Expanding AI/ML Coursework on Your Campus Using Jupyter Notebooks powered by NRP

Building a Lesson Plan for AI/ML Coursework

J. Alex Hurt
University of Missouri

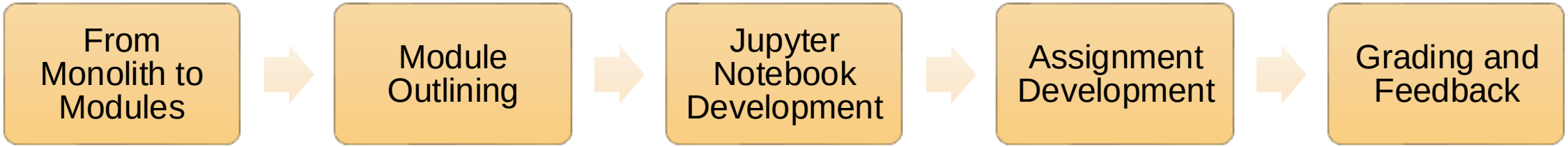
The Power of Jupyter

- Jupyter Notebooks are exceptionally powerful tools for AI/ML Coursework
- So much of AI/ML is hands-on, practical experience, so the capability of interactive learning offered by Jupyter on NRP is crucial
  - NRP Jupyter can even link in NVIDIA GPUs for teaching PyTorch / Tensorflow with real GPU acceleration
- Transitioning from traditional, lecture-based coursework to interactive, hands-on coursework can be time consuming and difficult, but it doesn't have to be!
- AI/ML Courses Transitioned to Jupyter on NRP at MU:
  - Introduction to Machine Learning and Pattern Recognition
  - Supervised Machine Learning
  - Computer Vision
  - Unsupervised Machine Learning
  - Cloud Computing

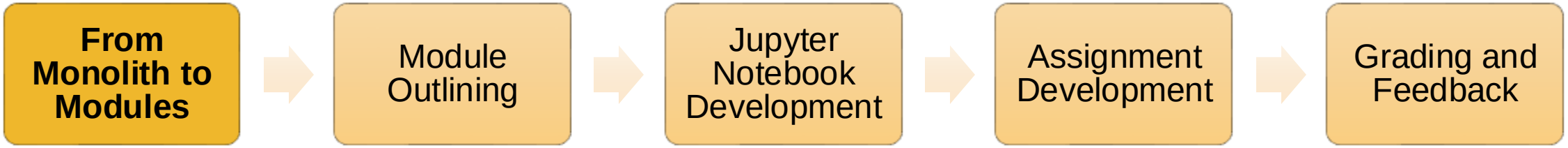
From Syllabus to Jupyter

- This presentation will walk through one of many possible workflows to go from a traditional course syllabus to a Jupyter-based lesson plan
- Steps:
    1. From Monolith to Modules
    2. Module Outlining
    3. Jupyter Notebook Development
    4. Assignment Development
    5. Grading and Feedback
- The next portion of the tutorial will discuss steps 2 and 3 in much more detail

From Syllabus to Jupyter

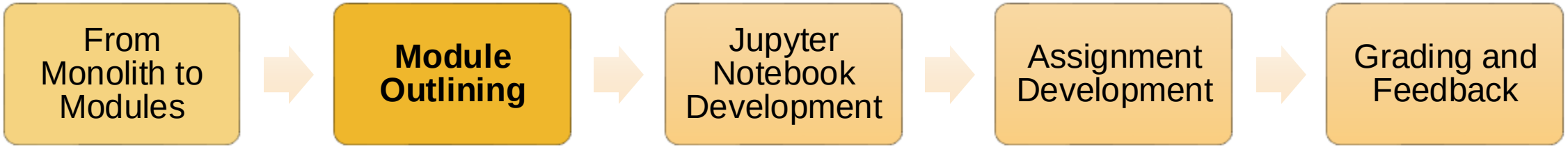


Step 1: From Monolith to Modules



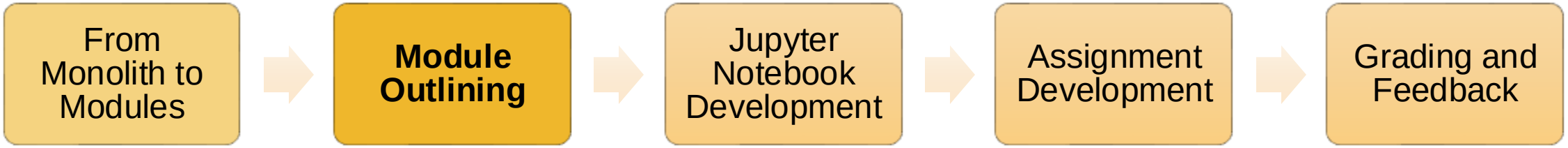
- The first step towards Jupyter-based coursework is to break the course down into a set of **modules**
- This is often easily or already done for many courses, such as courses derived from textbooks, as chapters of a textbook translate into Modules easily!
- Example: Introduction to Machine Learning
  - Module 1: Types of Machine Learning (Supervised vs. Unsupervised)
  - Module 2: Basic Statistical Models (Linear and Logistic Regression)
  - Module 3: Linear Models (Perceptron, MLP, SGD)
  - Module 4: Feature Extraction and Dimensionality (PCA, LDA, Multi-Step Pipelines)
  - Module 5: Classification Models (Decision Tree, KNN, SVM)

Step 2: Module Outlining



- Build a Word-document of the outlines of your modules.
  - What are the key **concepts** to be covered?
  - What types of **activities** would help the students learn the topics of this module?
- Example: Introduction to Machine Learning
  - Module 5: Classification Algorithms
    - K-Nearest Neighbors
    - Decision Trees
    - Support Vector Machines
    - Model Underfitting and Overfitting
    - Scoring Classification Algorithms

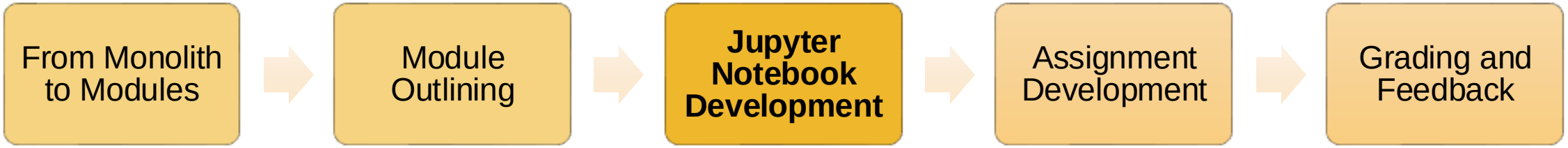
Step 2: Module Outlining



- Module 5: Classification Algorithms
  - K-Nearest Neighbors
  - Decision Trees
  - Support Vector Machines
  - Model Underfitting and Overfitting
  - Scoring Classification Algorithms

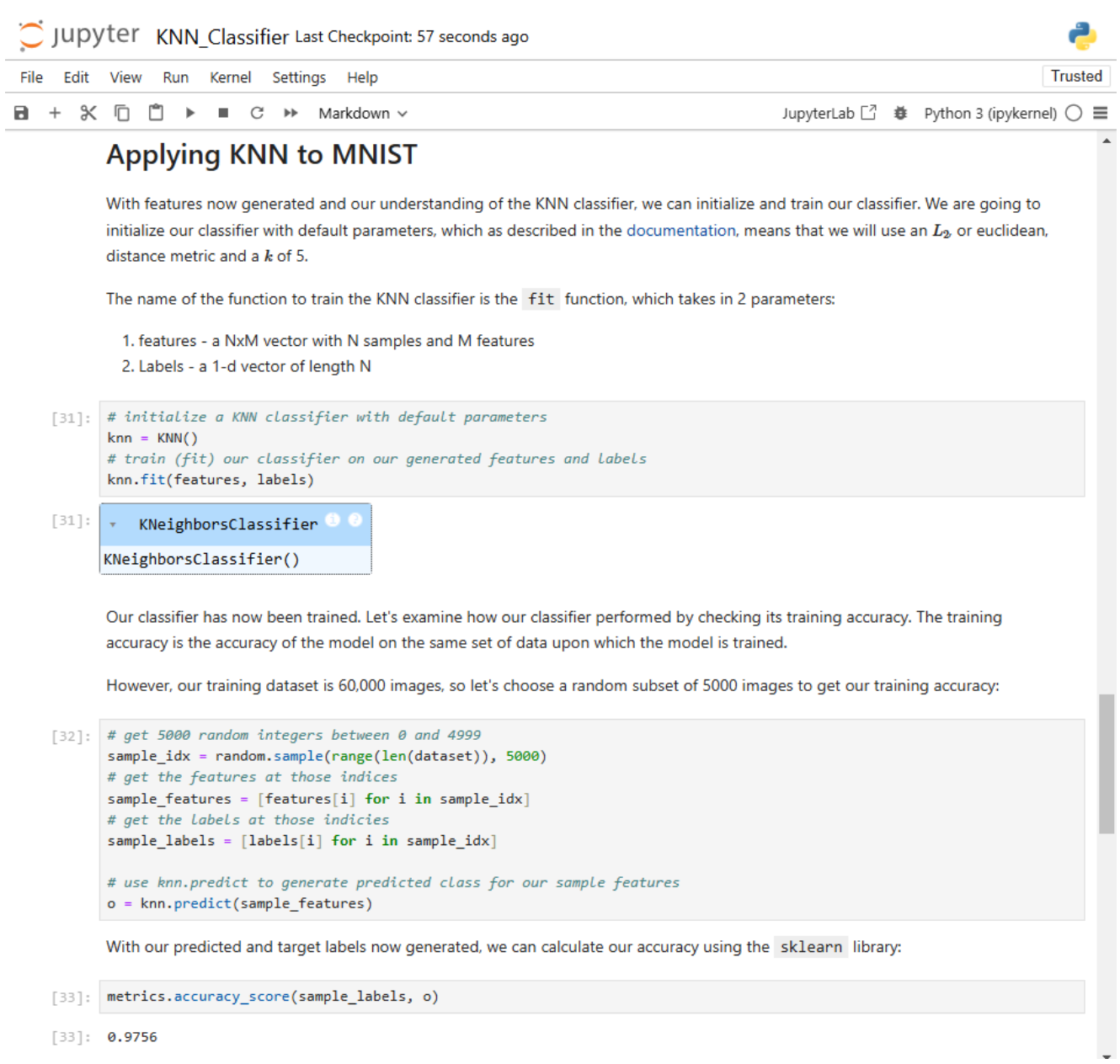
Each major concept or idea from your module becomes its own ***Jupyter Notebook***

Step 3: Jupyter Notebook Development

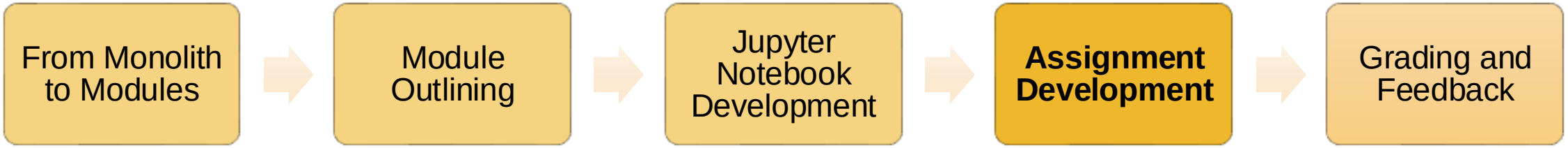


- For each major topic / concept in a given module, build a Jupyter Notebook
  - Ability to embed Markdown into Notebooks allows for guided walkthroughs of course material
  - Link to lecture slides as needed
  - Link to external resources as needed
- Course notebooks can be given asynchronously or notebooks can be used as lecture material
  - My general philosophy: Provide notebooks to students **AND** walk through them during lecture to allow for detailed coverage and allow for questions
- Encouraging participation
  - You can have students submit the walkthrough notebooks through your LMS (Canvas, Blackbord,etc) as a type of Participation grade

Step 3: Jupyter Notebook Development

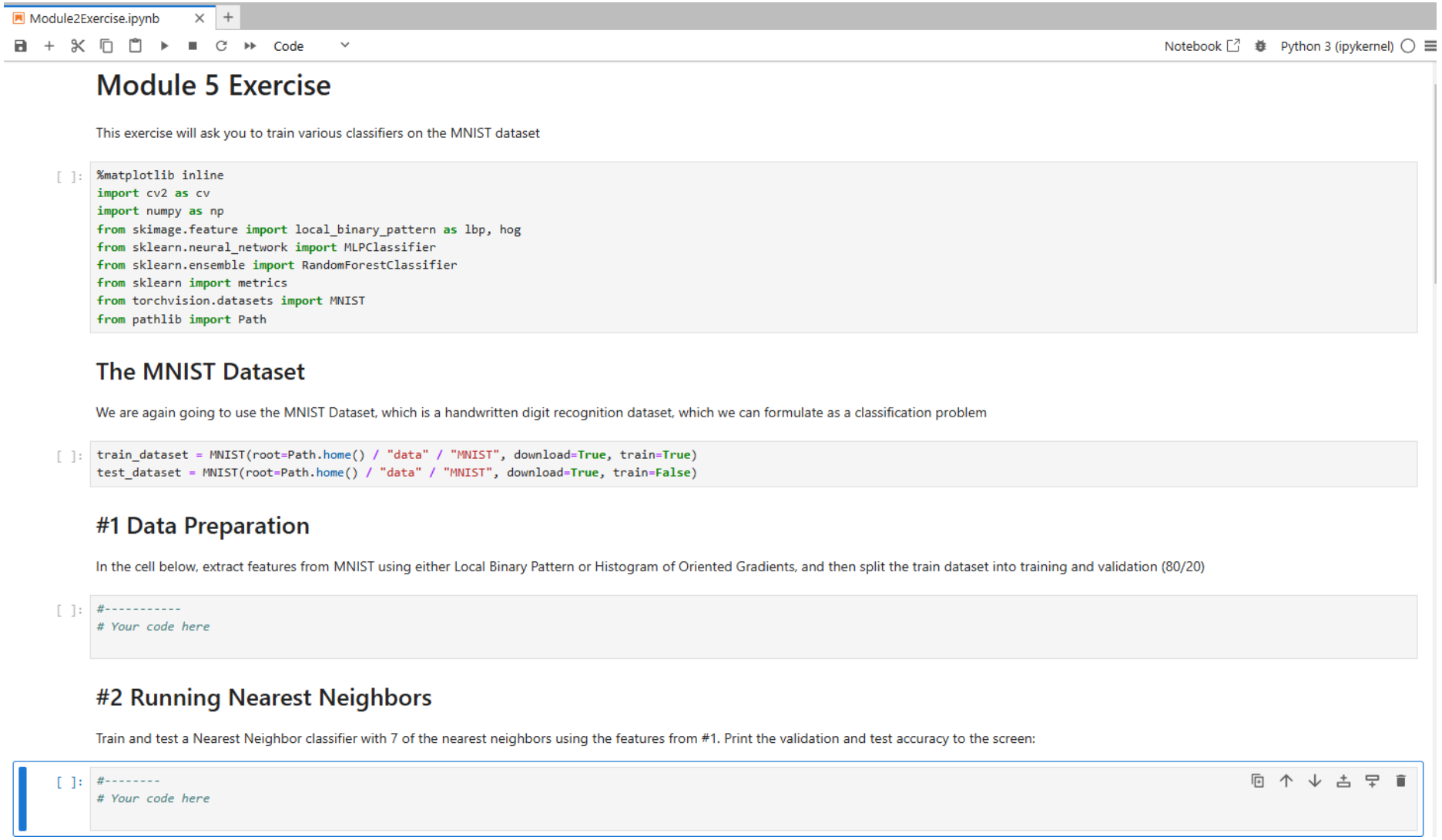


Step 4: Assignment Development

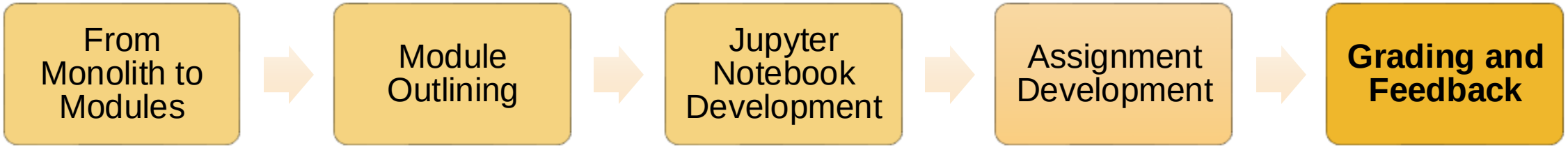


- Assignments for homework / projects can be done via empty Jupyter Notebooks
  - Notebook contains prompts and space for students to work out the problem
  - Be sure to number each cell that students are to fill in for grading purposes
- If distributed with the course notebooks, assignments can even **link to** the course notebooks so students know where to look for more information on topics and ideas
- For submission to LMS systems, have students export notebooks as HTML
  - Most LMS systems can read and display HTML, and HTML maintains the formatting and output of Jupyter notebooks
- For projects that span multiple modules, students can develop Python functions / classes during each module, and then export notebooks as Python scripts and glue them together in Final Projects

Step 4: Assignment Development



Step 5: Grading and Feedback



- Using the grading system built-in to the LMS, you can provide feedback on each question as long as you have numbered your exercises in the Assignment
  - Leaving comments that reference what was done incorrectly on a given question

- After submission and grading, you can release a version of the assignment that is completely solved as a study-guide for future assignments and to ensure all students have a version that is fully functional
  - Personal Opinion: Providing a Solutions notebook for each assignment has helped students grasp concepts better as they see how to go from nothing to a full solution

Step 5: Grading and Feedback

#1 Data Preparation

In the cell below, extract features from MNIST using either Local Binary Pattern or Histogram of Oriented Gradients, and then split the train dataset into training and validation (80/20)

[7]:
```
#-----------
# Your code here

X_train = np.empty([len(train_dataset), 81])
y_train = np.empty(len(train_dataset))
X_test = np.empty([len(test_dataset), 81])
y_test = np.empty(len(test_dataset))

for i, (img, label) in enumerate(train_dataset):
    X_train[i] = hog(img)
    y_train[i] = label

for i, (img, label) in enumerate(test_dataset):
    X_test[i] = hog(img)
    y_test[i] = label

X_subtrain, X_val, y_subtrain, y_val = train_test_split(X_train, y_train, test_size=0.2)
```

#2 Running Nearest Neighbors

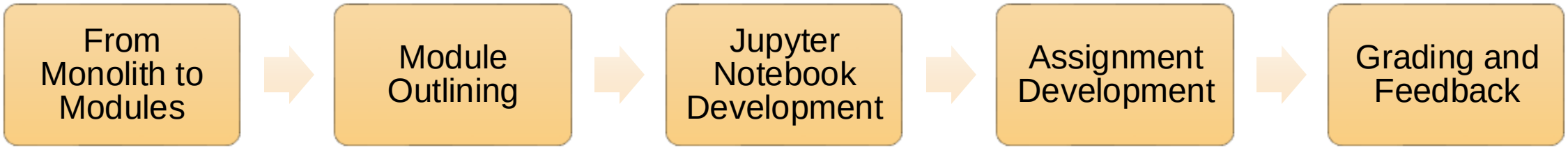
Train and test a Nearest Neighbor classifier with 7 of the nearest neighbors using the features from #1. Print the validation and test accuracy to the screen:

[8]:
```
#--------
# Your code here

model = KNeighborsClassifier(7)
model.fit(X_subtrain, y_subtrain)

print(model.score(X_val, y_val))
print(model.score(X_test, y_test))
```

```
0.96225
0.9574
```

Thank you!

Questions?



Resources

- NRP Homepage: https://nationalresearchplatform.org/
- NRP Documentation: https://ucsd-prp.gitlab.io/
- MU Resources for AI/ML Researchers using NRP: https://github.com/MUAMLL/nautilus
  - Sample Dockerfiles, YAMLs, and Wiki
  - Deploying JupyterHub on NRP: https://github.com/MUAMLL/nautilus/wiki/JupyterHub
- GP-ENGINE Tutorial Resources: https://github.com/MUAMLL/gp-engine-tutorials
  - Step-by-step for getting started using NRP including account creation, PVC setup, and basic pod and job running
- Matrix Chat: https://nationalresearchplatform.org/updates/matrix-chat-for-nautilus-users/
  - Chat-service (similar to slack) for discussion and help using NRP