

HW1-1 Prompts to ChatGPT and Process

1. CRISP-DM framework

Prompt:

“write a python code to generate 100 random nodes on 2-D plane and make it to have trends to do linear regression to it”

GPT:

Gave me main program code as backbone.

Here's the Python code structured using the CRISP-DM framework:

python

複製程式碼

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# CRISP-DM Phase 3: Data Preparation
np.random.seed(0) # reproducibility
X = np.random.rand(100, 1) * 10 # Random X values between 0 and 10

# Define the linear relationship Y = 2.5X + 5 + some noise
m_true = 2.5 # True slope
b_true = 5 # True intercept
noise = np.random.randn(100, 1) # Random noise
Y = m_true * X + b_true + noise # Y with noise

# CRISP-DM Phase 4: Modeling
model = LinearRegression()
model.fit(X, Y) # Fit the model
Y_pred = model.predict(X) # Predicted Y values

# CRISP-DM Phase 5: Evaluation
# Print model parameters
print(f"Estimated Slope (m): {model.coef_[0][0]}")
print(f"Estimated Intercept (b): {model.intercept_[0]}")

# Visualize the results
plt.scatter(X, Y, color='blue', label='Data Points') # Actual data
plt.plot(X, Y_pred, color='red', label='Fitted Line') # Fitted regression line
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Linear Regression Fit')
plt.legend()
plt.show()
```

2. Make a GUI for website

Prompt:

“now provide a GUI(by streamlit or flask) which user can adjust coefficient of linear equation with. And then produce plot with scattered nodes on it.”

GPT:

Gave me proper instructions to install streamlit and made program for me.

Steps:

1. **Streamlit Setup:** Install `streamlit` if you haven't already:

bash

複製程式碼

```
pip install streamlit
```

2. **Create the Interactive App:**

- Users can adjust the slope, intercept, and noise level using sliders.
- The plot will display the scatter of random points along with the adjusted linear regression line.此圖將顯示隨機點的散佈情況以及調整後的線性迴歸線。

Streamlit Code:

python

複製程式碼

```
import streamlit as st
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Streamlit Title
st.title("Interactive Linear Regression Simulator")

# Step 1: User Input for coefficients using sliders
st.sidebar.header("Adjust the Linear Equation")
slope = st.sidebar.slider("Slope (m)", min_value=-10.0, max_value=10.0, value=2.5,
intercept = st.sidebar.slider("Intercept (b)", min_value=-10.0, max_value=10.0, va
```

```
intercept = st.sidebar.slider("Intercept (b)", min_value=-10.0, max_value=10.0, value=0.0)
noise_level = st.sidebar.slider("Noise Level", min_value=0.0, max_value=10.0, value=1.0)

# Step 2: Generate Random Data (X, Y) based on user input
np.random.seed(42) # For reproducibility
X = np.random.rand(100, 1) * 10 # Random X values between 0 and 10
noise = np.random.randn(100, 1) * noise_level # Random noise scaled by user choice
Y = slope * X + intercept + noise # Linear equation with noise

# Step 3: Perform Linear Regression (for visualization purposes)
model = LinearRegression()
model.fit(X, Y)
Y_pred = model.predict(X)

# Step 4: Plot the Data Points and the Regression Line
fig, ax = plt.subplots()
ax.scatter(X, Y, color='blue', label='Data Points') # Scatter plot
ax.plot(X, Y_pred, color='red', label='Fitted Line') # Fitted regression line
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_title(f"Linear Regression: Y = {slope:.2f}X + {intercept:.2f}")
ax.legend()

# Step 5: Display the Plot in Streamlit
st.pyplot(fig)

# Step 6: Display the equation parameters
st.write(f"### Equation Parameters: Y = {slope:.2f}X + {intercept:.2f}")
st.write(f"### Noise Level: {noise_level}")
```

How This Works:

1. **Sidebar Controls:** Users can adjust the slope, intercept, and noise level of the generated data using the sliders in the sidebar.
2. **Scatter Plot:** The main area displays the scatter plot of the generated data points and the regression line fitted based on the user-defined slope and intercept.
3. **Dynamic Plotting:** As the user adjusts the sliders, the plot updates automatically to reflect the new equation and noise level.

To Run the App:

1. Save the above code in a Python file, say `linear_regression_app.py`.
2. Open a terminal or command prompt and run:

```
bash
```

[複製程式碼](#)

```
streamlit run linear_regression_app.py
```

3. A web browser will open showing the app with interactive sliders and the plot.

Dependencies:

Make sure to have the following libraries installed:

```
bash
```

[複製程式碼](#)

```
pip install streamlit scikit-learn matplotlib numpy
```

This app allows users to explore how changing the slope, intercept, and noise affects the distribution of points and the resulting regression line interactively! Let me know if you'd like to

3. Deploy app with github and Streamlit Cloud

Prompt:

“display this streamlit based website on github if possible”

GPT:

made me a tutorial

Yes, you can deploy your Streamlit-based app on **GitHub** and run it directly using **Streamlit Cloud** (previously known as Streamlit Sharing). Below are the steps to get your Streamlit app hosted on GitHub and deploy it:

Steps to Publish Streamlit App on GitHub and Deploy

Step 1: Create a GitHub Repository

1. Go to [GitHub](#), log in, and create a new repository.
 - Click on the "+" symbol at the top-right and select "New repository."
 - Name the repository (e.g., `streamlit-linear-regression-app`) and make it public.
 - Add a description and click on **Create repository**.

Step 2: Push Your Streamlit Code to GitHub

1. Initialize your local project and push your code to GitHub.

If you haven't already initialized your local directory with git, follow these steps:

bash

複製程式碼

```
# Initialize a git repository
git init

# Add all files to the repository
git add .

# Commit the changes
git commit -m "Initial commit of Streamlit app"
```



```
# Add the GitHub repository URL
git remote add origin https://github.com/yourusername/streamlit-linear-regression-app

# Push the code to GitHub
git push -u origin master
```

複製程式碼

Ensure that your repository contains:

- The `linear_regression_app.py` script.
- A `requirements.txt` file listing the dependencies for the project.

Step 3: Add `requirements.txt` for Dependencies

Create a `requirements.txt` file in the repository to list the dependencies for Streamlit and the libraries used:

```
bash
```

複製程式碼

```
# requirements.txt
streamlit
scikit-learn
matplotlib
numpy
```

Step 4: Deploy to Streamlit Cloud

1. Once your code is on GitHub, go to **Streamlit Cloud**: streamlit.io/cloud and log in.
2. Click on **"New app"**.
3. Select the repository (`streamlit-linear-regression-app`) from your GitHub.

```
streamlit
scikit-learn
matplotlib
numpy
```

複製程式碼

Step 4: Deploy to Streamlit Cloud

1. Once your code is on GitHub, go to **Streamlit Cloud**: streamlit.io/cloud and log in.
2. Click on "New app".
3. Select the repository (`streamlit-linear-regression-app`) from your GitHub.
4. In the "Main file path", enter the name of your Python file, e.g., `linear_regression_app.py`.
5. Click **Deploy**.

That's it! 🎉

Your app will be live on Streamlit Cloud, and you'll get a URL like this:

```
https://yourusername-streamlit-linear-regression-app.streamlit.app
```

This URL will allow others to access and interact with your app directly.

Additional Tip:

- You can add a `README.md` file to your GitHub repository with a description of your app and a link to the live app.
- Make sure your `requirements.txt` lists all necessary dependencies so that the app can run smoothly on Streamlit Cloud.

Let me know if you need further assistance or clarification!

