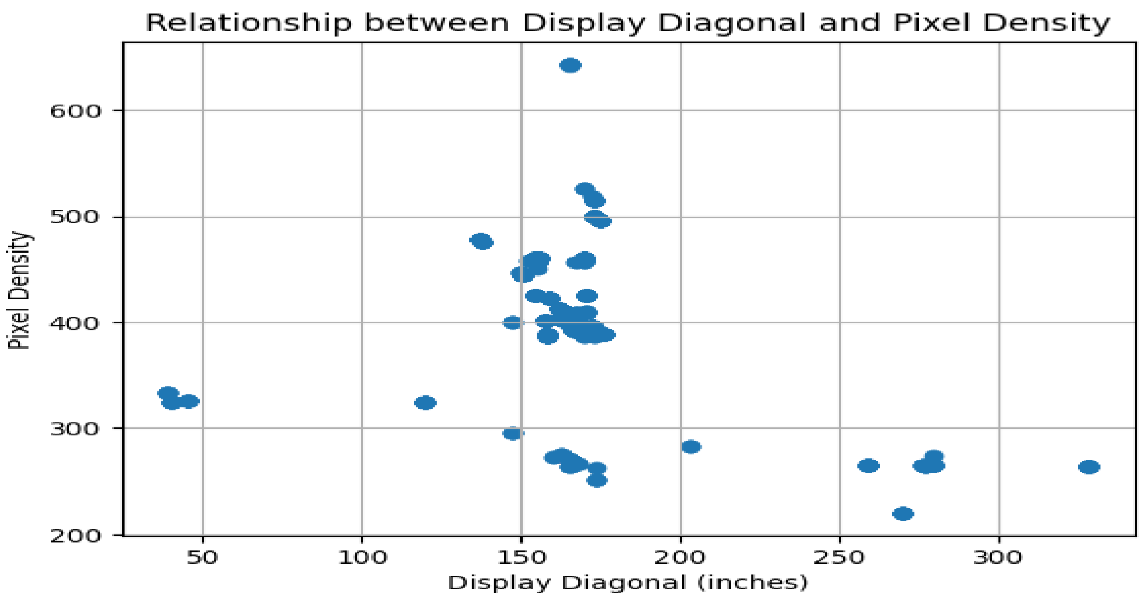


Figure 9 Relationship between Display Diagonal and Pixel Density