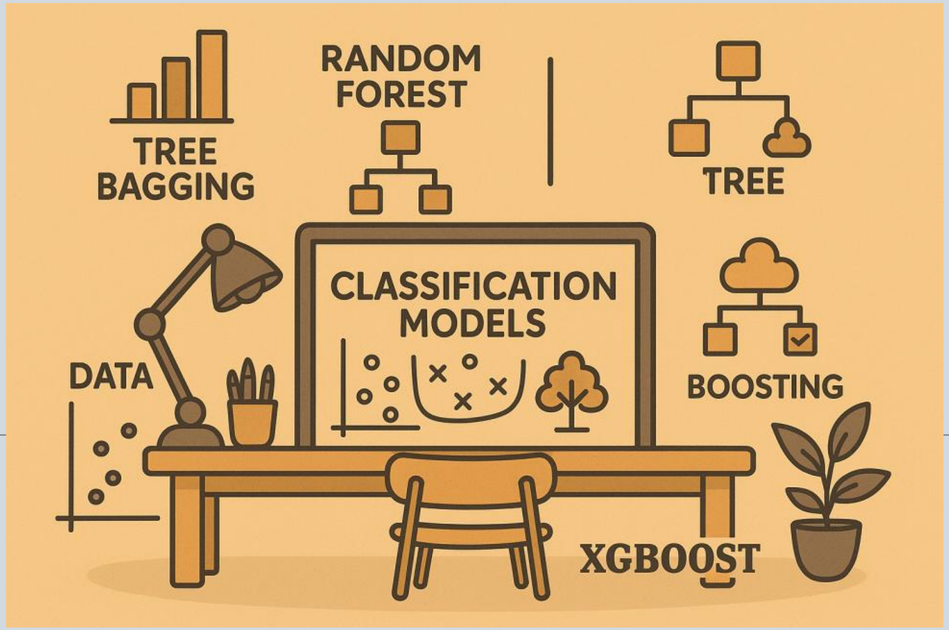
Advanced Classification Models

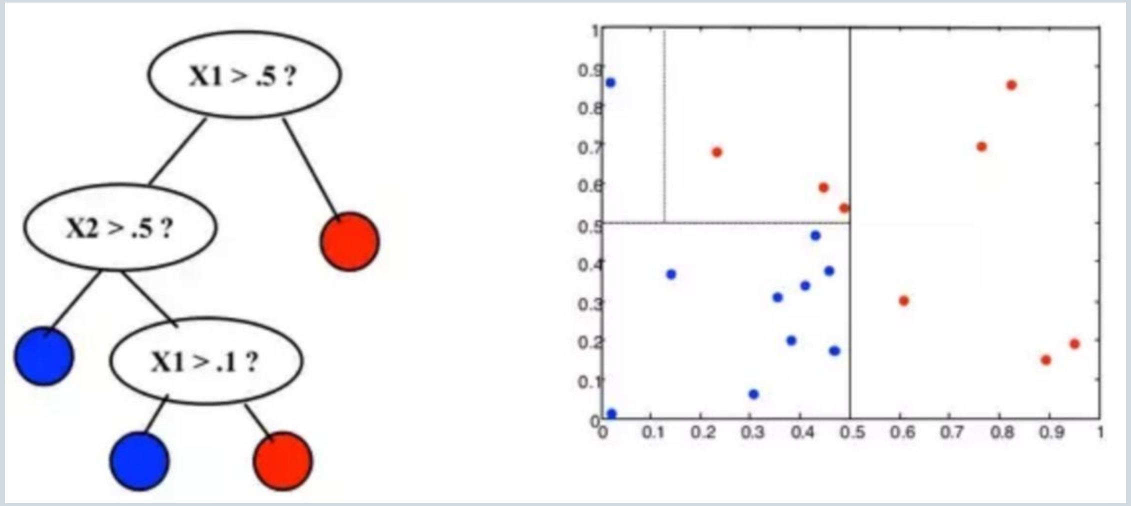


DR. ZVI BEN AMI

Agenda

- Ensemble Learning
- Tree Bagging
  - Random Forest
- Tree boosting
  - AdaBoost
  - XgBoost
  - CatBoost
- Distances based models
  - KNN
- Introduction to Image classifications

Decision Tree will Overfit!



Fully developed decision tree classifier:

- Training set error is always zero!
- Lots of variance (similar inputs may lead to very different outputs)

User must introduce some bias towards simpler trees

Where Can I Find The Shirt?

Source*	Past success
Shop in Ashkelon	50%
Shop in Jerusalem	70%
Online Shop	90%

* Assume independent probabilities

- What is my chance to find the shirt?

Where Can I Find The Shirt?

Source*	Past Success
Shop in Ashkelon	50%
Shop in Jerusalem	70%
Online Shop	90%

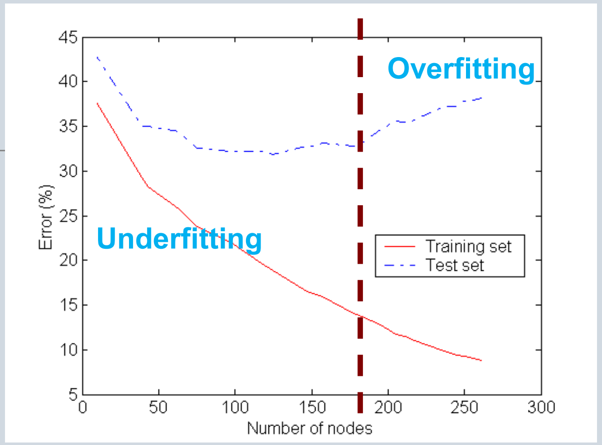
* Assume independent probabilities

- What is my chance to find the shirt?

$$p(t - shirt|Ashkelon\&Jerusalm) = 0.5 + (1 - 0.5) * 0.7 = 0.85$$

$$p(t - shirt|Ashkelon\&Jerusalm\&Online) = 0.85 + (1 - 0.85) * 0.9 = 0.985$$

Ensemble Learning



- Best learning results are produced by meta-models: models that combine several machine learning models.

Bagging (Bootstrap Aggregating):

- Reduce **overfitting** by creating a bunch of models. Each model overfits, but their average reduces overfitting.

Boosting aggregate weak learners to produce strong learners

- Reduce **underfitting** by creating a sequence of models, each trained to reduce errors of the previous one.

Stacking (improvement over boosting):

- Trains a model that optimally combines predictions from numerous models.

Ensemble Learning - Bagging

Bagging (Bootstrap Aggregating):

When to use: when basic method tends to **Overfit**

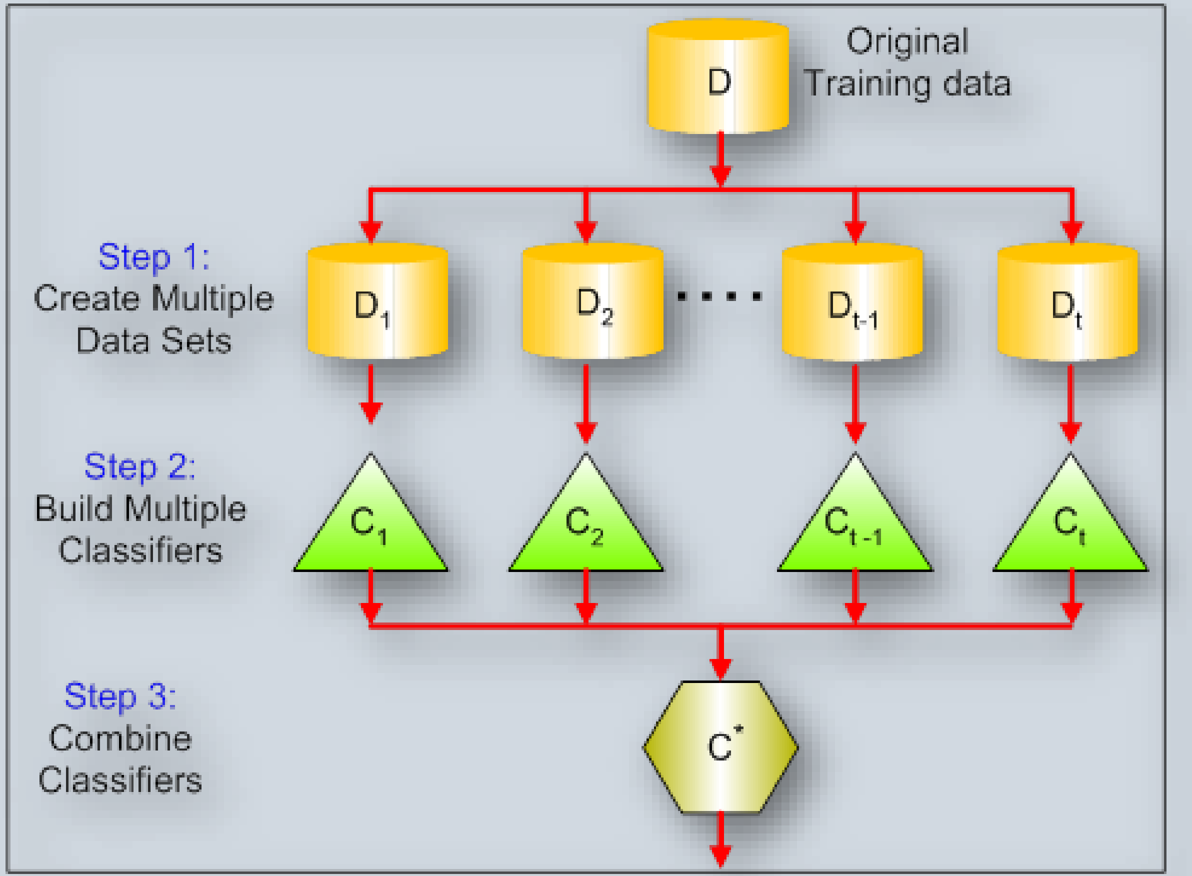
- Each of the artificial training sets:
  - Has the same size as the original training set
  - Generated by pooling data with repetitions (with replacement) from the original dataset
- All models have comparable performance (if compared to the one trained on the entire dataset)
- Models run in parallel (independently)
- The models are combined by majority vote (classification) or averaging (regression)
- Objective: reduce variance of the prediction
  - In general, overfitted model is very sensitive to the change in the input -> large variance in results.
  - Alternative view: different training sets result in very different models.

Random Forest: Bagging with Forests

The objective is to reduces variance of prediction by combining many independent classifiers

Random forests fight data overfitting

1. **Create Multiple Datasets (**each of the same size as the original)
2. **Build Multiple Classifiers**
3. **Combine Classifiers**



Decision Tree

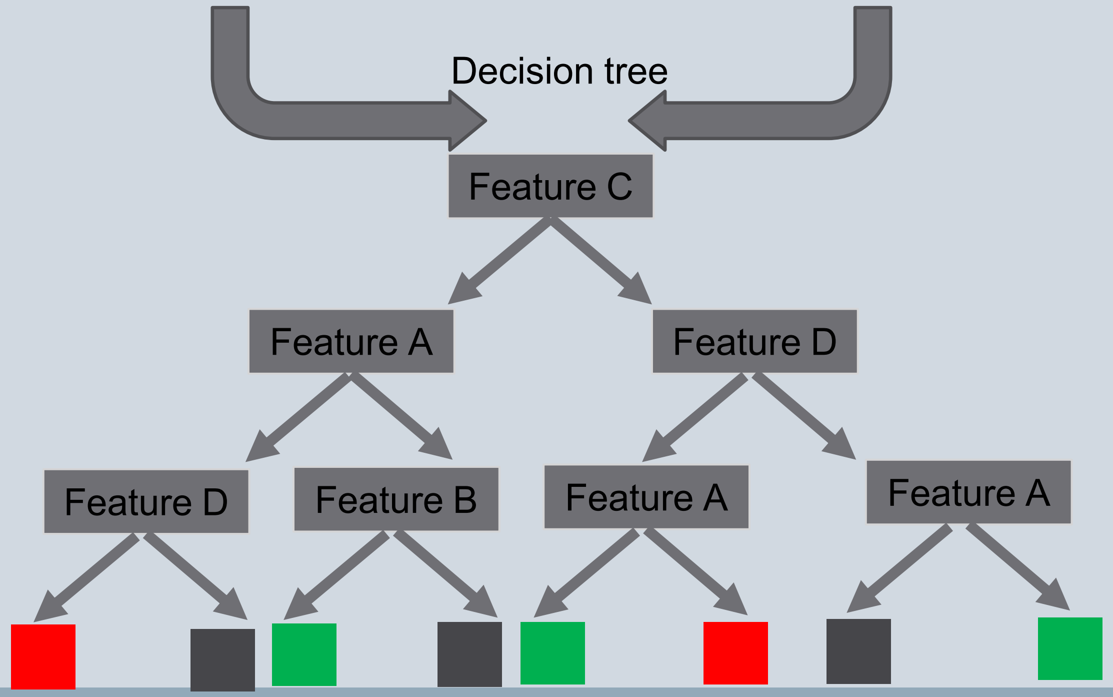
Feature B of the sample 1

$$X = \begin{bmatrix} f_{A1} & f_{B1} & f_{C1} & f_{D1} \\ f_{A2} & f_{B2} & f_{C2} & f_{D2} \\ \vdots & \vdots & \vdots & \vdots \\ f_{AN} & f_{BN} & f_{CN} & f_{DN} \end{bmatrix} \qquad Y = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_N \end{bmatrix}$$

N Samples
Features: A, B, C, D

Classes

Not stable
There are many alternatives
Each alternative yields different results.

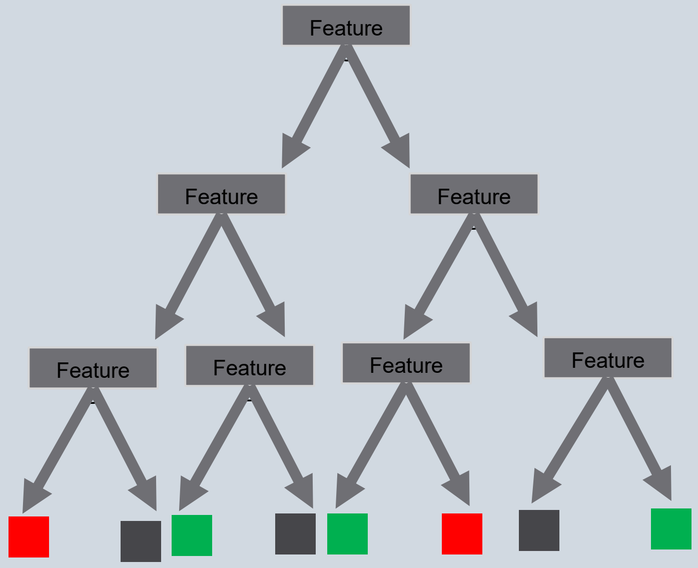


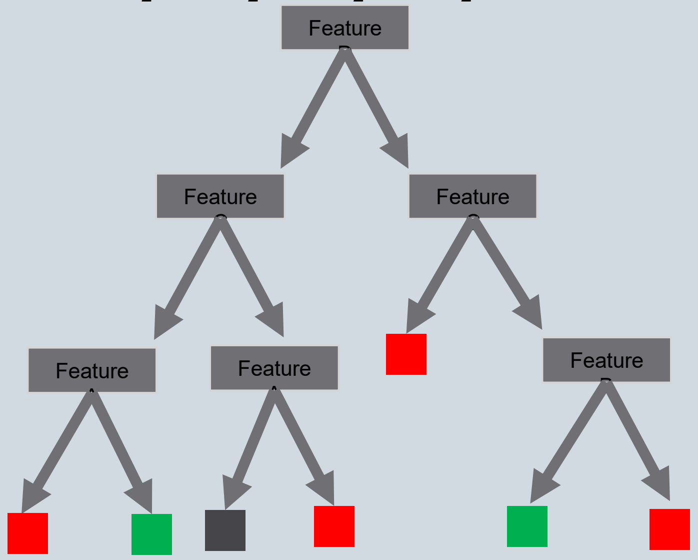
Random Forest: Bagging

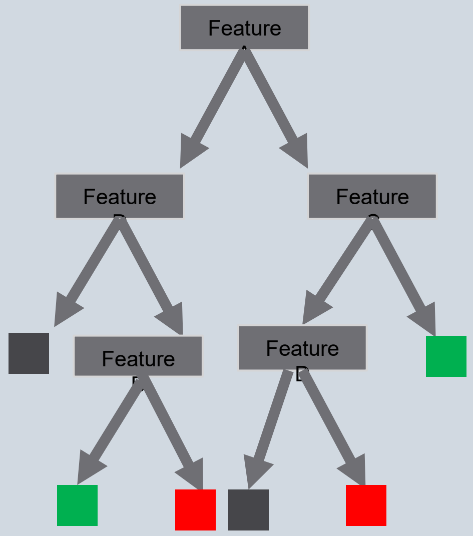
$$X_1 = \begin{bmatrix} f_{A7} & f_{B7} & f_{C7} & f_{D7} \\ f_{A12} & f_{B12} & f_{C12} & f_{D12} \\ \vdots & \vdots & \vdots & \vdots \\ f_{AN_1} & f_{BN_1} & f_{CN_1} & f_{DN_1} \end{bmatrix} \quad Y_1 = \begin{bmatrix} C_7 \\ C_{12} \\ \vdots \\ C_{N_1} \end{bmatrix}$$

$$X_2 = \begin{bmatrix} f_{A4} & f_{B4} & f_{C4} & f_{D4} \\ f_{A8} & f_{B8} & f_{C8} & f_{D8} \\ \vdots & \vdots & \vdots & \vdots \\ f_{AN_2} & f_{BN_2} & f_{CN_2} & f_{DN_2} \end{bmatrix} \quad Y_2 = \begin{bmatrix} C_4 \\ C_8 \\ \vdots \\ C_{N_2} \end{bmatrix}$$

$$X_3 = \begin{bmatrix} f_{A5} & f_{B5} & f_{C5} & f_{D5} \\ f_{A6} & f_{B6} & f_{C6} & f_{D6} \\ \vdots & \vdots & \vdots & \vdots \\ f_{AN_3} & f_{BN_3} & f_{CN_3} & f_{DN_3} \end{bmatrix} \quad Y_3 = \begin{bmatrix} C_5 \\ C_6 \\ \vdots \\ C_{N_4} \end{bmatrix}$$



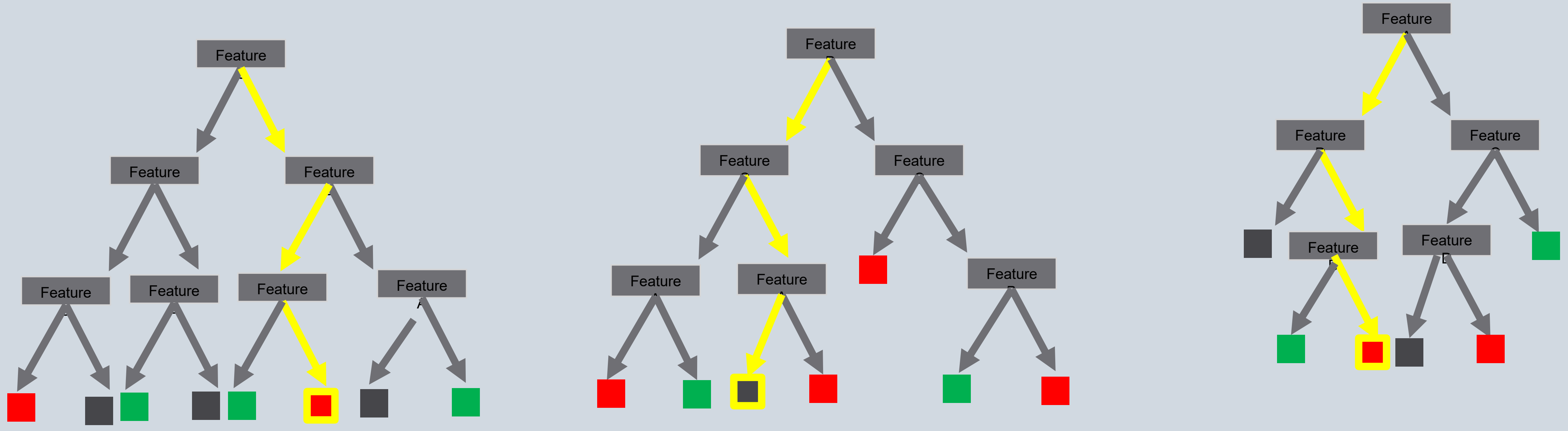




$$X = \begin{bmatrix} f_{A1} & f_{B1} & f_{C1} & f_{D1} \\ f_{A2} & f_{B2} & f_{C2} & f_{D2} \\ \vdots & \vdots & \vdots & \vdots \\ f_{AN} & f_{BN} & f_{CN} & f_{DN} \end{bmatrix} \qquad Y = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_N \end{bmatrix}$$

Classifying With Random Forest

Given an observation (a set of features)



Application of these trees yields: 2x 🟥 and 1x ■ result 🟥

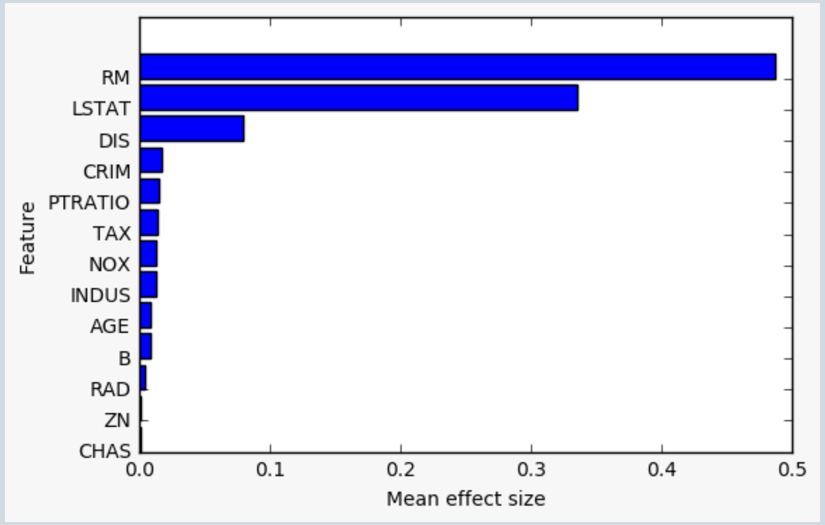
Random Forest

Advantages

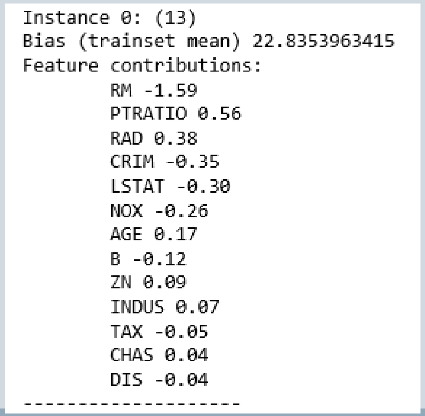
- Can work on classification and regression problems
- Scalable – can handle datasets with thousands of features.
- Can be used to identify the most significant variables
- Parallelizable

Disadvantages

- Regression random forests do not yield continues results.
- Black box – too complex to interpret.
- (Relatively) Slow to make predictions



Getting insights from the tree



```
Instance 0: (13)
Bias (trainset mean) 22.8353963415
Feature contributions:
        RM -1.59
        PTRATIO 0.56
        RAD 0.38
        CRIM -0.35
        LSTAT -0.30
        NOX -0.26
        AGE 0.17
        B -0.12
        ZN 0.09
        INDUS 0.07
        TAX -0.05
        CHAS 0.04
        DIS -0.04
--------------------
```

Classification_Models_Part_B.ipynb

How Good Is Random Forest?

Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems. *Journal of Machine Learning Research*, *15*(1), 3133-3181.

- Evaluate 179 classifiers arising from 17 families
- 121 data sets - the whole UCI data base
- Find that Random Forest is the overall best classifier

Ensemble Learning - Boosting

Boosting: aggregate weak learners to produce strong learners

- The original dataset is ***split*** into smaller datasets
- Each model has weaker performance (if compared to the one trained on the entire dataset)
- Models run sequentially. New training sets are constructed while giving misclassified samples higher preference
- Combination of models (e.g. majority vote) produces better predictions
- Objective: improve prediction, reduce bias
- When to use: when basic method tends to **Underfit** (not flexible enough, can’t describe the real relationship in the data)
- Works because every model focuses on fixing misclassification of previous models.

Tree Boosting

- Random Forest – bagging of trees. Each tree has low bias, but high variance (overfit).
- Boosted trees: sequential decision using a bunch of shallow trees (high bias, low variance).
- Adaboost (Adaptive Boosting - sklearn.ensemble.AdaBoostClassifier)
  - Repeatedly retrains *any model*. Misclassifications of the previous step are assigned higher weight in the following step.

- Initialize:
  - Each sample is assigned a weight – how important is it to classify it correctly?

- Train a classifier.
  - Assign new weights to each sample, based on their classification error.
  - Assign weight to the classifier, based on its overall accuracy.
- Repeat until:
  - Either all samples are classified correctly
  - Or maximum allowed number of classifiers trained.

Classification_Models_Part_B.ipynb

Tree Boosting

- Random Forest – bagging of trees. Each tree has low bias, but high variance (overfit).
- Boosted trees: sequential decision using a bunch of shallow trees (high bias, low variance).
- Adaboost (Adaptive Boosting - sklearn.ensemble.AdaBoostClassifier)
    - Repeatedly retrains *any model*. Misclassifications of the previous step are assigned higher weight in the following step.

Gradient Tree Boosting

- Consequent model is trained to predict errors of the earlier model.
- XGBoost (eXtreme Gradient Boosting)– specific implementation of boosting: fast way of finding a sequence of shallow trees and weighting their decisions together.
- Many alternative implementations for tree boosting exist: GBM (R),LightGBM (Microsoft), CatBoost (Yandex)

- Difference from Random forest:
- Random Forest – overfit; XGBoost underfit.
- Boosting is sequential (RF – parallel)
- pip install xgboost

Classification_Models_Part_B.ipynb

CatBoost

- CatBoost (Yandex): gradient-boosted decision trees with first-class categorical feature support.
- Core ideas:
  - Ordered boosting to reduce target leakage.
  - Target statistics for categorical encoding done on the fly with permutations.
  - Symmetric (oblivious) trees → fast inference, regularization by design.
  - Works well out-of-the-box (strong defaults, fewer knobs than XGBoost/LightGBM).
  - Handles missing values and text/categorical features natively.
  - Optional GPU training Requires a CUDA-enabled environment and a GPU build of CatBoost.

- When to Prefer CatBoost
  - Many high-cardinality categoricals (IDs, product codes, clinics, teams).
  - Mixed tabular data where manual one-hot would explode feature space.
  - Need strong default performance with minimal tuning.
  - Want fast, compact models for production inference.

Classification_Models_Part_B.ipynb

Ensemble Learning Stacking

Stacking (improvement over boosting - Improve Predictions) – in python: *StackingClassifier*

- Perform boosting
- Train another predictive model (e.g. logit) using input and prediction of the boosted models to assess performance of each of the models. Use *out of sample data*.
- Combine predictions by appropriately weighting the models in the ensemble.
- Objective: reduce variance and improve prediction

Stacking Ensemble

- Step 1 – Base Models (Level 0):
  - Train diverse models on the same dataset (e.g., Decision Tree, Logistic Regression, KNN, Random Forest).
- Step 2 – Meta-Model (Level 1):
  - Collect predictions from base models. Use them as input features for a new model (often Logistic Regression or XGBoost).
- Step 3 – Out-of-Sample Predictions:
  - Train the meta-model only on predictions from a validation set (to avoid overfitting).
- Step 4 – Final Prediction:
  - Meta-model learns which base models perform best under different conditions and combines them optimally.

- Key Idea: Stacking *learns how to combine models*, rather than just averaging or voting.

Distance-based Classification

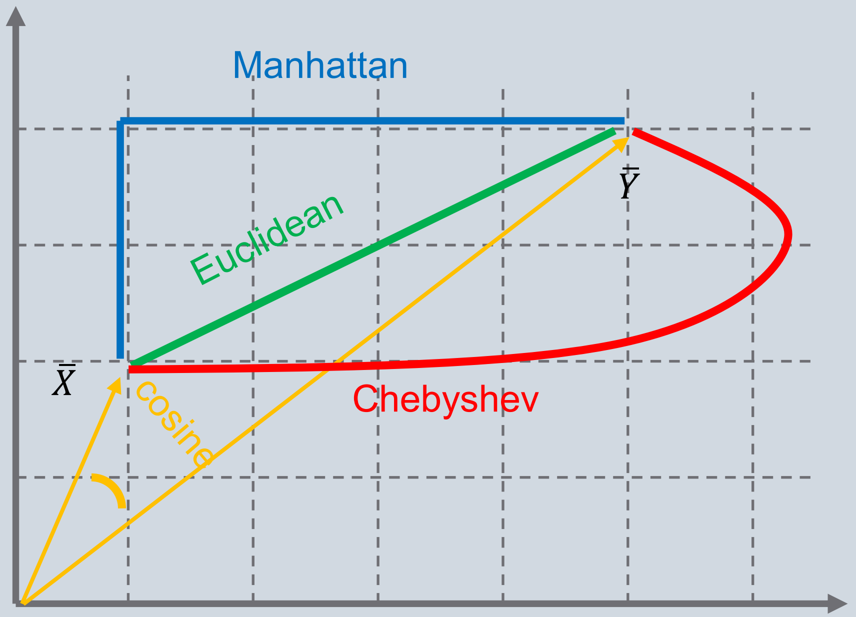
- Classification (General):
  - Given a collection of records, where each record is a set of attributes and a class (category), the objective is to find a model for the class as a function of the values of other attributes.
  - The model is inferred from a training set
  - Trained model can make accurate predictions for the (previously unseen) test set
- Distance-based classification:
  - Place items in class to which they are “closest”
  - Must determine distance between an item and a class
  - Classes can be represented by the central value (Centroid), representative point (Medoid) or a set of points.

20

Similarity Measures

Given two vectors, $\bar{X}$ and $\bar{Y}$, find how "similar" they are

- Euclidean Distance: $Euclidean(\bar{X},\bar{Y}) = \sqrt{\sum(x_i - y_i)^2}$
- Manhattan Distance: $Mangattan(\bar{X},\bar{Y}) = \sqrt{\sum|x_i - y_i|}$
- Minkowski Distance: $Minkowski(\bar{X},\bar{Y}) = \sqrt[\gamma]{\sum|x_i - y_i|^\gamma}$
- Cosine Similarity: $CosineSim(\bar{X},\bar{Y}) = \frac{\sum x_i y_i}{\sqrt{x_i^2}\sqrt{y_i^2}}$



γ	$Minkowski(\bar{X},\bar{Y})$ **equivalent**	**Common name**	**Main Properties**
1	$Manhattan(\bar{X},\bar{Y})$	$L1$	Robust to outliers
2	$Euclidean(\bar{X},\bar{Y})$	$L2$	Invariant to rotation
∞	$Chebyshev(\bar{X},\bar{Y})$	Lmax-norm	$\sim\max(|x_i - y_i|)$

Similarity/Distance Between Data Items

Each data item is represented with a set of attributes (consider them to be numeric for the time being)

Michael:
Age=30
Income=105K
No. of children=4

Andrea:
Age=35
Income=200K
No. of children =1

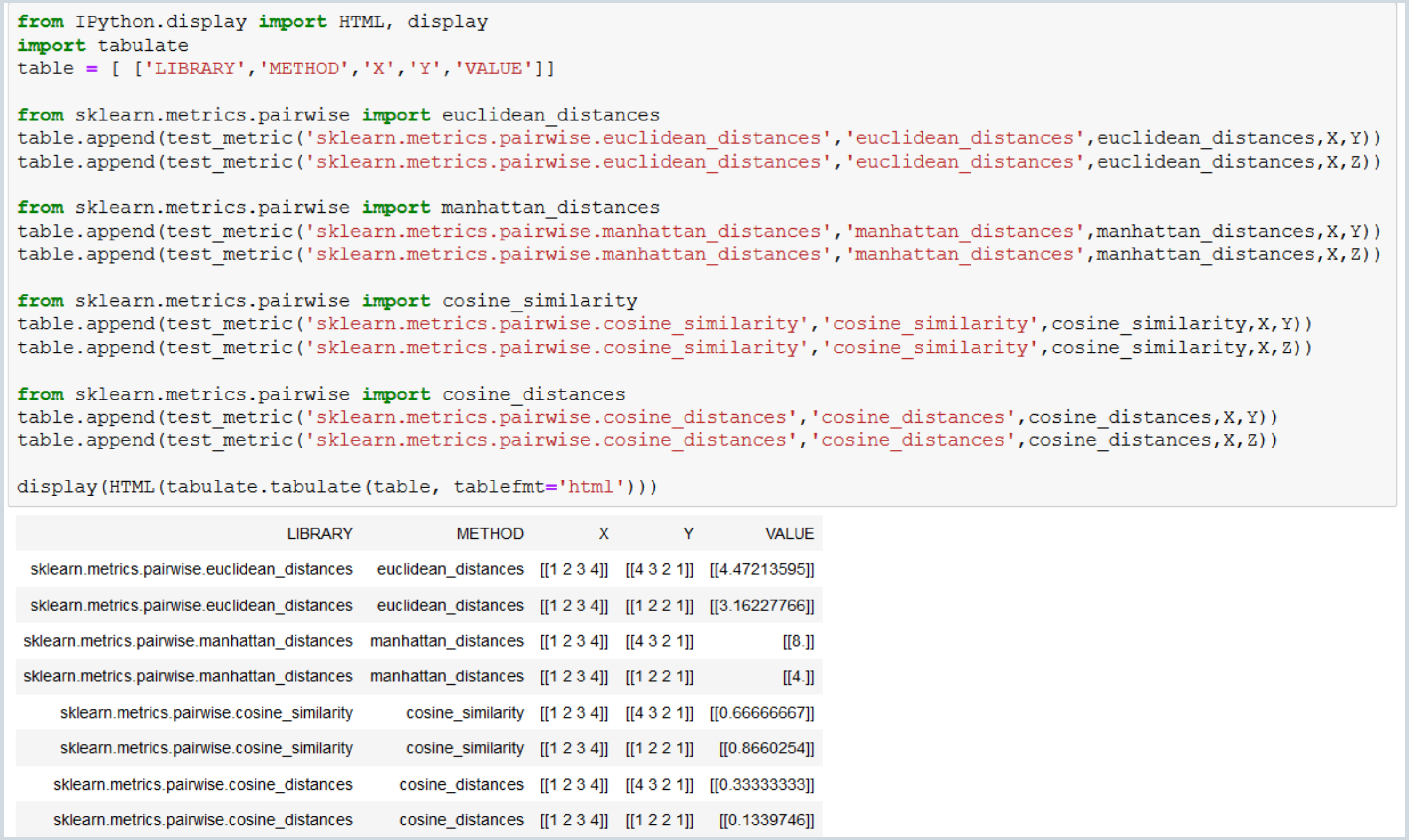
The Euclidean distance between Michael and Andrea is:

$$Distance(John, Rachel) = \sqrt{(30-35)^2 + (105K - 200K)^2 + (4-1)^2}$$

$$Euclidean(\bar{X}, \bar{Y}) = \sqrt{\sum (x_i - y_i)^2}$$

Distances in Python

Classification_Models_Part_B.ipynb



```python
from IPython.display import HTML, display
import tabulate
table = [ ['LIBRARY','METHOD','X','Y','VALUE']]

from sklearn.metrics.pairwise import euclidean_distances
table.append(test_metric('sklearn.metrics.pairwise.euclidean_distances','euclidean_distances',euclidean_distances,X,Y))
table.append(test_metric('sklearn.metrics.pairwise.euclidean_distances','euclidean_distances',euclidean_distances,X,Z))

from sklearn.metrics.pairwise import manhattan_distances
table.append(test_metric('sklearn.metrics.pairwise.manhattan_distances','manhattan_distances',manhattan_distances,X,Y))
table.append(test_metric('sklearn.metrics.pairwise.manhattan_distances','manhattan_distances',manhattan_distances,X,Z))

from sklearn.metrics.pairwise import cosine_similarity
table.append(test_metric('sklearn.metrics.pairwise.cosine_similarity','cosine_similarity',cosine_similarity,X,Y))
table.append(test_metric('sklearn.metrics.pairwise.cosine_similarity','cosine_similarity',cosine_similarity,X,Z))

from sklearn.metrics.pairwise import cosine_distances
table.append(test_metric('sklearn.metrics.pairwise.cosine_distances','cosine_distances',cosine_distances,X,Y))
table.append(test_metric('sklearn.metrics.pairwise.cosine_distances','cosine_distances',cosine_distances,X,Z))

display(HTML(tabulate.tabulate(table, tablefmt='html')))
```

LIBRARY	METHOD	X	Y	VALUE
sklearn.metrics.pairwise.euclidean_distances	euclidean_distances	[[1 2 3 4]]	[[4 3 2 1]]	[[4.47213595]]
sklearn.metrics.pairwise.euclidean_distances	euclidean_distances	[[1 2 3 4]]	[[1 2 2 1]]	[[3.16227766]]
sklearn.metrics.pairwise.manhattan_distances	manhattan_distances	[[1 2 3 4]]	[[4 3 2 1]]	[[8.]]
sklearn.metrics.pairwise.manhattan_distances	manhattan_distances	[[1 2 3 4]]	[[1 2 2 1]]	[[4.]]
sklearn.metrics.pairwise.cosine_similarity	cosine_similarity	[[1 2 3 4]]	[[4 3 2 1]]	[[0.66666667]]
sklearn.metrics.pairwise.cosine_similarity	cosine_similarity	[[1 2 3 4]]	[[1 2 2 1]]	[[0.8660254]]
sklearn.metrics.pairwise.cosine_distances	cosine_distances	[[1 2 3 4]]	[[4 3 2 1]]	[[0.33333333]]
sklearn.metrics.pairwise.cosine_distances	cosine_distances	[[1 2 3 4]]	[[1 2 2 1]]	[[0.1339746]]

KNN - K Nearest neighbors

A group of algorithms capable to find neighboring points in multi-dimensional space.

Can be used to

- segment (group) data
- Recommender systems
- Search engines

KNN can be unsupervised (for clustering) or supervised (for classification)

KNN can use different distance metrics (e.g. Euclidian - sklearn.neighbors.DistanceMetric)

KNN can retrieve

- k nearest neighbors
- All neighbors within specified distance
- Neighbors until drop off
- Predefined number of groups

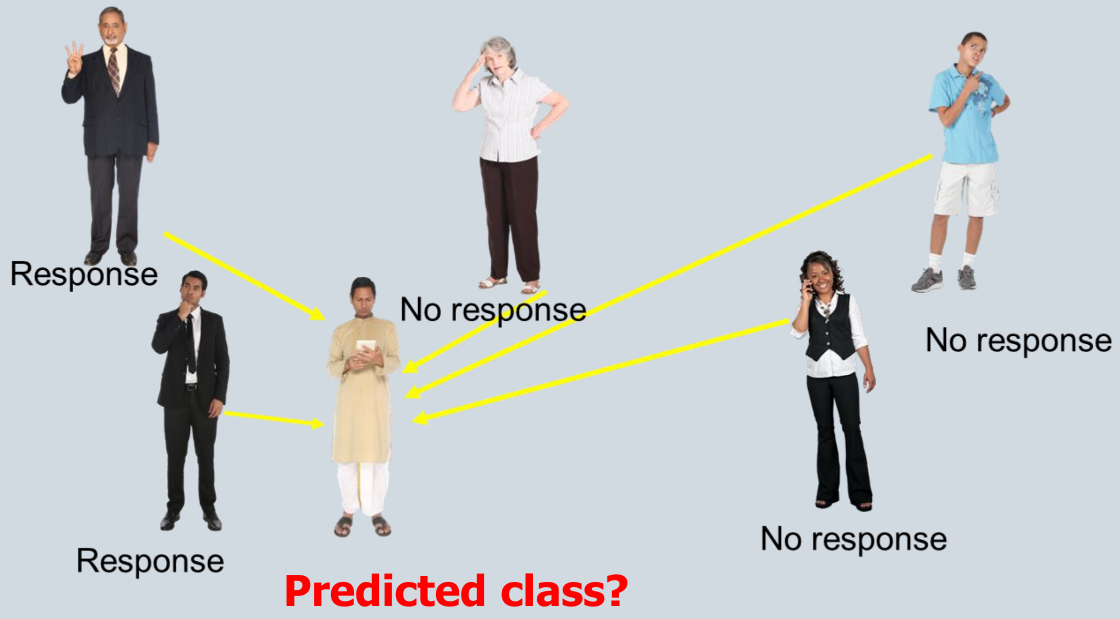
KNN finds nearest neighbors. For clustering/classification one has to vote for labels.

- Classification provides sample labels

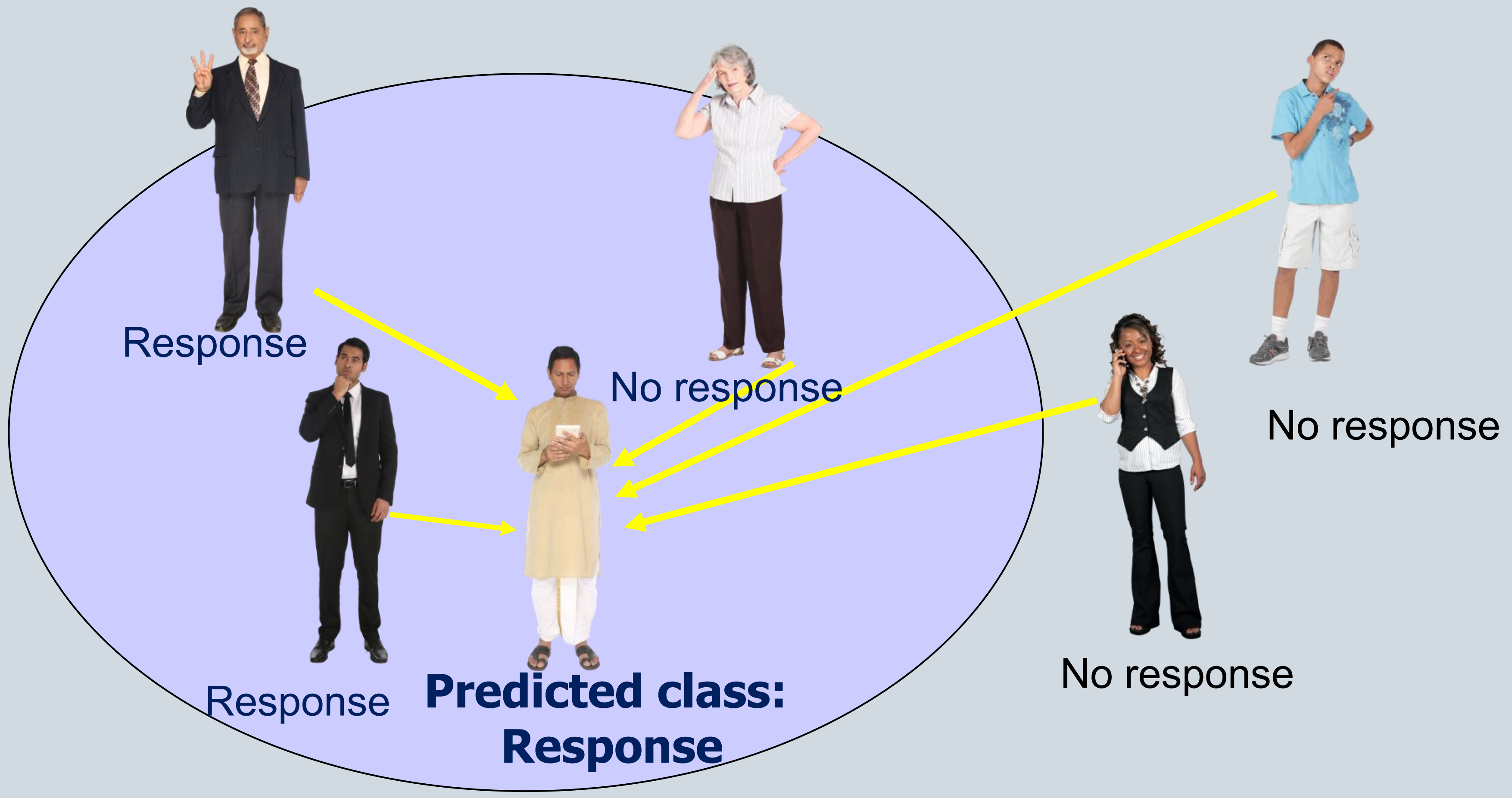
KNN for Prediction

To make a prediction for a new example E:

- Calculate the distance between E and all examples in the training set
- Select k examples closest to E in the training set
- Combine the k values for the target variable using some combining function (for example: mode, mean, median)



KNN Prediction



KNN Classifier – Example (K=3)

Customer	Age	Income (K)	No Children	Response
Michael	30	105	4	Yes
Andrea	35	200	1	No
Ester	65	150	3	No
David	66	120	3	No
Amit	23	50	0	Yes
John	31	100	2	?

KNN Classifier – Example (K=3)

Customer	Age	Income (K)	No Children	Response	Distance to John
Michael	30	105	4	Yes	$distance_{to_{John}} = \sqrt{(30-31)^2+(105-100)^2+(4-2)^2}$ **5.477**
Andrea	35	200	1	No	$distance_{to_{John}} = \sqrt{(35-31)^2+(200-100)^2+(1-2)^2}$ **100.084**
Ester	65	450	3	No	$distance_{to_{John}} = \sqrt{(65-31)^2+(450-100)^2+(3-2)^2}$ **351.648**
David	66	120	3	No	$distance_{to_{John}} = \sqrt{(66-31)^2+(120-100)^2+(3-2)^2}$ **40.323**
Amit	23	50	0	Yes	$distance_{to_{John}} = \sqrt{(23-31)^2+(50-100)^2+(0-2)^2}$ **50.675**
John	31	100	2	Yes	

KNN Classifier Strengths and Weaknesses

Strengths:

- Simple to implement and use
- Comprehensible – easy to explain prediction
- Robust to noisy data by averaging k-nearest neighbors
- Some appealing applications
  - consider collaborative filtering for personalized recommendations
- Distance function can be tailored using domain knowledge

Weaknesses:

- Need a lot of space to store all examples
- Takes (much) more time to classify a new example than with a parsimonious model (need to calculate and compare distance from new example to all training examples)
- Distance function must be designed carefully, using domain knowledge

KNN Classifier – Normalize Data

Michael:
Age=30
Income=105K
No. of children=4

Andrea:
Age=35
Income=200K
No. of children=1

Distance (Michael, Andrea)=sqrt $[(30-31)^2$ **$+(105{,}000-100{,}000)^2$** $+(1-2)^2]$

Distance between neighbors can be <u>dominated</u> by an attribute with relatively large values (e.g., income in our example).

Important to *normalize* (e.g., map numbers to numbers between 0-1)

Example: Income

Highest income = 500K

John's income is normalized to 95/500, Rachel's income is normalized to 215/500, etc.)

(there are more sophisticated ways to normalize)

KNN Classifier – Example (K=3)

Customer	Age	Income (K)	No Children	Response	Distance to John
Michael	**30/66 = 0.455**	**105/450 = 0.233**	**4/4 = 1**	**Yes**	$\boldsymbol{distance_{to_{John}} = \sqrt{(0.455-0.47)^2+(0.233-0.222)^2+(1-0.5)^2}}$ **0.506**
Andrea	**35/66 = 0.530**	**200/450 = 0.444**	**1/4 = 0.25**	**No**	$\boldsymbol{distance_{to_{John}} = \sqrt{(0.53-0.47)^2+(200-0.222)^2+(0.25-0.5)^2}}$ **0.340**
Ester	65/66 = 0.985	450/450 = 1	3/4 = 0.75	No	$distance_{to_{John}} = \sqrt{(0.985-0.47)^2+(450-0.222)^2+(0.75-0.5)^2}$ **0.966**
David	66/66 = 1	120/450 = 0.267	3/4 = 0.75	No	$distance_{to_{John}} = \sqrt{(1-0.47)^2+(120-0.222)^2+(0.75-0.5)^2}$ **0.588**
Amit	23/66 = 0.348	50/450 = 0.111	0/4 = 0	Yes	$distance_{to_{John}} = \sqrt{(0.348-0.47)^2+(0.111-0.222)^2+(0-0.5)^2}$ **0.527**
John	31/66 = 0.470	100/450 = 0.222	2/4 = 0.5	Yes	

Categorical Features in KNN

- Challenge:
  - KNN relies on distance metrics, naturally numeric.
  - Categorical features (e.g., color) lack natural distances.
- Approaches:
  - One-Hot Encoding → binary vectors (good for low-cardinality).
  - Ordinal Encoding → maps categories to integers, but may impose false order.
  - Specialized distances:
    - Hamming
    - Gower

Hamming and Gower Distances

- Hamming Distance:
  - Counts mismatches between categorical features.
  - Example:
    - [Red, Large] vs [Blue, Large] → 1 mismatch.

$$d_{ij} = \begin{cases} 0 \; if \; category \; is \; the \; same \\ 1 \; if \; category \; is \; different \end{cases}$$

- Gower Distance:
  - Handles mixed numeric + categorical features.
  - Numeric: normalized difference.
  - Categorical: 0 if same, 1 if different.
  - Example:
    - Age=30,Red vs Age=40,Blue
    - Age difference = 0.2
    - Color difference = 1
    - Average = (0.2+1)/2 = 0.6

Classification_Models_Part_B.ipynb

KNN: K Nearest neighbors Classifier

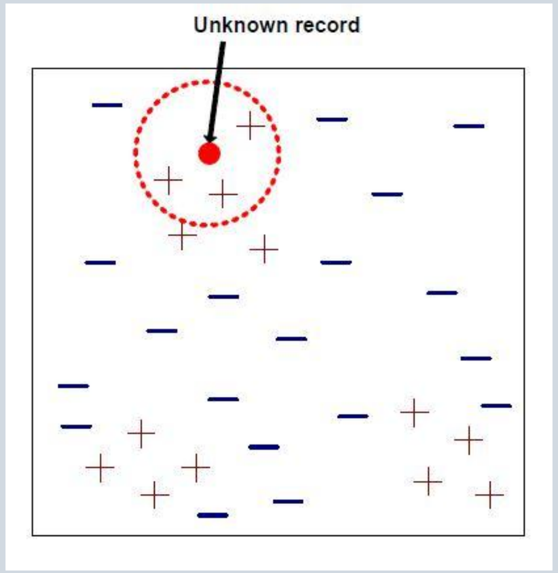
KNN uses k closest points to perform classification

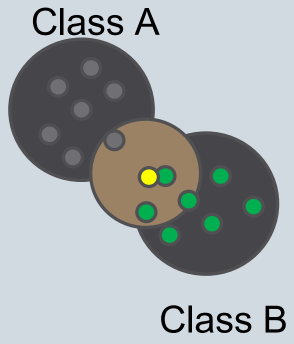
To train requires:

- The set of labeled records: attributes (X) and labels (y)
- Distance metric to compute distance between records
- k – the number of nearest neighbors to check

To classify unknown point

- Compute distance to training records
- Identify k nearest neighbors
- In classification: The item is placed in the class with the most number of close items (majority vote)
- In regression: The output is the average across values of k-nearest neighbors





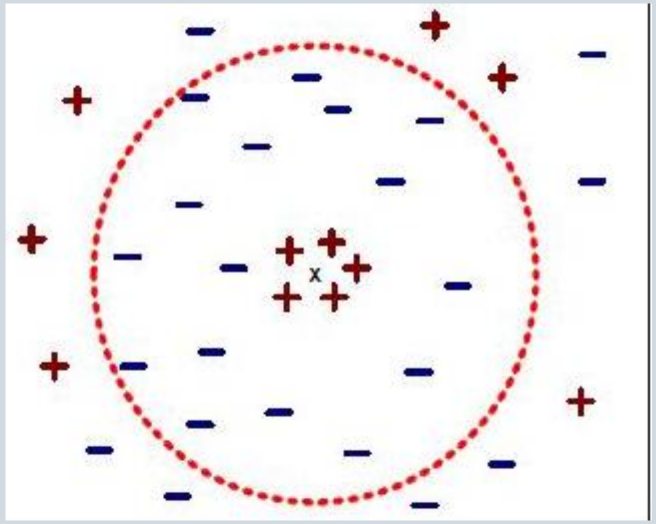
KNN – choosing k

If k is too small, sensitive to noise points

If k is too large, neighborhood may include points from other classes

KNN – CONSTRUCTION

- KNN does not create a model of the data
- KNN stored the training data for effective retrieval
    - Uses a number of approaches (e.g.: ball tree, kd-tree, brute force)
- https://www.analyticsvidhya.com/blog/2017/11/information-retrieval-using-kdtree/

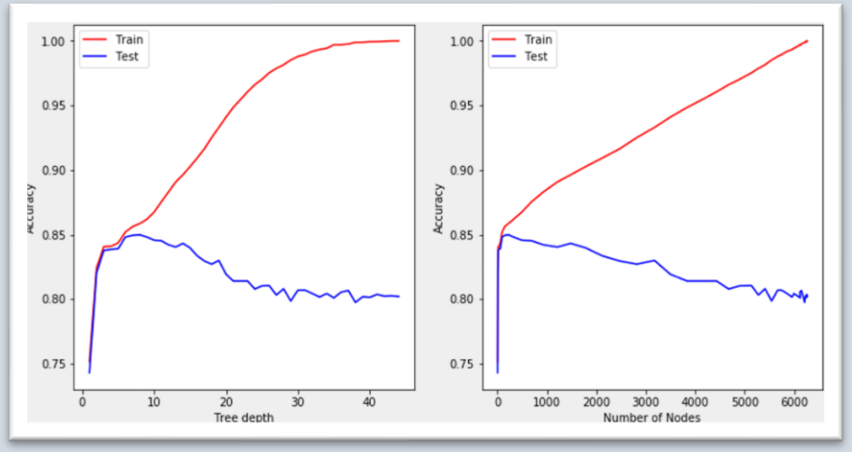


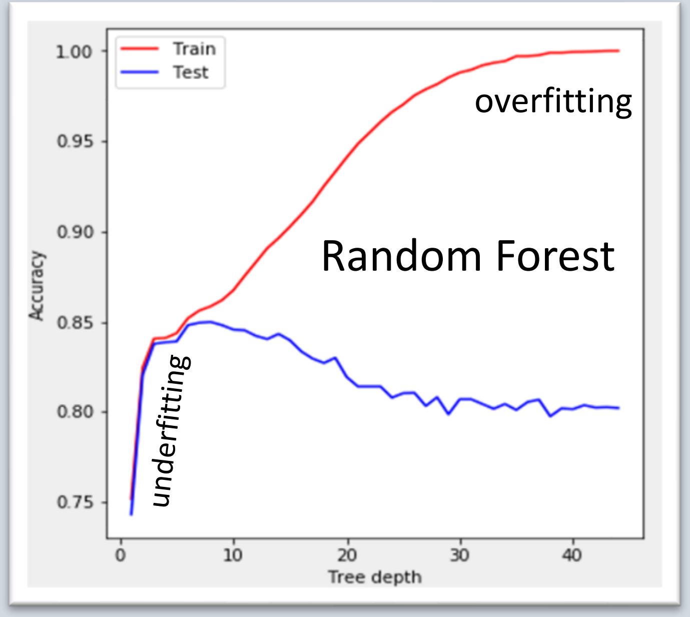
Classification_Models_Part_B.ipynb

Example

Classification_Models_Part_B.ipynb

Classifying people by income ($< \$50k$ vs $\geq \$50k$)





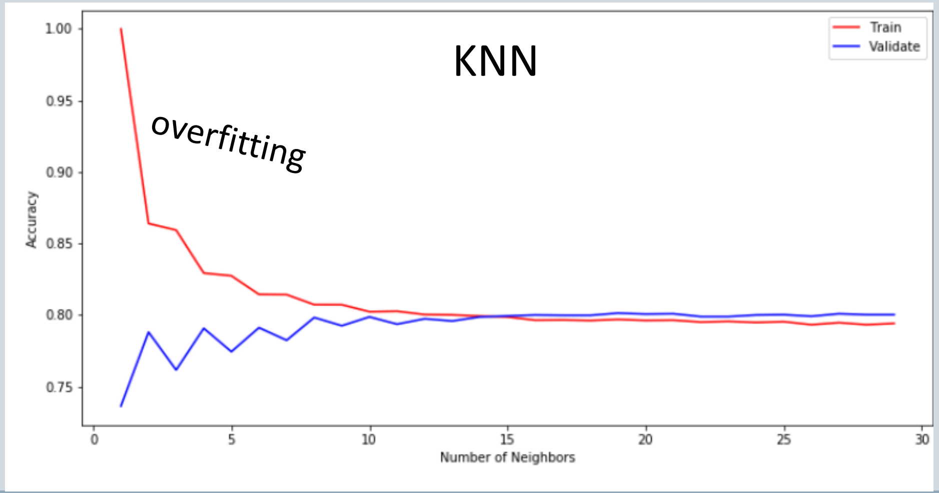
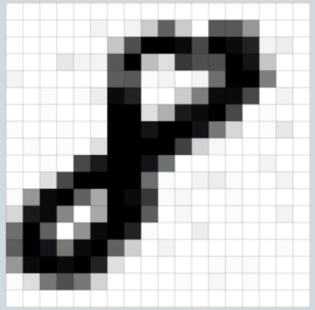
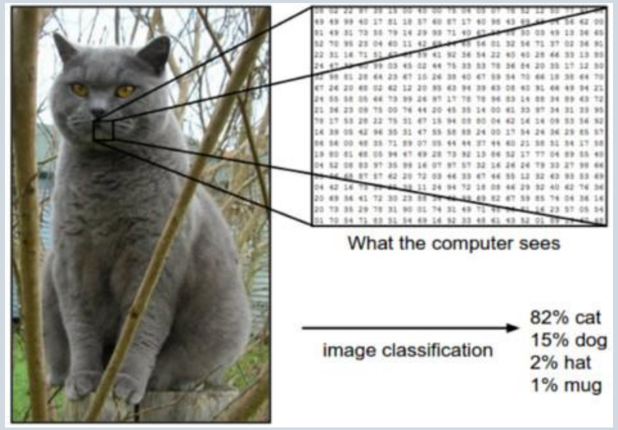
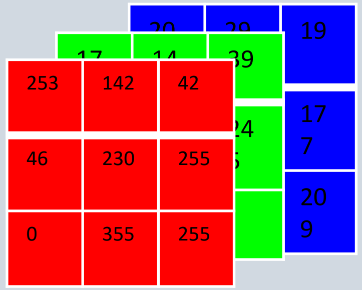


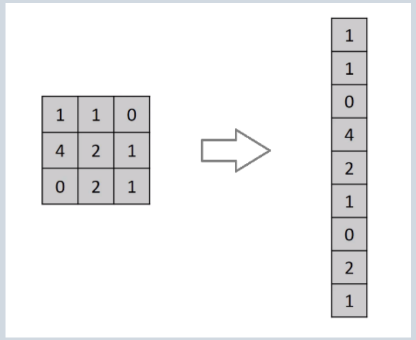
Image Classification



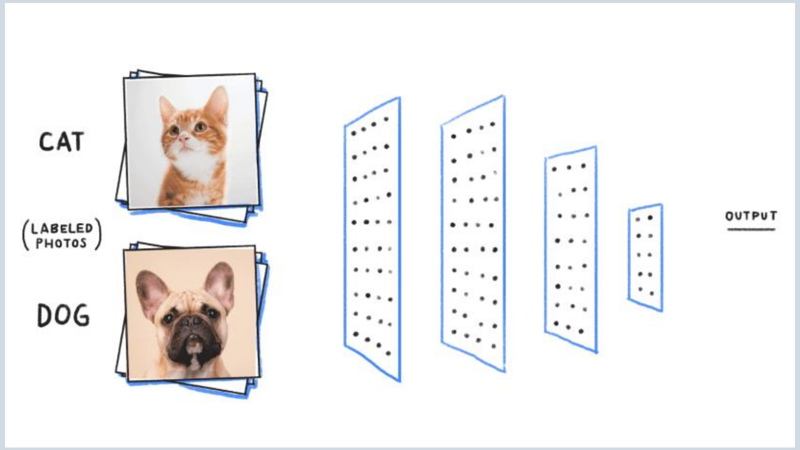
Source: https://cdn-images-1.medium.com/max/1200/1*zY1qFB9aFfZz66Yxxol2aw.gif







Source: https://www.kdnuggets.com/2017/08/convolutional-neural-networks-image-recognition.html



Source: https://becominghuman.ai/building-an-image-classifier-using-deep-learning-in-python-totally-from-a-beginners-perspective-be8dbaf22dd8

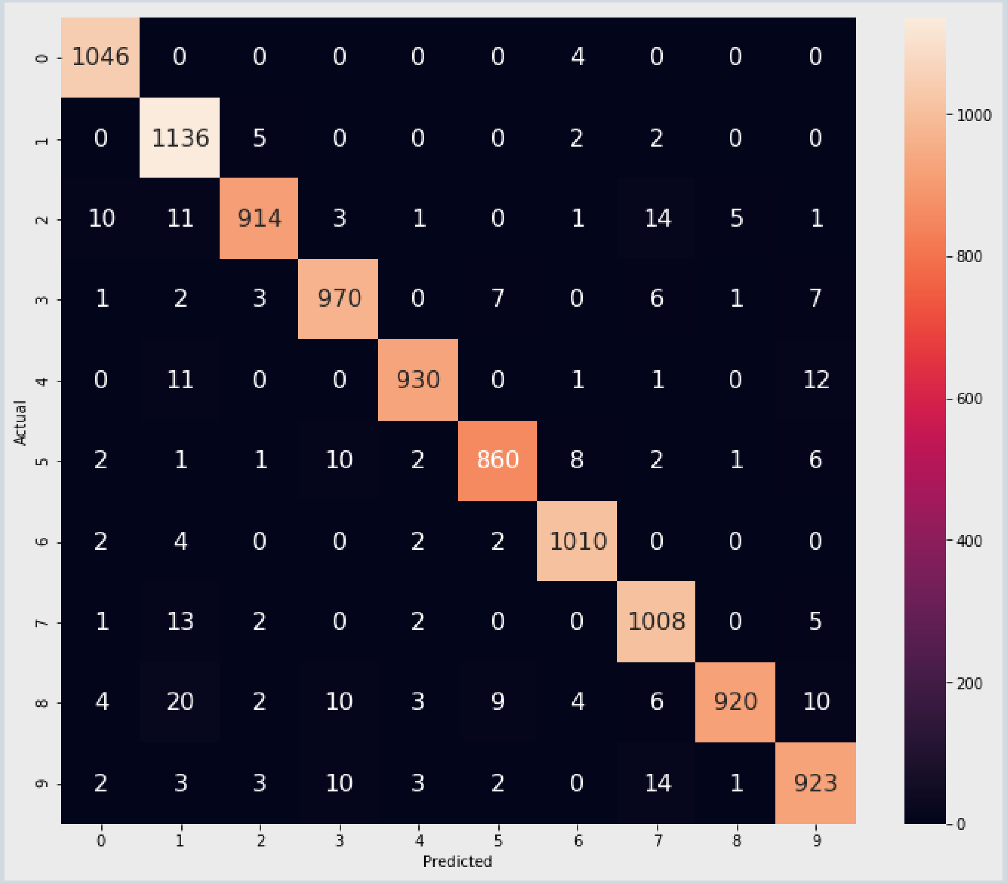
Image Classification

Classification_Models_Part_B.ipynb

Can we use classification to recognize handwritten digits?

Classified as

True Value

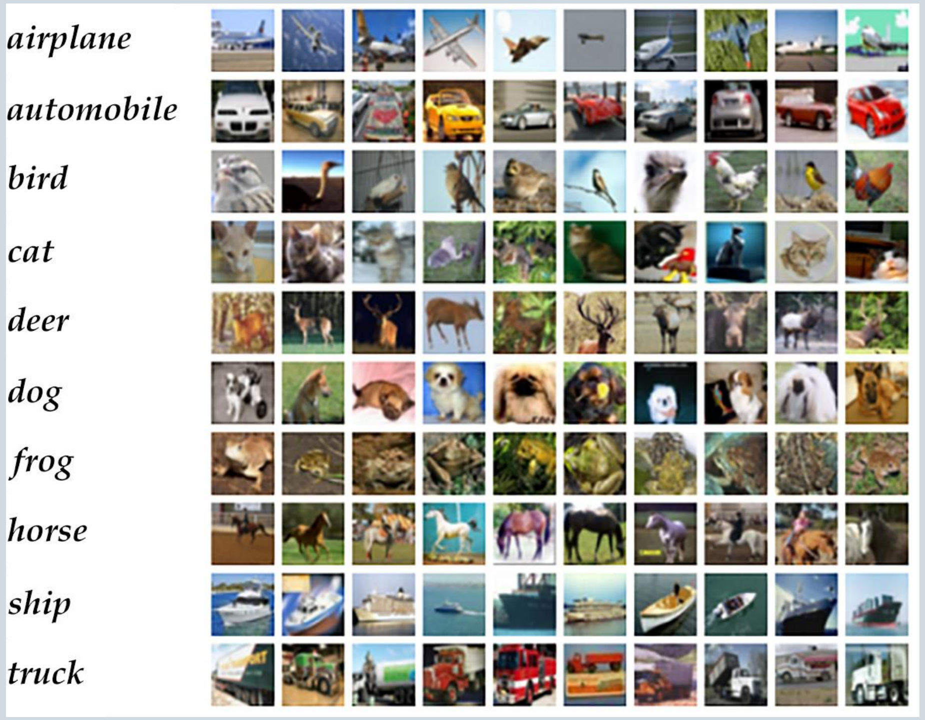


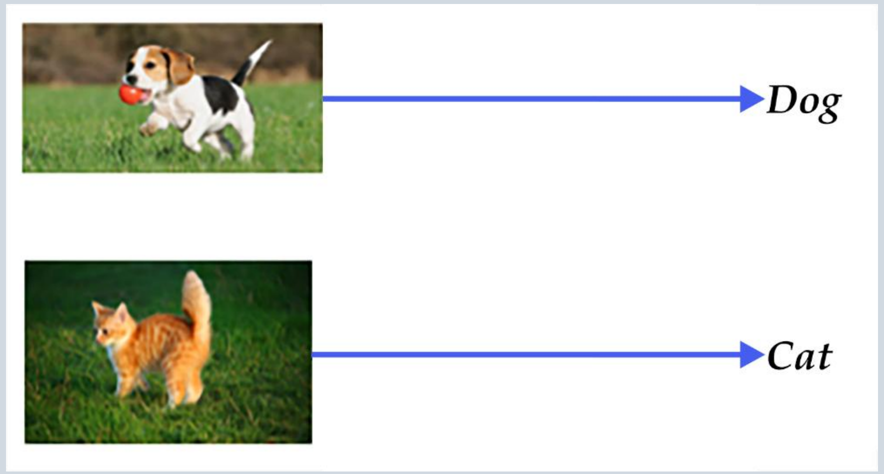
Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	1046	0	0	0	0	0	4	0	0	0
1	0	1136	5	0	0	0	2	2	0	0
2	10	11	914	3	1	0	1	14	5	1
3	1	2	3	970	0	7	0	6	1	7
4	0	11	0	0	930	0	1	1	0	12
5	2	1	1	10	2	860	8	2	1	6
6	2	4	0	0	2	2	1010	0	0	0
7	1	13	2	0	2	0	0	1008	0	5
8	4	20	2	10	3	9	4	6	920	10
9	2	3	3	10	3	2	0	14	1	923

Image Classification

Train Data Set:

- Millions of images from the web.
- Manually labeled categories.





THANK YOU FOR LISTENING

ZVI.BENAMI@MAIL.HUJI.AC.IL