Instance-based learning is a type of machine learning algorithm that makes predictions based on a set of known data points, called instances. Instead of building a model to represent the relationships between input features and output targets, instance-based learning stores all the training examples and uses them to make predictions on new data.

To make a prediction for a new input instance, the algorithm searches the training set for instances that are similar to the new input and then uses the outputs associated with those similar instances to compute a prediction for the new instance. The similarity between instances is usually measured using a distance metric such as Euclidean distance or cosine distance.

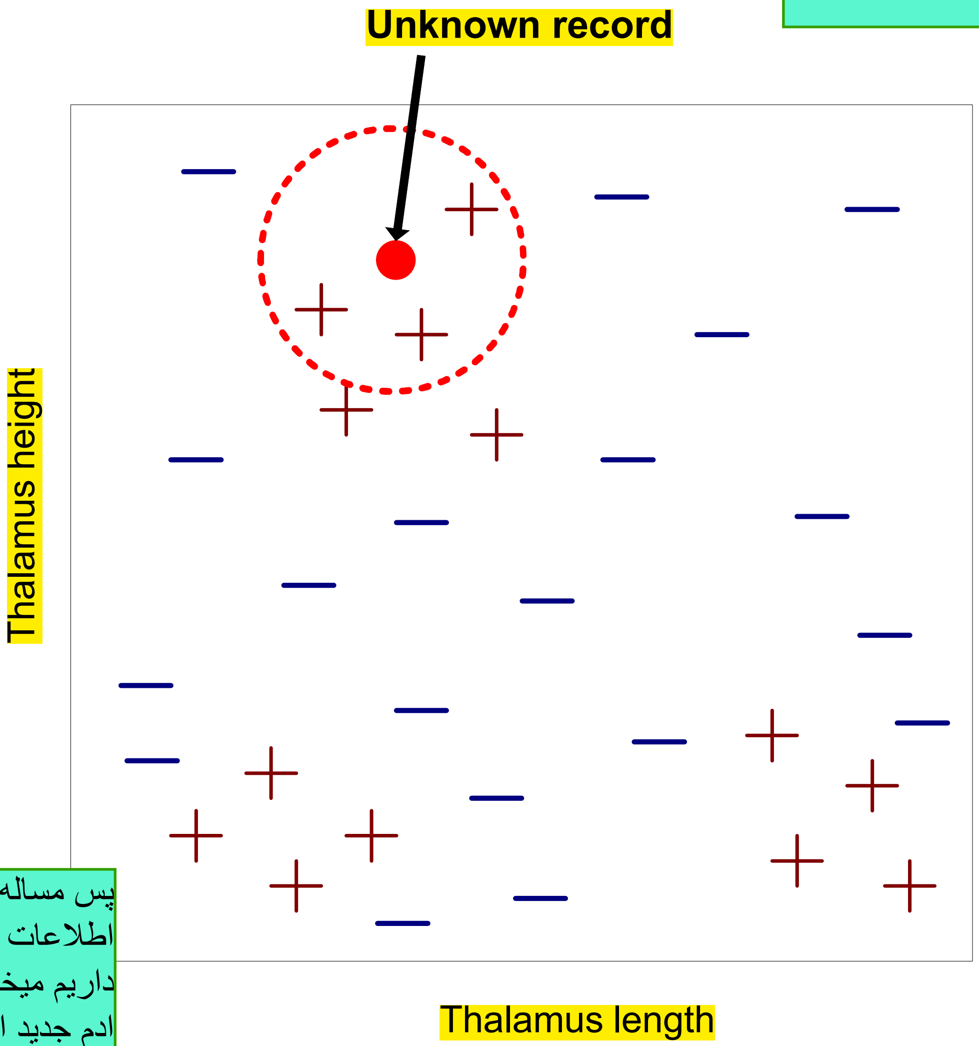
Instance-based learning is particularly useful when the relationship between inputs and outputs is complex or unknown, or when the training data is noisy or incomplete. Because it does not require the construction of an explicit model, instance-based learning can be more computationally efficient and easier to implement than other machine learning algorithms.

INSTANCE-BASED LEARNING

روش های مبتنی برنمونه ها
از داده های اموزشی مستقیم برای پیش بینی نمونه
تستی که ما برچسبش را نمیدانیم سعی میکنه کمک
بگیره
ر درخت تصمیم ما سعی میکردیم داده ها را به فضایی
مثل فضای مد ببریم و یک درختی بسازیم ولی اینجا به
کمک خود داده ها تصمیم میگیریم

Basic Idea_

پزشک ها میگن مثلا اگه اندازه ی طول تالاموس زیاد شد فلان بیماری را داره یا اگه ارتفاعش زیاد شد فلان بیماری
پزشک ها از ما یه سوالی پرسیدن:
ما در بیماری های مختلف، طول و ارتفاع تالاموس شون را اندازه گرفتیم و میدانیم که مثلا افرادی که طول تالاموسشون کمه و ارتفاعش زیاده آدم هایی هستند که بیماری دارند



https://en.wikipedia.org/wiki/File:Thalamus_small.gif

پس مساله ی ما اینطوریه که یه سری اطلاعات راجع به یه سری بیمارو ادم سالم داریم میخاهیم تصمیم گیری کنیم که وقتی یه ادم جدید اومد که ما طول و ارتفاع تالاموسش را داشتیم باید بش بگیم مریض است یا سالم؟
پس این فضا میشه داده های اموزش ما با روش های ماشین لرنینگ باید برچسب این ادم جدید را مشخص کنیم

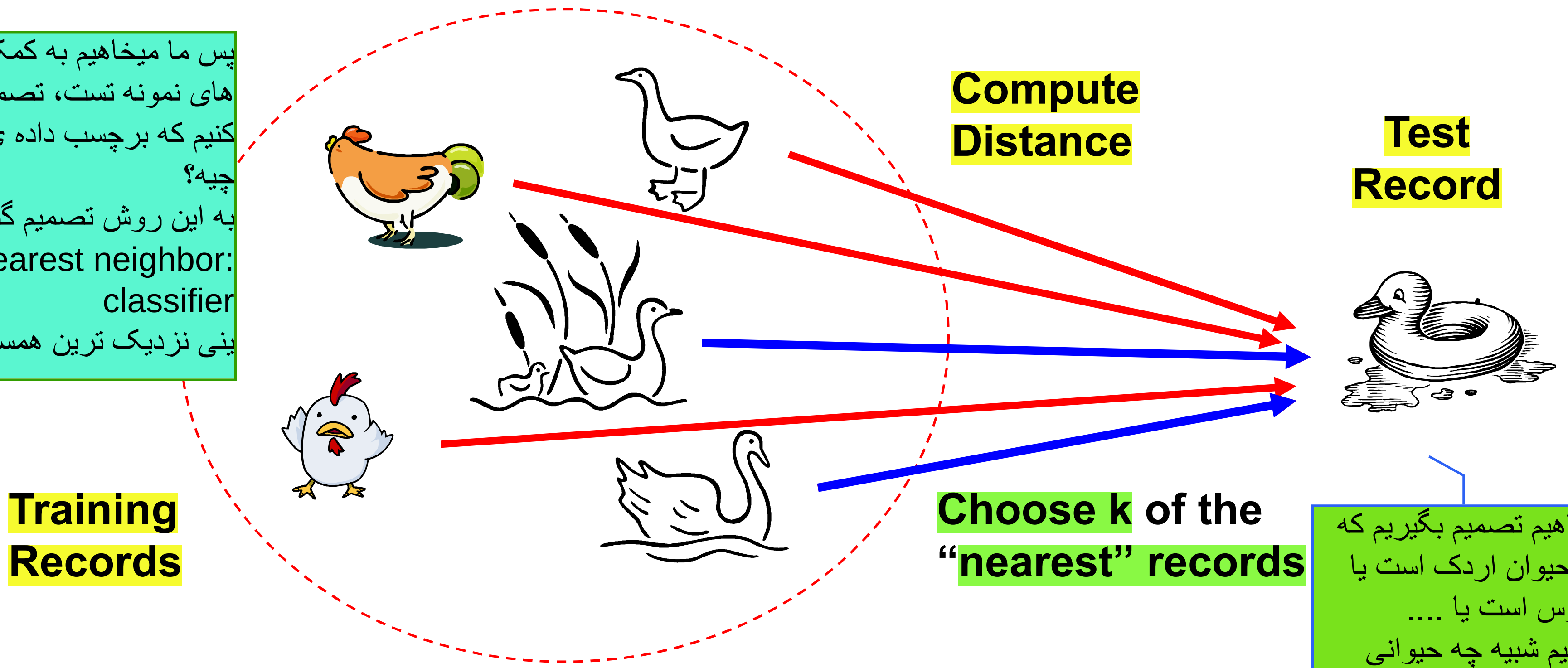
Nearest Neighbor Classifiers

این روش میگه نگاه کن اون نمونه ای که برای تست اوردی نزدیک هاش چه نمونه های دیگری هستند و از نمونه های نزدیکش کمک بگیر

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck

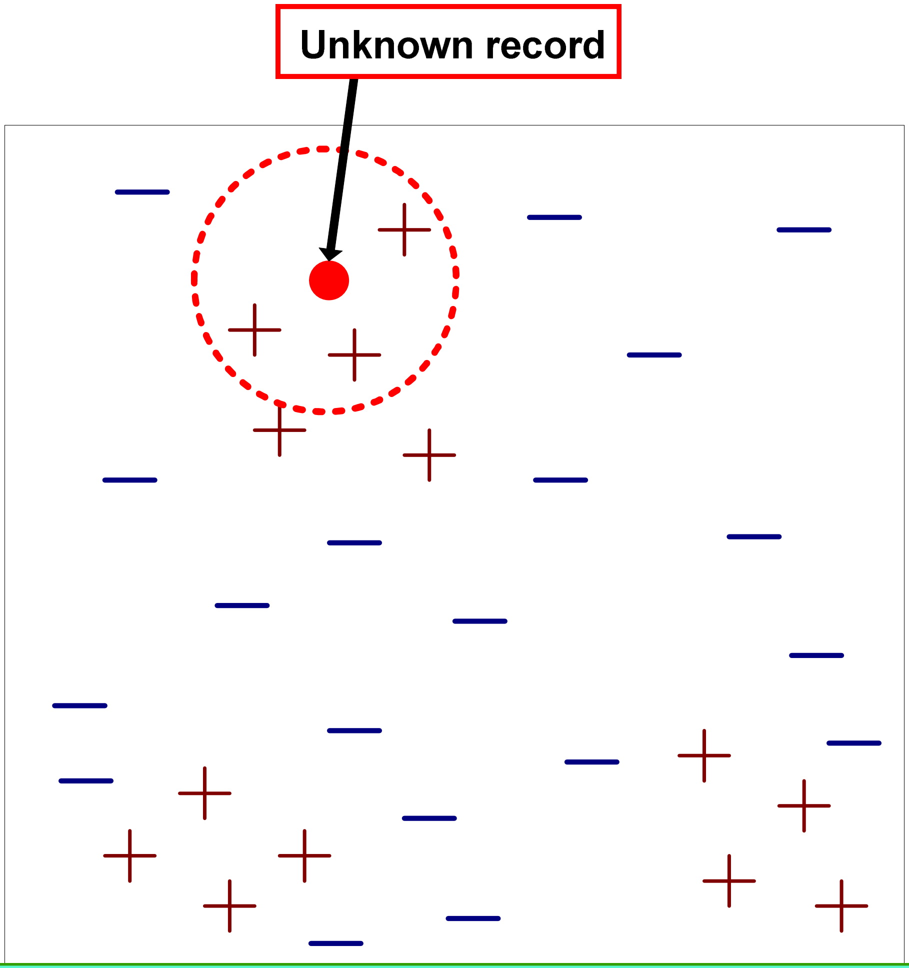
ایده پایه:
- اگر مثل اردک راه میرود، مثل اردک کوک میکند، احتمالاً اردک است

پس ما میخاهیم به کمک همسایه های نمونه تست، تصمیم گیری کنیم که برچسب داده ی جدید چیه؟
به این روش تصمیم گیری میگیم nearest neighbor: classifier
ینی نزدیک ترین همسایه ها



میخاهیم تصمیم بگیریم که این حیوان اردک است یا خروس است یا .... میبینیم شبیه چه حیوانی رفتار میکنه میگیم احتمالاً همونه

Nearest-Neighbor Classifiers



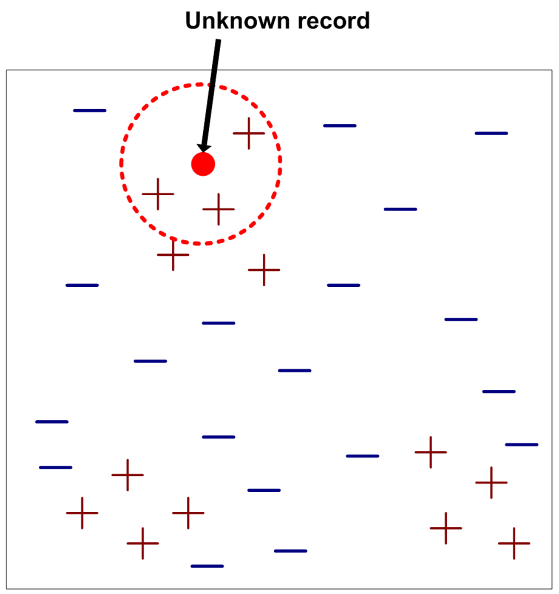
- Requires the following:
  - A set of labeled records(Feature space)
  - Proximity metric to compute distance/similarity between a pair of records
    - e.g., Euclidean distance
  - The value of *k*, the number of nearest neighbors to retrieve
  - A method for using class labels of K nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

چالش های این روش: ۱. چندتا از نزدیک ترین را درنظر بگیریم؟ مثلا ۲تا از نزدیک ترین یا ۳تا یا ۴تا؟ پس مقدار k را باید معلوم کنیم.
۲. الان فضا۲بعدی است و میشه نزدیک ها را دید اگه فضا ۱۰۰ بعدی بود باید چیکار کنیم؟ اینجا تالاموس ۲ بعد داشت وراحت بود اگه ۵ بعد بود چی؟ نمیشه به همین راحتی رسم کرد واز روی شکل تصمیم گرفت
پس باید روش های اندازه گیری نزدیک ترین ها را باید بهش یاد بدیم .
پس یه سری معیار برای فاصله باید داشته باشیم پس فاصله یابی هم یکی از چالش هاست
۳. feature space یه چالش دیگه است ینی از بین فیچرهایی که داریم کدوم ها را انتخاب کنیم؟ مثلا فرض کنید بعلاوه ی طول و ارتفاع، یه سری ویژگی های دیگه هم از مساله بمون داده باشند پس انتخاب فیچرها برای تصمیم گیری مهم است

How to Determine the class label of a Test Sample?

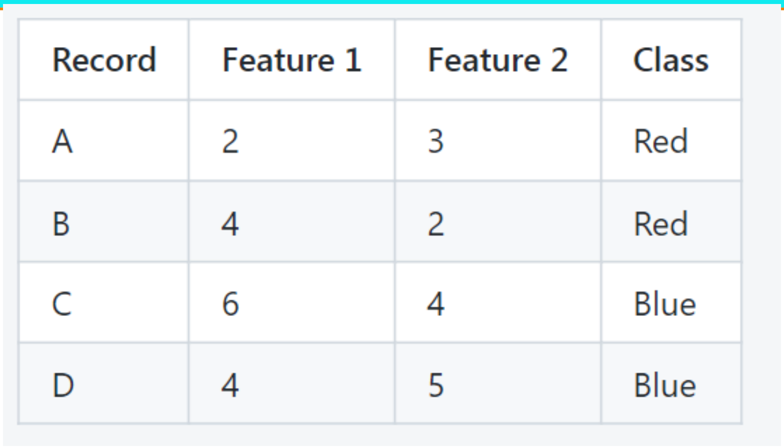
-
- Take the majority vote of class labels among the k-nearest neighbors
- Weight the vote according to distance
  - weight factor, $w = 1/d^2$

فرض کنید ما نزدیک ترین همسایه هامون را میدانیم کیا هستن حالا باید براساس این همسایه ها راجع به نقطه ی ناشناخته تصمیم گیری کنیم.
یه راه اینه ببینیم توی این فاصله همسایه ها چه کلاس و برچسبی بیشتر از همه تکرار شده یعنی رای گیری کنیم وببینیم کدوم کلاس برنده میشه.
پس برچسب اون داده جدیده هم میذاریم اون کلاسی که برنده شده.
روش هوشمندانه تر اینه که درسته داریم به نزدیک ترین ها نگاه میکنیم ولی بهتره به صورت وزن دار نگاه کنیم.
مثلا با یک وزنی از فاصله رای گیری کنیم از همسایه ها



Sure, let's walk through an example of finding the k-Nearest Neighbors (k-NN) for a test instance using the Euclidean distance metric.

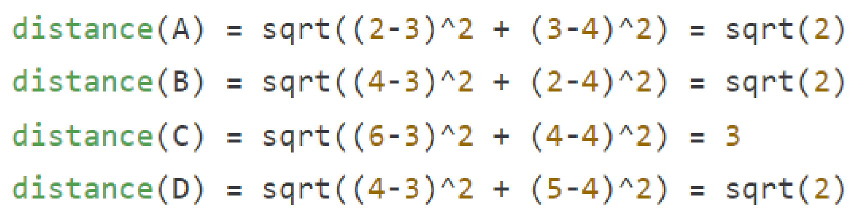
Suppose we have a dataset with the following records:



Record	Feature 1	Feature 2	Class
A	2	3	Red
B	4	2	Red
C	6	4	Blue
D	4	5	Blue

We want to find the k=3 nearest neighbors of a test instance with features (3, 4). Here are the steps we could follow:

Compute the Euclidean distance between the test instance and each training instance. For example, the distances between the test instance (3, 4) and the training instances are:



```
distance(A) = sqrt((2-3)^2 + (3-4)^2) = sqrt(2)
distance(B) = sqrt((4-3)^2 + (2-4)^2) = sqrt(2)
distance(C) = sqrt((6-3)^2 + (4-4)^2) = 3
distance(D) = sqrt((4-3)^2 + (5-4)^2) = sqrt(2)
```

Sort the training instances by their distances to the test instance, in ascending order.
In this example, the sorted list of training instances and their distances to the test instance are:

(A, sqrt(2))
(B, sqrt(2))
(D, sqrt(2))
(C, 3)
Select the k nearest neighbors from the sorted list.
Since k=3, we select the first three instances from the sorted list:

(A, sqrt(2))
(B, sqrt(2))
(D, sqrt(2))
Determine the class of the test instance by majority vote among the k nearest neighbors.
In this case, all three nearest neighbors belong to the Red class, so we predict that the test instance also belongs to the Red class.

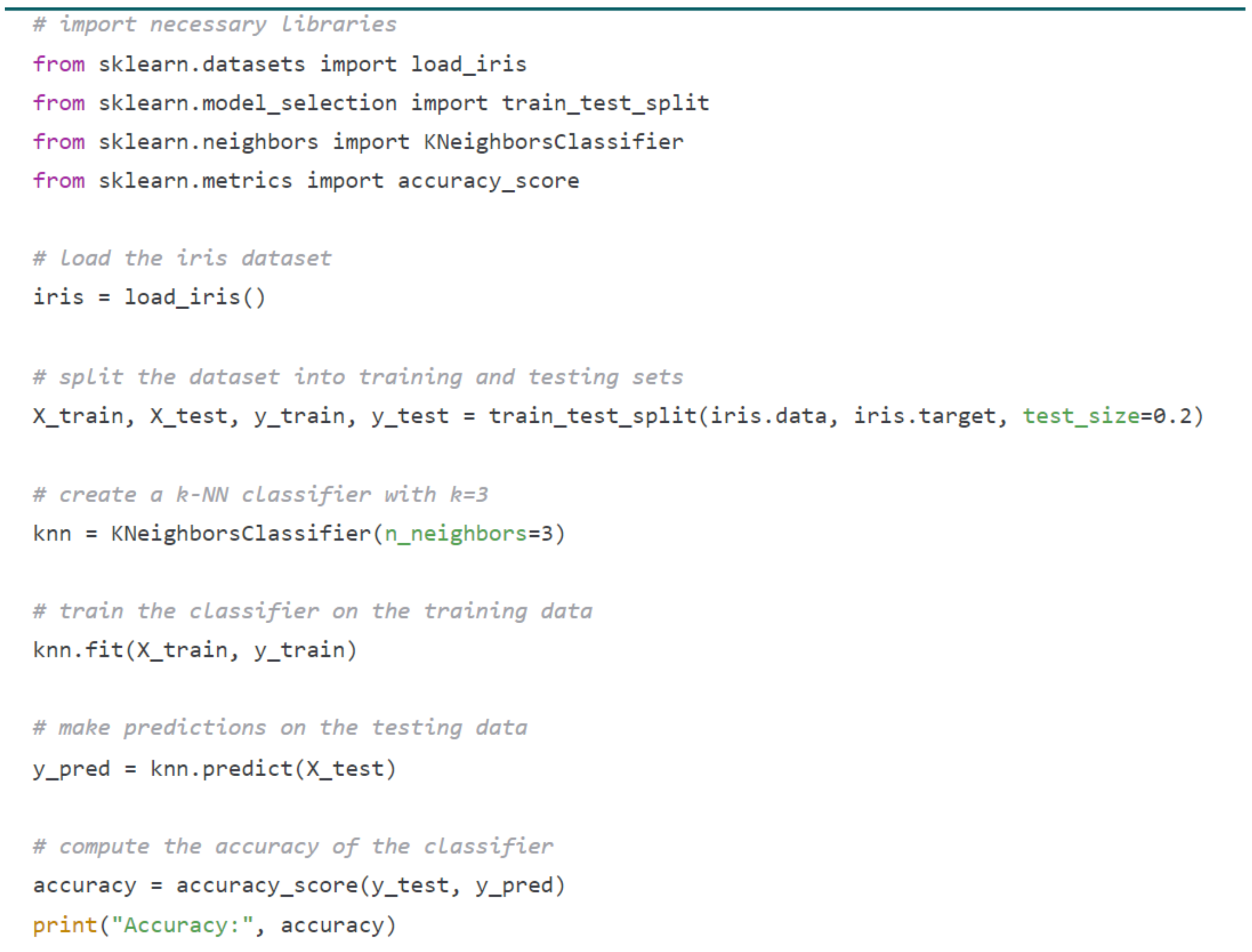
Therefore, using the k-NN algorithm with k=3 and the Euclidean distance metric, we predict that the class of the test instance (3, 4) is Red.

Sure, let's use the famous Iris dataset as an example for k-Nearest-Neighbor (k-NN) classification.

The Iris dataset contains measurements of the sepal length, sepal width, petal length, and petal width of 150 iris flowers belonging to three different species: Setosa, Versicolor, and Virginica. We will use these measurements to train a k-NN classifier that can predict the species of new iris flowers based on their measurements.

Here are the steps we would follow:

Load the dataset into memory.
Split the dataset into training and testing sets.
Choose a value for k (the number of nearest neighbors to consider).
For each test instance, compute the Euclidean distance between its feature vector and the feature vectors of all training instances.
Select the k instances with the smallest distances to the test instance.
Determine the class of the test instance by majority vote among the k nearest neighbors.
Repeat steps 4-6 for all test instances.
Compute the accuracy of the classifier by comparing its predictions with the true labels in the testing set.
Here is some sample Python code that implements this algorithm using scikit-learn library:



```python
# import necessary libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# load the iris dataset
iris = load_iris()

# split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.2)

# create a k-NN classifier with k=3
knn = KNeighborsClassifier(n_neighbors=3)

# train the classifier on the training data
knn.fit(X_train, y_train)

# make predictions on the testing data
y_pred = knn.predict(X_test)

# compute the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Handling missing values in k-NN can be challenging, but there are several approaches that can be used.

One approach is to simply remove any data points that have missing values from both the training and test sets. This is the simplest approach, but it can result in significant loss of data if many data points have missing values.

Another approach is to impute the missing values with some estimate before performing k-NN. One way to do this is to use mean or median imputation, where the missing value is replaced with the mean or median value of the feature in the training set. Another way is to use regression imputation, where a regression model is trained on the non-missing values of the feature and used to predict the missing values.

When imputing missing values, it is important to use only the training set to estimate the missing values, and not the test set. This is because using information from the test set during training can lead to overfitting and poor generalization performance.

Once the missing values have been imputed, k-NN can be applied as usual. However, it's important to keep in mind that imputing missing values can introduce bias into the model, so it's important to carefully consider the potential impact on the results.

To normalize a vector to have unit length, you need to divide each component of the vector by the magnitude of the vector. The magnitude of a vector is calculated as the square root of the sum of the squares of its components.

Here's an example: Suppose you have a vector v = [3, 4]. To normalize this vector, first you calculate its magnitude:

|v| = sqrt(3^2 + 4^2) = 5

Then, you divide each component of the vector by its magnitude:

v_normalized = v/|v| = [3/5, 4/5]

Now, the resulting normalized vector has unit length, which means its magnitude is equal to 1.
In mathematics, a vector is said to have unit length if its magnitude or length is equal to 1.

For example, if we have a two-dimensional vector v = [3, 4], as I mentioned before, the magnitude of this vector is:

|v| = sqrt(3^2 + 4^2) = 5

To normalize this vector, we divide each component by its magnitude to get the normalized vector v_normalized = [3/5, 4/5]. The magnitude of the normalized vector is:

|v_normalized| = sqrt((3/5)^2 + (4/5)^2) = 1

So, the normalized vector has unit length since its magnitude is equal to 1.

This concept of unit length is important in many areas of mathematics and science, including physics, engineering, and computer graphics, where it is used to represent directions and orientations of objects.

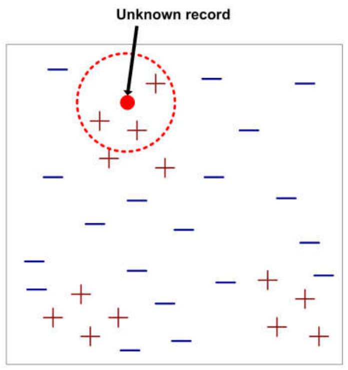
انتخاب معیار مجاورت اهمیت دارد

Choice of proximity measure matters

- For documents, cosine is better than correlation or Euclidean

1 1 1 1 1 1 1 1 1 1 1 0		0 0 0 0 0 0 0 0 0 0 0 1	
0 1 1 1 1 1 1 1 1 1 1 1	vs	1 0 0 0 0 0 0 0 0 0 0 0	

Euclidean distance = 1.4142 for both pairs, but the cosine similarity measure has different values for these pairs.



معیارهای اندازه گیری فاصله وابسته به مساله است.
انتخاب های ما مثل model selection است که قراره بین فرضیات مختلف یکی را انتخاب کنیم
وقتی مدل های مختلف داریم کدوم را انتخاب میکنیم؟ از داده ها ایده میگیریم
روش اصلی: یه تعدادی از داده ها را جدا میکنیم و برای تصمیم گیری معیار فاصله ازشون استفاده میکنیم

here's an example of k-NN with cosine proximity:

Suppose we have a dataset of customer reviews for a set of products, where each review is represented as a vector of term frequencies. We want to use k-NN to classify new reviews as either positive or negative based on their similarity to the reviews in the training set.

To measure similarity between reviews, we can use cosine proximity, which measures the cosine of the angle between two vectors in a high-dimensional space. The cosine similarity ranges from -1 to 1, where -1 indicates opposite directions (maximal dissimilarity), 0 indicates orthogonality (no similarity), and 1 indicates parallel directions (maximal similarity).

Here's how k-NN with cosine proximity works:

Calculate the cosine similarity between the query review and each review in the training set.
Select the k reviews with the highest cosine similarity to the query review.
Compute the class label for the query review based on the majority vote of the k nearest neighbors.
Assign the query review to the predicted class.
For example, suppose we have a training set with three reviews:

Review 1: "This product is great and works well."
Review 2: "I don't like this product, it's too expensive."
Review 3: "This product is okay, but I've had better."
And suppose we want to classify the following query review:
Query Review: "I love this product, it's amazing!"We can represent each review as a vector of term frequencies, for example:
Review 1: [1, 1, 1, 1, 1, 0, 0, 0]
Review 2: [0, 0, 0, 1, 0, 1, 1, 1]
Review 3: [1, 0, 1, 1, 0, 0, 1, 0]
Query Review: [1, 0, 0, 1, 0, 0, 0, 1]
To use k-NN with cosine proximity, we can first normalize the vectors to have unit length:
Review 1: [0.41, 0.41, 0.41, 0.41, 0.41, 0.00, 0.00, 0.00]
Review 2: [0.00, 0.00, 0.00, 0.33, 0.00, 0.33, 0.33, 0.33]
Review 3: [0.53, 0.00, 0.53, 0.53, 0.00, 0.00, 0.53, 0.00]
Query Review: [0.71, 0.00, 0.00, 0.71, 0.00, 0.00, 0.00, 0.71]
Then we can calculate the cosine similarity between the query review and each training review:
Cosine Similarity:
Review 1: 0.55
Review 2: 0.13
Review 3: 0.49
Suppose we choose k=2. The two nearest neighbors to the query review are Review 1 and Review 3, both of which are positive. Thus, we predict that the query review is positive.

In the k-nearest neighbor (k-NN) algorithm, time series data can be used as input. Standardization is a preprocessing step that involves transforming the data to have a mean of 0 and a standard deviation of 1. This is also called z-score normalization.
In the context of the given statement, it means that before applying the k-NN algorithm to time series data, the time series values are first standardized by subtracting the mean value from each value and then dividing the result by the standard deviation of the values. This results in all the values being centered around zero with a consistent range. The purpose of doing this is to ensure that the different features or variables in the data are on a similar scale, so that they contribute equally to the distance metric used to find the k-nearest neighbors.

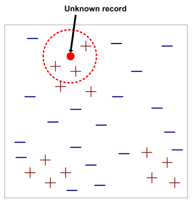
Nearest Neighbor Classification

اسکیل میکنیم تا یک واحد افزایش یه اتریبیوتی معادل یه واحد افزایش
دیگری بشه یا متناسب بشه
تقسیم بر کوچکترین یا بزرگترین مقدار کنیم
تقسیم بر میانگین کنیم
در مسائل تایم سری معمولا تقسیم بر میانگین یا انحراف معیار میکنند

مثال:
- قد یک فرد ممکن است از 1.5 متر تا 1.8 متر متغیر باشد
- وزن یک فرد ممکن است از 90 پوند تا 300 پوند متغیر باشد
- درآمد یک فرد ممکن است از 10 هزار دلار تا 1 میلیون دلار
متغیر باشد

پیش پردازش داده ها اغلب مورد نیاز است
- ممکن است برای جلوگیری از تسلط معیارهای فاصله
توسط یکی از ویژگیها، ویژگیها باید مقیاس شوند

- **Data preprocessing is often required**
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
    - Example:
      - height of a person may vary from 1.5m to 1.8m
      - weight of a person may vary from 90lb to 300lb
      - income of a person may vary from \$10K to \$1M



سری های زمانی معمولاً به گونه ای استاندارد می شوند که 0 به معنای انحراف معیار 1 باشد.

  - Time series are often standardized to have 0 means a standard deviation of 1

فیچرهامون متفاوت هستند مثلا یه فیچر جنسش طول است یکی سن و یکی وزن باشه یا مقیاس هاشون ممکن است فرق کنه مثلا میگیم تالاموس چندمیلی متر است ولی مثلا اگه بخاهیم راجع به شدت رنگ بافتش حرف بزنیم میگیم مثلا ۰ تا ۲۵۵ است پس مقیاس فضایی که داریم بهش نگاه میکنیم کوچک و بزرگ است و باید در یک مقیاس بهش نگاه کنیم

وقتی داریم نزدیک ترین فاصله ها را حساب میکنیم این مقیاس های متفاوت کار را خراب میکنند. مثلا فاصله ی اقلیدوسی که میگیریم داریم به همه ی ویژگی هامون به یک شکل نگاه میکنیم در حالی که عددبزرگ ها توی فرمول فاصله موثرتر میشوند پس مقیاسشون میکنیم که جنسشون یکی شه

Yes, when we use the z-score normalization method to standardize a dataset, the resulting dataset will have a mean of 0. This is because the z-score formula involves subtracting the mean of the dataset from each data point, which centers the distribution around 0.

The formula for calculating the z-score of a data point x in a dataset with mean μ and standard deviation σ is:
$z = (x - \mu) / \sigma$
By subtracting the mean from each data point in the dataset, the new mean of the dataset becomes 0.
Standardizing the data in this way allows us to compare and analyze different variables or features on a similar scale, regardless of their original units or scales.
When we apply the z-score normalization method to standardize a dataset, the resulting dataset will have a standard deviation of 1. This is because the z-score formula involves dividing each data point by the standard deviation of the dataset, which scales the data to have a standard deviation of 1.
By dividing each data point in the dataset by the standard deviation, the new standard deviation of the dataset becomes 1. Standardizing the data in this way allows us to compare and analyze different variables or features on a similar scale, regardless of their original units or scales.
here is an example to demonstrate how applying the z-score normalization method results in a dataset with mean=0 and standard deviation=1:

Suppose we have a dataset of 5 numbers: {4, 6, 8, 10, 12}.

The mean of this dataset is:

$\mu = (4 + 6 + 8 + 10 + 12) / 5 = 8$

The standard deviation of this dataset is:

σ = sqrt(((4-8)^2 + (6-8)^2 + (8-8)^2 + (10-8)^2 + (12-8)^2) / 5) = 2.83

Now, let's apply the z-score normalization method to standardize this dataset:

z-score for 4: $(4 - 8)/2.83 = -1.41$
z-score for 6: $(6 - 8)/2.83 = -0.71$
z-score for 8: $(8 - 8)/2.83 = 0$
z-score for 10: $(10 - 8)/2.83 = 0.71$
z-score for 12: $(12 - 8)/2.83 = 1.41$

As you can see, after standardizing the dataset, the mean is now 0 and the standard deviation is now 1. This means that the dataset is now centered around 0 and scaled to have a consistent range, making it easier to compare and analyze the different values in the dataset.

The choice of the value of k in the k-NN algorithm can have a significant impact on the accuracy of the classifier. A small value of k (e.g., k=1) may result in overfitting, where the model is too sensitive to noise or outliers in the data, while a large value of k may result in underfitting, where the model oversimplifies the classification decision boundary and may miss important patterns in the data.

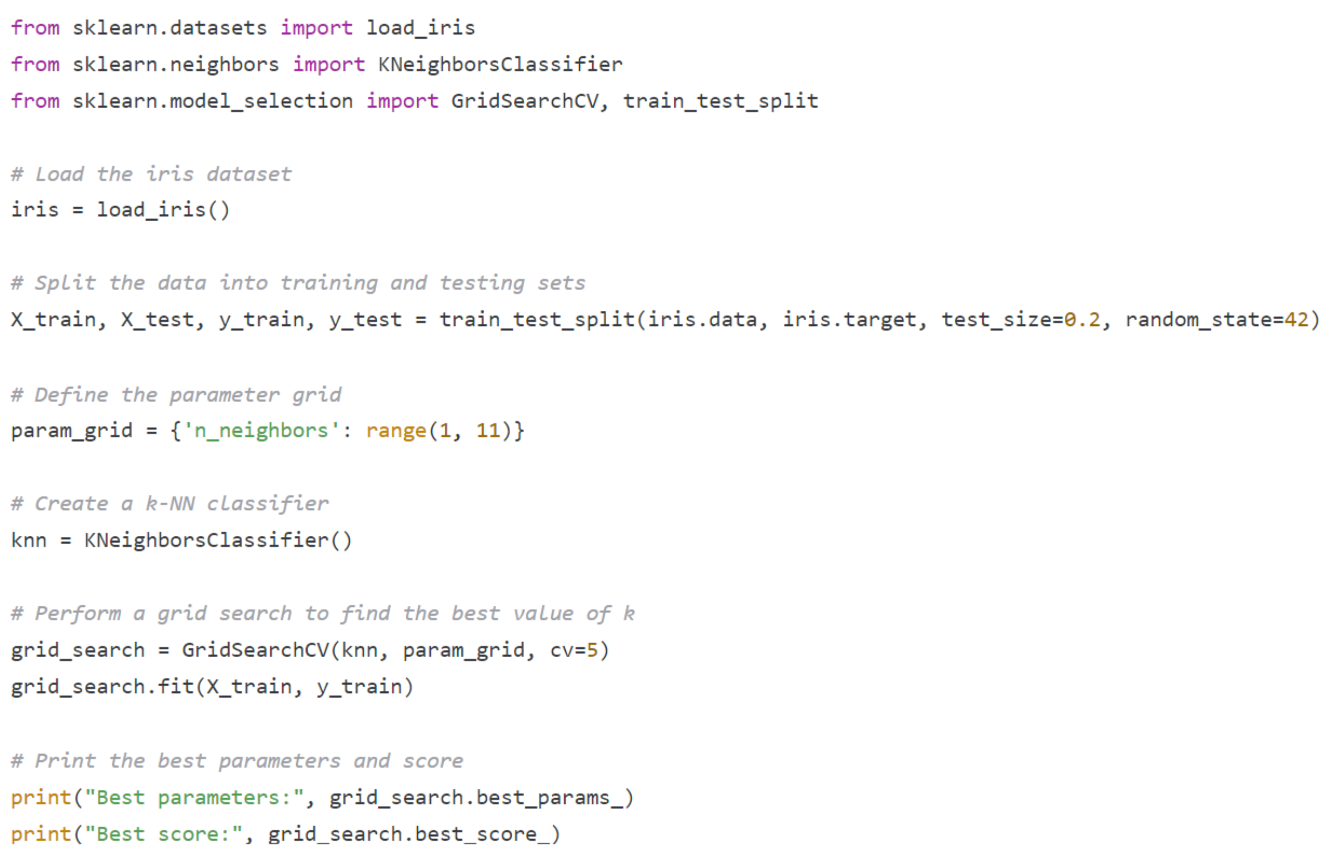
To find the best value of k for the k-NN classifier, we typically use a technique called cross-validation. The basic idea behind cross-validation is to split the available labeled data into two sets: a training set and a validation set. The model is trained on the training set, and its performance is evaluated on the validation set. This process is repeated several times with different splits of the data, and the average performance over all the splits is used as an estimate of the model's generalization performance.

Here are the general steps to find the best k for a k-NN classifier:

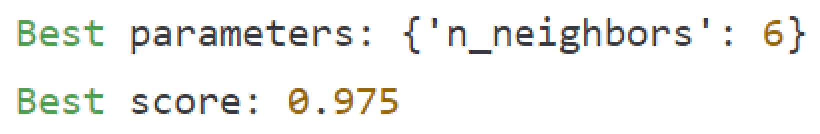
Split the labeled data into a training set and a validation set.
Choose a range of values for k to be tested.
For each value of k, train a k-NN classifier on the training set using that value of k.
Evaluate the performance of the classifier on the validation set using an appropriate metric, such as accuracy, precision, recall, F1 score, or area under the receiver operating characteristic (ROC) curve.
Repeat steps 1-4 several times with different splits of the data to get an estimate of the classifier's performance on unseen data.
Select the value of k that gives the best performance on the validation set.
It is important to note that the optimal value of k may vary depending on the specific dataset and problem being addressed, so it is important to choose a range of values for k that covers a broad range of possibilities in order to find the best value for the given problem.

Here's an example of how to use cross-validation to find the best value of k for a k-nearest neighbors classifier:
The GridSearchCV object automatically fits the model with each combination of hyperparameters in the parameter grid, and returns the best combination based on the validation score.

After running the code, you should see the output showing the best parameters (i.e., the value of k that maximized the validation score) and the corresponding score.



```python
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV, train_test_split

# Load the iris dataset
iris = load_iris()

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.2, random_state=42)

# Define the parameter grid
param_grid = {'n_neighbors': range(1, 11)}

# Create a k-NN classifier
knn = KNeighborsClassifier()

# Perform a grid search to find the best value of k
grid_search = GridSearchCV(knn, param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Print the best parameters and score
print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)
```



```
Best parameters: {'n_neighbors': 6}
Best score: 0.975
```

In this example, we used the Iris dataset and split it into training and testing sets with a 80/20 ratio. The best value of k was found to be 6 based on a 5-fold cross-validation procedure, with a corresponding validation score of 0.975.

Note that your actual results may vary slightly due to randomness in the train-test split and cross-validation folds.

Nearest Neighbor Classification…

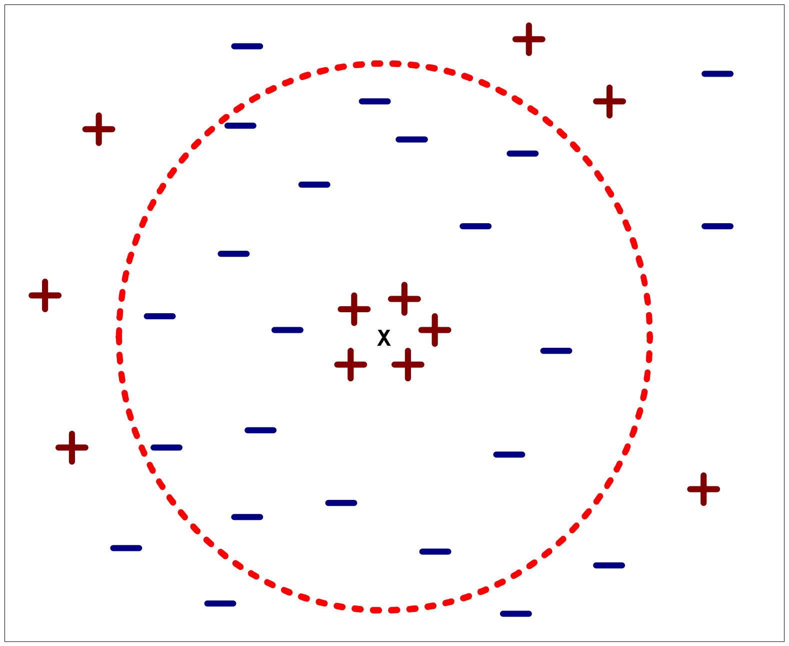
- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes

سوال:
چندتا همسایه باید درنظر بگیریم؟

اطلاعات کمی برای قضاوت داریم استفاده میکنیم به نویز حساس میشه و ممکنه غلط قضاوت کنیم

به جای ۱۰ تا همسایه مثلا ۲۰۰تا همسایه رو درنظر بگیریم ممکنه همسایه ها به مورد ما مربوط نباشن دوباره قضاوتمون اشتباه میشه
پس یه ترید افی هست برای تعدادی که انتخاب میشه

گه k را کوچک درنظر بگیریم یعنی تعدادهمسایه ها را کم بگیریم، ینی اطلاعاتمون برای قضاوت کردن کمه
مثال: دادگاه های امریکا میگن شرایط این پرونده شبیه کدوم یکی از پرونده های قبلی است؟



Nearest Neighbor Classification…

نحوه ی تصمیم گیری چطوری میشه در نواحی مختلف؟ میشه یه قانونی تعریف کرد؟ وقتی که یک همسایه رو درنظر بگیریم فقط

- Nearest neighbor classifiers are local classifiers

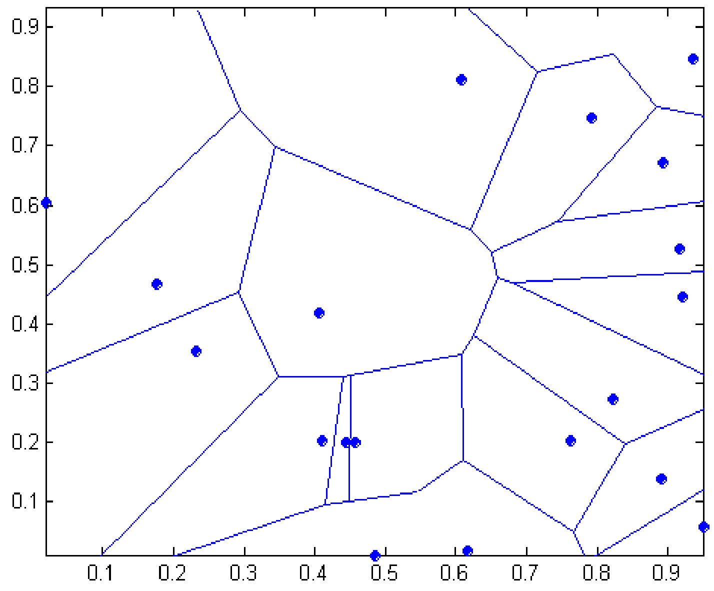
- They can produce decision boundaries of arbitrary shapes.

The statement "nearest neighbor classifiers are local classifiers" means that the k-nearest neighbor (k-NN) algorithm makes predictions based on local information, specifically by comparing each new instance to its k nearest neighbors in the training set.
In other words, the k-NN classifier does not try to build a global model of the data distribution or decision boundary. Instead, it simply stores the entire training dataset and makes predictions for new instances based on the class labels of the k nearest training instances, which are typically measured using a distance metric such as Euclidean distance.

Because the k-NN algorithm operates locally, it can be sensitive to the structure of the data and the choice of k. Specifically, if the data has complex structure or nonlinear relationships, a small value of k may be more appropriate to capture the local structure, while a larger value of k may be more appropriate for smoother or simpler data distributions.

k=1
ینی یه همسایه انتخاب کنیم

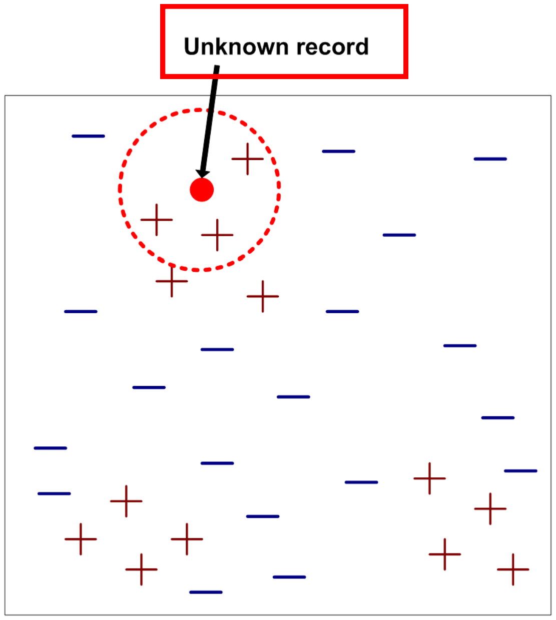
1-nn decision boundary is a Voronoi Diagram



Nearest Neighbor Classification…

- How to handle missing values in training and test sets?

یه سری از فیچرها را نداریم



در مسائل کلسیفیکیشن ممکن است با داده هایی روبرو باشیم که همه ی فیچر هاشون را ممکنه نداشته باشیم
مثلا یه تعداد بیمار هستند که فقط عرض تالاموسشون را داریم و طولش را نداریم یا اندازه گیری نکردن یا کردن و فهمیدن اشتباه اندازه گرفتند
پس با یه شرایطی روبرو هستیم که یه سری از مقادیر را نداریم
سوال: آیا باید بیخیال اون داده ها بشیم یا مدلمون میتونه یه کاری براشون بکنه؟
راه حل:فاصله ها را روی بقیه ی ویژگی ها بسنجیم مثل اینه که از یک بعد خاص به داده ها نگاه کنیم
این باعث میشه به یه نحوی یه جوابی بگیریم و بی جواب نمونیم که ینی داده هامون هدر نرفته

مثلا اگه طول های تالاموس را نداریم فقط با عرض ها فاصله سنجی را انجام بدیم
در محاسبه ی نزدیکترین ها از طول مثلا استفاده نکن (برای کل داده ها!!!!)

معیار فاصله حساب کردن را میبریم روی ویژگی هایی که missing valueندارند

Nearest Neighbor Classification…

- **How to handle missing values in training and test sets?**
  - Proximity computations normally require the presence of all attributes
  - Some approaches use the subset of attributes present in two instances
    - This may not produce good results since it effectively uses different proximity measures for each pair of instances
    - Thus, proximities are not comparable

چگونه مقادیر از دست رفته در مجموعه های آموزشی و تست را مدیریت کنیم؟
- محاسبات مجاورت معمولاً به وجود همه صفات نیاز دارند
- برخی از رویکردها از زیرمجموعه ویژگی های موجود در دو مورد استفاده می کنند
این ممکن است نتایج خوبی ایجاد نکند زیرا به طور موثر از معیارهای مجاورت متفاوتی برای هر جفت نمونه استفاده می کند.
بنابراین، مجاورت ها قابل مقایسه نیستند

K-NN Classificiers…

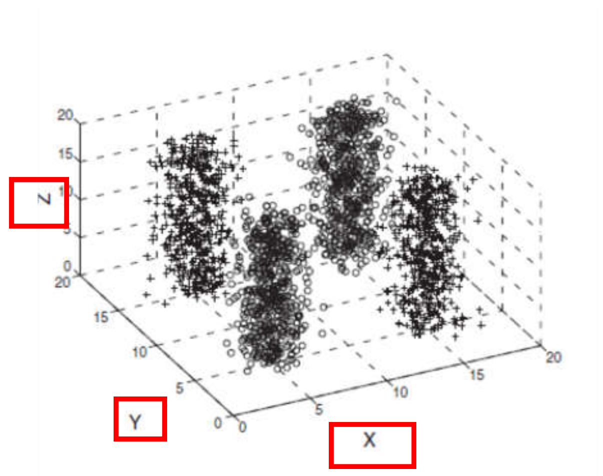
Handling Irrelevant and Redundant Attributes

مثلا اینجا سه بعد x,y,z داریم و میدانیم که یکی از این بعدها نویز داره اگه فاصله را روی هر۳تا بعد تعریف کنیم، فاصله ی ما همیشه به خاطر وجود این بعد نویزی یک مقدار ثابتی داره و یه بایاسی داره که با مقداروقعی نمیخونه چون ما از این فاصله ها برای تصمیم گیری کلاس ها میخاهیم استفاده نیم بهتره شاید این بعد را حذف کنیم و فاصله را فقط برحسب x,y بسنجیم

- Irrelevant attributes add noise to the proximity measure
- Redundant attributes bias the proximity measure towards certain attributes

زمانی که با ویژگی های نامرتبط و ویژگی های تکراری سروکار داشته باشیم

معیار فاصله حساب کردن را میبریم روی مقادیری که missing value ندارند



(a) Three-dimensional data with attributes X, Y, and Z.

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}$$

محاسبه ی فاصله روی n بعد

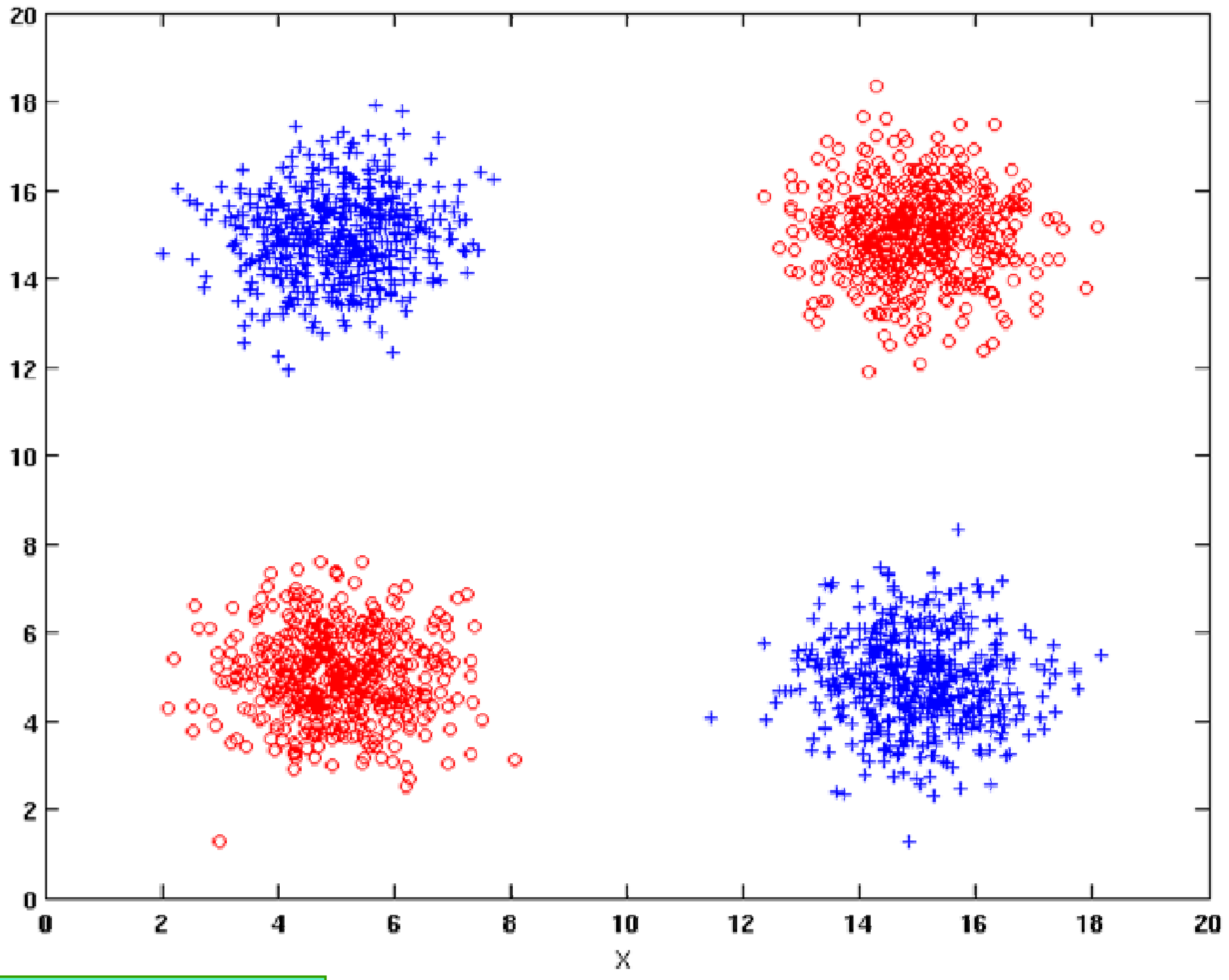
اگه نویز روی یکی از بعدهای داده ها باشه یا یه بعدی نامرتبط باشه چه اتفاقی میفته؟ باعث میشه نتیجه بایاس شه و به یک مقداری متمایل بشه

پس زمانی که با ویژگی های نامرتبط و تکراری سروکار داریم اگه انالیز نکنیم وبراشون کاری نکنیم، مقداری که از فرمول distance بدست میاد خراب میشه و توی فاصله حساب کردن مشکل درست میکنه

۲تا اتریبیوت را باید باهم ببینیم
نمیشه تک تک نگاشون کرد

K-NN Classifiers: Handling attributes that are interacting

اتریبیوت هایی که به هم مرتبط هستند و وابسته هستند به یکدیگر چی؟
یه تعامل و interactionای بینشون هست و باید دوتایی بشون نگاه کنیم
ایا درصورت وجود نویز میشه حذف کرد؟



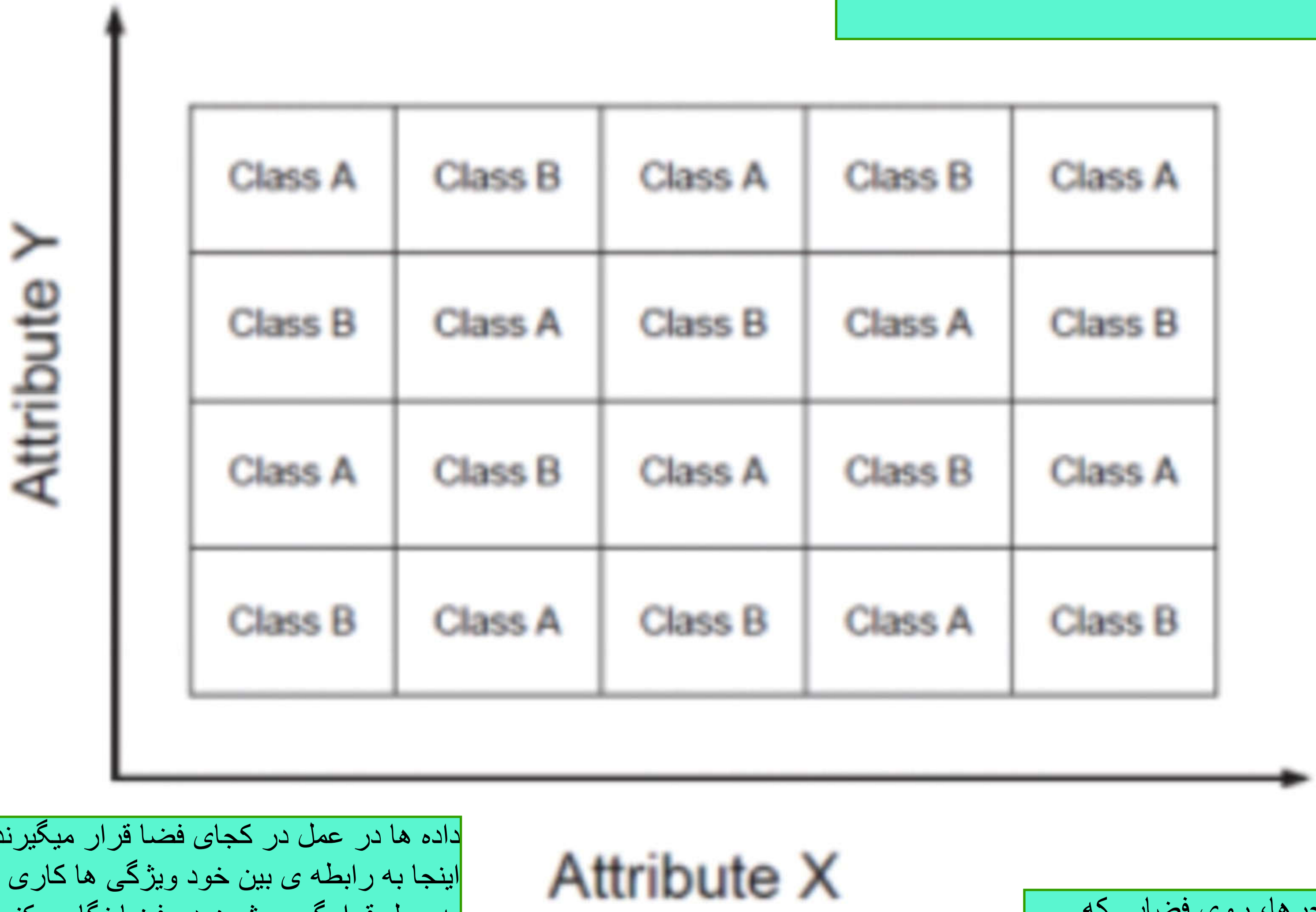
ما برای تصمیم گیری هم نیاز به ایکس داریم و هم وای
پس دوتاشون را میخاهیم
مثلا در مقدار ۴ هم یه کلاسی داریم که مقدارش قرمز است و هم کلاسی که مقدارش ابی است
در مقدار ۱۴ هم، یه نمونه ابی و یه نمونه قرمز داریم
پس کلسیفیکیشن اینجا راحت نیست!
از بعد y هم همینطوره پس باید دوبعدی به داده ها نگاه کنیم

اینجا داده ها را باید از دوبعد نگاهشون کرد به جای اینکه فقط از یه بعد نگاه کنیم

درحالت دوبعدی فضا وقتی میخاد قسمت بندی بشه، ناحیه ناحیه میشه
فضا به صورت شبکه بندی باشه

Handling attributes that are interacting

هرچه فیچرها مستقل از هم باشند بهتره
وقتی فیچرها به هم مرتبط میشوند کار کلسیفیکیشن خیلی سخت میشه



داده ها در عمل در کجای فضا قرار میگیرند؟
اینجا به رابطه ی بین خود ویژگی ها کاری نداریم
به محل قرارگیری شون در فضا نگاه میکنیم
نحوه ی قرارگیری برچسب ها مهم میشه

نحوه ی ارتباط فیچرها، روی فضایی که میخاهیم فاصله را حساب کنیم اثر میگذاره

k-NN is a distance-based algorithm that uses a similarity measure, typically Euclidean distance, to find the k-nearest neighbors of a query instance in the training set. However, one potential issue with distance-based algorithms like k-NN is that they can be affected by the scale and interactions between attributes.

When features are of different scales, some attributes may dominate the distance metric, leading to biased predictions. One way to handle this issue is to normalize the features to have zero means and unit variances, so that all attributes contribute equally to the distance metric.

Another issue arises when there are interacting attributes, meaning that the relationship between two attributes is not linear or additive. For example, in a dataset of housing prices, the interaction between the number of bedrooms and bathrooms may have a significant effect on the price, whereas each attribute alone may not be as predictive.

To handle interacting attributes, one approach is to perform feature engineering to create new features that capture the interactions between attributes. For example, we could add a feature for the product of the number of bedrooms and bathrooms, which may capture the non-linear relationship between these two attributes.

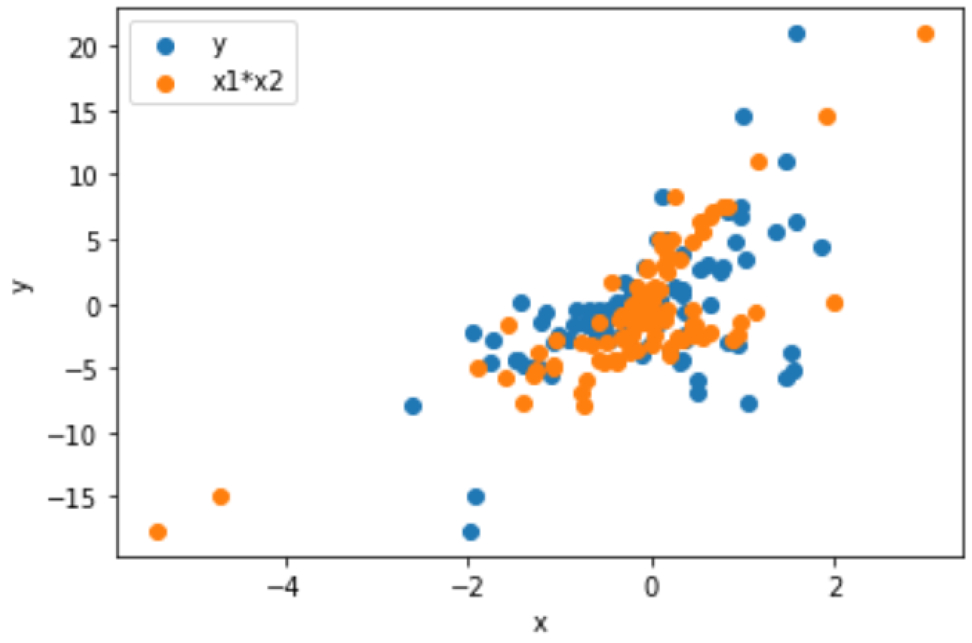
Alternatively, we could use a more advanced machine learning algorithm that can automatically learn non-linear relationships and interactions between attributes, such as decision trees, random forests, or neural networks. These algorithms can model complex relationships between attributes without explicitly creating new features, thus avoiding the need for manual feature engineering.

here's an example of a dataset with interacting attributes and a scatter plot showing the interaction:
In this example, we generated a synthetic dataset with 2 input features (x1 and x2) and a target variable y that has an interaction term 4*x1*x2. We added some random noise to the target variable to make it more realistic.

The scatter plot shows the relationship between x1 and y versus the relationship between x1*x2 and y. As we can see, the interaction term captures a non-linear relationship between the two input features and the target variable, which cannot be modeled well by a simple linear model.

By including the interaction term in the model, we can capture this non-linear relationship and improve the accuracy of our predictions.

```python
import numpy as np
import matplotlib.pyplot as plt

# Generate synthetic data with an interaction between x1 and x2
np.random.seed(42)
n_samples = 100
x1 = np.random.normal(size=n_samples)
x2 = np.random.normal(size=n_samples)
y = 2*x1 + 3*x2 + 4*x1*x2 + np.random.normal(scale=0.5, size=n_samples)

# Plot the interaction between x1 and x2
fig, ax = plt.subplots()
ax.scatter(x1, y, label='y')
ax.scatter(x1*x2, y, label='x1*x2')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.legend()
plt.show()
```



In the scatter plot, we can see that there is a clear non-linear relationship between x1*x2 and y, which is not captured by the relationship between x1 and y. This suggests that including an interaction term between x1 and x2 in the model could improve its predictive accuracy.

Improving KNN Efficiency

معمولا در مسائل بیگ دیتا از این روش ها استفاده میشه

- Avoid having to compute distance to all objects in the training set

فاصله و Distance حساب کردن زمان بر و هزینه بره نباید برای همه ی ابجکت هایی که توی ترینینگ ست هست بیایم حسابشون کنیم

  - Multi-dimensional access methods (k-d trees)
  - Fast approximate similarity search
  - Locality Sensitive Hashing (LSH)

- Condensing

متراکم شدن
- مجموعه کوچکتری از اشیاء را که عملکرد یکسانی دارند تعیین کنید

  - Determine a smaller set of objects that give the same performance

از محاسبه فاصله تا تمام اشیاء در مجموعه آموزشی خودداری کنید
- روش های دسترسی چند بعدی (درخت k-d)
- جستجوی سریع تشابه تقریبی
– محل هش حساس (LSH)

- Editing

ویرایش
- برای بهبود کارایی اشیاء را حذف کنید

  - Remove objects to improve efficiency

میشه به جای اینکه تک تک رکوردها را توی دیتابیس ذخیره کنیم، خوشه خوشه ذخیره کنیم (برای حل مشکل فاصله حساب کردن های خیلی زیاد ومحاسبات خیلی بالا میشه نمونه ها را فشرده کنیم)
بعد وقتی یه نمونه برای مقایسه اومد به جای مقایسه باهمه ی رکوردها، فقط با نماینده ی اون نمونه ها مقایسه را انجام میدیم و سرعت مقایسه کردن بالا میره

باید یه سرچی انجام بدیم و فاصله ی نمونه فعلی را بانمونه های قبلی بسنجیم و چون اردر عملیات سرچ بالاست، برای فاصله حساب کردن معمولا از رویکردهای هشینگ استفاده میکنند

There are several techniques that can be used to improve the efficiency of KNN algorithm:

Feature Selection: KNN algorithm is sensitive to irrelevant and redundant features, therefore selecting only the most relevant features can significantly reduce the dimensionality of the problem and subsequently improve the efficiency.

Distance Metrics: The choice of distance metric can have a significant impact on the performance of KNN algorithm. For instance, using Euclidean distance may not work well for high-dimensional data, whereas using Cosine similarity or Mahalanobis distance may yield better results.

KD-Tree: A KD-tree is a binary tree structure that recursively partitions data into smaller regions based on the distance measure. KD-trees can be used to speed up the search process in KNN algorithm by reducing the number of distance calculations required.

Approximate Nearest Neighbor: The approximate nearest neighbor (ANN) algorithm is a technique that trades off accuracy for speed. ANN algorithms can be used to efficiently find an approximate solution to KNN problem.

Parallelization: KNN algorithm can benefit from parallel processing techniques such as multi-threading or distributed computing. This can help to reduce the computation time for large datasets.

By employing these techniques, it is possible to significantly improve the efficiency of KNN algorithm without sacrificing its accuracy.

BAYESIAN METHOD

Basic Idea_

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Unknown record

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

Using Bayes Theorem for Classification

هر ویژگی و برچسب کلاس را به عنوان متغیرهای تصادفی در نظر بگیرید

- Consider each attribute and class label as random variables
- Given a record with attributes

$$(X_1, X_2,\ldots, X_d),$$

اتریبیوت هایی که داریم را به عنوان متغیرتصادفی درنظر میگیریم
داده ی جدیدی که میخایم پیشبینی کنیم مثل یه آزمایش است که براساس متغیر های تصادفی که قبلا مقدار گرفتن، نتیجه ی این آزمایش چی میشه؟
هدف: پیشبینی کلاس و یا اخرین اتریبیوت

the goal : predict class Y

دراینجا مشاهداتمون یک فضای احتمالاتی داره ینی میتونیم بگیم احتمال اینکه وضعیت تاهل مجرد باشه و کلاس یا لیبل اخریمون yes باشه چقدره؟
هدف: به کمک قوانین بیز شرایط داده ها را دربیاریم وتصمیم گیری کنیم برای داده های جدید

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

میخاهیم احتمال یس یا no بودن رکورد اخری را اندازه بگیریم

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Bayes Classifier

- A probabilistic framework for solving classification problems
- Conditional Probability:

$$P(Y \mid X) = \frac{P(X,Y)}{P(X)}$$

$$P(X \mid Y) = \frac{P(X,Y)}{P(Y)}$$

- Bayes theorem:

$$P(Y \mid X) = \frac{P(X \mid Y)P(Y)}{P(X)}$$

Using Bayes Theorem for Classification

- Consider each attribute and class label as random variables
- Given a record with attributes

  $(X_1, X_2, \ldots, X_d)$,

  the goal : predict class Y

  – Specifically, we want to find the value of Y that maximizes $P(Y| X_1, X_2, \ldots, X_d)$

- Can we estimate $P(Y| X_1, X_2, \ldots, X_d)$ directly from data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Using Bayes Theorem for Classification

- Approach:
  - compute posterior probability $P(Y \mid X_1, X_2, \ldots, X_d)$ using the Bayes theorem

$$\boxed{P(Y \mid X_1 X_2 \ldots X_n) = \frac{P(X_1 X_2 \ldots X_d \mid Y) P(Y)}{P(X_1 X_2 \ldots X_d)}}$$

  - *Maximum a-posteriori*: Choose Y that maximizes $P(Y \mid X_1, X_2, \ldots, X_d)$

  - Equivalent to choosing value of Y that maximizes $P(X_1, X_2, \ldots, X_d|Y)\, P(Y)$

- How to estimate $P(X_1, X_2, \ldots, X_d \mid Y)$?

- Let X be a data sample with unknown class label
- Let Y be a hypothesis that X belongs to class C
- Classification is to determine P(Y|X) (posteriori probability)
- احتمال اولیه P(Y) (prior probability): the initial probability

  E.g., X will buy computer, regardless of age, income, etc.
- P(X): probability that sample data is observed
- P(X|Y) (likelihood): the probability of observing the sample X, given that the hypothesis holds

  E.g., Given that X will buy computer, the prob. that X is 31..40, medium income, etc

$$\underbrace{P(Y \mid X)}_{\text{posteriori probability}} = \frac{\underbrace{P(X \mid Y)}_{\text{likelihood}}\,\underbrace{P(Y)}_{\text{prior probability}}}{P(X)}$$

Example Data

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Evade**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

- We need to estimate
  P(Evade = Yes | X) and P(Evade = No | X)

  In the following we will replace

  Evade = Yes by Yes, and

  Evade = No by No

Example Data

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Using Bayes Theorem:

- $$\boxed{P(\text{Yes} \mid X)} = \frac{P(X \mid \text{Yes})P(\text{Yes})}{P(X)}$$
- $$\boxed{P(\text{No} \mid X)} = \frac{P(X \mid \text{No})P(\text{No})}{P(X)}$$
- How to estimate $\boxed{P(X \mid \text{Yes})}$ and $\boxed{P(X \mid \text{No})}$?

هدف اصلی محاسبه ی اینهاست
هرکدام ماکس بشه => میشه لیبل کلاس ما

Conditional Independence

- **X** and **Y** are conditionally independent given **Z** if
  $P(\mathbf{X}|\mathbf{YZ}) = P(\mathbf{X}|\mathbf{Z})$

  Equivalently

  $$P(XY|Z) = P(X|Z)P(Y|Z)$$

مثال: طول بازو و مهارت خواندن
- کودک خردسال در مقایسه با بزرگسالان دارای طول بازوی کوتاه تر و مهارت های خواندن محدودتر است
- اگر سن ثابت باشد، هیچ رابطه آشکاری بین طول بازو و مهارت خواندن وجود ندارد
- طول بازو و مهارت های خواندن به صورت شرطی مستقل از سن هستند

- Example: Arm length and reading skills
  - Young child has shorter arm length and limited reading skills, compared to adults
  - If age is fixed, no apparent relationship between arm length and reading skills
  - Arm length and reading skills are conditionally independent given age

Naïve Bayes Classifier

در این روش فرض میکنیم که Xها از هم مستقل هستند یعنی x1,x2,x3,... به شرط y از هم مستقل میشوند.
وقتی دوتا متغیر تصادفی نسبت به یک متغیر دیگر مستقل شوند => conditionally independent که باشند این اتفاق میفته:

- Assume independence among attributes X_i when class is given:
  - $P(X_1, X_2, \ldots, X_d | Y_j) = P(X_1 | Y_j)\, P(X_2 | Y_j) \ldots P(X_d | Y_j)$
  - Now we can estimate $P(X_i | Y_j)$ for all X_i and Y_j combinations from the training data
  - New point is classified to Y_j if $P(Y_j) \prod P(X_i | Y_j)$ is maximal.

الان محاسباتمون خیلی ساده شد چون اینارو میشه جداجدا حساب کرد

ضرب

Naïve Bayes on Example Data

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

این سه تا ویژگی
مستقل از هم هستند
پس میشه احتمال
ای شرطی شون را
در هم ضرب کرد

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Evade**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

P(X | Yes) =

P(Refund = No | Yes) x
P(Divorced | Yes) x
P(Income = 120K | Yes)

این داده که پیوسته
است چکارش کنیم؟
continues است

P(X | No) =

P(Refund = No | No) $\boxed{x}$
P(Divorced | No) $\boxed{x}$
P(Income = 120K | No)

Estimate Probabilities from Data

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- $P(y)$ = fraction of instances of class y
  - e.g., $P(No) = 7/10$, $P(Yes) = 3/10$

اتریبیوت های دسته ی مثل وضعیت تاهل که ۳تا حالت داره:
single,married divorced

- For categorical attributes:

$$P(X_i = c | y) = n_c / n$$

  - where $|X_i = c|$ is number of instances having attribute value $X_i = c$ and belonging to class y
  - Examples:

$P(Status=Married|No) = 4/7$
$P(Refund=Yes|Yes)=0$

Estimate Probabilities from Data

عملیات های شکستن و تقسیم بندی و بین کردن به بحث های پیش پردازش برمیگرده مثلا یه سری چارچوب و قوانین هست که چطوری اینا را بشکنیم بهتر بشه مدلمون

- For continuous attributes:
  - Discretization: Partition the range into bins:
    - Replace continuous value with bin value
      - Attribute changed from continuous to ordinal

میتوانیم بازه ی پیوسته ای که داریم را به یه تعداد مشخصی از بازه های فیکس بشکنیم وتقسیم کنیم بعد دیگه به راحتی میشه شمرد: مثلا بگیم اگه سطح درامد از مقدار ۱۰۰ تا ۱۲۰ هزار بود تو یه دسته قرار بگیره

اینطوری شمارش داده ها برای احتمال پیداکردن خیلی راحتتر میشه

میتوانیم فرض کنیم که ستون درامدمون از یک توزیعی پیروی میکنه چون توزیع بمون احتمال میده پس میشه از میانگین و واریانس و انحراف معیار که به صورت احتمالی هم کار میکنه استفاده کرد به جای پارتیشن کردن مقادیر پیوسته مثل درامد

Probability density estimation:

- Assume attribute follows a normal distribution
- Use data to estimate parameters of distribution (e.g., mean and standard deviation)
- Once probability distribution is known, use it to estimate the conditional probability $P(X_i|Y)$

- تخمین چگالی احتمال:
فرض کنید اتریبیوت از توزیع نرمال پیروی می کند.
استفاده از داده ها برای تخمین پارامترهای توزیع (به عنوان مثال، میانگین و انحراف استاندارد).
هنگامی که توزیع احتمال مشخص شد، از آن برای تخمین احتمال شرطی P(Xi|Y) استفاده کنید.

برای صفات پیوسته:
- گسسته سازی: محدوده را به bin ها تقسیم کنید: مقدار پیوسته را با مقدار bin جایگزین کنید
- تغییر اتریبیوت از پیوسته به ترتیبی

معمولا یک توزیع نرمال استفاده میکنیم
برای نمایش یک توزیع نرمال به ۲تا پارامتر
نیاز داریم: ۱.میانگین ۲.واریانس یا انحراف

Estimate Probabilities from Data

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$\boxed{P(X_i \mid Y_j)} = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(X_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

میانگین

واریانس

  – One for each (X_i, Y_j) pair

- For (Income, Class=No):
  – If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

2975 ^ 1/2 = 54.54

احتمال اینکه برچسب
مونه ی من با درامد ۱۲۰
، کلاسش no باشه عدد
0.0072 است

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

Naïve Bayes Classifier:

P(Refund = Yes | No) = 3/7
P(Refund = No | No) = 4/7
P(Refund = Yes | Yes) = 0
P(Refund = No | Yes) = 1
P(Marital Status = Single | No) = 2/7
P(Marital Status = Divorced | No) = 1/7
P(Marital Status = Married | No) = 4/7
P(Marital Status = Single | Yes) = 2/3
P(Marital Status = Divorced | Yes) = 1/3
P(Marital Status = Married | Yes) = 0

For Taxable Income:
If class = No: sample mean = 110
sample variance = 2975
If class = Yes: sample mean = 90
sample variance = 25

- P(X | No) = P(Refund=No | No) $\times$ P(Divorced | No) $\times$ P(Income=120K | No) = 4/7 $\times$ 1/7 $\times$ 0.0072 = 0.0006
- P(X | Yes) = P(Refund=No | Yes) $\times$ P(Divorced | Yes) $\times$ P(Income=120K | Yes) = 1 $\times$ 1/3 $\times$ 1.2 $\times$ 10^{-9} = 4 $\times$ 10^{-10}

Since P(X|No)P(No) > P(X|Yes)P(Yes)
Therefore P(No|X) > P(Yes|X)
=> Class = No

p(no)
را از کجا اورد؟

P(Yes) = 3/10
P(No) = 7/10

Naïve Bayes Classifier can make decisions with partial information about attributes in the test record

Even in absence of information about any attributes, we can use Apriori Probabilities of Class Variable:

P(Yes) = 3/10

P(No) = 7/10

$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$

Naïve Bayes Classifier:

P(Refund = Yes | No) = 3/7
P(Refund = No | No) = 4/7
P(Refund = Yes | Yes) = 0
P(Refund = No | Yes) = 1
P(Marital Status = Single | No) = 2/7
P(Marital Status = Divorced | No) = 1/7
P(Marital Status = Married | No) = 4/7
P(Marital Status = Single | Yes) = 2/3
P(Marital Status = Divorced | Yes) = 1/3
P(Marital Status = Married | Yes) = 0

For Taxable Income:
If class = No: sample mean = 110
sample variance = 2975
If class = Yes: sample mean = 90
sample variance = 25

If we only know that marital status is Divorced, then:

P(Yes | Divorced) = 1/3 x 3/10 / P(Divorced)

P(No | Divorced) = 1/7 x 7/10 / P(Divorced)

If we also know that Refund = No, then

P(Yes | Refund = No, Divorced) = 1 x 1/3 x 3/10 / P(Divorced, Refund = No)

P(No | Refund = No, Divorced) = 4/7 x 1/7 x 7/10 / P(Divorced, Refund = No)

If we also know that Taxable Income = 120, then

P(Yes | Refund = No, Divorced, Income = 120) = 1.2×10^{-9} x 1 x 1/3 x 3/10 / P(Divorced, Refund = No, Income = 120)

P(No | Refund = No, Divorced Income = 120) = 0.0072 x 4/7 x 1/7 x 7/10 / P(Divorced, Refund = No, Income = 120)

مشکلات این کلسیفایر؟
اگه داده هامون ناقص باشند چی میشه؟ مثلا یه رکوردی داشته باشیم که سطح درامدش رو ندانیم ولی بقیه ی اتریبیوت هارو بدانیم مثلا بدانیم که متاهل است و ... چه مقادیری دارند
یعنی missing features داشته باشیم آیا کلسیفایر با روش بیزین جواب درست میده بمون؟
چون اون اتریبیوت را نداریم احتمالش هم حساب نمیشه پس برامون مهم نیست و براساس اونایی که هستند تصمیم میگیریم =>مدل بیزین برای مقادیر گم شده هم درست کار میکنه

Issues with Naïve Bayes Classifier

Given a Test Record:

X = (Married)

Naïve Bayes Classifier:

P(Refund = Yes | No) = 3/7
P(Refund = No | No) = 4/7
P(Refund = Yes | Yes) = 0
P(Refund = No | Yes) = 1
P(Marital Status = Single | No) = 2/7
P(Marital Status = Divorced | No) = 1/7
P(Marital Status = Married | No) = 4/7
P(Marital Status = Single | Yes) = 2/3
P(Marital Status = Divorced | Yes) = 1/3
P(Marital Status = Married | Yes) = 0

For Taxable Income:
If class = No: sample mean = 110
sample variance = 2975
If class = Yes: sample mean = 90
sample variance = 25

P(Yes) = 3/10

P(No) = 7/10

P(Yes | Married) = 0 x 3/10 / P(Married)

P(No | Married) = 4/7 x 7/10 / P(Married)

= p(married | no) * p(no) / p(married)

چالش دیگه:
اگه احتمال شرطی صفر بشه کل جواب صفر میشه
یعنی نمیتوانه تصمیم گیری کنه اگه احتمال صفر بشه
مثل اینه که یه شرایطی پیش بیاد و ما براش رکورد نداشته باشیم یا تعداد داده ها کم باشه!
ثل پاک کردن رکورد۷م در مثال بعدی که باعث صفرشدن احتمال میشه

Issues with Naïve Bayes Classifier

$$P(X_1, X_2, \ldots, X_d | Y_j) = P(X_1| Y_j)\, P(X_2| Y_j) \ldots P(X_d| Y_j)$$

If one of the conditional probabilities is zero, then the entire expression becomes zero

اگر یکی از احتمالات شرطی صفر باشد، کل عبارت صفر می شود

Issues with Naïve Bayes Classifier

Consider the table with Tid = 7 deleted

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Naïve Bayes Classifier:

P(Refund = Yes | No) = 2/6
P(Refund = No | No) = 4/6
P(Refund = Yes | Yes) = 0
P(Refund = No | Yes) = 1
P(Marital Status = Single | No) = 2/6
P(Marital Status = Divorced | No) = 0
P(Marital Status = Married | No) = 4/6
P(Marital Status = Single | Yes) = 2/3
P(Marital Status = Divorced | Yes) = 1/3
P(Marital Status = Married | Yes) = 0/3
For Taxable Income:
If class = No: sample mean = 91
sample variance = 685
If class = No: sample mean = 90
sample variance = 25

Given X = (Refund = Yes, Divorced, 120K)

$P(X \mid No) = 2/6 \times 0 \times 0.0083 = 0$

$P(X \mid Yes) = 0 \times 1/3 \times 1.2 \times 10^{-9} = 0$

p(Refund = yes| Yes) =0

P(Marial Status = Divorced | No) = 0

Naïve Bayes will not be able to classify X as Yes or No!

راه حل مشکل صفرشدن احتمال ها:
تقریب زدن احتمال ها مثلا صورت کسر را بایک عددکوچکی جمع کنیم تا هیچ وقت صفر تولید نشه
تخمین احتمالاتی :)
probability estimation

Issues with Naïve Bayes Classifier

- If one of the conditional probabilities is zero, then the entire expression becomes zero
- Need to use other estimates of conditional probabilities than simple fractions
- Probability estimation:

$$\boxed{\text{original}}:\ P(X_i = c|y) = \boxed{\frac{n_c}{n}}$$

$$\boxed{\text{Laplace Estimate}}:\ P(X_i = c|y) = \boxed{\frac{n_c + 1}{n + v}}$$

$$\boxed{\text{m}-\text{estimate:}}\ P(X_i = c|y) = \boxed{\frac{n_c + mp}{n + m}}$$

n: number of training instances belonging to class y

n_c: number of instances with $X_i = c$ and $Y = y$

v: total number of attribute values that X_i can take

p: initial estimate of $(P(X_i = c|y))$ known apriori

m: hyper-parameter for our confidence in p

Example of Naïve Bayes Classifier

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$\boxed{P(A|M)} = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$\boxed{P(A|N)} = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$\boxed{P(A|M)P(M)} = 0.06 \times \boxed{\frac{7}{20}} = 0.021$$

$$\boxed{P(A|N)P(N)} = 0.004 \times \boxed{\frac{13}{20}} = 0.0027$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

P(A|M)P(M) > P(A|N)P(N)

=> Mammals

Naïve Bayes (Summary)

1.نسبت به نویز مقاوم است
۲. اگه میسینگ ولیوز داشته باشیم بازم این روش بمون یه جوابی میده
۳. اگه متغیرهامون مستقل از هم باشند کار میکنه و جواب میده
۴. عیبی که داره برای متغیرها و اتریبیوت های وابسته است که کرلیشن دارند میتواند احتمال شرطی را خراب کنه ما در این مدل فرض کردیم متغیرها که همان اتریبیوت هاموه هستند از هم مستقل اند

- Robust to isolated noise points

- Handle missing values by ignoring the instance during probability estimate calculations

- Robust to irrelevant attributes

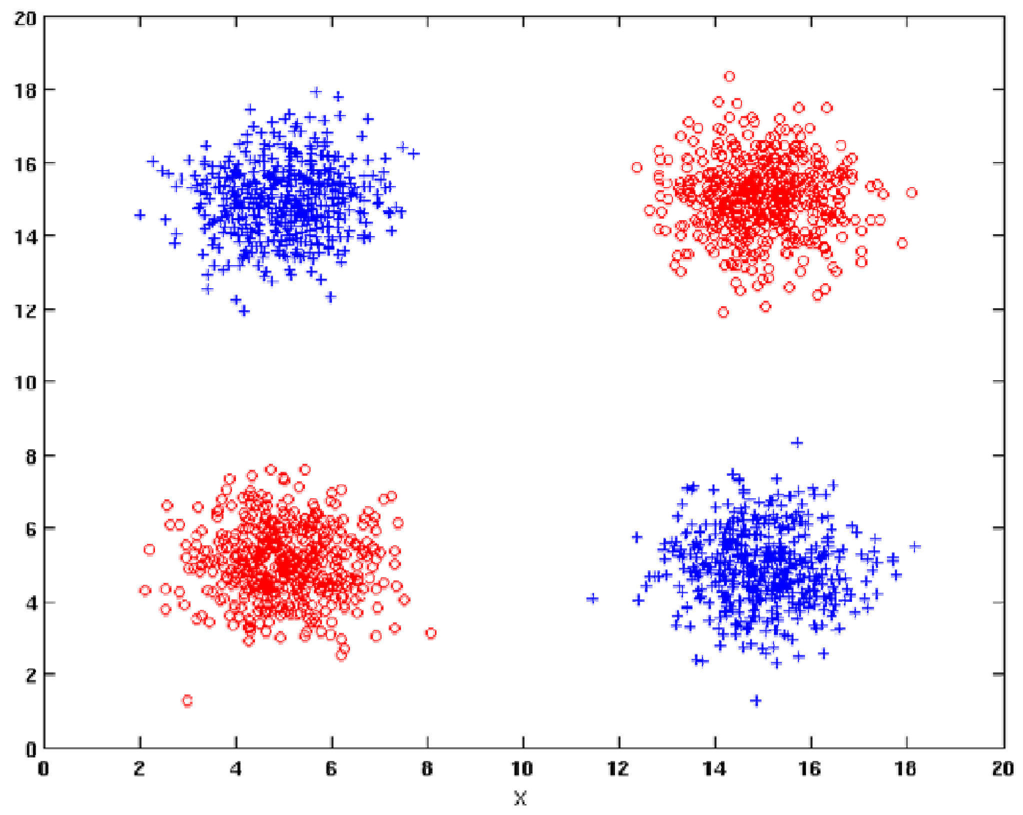
- Redundant and correlated attributes will violate class conditional assumption
  - Use other techniques such as Bayesian Belief Networks (BBN)

عیب این روش

مقاوم به نقاط نویز جدا شده
مقادیر گمشده را با نادیده گرفتن نمونه در طول محاسبات برآورد احتمال مدیریت کنید
مقاوم به ویژگی های نامربوط
صفات زائد و مرتبط، فرض شرطی کلاس را نقض خواهند کرد
- استفاده از تکنیک های دیگر مانند شبکه های باور بیزی (BBN)

Naïve Bayes

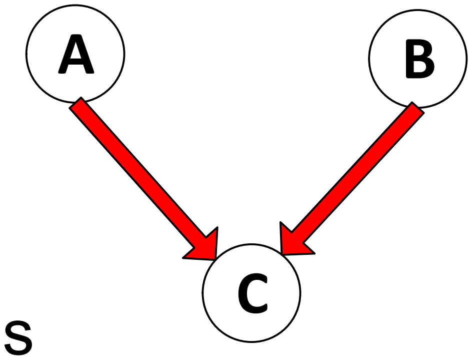
- How does Naïve Bayes perform on the following dataset?



Conditional independence of attributes is violated

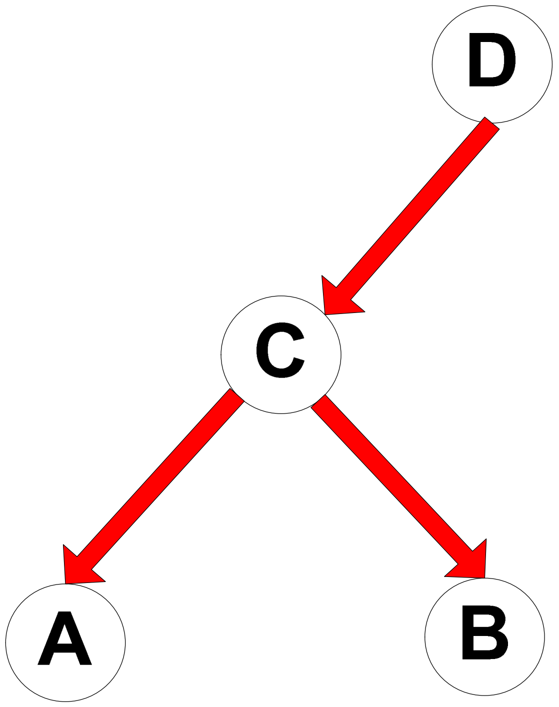
Bayesian Belief Networks

- Provides graphical representation of probabilistic relationships among a set of random variables
- Consists of:
  - A directed acyclic graph (dag)
    - Node corresponds to a variable
    - Arc corresponds to dependence relationship between a pair of variables



  - A probability table associating each node to its immediate parent

Conditional Independence


$$\begin{tikzcd} & & D \arrow[dl] \\ & C \arrow[dl] \arrow[dr] & \\ A & & B \end{tikzcd}$$

D is parent of C

A is child of C

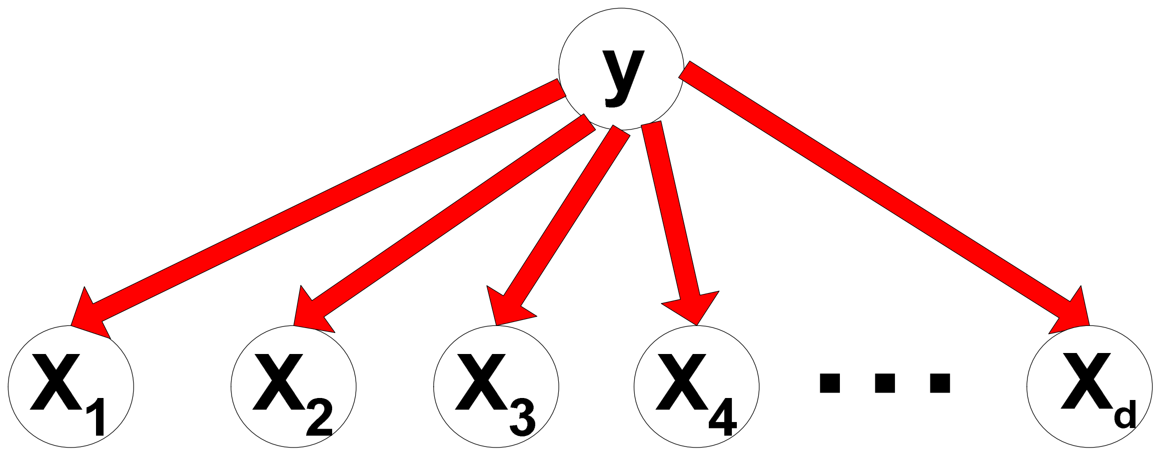
B is descendant of D

D is ancestor of A

- A node in a Bayesian network is conditionally independent of all of its nondescendants, if its parents are known

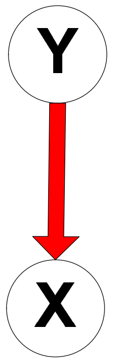
Conditional Independence

- Naïve Bayes assumption:



Probability Tables

- If X does not have any parents, table contains prior probability P(X)
- If X has only one parent (Y), table contains conditional probability P(X|Y)

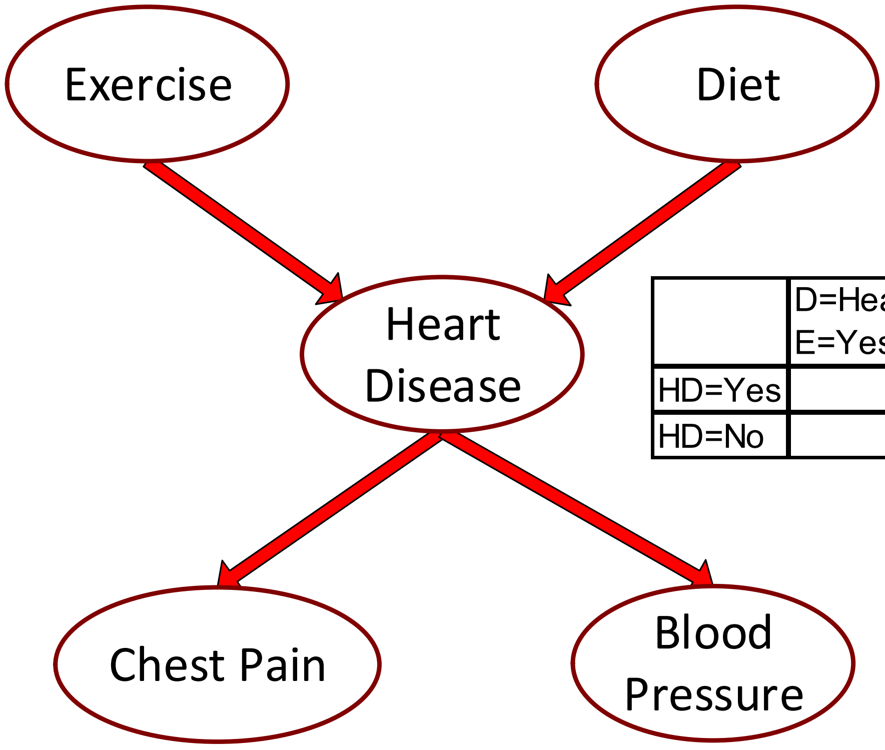

$$\begin{tikzcd} Y \arrow[d] \\ X \end{tikzcd}$$

- If X has multiple parents (Y_1, Y_2,…, Y_k), table contains conditional probability P(X|Y_1, Y_2,…, Y_k)

Example of Bayesian Belief Network

Exercise=Yes	0.7
Exercise=No	0.3

Diet=Healthy	0.25
Diet=Unhealthy	0.75



	D=Healthy E=Yes	D=Healthy E=No	D=Unhealthy E=Yes	D=Unhealthy E=No
HD=Yes	0.25	0.45	0.55	0.75
HD=No	0.75	0.55	0.45	0.25

	HD=Yes	HD=No
CP=Yes	0.8	0.01
CP=No	0.2	0.99

	HD=Yes	HD=No
BP=High	0.85	0.2
BP=Low	0.15	0.8

Example of Inferencing using BBN

- Given: X = (E=No, D=Yes, CP=Yes, BP=High)
  - Compute P(HD|E,D,CP,BP)?
- P(HD=Yes| E=No,D=Yes) = 0.55
  P(CP=Yes| HD=Yes) = 0.8
  P(BP=High| HD=Yes) = 0.85
  - P(HD=Yes|E=No,D=Yes,CP=Yes,BP=High) $\propto 0.55 \times 0.8 \times 0.85 = 0.374$
- P(HD=No| E=No,D=Yes) = 0.45
  P(CP=Yes| HD=No) = 0.01
  P(BP=High| HD=No) = 0.2
  - P(HD=No|E=No,D=Yes,CP=Yes,BP=High) $\propto 0.45 \times 0.01 \times 0.2 = 0.0009$

Classify X as Yes