Week 3: Feature Engineering and Intro to ML

MAISI

Building Interpretable AI in Healthcare
Michigan Data Science Team - Winter 2026

Week 3 Agenda

01. Icebreaker!

Every great project work session starts off with a great icebreaker.

02. Feature Engineering

What is feature engineering, and why is it an important step to getting meaningful model output?

03. Intro to Machine Learning

What is machine learning, what can we do with it, and what kinds of models can we use?

04. Hands-On Data Science!

It's time to start training those models, engineering those features, and getting closer to our end goal!

01 Fun Icebreaker!!

Get to know us and start to learn about each other with a fun icebreaker!

Icebreaker - Week 3

What would your dream exotic pet be? (No cats, dogs, hamsters, etc.)

Share with the people around you!! :)

Note: that's actually a Fennec Fox!

02 Feature Engineering

What is feature engineering, and why is it an important step to getting meaningful model output?

What is Feature Engineering?

- Feature engineering is the process of turning cleaned (yay for EDA!) input data into interpretable and ready-to-use features in a machine learning model.
    - The first steps in this are EDA and feature selection, which we covered last week.

- Processes Involved in Feature Engineering
    - Feature Creation - combining existing features into composite measures
    - Feature Selection - choosing the most relevant features (preprocessing)
    - Feature Scaling - making sure all features contribute equally to improve model

What does Feature Engineering Do?

- Improves Model Accuracy
  - Choosing the right features helps the model learn better
    - Better learning, better predictions!
- Reduces Overfitting
  - By using fewer features that hold more importance, we prevent the model from memorizing the data on which we train it
- Boosts Interpretability!!
  - When we choose our features well, we can establish a better understand of why our model makes its predictions

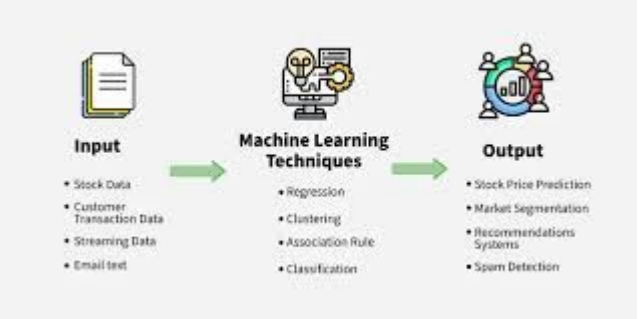
03 Intro to Machine Learning

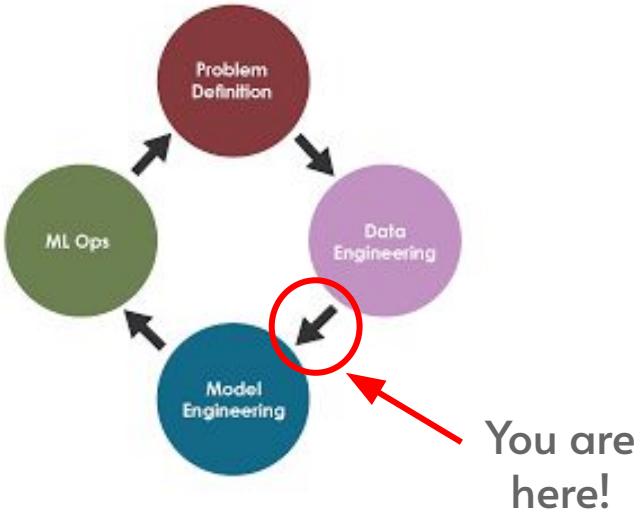
What is machine learning, what can we do with it, and what kinds of models can we use?

What is Machine Learning?

- **Machine Learning vs. AI:** Machine Learning is a subset of AI that enables computers to *learn* from data and make predictions, without needing to be explicitly programmed for every specific task.

- Main Types of Machine Learning
  - **Supervised Learning:** models trained on labeled training data to predict outcomes.
  - Unsupervised Learning: models that find patterns in unlabeled data.
  - Reinforcement Learning: models learn through rewards system implemented by developers.





The Supervised Machine Learning Process

- **Collect/Process the Data:** In order to do supervised machine learning, we need data that is "labeled," which means that each observational unit (row) has a known correct target value, which we call the "label."

- **Split the Dataset:** We need to do a train-test split, where we divide our data into groups we train our model on (typically about 70-80%) and that on which we test it.

- **Train Your Model:** Feed the data into a model of your choice (which we will cover in a second), giving inputs as 'x' and your target variable as 'y.'

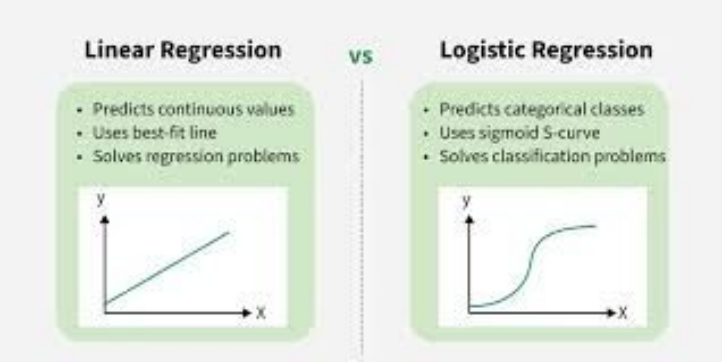
- **Validate and Test:** Run the model on your training data and evaluate the accuracy of your output. If there is more to be desired, try adjusting weights or changing model!

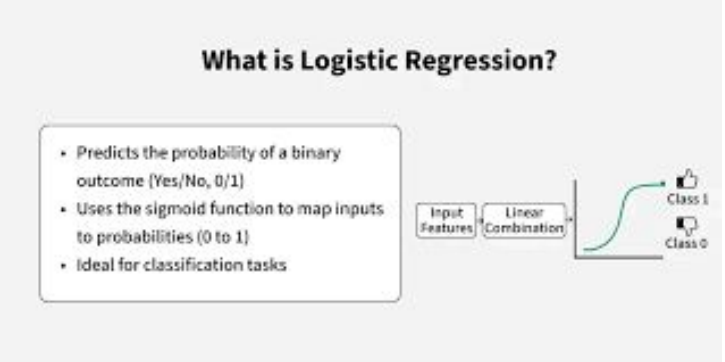
ML Algorithms - Binary Logistic Regression

- **Binary logistic regression** works well in cases where we have a *binary target variable*
  - Ex. Stroke/No Stroke, Lung Cancer/None

- Considered a classification algorithm, as it attempts to fit points along a sigmoid curve ranging from [0, 1] for our binary target

- Models the **log-odds** (ln(odds of an event)) as a linear combination of independent variables 'x,' where a certain threshold (usually 0.5) is used to determine to which category an observation with given log-odds should be assigned.





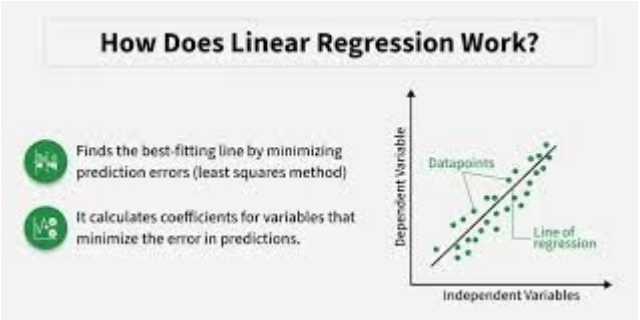
ML Algorithms - (Multiple) Linear Regression

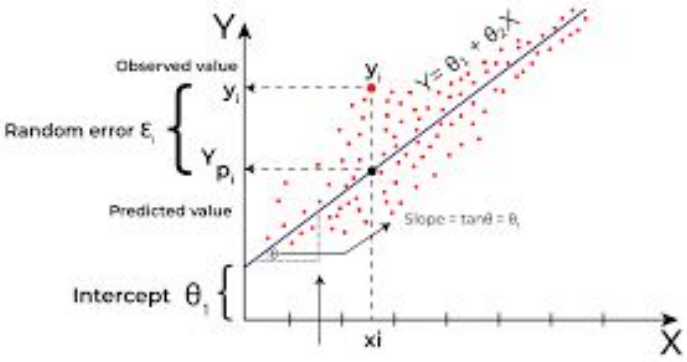
- Boils down to a very familiar concept: **$y = mx + b$**
  - If you're doing <u>multiple</u> linear regression, this formula includes multiple input vars:
    $y = b + m_1x_1 + m_2x_2 + \ldots + m_nx_n$

- Plots a "best fit" line through the data that *minimizes squared errors* of observed vs. expected values.
  - Always fitted **linearly**

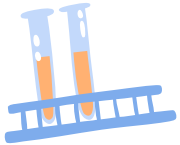
- Goal: find a best fit line that most accurately models the spread of data points for the target.

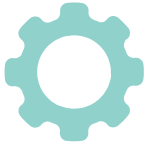
- Key Assumptions
  - Assumes linear relationship between x and y
  - (For multiple) No multicollinearity





04

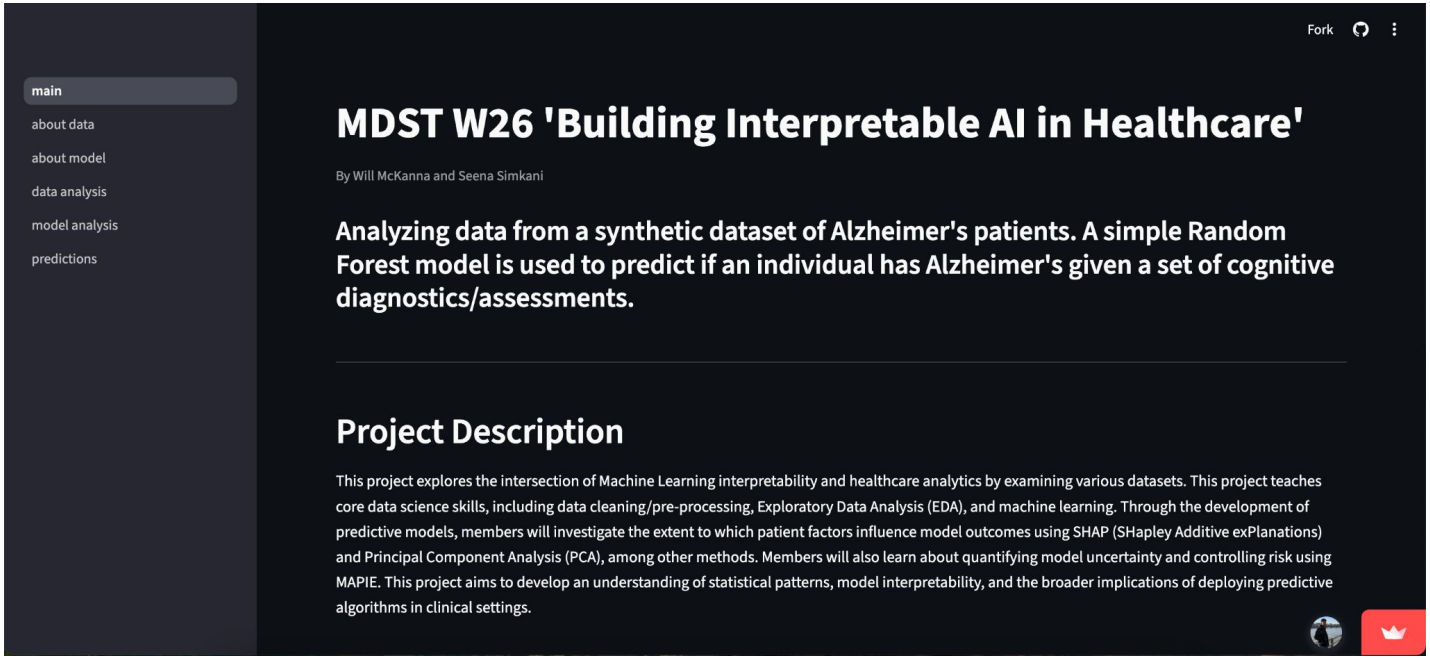




References

Here are a handful of useful guides to high-utility functions, Git demos, and Colab explanations!

Our Demo! (for reference)



Link:
https://mdst-ai-in-healthcare.streamlit.app/

Project Timeline

Week 1 (01/25): Icebreaker/EDA intro, choosing a good dataset

Week 2 (02/01): Data Visualization, Further EDA

Week 3 (02/08): Feature Engineering and Intro to Machine Learning

Week 4 (02/15): Machine Learning and Model Optimization

Week 5 (02/22): Conformal Predictions, Risk Control, and Multi-Class Calibration

NO MEETING 03/01 OR 03/08 – SPRING BREAK

Week 6 (03/15): SHAP Values and Feature Importance Interpretation

Week 7 (03/22): Intro to Streamlit and Project Work Time

Weeks 8 + 9 (03/29 + 04/05): Final Project work time + Data Science Night prep!

Important Functions to Know (pt. 1)

Syntax	Description
import pandas as pd	Import the **pandas** library, using the alias **pd** for convenience
import numpy as np	Import the **NumPy** library, using alias **np** for numerical operations
import seaborn as sns	Import **Seaborn** for statistical data visualization
df = pd.read_csv('file.csv')	Load a **CSV** file into a **pandas DataFrame** for data manipulation
df.head()	Returns the **first 5 rows** of the DataFrame, useful for quickly inspecting data
df.info()	Provides a concise **summary** of the DataFrame, including **data types** and **non-null values**

Important Functions to Know (pt. 2)

Syntax	Description
df.describe()	Generates **descriptive statistics** of numerical columns (mean, median, quartiles, etc.)
df[‘column_name’]	Access a **specific column** in the DataFrame, works like a key in a dictionary
df.drop(columns=[‘col’...])	**Drops specified columns** from the DataFrame,use inplace=True to modify the original DataFrame
df.index	Returns the **index labels** of the DataFrame
df.isnull()	returns a DataFrame of **boolean values**, where each entry indicates whether the corresponding value in df is NaN (missing)

Important Functions to Know (pt. 3)

Syntax	Description
df[‘column’].value_counts()	Returns the **count** of **unique values** in a specific column
df[‘column’].mean()	Returns the **mean value** of a numerical column
df.corr()	Computes **correlation** for numerical columns to understand relationships
df.columns	Lists all **column names** in the **DataFrame**, useful for renaming or viewing dataset structure
df.shape	Returns **# of rows and columns**

Important Functions to Know (pt. 4)

Syntax	Description
df.rename(columns={'old' : 'new'}, inplace=True)	Renames **specific columns** to new names
df.groupby(‘category’)[‘value’]	Groups the DataFrame using a specified column to perform **aggregate functions** (e.g., .sum(), .mean())
df[‘new_column’] = df[‘column’].apply(function)	**Applies** a function to each element or column/row
df.shape	Returns a **tuple** representing the **dimensions** (rows, columns) of the DataFrame
df.dropna()	Removes rows or columns containing **missing values**

Important Functions to Know (pt. 5)

Syntax	Description
df.to_numpy()	Converts Pandas DataFrame into a **NumPy array**
np.shape	Returns a **tuple** representing the **dimensions** (rows, columns) of the array (similar to df.shape)
np.reshape()	**Reshapes** an original numpy array to the specified dimensions (rows, columns)
np.array()	Creates an **N-dimensional array** (ndarray) which is more **memory efficient** than standard python lists
np.unique()	Returns the **unique elements** in a NumPy array (also returns the **counts of each element** given the keyword argument, return_counts=True)

Important Functions to Know (pt. 6)

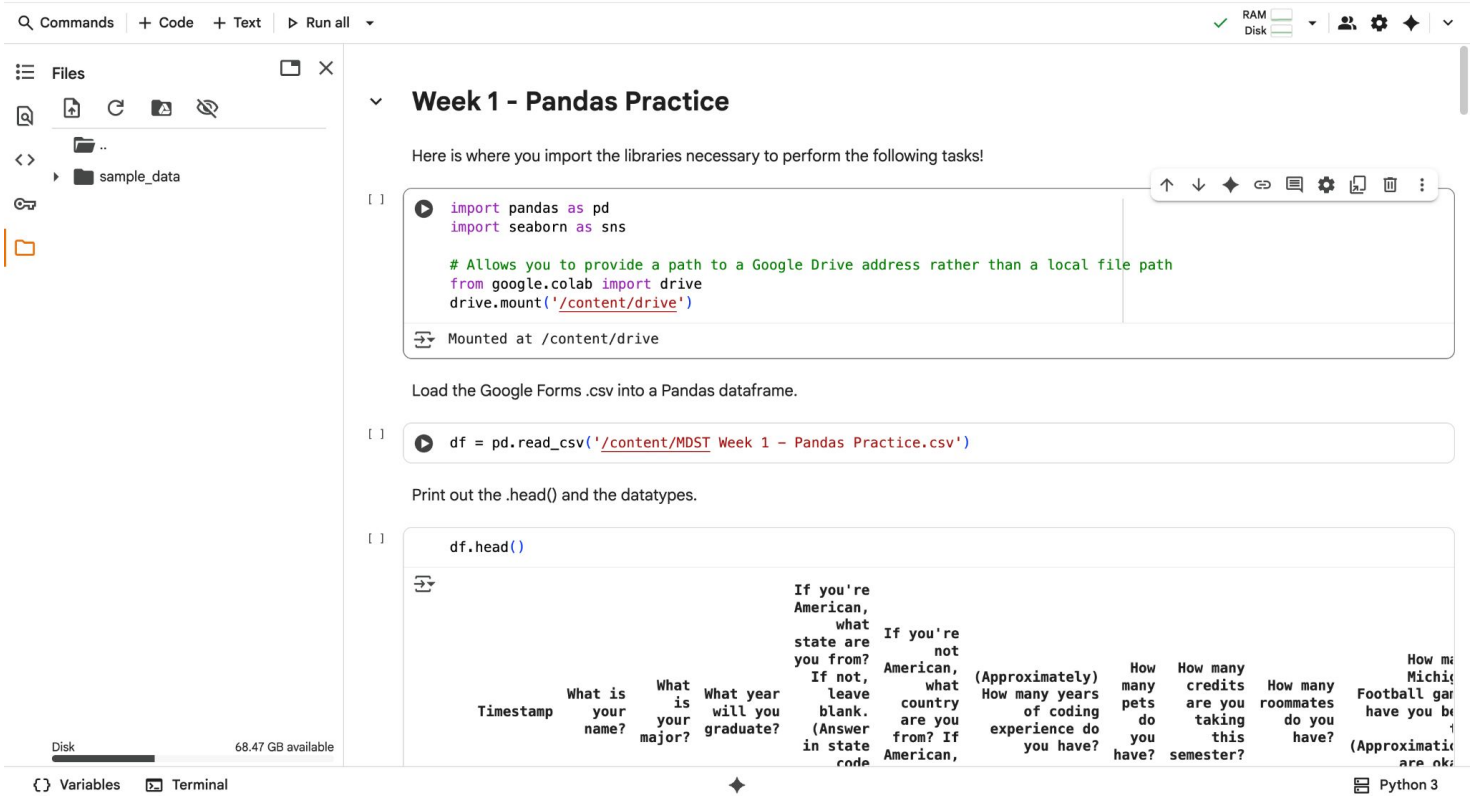
Syntax	Description
pd.cut()	Turns **continuous numerical data** into **categorical data**
pd.get_dummies(drop_first = True)	**Dummy encodes** categorical data (used with drop_first argument)
OrdinalEncoder()	Function for **encoding categorical information** that has an order
train_test_split(X, y, test_size=0.2, random_state=42)	**Splits data** into a **training set** and a **testing set**. The test set size can be changed
LinearRegression()	Used for initializing a **linear regression** model
LogisticRegression()	Used for initializing a **logistic regression** model

Important Functions to Know (pt. 7)

Syntax	Description
StandardScaler()	Turns **continuous numerical data** into **categorical data**
model.fit(X_train, y_train)	**Fits a model** to the training data
model.predict(X_test)	Uses the **test data** to make predictions
.coef	Attribute for getting the **models coefficients** (for linear and logistic regression)
mean_squared_error()	Calculates the **average squared distance** between the predicted and actual data points
mean_absolute _error()	Calculates the **average distance** between the predicted and actual values

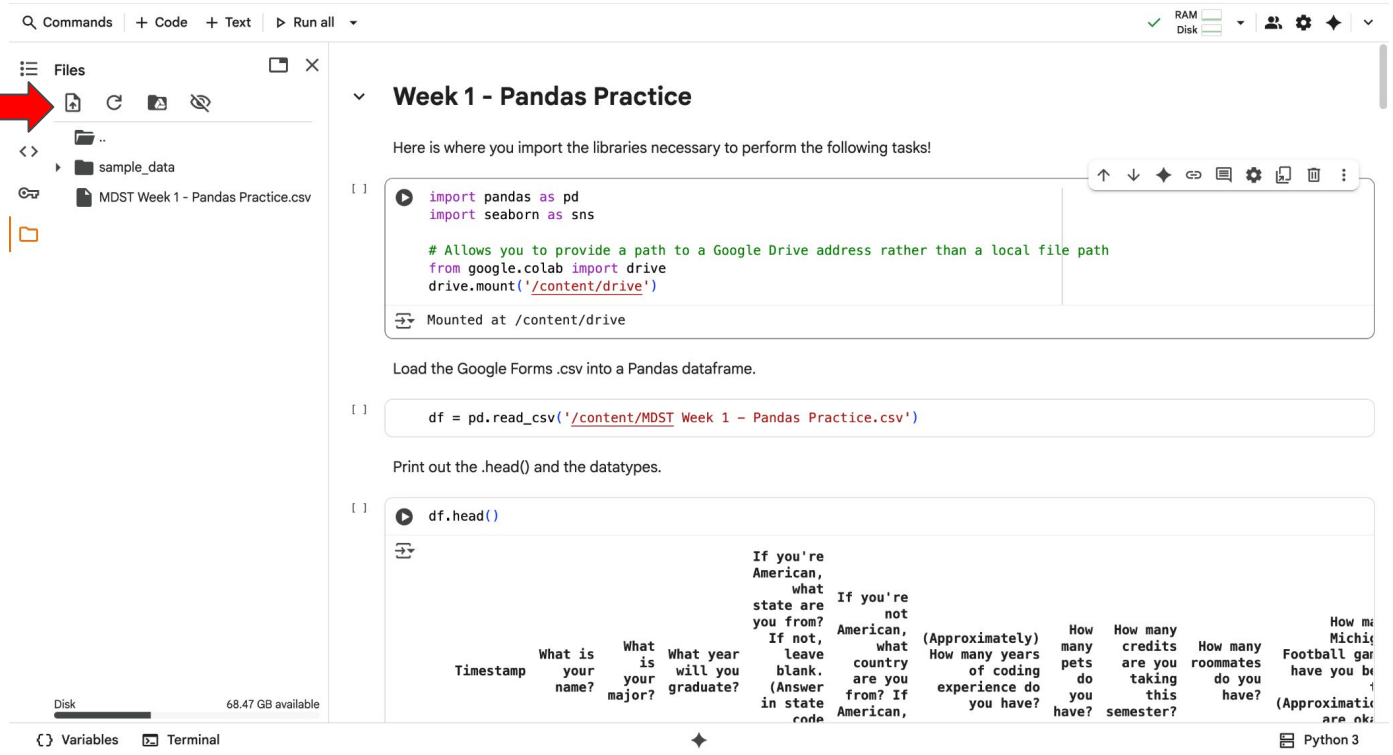
How to Upload Files into Colab

Click on the folder in the sidebar



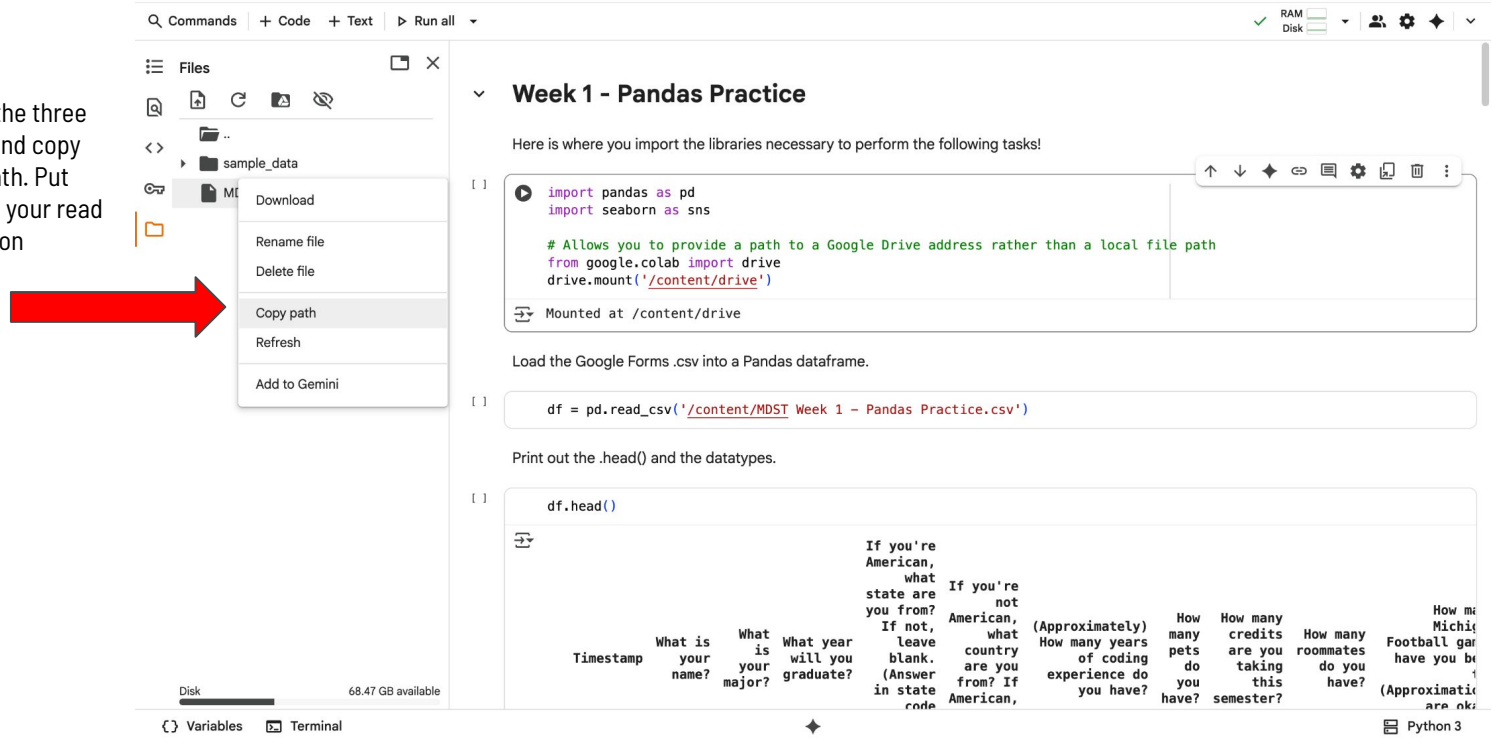
How to Upload Files into Colab

Click the upload button and select the file you want to upload



How to Upload Files into Colab

Click the three dots and copy the path. Put this in your read function



How to Mount Your Google Drive

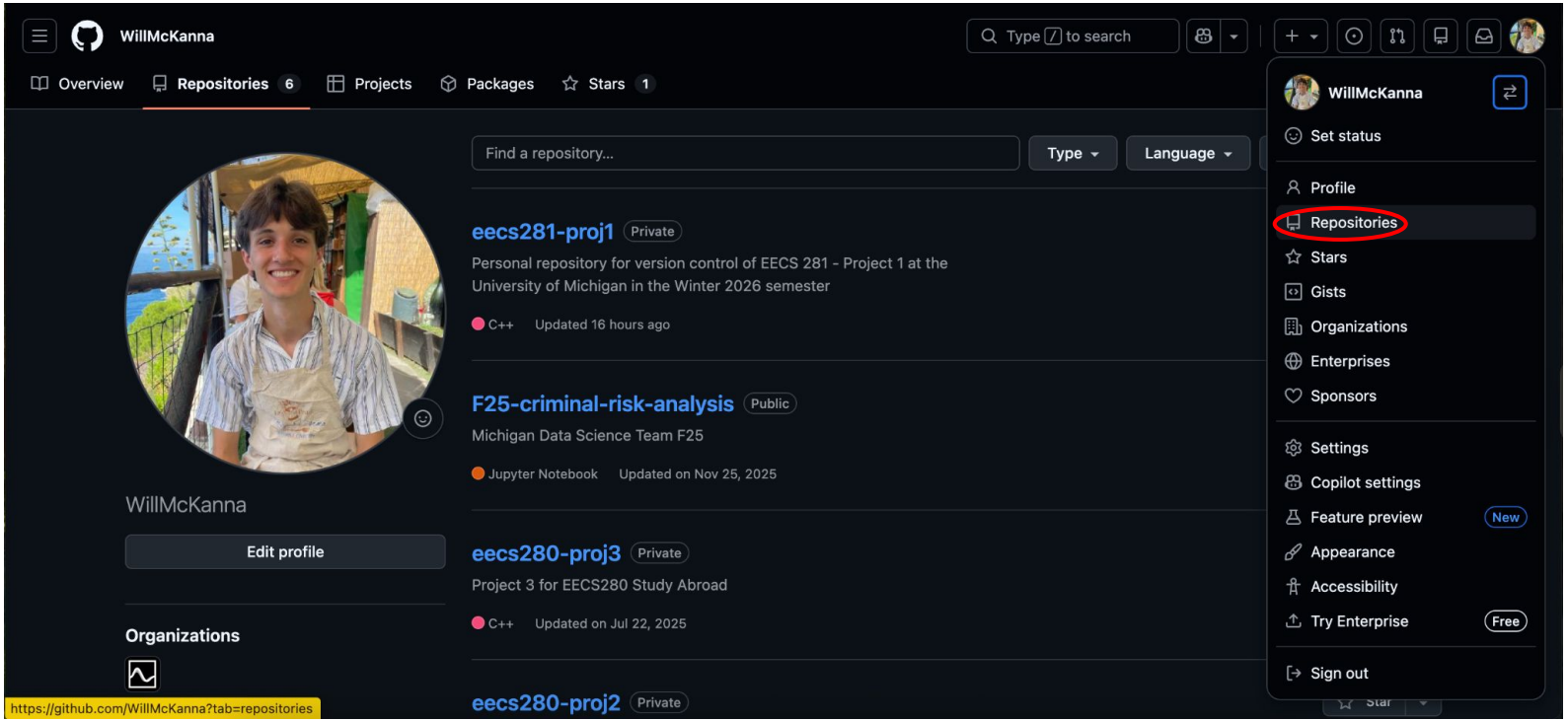
```
from google.colab import drive

drive.mount('/content/drive')


pd.read_csv('/content/drive/MyDrive/[FILE NAME]')
```

****NOTE:** If you saved your dataset in a folder (not just loose in your MyDrive folder), you will need to add further code to the file path before the file name in the final line. For example, if I saved my data called “alzheimers.csv” in a folder called “MDST-W26,” my read_csv statement would read:

```
pd.read_csv('/content/drive/MyDrive/MDST-W26/alzheimers.csv')
```

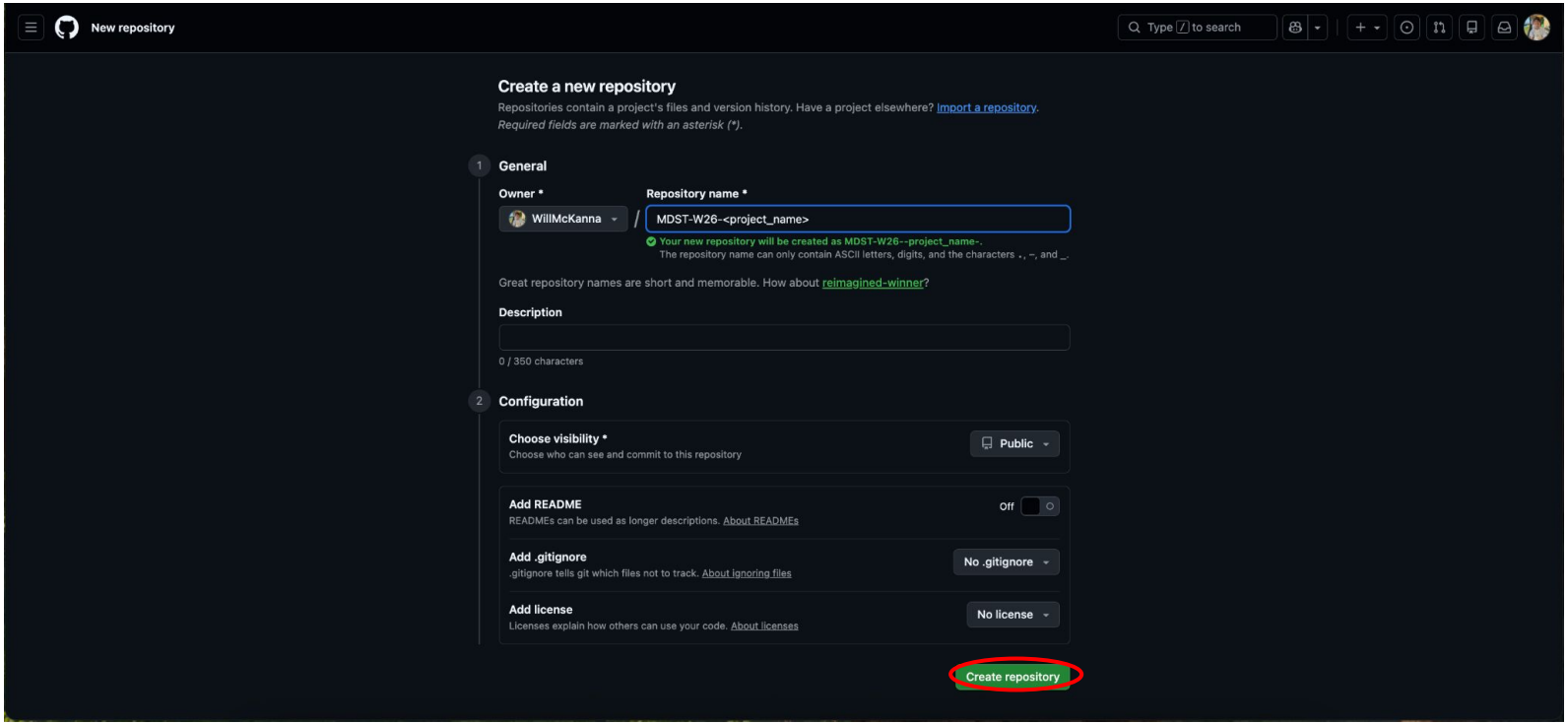
How to Create a GitHub Repository (yay for version control!)

I. Navigate to GitHub, click your profile picture, and select “Repositories.” Then, click the green “New” button next to the “Language” dropdown menu



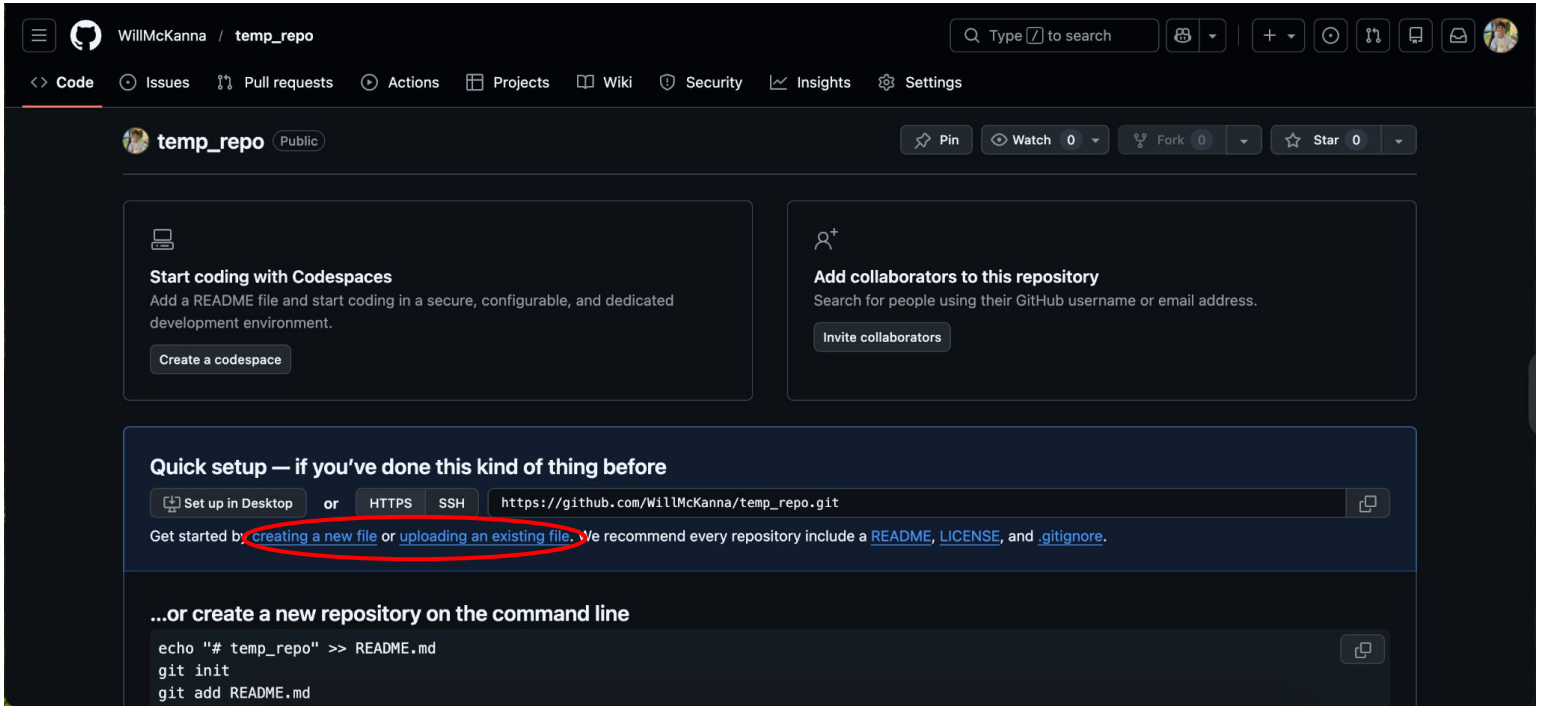
How to Create a GitHub Repository (yay for version control!)

2. Name your repository something memorable, then click “Create Repository”



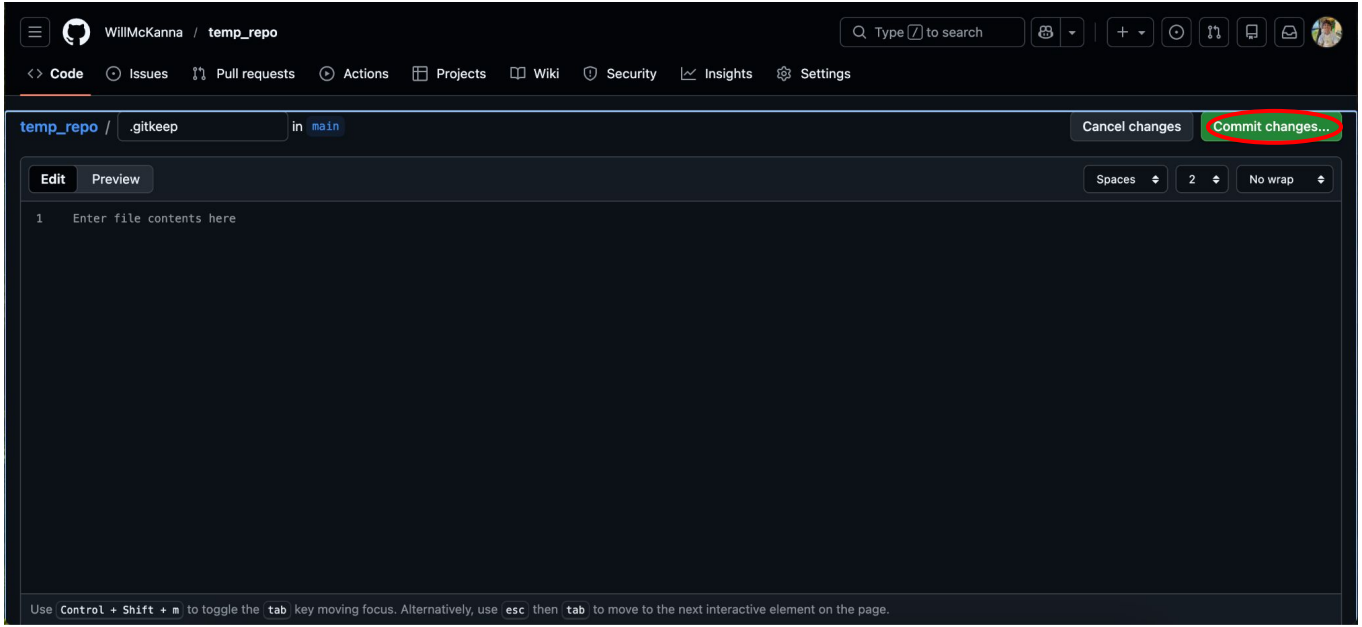
How to Create a GitHub Repository (yay for version control!)

3. Then, populate your repo either with your files, OR click “creating a new file”



How to Create a GitHub Repository (yay for version control!)

3(b). If you click “create a new file,” entitle it “.gitkeep” and click “Commit changes”



How to Create a GitHub Repository (yay for version control!)

4. Your repo is made! Continue adding to it by clicking the “Add file” dropdown menu