



Machine Learning



DR. BRIDGETTE TOWNSEND

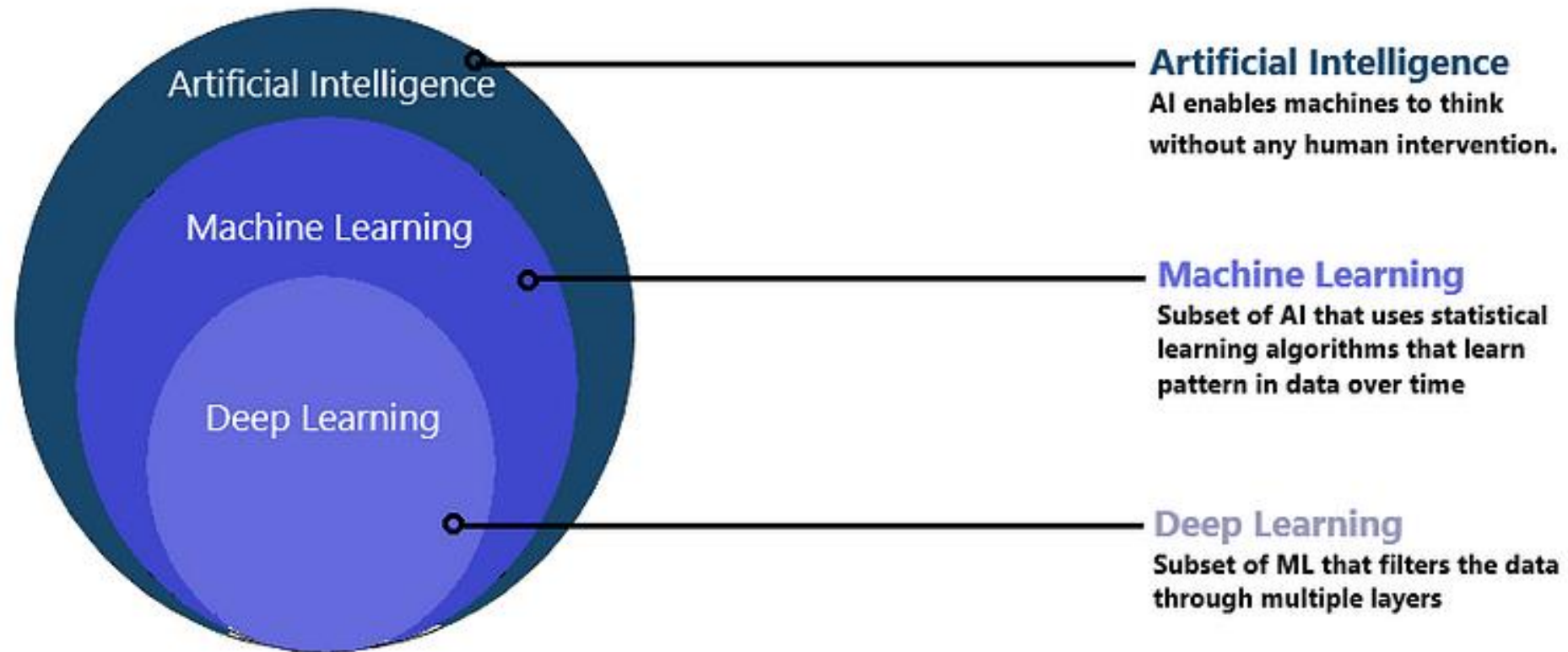
AUG – OCT 2023

8-WEEK FALL COURSE

Week 1:

Introduction to Machine Learning

M/L is a subfield of AI

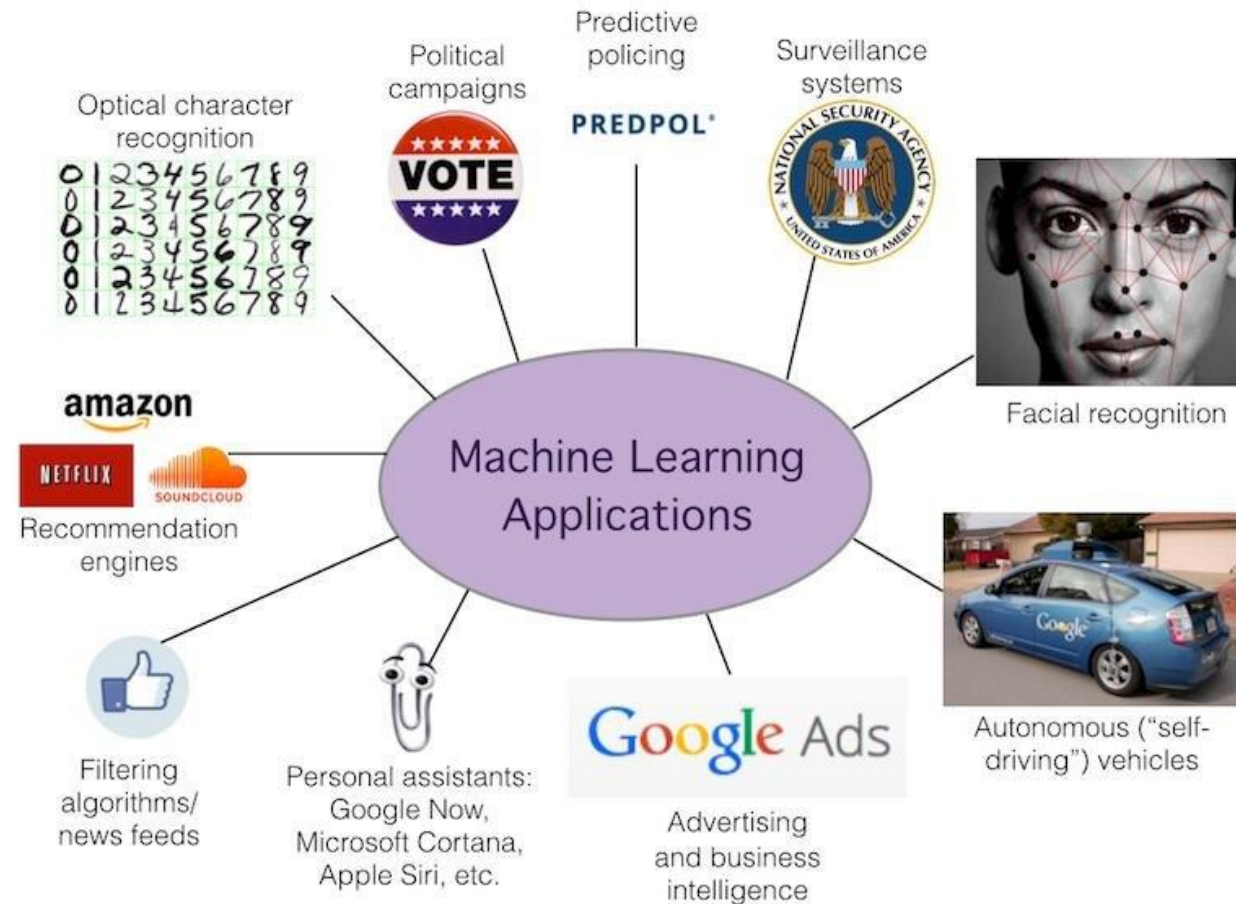


Know your task, Know your data

Machine Learning Process



Types of M/L Applications



Common M/L Algorithms

SVM: Support Vector Machine

LR: Logistic Regression

DT: Decision Tree

RF: Random Forest

GNB: Gaussian Naïve Bayes

MNB: Multinomial Navie Bayes

CNB: Complement Naïve Bayes

KNN: K-Nearest Neighbor

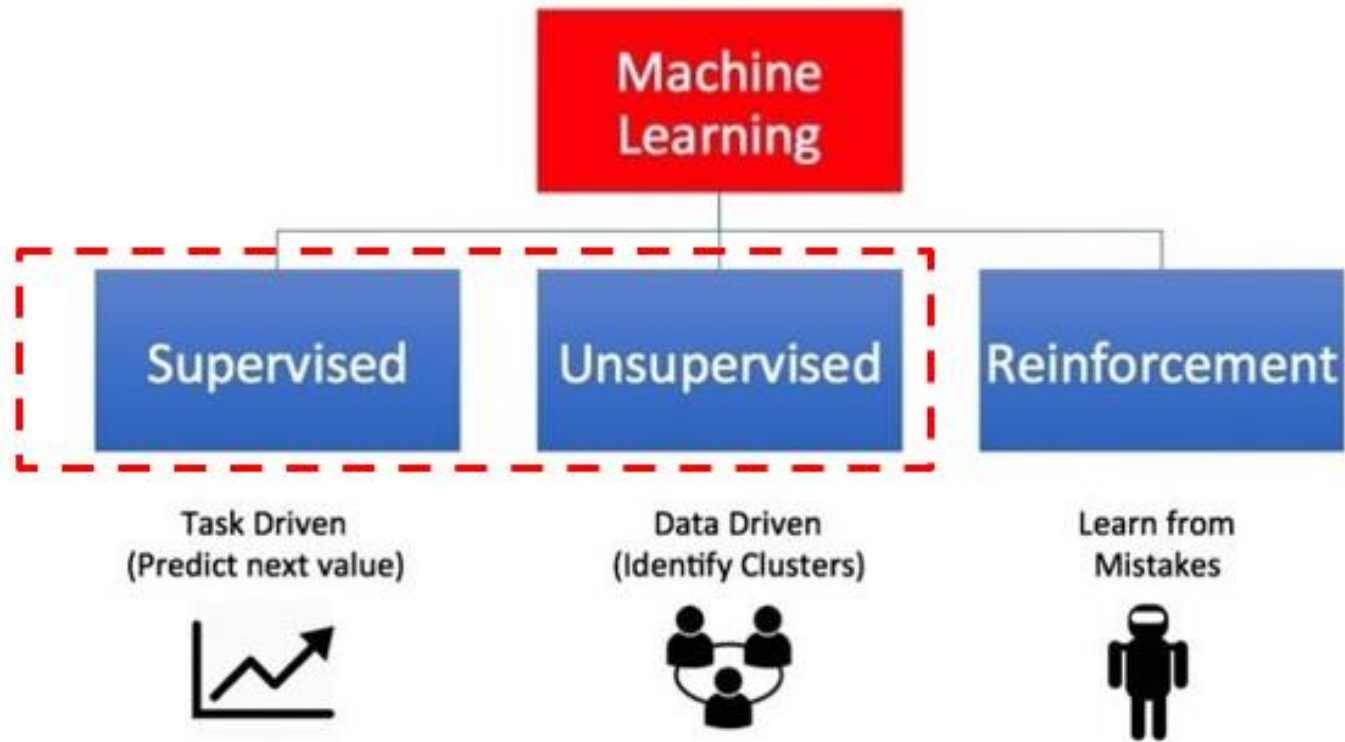
PTN: Perceptron

Questions to ask when looking for a M/L solution

- What question(s) am I trying to answer? Do I think the data collected can answer that question?
- What is the best way to phrase my question(s) as a M/L problem?
- Have I collected enough data to represent the problem I want to solve?
- What features of the data did I extract and will these enable the right predictions?
- How will I measure success in my application?
- How will the M/L solution interact with other parts of my research or product?

Types of M/L Algorithms

Types of Machine Learning



Supervised Learning

Labels

- Serve as the correct (target) answers needed by the model/algorithm to learn to predict

Divided into two main types

- Classification
 - Spam, not spam
 - Yes, no
- Regression
 - Housing prices
 - Income based on education, age, or where they live

Unsupervised Learning

No Labels

Focuses on discovering patterns, structures, and relationships within the data itself.

Divided into two main types

- Clustering
 - Splitting customers into groups for targeted marketing
- Dimensionality Reduction
 - Feature extraction
 - Simplify the data without losing too much information
 - car's mileage may be very correlated with its age, so the dimensionality reduction algorithm will merge them into one feature that represents the car's wear and tear.

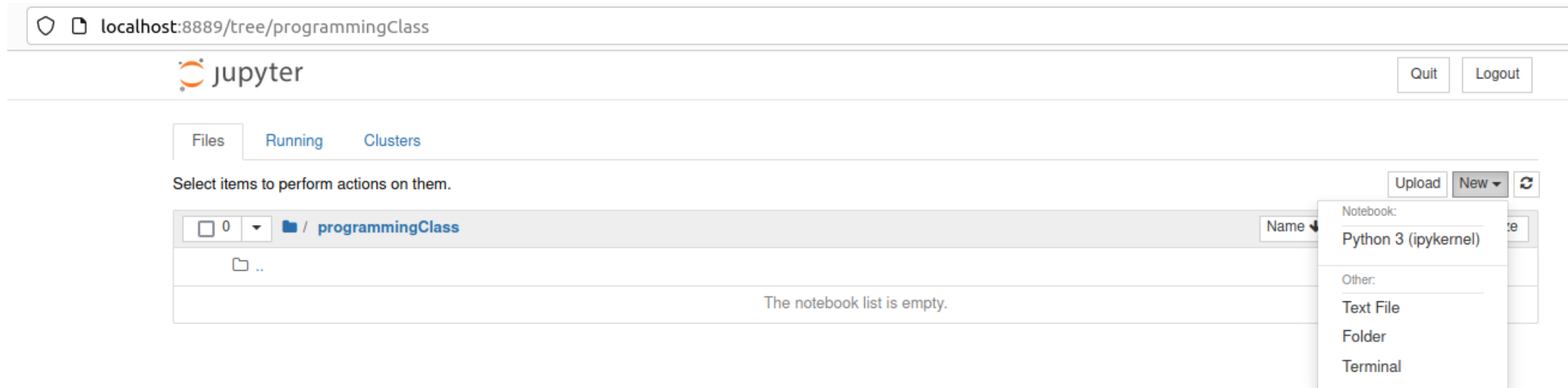
Python and scikit-learn

- Python is a general-purpose programming language
 - Not only is it simple and easy to learn, it has a large library of machine learning libraries.
- Scikit-learn is a machine learning library for Python.
 - A wide range of machine learning algorithms, including classification, regression, clustering, and dimensionality reduction.
 - Well-documented API which makes it easy to find information on how to use it
 - Leverage the datasets bundled within the library

Jupyter Notebook Tutorials and Guidance

- Course Resources have installation guides
- Development environment used in this class

Starting a notebook



Opening Notebook

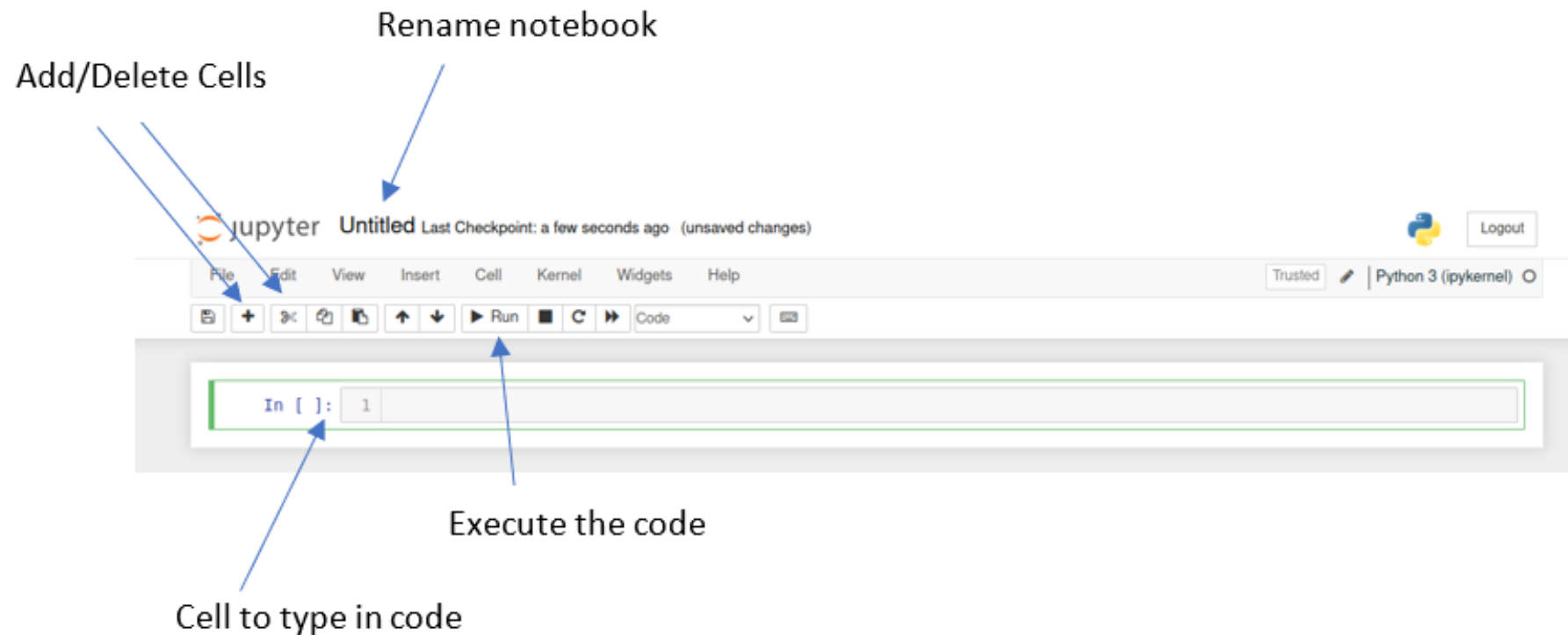
 Quit Logout

Files Running Clusters

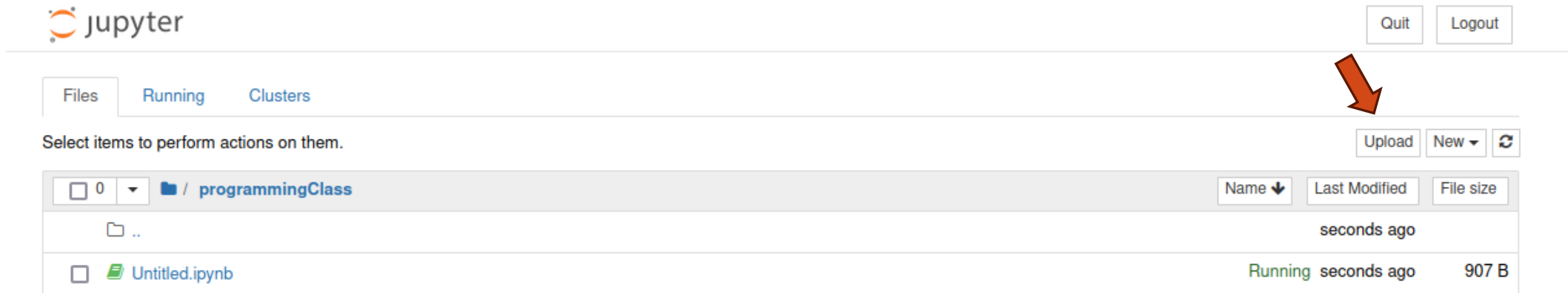
Select items to perform actions on them. Upload New ▾ ↻

<input type="checkbox"/> 0 ▾	📁 / programmingClass	Name ▾	Last Modified	File size
	📁 ..		seconds ago	
<input type="checkbox"/>	📄 Untitled.ipynb		Running seconds ago	907 B

Rename the notebook



Upload Files



The image shows the JupyterLab web interface. At the top left is the Jupyter logo. To the right are 'Quit' and 'Logout' buttons. Below these are tabs for 'Files', 'Running', and 'Clusters'. The 'Files' tab is active, displaying the text 'Select items to perform actions on them.' Below this is a file browser table for the directory '/ programmingClass'. The table has columns for 'Name', 'Last Modified', and 'File size'. It lists a '..' directory and a file named 'Untitled.ipynb' which is in a 'Running' state. An 'Upload' button is located above the table, and a red arrow points to it from the right side of the interface.

jupyter

Quit Logout

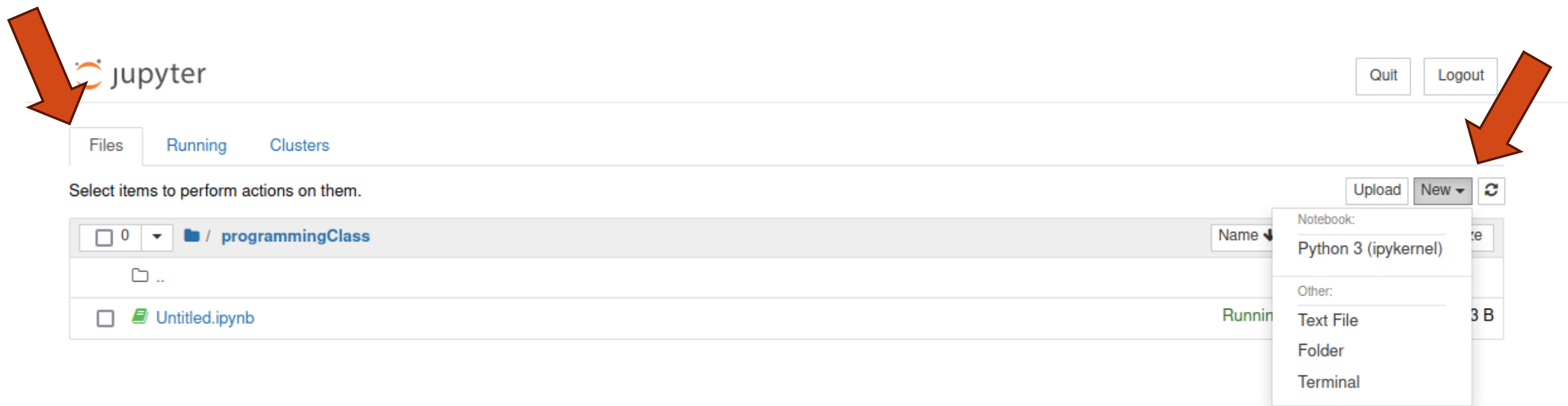
Files Running Clusters

Select items to perform actions on them.

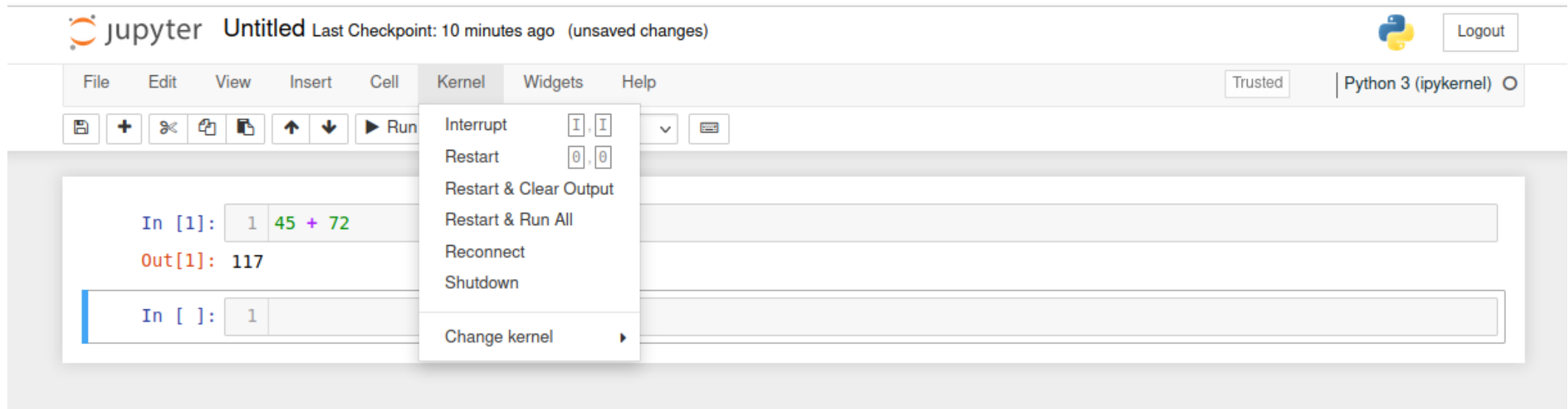
Upload New ↕

	Name ↓	Last Modified	File size
<input type="checkbox"/> 0	..	seconds ago	
<input type="checkbox"/>	Untitled.ipynb	Running seconds ago	907 B

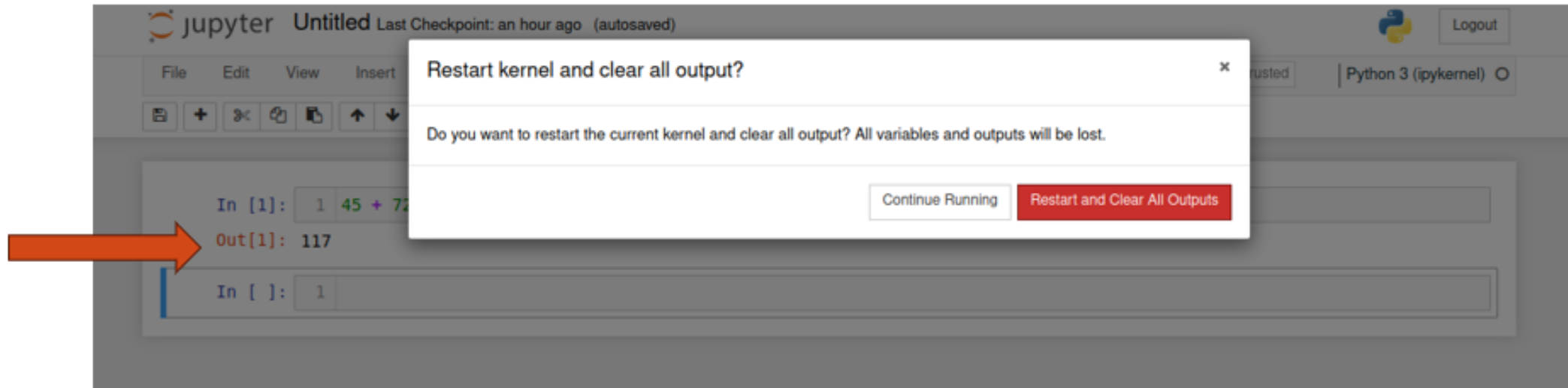
Create Folders/Create New Notebook/Launch Terminal



Reset/Return the state of the notebook



Restart and Clear all output



Missing a module Error

“ModuleNotFoundError” for scipy

```
In [7]: 1 import scipy as sp
        2 print("scipy version: {}".format(sp.__version__)) # __ are two underscores

-----
ModuleNotFoundError                                Traceback (most recent call last)
Cell In[7], line 1
----> 1 import scipy as sp
      2 print("scipy version: {}".format(sp.__version__)) # __ are two underscores

ModuleNotFoundError: No module named 'scipy'
```

Missing a module Error

“`ModuleNotFoundError`” for sklearn

```
In [10]: 1 import sklearn
          2 print("scikit-learn version: {}".format(sklearn.__version__)) # __ are two underscores

-----
ModuleNotFoundError                                Traceback (most recent call last)
Cell In[10], line 1
----> 1 import sklearn
      2 print("scikit-learn version: {}".format(sklearn.__version__)) # __ are two underscores

ModuleNotFoundError: No module named 'sklearn'
```

sklearn is deprecated scikit-learn

```
baldwin@ubuntu:~/jupy$ pip install sklearn
Collecting sklearn
  Downloading sklearn-0.0.post7.tar.gz (3.6 kB)
  Preparing metadata (setup.py) ... error
error: subprocess-exited-with-error

  × python setup.py egg_info did not run successfully.
  | exit code: 1
  ╰─> [18 lines of output]
    The 'sklearn' PyPI package is deprecated, use 'scikit-learn'
    rather than 'sklearn' for pip commands.

    Here is how to fix this error in the main use cases:
    - use 'pip install scikit-learn' rather than 'pip install sklearn'
    - replace 'sklearn' by 'scikit-learn' in your pip requirements files
      (requirements.txt, setup.py, setup.cfg, Pipfile, etc ...)
    - if the 'sklearn' package is used by one of your dependencies,
      it would be great if you take some time to track which package uses
      'sklearn' instead of 'scikit-learn' and report it to their issue tracker
    - as a last resort, set the environment variable
      SKLEARN_ALLOW_DEPRECATED_SKLEARN_PACKAGE_INSTALL=True to avoid this error

    More information is available at
    https://github.com/scikit-learn/sklearn-pypi-package

    If the previous advice does not cover your use case, feel free to report it at
    https://github.com/scikit-learn/sklearn-pypi-package/issues/new
    [end of output]

note: This error originates from a subprocess, and is likely not a problem with pip.
error: metadata-generation-failed

  × Encountered error while generating package metadata.
  ╰─> See above for output.

note: This is an issue with the package mentioned above, not pip.
hint: See above for details.
```

Upgrade options

```
baldwin@ubuntu:~/jupy$ pip install scipy
Collecting scipy
  Downloading scipy-1.11.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (36.3 MB)
    36.3/36.3 MB 9.9 MB/s eta 0:00:00
Requirement already satisfied: numpy<1.28.0,>=1.21.6 in ./jup_notebook/lib/python3.10/site-packages (from scipy) (1.24.2)
Installing collected packages: scipy
Successfully installed scipy-1.11.2

[notice] A new release of pip is available: 23.0.1 -> 23.2.1
[notice] To update, run: pip install --upgrade pip
baldwin@ubuntu:~/jupy$ pip install --upgrade pip
Requirement already satisfied: pip in ./jup_notebook/lib/python3.10/site-packages (23.0.1)
Collecting pip
  Using cached pip-23.2.1-py3-none-any.whl (2.1 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 23.0.1
    Uninstalling pip-23.0.1:
      Successfully uninstalled pip-23.0.1
  Successfully installed pip-23.2.1
baldwin@ubuntu:~/jupy$
```

How to install packages

From the terminal command line type **pip install <name of package>**

import pandas

pandas is the “name of package to install”

Example:

- **pip install pandas**

Homework #1

- Setup of Jupyter Notebook, install necessary python packages, and confirm by providing versions of the installed libraries (listed in homework handout)
- Homework handout is in the Files->Homework folder in canvas

Week 2:

Classification with k-Nearest Neighbors and
the Iris Dataset Part 1

Revisit

Last week we started with a brief introduction to M/L

We covered the distinction between of Supervised and Unsupervised M/L Learning

Discussed the differences between Artificial Intelligence (A/I), Machine Learning (M/L) and Deep Learning (D/L)

Discussed the tools we will use in the course, setting up the Python environment using Jupyter Notebook and how to install the required Python packages

Had our first homework assignment to confirm the successful setup of the development environment and printed out the versions of each of the Python Packages installed.

M/L Types/Categories Revisited

Different Machine Learning Types	Conceptual Examples	Subfields
Supervised Learning	<ul style="list-style-type: none">➤ Labeled Data➤ Direct Feedback➤ Predict outcome/failure	<ul style="list-style-type: none">➤ Classification➤ Regression
Unsupervised Learning	<ul style="list-style-type: none">➤ No Labels➤ No Feedback➤ Find hidden structures/patterns in data	<ul style="list-style-type: none">➤ Clustering➤ Dimensionality Reduction
* Reinforcement Learning (beyond the scope of this course)	<ul style="list-style-type: none">➤ Decision process➤ Reward systems➤ Learn series of actions	

Key M/L Terminology

Labels

- A label is the thing we are predicting.
 - The y variable in a simple linear regression
 - The label could be the future price of wheat, an animal in a picture, a meaning of an audio clip, or just about anything

Features

- A feature is an input variable
 - The x variable in a simple linear regression
 - In a spam detector example, the features could include:
 - Word in the email text
 - Sender's address
 - Time of day the email was sent
 - A phrase in the email "one weird trick"

Key M/L Terminology (2)

Examples

- An example is a particular instance of data x .
 - We can break the examples into 2 categories
 - labeled examples
 - unlabeled examples
- A labeled example includes both feature(s) and the label
 - Labeled examples: {features, label}: (x,y)
 - Labeled examples are used to train the model.
 - In the spam example, the labeled examples would be the individual emails, users could mark them as either: spam/not spam

Housing Median Age (feature)	Total Rooms (feature)	Total Bedrooms (feature)	Median House Value (label)
15	5612	1283	66900
19	7650	1901	80100
17	720	174	85700

Key M/L Terminology (3)

- An unlabeled example contains features but not labels
 - unlabeled examples: {features, ?}: (x,?)
 - Once we have trained the model with labeled examples, we use that model to predict the label on unlabeled examples
 - In the spam example, unlabeled examples are new emails that human have not labeled, spam/not spam

Housing Median Age (feature)	Total Rooms (feature)	Total Bedrooms (feature)
15	5612	1283
19	7650	1901
17	720	174

Key M/L Terminology (4)

Models

- A model defines the relationship between features and label.
 - Example, a spam detection model might associate certain features with 'spam'
- Two phases of a model:
 - Training means creating/learning the model.
 - Show the model labeled examples and the model will gradually learn the relationship between features and labels
 - Inference means applying the trained model to unlabeled examples
 - Use the trained model to make useful predictions (y')
 - Example, during inference, you can predict the median house value for a new unlabeled example

Classification vs Regression

- A Classification model predicts discrete values
 - Is a given email message spam or not spam?
 - Is this an image of a dog, a cat, or a hamster?
- A Regression model predicts continuous values
 - What is the value of the house in California?
 - Housing prices
 - What is the probability that a user will click on this ad?
 - Income based on education, age, or where they live

Classifying Iris Species

Iris Dataset

3 different species (Setosa, Versicolor, and Virginica)

Measurements associated with each iris flower

- Length and width of the petals
- Length and width of the sepals
- All measurements in centimeters

Classification Type of Problem (Supervised Learning)

- Build a M/L model that can learn from the measurements of the Iris Species in order to make predictions for a new Iris

Features and Labels of the Iris Dataset

3 labels and 4 features

Iris Flowers (label)	Sepal Length (feature)	Sepal width (feature)	Petal Length (feature)	Petal Width (feature)
Setosa	5.1	3.5	1.4	0.2
Virginica	4.9	3.0	1.4	0.2
Versicolor	4.6	3.2	1.3	0.2

Week 3:

mid-term week

Homework Review: Iris Species

Load the necessary libraries and Iris Dataset

```
In [1]: 1 # add comment
        2 import pandas as pd
        3
        4 from sklearn.datasets import load_iris
        5 iris_dataset = load_iris()
        6
        7 # nothing will be printed to output but you must run this cell
```

Print data and attributes available in the dataset

The formatted string (.format()) method is used to replace {} the result of iris_dataset.keys(), which is a list-like object containing the keys (dict_keys) of the dataset.

```
In [2]: 1 # add comment
        2 print("Keys of iris_dataset: \n{}".format(iris_dataset.keys()))
```

```
Keys of iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

Print out specific information

In Python, dictionaries are used to store key-value pairs, where keys are unique identifiers, and values are associated data or objects

What are the target names (Labels) and features.

Structured datasets are organized as tables with rows and columns. Each row represents a single data point or observation, while each column represents a feature or a label.

```
In [4]: 1 # comment
        2 print("Target name: {}".format(iris_dataset['target_names']))
```

```
Target name: ['setosa' 'versicolor' 'virginica']
```

```
In [5]: 1 # comment explain features
        2 print("Feature names: \n{}".format(iris_dataset['feature_names']))
```

```
Feature names:
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

Understand how the data is stored

‘type’

Tells you the data is stored in a NumPy array.

```
In [6]: 1 # comment explain "type"  
        2 print("Type of data: {}".format(type(iris_dataset['data'])))
```

```
Type of data: <class 'numpy.ndarray'>
```

Provide the number of rows (samples) and columns (features) in the dataset

`.shape`

An attribute of arrays and matrices in libraries like NumPy and Pandas. Provides the dimensions the number of rows and columns.

Each row represents a sample (i.e.. individual iris flower), and each column represents a different feature (i.e., sepal length, sepal width, petal length, petal width).

```
In [7]: 1 # comment explain "shape"
        2 print("Shape of data: {}".format(iris_dataset['data'].shape))
```

```
Shape of data: (150, 4)
```


Print the description to understand the dataset

```
In [3]: 1 # comment  
        2 print(iris_dataset['DESCR'])
```

```
.._iris_dataset:  
  
Iris plants dataset  
-----  
  
**Data Set Characteristics:**  
  
:Number of Instances: 150 (50 in each of three classes)  
:Number of Attributes: 4 numeric, predictive attributes and the class  
:Attribute Information:  
  - sepal length in cm  
  - sepal width in cm  
  - petal length in cm  
  - petal width in cm  
  - class:  
    - Iris-Setosa  
    - Iris-Versicolour  
    - Iris-Virginica
```

Specify how many columns to display

`[:5]`

In Python, slicing is used to extract a portion of a sequence or array.

Indicates you want only the first five columns of the data.

```
In [8]: 1 # comment
        2 print("First 5 columns of data: \n{}".format(iris_dataset['data'][:5]))
```

First 5 columns of data:

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
```

Rename and list the first few rows

Improving the readability of dataset as well as make it easier when analyzing the data

Renamed the columns and list first 5 rows (samples)

- typically used when you want to give more descriptive names to the columns of your DataFrame.

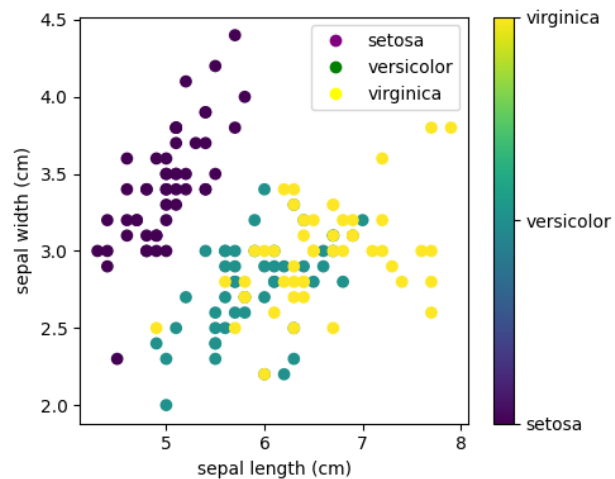
```
In [10]: 1 # add comment that explain why this was done
          2 iris_df.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
          3 iris_df.head()
          4
```

Out[10]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Visualize the data

```
In [15]: 1 # Two feature scatter plot of the iris dataset using matplotlib
2
3 plt.figure(figsize=(5,4))
4 plt.scatter(iris_dataset.data[:, X], iris_dataset.data[:,y], c=iris_dataset.target)
5 plt.colorbar(ticks=[0,1,2], format=formatter)
6 plt.xlabel(iris_dataset.feature_names[X])
7 plt.ylabel(iris_dataset.feature_names[y])
8
9
10 class_labels = iris_dataset.target_names
11 legend_aliases = [plt.Line2D([0,0],[0,0], color=['purple','green','yellow'][i], marker='o', linestyle='', label=l
12                     for i, label in enumerate(class_labels))
13
14
15 plt.legend(handles=legend_aliases)
16
17
18 plt.tight_layout()
19 plt.show()
```



Comments were provided in cells not covered.

Homework

No homework....it's mid-term week....go be great and study hard this week.

Next Class

THE 2ND PART OF WORKING WITH K-NEAREST NEIGHBORS
ALGORITHM

Week 4:

Classification with k-Nearest Neighbors and
the Iris Flowers Dataset Part 2

Revisit

Understanding Supervised Learning

- Supervised learning is a type of machine learning where the algorithm learns from labeled training data.
- The training data includes input features (independent variables) and corresponding labels (dependent variable or target).
- The goal is to learn a mapping from inputs to outputs so that the algorithm can make accurate predictions on new, unseen data.

Features and Labels:

- Features are the input variables used to make predictions.
- Labels are the target variables or outcomes that the model aims to predict.
- In the Iris dataset, the features are the measurements (sepal length, sepal width, petal length, petal width), and the labels are the species of iris flowers.

Explored the Iris Dataset

Identified the features (sepal length, sepal width, petal length, petal width) and target (flower species) in the dataset

Understand basic data exploration techniques

Understand how to visualize the data (i.e., scatter plots, histograms, bar charts)

Scikit-Learn

How to load datasets using Scikit-Learn (**DONE**)

Familiarity with the structure of the Iris dataset (**DONE**)

Splitting data into training and testing sets (**Up next**)

Using Scikit-Learn's preprocessing tools (**Up next**)

Training models from Scikit-Learn's various algorithms (**Up next**)

Evaluating model performance using metrics (i.e., accuracy) (**Up next**)

Model Training, Testing, and Evaluation

Training involves using a labeled dataset to teach the model.

- Learn what the relationship is between features and labels.

Test the model on a separate dataset (testing set) to evaluate its performance on unseen data.

- After training, test.

Evaluate the model.

- Measure its accuracy (i.e., confusion matrix, score)
- After Training and Testing, evaluate

Run multiple models to compare their performance against each other

- Best Practice
- Model Selection:
 - No such thing as a one-size-fits-all machine learning model that works best for every problem.
 - Different algorithms have different strengths and weaknesses.
 - By running multiple models, you can explore which model type is most suitable for your specific problem.
- Hyperparameter Tuning:
 - Models often have hyperparameters that need to be tuned for optimal performance.
 - Running multiple models with different hyperparameter settings helps you find the best configuration.

Steps to Building a Classification Model

Choose an appropriate classification algorithm (e.g., **k-nearest neighbors**, decision trees)

Preprocess the data (scaling, splitting)

Train the model

Make predictions on test data

Evaluate the model's accuracy

Splitting the Data for Training and Testing

Splitting data into training and testing sets is a crucial step in machine learning.

It allows you to evaluate the performance of your machine learning model on unseen data.

Steps to split the data into training and testing sets.

- Use the **train_test_split** function to split the data into training and testing sets.
 - By default, `train_test_split` reserves 75% of the data for training and 25% for testing
- Specify the features (X) and labels (y) as arguments, along with the desired test size (the proportion of the dataset to include in the test split).
 - To specify different splits, you can set the sizes of the testing and training sets with the `train_test_split` function's keyword arguments **test_size** and **train_size**

Experimentation and Interpretation

Trying different algorithms and parameters

Analyzing the model's performance and interpreting the results

Understanding the implications of accuracy

- While accuracy is a valuable metric for evaluating classification models, it should not be used in isolation.

You want to have a comprehensive assessment of a model's performance

scikit-learn classification_report

Generates a report with the following information for each class and an additional row for macro and weighted averages (Used to provide a comprehensive assessment):

Precision:

- Measures the accuracy of positive predictions for a given class.
- It is calculated as the ratio of true positives to the sum of true positives and false positives.
- Precision is the ability of the classifier not to label a positive sample as negative.

Recall (Sensitivity or True Positive Rate):

- Measures the ability of the classifier to identify all relevant instances of a class in the dataset.
- It is calculated as the ratio of true positives to the sum of true positives and false negatives.

F1-Score:

- Harmonic mean of precision and recall.
- It provides a balanced measure of a model's performance.
- It is calculated as $2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$.

Support:

- Support represents the number of samples in each class.
- The count of true instances for each class in the dataset.

Macro Avg (Macro Average):

- This row shows the unweighted average of precision, recall, and F1-score across all classes.
- All classes equally, regardless of their size.

Weighted Avg (Weighted Average):

- This row shows the weighted average of precision, recall, and F1-score, where each class's score is weighted by its support. It gives more importance to larger classes.

New Package Alert (Seaborn)

pip install seaborn

```
: 1 import matplotlib.pyplot as plt
2 import seaborn as sns
3 from sklearn.metrics import confusion_matrix
4
5 # Assuming you have a confusion matrix
6 # conf_matrix = confusion_matrix(y_test, y_pred)
7
8 # Define class labels
9 class_labels = iris.target_names
10
11 # Create a heatmap
12 plt.figure(figsize=(8, 6))
13 sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=class_labels, yticklabels=class_labels)
14 plt.xlabel('Predicted Labels')
15 plt.ylabel('Actual Labels')
16 plt.title('Confusion Matrix')
17 plt.show()
```

ModuleNotFoundError Traceback (most recent call last)

Cell In[23], line 2
1 import matplotlib.pyplot as plt
----> 2 import seaborn as sns
3 from sklearn.metrics import confusion_matrix
5 # Assuming you have a confusion matrix
6 # conf_matrix = confusion_matrix(y_test, y_pred)
7
8 # Define class labels

ModuleNotFoundError: No module named 'seaborn'

Confusion Matrix

A table that is often used to describe the performance of a classification model (classifier) on a set of test data for which the true values are known.

Consists of four main components:

True Positives (TP):

- True Positives are the cases where the model correctly predicted the positive class (**correctly identified as true**).

True Negatives (TN):

- True Negatives are the cases where the model correctly predicted the negative class (**correctly identified as false**).

False Positives (FP):

- False Positives are the cases where the model predicted the positive class when it was actually negative (**incorrectly identified as true**). These are also known as Type I errors.

False Negatives (FN):

- False Negatives are the cases where the model predicted the negative class when it was actually positive (**incorrectly identified as false**)
- Also known as Type II errors.

Let's look at how all of this is done

Homework #3

Create train and test sample data sets.

Create a k-NN classification model.

Evaluate the accuracy of the k-NN classification model.

Run multiple models to find the best one.

Visualize the confusion matrix using a heatmap.

Next Class

TIME SERIES AND SIMPLE LINEAR REGRESSION

Week 5:

Time Series and Simple Linear Regression

Time Series Data

Time series data is a type of data in which observations, measurements, or data points are collected or recorded sequentially over time at regular or irregular intervals.

Typical Components :

- **Trend:**
 - The trend component represents the long-term movement or behavior in the data. Captures gradual and persistent changes in the time series over time.
 - Example: Monthly sales data for a retail store may exhibit an upward trend if the store's business is growing. Conversely, if sales decrease steadily over time, it indicates a downward trend.
- **Seasonality:**
 - Regular, repeating patterns in the data that occur at fixed intervals, such as daily, weekly, or yearly.
 - Example: Daily temperature data often shows seasonality with higher temperatures during the summer and lower temperatures during the winter.
- **Cyclic Patterns:**
 - Similar to seasonality but have longer and less regular cycles. Fluctuations not tied to fixed calendar intervals.
 - Example: The real estate market may exhibit cyclic patterns with periods of housing booms and busts, which can last several years.
- **Irregular or Residual:**
 - Also known as the residual, represents random variations or noise in the data that cannot be attributed to the trend, seasonality, or cyclic patterns.
 - Unexpected or irregular events.
 - Example: Stock price data can have irregular fluctuations due to market noise, news events, or unexpected corporate announcements.

Modeling and Forecasting With Trend Component

Modeling: Modeling the trend is crucial for capturing the overall direction of the data. Various regression and smoothing techniques can be used to estimate and model the trend component. Linear regression is a common approach.

Forecasting: The trend component provides valuable information for forecasting future values. Make predictions about the data's future behavior.

Modeling and Forecasting With Seasonality Component

Modeling: Seasonality can be modeled using techniques like seasonal decomposition of time series (STL) or seasonal autoregressive integrated moving average (SARIMA) models. Involves estimating seasonal effects and incorporating them into the forecasting model.

Forecasting: Seasonality is critical for forecasting when certain patterns are expected to repeat. Models that account for seasonality can generate more accurate predictions for specific time intervals or periods.

Modeling and Forecasting With Cyclical Component

Modeling: Modeling cyclic patterns can be more challenging due to their less regular nature. Time series decomposition techniques or advanced statistical methods may be employed to capture these cycles.

Forecasting: Recognizing cyclic patterns can aid in forecasting, especially when these cycles are expected to affect future values.

Modeling and Forecasting With Irregular or Residual Component

Modeling: While the irregular component is not explicitly modeled, its presence is essential for modeling and forecasting accuracy. By removing the trend, seasonality, and cyclic components from the data, you are left with the residual component.

Forecasting: The irregular component is often treated as random noise and is not directly forecasted. However, accounting for it in model residuals helps quantify the uncertainty in forecasts and assess model accuracy.

Take Away

Time series components provide a structured framework for understanding the various underlying patterns in time series data.

Modeling and forecasting benefit from component-based analysis because it allows you to:

- Isolate Patterns: By separating the trend, seasonality, and cyclic components, you can focus on modeling and forecasting each pattern independently.
- Improve Model Accuracy: Models that account for these components can provide more accurate forecasts by capturing the data's underlying dynamics.
- Identify Anomalies: Detecting irregular patterns and anomalies in the residual component is crucial for data quality and identifying unexpected events.
- Extrapolate Future Behavior: Incorporating trend and seasonality in forecasting models enables predictions of future values based on historical patterns.

In practice, a combination of statistical techniques and domain knowledge is often used to analyze and model time series data effectively.

Let's look at how all of this is done

Homework #4

Generate synthetic time series data with a Linear Trend and Random Noise

Split data into training and test data

Train a simple linear regression model

Evaluate the model's performance using the mean squared error on the test data

Plot Actual vs Predicted values to visualize performance