

# COSC 325: Introduction to Machine Learning

Dr. Hector Santos-Villalobos

**Dr. Santos**



THE UNIVERSITY OF  
TENNESSEE  
KNOXVILLE

# Lecture 04: SciKit-Learn and Nearest Neighbor Classifier



THE UNIVERSITY OF  
TENNESSEE  
KNOXVILLE



# Class Announcements

## Homework:

Homework #1 due this Sunday

- Check instructions
- Check previous posts

Homework #2 published, due 09/08

## Lectures:

We may still need to finish Scikit-learn material. I will record any remaining content.

## Quizzes:

Quiz #2 published

## Course Project:

Give feedback by 09/10

Propose new datasets/use cases by 09/05

Friday, the last day to propose a team

# Today's Topics

## *Wrap up Numpy*



## *Matplotlib and Pandas*



## *Scikit-learn*





# Last Lecture

- Notation
  - Sample: vector feature  $x$ ,  $(m, 1)$ 
    - List of numbers
    - All modalities map to a list of numbers
  - Set of samples: matrix  $X$ ,  $(n, m)$
  - Rows -> samples  $(n)$
  - Columns -> features  $(m$  or  $n_x)$
  - Targets -> One per sample. Vector  $(n, 1)$
- Capacity
  - Ability of the model to fit the data
  - Increases with # of parameters
- Numpy Python library



**Let's go back to Numpy**

# Pop Quiz

1 | MULTIPLE CHOICE

POINTS: 1 |  Edit 

In Python, perform the following steps:

1. Create three Numpy arrays  $\mathbf{a} = [1, 2, 3]$ ,  $\mathbf{b} = [4, 5, 6]$ , and  $\mathbf{c} = [3, 2, 1]$  with shape (1, 3).
2. Add 5 to  $\mathbf{a}$ , divide  $\mathbf{b}$  by 4, and multiply  $\mathbf{c}$  by 3.
3. Concatenate  $\mathbf{a}$  and  $\mathbf{c}$  on the zero axis and store the result on variable  $\mathbf{d}$ .
4. Finally, store the dot product between  $\mathbf{d}$  and  $\mathbf{c}$  on variable  $\mathbf{f}$ . (HINT: Do you need to reshape  $\mathbf{c}$  before computing the dot product?)

What are the contents and shape of  $\mathbf{f}$ ?

- A.  $\mathbf{f}=[120]$ ,  $\mathbf{f.shape}=(1,1)$
- B.  $\mathbf{f}=[126]$ ,  $\mathbf{f.shape}=(1)$
- C.  $\mathbf{f}=[[120][126]]$ ,  $\mathbf{f.shape}=(2,1)$
- D.  $\mathbf{f}=[[120,126]]$ ,  $\mathbf{f.shape}=(1,2)$

# Pop Quiz

1 | MULTIPLE CHOICE

POINTS: 1 |  Edit 

In Python, perform the following steps:

1. Create three Numpy arrays  $\mathbf{a} = [1, 2, 3]$ ,  $\mathbf{b} = [4, 5, 6]$ , and  $\mathbf{c} = [3, 2, 1]$  with shape (1, 3).
2. Add 5 to  $\mathbf{a}$ , divide  $\mathbf{b}$  by 4, and multiply  $\mathbf{c}$  by 3.
3. Concatenate  $\mathbf{a}$  and  $\mathbf{c}$  on the zero axis and store the result on variable  $\mathbf{d}$ .
4. Finally, store the dot product between  $\mathbf{d}$  and  $\mathbf{c}$  on variable  $\mathbf{f}$ . (HINT: Do you need to reshape  $\mathbf{c}$  before computing the dot product?)

What are the contents and shape of  $\mathbf{f}$ ?

- A.  $\mathbf{f}=[120]$ ,  $\mathbf{f.shape}=(1,1)$
- B.  $\mathbf{f}=[126]$ ,  $\mathbf{f.shape}=(1)$
- C.  $\mathbf{f}=[[120][126]]$ ,  $\mathbf{f.shape}=(2,1)$
- D.  $\mathbf{f}=[[120,126]]$ ,  $\mathbf{f.shape}=(1,2)$

```
import numpy as np

a = np.array([1, 2, 3])
print(f"Original shape of a is {a.shape}")
a = a.reshape((1,3)) + 5
print(f"Shape of a is {a.shape}")
b = np.array([[4, 8, 12]]) / 4
print(f"Shape of b is {b.shape}")
c = np.array([[3, 2, 1]]) * 3
print(f"Shape of c is {c.shape}")

d = np.concatenate((a, c), axis=0)
print(f"Shape of d is {d.shape}")

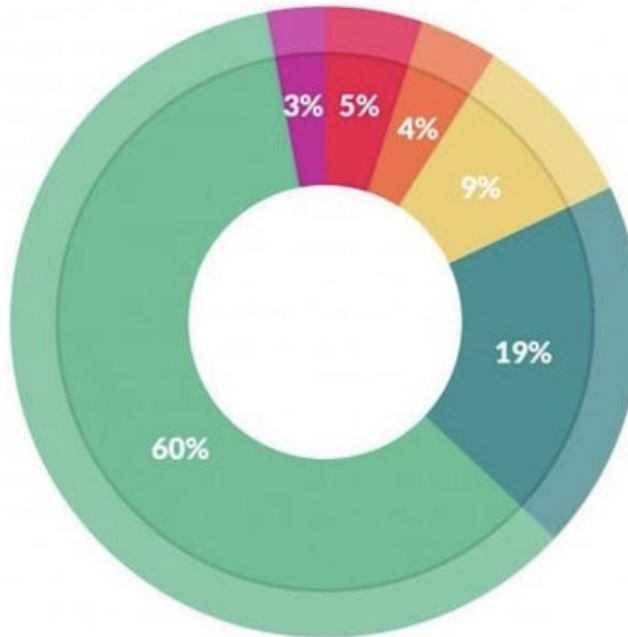
# take the transpose of c and compute the dot product between c and d
f = np.dot(d, c.T)
print(f"f value is {f}")
print(f"f shape is {f.shape}")
```

```
Original shape of a is (3,)
Shape of a is (1, 3)
Shape of b is (1, 3)
Shape of c is (1, 3)
Shape of d is (2, 3)
f value is [[120]
 [126]]
f shape is (2, 1)
```



# Data

- Data is fundamental to what we do in machine learning
- We need data to learn from, but data isn't always handed to us in the best format
- We likely don't know much about the data a priori



What data scientists spend the most time doing

- Building training sets: 3%
- Cleaning and organizing data: 60%
- Collecting data sets; 19%
- Mining data for patterns: 9%
- Refining algorithms: 4%
- Other: 5%

# Pandas

 [About us](#) [Getting started](#) [Documentation](#) [Community](#) [Contribute](#)

**pandas**

**pandas** is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

[Install pandas now!](#)

### Latest version: 2.2.2

- What's new in 2.2.2
- Release date: Apr 10, 2024
- Documentation (web)
- Download source code

### Follow us



### Getting started

- Install pandas
- Getting started

### Documentation

- User guide
- API reference
- Contributing to pandas
- Release notes

### Community

- About pandas
- Ask a question
- Ecosystem

### Recommended books



<https://pandas.pydata.org/>

# Pandas

- **Open-source** data analysis and manipulation tool
- A fast and efficient **DataFrame** object for data manipulation with integrated indexing
- Tools for **reading and writing data** between in-memory data structures and different formats
- Intelligent **data alignment** and integrated handling of **missing data**: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form;
- Flexible **reshaping** and pivoting of data sets;
- Intelligent label-based **slicing, fancy indexing, and subsetting** of large data sets;
- Columns can be inserted and deleted from data structures for **size mutability**;
- Aggregating or transforming data with a powerful **group by** engine allowing split-apply-combine operations on data sets;
- High-performance **merging and joining** of data sets;
- Highly **optimized for performance**, with critical code paths written in Cython or C.

# Scikit-learn

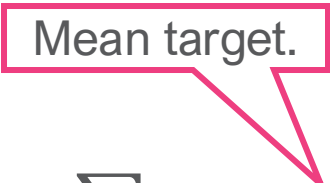
The screenshot shows the Scikit-learn website homepage. At the top, there is a navigation bar with links for 'Install', 'User Guide', 'API', 'Examples', 'Community', and 'More'. The version '1.5.1 (stable)' is displayed in the top right corner. The main header features the 'scikit-learn' logo and the tagline 'Machine Learning in Python'. Below the header, there are two buttons: 'Getting Started' and 'Release Highlights for 1.5'. A list of key features is provided: 'Simple and efficient tools for predictive data analysis', 'Accessible to everybody, and reusable in various contexts', 'Built on NumPy, SciPy, and matplotlib', and 'Open source, commercially usable - BSD license'. The page is divided into three main sections: 'Classification', 'Regression', and 'Clustering'. Each section includes a brief description, applications, and algorithms. The 'Classification' section features a grid of 15 small scatter plots. The 'Regression' section includes a line graph titled 'Predicted average energy transfer during the week'. The 'Clustering' section shows a scatter plot with colored regions and white crosses representing centroids.

<https://scikit-learn.org/>

# Coefficient of Determination

In statistics, the **coefficient of determination**, denoted  $R^2$  or  $r^2$  and pronounced "R squared", is the proportion of the variation in the dependent variable that is predictable from the independent variable(s). --Wikipedia

*Range from 0-1.*

$$SS_{residual} = \sum_i (y^{(i)} - \hat{y}^{(i)})^2 \quad SS_{total} = \sum_i (y^{(i)} - \bar{y})^2 \quad R^2 = 1 - \frac{SS_{residual}}{SS_{total}}$$


# Overview K-Nearest Neighbor (KNN)

- Intuition: Similar features  $x$  tend to have similar targets  $y$ .
- Training: Store all samples and labels as reference
- Inference:
  - Compute the distance from input sample to all training samples (e.g., Euclidian distance)
  - Identify the closest  $k$  training samples
  - Assign the most common training label

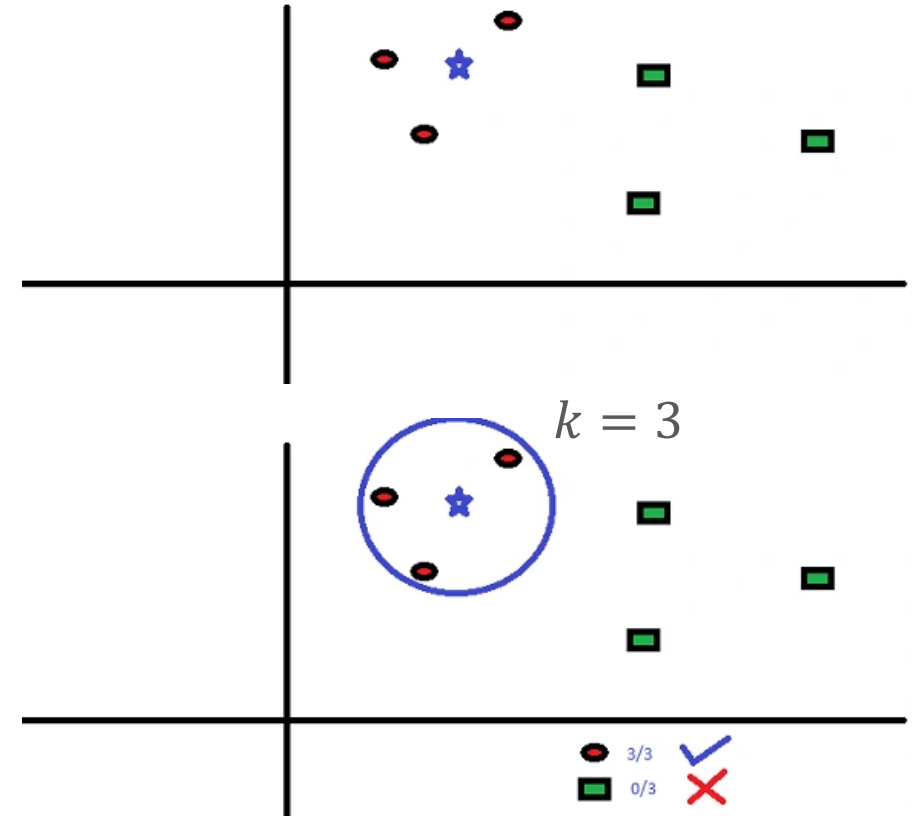


Image source: <https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>



# Pandas and Scikit-Learn Notebook

# Next Lectures

- Linear Regression
- Gradient Descent

