

Python for Data Analytics

Xi Li (李曦)

Professor of Marketing

Professor of Information and Innovation Management

Director, Asia Case Research Centre

Associate Director, Institute of Digital Economy and Innovation

Ph.D. (Management), University of Toronto

M.Phil. (Operations Research), HKUST

B.E. (Computer Science), Tsinghua University

Email: xili@hku.hk

Office hour: by appointment

Course Website

Here: https://ximarketing.github.io/_pages/teaching/

Password: **ATC**

All course materials will be posted on this website.

Python

You should already know about these...

- Install modules in Python
- pip install
- Running Python codes
- Branch and Loop
- Functions

Let Vincent (or me) know if you don't know about these topics.

Python

In the next a few classes, we are learning more things!

- Statistical Tests
- Statistical Analysis and Basic Machine Learning: Linear and Logistic Regression, Multinomial Logit Models, Artificial Neural Networks, and Clustering
- Advanced Algorithms: Probabilistic, Dynamic Programming, Divide and Conquer
- Unstructured Data: Text and Images
- API and Data Scraping

Python

Our plan is too ambiguous and we only have 15 hours left.
How can we learn so many topics in 5 sessions?

Schedule

Classes 6 and 7: Python for Data Analytics

Class 8: Advanced Algorithms

Class 9: Unstructured Data Analysis

Class 10: API, Data Scraping and Class Review

Statistical Analyses and Visualization with GPT

Reading a document online

I have prepared an online CSV file, which is available [here](#).
Alternatively, its URL can be found here:

https://ximarketing.github.io/data/python_data1.csv

Now, let's ask GPT about how to load this CSV file to python.

Which LLM should I use?

HKU has subscribed to a few GPT models:

Login [here](https://chatgpt.hku.hk/) to use ChatGPT: <https://chatgpt.hku.hk/>

- GPT 4o > GPT 4 > GPT 3.5.
- GPT 4o can handle images while other versions cannot.

You can also use [POE](https://poe.com/) assistant here: <https://poe.com/>

- POE assistant can also handle images as input

You can also use [DeepSeek](https://www.deepseek.com) here: <https://www.deepseek.com>

- DeepSeek can also be used to process images as input



Xi Li

I want to read the data https://ximarketing.github.io/data/python_data1.csv to python. Show me the code

17/02/2025, 09:16:59



GPT-3.5 (Tokens used 174)

You can use the following Python code to read data from a CSV file hosted on a website:

```
import pandas as pd

url = "https://ximarketing.github.io/data/python_data1.csv"
df = pd.read_csv(url)

print(df)
```

Make sure you have the `pandas` library installed in your Python environment to run this code. You can install it using `pip install pandas`. This code will read the data from the CSV file at the specified URL and store it in a pandas DataFrame called `df`.

17/02/2025, 09:16:59

From your disk

Xi Li

I want to read a csv file to python from my disk. Show me the code

19/02/2025, 17:30:36

 GPT-3.5 (Tokens used 191)

Here is a sample code that shows how to read a CSV file into Python using the pandas library:

```
import pandas as pd

# Replace 'file_path.csv' with the path to your CSV file
file_path = 'file_path.csv'

# Read the CSV file into a DataFrame
df = pd.read_csv(file_path)

# Display the contents of the DataFrame
print(df)
```



From your disk

Xi Li

Can you give me an example for path?

19/02/2025, 17:30:55

 **GPT-3.5** (Tokens used 300)

Sure! Here is an example of a file path to a CSV file on Windows:

```
file_path = 'C:\\Users\\YourUsername\\Documents\\data.csv'
```

And here is an example of a file path to a CSV file on macOS or Linux:

```
file_path = '/Users/YourUsername/Documents/data.csv'
```

Make sure to replace 'YourUsername' with your actual username and 'data.csv' with the actual name of your CSV file.

19/02/2025, 17:30:55





```
1 import pandas as pd
2 url = "https://ximarketing.github.io/data/python_data1.csv"
3 df = pd.read_csv(url)
4 print(df)
```

Plotting the relationship between age and credit_score

Xi Li

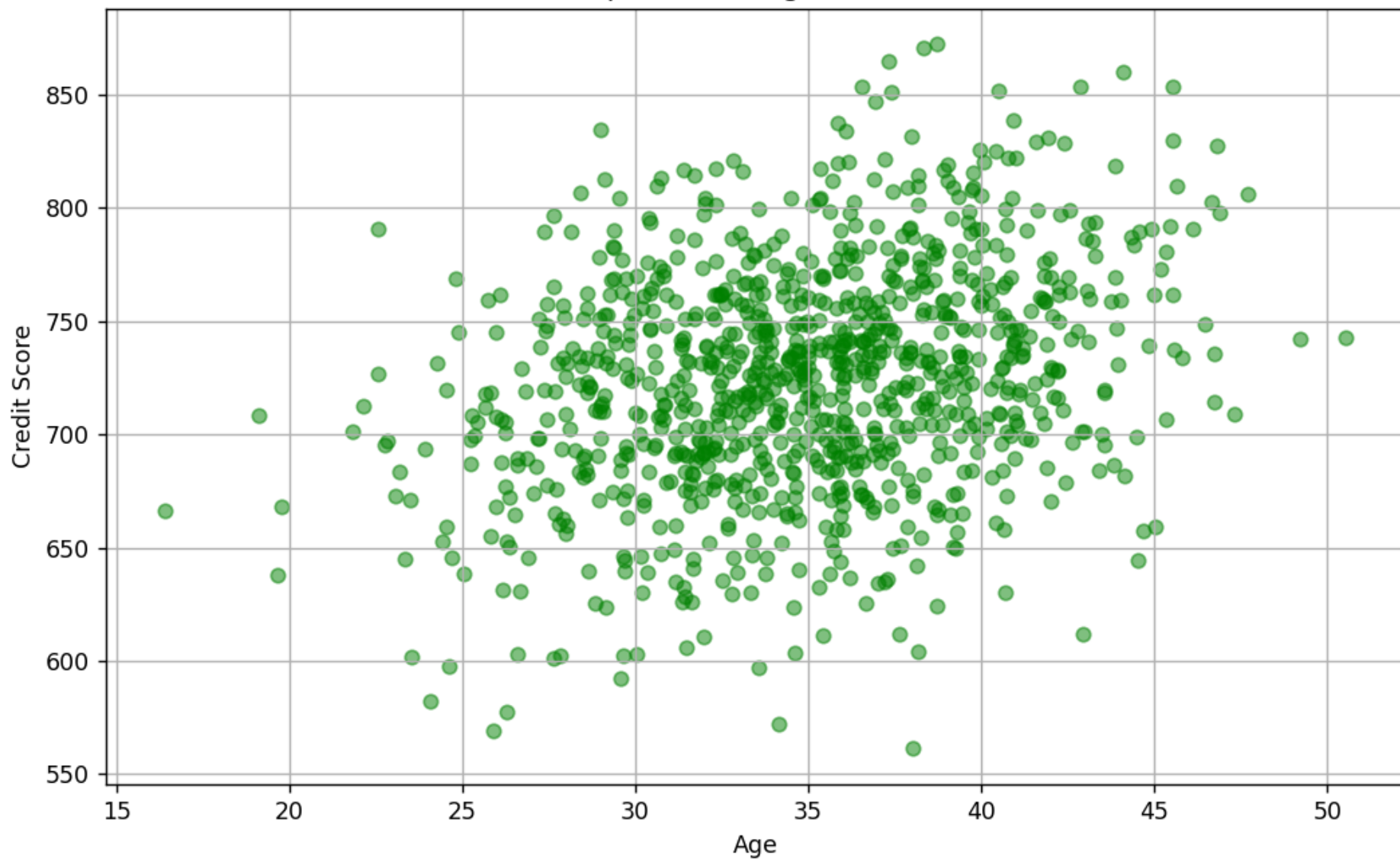
Plotting the relationship between age and credit_score

17/02/2025, 09:23:56

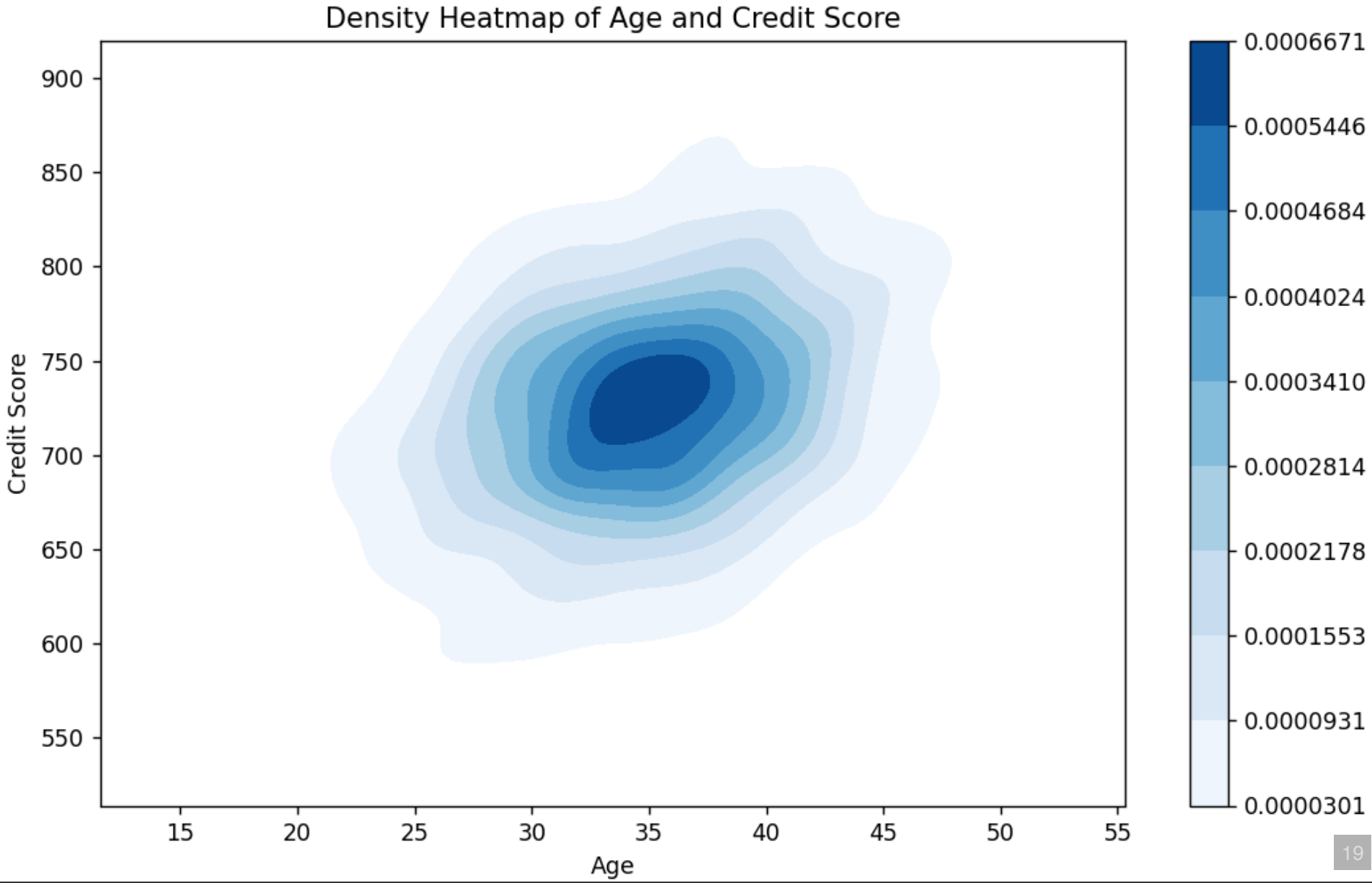


```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 url = "https://ximarketing.github.io/data/python_data1.csv"
5 df = pd.read_csv(url)
6
7 plt.figure(figsize=(10, 6))
8 plt.scatter(df['age'], df['credit_score'], color='green', alpha=0.5)
9 plt.title('Relationship between Age and Credit Score')
10 plt.xlabel('Age')
11 plt.ylabel('Credit Score')
12 plt.grid(True)
13 plt.show()
```

Relationship between Age and Credit Score



You can also ask for a density heatmap.



Code for generating density heatmap:



```
1 import pandas as pd
2 import seaborn as sns
3 import matplotlib.pyplot as plt
4
5 url = "https://ximarketing.github.io/data/python_data1.csv"
6 df = pd.read_csv(url)
7
8 plt.figure(figsize=(10, 6))
9 sns.kdeplot(x=df['age'], y=df['credit_score'], cmap='Blues',
10            shade=True, cbar=True)
11 plt.title('Density Heatmap of Age and Credit Score')
12 plt.xlabel('Age')
13 plt.ylabel('Credit Score')
14 plt.show()
```

Testing the correlation between age and credit score:

Xi Li

I want to test the correlation between age and credit_score. Show me the code

17/02/2025, 09:35:55

Xi Li

I also need the significance of the test

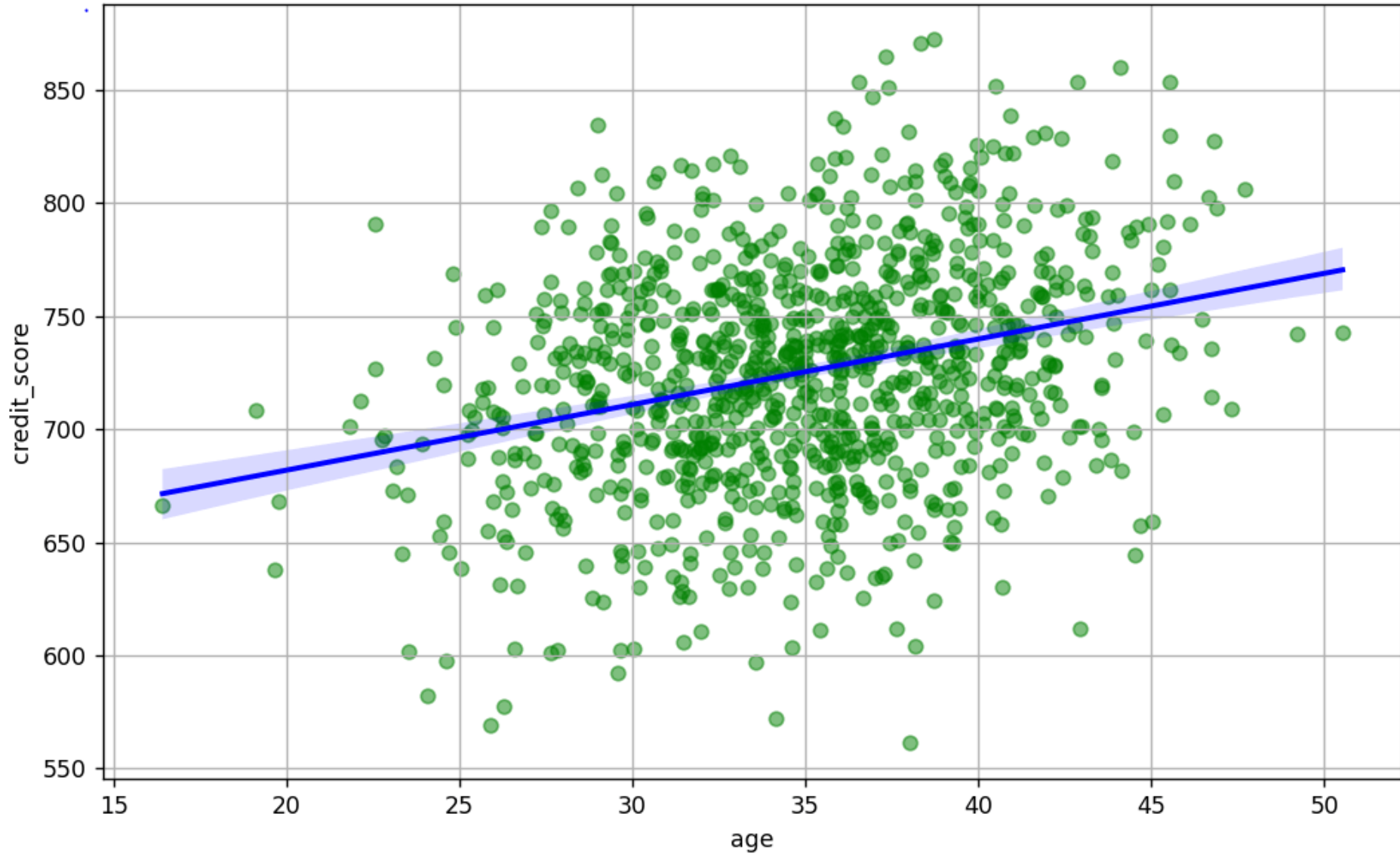
17/02/2025, 09:36:52

```
1 import pandas as pd
2 url = "https://ximarketing.github.io/data/python_data1.csv"
3 df = pd.read_csv(url)
4 correlation = df['age'].corr(df['credit_score'])
5 print("Pearson correlation coefficient between age and
  credit_score:", correlation)
```

Adding a Regression Line

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 url = "https://ximarketing.github.io/data/python_data1.csv"
5 df = pd.read_csv(url)
6 plt.figure(figsize=(10, 6))
7 plt.scatter(df['age'], df['credit_score'], color='green', alpha=0.5)
8 plt.title('Relationship between Age and Credit Score')
9 plt.xlabel('Age')
10 plt.ylabel('Credit Score')
11 plt.grid(True)
12 sns.regplot(x='age', y='credit_score', data=df, scatter=False, color='blue')
13 plt.show()
```

Relationship between Age and Credit Score



Adding regression output

```
Method: Least Squares F-statistic: 93.84
Date: Mon, 17 Feb 2025 Prob (F-statistic): 2.85e-21
Time: 09:50:13 Log-Likelihood: -5288.6
No. Observations: 1000 AIC: 1.058e+04
Df Residuals: 998 BIC: 1.059e+04
Df Model: 1
Covariance Type: nonrobust
```

```
=====
              coef  std err      t  P>|t|  [0.025  0.975]
-----
const      623.9154  10.567   59.042  0.000   603.179   644.652
age         2.9000    0.299    9.687  0.000    2.313    3.487
=====
```

```
=====
Omnibus: 2.290 Durbin-Watson: 2.067
Prob(Omnibus): 0.318 Jarque-Bera (JB): 2.174
Skew: -0.110 Prob(JB): 0.337
Kurtosis: 3.063 Cond. No. 246.
=====
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

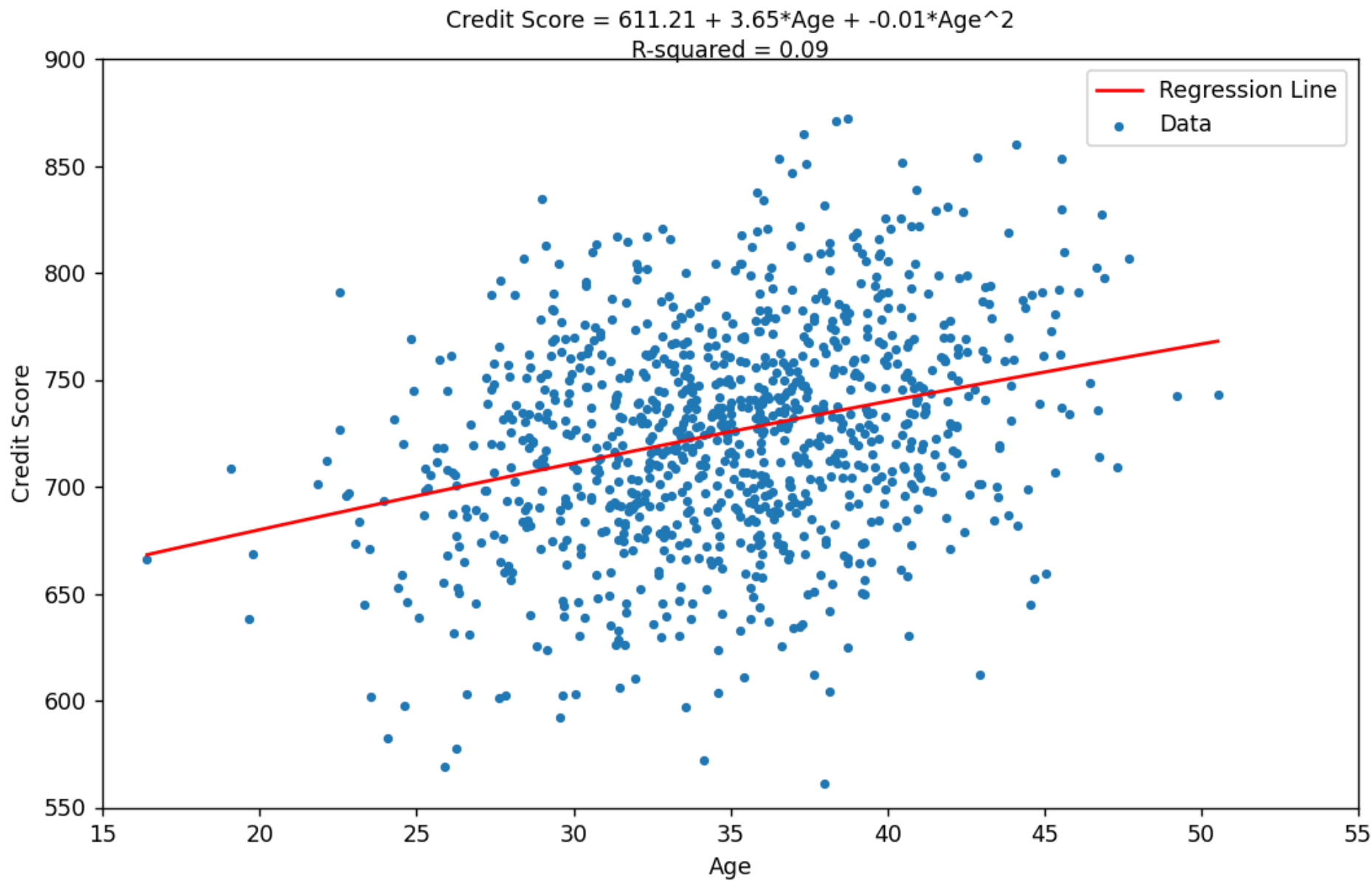
Let 's Try Quadratic Regression Now!

$$\text{credit score}_i = \alpha + \beta_1 \times \text{age}_i + \beta_2 \times \text{age}_i^2$$

```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import statsmodels.api as sm
5 url = "https://ximarketing.github.io/data/python_data1.csv"
6 data = pd.read_csv(url)
7 data['age_squared'] = data['age'] ** 2
8 X = data[['age', 'age_squared']]
9 X = sm.add_constant(X) # Add a constant term for the intercept
10 y = data['credit_score']
11 model = sm.OLS(y, X).fit()
12 print(model.summary())
13 plt.figure(figsize=(10, 6))
14 X_pred = np.linspace(data['age'].min(), data['age'].max(), 100)
15 X_pred_squared = X_pred ** 2
16 X_pred_combined = np.column_stack((np.ones_like(X_pred), X_pred, X_pred_squared))
17 y_pred = model.predict(X_pred_combined)
18 plt.plot(X_pred, y_pred, color='red', label='Regression Line')
19 plt.scatter(data['age'], data['credit_score'], label='Data', s=10)
20 equation = f'Credit Score = {model.params[0]:.2f} + {model.params[1]:.2f}*Age + {model.params[2]:.2f}*Age^2'
21 r_squared = f'R-squared = {model.rsquared:.2f}'
22 plt.text(0.5, 1.05, equation, ha='center', va='center', transform=plt.gca().transAxes,
23         fontsize=10)
24 plt.text(0.5, 1.01, r_squared, ha='center', va='center', transform=plt.gca().transAxes,
25         fontsize=10)
26 plt.xlabel('Age')
27 plt.ylabel('Credit Score')

```



Exercise 1:

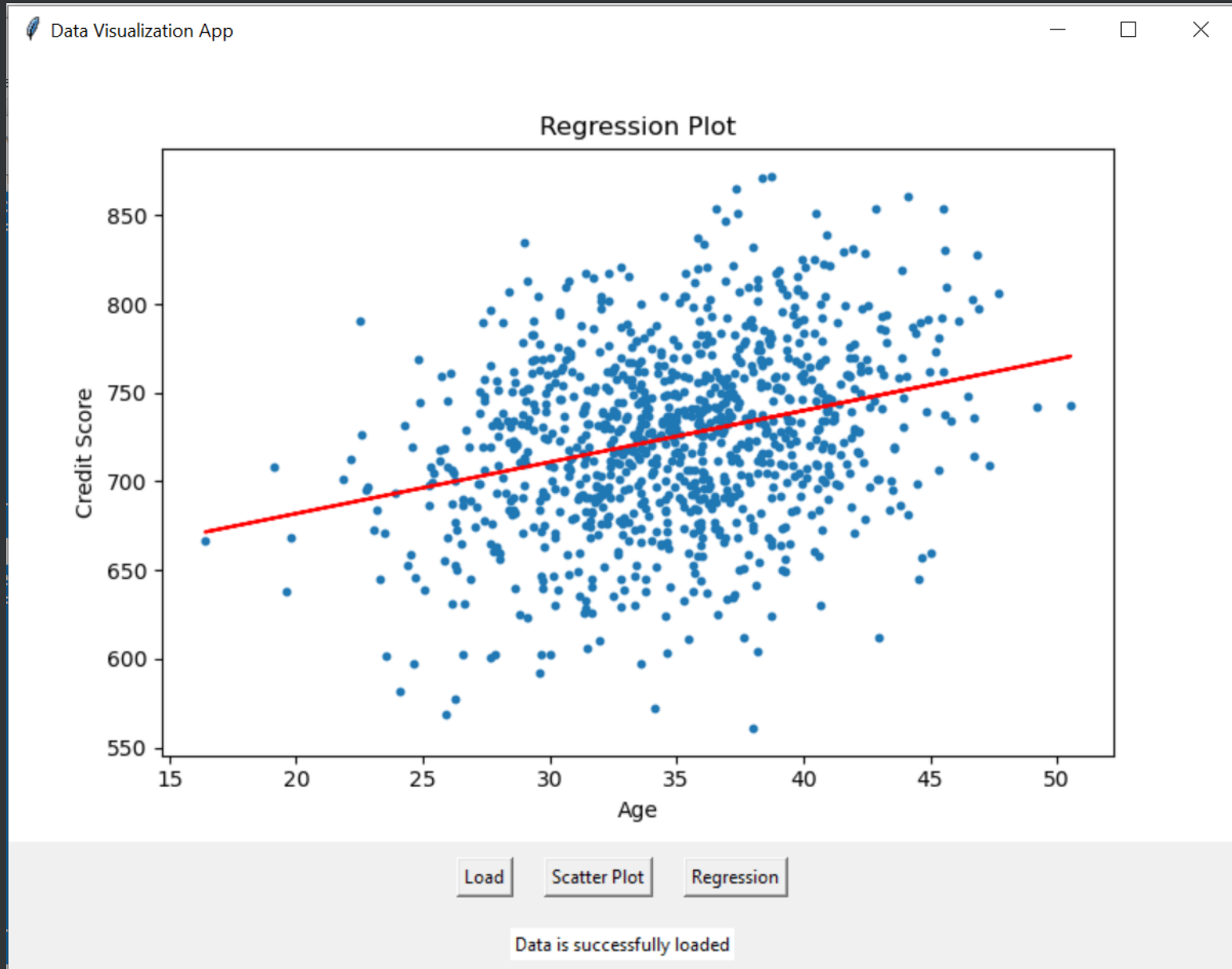
Use GPT to design a desktop APP to visualize your scatter plot and regression!

Exercise 1:

Use GPT to design a desktop APP to visualize your scatter plot and regression!

Note: Creating a mobile APP is possible but much more complex. You also need different APPs for iOS and Android. When creating the desktop APP, it would be easy to use the tkinter module (you can inform GPT).

This is my APP!



Exercise 2:

Use GPT to design a desktop APP to allow users to click on the screen to input the points and generate a regression line fitting the points.

```

1 import tkinter as tk
2 import numpy as np
3 from sklearn.linear_model import LinearRegression
4 points = []
5 def generate_point(event):
6     x = event.x
7     y = event.y
8     points.append((x, y))
9     canvas.create_oval(x - 2, y - 2, x + 2, y + 2, fill="red")
10 def perform_regression():
11     X = [point[0] for point in points]
12     y = [point[1] for point in points]
13     X = np.array(X).reshape(-1, 1)
14     y = np.array(y)
15     model = LinearRegression()
16     model.fit(X, y)
17     y_pred = model.predict(X)
18     for i in range(len(X) - 1):
19         x1, y1 = X[i][0], y_pred[i]
20         x2, y2 = X[i + 1][0], y_pred[i + 1]
21         canvas.create_line(x1, y1, x2, y2, fill="blue")
22     equation = "y = {:.2f}x + {:.2f}".format(model.coef_[0], model.intercept_)
23     canvas.create_text(10, 10, anchor='nw', text=equation, fill='black', font=('Arial', 12))
24 def clear_canvas():
25     canvas.delete("all")
26     del points[:]
27 app = tk.Tk()
28 app.title("Regression App")
29 canvas = tk.Canvas(app, width=800, height=500, bg="white")
30 canvas.grid(row=0, column=0, columnspan=2, padx=10, pady=10)
31 canvas.bind("<Button-1>", generate_point) # Bind left mouse click to generate_point function
32 regression_button = tk.Button(app, text="Regression", command=perform_regression)
33 regression_button.grid(row=1, column=0, padx=10, pady=10)
34 clear_button = tk.Button(app, text="Clear", command=clear_canvas)
35 clear_button.grid(row=1, column=1, padx=10, pady=10)
36 app.mainloop()

```

t-test: What is a *t*-test?

In a *t*-test, we try to compare whether two groups have the same mean. For example, suppose that we want to know whether HKU graduates, on average, make higher salaries than CUHK graduates do. In this case, we would like to compare the means of HKU graduates and CUHK graduates.

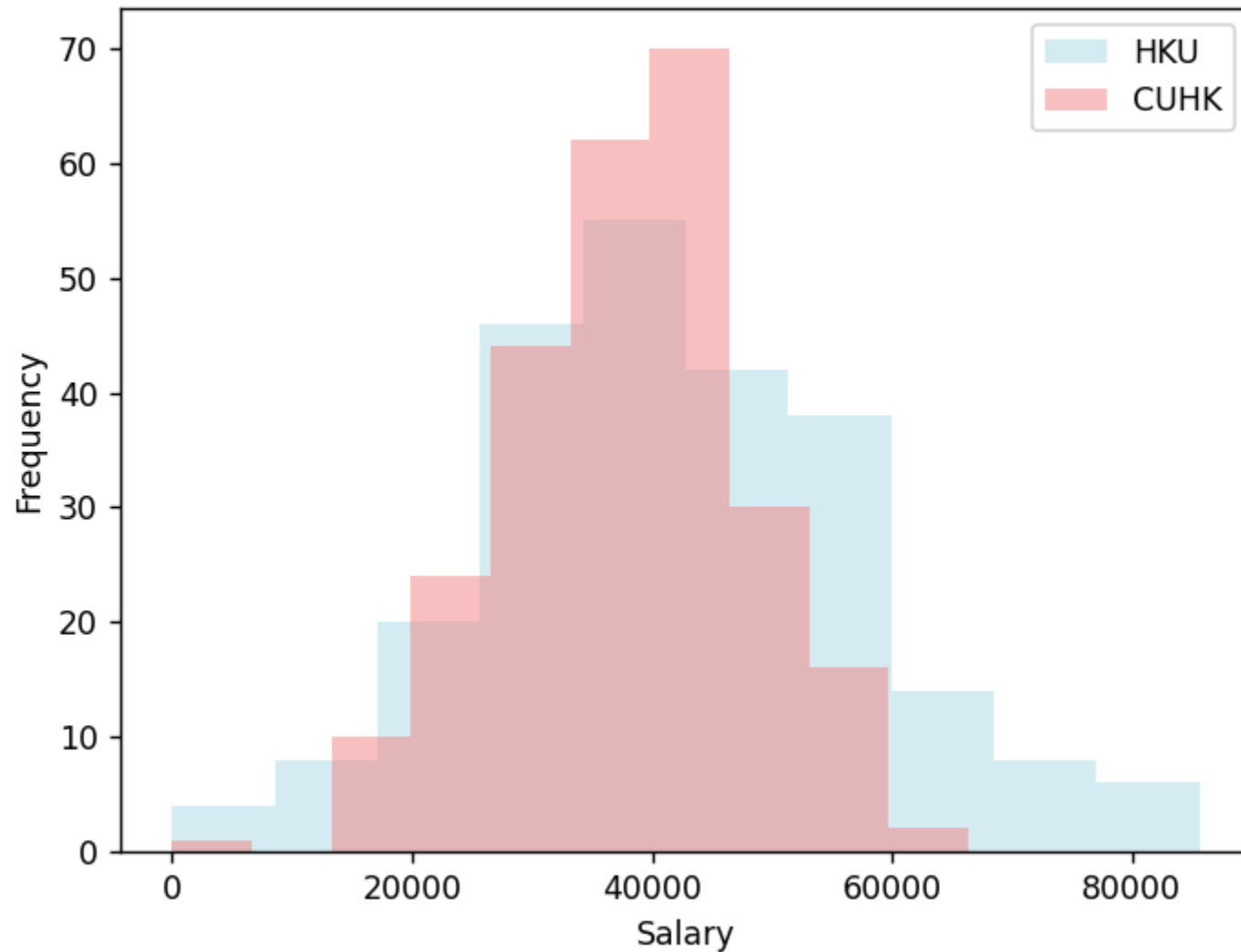
The dataset is [here](https://ximarketing.github.io/data/python_data2.csv):

https://ximarketing.github.io/data/python_data2.csv

Let's view the histogram first (with the help of GPT)!

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 data = pd.read_csv("https://ximarketing.github.io/data/python_data2.csv")
4 salary_hku = data[data['School'] == 'HKU']['Salary']
5 salary_cuhk = data[data['School'] == 'CUHK']['Salary']
6 plt.hist(salary_hku, bins=10, alpha=0.5, label='HKU', color='lightblue')
7 plt.hist(salary_cuhk, bins=10, alpha=0.5, label='CUHK',
8          color='lightcoral')
9 plt.xlabel('Salary')
9 plt.ylabel('Frequency')
10 plt.title('Salary Histogram of HKU and CUHK')
11 plt.legend()
12 plt.show()
```

Salary Histogram of HKU and CUHK



Ask GPT to help you generate the code for running a t -test to compare the salaries of the two schools!

The code I got (with a bit modification):

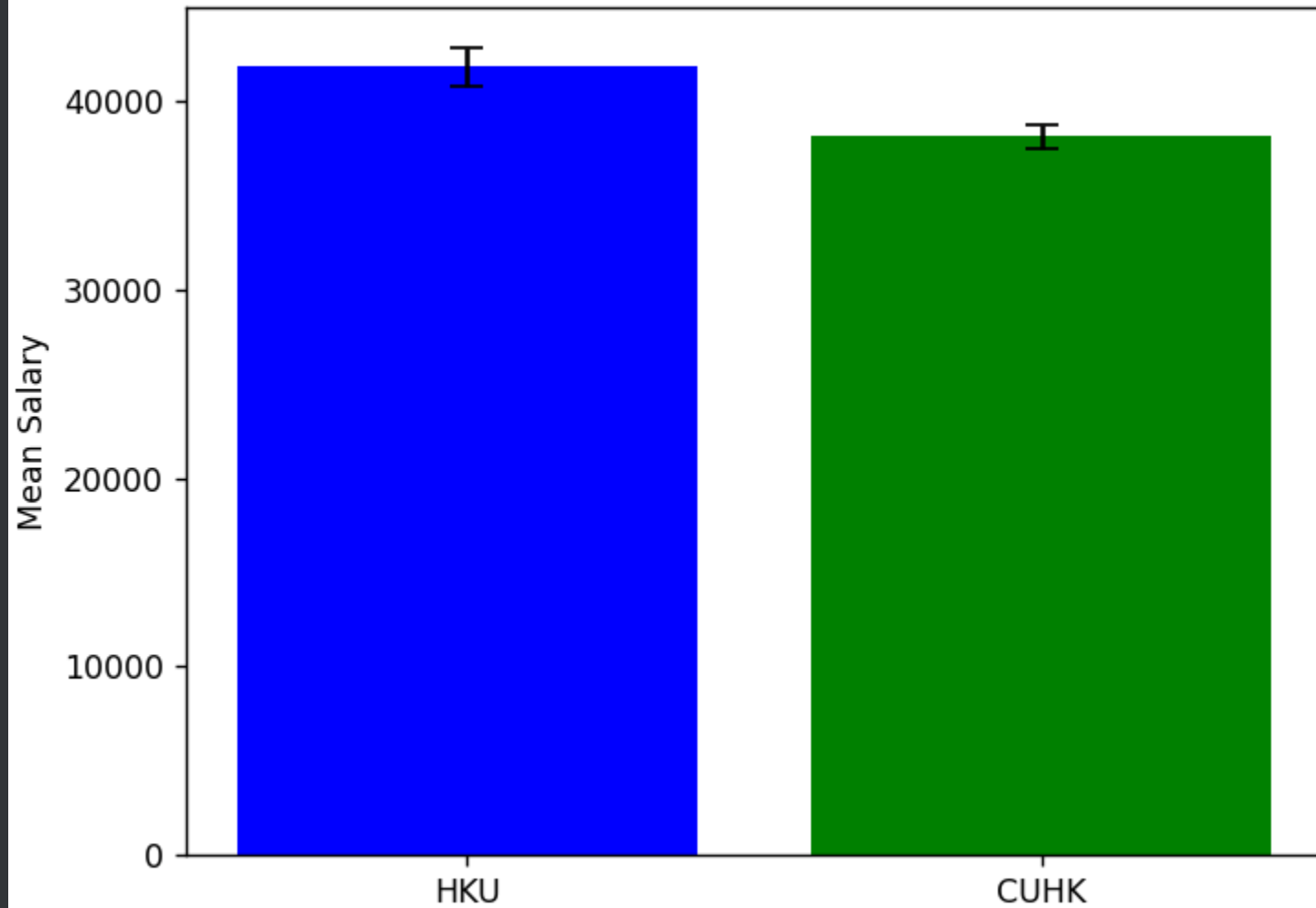


```
1 import pandas as pd
2 from scipy import stats
3 # Load the data from the CSV file
4 data = pd.read_csv("https://ximarketing.github.io/data/python_data2.csv")
5 # Separate the salary data for HKU and CUHK graduates
6 hku_salaries = data[data['School'] == 'HKU']['Salary']
7 cuhk_salaries = data[data['School'] == 'CUHK']['Salary']
8 # Calculate the means
9 hku_mean = hku_salaries.mean()
10 cuhk_mean = cuhk_salaries.mean()
11 # Perform t-test
12 t_statistic, p_value = stats.ttest_ind(hku_salaries, cuhk_salaries)
13 print("Mean salary for HKU graduates:", hku_mean)
14 print("Mean salary for CUHK graduates:", cuhk_mean)
15 print("T-Statistic:", t_statistic)
16 print("P-Value:", p_value)
```

Asking for visualization (including standard error)

```
1 import pandas as pd
2 from scipy import stats
3 import matplotlib.pyplot as plt
4 import numpy as np
5 data = pd.read_csv("https://ximarketing.github.io/data/python_data2.csv")
6 hku_salaries = data[data['School'] == 'HKU']['Salary']
7 cuhk_salaries = data[data['School'] == 'CUHK']['Salary']
8 hku_mean = hku_salaries.mean()
9 cuhk_mean = cuhk_salaries.mean()
10 hku_std = hku_salaries.std()
11 cuhk_std = cuhk_salaries.std()
12 hku_se = hku_std / np.sqrt(len(hku_salaries))
13 cuhk_se = cuhk_std / np.sqrt(len(cuhk_salaries))
14 t_statistic, p_value = stats.ttest_ind(hku_salaries, cuhk_salaries)
15 plt.bar(['HKU', 'CUHK'], [hku_mean, cuhk_mean], yerr=[hku_se, cuhk_se],
16         capsize=5, color=['blue', 'green'])
17 plt.ylabel('Mean Salary')
18 plt.title('Mean Salary for HKU and CUHK Graduates with Standard Errors')
19 plt.show()
```

Mean Salary for HKU and CUHK Graduates with Standard Errors



Exercise:

Use GPT to design a desktop APP. The APP takes input from the user (the values of two groups), runs the t -test and outputs the result to the user.

χ^2 test: What is a χ^2 test?

Can you propose an example of a χ^2 test?

Example of χ -Square Test

Suppose that we are interested in whether education affects marriage status, and we have collected the following data:

	Primary School	Secondary School	Bachelor	Master	Doctor
Single	12	18	36	19	4
Married	45	59	107	47	8
Separated	11	12	18	2	0
Divorced	12	20	23	5	2

Example of χ -Square Test

What conclusions can you draw from the table?

	Primary School	Secondary School	Bachelor	Master	Doctor
Single	12	18	36	19	4
Married	45	59	107	47	8
Separated	11	12	18	2	0
Divorced	12	20	23	5	2

My code produced by GPT

```
1 import numpy as np
2 from scipy.stats import chi2_contingency
3
4 # Create a 2x2 contingency table (replace the values with your own data)
5 observed_table = np.array([
6     [12,18,36,19,4],
7     [45,59,107,47,8],
8     [11,12,18,2,0],
9     [12,20,23,5,2]])
10
11 # Perform the Chi-square test
12 chi2_stat, p_val, dof, expected_table = chi2_contingency(observed_table)
13
14 print("Chi-square Statistic:", chi2_stat)
15 print("P-value:", p_val)
16 print("Degrees of Freedom:", dof)
17 print("Expected Table:")
18 print(expected_table)
```

```
1 import numpy as np
2 from scipy.stats import chi2_contingency
3
4 # Create a 2x2 contingency table (replace the values with your own data)
5 observed_table = np.array([
6     [12,18,36,19,4],
7     [45,59,107,47,8],
8     [11,12,18,2,0],
9     [12,20,23,5,2]])
10
11 # Perform the Chi-square test
12 chi2_stat, p_val, dof, expected_table = chi2_contingency(observed_table)
13
14 print("Chi-square Statistic:", chi2_stat)
15 print("P-value:", p_val)
16 print("Degrees of Freedom:", dof)
17 print("Expected Table:")
18 print(expected_table)
```

The significance value is $p = 0.206 > 5\%$, implying that “**there is no evidence that education affects your marriage status.**”

χ -Square Test: From Documents

```
1 import numpy as np
2 import pandas as pd
3 from scipy import stats
4 import matplotlib.pyplot as plt
5 import numpy as np
6 data = pd.read_csv('https://ximarketing.github.io/data/python_data3.csv')
7 table = pd.crosstab(data.Gender, columns=data.Purchase)
8 print(table)
9 print(stats.chi2_contingency(table, correction=False))
```

Logistic Regression

What's it?

In short, a logistic regression specifies the following function:

$$\Pr[Y=1] = \frac{\exp(\beta X)}{1 + \exp(\beta X)}$$

Why do we choose this “weird” function?

Exercise: Consider the following dataset
<https://ximarketing.github.io/data/loan.csv>

Age	Income	LoanAmount	CreditScore	MonthsEmployed	InterestRate	LoanTerm	Education	EmploymentType	MaritalStatus	HasMortgage	LoanPurpose	Default
56	85994	50587	520	80	15.23	36	Bachelors	Full-time	Divorced	Yes	Other	0
69	50432	124440	458	15	4.81	60	Masters	Full-time	Married	No	Other	0
46	84208	129188	451	26	21.17	24	Masters	Unemployed	Divorced	Yes	Auto	1
32	31713	44799	743	0	7.07	24	High School	Full-time	Married	No	Business	0
60	20437	9139	633	8	6.51	48	Bachelors	Unemployed	Divorced	No	Auto	0
25	90298	90448	720	18	22.72	24	High School	Unemployed	Single	Yes	Business	1

The data is about the loan default information, where the outcome is Default (1 = default, 0 = no default).

Education includes: high school, masters, bachelors, and PhD

Employment type includes: Full-time, part-time, unemployed, and self-employed

Marital status includes: single, married, and divorced

Loan purpose includes: auto, business, education, home, and other.

Age	Income	LoanAmount	CreditScore	MonthsEmployed	InterestRate	LoanTerm	Education	EmploymentType	MaritalStatus	HasMortgage	LoanPurpose	Default
56	85994	50587	520	80	15.23	36	Bachelors	Full-time	Divorced	Yes	Other	0
69	50432	124440	458	15	4.81	60	Masters	Full-time	Married	No	Other	0
46	84208	129188	451	26	21.17	24	Masters	Unemployed	Divorced	Yes	Auto	1
32	31713	44799	743	0	7.07	24	High School	Full-time	Married	No	Business	0
60	20437	9139	633	8	6.51	48	Bachelors	Unemployed	Divorced	No	Auto	0
25	90298	90448	720	18	22.72	24	High School	Unemployed	Single	Yes	Business	1

We would like to run a logistic regression to predict which type of borrowers are more likely to default in the future.



```
1 import pandas as pd
2 import statsmodels.api as sm
3
4 url = 'https://ximarketing.github.io/data/loan.csv'
5 data = pd.read_csv(url)
6
7 categorical_vars = ['Education', 'EmploymentType', 'MaritalStatus',
8                    'HasMortgage', 'LoanPurpose']
9
10 numerical_vars = ['Age', 'Income', 'LoanAmount', 'CreditScore',
11                  'MonthsEmployed', 'InterestRate', 'LoanTerm']
12
13 X = data[numerical_vars + [col for col in data.columns if
14                           col.startswith(tuple(categorical_vars))]]
15 y = data['Default']
16
17 X = sm.add_constant(X)
18 model = sm.Logit(y, X.astype(float))
19 result = model.fit()
20
21 print(result.summary())
```

How to interpret this result?

Age	-0.0392	0.000	-84.774	0.000	-0.040	-0.038
Income	-8.753e-06	1.7e-07	-51.488	0.000	-9.09e-06	-8.42e-06
LoanAmount	4.229e-06	9.31e-08	45.398	0.000	4.05e-06	4.41e-06
CreditScore	-0.0008	4.09e-05	-18.343	0.000	-0.001	-0.001
MonthsEmployed	-0.0097	0.000	-50.927	0.000	-0.010	-0.009
InterestRate	0.0687	0.001	67.565	0.000	0.067	0.071
LoanTerm	9.458e-05	0.000	0.248	0.804	-0.001	0.001
Education_High School	0.0783	0.018	4.404	0.000	0.043	0.113
Education_Masters	-0.1301	0.018	-7.059	0.000	-0.166	-0.094
Education_PhD	-0.1759	0.019	-9.490	0.000	-0.212	-0.140
EmploymentType_Part-time	0.2816	0.019	14.790	0.000	0.244	0.319
EmploymentType_Self-employed	0.2362	0.019	12.293	0.000	0.199	0.274
EmploymentType_Unemployed	0.4416	0.019	23.672	0.000	0.405	0.478
MaritalStatus_Married	-0.2267	0.016	-14.155	0.000	-0.258	-0.195
MaritalStatus_Single	-0.0644	0.016	-4.139	0.000	-0.095	-0.034
HasMortgage_Yes	-0.1560	0.013	-12.007	0.000	-0.181	-0.130
LoanPurpose_Business	0.0442	0.020	2.193	0.028	0.005	0.084
LoanPurpose_Education	-0.0169	0.020	-0.828	0.408	-0.057	0.023
LoanPurpose_Home	-0.1935	0.021	-9.234	0.000	-0.235	-0.152
LoanPurpose_Other	-0.0071	0.020	-0.346	0.729	-0.047	0.033

Multinomial Logit Model

What's it?

Multinomial Logit Model

In a linear regression, our dependent variable is a number.

In a logistic regression, our dependent variable is a probability, e.g., the probability of success.

In a multinomial logit regression, our dependent variable is a choice, usually a choice among different brands. In the above example, HKU president's choice is one of the four brands, BMW, Tesla, Audi, and Mercedes Benz.

Suppose that consumers have three choices, A, B, C .

Now, given X_i , we would like to come up with three functions $f_A(X_i)$, $f_B(X_i)$ and $f_C(X_i)$, such that

$$\Pr[Y_i = A] \approx f_A(X_i),$$

$$\Pr[Y_i = B] \approx f_B(X_i),$$

$$\Pr[Y_i = C] \approx f_C(X_i).$$

These functions are:

$$f_A(X_i) = \frac{\exp(\alpha_A + \beta_A X_i)}{\exp(\alpha_A + \beta_A X_i) + \exp(\alpha_B + \beta_B X_i) + \exp(\alpha_C + \beta_C X_i)}$$

$$f_B(X_i) = \frac{\exp(\alpha_B + \beta_B X_i)}{\exp(\alpha_A + \beta_A X_i) + \exp(\alpha_B + \beta_B X_i) + \exp(\alpha_C + \beta_C X_i)}$$

$$f_C(X_i) = \frac{\exp(\alpha_C + \beta_C X_i)}{\exp(\alpha_A + \beta_A X_i) + \exp(\alpha_B + \beta_B X_i) + \exp(\alpha_C + \beta_C X_i)}$$

Let's analyze the following data:

<https://ximarketing.github.io/data/bankchoice.csv>

	Choice	Age	Female	Income	Education	Job
1	CCB	62	0	7	2	Industry
2	CCB	34	1	4	5	Retired
3	CCB	68	0	5	2	Industry
4	CCB	60	0	3	2	Education
5	ICBC	18	0	6	2	Industry
6	CCB	18	0	4	3	Student
7	CCB	51	0	7	3	Education
8	CCB	25	1	3	2	Unemployed
9	BOC	42	1	4	4	Education
10	CCB	71	1	5	3	Service
11	BOC	23	0	4	5	Student
12	BOC	30	0	2	5	Retired

The code I got from GPT:



```
1 import pandas as pd
2 import statsmodels.api as sm
3 from patsy import dmatrices
4 data = pd.read_csv('https://ximarketing.github.io/data/bankchoice.csv')
5 y, X = dmatrices('Choice ~ Age + Income + Education + C(Job, Treatment)',
6 data, return_type='dataframe')
7 model = sm.MNLogit(y, X).fit()
8 summary = model.summary2()
9 pd.set_option('display.max_columns', None)
10 pd.set_option('display.width', None)
11 choices = y.design_info.column_names
12 choices.pop()
13 for i, choice in enumerate(choices, start=1):
14     print(f"Choice: {choice}")
15     print(summary.tables[i])
16     print('\n')
```

Do you understand the output?

Choice: Choice[ABC]								
	y = 0	Coef.	Std.Err.	t	P> t	[0.025	0.975]	
Intercept	Intercept	0.473001	0.195306	2.421850	1.544173e-02	0.090209	0.855793	
C(Job, Treatment)[T.Finance]	C(Job, Treatment)[T.Finance]	4.291024	1.009358	4.251242	2.125879e-05	2.312719	6.269329	
C(Job, Treatment)[T.Government]	C(Job, Treatment)[T.Government]	0.520523	0.246418	2.112360	3.465559e-02	0.037553	1.003493	
C(Job, Treatment)[T.Industry]	C(Job, Treatment)[T.Industry]	2.442084	0.518532	4.709615	2.481853e-06	1.425781	3.458387	
C(Job, Treatment)[T.Retired]	C(Job, Treatment)[T.Retired]	-2.563242	0.145518	-17.614645	1.901798e-69	-2.848451	-2.278032	
C(Job, Treatment)[T.Service]	C(Job, Treatment)[T.Service]	-0.042547	0.189207	-0.224869	8.220814e-01	-0.413386	0.328292	
C(Job, Treatment)[T.Student]	C(Job, Treatment)[T.Student]	-1.018968	0.161727	-6.300563	2.965671e-10	-1.335947	-0.701990	
C(Job, Treatment)[T.Unemployed]	C(Job, Treatment)[T.Unemployed]	0.226454	0.182337	1.241956	2.142528e-01	-0.130919	0.583828	
Age	Age	-0.032753	0.002334	-14.031712	9.971443e-45	-0.037328	-0.028178	
Income	Income	0.613154	0.028165	21.769850	4.480917e-105	0.557951	0.668357	
Education	Education	1.272112	0.037794	33.658878	2.312053e-248	1.198036	1.346187	

Choice: Choice[BOC]

		y = 1	Coef.	Std.Err.	t	P> t	[0.025	0.975]
Intercept	Intercept		0.836615	0.193845	4.315904	1.589513e-05	0.456686	1.216543
C(Job, Treatment)[T.Finance]	C(Job, Treatment)[T.Finance]		4.021791	1.009072	3.985632	6.730076e-05	2.044046	5.999537
C(Job, Treatment)[T.Government]	C(Job, Treatment)[T.Government]		0.843469	0.244775	3.445892	5.691779e-04	0.363718	1.323220
C(Job, Treatment)[T.Industry]	C(Job, Treatment)[T.Industry]		2.159684	0.517983	4.169410	3.053894e-05	1.144456	3.174913
C(Job, Treatment)[T.Retired]	C(Job, Treatment)[T.Retired]		-3.167509	0.144007	-21.995543	3.177106e-107	-3.449758	-2.885261
C(Job, Treatment)[T.Service]	C(Job, Treatment)[T.Service]		-0.328630	0.187726	-1.750587	8.001701e-02	-0.696565	0.039305
C(Job, Treatment)[T.Student]	C(Job, Treatment)[T.Student]		-1.415400	0.160125	-8.839354	9.627386e-19	-1.729239	-1.101561
C(Job, Treatment)[T.Unemployed]	C(Job, Treatment)[T.Unemployed]		-0.858125	0.181573	-4.726057	2.289217e-06	-1.214002	-0.502248
Age	Age		-0.017986	0.002319	-7.756240	8.748421e-15	-0.022530	-0.013441
Income	Income		0.882860	0.028079	31.441465	5.491352e-217	0.827825	0.937895
Education	Education		0.809948	0.037629	21.524823	9.116791e-103	0.736197	0.883699

Choice: Choice[CCB]

		y = 2	Coef.	Std.Err.	t	P> t	[0.025	0.975]
Intercept	Intercept		-0.709650	0.216728	-3.274379	1.058945e-03	-1.134429	-0.284871
C(Job, Treatment)[T.Finance]	C(Job, Treatment)[T.Finance]		5.027076	1.013953	4.957901	7.125898e-07	3.039766	7.014387
C(Job, Treatment)[T.Government]	C(Job, Treatment)[T.Government]		2.005531	0.263119	7.622133	2.495179e-14	1.489827	2.521236
C(Job, Treatment)[T.Industry]	C(Job, Treatment)[T.Industry]		4.737702	0.525177	9.021162	1.861047e-19	3.708375	5.767029
C(Job, Treatment)[T.Retired]	C(Job, Treatment)[T.Retired]		-0.591359	0.168584	-3.507807	4.518174e-04	-0.921776	-0.260941
C(Job, Treatment)[T.Service]	C(Job, Treatment)[T.Service]		1.289505	0.209591	6.152477	7.628194e-10	0.878714	1.700296
C(Job, Treatment)[T.Student]	C(Job, Treatment)[T.Student]		0.707630	0.183555	3.855134	1.156664e-04	0.347868	1.067391
C(Job, Treatment)[T.Unemployed]	C(Job, Treatment)[T.Unemployed]		1.207999	0.203195	5.945033	2.764005e-09	0.809745	1.606253
Age	Age		-0.003158	0.002413	-1.308492	1.907064e-01	-0.007888	0.001572
Income	Income		0.517294	0.028628	18.069586	5.532719e-73	0.461184	0.573404
Education	Education		0.209854	0.039220	5.350756	8.758775e-08	0.132985	0.286723

Handwriting Digit Prediction

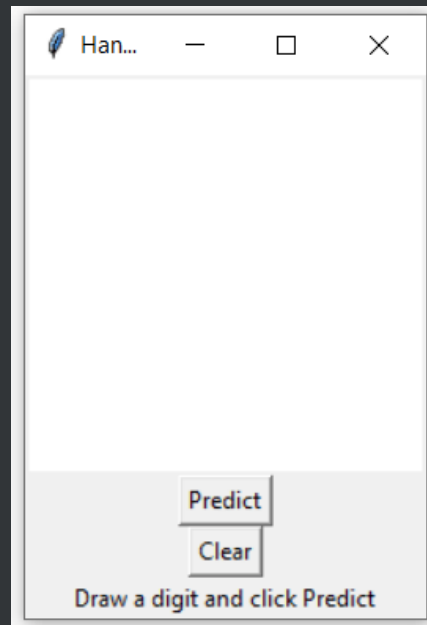
We can further build a handwriting digit prediction model based on multinomial logit regression.

Input: $x_i \in \{0, 1\}$, the color of pixel i , black vs. white.

Output: $Y \in \{0, 1, \dots, 9\}$.

Handwriting Digit Prediction

Let's try the following one, which is created by GPT.



Code for data training

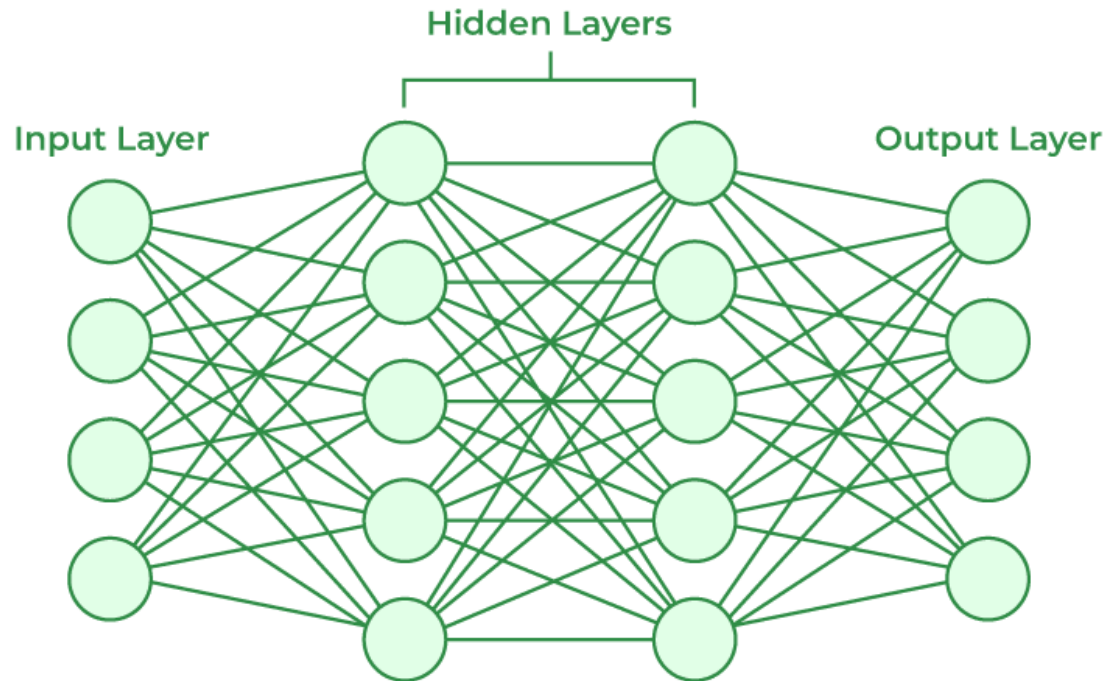
```
1 from sklearn.linear_model import LogisticRegression
2 from sklearn.datasets import fetch_openml
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score
5 import joblib
6 # Load the MNIST dataset
7 mnist = fetch_openml('mnist_784', version=1)
8 # Normalize the data
9 X = mnist.data / 255.0
10 y = mnist.target.astype(int)
11 # Split the data into training and testing sets
12 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
13 # Train the logistic regression model
14 model = LogisticRegression(solver='lbfgs', max_iter=1000, multi_class='multinomial')
15 model.fit(X_train, y_train)
16 # Evaluate the model
17 y_pred = model.predict(X_test)
18 accuracy = accuracy_score(y_test, y_pred)
19 print(f'Accuracy: {accuracy * 100:.2f}%')
20 # Save the model
21 joblib.dump(model, 'logistic_regression_model.pkl')
22 import tkinter as tk
23 from tkinter import *
24 import numpy as np
25 from PIL import Image, ImageDraw, ImageOps
26 import joblib
27 # Load the trained model (make sure you have a trained model saved as 'logistic_regression_model.pkl')
28 model = joblib.load('logistic_regression_model.pkl')
```

Artificial Neural Network

Artificial Neural Network

- Logistic regression is the extension of linear regression.
- Multinomial logit model is the extension of logistic regression.
- Artificial neural network (and deep neural network) is the extension of multinomial logit model.

Artificial Neural Network



<https://www.youtube.com/embed/bfmFfD2RIcg?enablejsapi=1>

Artificial Neural Network

An artificial neural network has a few layers:

- Input layer: Which takes input from data x_i
- Output layer: Which produces an out $\Pr[y_i = 1]$
- Hidden layers: There are one or more hidden layers which are responsible for making calculations (i.e., logistic function).

Artificial Neural Network

Why do we create artificial neural networks like this?

This is because human brains also work in similar ways! We also have cells in our brain which calculates logistic function! So, an artificial neural network simply mimics our human brain.

Artificial Neural Network

In mathematics, there is a famous “universal approximation theorem,” which states that, if you have an artificial neural work that is large enough, theoretically, you should be able to use it to approximate any functions...

What can we do?

- Autonomous driving
- Play GO
- Digit recognition...

Artificial Neural Network

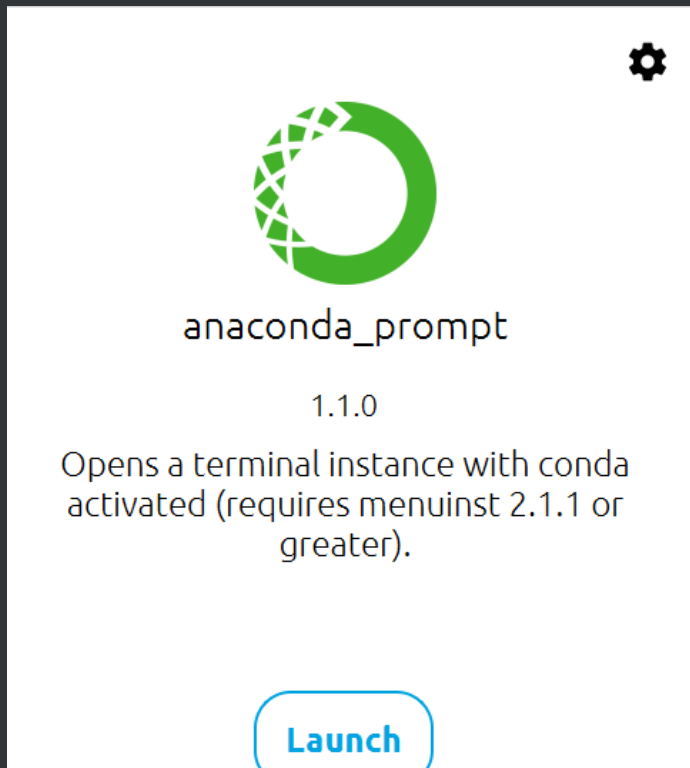
Again, let us build an artificial neural network in python for digit recognition with the help of GPT!

Artificial Neural Network

Again, let us build an artificial neural network in python for digit recognition with the help of GPT!

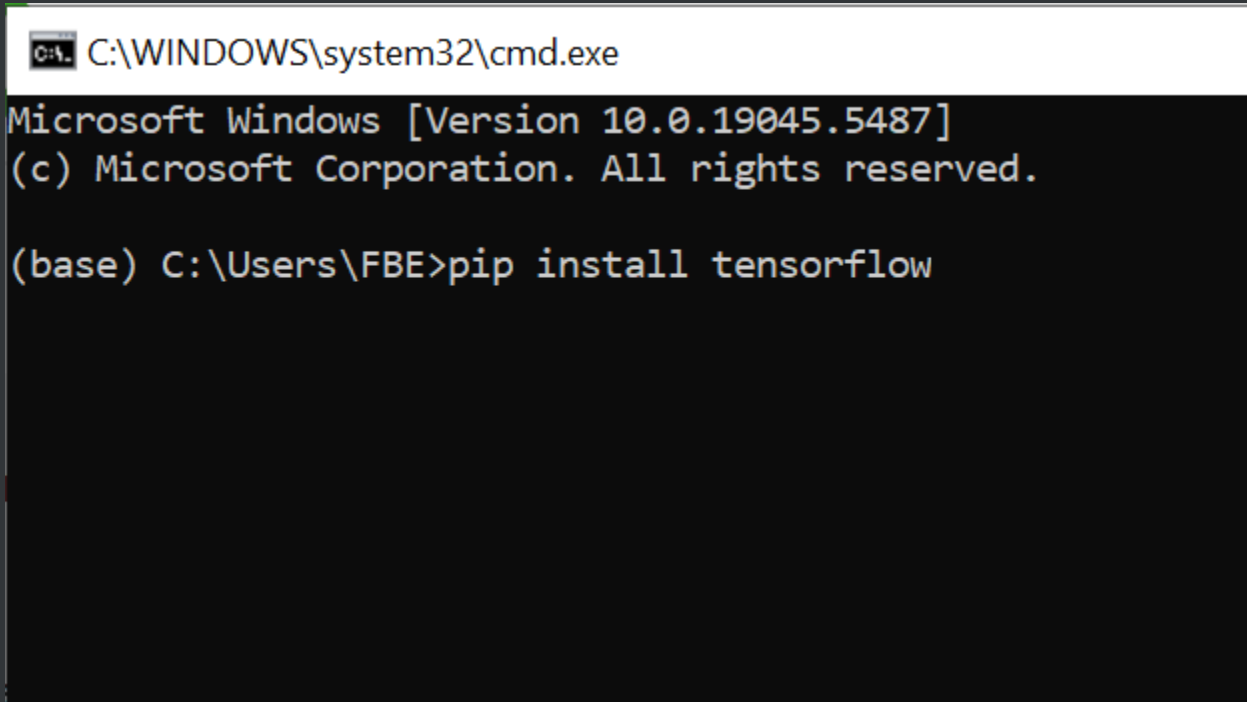
Note: You will need to use the module tensorflow for your ANN. The installation of tensorflow is a bit different.

Installing TensorFlow



On your Anaconda, lunch
anaconda prompt

Installing TensorFlow

A screenshot of a Windows Command Prompt window. The title bar at the top reads "C:\WINDOWS\system32\cmd.exe". The window content shows the following text: "Microsoft Windows [Version 10.0.19045.5487]" followed by "(c) Microsoft Corporation. All rights reserved." on the next line. Below that, the prompt "(base) C:\Users\FBE>" is followed by the command "pip install tensorflow".

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.5487]
(c) Microsoft Corporation. All rights reserved.

(base) C:\Users\FBE>pip install tensorflow
```

Input "pip install tensorflow", and enter the command

Pie Chart

Can you create a pie chart that visualize the sales of different EV brands?



比亞迪 (BYD) 848



特斯拉 (Tesla) 202



極氪 (Zeekr) 195

寶馬 (BMW) 144

上汽名爵 (MG) 92

小鵬 (XPena) 57

上汽大通 (Maxus) 55

騰勢 (DENZA) 50

資料來源：
香港電動車資料庫

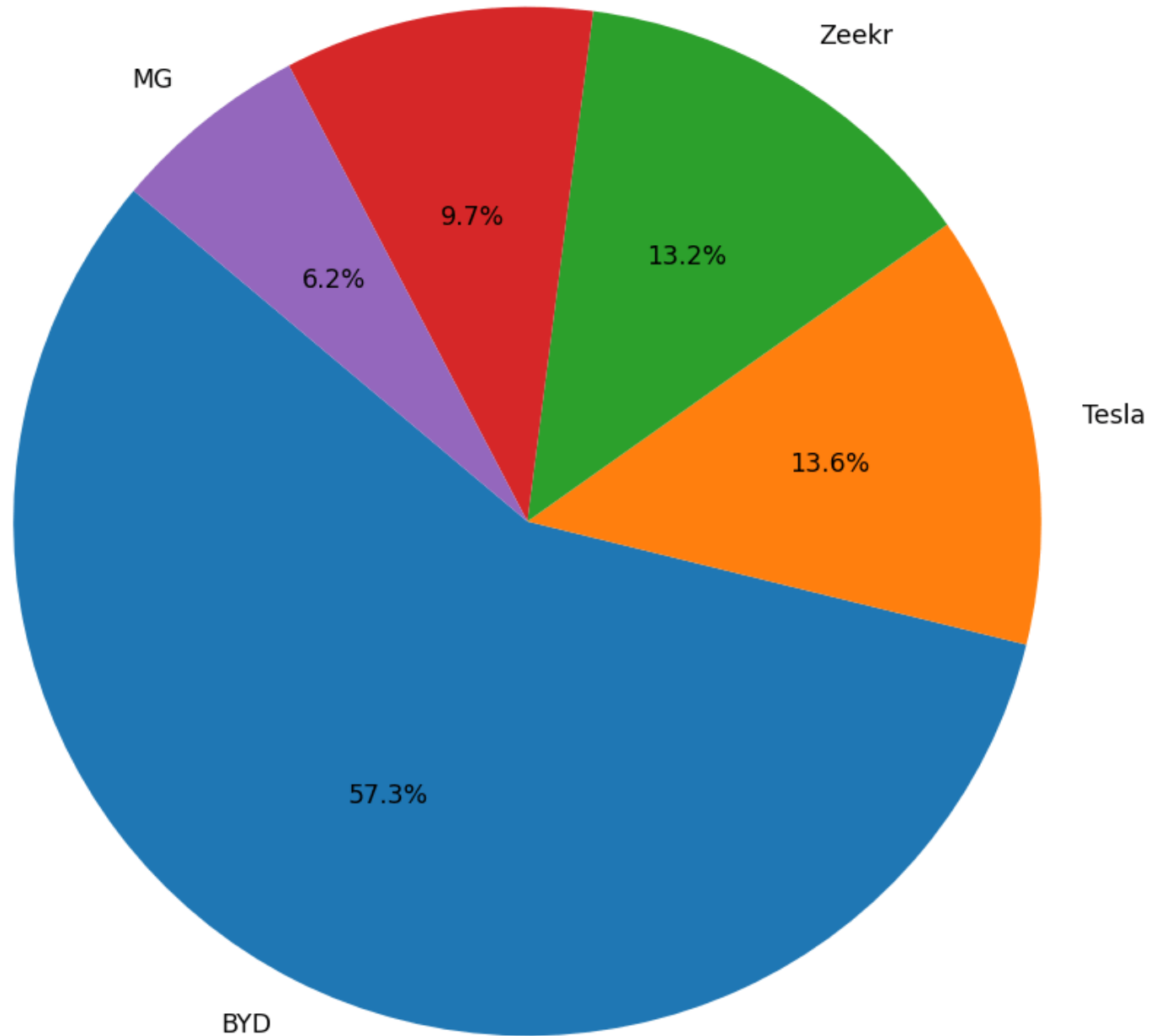
獲 35% 車主選擇、Tesla 暴跌近 8 成

比亞迪奪香港私家車銷量第 1 位

Code generated by GPT

```
1 import matplotlib.pyplot as plt
2 # Data for car sales
3 brands = ['BYD', 'Tesla', 'Zeekr', 'BMW', 'MG']
4 sales = [848, 202, 195, 144, 92]
5 # Create a pie chart
6 plt.figure(figsize=(8, 8))
7 plt.pie(sales, labels=brands, autopct='%1.1f%%', startangle=140)
8 plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a
   circle.
9 # Add a title
10 plt.title('Car Sales Distribution')
11 # Display the pie chart
12 plt.show()
```

Car Sales Distribution



I am making it more interactive (plotly needed)

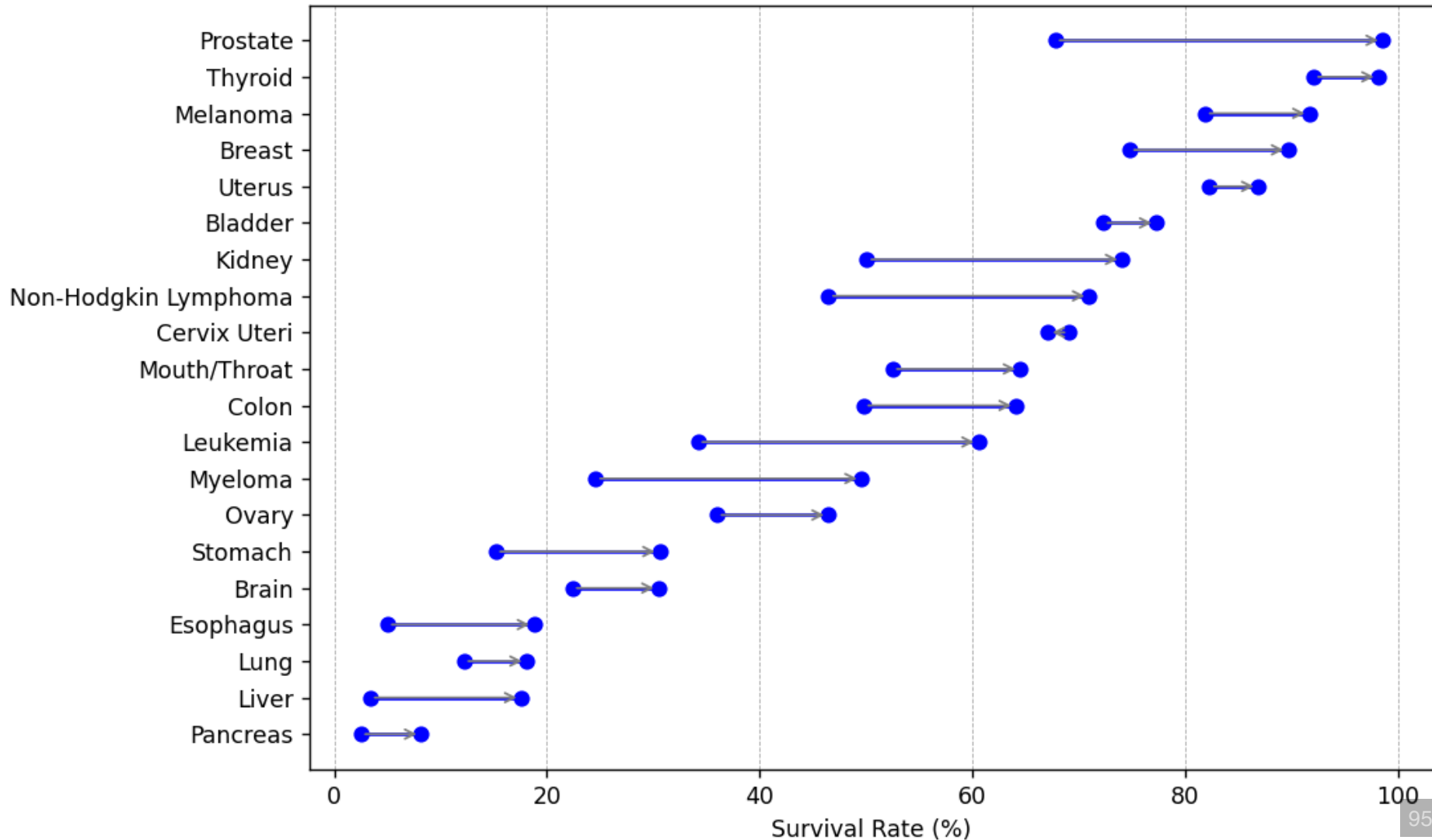


```
1 import plotly.graph_objects as go
2 # Data for car sales
3 brands = ['BYD', 'Tesla', 'Zeekr', 'BMW', 'MG']
4 sales = [848, 202, 195, 144, 92]
5 # Create a pie chart trace
6 fig = go.Figure(data=[go.Pie(labels=brands, values=sales,
7     hoverinfo='label+percent', textinfo='value', pull=[0, 0, 0, 0, 0])])
8 # Update layout for interactivity
9 fig.update_layout(title='Car Sales Distribution', showlegend=False)
10 # Show the figure
11 fig.show()
```

Dumbbell chart

Try to create a dumbbell chart like this one!

Cancer Survival Rate Comparison (1970 vs. 2010)



The data is available [here](http://ximarketing.github.io/data/cancer.csv):

<http://ximarketing.github.io/data/cancer.csv>

Try to use GPT to create the dumbbell chart for you!

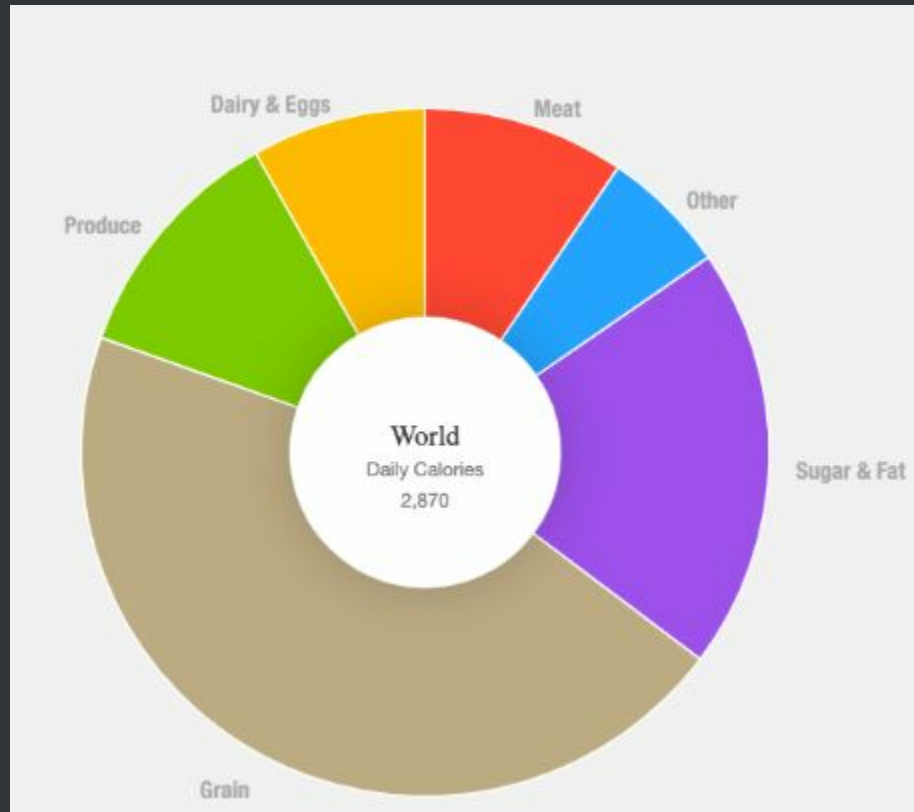
```

1 import matplotlib.pyplot as plt
2 import pandas as pd
3 data = pd.read_csv('https://ximarketing.github.io/data/cancer.csv')
4 # Sort the data based on 2010 survival rate
5 data_sorted = data.sort_values(by='2010', ascending=False)
6 data_sorted = data_sorted.iloc[::-1]
7 data_sorted.reset_index(drop=True, inplace=True)
8 # Plotting
9 plt.figure(figsize=(10, 6))
10 plt.subplots_adjust(left=0.22) # Adjust the left margin
11 for i, row in data_sorted.iterrows():
12     plt.plot([row['2010'], row['1970']], [i, i], marker='o', color='b') #
    Dumbbell circles
13     plt.annotate('', xy=(row['2010'], i), xytext=(row['1970'], i),
14                 arrowprops=dict(arrowstyle='->', connectionstyle='arc3',
    color='gray')) # Arrows
15 # Adding labels and title
16 plt.yticks(range(len(data_sorted)), data_sorted['Cancer'])
17 plt.xlabel('Survival Rate (%)')
18 plt.ylabel('Cancers')
19 plt.title('Cancer Survival Rate Comparison (1970 vs. 2010)')
20 # Showing the plot
21 plt.grid(axis='x', linestyle='--', linewidth=0.5)
22 plt.show()

```

Generating Visualization by Uploading Images

Example: Generate a Donut Chart
Image can be downloaded [here](#)





Xi Li

Give me the python code to produce this donut chart



donut.JPG
JPG 14.71 KB



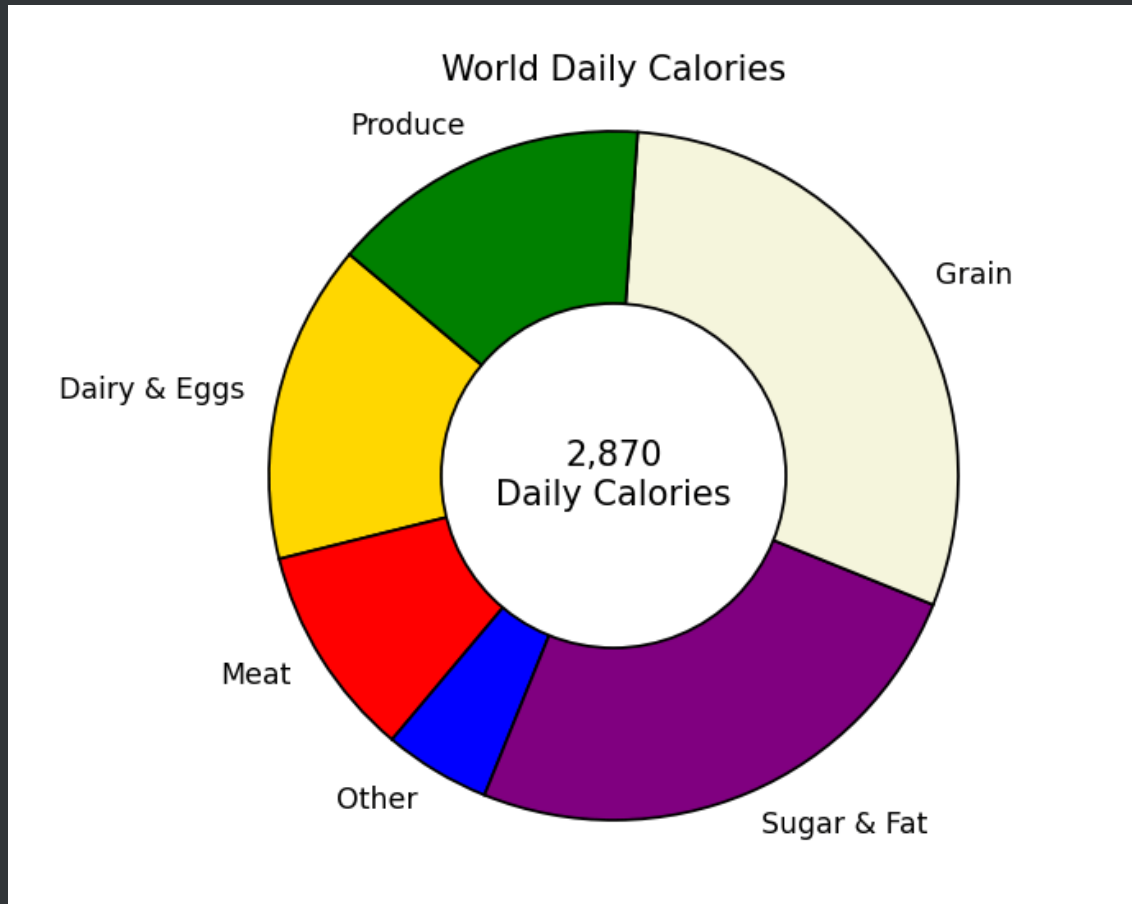
19/02/2025, 08:51:42



GPT-4o (Tokens used 685)

Certainly! Below is an example of how to create a donut chart similar to the one in the image using Python with the help of the `matplotlib` library.

My Output from GPT



Example: Generate a Population Pyramid

Image can be downloaded [here](#)

Xi Li

I want to generate a population pyramid like this in Python. Give me the code



population.JPG
JPG 98.34 KB



18/02/2025, 21:11:25



GPT-4o (Tokens used 1370)

Sure! To generate a population pyramid similar to the one you provided using Python, you can use the `matplotlib` library. Here is an example code:





```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 # Sample data similar to the provided population pyramid
4 age_groups = ['0-4', '5-9', '10-14', '15-19', '20-24', '25-29', '30-34', '35-39',
5               '40-44', '45-49', '50-54', '55-59', '60-64', '65-69', '70-74',
6               '75-79', '80-84', '85-89', '90-94', '95-99', '100+']
7 male_population = [20000, 18000, 17000, 16000, 15000, 14000, 13000, 12000,
8                   11000, 10000, 9000, 8000, 7000, 6000, 5000, 4000,
9                   3000, 2000, 1000, 500, 100]
10 female_population = [19000, 17000, 16000, 15000, 14000, 13000, 12000,
11                     11000, 10000, 9000, 8000, 7000, 6000, 5000, 4000,
12                     3000, 2000, 1000, 500, 300, 100]
13 # Convert the populations to numpy arrays for easier manipulation
14 male_population = np.array(male_population)
15 female_population = np.array(female_population)
16 # Create the figure and axis
17 fig, ax = plt.subplots(figsize=(10, 8))
18 # Plot the male and female bars
19 ax.barh(age_groups, male_population, color='blue', label='Males')
20 ax.barh(age_groups, -female_population, color='red', label='Females')
21 # Add labels and title
22 ax.set_xlabel('Population')
23 ax.set_title('Population Pyramid for Hong Kong, December 2020')
24 ax.legend()
25 # Customize the x-axis to show positive numbers for female population
26 ax.set_xticks(np.arange(-20000, 21000, 5000))
27 ax.set_xticklabels([str(abs(x)) for x in np.arange(-20000, 21000, 5000)])
28 # Show the plot
29 plt.show()
```

How about an animated GIF?

Image can be downloaded [here](#)

Xi Li

I want a dynamic GIF like this showing how population pyramid changes over time. Show me python code

 7_pew_research_...
GIF 188.85 KB 

18/02/2025, 21:58:06



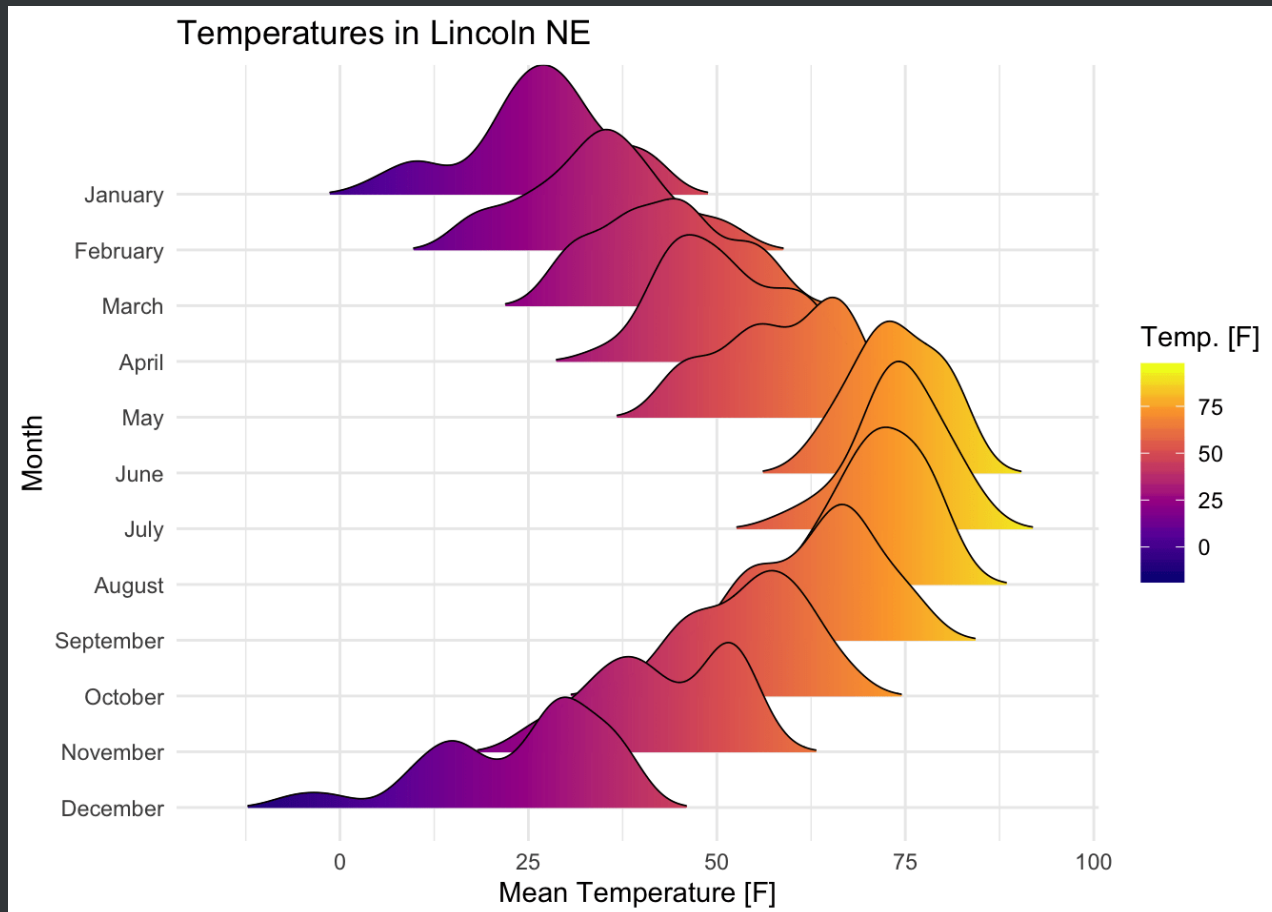
GPT-4o (Tokens used 1219)

Creating a dynamic GIF like the one you've shown requires several steps, including data preparation, plotting the population pyramids for each year, and then combining these plots into an animated GIF. Here's an example of how you can achieve this using Python with libraries such as `matplotlib`, `pandas`, and `imageio`.

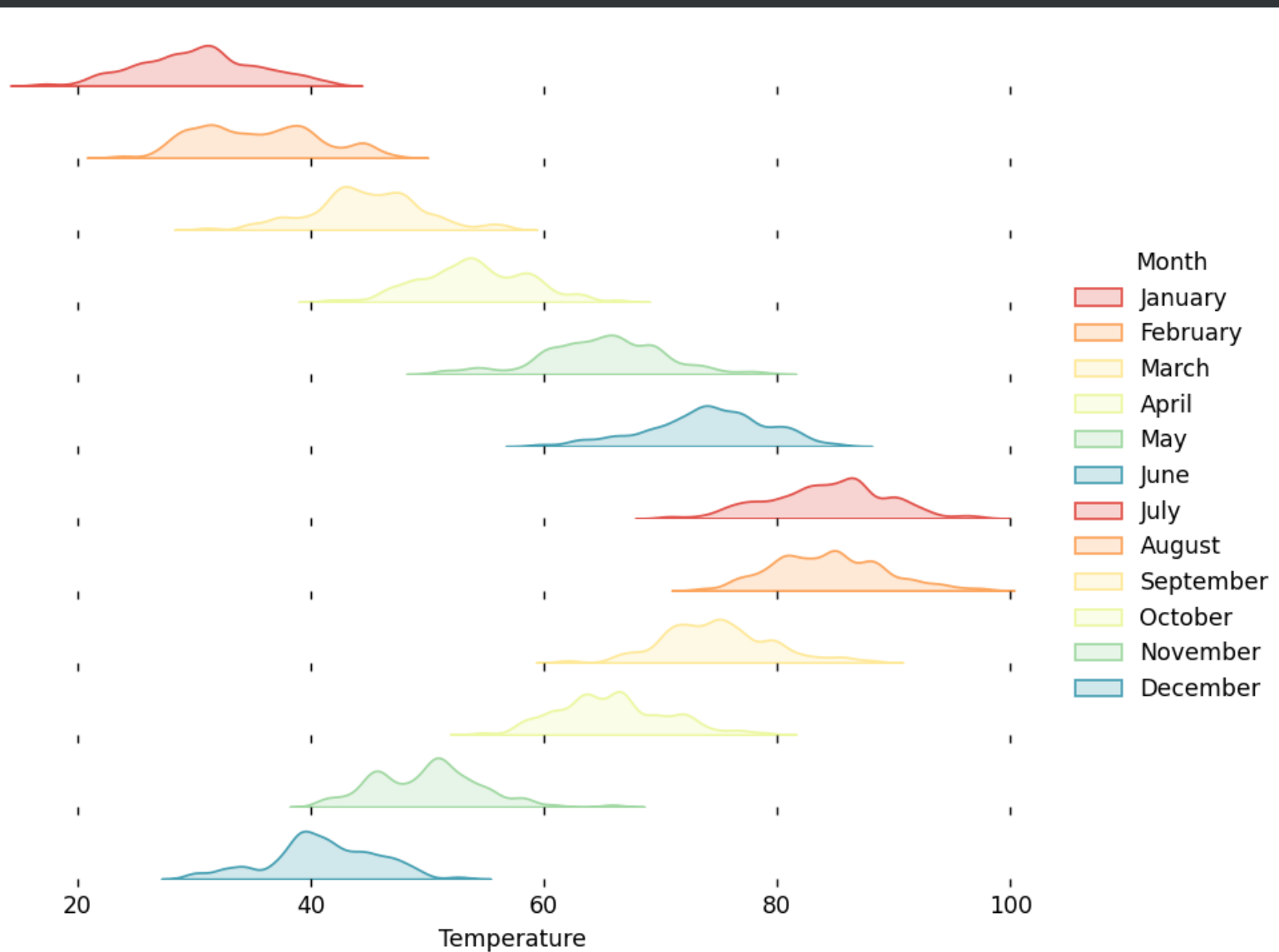


Generate a Figure like this -- I don't even know its name!

Image can be downloaded [here](#)

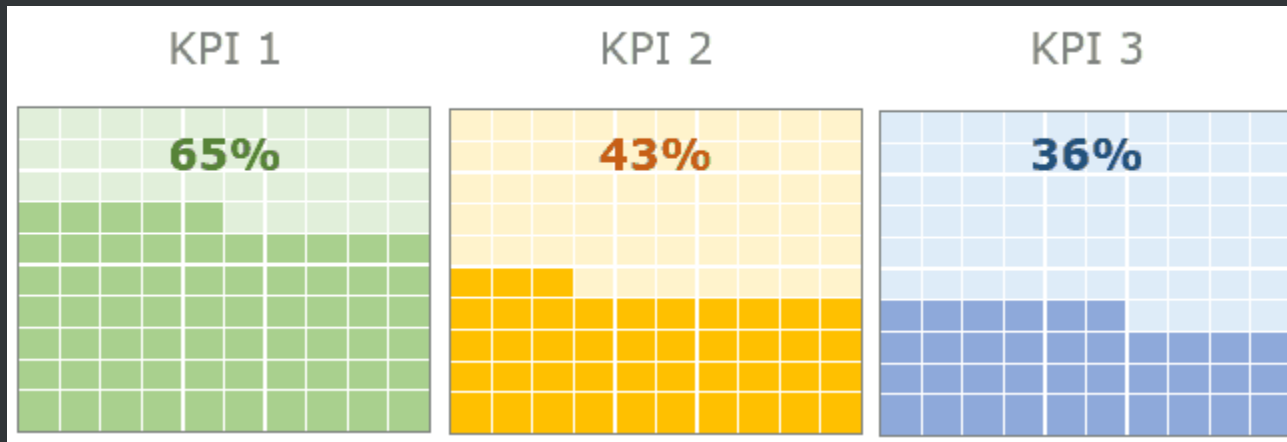


Not too bad...



Generate a Figure like this -- I don't even know its name!

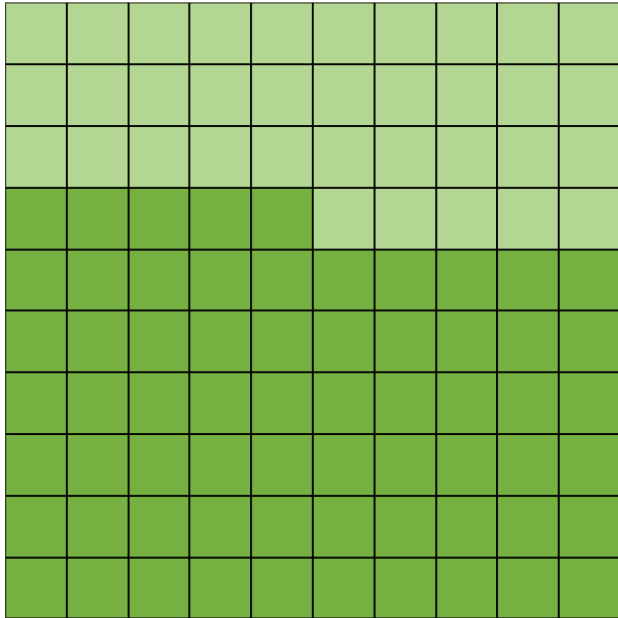
Image can be downloaded [here](#)



After a few trials, I got it here:

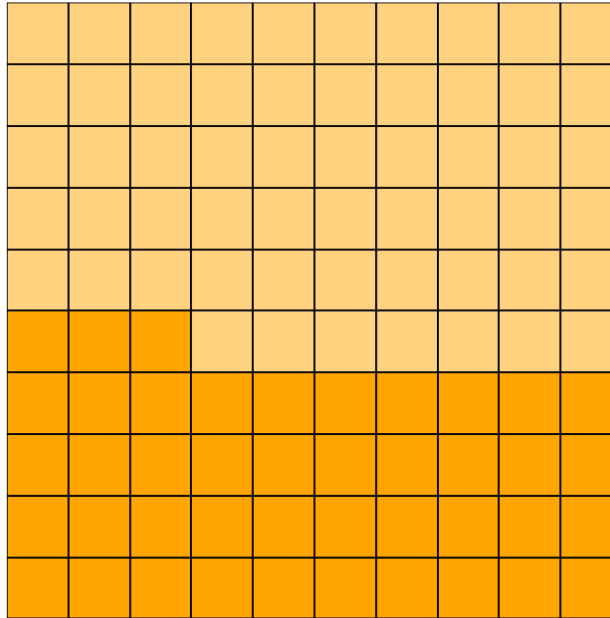
65%

KPI 1



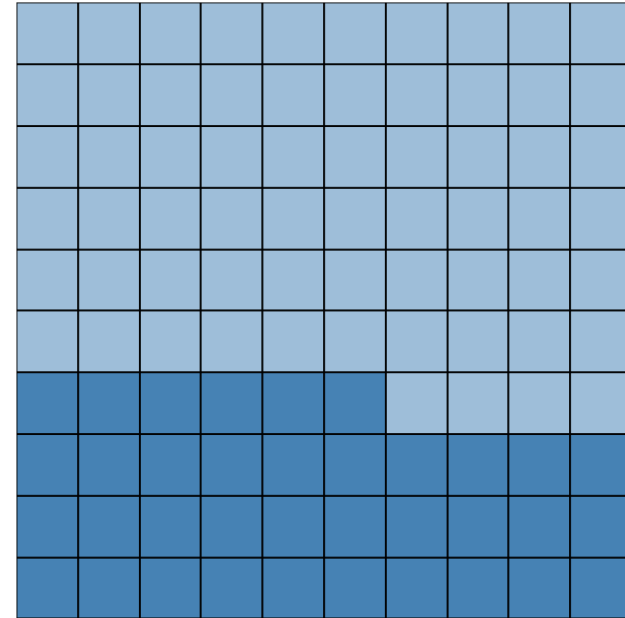
43%

KPI 2

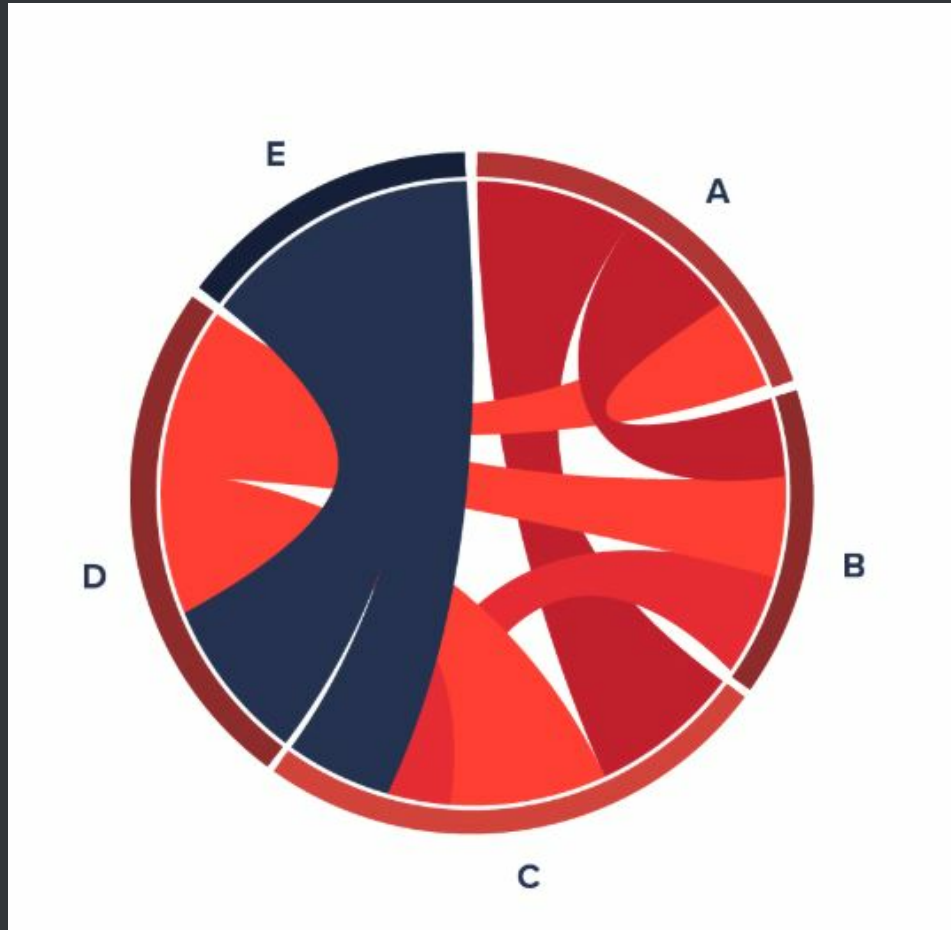


36%

KPI 3

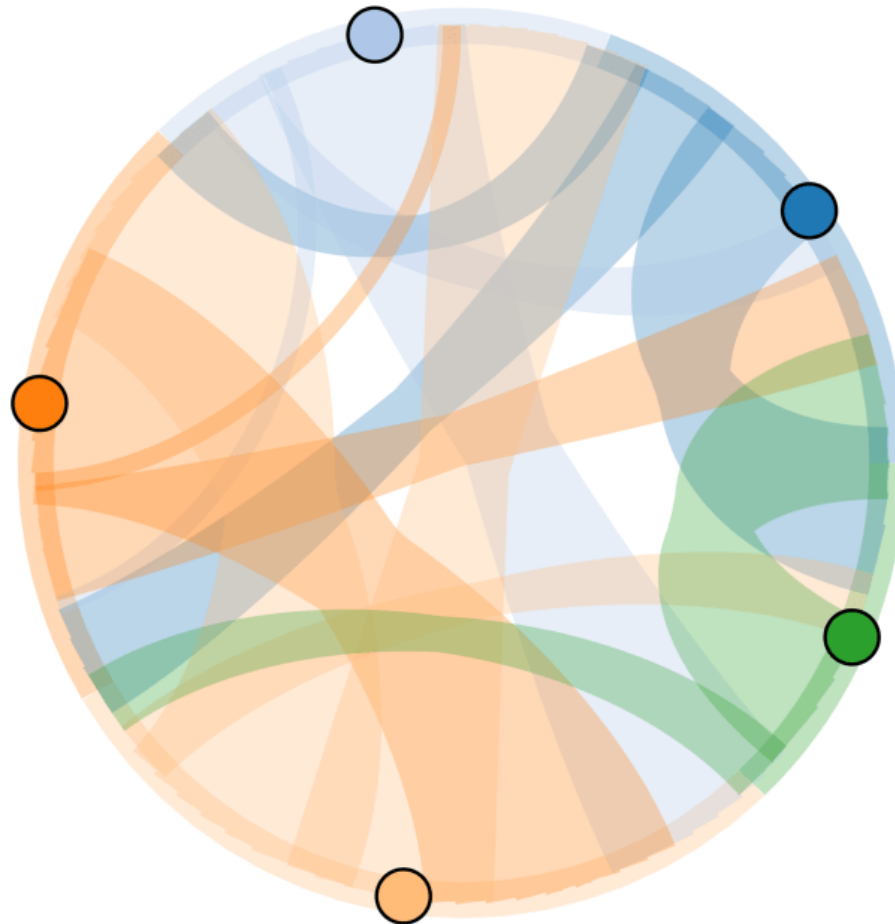


Generate a Figure like this -- It's called a Chord Diagram.
Image can be downloaded [here](#)



It's better than the original one.

(It may not run on Spyder; try Jupyter Notebook on Anaconda)



Assignment (20%)

Assignment (20%)

In this assignment, you need to develop a Python desktop APP using the “tkinter” module with the help of GPT. The main function of the APP is to load, analyze and visualize data. The requirements are as follows.

1. Data loading: The APP should be able to
 - load data itself from an online source (e.g., URL) **OR**,
 - allow the user to input data herself manually **OR**,
 - allow the user to upload a data file to the APP from her device.

Assignment (20%)

In this assignment, you need to develop a Python desktop APP using the “tkinter” module with the help of GPT. The main function of the APP is to load, analyze and visualize data. The requirements are as follows.

2. Data analysis: The APP has a user-interface which allows the user to simply (e.g., by clicking a button) perform some data analyses (chosen by the developer).

Assignment (20%)

In this assignment, you need to develop a Python desktop APP using the “tkinter” module with the help of GPT. The main function of the APP is to load, analyze and visualize data. The requirements are as follows.

3. Data visualization: The APP should offer users of the APP a convenient way to visualize the output of data analysis.

Example

An APP that loads data from a URL. Then, the user can choose the dependent variable and independent variable and click a button to run a regression. The APP demonstrates the regression equation, visualizes the result and explains the meaning.

Example

An APP that asks the user for data and runs different types of t -tests. The user chooses the type of t -test and inputs the data himself/herself. Then, the APP visualizes the result and explains the meaning to the user.

Deliverables

- Your code;
- Your prompt when using GPT or other LLMs, which explains how you created the APP;
- If the APP needs any external data files, also submit your data files (size limit is 5 MB).
- **Make sure your code runs well.**

Deadline

April 5, 2025, 23:59 PM.

This is a group assignment.

We may select and demonstrate some good projects in class.