
# COMPSCI 389
# Introduction to Machine Learning

**Days:** Tu/Th. **Time:** 2:30 – 3:45 **Building:** Morrill 2 **Room:** 222

**Topic 3.0: Models, Algorithm Template, Nearest Neighbor**

Prof. Philip S. Thomas (pthomas@cs.umass.edu)

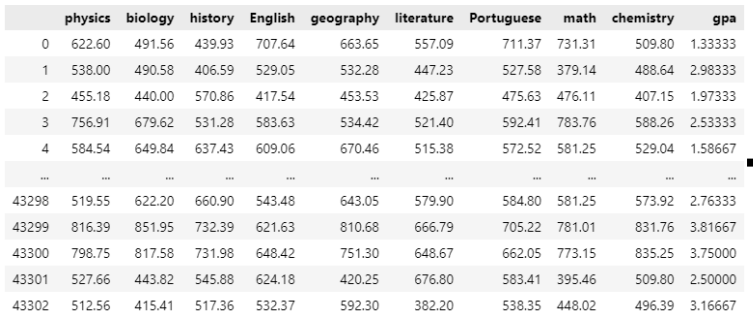
# Review

- Input output pair $(X, Y)$
  - $X$: Input, features, attributes, covariates, or predictors
    - Numerical (discrete/continuous), categorical (nominal/ordinal), etc.
  - $Y$: Output, label, or target
    - Regression: $Y$ is continuous.
    - Classification: $Y$ is discrete
- Data set: $(X_i, Y_i)_{i=1}^n$
- Query: An additional input $X$
- Goal: Predict the label $Y$ associated with $X$.

# Models (Supervised Learning)

- A **model** is a mechanism that maps input data to predictions.
- (Offline) **ML algorithms** take data sets as input and produce models as output.

**Online** ML algorithms can receive data over time, improving their models as more data becomes available.



| | physics | biology | history | English | geography | literature | Portuguese | math | chemistry | gpa |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 622.60 | 491.56 | 439.93 | 707.64 | 663.65 | 557.09 | 711.37 | 731.31 | 509.80 | 1.33333 |
| 1 | 538.00 | 490.58 | 406.59 | 529.05 | 532.28 | 447.23 | 527.58 | 379.14 | 488.64 | 2.98333 |
| 2 | 455.18 | 440.00 | 570.86 | 417.54 | 453.53 | 425.87 | 475.63 | 476.11 | 407.15 | 1.97333 |
| 3 | 756.91 | 679.62 | 531.28 | 583.63 | 534.42 | 521.40 | 592.41 | 783.76 | 588.26 | 2.53333 |
| 4 | 584.54 | 649.84 | 637.43 | 609.06 | 670.46 | 515.38 | 572.52 | 581.25 | 529.04 | 1.58667 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 43298 | 519.55 | 622.20 | 660.90 | 543.48 | 643.05 | 579.90 | 584.80 | 581.25 | 573.92 | 2.76333 |
| 43299 | 816.39 | 851.95 | 732.39 | 621.63 | 810.68 | 666.79 | 705.22 | 781.01 | 831.76 | 3.81667 |
| 43300 | 798.75 | 817.58 | 731.98 | 648.42 | 751.30 | 648.67 | 662.05 | 773.15 | 835.25 | 3.75000 |
| 43301 | 527.66 | 443.82 | 545.88 | 624.18 | 420.25 | 676.80 | 583.41 | 395.46 | 509.80 | 2.50000 |
| 43302 | 512.56 | 415.41 | 517.36 | 532.37 | 592.30 | 382.20 | 538.35 | 448.02 | 496.39 | 3.16667 |

Data Set

A query can be one **or more** feature vectors.

Query

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 798.75 | 817.58 | 731.98 | 648.42 | 751.30 | 648.67 | 662.05 | 773.15 | 835.25 |
| 527.66 | 443.82 | 545.88 | 624.18 | 420.25 | 676.80 | 583.41 | 395.46 | 509.80 |


Predictions are given for each feature vector in the query.

3.75000
2.50000

Prediction

# Training/Fitting

- ML algorithms could take a data set and query at the same time and output a prediction for the query.
  - Each time a new query is given, the algorithm re-processes the entire data set.
- **Idea**: More efficient to preprocess the data set, computing relevant statistics and quantities.
  - Given a query, the algorithm might reference the statistics and quantities it computed without re-referencing the data set at all!
  - This pre-processing of the data set is called **training.**
    - Sometimes: “Training the model”
    - Sometimes: “Fitting the model to data”
    - Sometimes: “Pre-processing data”

# Scikit-Learn Models

- Scikit-Learn is a popular ML library in python.
- It has objects called “models”.
  - These “models” are more than just models – they are complete ML algorithms.

# Scikit-Learn Models

- Scikit-Learn models implement the functions:
  - **`fit(self, X, y)`**: The function for fitting the model to the data (training the model given the data / preprocessing the data).
    - `X`: A 2D array-like structure (e.g., DataFrame) representing the features. Each row is a point and each column is a feature.
    - `y`: A 1D array-like structure (e.g., Series) representing the target values
    - Returns `self` to simplify chaining together operations.
  - **`predict(self, X)`**: The function for producing predictions given queries.
    - `X`: A 2D array-like structure representing the data for which predictions are to be made. Each row is a sample and each column is a feature.
    - Returns a numpy array of predicted labels/values.
- **Note**: Ideally fit and predict are compatible with `X` and `y` being DataFrames or numpy arrays.

# Scikit-Learn Models

```
from sklearn.base import BaseEstimator
import numpy as np

class CustomMLAlgorithm(BaseEstimator):
    def __init__(self, param1=1, param2=2):
        # Initialization code
        self.param1 = param1
        self.param2 = param2

    def fit(self, X, y):
        # Training code
        # Implement your training algorithm here
        return self

    def predict(self, X):
        # Prediction code
        # Implement your prediction algorithm here
        return np.zeros(len(X))
```

- Given data set (X,y) and query:

```
model = CustomMLAlgorithm()
model.fit(X,y)
predictiosn = model.predict(query)
```

# Nearest Neighbor

- A particularly simple yet effective ML algorithm based on the core idea:

  *When presented with a query, find the data point (row) that is most similar to the query and give the label associated with this most-similar point as the prediction.*

- We can map this to fit/predict functions:
  - `fit`: Store the data
  - `predict`: For each query row do the following
    - Loop over each row in the training data, computing the Euclidean distance between the query and the row.
    - Create an array holding the labels from the rows with the smallest distance to the query feature vector (often just one element).
    - Return an arbitrary (e.g., random) element of the array.

| Query: | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 414.7 | 456.95 | 705.58 | 499.43 | 513.53 | 543.15 | 408.51 | 384.38 | 442.49 | | | | |
| | | | | | | | | | | | | |
| physics | biology | history | English | geograp | literatu | Portugu | math | chemistry | | distance | | gpa |
| 622.6 | 491.56 | 439.93 | 707.64 | 663.65 | 557.09 | 711.37 | 731.31 | 509.8 | | | | 1.33333 |
| 538 | 490.58 | 406.59 | 529.05 | 532.28 | 447.23 | 527.58 | 379.14 | 488.64 | Nearest Neighbor | | | 2.98333 |
| 455.18 | 440 | 570.86 | 417.54 | 453.53 | 425.87 | 475.63 | 476.11 | 407.15 | → | | Prediction → | 1.97333 |
| 756.91 | 679.62 | 531.28 | 583.63 | 534.42 | 521.4 | 592.41 | 783.76 | 588.26 | | | | 2.53333 |
| 584.54 | 649.84 | 637.43 | 609.06 | 670.46 | 515.38 | 572.52 | 581.25 | 529.04 | | | | 1.58667 |
| 325.99 | 466.74 | 597.06 | 554.43 | 535.77 | 717.03 | 477.6 | 503.82 | 422.92 | | | | 1.66667 |
| 622.6 | 587.04 | 598.85 | 603.32 | 690.7 | 652.86 | 533.05 | 755.3 | 628.73 | | | | 3.72333 |
| 527.65 | 559.99 | 758.37 | 669.71 | 645.62 | 648.67 | 539.23 | 470.78 | 486.13 | | | | 3.08333 |
| 647.64 | 687.83 | 630.61 | 613.95 | 557.43 | 739.94 | 557.27 | 557.14 | 632.54 | | | | 0 |
| | | | | | | | | | | | | |

# See IPython Notebook (3.1)

# Intermission

- Class will resume in 5 minutes.
- Feel free to:
  - Stand up and stretch.
  - Leave the room.
  - Talk to those around you.
  - **Write a question on a notecard and add it to the stack at the front of the room.**