

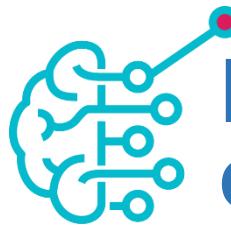
Which of the following is/are CORRECT?

- A. If we want to use the model for prediction, we will pass in a list of input and these inputs correspond to the features that we use to build the model.
- B. It is important to periodically retrain the model.
- C. A model's performance on training data is always a good indicator of its performance on new, unseen data.
- D. Re-training a model means starting from scratch with only new data.
- E. You must always re-train a model if you want to make predictions on new data.
- F. Overfitting occurs when a model performs well on training data but poorly on test data.
- G. The prediction step does not require access to the original training data.



Multiple Choice

Model Deployment



Machine Learning / Data Science Life Cycle

Cross-Industry Standard Process for Data Mining (CRISP-DM)

1. Business Understanding

- Use case requirements and objectives
- Assess situation
- Development plan

6. Deployment

- Deployment plan
- Plan monitoring & maintenance
- Final reports and visualizations
- Project Review

5. Evaluation

- Model testing
- Results evaluation
- Next steps determination

2. Data Understanding

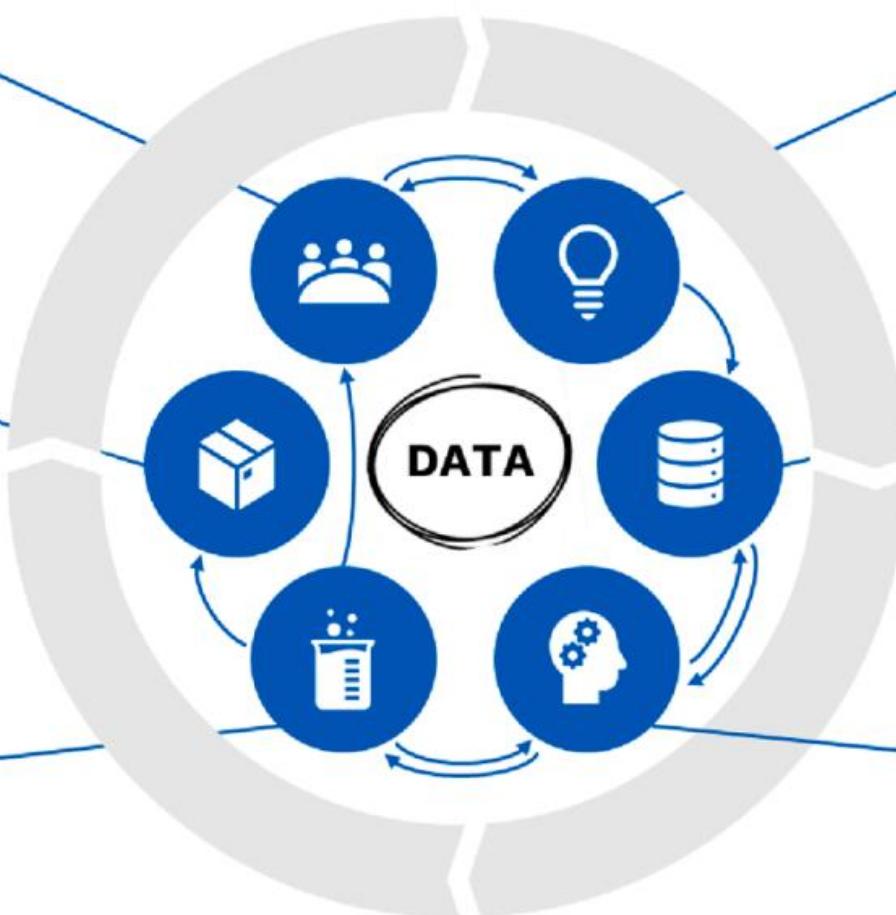
- Data sources identification
- Data gathering & integration
- First explorative & insights
- Data quality

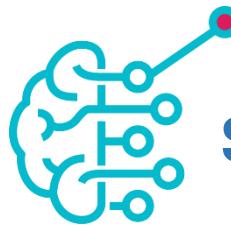
3. Data Preparation

- Data cleaning & transformation
- Further data exploration
- Dimensionality reduction
- Features engineering

4. Modelling

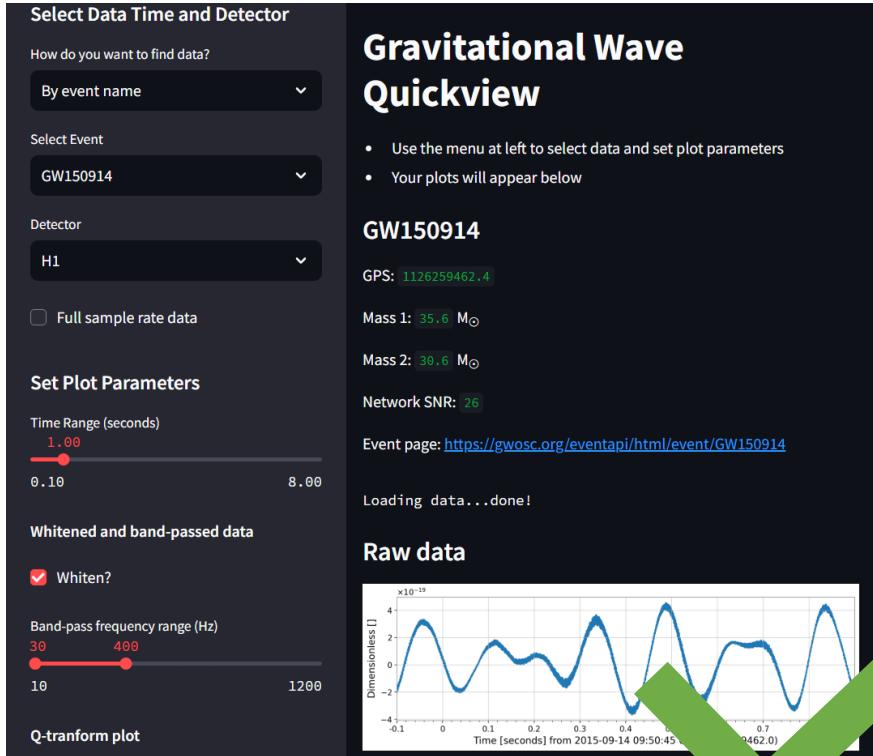
- Model class selection & design
- Build test design
- Hyperparameter optimization
- Models training & validation

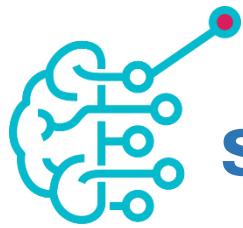




streamlit_app.py

- Design overall webpage aesthetics

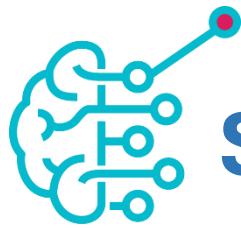




streamlit_app.py

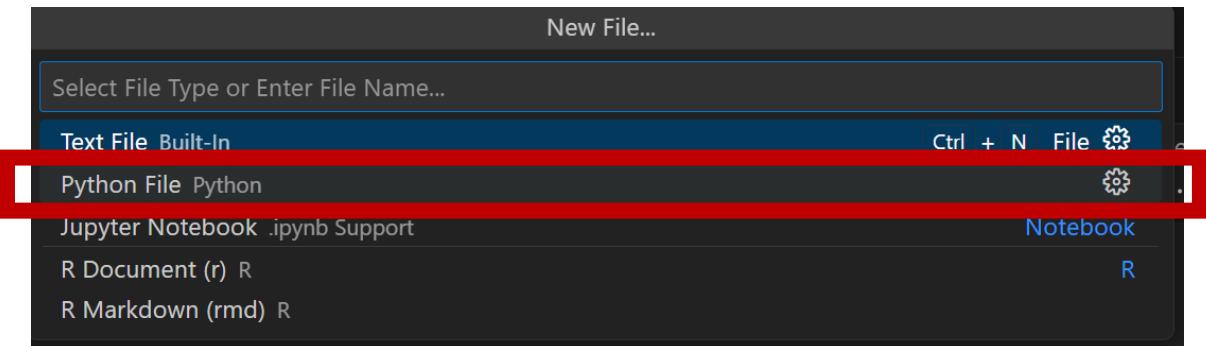
```
import joblib  
model = joblib.load("model.pkl")
```

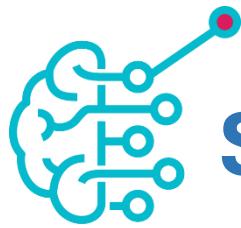
- Load trained model code
- Read new unseen data (user input data)
- Data pre-processing codes
- Prediction codes



Streamlit app development

- Code your **app.py** file **locally first**
 - You can code directly in ***.py**
 - Alternatively, code in ***.ipynb** first and then export it to ***.py**
- Use **streamlit run app.py** to launch app locally and make adjustments
- Once satisfied with the app, we will upload to Streamlit cloud for easier access





Streamlit template

- Ensure you have **joblib** library installed

- Save trained model

```
joblib.dump(model, "model.pkl")
```

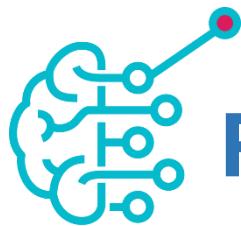
- Load trained model

```
joblib.load("model.pkl")
```

- Import all required libraries in the Python code

```
import joblib  
import streamlit as st  
import numpy as np  
import pandas as pd
```

```
## Load trained model  
model = joblib.load("model.pkl")
```



Retrieving user inputs

HDB Resale Price Prediction

Select Town

ANG MO KIO

Select Flat Type

2 ROOM

Select Storey Range

10 TO 12

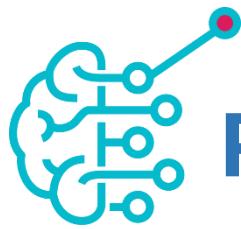
Floor Area (sqm)

30 70 200

Predict Price

```
## Streamlit app
st.title("HDB Resale Price Prediction")  
  
## User inputs
town_selected = st.selectbox("Select Town", towns)
```

```
## User inputs
floor_area_selected = st.slider("Floor Area (sqm)",  
                                 min_value=30,  
                                 max_value=200,  
                                 value=70)
```



Predicting user input

Select Flat Type
2 ROOM

Select Storey Range
10 TO 12

Floor Area (sqm)
30 70 200

Predict Price

Predicted Resale Price: \$391,799.03

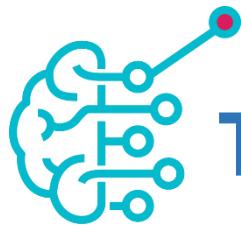
```
## Predict button
if st.button("Predict HDB price"):

    ## Create dict for input features
    input_data = {
        'town': town_selected, ...
    }

    ## Convert input data to a DataFrame
    df_input = pd.DataFrame({
        'town': [town_selected], ...
    })

    ## Data preprocessing
    ...

    ## Predict
    y_unseen_pred = model.predict(df_input)[0]
    st.success(f"Predicted Price: ${y_unseen_pred:,.2f}")
```



Testing out streamlit_app.py locally

The screenshot shows a terminal window in VS Code. The terminal tab is selected at the bottom. The command history shows:

```
PS C:\Users\    > cd Downloads
PS C:\Users\ \Downloads> streamlit run app.py
```

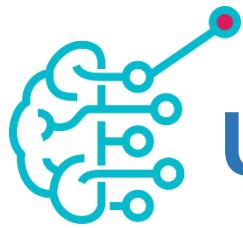
Below the terminal, a message says "You can now view your Streamlit app in your browser." and provides local and network URLs.

Annotations with yellow numbers 1 and 2 highlight the "Terminal" menu item and the "New Terminal" option respectively. A red box highlights the command history. Brackets on the right side group annotations for the command history and the URLs.

Change directory to where the app.py and model.pkl files are

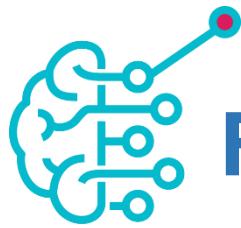
streamlit run app.py

Hold **Ctrl** and click on the Local URL to view your app



Uploading files to GitHub

1. Create GitHub account to store the model and code files
2. Create **PUBLIC** repository and upload the following
 - `requirements.txt`
 - `streamlit_app.py`
 - `model.pkl`
 - Supporting files



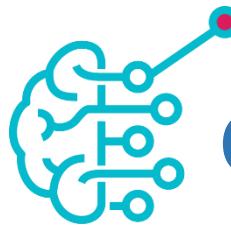
Files required to be uploaded to GitHub

- **requirements.txt**
 - State all required libraries in Python code so that Streamlit can install the libraries and run the code
- **streamlit_app.py**
 - Streamlit app (user interface) code
- **model.pkl**
 - Trained model
- Supporting files
 - Background image
 - Dataset (if you want to show additional features in your app)

The screenshot shows a GitHub repository named 'titanic'. The repository is public and was generated from the 'streamlit/blank-app-template'. It contains the following files:

File	Description	Last Commit
.devcontainer	Initial commit	last month
.github	Initial commit	last month
.gitignore	Initial commit	last month
LICENSE	Initial commit	last month
README.md	Initial commit	last month
requirements.txt	Update requirements.txt	last month
streamlit_app.py	Update and rename app....	last month
titanic_model.pkl	Rename titanic_model_pi...	last month

The repository has 0 stars, 0 forks, and 0 watching. It also has no releases published.



Creating Streamlit app

1. Create Streamlit account
2. Create app
3. Deploy a public app from GitHub

The screenshot shows the Streamlit app creation process. At the top right, a red box highlights the "Create app" button with the number "1". Below it, the "apps" section lists two apps: "mldp · main · app.py" and "titanic · main · streamlit_app.py". In the center, a blue arrow points left to "Back". Below it, the heading "What would you like to do?" is followed by three options:

- Deploy a public app from GitHub** (highlighted with a red box and the number "2").

My code is ready on a GitHub repo, and it is totally awesome.

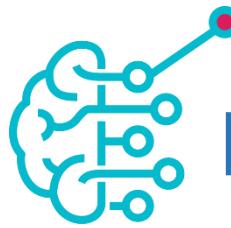
[Deploy now](#)
- Deploy a public app from a template**

I want to see what kind of amazing concoctions you have for me.

[Check out templates](#)
- Deploy a private app in Snowflake**

I want unlimited enterprise-grade apps, with the security of Snowflake.

[Start trial →](#)



Deploying Streamlit app

← Back

Deploy an app

Repository ②

/repo

Branch

master

Main file path

streamlit_app.py

App URL (optional)

Paste GitHub URL

1

Advanced settings

2

Deploy

5

// MLDP • MACHINE LEARNING FOR DEVELOPERS (CAI2C08)

TEMASEK POLYTECHNIC | School of Informatics & IT

Copy app .py link from GitHub
Do NOT paste repository link!

← Back

Deploy an app

Repository ② /repo

Branch master

Main file path streamlit_app.py

App URL (optional) .streamlit.app

Advanced settings

Python version 3.13

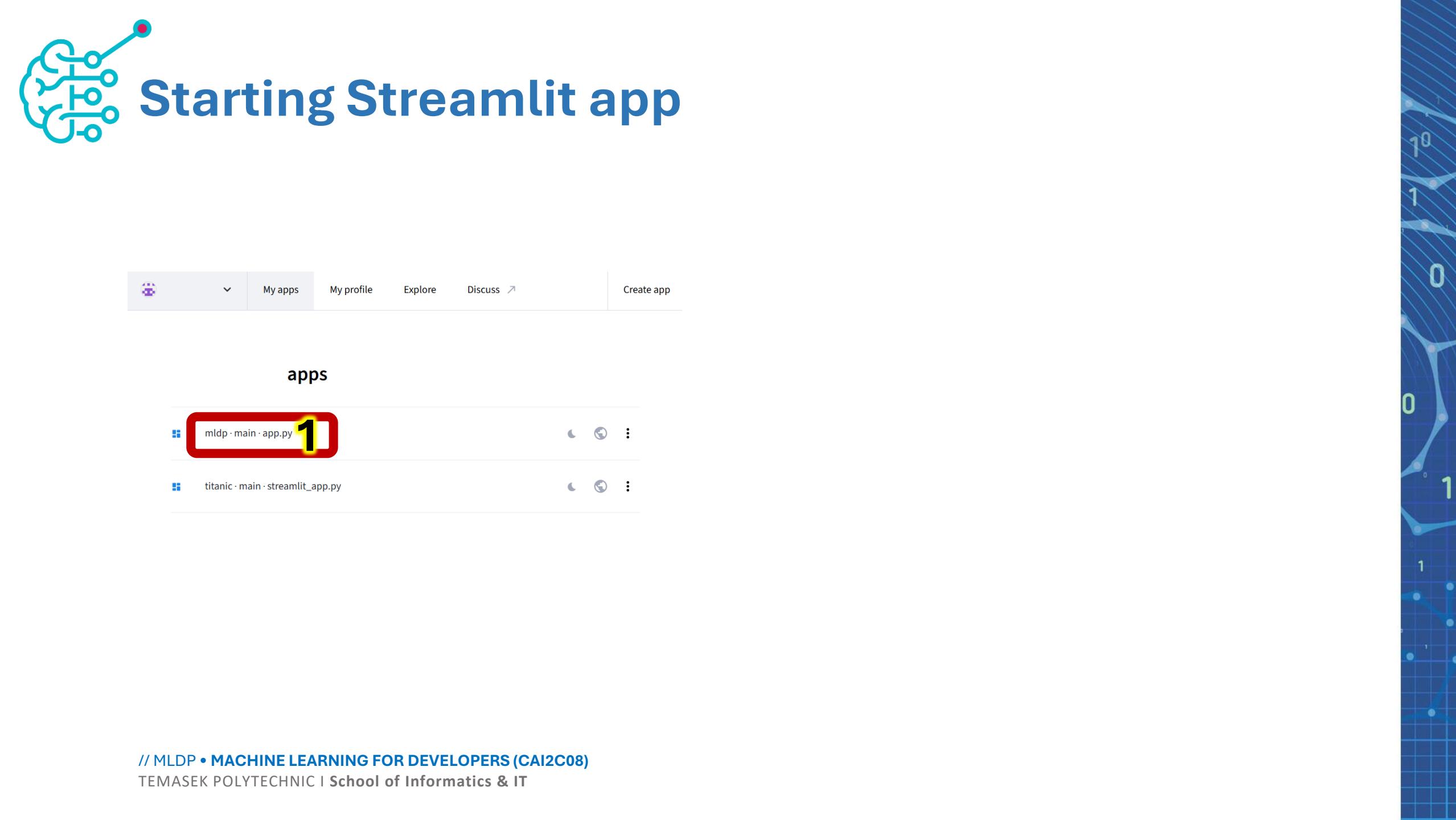
Secrets

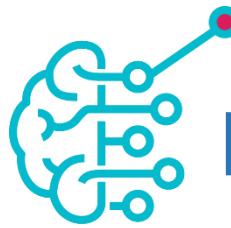
Provide environment variables and other secrets to your app using TOML format. This information is encrypted and served securely to your app at runtime. Learn more about Secrets in our docs.

Changes take around a minute to propagate.

```
DB_USERNAME = "myuser"  
DB_TOKEN = "abcdef"  
  
[some_section]  
some_key = 1234
```

Save 4

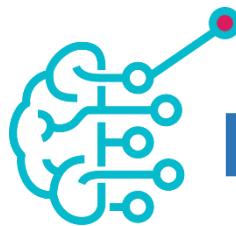




Project Specification Deployment Rubrics

Component	Evaluation Criteria	A (≥ 80%)	B (70 to <80%)	C (60 to <70%)	D (50 to <60%)	F (<50%)
Deployment of Model (10%)	Correct prediction of output without error (4%)	<ul style="list-style-type: none">Web app produces correct outputs for all tested casesNo errors or crashesInclude input validation and user-facing error message(s)	<ul style="list-style-type: none">Web app produces correct outputs in all tested casesMinor warnings that do not affect outputs	<ul style="list-style-type: none">Web app produces correct outputs for most tested casesOccasional errors or crashes	<ul style="list-style-type: none">Web app produces correct outputs for some tested casesFrequent errors or crashes	<ul style="list-style-type: none">Web app fails to produce correct outputsPersistent errors or crashesUnusable for intended purpose

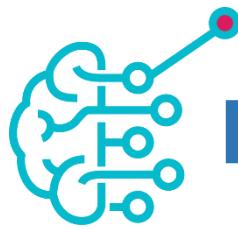
- Error: Prediction is counterintuitive with the inputs
 - e.g. Increase sqft but houseprice decreases



Project Specification Deployment Rubrics

Component	Evaluation Criteria	A (≥ 80%)	B (70 to <80%)	C (60 to <70%)	D (50 to <60%)	F (<50%)
Deployment of Model (10%)	Interactive application (6%)	<ul style="list-style-type: none">Highly interactive, responsive, and user-friendly appAll interactive elements (buttons, forms, feedback) work smoothly and enhance user experienceDesign and visuals are appealing, and labels are suitable for target audience	<ul style="list-style-type: none">Interactive, responsive, and user-friendly appDesign and visuals are consistent (e.g., at most 2 minor issues), and labels are mostly clear (e.g., at most 2 vague terms) for target audience	<ul style="list-style-type: none">Interactive, responsive, and user-friendly appSome elements may not respond as expected; user experience is acceptable but not engaging	Largely static interface	<ul style="list-style-type: none">No interactivityPresent many broken elements that do not work or respond poorly; user experience is frustrating

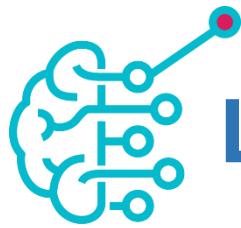
- Appealing design and visuals
 - e.g., Blank vs appropriate background image
- Suitable for target audience
 - e.g., Labels easily understood?
 - Instead of typing in BMI, ask for height and weight instead.



E-Learning & References

- Choose a Streamlit app from the gallery
 - Comment on the aesthetics and user interface:
 - 2 areas that are done well
 - 1 area for improvement

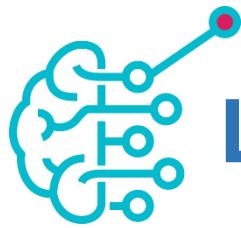
Topic	Source
Streamlit	<ul style="list-style-type: none">• https://streamlit.io/gallery• https://docs.streamlit.io/deploy/streamlit-community-cloud/get-started• https://www.datacamp.com/tutorial/streamlit• https://demo-ai-assistant.streamlit.app/?ref=streamlit-io-gallery-favorites
GitHub	<ul style="list-style-type: none">• https://learn.microsoft.com/en-us/training/modules/introduction-to-github/• https://learn.microsoft.com/en-us/training/modules/upload-project-github/



Learning Outcomes

1. [Machine Learning Fundamentals] Explain the fundamental concepts of machine learning;
 - 1.1 describe machine learning and its goals related to learning;
 - 1.2 identify different data types and characteristics or data;
 - 1.3 play explain the learning tasks and performance of machine learning algorithms.

2. [Machine Learning Development] Apply machine learning methods to solve problems;
 - 2.1 explain machine learning concepts;
 - 2.2 describe machine learning algorithms;
 - 2.3 perform data analysis and necessary data cleaning;
 - 2.4 train machine learning models;
 - 2.5 evaluate machine learning models.



Learning Outcomes

3. [Model Deployment] Deploy a machine learning model to an application;
 - 3.1 identify a platform and framework for the deployment of an application;
 - 3.2 develop an application on a platform;
 - 3.3 deploy a machine learning model on to an online platform.