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

Scan the QR code using WeChat to post live comments on our screen:

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?

Python

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

We have GPT (and other LLMs such as DeepSeek)!

Our objective is to do most of coding with GPT and only make slight modifications ourselves.

Python

Have you used LLM for coding before?

IBM

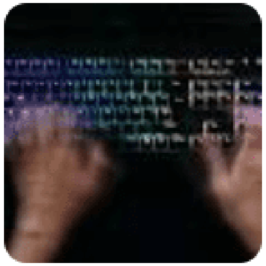
AI isn’t just making it easier to code. It makes coding more fun.

Generative AI serving as a coding assistant is helping software engineers not only do their jobs more quickly, but is boosting developer...

12 hours ago

The New Stack

Augment Code: An AI Coding Tool for ‘Real’ Development Work



While Bolt is aimed at a broad user base, Augment Code is an AI coding tool specifically for working developers. We check it out.

1 day ago

2025年99%代码AI生成，OpenAI高管宣告没有退路，人类将被全面超越

新智元 · 2025年03月17日 19:32

AI不会让所有人平等

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 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 assistant here: https://poe.com/

- POE assistant can also handle images as input

You can also use DeepSeek 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

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

Plotting the relationship between age and credit_score

Xi Li

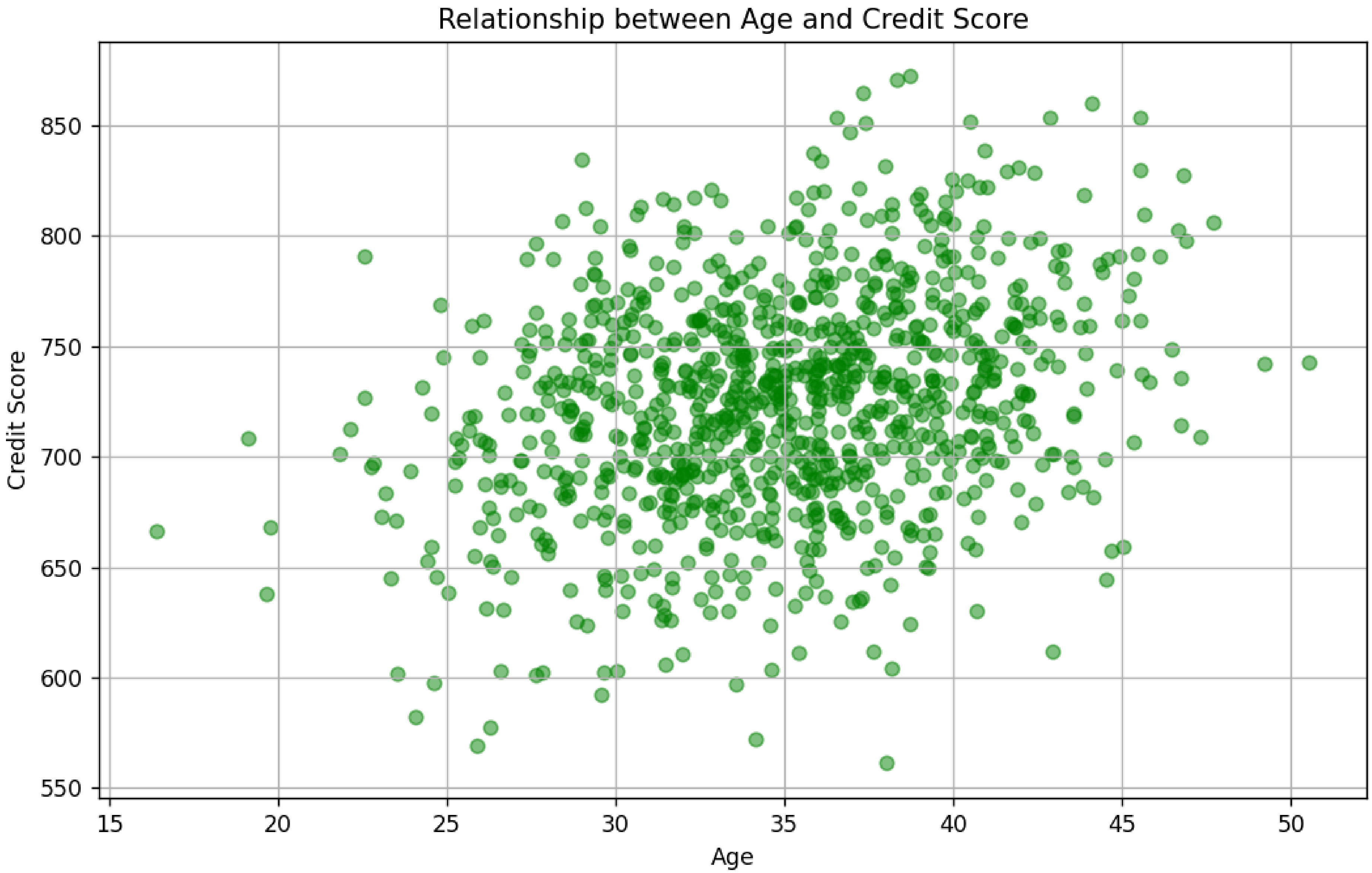
Plotting the relationship between age and credit_score

17/02/2025, 09:23:56

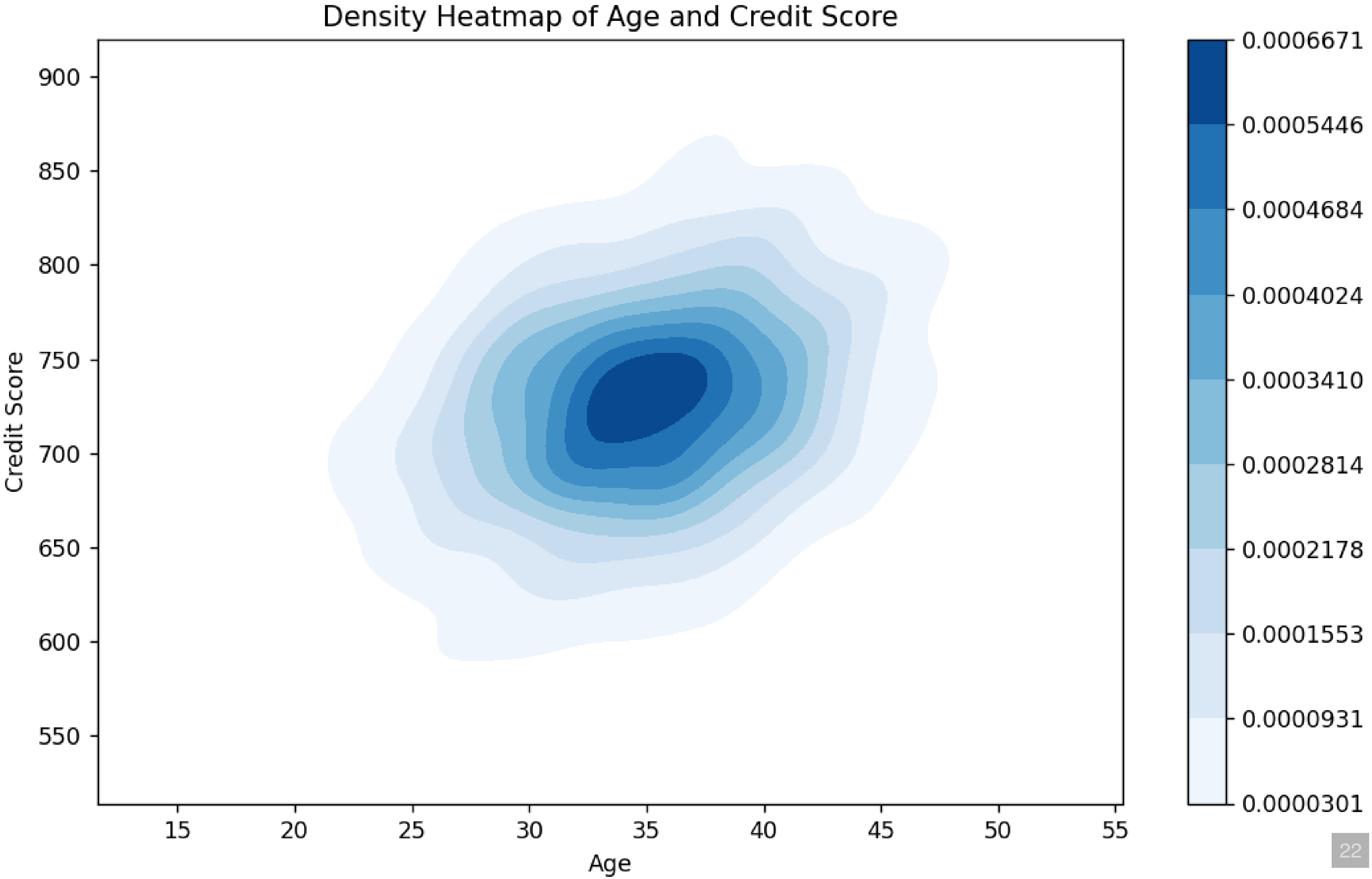
```python
import pandas as pd
import matplotlib.pyplot as plt

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

plt.figure(figsize=(10, 6))
plt.scatter(df['age'], df['credit_score'], color='green', alpha=0.5)
plt.title('Relationship between Age and Credit Score')
plt.xlabel('Age')
plt.ylabel('Credit Score')
plt.grid(True)
plt.show()
```


Relationship between Age and Credit Score
Credit Score
850
800
750
700
650
600
550
15
20
25
30
35
40
45
50
Age

You can also ask for a density heatmap.



Code for generating density heatmap:

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

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

plt.figure(figsize=(10, 6))
sns.kdeplot(x=df['age'], y=df['credit_score'], cmap='Blues',
shade=True, cbar=True)
plt.title('Density Heatmap of Age and Credit Score')
plt.xlabel('Age')
plt.ylabel('Credit Score')
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

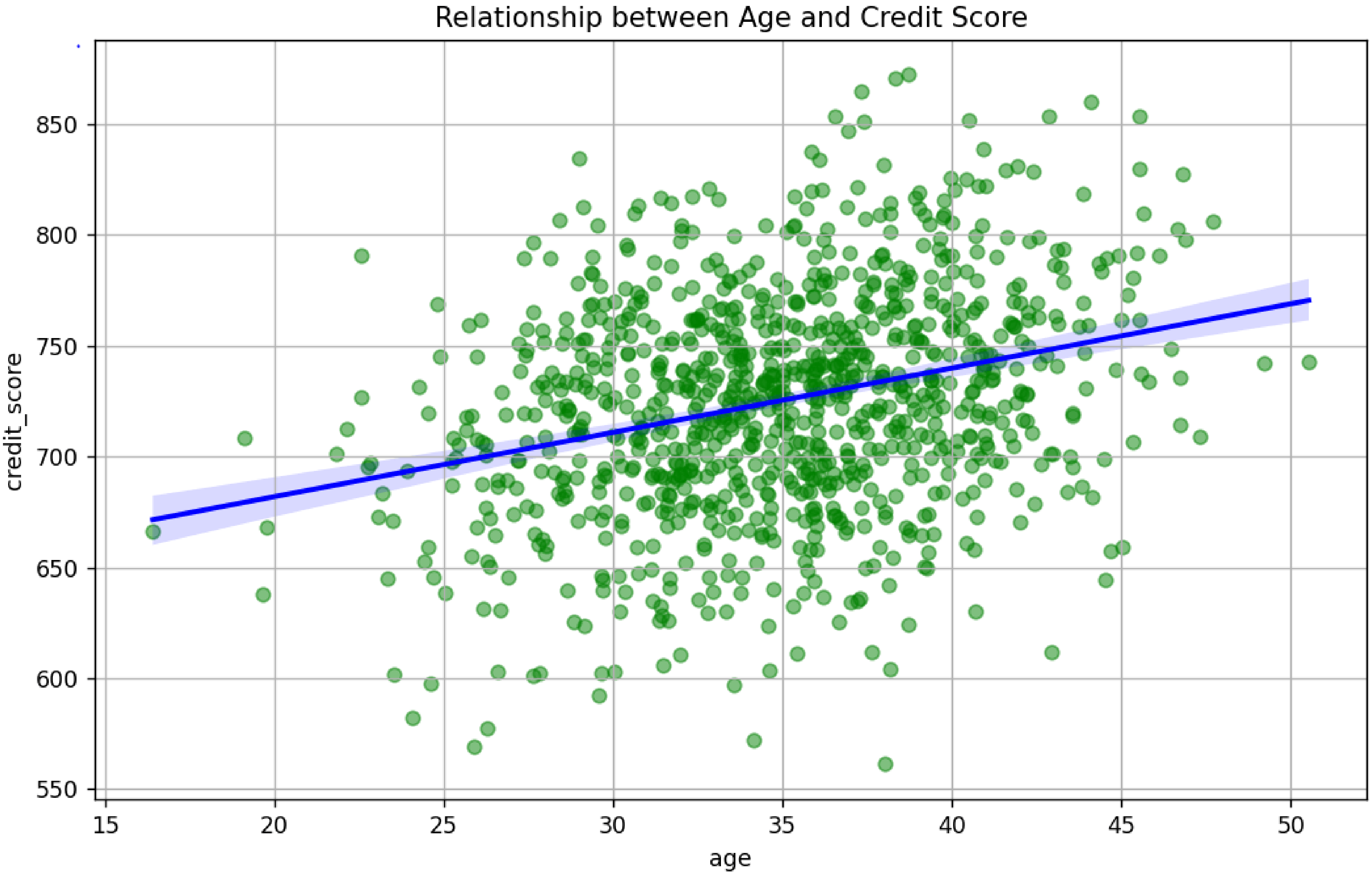
I also need the significance of the test

17/02/2025, 09:36:52

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

Adding a Regression Line

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


Relationship between Age and Credit Score
credit_score
age
850
800
750
700
650
600
550
15
20
25
30
35
40
45
50

Adding regression output

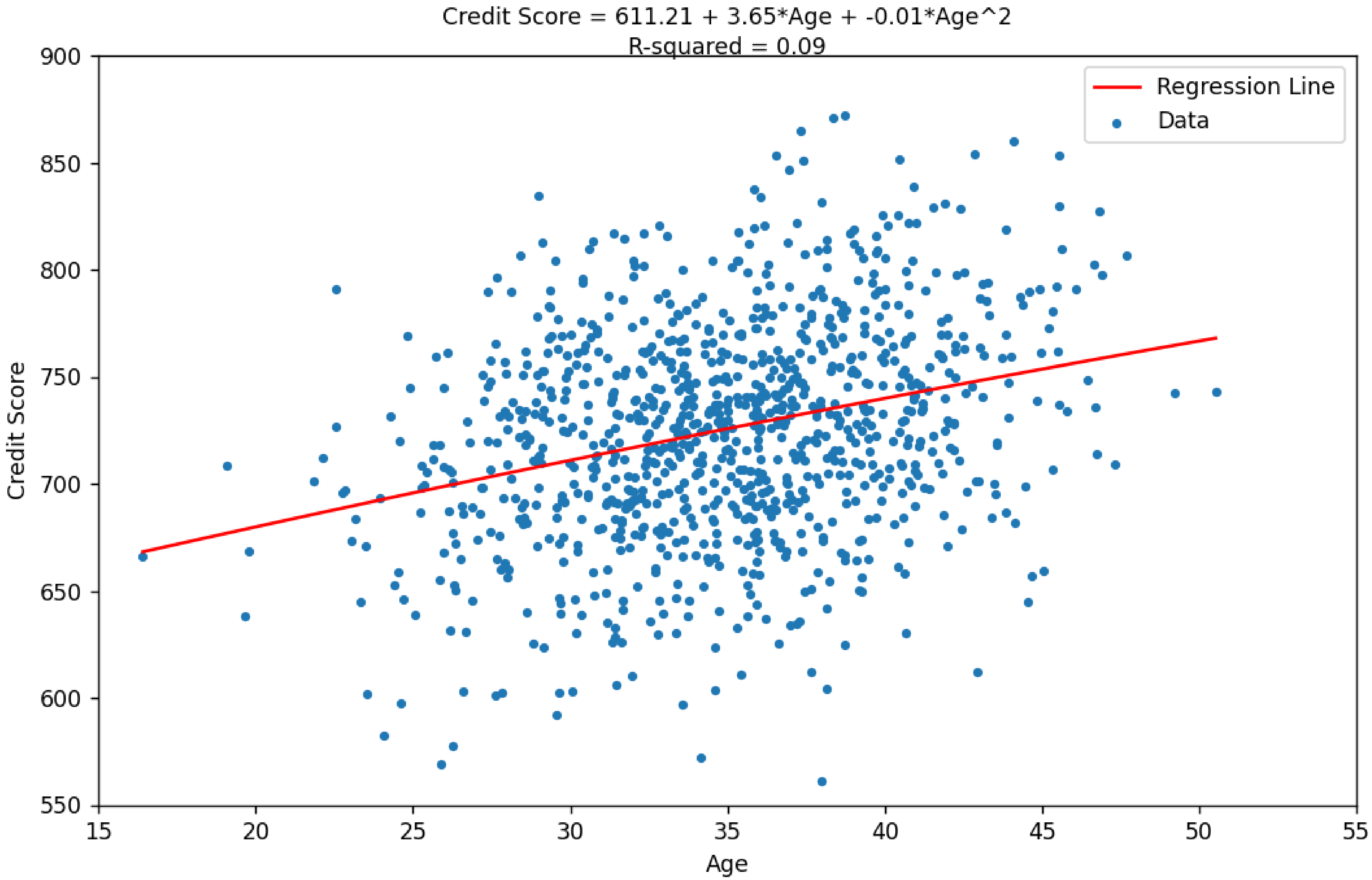
```
              Method:           Least Squares   F-statistic:                 93.84
          Date:           Mon, 17 Feb 2025   Prob (F-statistic):          2.85e-21
             Time:                  09:50:13   Log-Likelihood:              -5288.6
        No. Observations:                1000   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!

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

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


Credit Score = 611.21 + 3.65*Age + -0.01*Age^2
R-squared = 0.09
Regression Line
Data
900
850
800
750
700
650
600
550
Credit Score
15
20
25
30
35
40
45
50
55
Age

Exercise 1:

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

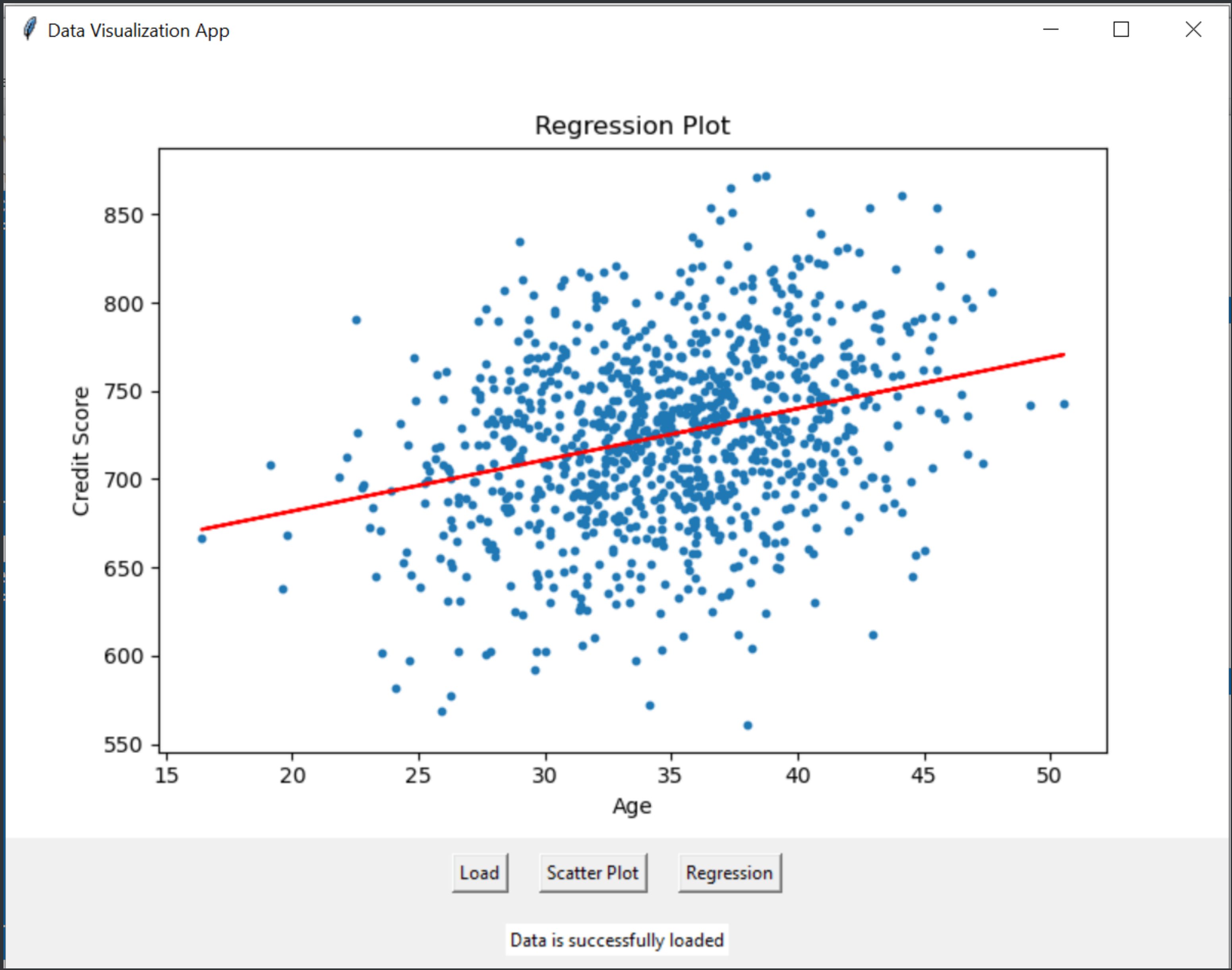
Exercise 1:

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.

```python
import tkinter as tk
import numpy as np
from sklearn.linear_model import LinearRegression
points = []
def generate_point(event):
    x = event.x
    y = event.y
    points.append((x, y))
    canvas.create_oval(x - 2, y - 2, x + 2, y + 2, fill="red")
def perform_regression():
    X = [point[0] for point in points]
    y = [point[1] for point in points]
    X = np.array(X).reshape(-1, 1)
    y = np.array(y)
    model = LinearRegression()
    model.fit(X, y)
    y_pred = model.predict(X)
    for i in range(len(X) - 1):
        x1, y1 = X[i][0], y_pred[i]
        x2, y2 = X[i + 1][0], y_pred[i + 1]
        canvas.create_line(x1, y1, x2, y2, fill="blue")
    equation = "y = {:.2f}x + {:.2f}".format(model.coef_[0], model.intercept_)
    canvas.create_text(10, 10, anchor='nw', text=equation, fill='black', font=('Arial', 12))
def clear_canvas():
    canvas.delete("all")
    del points[:]
app = tk.Tk()
app.title("Regression App")
canvas = tk.Canvas(app, width=800, height=500, bg="white")
canvas.grid(row=0, column=0, columnspan=2, padx=10, pady=10)
canvas.bind("<Button-1>", generate_point)  # Bind left mouse click to generate_point function
regression_button = tk.Button(app, text="Regression", command=perform_regression)
regression_button.grid(row=1, column=0, padx=10, pady=10)
clear_button = tk.Button(app, text="Clear", command=clear_canvas)
clear_button.grid(row=1, column=1, padx=10, pady=10)
app.mainloop()
```

t-test: What is a t-test?

Can you propose an example of a t-test?

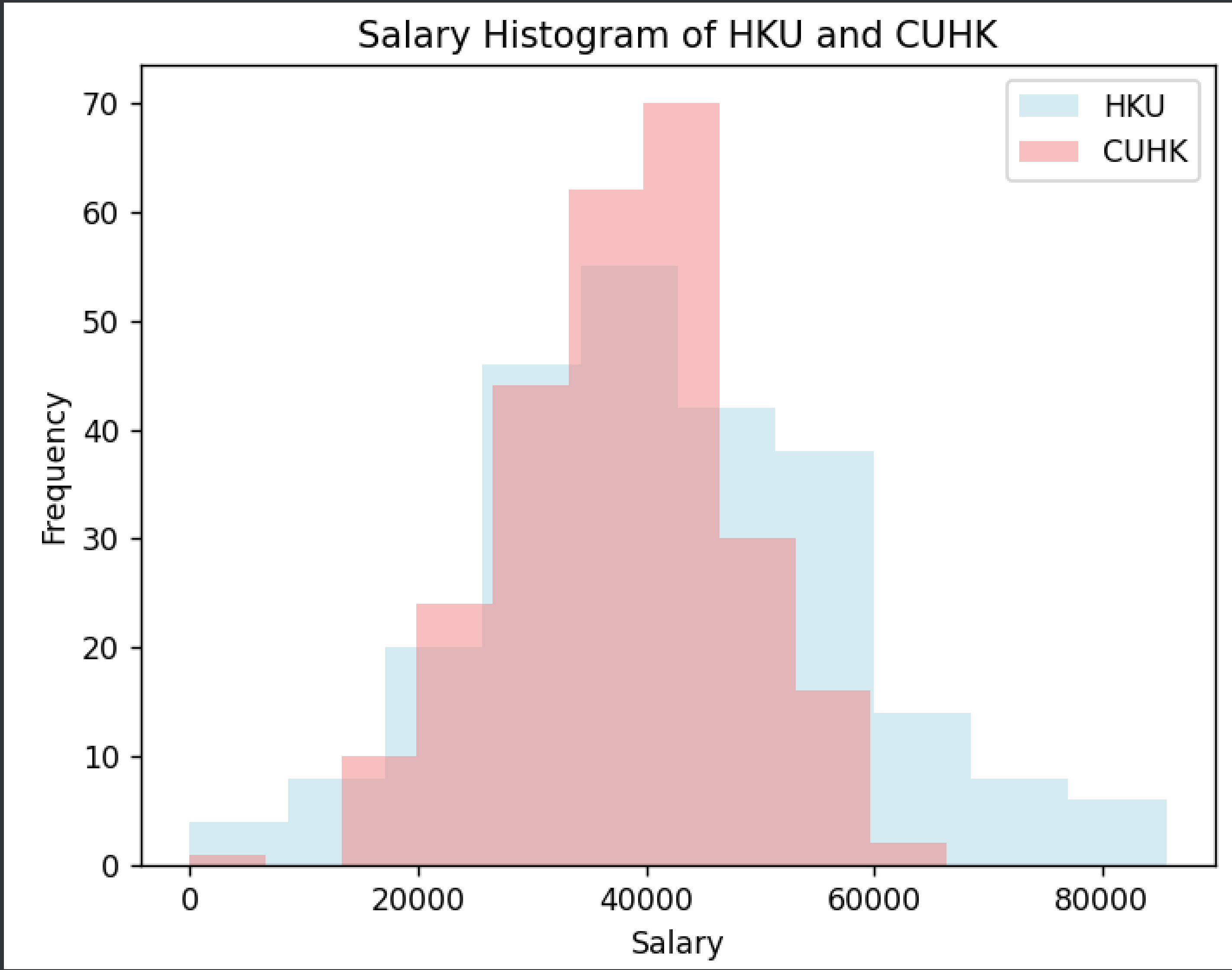
t-test: What is a t-test?

In a (2-sample) 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

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

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



Salary Histogram of HKU and CUHK
HKU
CUHK
Frequency
Salary
0
10
20
30
40
50
60
70
0
20000
40000
60000
80000

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):

```
import pandas as pd
from scipy import stats
# Load the data from the CSV file
data = pd.read_csv("https://ximarketing.github.io/data/python_data2.csv")
# Separate the salary data for HKU and CUHK graduates
hku_salaries = data[data['School'] == 'HKU']['Salary']
cuhk_salaries = data[data['School'] == 'CUHK']['Salary']
# Calculate the means
hku_mean = hku_salaries.mean()
cuhk_mean = cuhk_salaries.mean()
# Perform t-test
t_statistic, p_value = stats.ttest_ind(hku_salaries, cuhk_salaries)
print("Mean salary for HKU graduates:", hku_mean)
print("Mean salary for CUHK graduates:", cuhk_mean)
print("T-Statistic:", t_statistic)
print("P-Value:", p_value)
```

How do you interpret the result?

```
In [17]: runfile('C:/Users/FBE/.spyder-py3/temp.py', wdir='C:/Users/FBE/.spyder-py3')
Mean salary for HKU graduates: 41845.68464730291
Mean salary for CUHK graduates: 38165.33976833977
T-Statistic: 3.134699938666276
P-Value: 0.001821836212611976
```

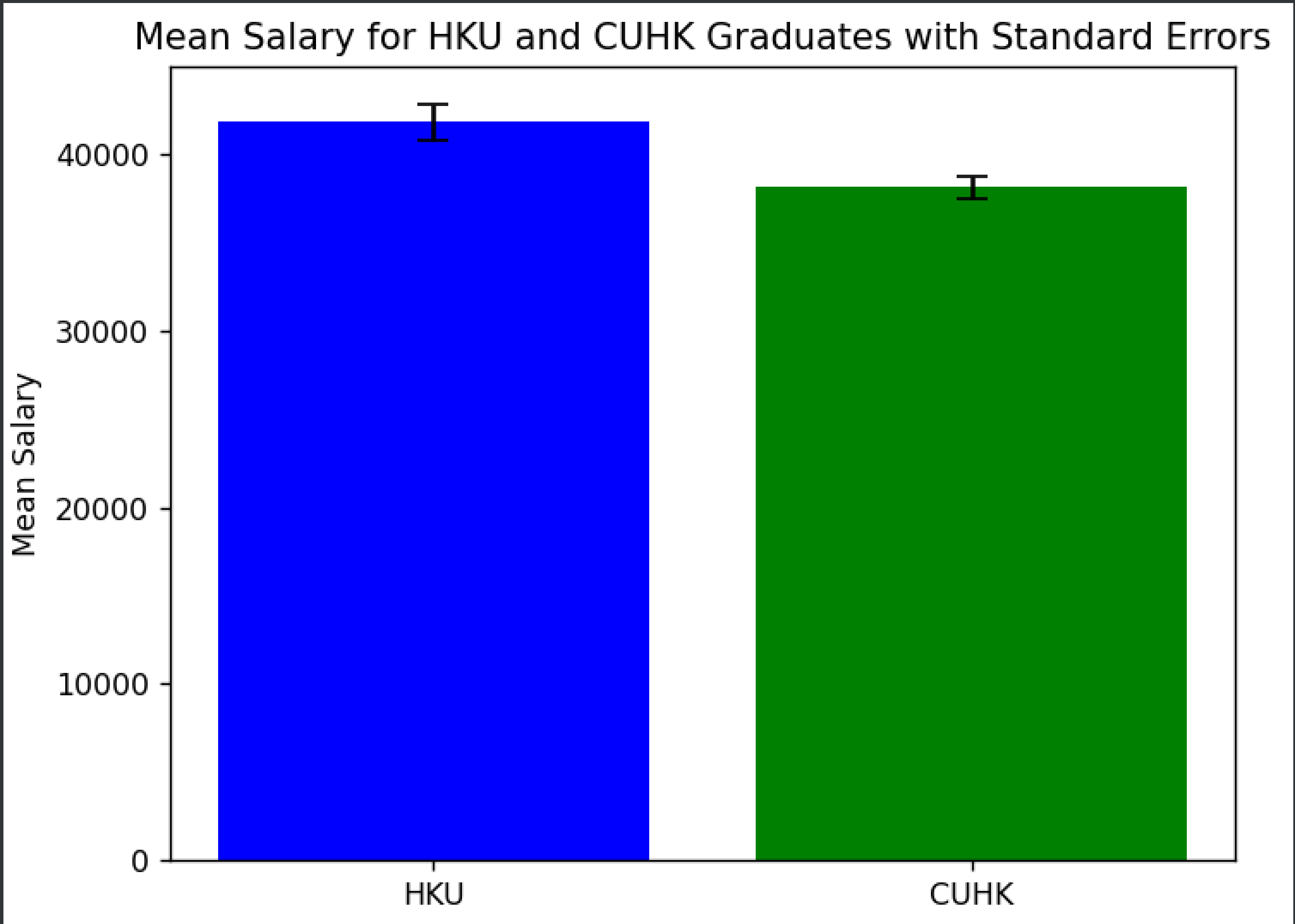
How do you interpret the result?

```
In [17]: runfile('C:/Users/FBE/.spyder-py3/temp.py', wdir='C:/Users/FBE/.spyder-py3')
Mean salary for HKU graduates: 41845.68464730291
Mean salary for CUHK graduates: 38165.33976833977
T-Statistic: 3.134699938666276
P-Value: 0.001821836212611976
```

- 41,845 > 38,165; it seems HKU students do better.
- p-value < 5%, the result is statistically significant.
- We can conclude that HKU students make significant higher salaries than CUHK students do.

Asking for visualization (including standard error)

```python
import pandas as pd
from scipy import stats
import matplotlib.pyplot as plt
import numpy as np
data = pd.read_csv("https://ximarketing.github.io/data/python_data2.csv")
hku_salaries = data[data['School'] == 'HKU']['Salary']
cuhk_salaries = data[data['School'] == 'CUHK']['Salary']
hku_mean = hku_salaries.mean()
cuhk_mean = cuhk_salaries.mean()
hku_std = hku_salaries.std()
cuhk_std = cuhk_salaries.std()
hku_se = hku_std / np.sqrt(len(hku_salaries))
cuhk_se = cuhk_std / np.sqrt(len(cuhk_salaries))
t_statistic, p_value = stats.ttest_ind(hku_salaries, cuhk_salaries)
plt.bar(['HKU', 'CUHK'], [hku_mean, cuhk_mean], yerr=[hku_se, cuhk_se],
capsize=5, color=['blue', 'green'])
plt.ylabel('Mean Salary')
plt.title('Mean Salary for HKU and CUHK Graduates with Standard Errors')
plt.show()
```


Mean Salary for HKU and CUHK Graduates with Standard Errors
Mean Salary
40000
30000
20000
10000
0
HKU
CUHK

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.

Types of t-test:

- One sample t-test: Test if the population mean is equal to (greater than, smaller than) a specific value or not. For instance, is the average GPA of students greater than 3.7?
- Two sample t-test: Test if the population means of two groups are equal or not. For instance, do females score better than males?
- Paired sample t-test: Test if the difference between paired measurements for a population is zero or not. For instance, do you get high salaries after taking the class?
- Left for your exercise!

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
Separed	11	12	18	2	0
Divorced	12	20	23	5	2

My code produced by GPT

```python
import numpy as np
from scipy.stats import chi2_contingency

# Create a 2x2 contingency table (replace the values with your own data)
observed_table = np.array([
                [12,18,36,19,4],
                [45,59,107,47,8],
                [11,12,18,2,0],
                [12,20,23,5,2]])

# Perform the Chi-square test
chi2_stat, p_val, dof, expected_table = chi2_contingency(observed_table)

print("Chi-square Statistic:", chi2_stat)
print("P-value:", p_val)
print("Degrees of Freedom:", dof)
print("Expected Table:")
print(expected_table)
```

```python
import numpy as np
from scipy.stats import chi2_contingency

# Create a 2x2 contingency table (replace the values with your own data)
observed_table = np.array([
                 [12,18,36,19,4],
                 [45,59,107,47,8],
                 [11,12,18,2,0],
                 [12,20,23,5,2]])

# Perform the Chi-square test
chi2_stat, p_val, dof, expected_table = chi2_contingency(observed_table)

print("Chi-square Statistic:", chi2_stat)
print("P-value:", p_val)
print("Degrees of Freedom:", dof)
print("Expected Table:")
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

```python
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.pyplot as plt
import numpy as np
data = pd.read_csv('https://ximarketing.github.io/data/python_data3.csv')
table = pd.crosstab(data.Gender, columns=data.Purchase)
print(table)
print(stats.chi2_contingency(table, correction=False))
```

χ-Square Test: From Documents

```
statistic=4.446721648705711, pvalue=0.03496831490600599, dof=1
```

Here, our result is $p = 0.034 < 5\%$, suggests that there is indeed an association between gender and purchase decision, or equivalently, we can say that gender has an effect on your purchase decision.

Logistic Regression
What’s it?

Logistic Regression: What's it?

GPT-3.5 (Tokens used 1876)

Logistic regression is a statistical model used to predict the probability of a binary outcome based on one or more predictor variables. It is commonly used for classification tasks where the dependent variable is categorical and has two possible outcomes (e.g., 0 or 1, Yes or No, True or False).

In logistic regression, the relationship between the predictor variables and the probability of the binary outcome is modeled using the logistic function (also known as the sigmoid function). The logistic function maps any real-valued input to a value between 0 and 1, representing the probability of the event occurring.

The logistic regression model estimates the coefficients (weights) for the predictor variables to maximize the likelihood of the observed data. These coefficients are used to calculate the log-odds (logit) of the event occurring, which is then transformed into a probability using the logistic function.

Key points about logistic regression:

1. It is a type of generalized linear model (GLM) that is suitable for binary classification problems.
2. The output of logistic regression is the probability of the event happening, which can be converted into class labels using a threshold (e.g., 0.5).
3. It assumes that the relationship between the predictor variables and the log-odds of the outcome is linear.
4. It is sensitive to outliers and multicollinearity.
5. Regularization techniques like L1 (Lasso) and L2 (Ridge) can be applied to prevent overfitting.

In short, a logistic regression specifies the following function:

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

Why do we choose this “weird” function?

Why do we choose this "weird" function?

The logistic function has a few advantages:

- For any input X, the value of the function is always nonnegative (i.e., probability is nonnegative).
- For any input X, the value of the function is always smaller than 1 (i.e., probability is smaller than 100%).
- The function is monotone with X, i.e., the probability is either increasing or decreasing with X.

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.

```python
import pandas as pd
import statsmodels.api as sm

url = 'https://ximarketing.github.io/data/loan.csv'
data = pd.read_csv(url)

categorical_vars = ['Education', 'EmploymentType', 'MaritalStatus', 'HasMortgage', 'LoanPurpose']
data = pd.get_dummies(data, columns=categorical_vars, drop_first=True)

numerical_vars = ['Age', 'Income', 'LoanAmount', 'CreditScore', 'MonthsEmployed', 'InterestRate', 'LoanTerm']
X = data[numerical_vars + [col for col in data.columns if col.startswith(tuple(categorical_vars))]]
y = data['Default']
X = sm.add_constant(X)
model = sm.Logit(y, X.astype(float))
result = model.fit()

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

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

- Older / High Income borrower are less likely to default.
- When loans are bigger, borrower are more likely to default.
- When credit scores are higher, borrowers are less likely to default...

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

For education, we take Bachelors as the benchmark. By comparing the numbers, we can predict the default risk:

High School > Bachelors > Masters > PhDs

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

For employment, we take full-time as the benchmark. By comparing the numbers, we can predict the default risk:

Unemployed > Part-time > Self-employed > Full-time

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

For marriage, we take divorced as the benchmark. By comparing the numbers, we can predict the default risk:

Divorced > Single > Married

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

For loan purpose, we take auto as the benchmark. By comparing the numbers, we can predict the default risk:

Business > Auto > Other > Education > Home

Multinomial Logit Model

What's it?

What is the brand of the HKU president’s car?

TESLA

Mercedes-Benz

BMW

Audi

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:

```python
import pandas as pd
import statsmodels.api as sm
from patsy import dmatrices
data = pd.read_csv('https://ximarketing.github.io/data/bankchoice.csv')
y, X = dmatrices('Choice ~ Age + Income + Education + C(Job, Treatment)', data, return_type='dataframe')
model = sm.MNLogit(y, X).fit()
summary = model.summary2()
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
choices = y.design_info.column_names
choices.pop()
for i, choice in enumerate(choices, start=1):
    print(f"Choice: {choice}")
    print(summary.tables[i])
    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

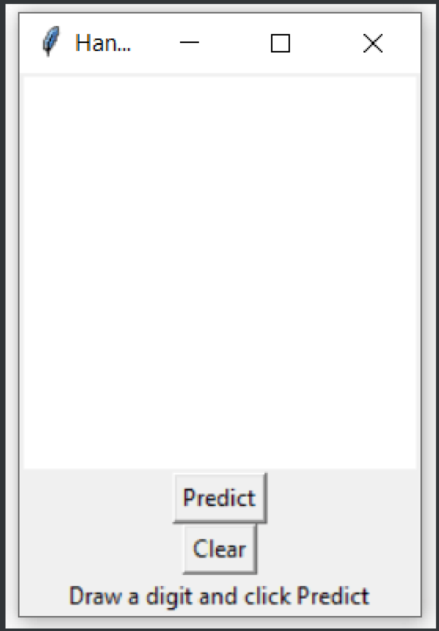
We can further build a handwriting digit prediction model based on multinomial logit regression.

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

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

Handwriting Digit Prediction

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



Code for data training

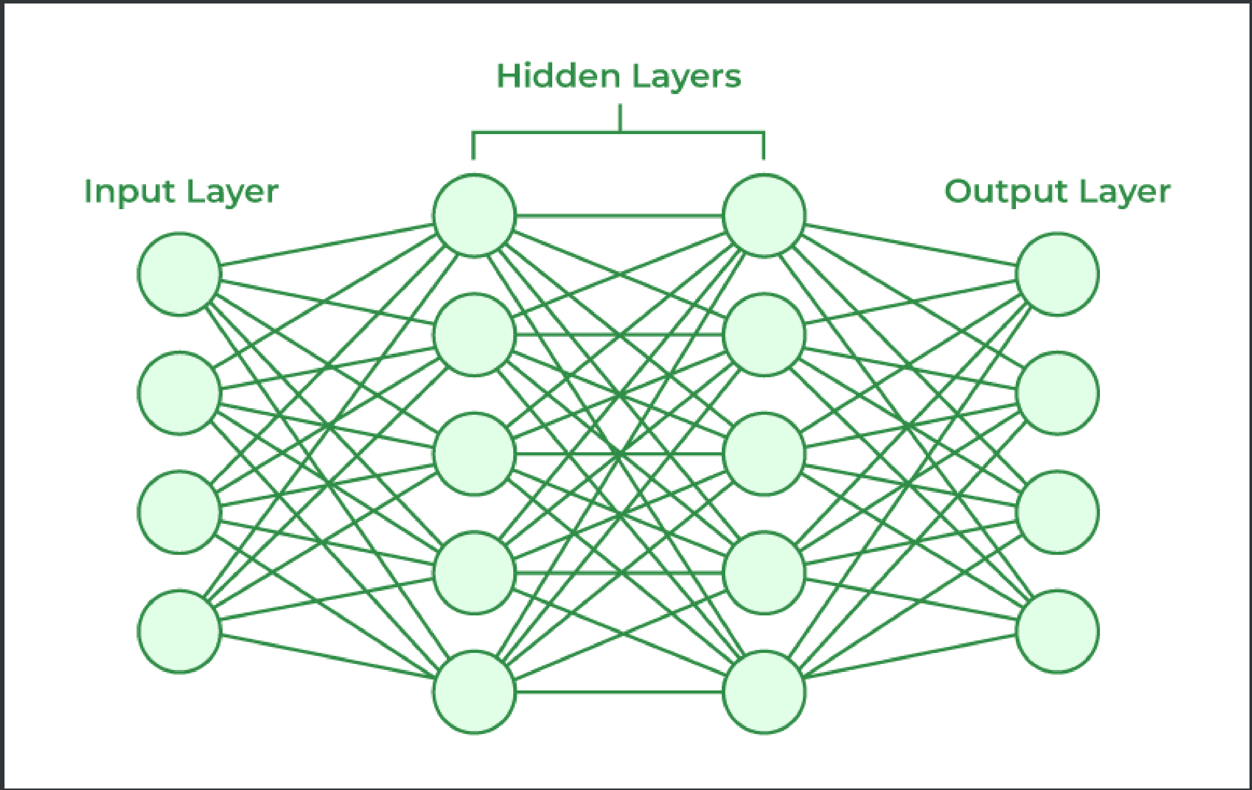
```python
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import joblib
# Load the MNIST dataset
mnist = fetch_openml('mnist_784', version=1)
# Normalize the data
X = mnist.data / 255.0
y = mnist.target.astype(int)
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
# Train the logistic regression model
model = LogisticRegression(solver='lbfgs', max_iter=1000, multi_class='multinomial')
model.fit(X_train, y_train)
# Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
# Save the model
joblib.dump(model, 'logistic_regression_model.pkl')
import tkinter as tk
from tkinter import *
import numpy as np
from PIL import Image, ImageDraw, ImageOps
import joblib
# Load the trained model (make sure you have a trained model saved as 'logistic_regression_mode
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 output $\Pr[y_i = 1]$
- 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 artifical 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

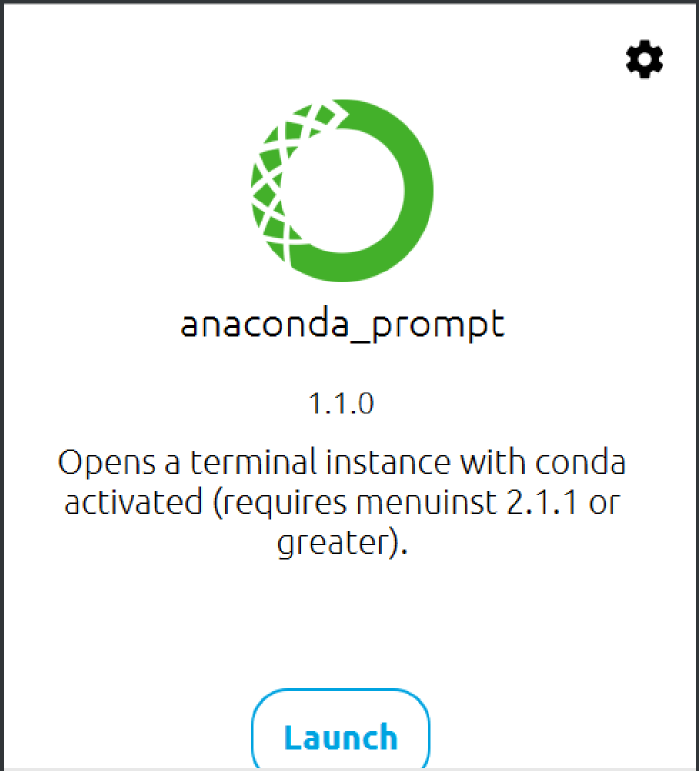
Again, let us build an artificial neural network in python for digit recognition with the help of GPT!

Artificial Neural Network

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

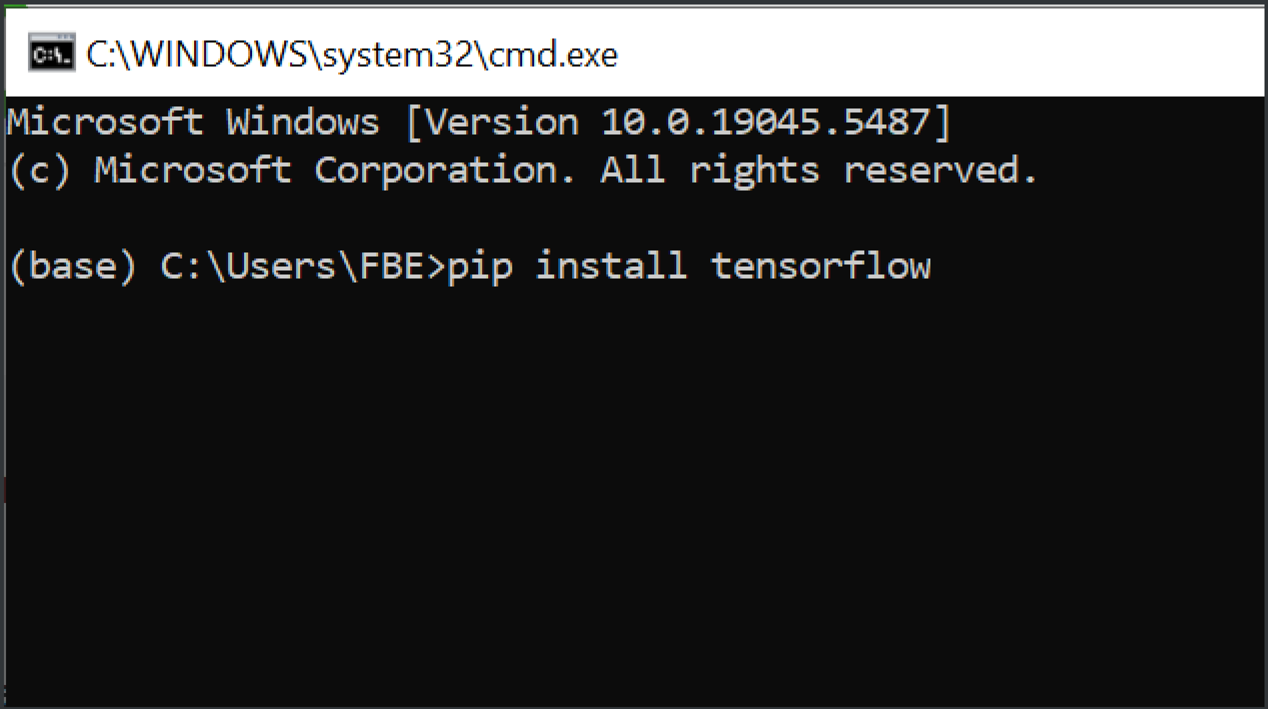
Note: You will need to use the module tensorflow for your ANN. The installation of tensorflow is a bit different.

Installing TensorFlow



On your Anaconda, lunch anaconda prompt

Installing TensorFlow



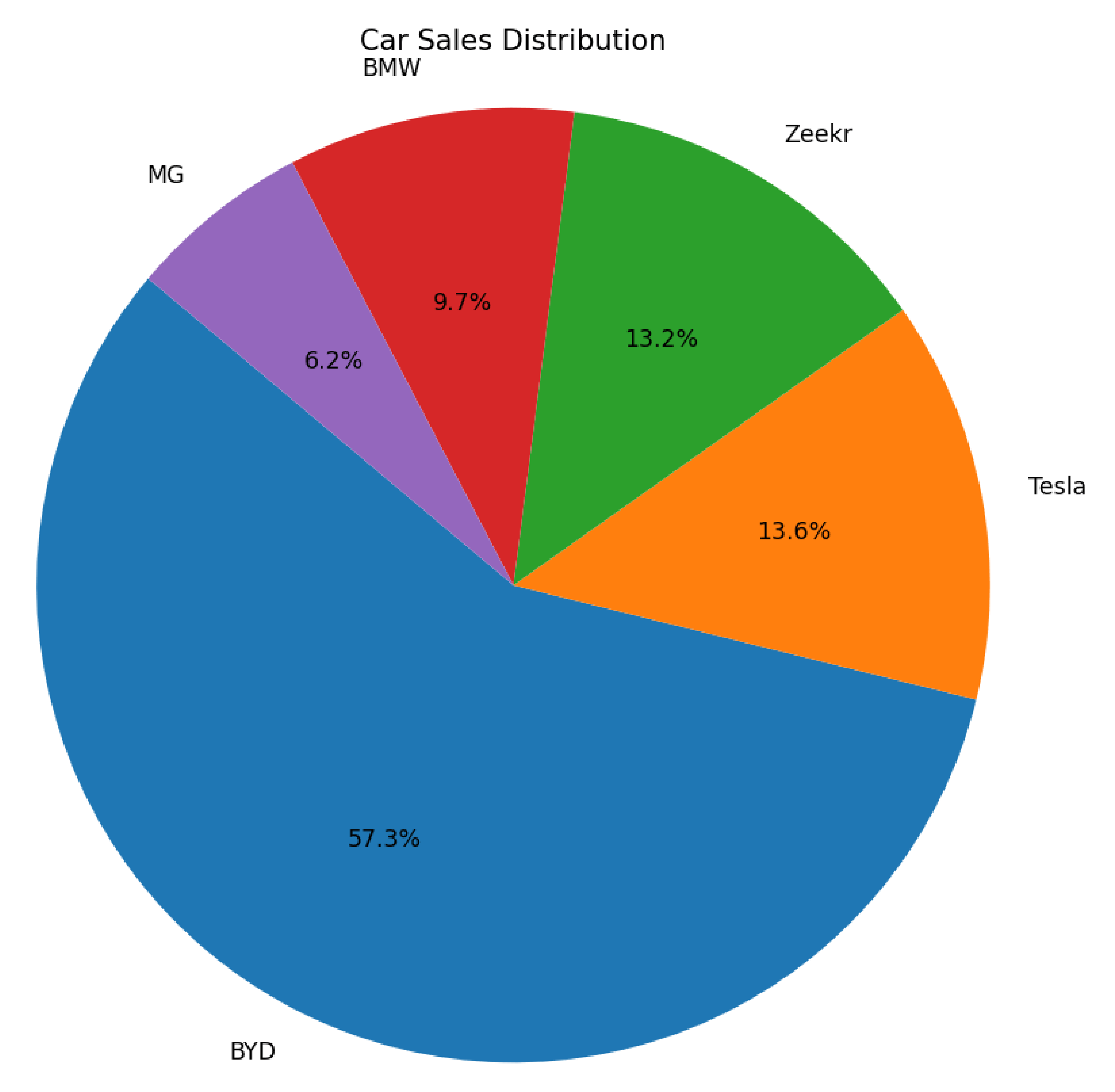
Input "pip install tensorflow", and enter the command

Pie Chart

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

Code generated by GPT

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

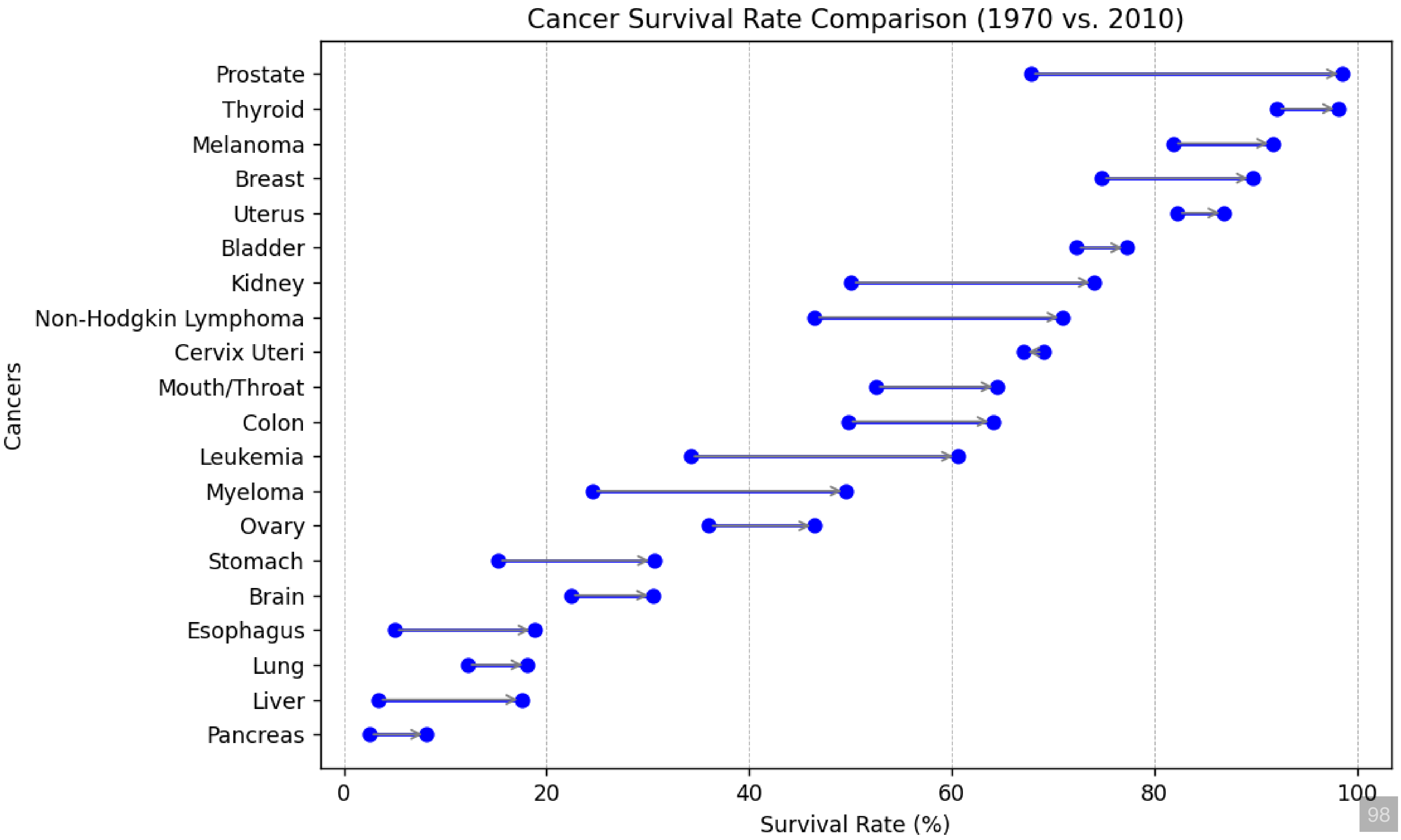

Car Sales Distribution
BMW
Zeekr
MG
9.7%
13.2%
6.2%
Tesla
13.6%
57.3%
BYD

I am making it more interactive (plotly needed)

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

Dumbbell chart

Try to create a dumbbell chart like this one!



The data is available here:

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

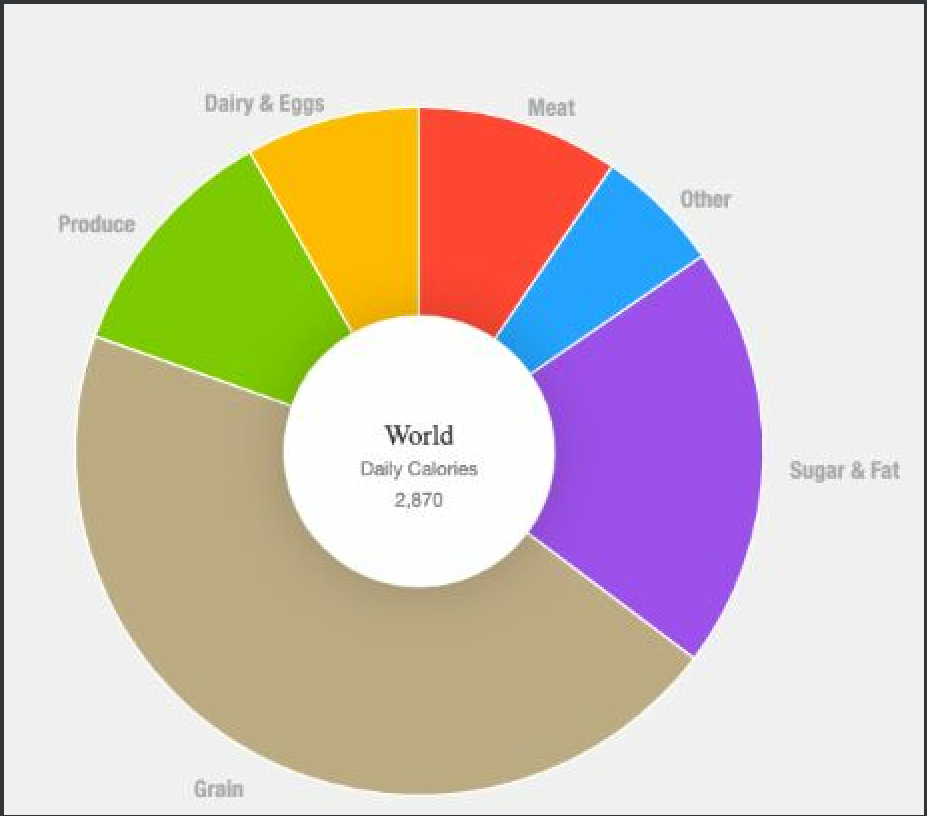
Try to use GPT to create the dumbbell chart for you!

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

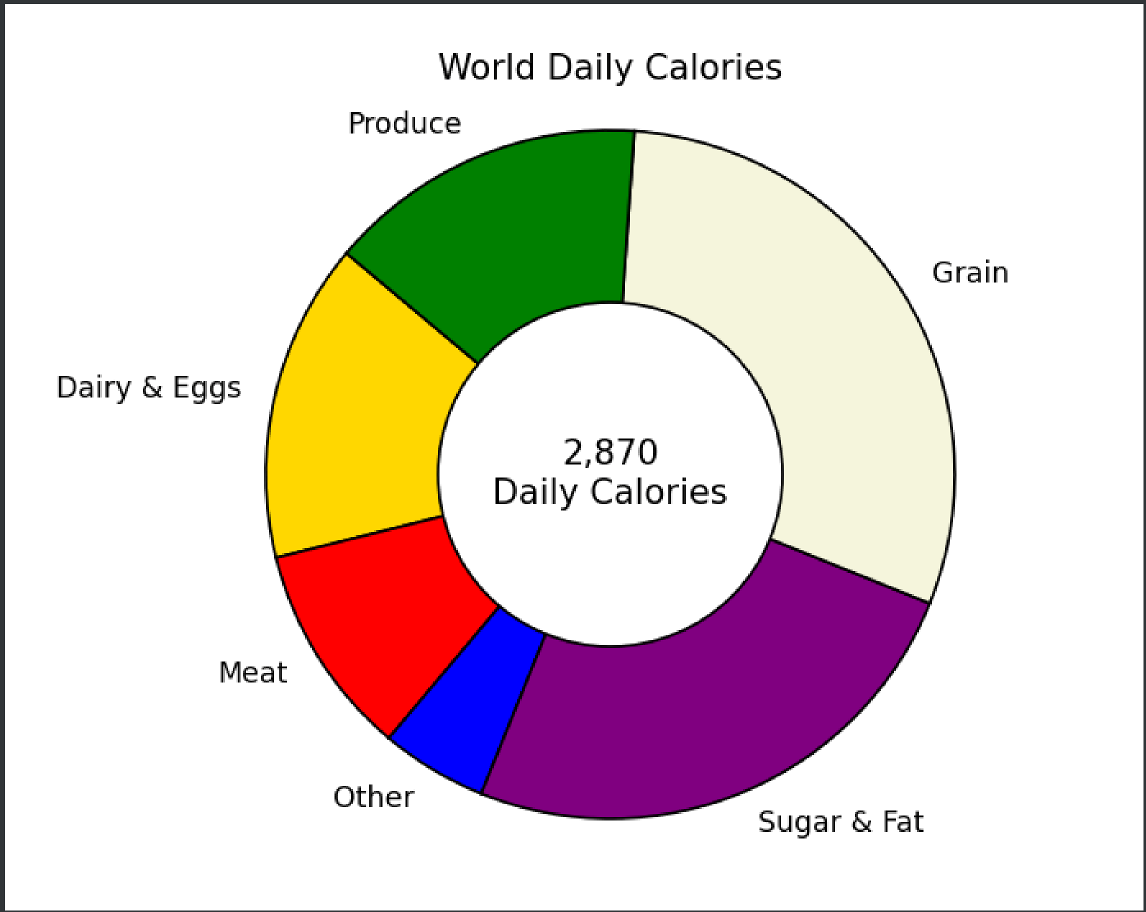
donut.JPG
JPG 14.71 KB

19/02/2025, 08:51:42

GPT-4o (Tokens used 685)

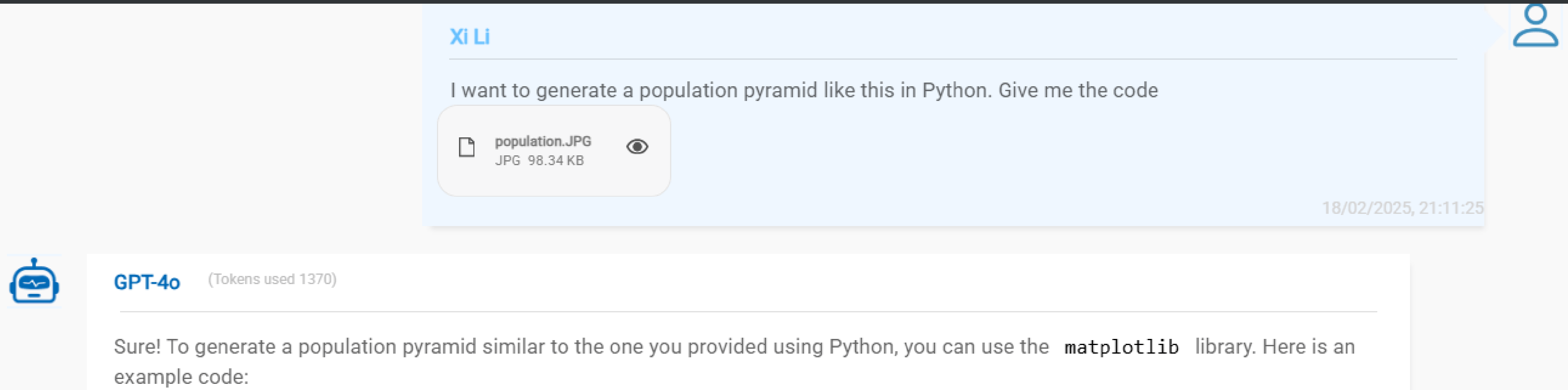
Certainly! Below is an example of how to create a donut chart similar to the one in the image using Python with the help of the `matplotlib` library.

My Output from GPT



Example: Generate a Population Pyramid

Image can be downloaded here



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

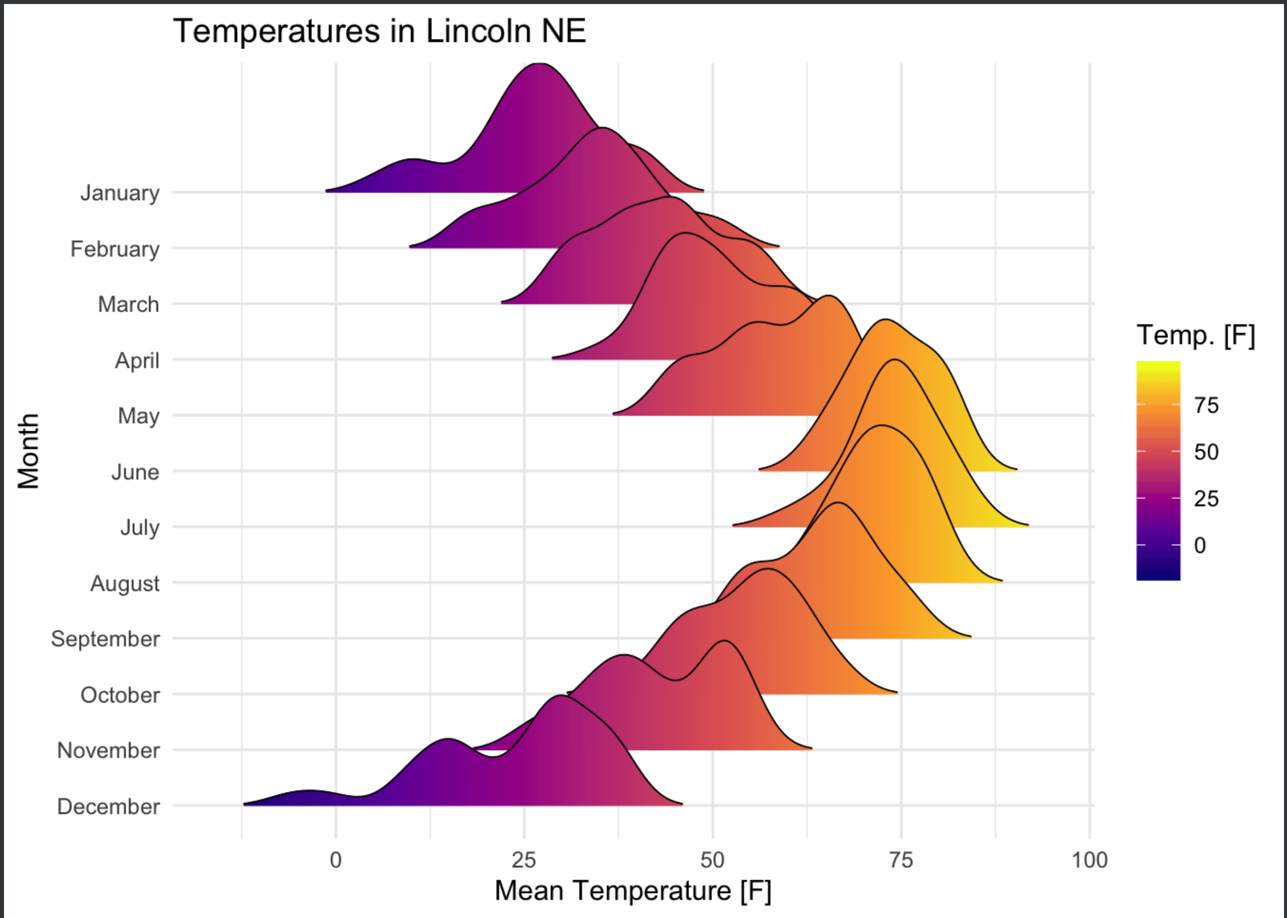
18/02/2025, 21:58:06

GPT-4o (Tokens used 1219)

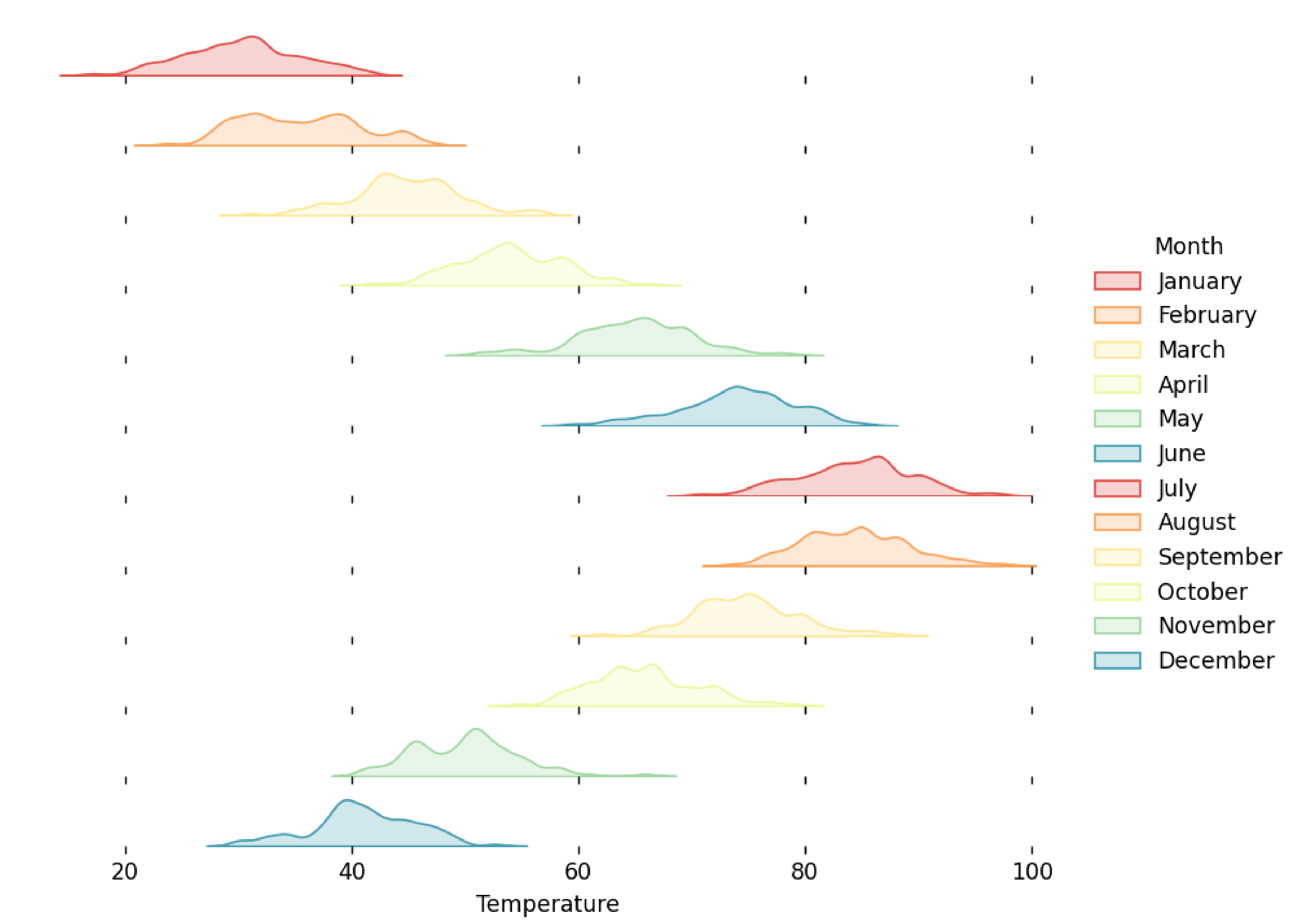
Creating a dynamic GIF like the one you've shown requires several steps, including data preparation, plotting the population pyramids for each year, and then combining these plots into an animated GIF. Here's an example of how you can achieve this using Python with libraries such as `matplotlib`, `pandas`,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

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

Image can be downloaded here



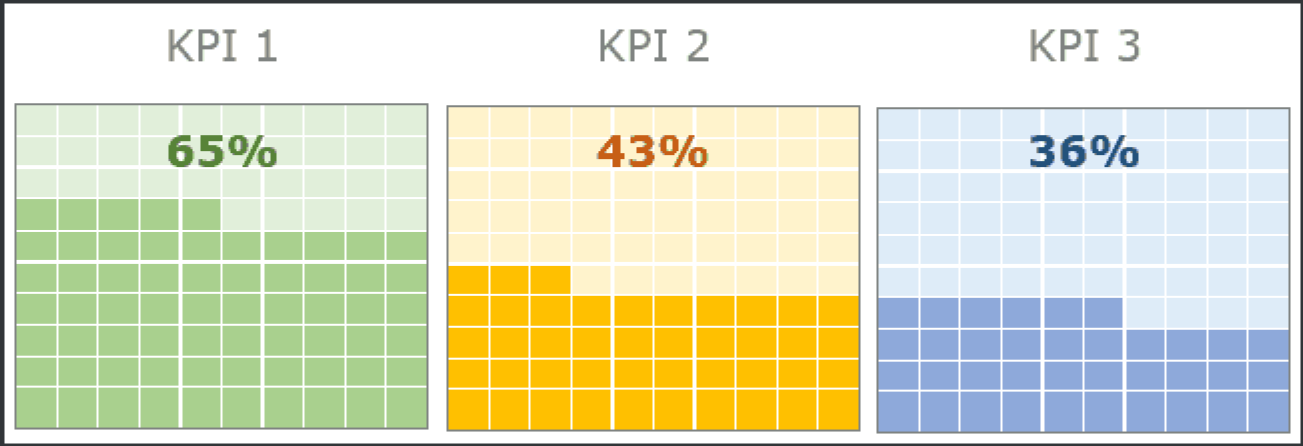
Not too bad...



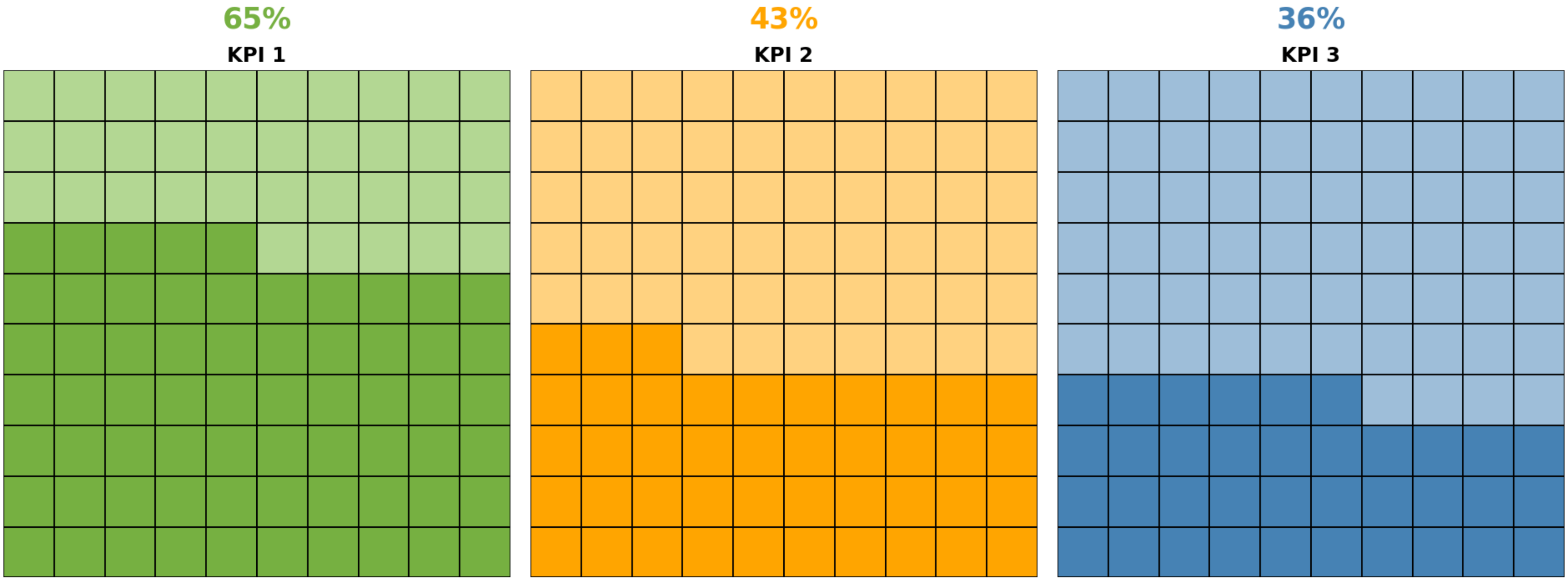
Month
January
February
March
April
May
June
July
August
September
October
November
December
20
40
60
80
100
Temperature

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

Image can be downloaded here

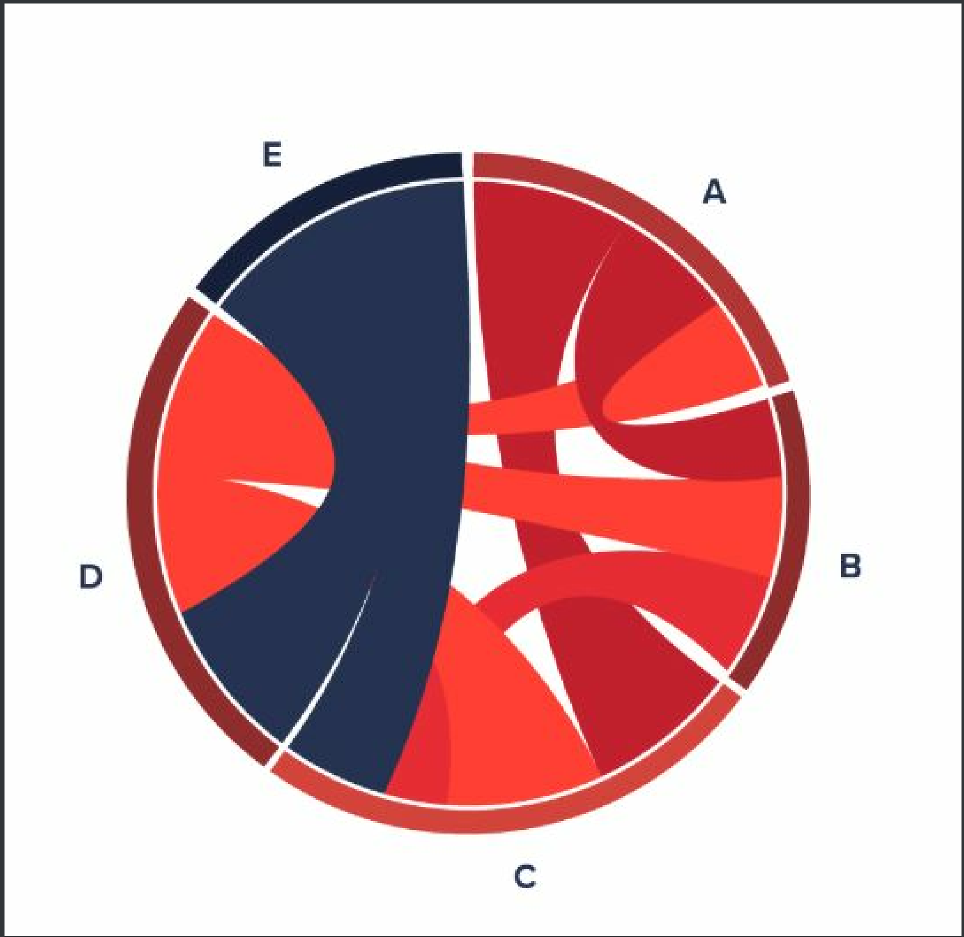


After a few trials, I got it here:



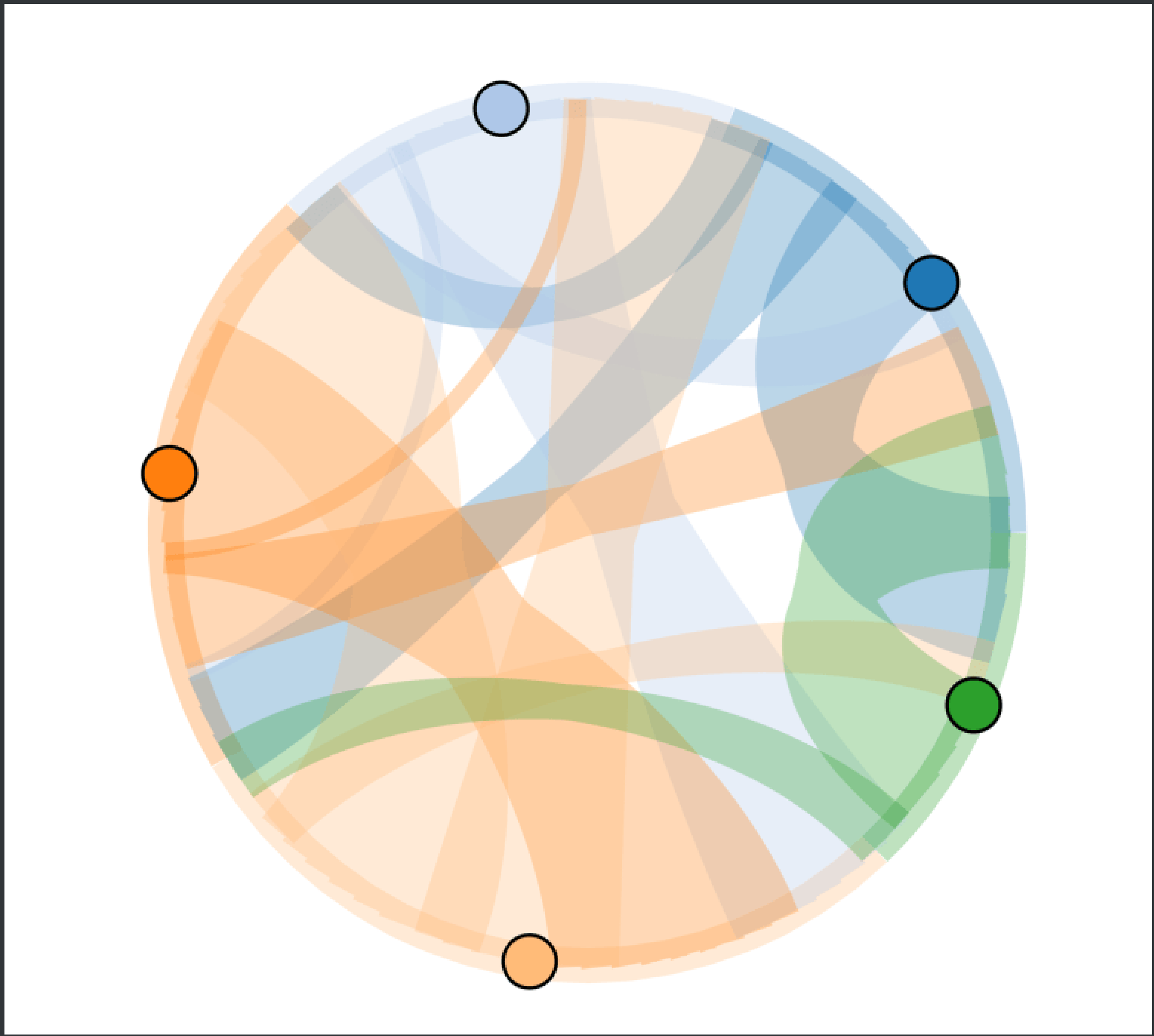
Generate a Figure like this -- It's called a Chord Diagram.

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[s] 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.