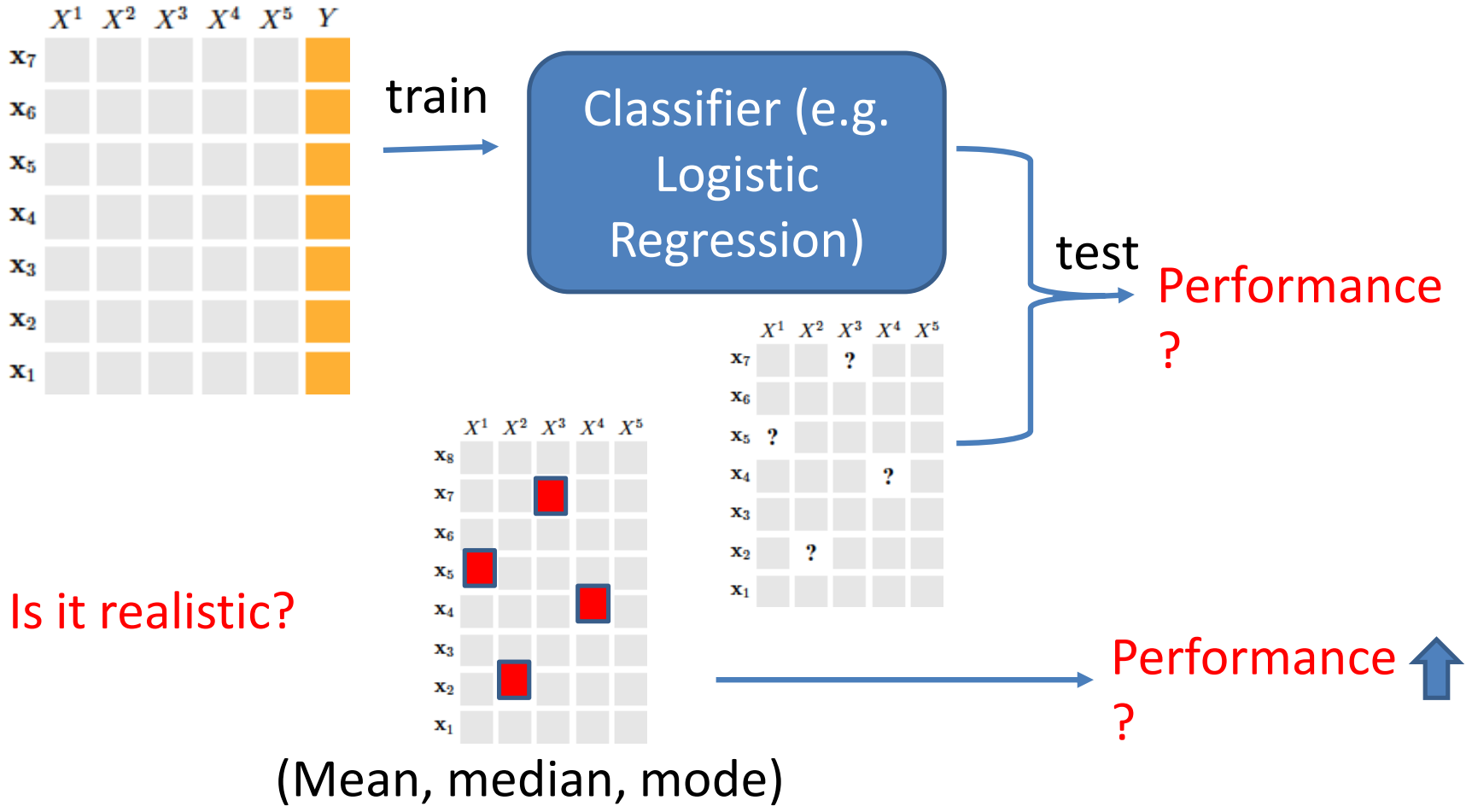
14.10.2024

BITS F464: Machine Learning (1st Sem 2024-25)

BAYESIAN LEARNING: GENERATIVE APPROACH TO CLASSIFICATION

Chittaranjan Hota, Sr. Professor
Dept. of Computer Sc. and Information Systems
hota@hyderabad.bits-pilani.ac.in

Motivation: Why generative models?

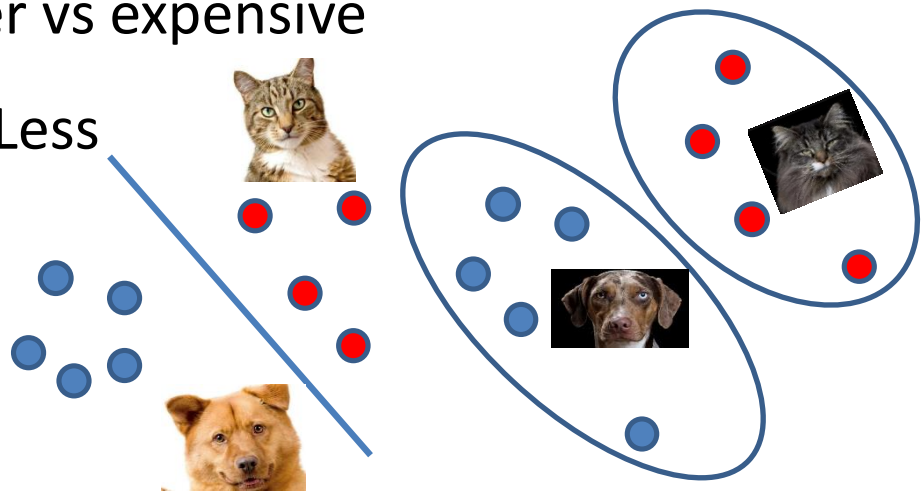


Hence, it is essential to handle test data the same way the train data was handled.

Discriminative Vs Generative Models

- Story: Assume that Rajesh scores 'A' grade in ML by comparing what he studied in other related courses where as Amit scores same 'A' grade in ML course by learning every detail of the algorithms taught (with several examples) in ML.
- Two different approaches only. Which one is what?
- Computationally: Cheaper vs expensive
- Outliers: More robust vs Less

Use cases: classification, supervised learning tasks vs denoising, density estimation, un-supervised learning tasks



Discriminative Vs Generative Models

- Both may be used for classification tasks

• Learn the decision boundary from samples.	• Learn the underlying probability distribution of data from samples.
• Estimates a function f: x -> y, by maximizing the conditional probability P(y|x)	• Estimates a function f: x -> y by maximizing joint probability of P(x,y)
• Assume some functional form for the probability such as P(y|x) and with training data estimate the parameters of P(y|x) i.e 'w' and bias	• Estimate the prior probability P(y) and the likelihood probability P(x|y) from the samples. Use Bayes rule to compute P(y|x)
• Ex: Logistic regression, RF	• Ex: Bayesian Networks, Naïve Bayes, HMM, GANs, …
Conditional probability of the target y, given an observation x.	Conditional probability of the observable x, given a target y.

Continued…

Which one can generate new data samples that are similar to the training data?

https://this-person-does-not-exist.com/ Nvidia/StyleGAN

Que: If you have small no. of labeled data, which model should you prefer?

Bayesian Learning: Generative Approach

- Bayesian learning uses probability to model data and quantify uncertainty of predictions.
- Each observed training data can incrementally decrease or increase the estimated probability of a hypothesis rather than completely eliminating a hypothesis if it is found to be inconsistent with a single example.
- Prior knowledge can be combined with observed data to determine the final probability of a hypothesis.
- Moreover new instances can be classified by combining predictions of multiple hypotheses.

Two Roles for Bayesian Methods

- Provides practical learning algorithms
  - Bayesian belief networks/ Bayesian networks learning
  - Naive Bayes learning
  - Combines prior knowledge (prior probabilities) with observed data
- Provides useful conceptual framework
  - Acts as a benchmark for evaluating other learning algorithms
- Practical difficulty: Initial knowledge of many probabilities, Significant computational cost.

Baye's Theorem

$$P(h|D) = \frac{P(D|h).P(h)}{P(D)}$$

Likelihood Prior

Posterior

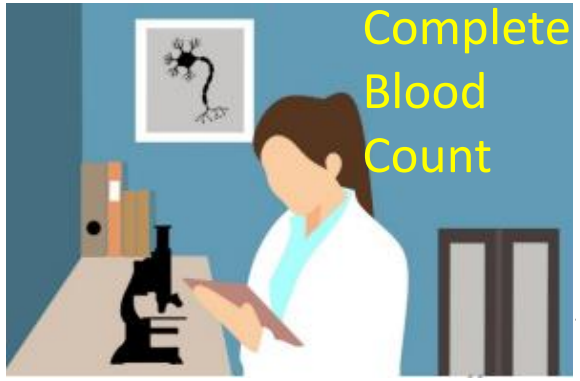
Marginal

Intuition:
when will $P(h|D)$ increase?

Thomas Bayes (1701-1761)

- Likelihood: represents the probability of observing 'D' given the hypothesis or model parameters.
- Marginal: represents the overall probability of observing the data or evidence, irrespective of any specific hypotheses.

$$P(h_i|D) = \frac{P(D|h_i).P(h_i)}{\sum_{k=1}^{n} P(D|h_k).P(h_k)}$$



Maximum A Posteriori (MAP)

- How does the learner choose the most probable hypothesis (h) out of many candidate hypotheses (H) given the observed data (D)?
- Any such maximally probable hypothesis is called maximum a posteriori (MAP) hypothesis. $\rightarrow h_{MAP}$

- $h_{MAP} = \underset{h \in H}{\text{argmax}}\ P(h|D)$

$= \underset{h \in H}{\text{argmax}}\ P(D|h).\ P(h)/\ P(D)$

$= \underset{h \in H}{\text{argmax}}\ P(D|h).\ P(h)$ Eq:1

Que: Why P(D) is dropped in the final equation?

What is argmax?

The argmax (arg max) function returns the argument or arguments (arg) for the target function that returns the maximum (max) value from the target function.

Example: Let $g(x) = x^2$ and x: 1 to 3

g(1) = 1 g(2) = 4 and g(3) = 9

→ argmax g(x) = 3

In ML: Say, a 3-class classification problem with red = 0, blue = 1, green = 2. Assume that the model predicts, pred= [0.2,0.1,0.7] as the probabilities of different classes. → argmax(pred) = ?

Maximum Likelihood Estimation

- MLE aims to find the parameter values that make the observed data most probable.
- In some cases, we will assume that every hypothesis in H is equally probable a priori ($P(h_i) = P(h_j)$ for all h_i and h_j in H). Then Eq:1 becomes:

$$\text{MLE} = \underset{h_i \in H}{\text{argmax}}\, P(D|h_i).$$

The likelihood tells us how likely one sample is relative to another.

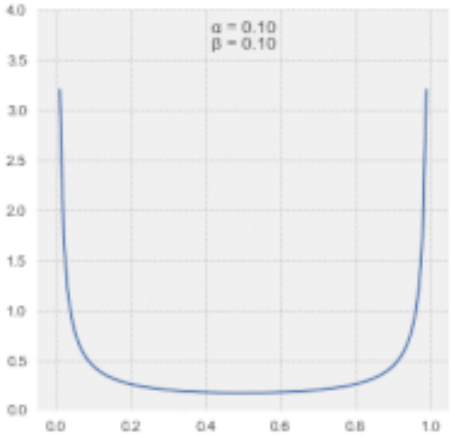
- In other words, MLE treats $P(h)/P(D)$ as a constant and does not allow us to inject our prior beliefs, P(h), about the likely values of *h*.

- An example: Let's say we flip the coin 10 times and observe 7 heads. Using MLE, we estimate the probability of getting heads as 7/10 or 0.7 because this value maximizes the likelihood of observing 7 heads in 10 flips.

Informally: if the parameters are correct then they will give larger probabilities for the observations, compared to wrong parameters.

Continued…

- Let's say we have a coin, and we want to estimate the probability of getting heads (H) when flipping it.
- Our prior belief about this probability might be represented by a Beta distribution, which is a common choice for modeling probabilities. Let Beta(α, β), with α=1, β=1, i.e uniform prior distribution (no bias towards getting heads)



- Now, let's say we flip the coin 10 times and get 7 heads (H) and 3 tails (T). To estimate the probability of getting heads using MLE in a Bayesian context, we update our prior belief with the observed data using Bayes' theorem:

$$P(\text{heads}|\text{data}) = \frac{P(\text{data}|\text{heads}) \times P(\text{heads})}{P(\text{data})}$$

- Since our prior is uniform (Beta(1, 1)), P(heads)=0.5 and P(tails)=0.5.
- Using binomial distribution: P(data|heads) = $\binom{10}{7} \times p^7 \times (1-p)^3$

In log-likelihood

- We want to estimate P(x = Head) = 1 - P(x = Tail) and hence the hypothesis space can be parameterized by a single variable $\boldsymbol{\theta}$ such that P(x = H) = $\boldsymbol{\theta}$, i.e, P(D|h) = P(D|$\boldsymbol{\theta}$).

- Assuming independence between events, we can write:
  P(D|h) = $\prod_{i=1}^{n} P(x_i|\boldsymbol{\theta})$

- Using log of the likelihood function because of notational convenience and also since the product of probabilities can be very small:

- log P(D|h) = log $\prod_{i=1}^{n} P(x_i|\boldsymbol{\theta})$ = $\sum_{i=1}^{n} \log P(x_i|\boldsymbol{\theta})$

- Hence, MLE = $\underset{h \in H}{\text{argmax}}$ P(D|h) = $\underset{\boldsymbol{\theta}}{\text{argmax}}$ $\sum_{i=1}^{n} \log P(x_i|\boldsymbol{\theta})$

❯ The maximum likelihood estimation (MLE) maximizes the log-likelihood.

For Linear Regression: Maximizing ML estimate is equivalent to minimizing least-square error. (seen before)

An Example

- Does patient have cancer or not?
- When patient takes a lab test, the possible results could be Positive (Pos) or Negative (Neg).

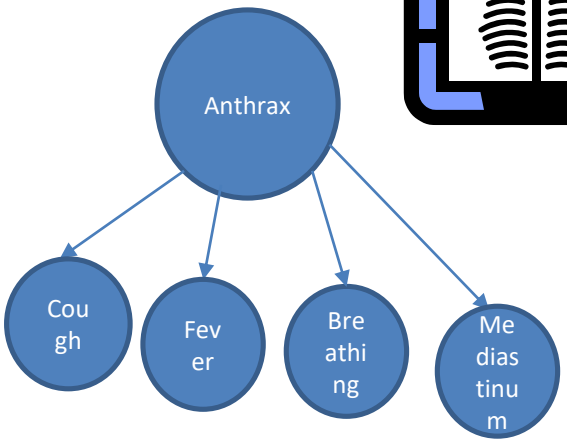
- P(cancer)=0.008, P(~cancer)=0.992, P(Pos|cancer)=0.98, P(Neg|cancer) = 0.02, P(Pos|~cancer) = 0.03, P(Neg|~cancer) = 0.97
- Suppose we now observe a new patient for whom the lab test returns a "Pos" result. Should we diagnose the patient as having cancer or not?

P(cancer|Pos)
=P(Pos|cancer).P(cancer)/{P(Pos|cancer).P(cancer)+P(Pos|~cancer).P(~cancer)} = 0.98X0.008/(0.98X0.008 + 0.03X0.992) = 0.21

→ h_{MAC} = ~cancer

Imp: while the posterior probability of cancer is significantly higher than its prior probability, the most probable hypothesis is still that the patient does not have cancer.

Bayesian Networks: An Example



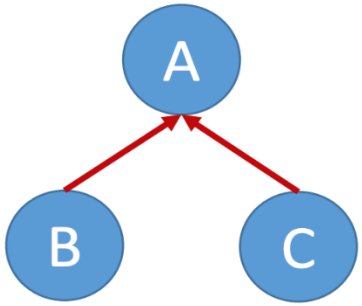
- Suppose you are trying to determine if a patient has inhalational anthrax. You observe the following symptoms:
  - The patient has a cough
  - The patient has a fever
  - The patient has difficulty breathing
- Are we not dealing with uncertainty?
  x-ray report confirms that the patient has a wide mediastinum.
- Most significant contribution in past decades

Spam filtering, robotics, diagnostic systems, detecting credit card fraud etc.

A Bayesian network is a probabilistic graphical model that represents a set of variables and their conditional dependencies via a directed acyclic graph.

Conditional Independence

•Topology of the network encodes conditional independence assertions:

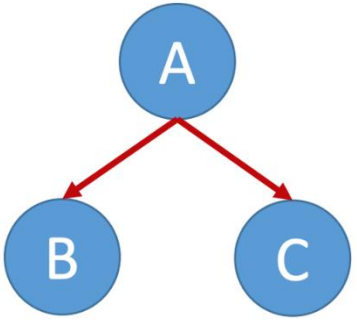


- Independent causes

$$P(A, B, C) = P(A|B, C)P(B)P(C)$$

- Marginal independence:

$$P(A, B, C) = P(A)P(B)P(C)$$



- Conditionally independent effects

$$P(A, B, C) = P(B|A)P(C|A)P(A)$$

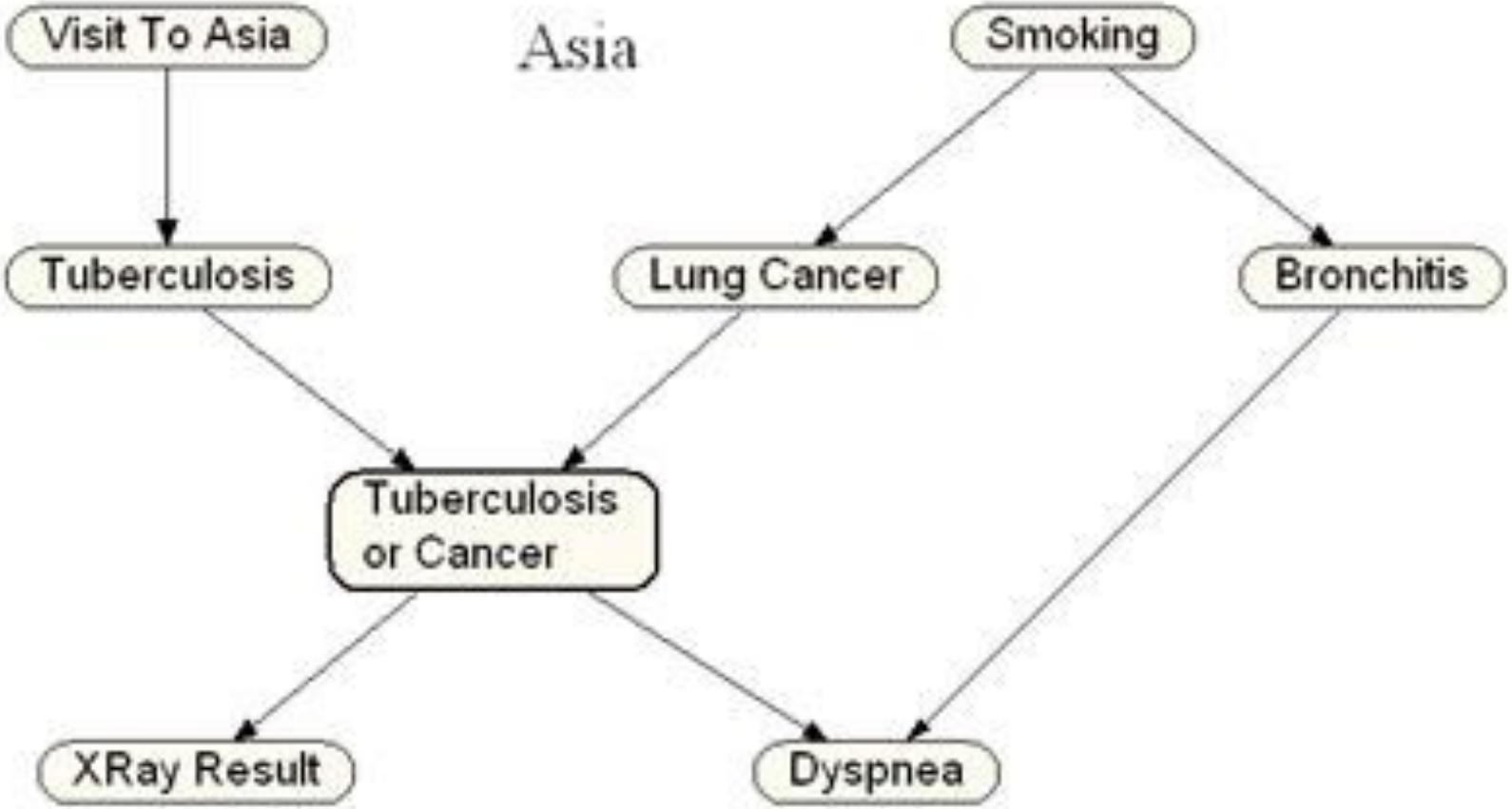
•Weather is independent of the other variables

•Toothache and Catch are conditionally independent given Cavity

---

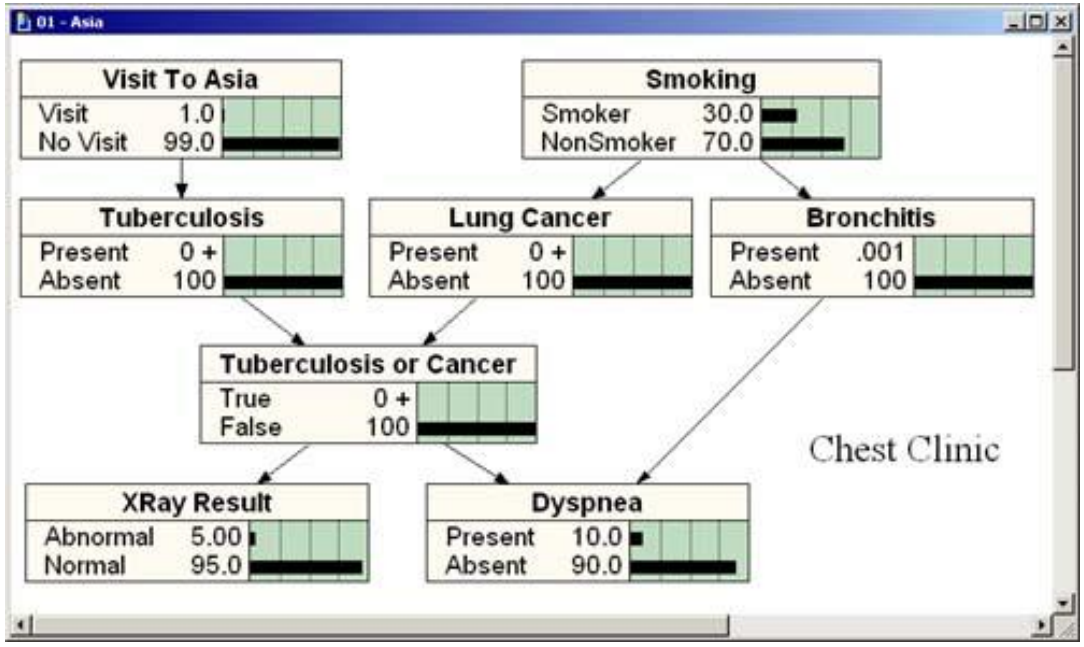
Bayesian Networks are also called as Bayesian Belief Networks.

An example with probabilities



[Source: Norsys presentation]

- 30% of the US population smokes.
- Lung cancer can be found in about 70 people per 100,000.
- TB occurs in about 10 people per 100,000.
- Bronchitis can be found in about 800 people per 100,000.
- Dyspnea can be found in about 10% of people, but most of that is due to asthma and causes other than TB, lung cancer, or bronchitis.

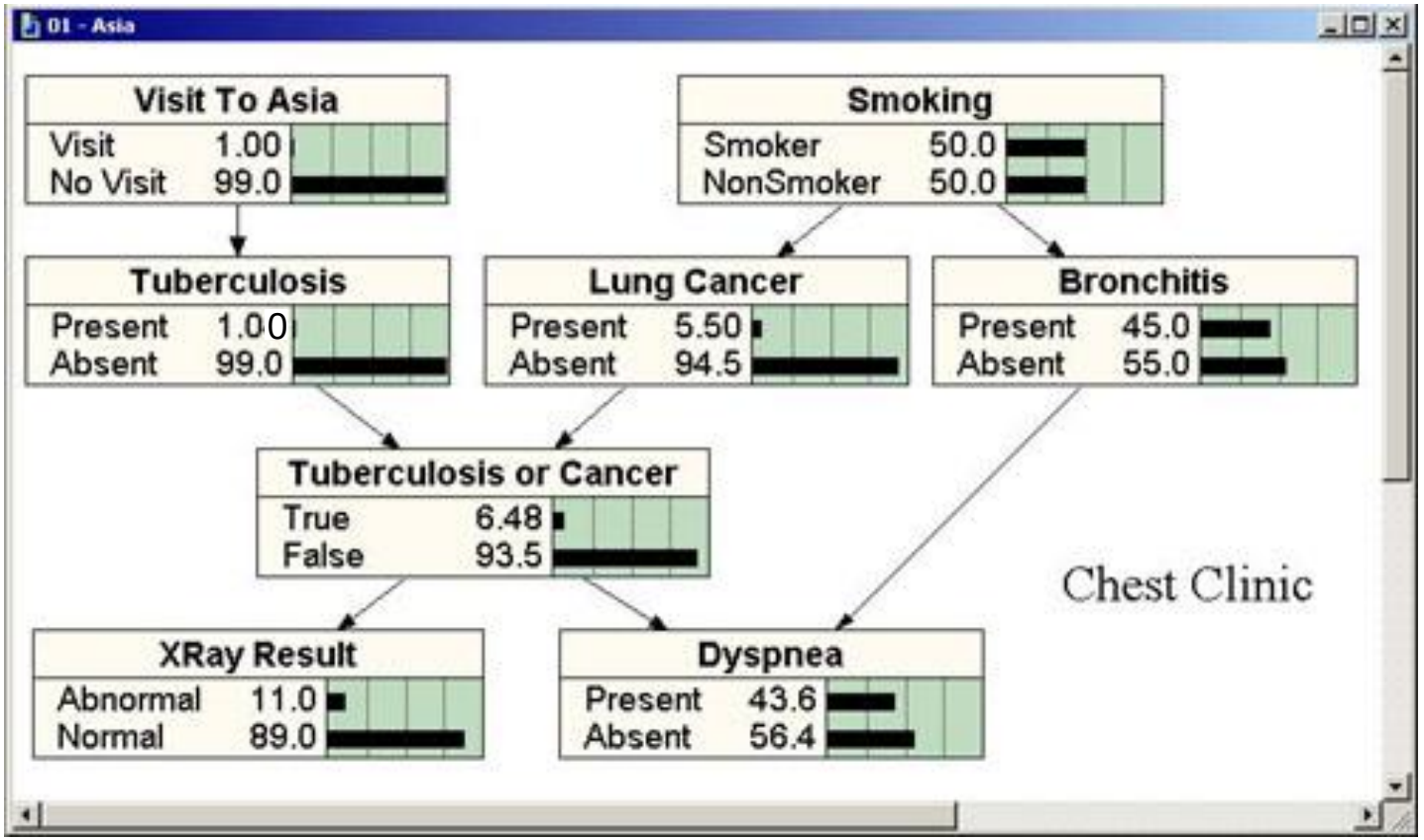


Most of them have been referred by their family physicians, and so the incidences of lung disease amongst that population is much higher, you would imagine.

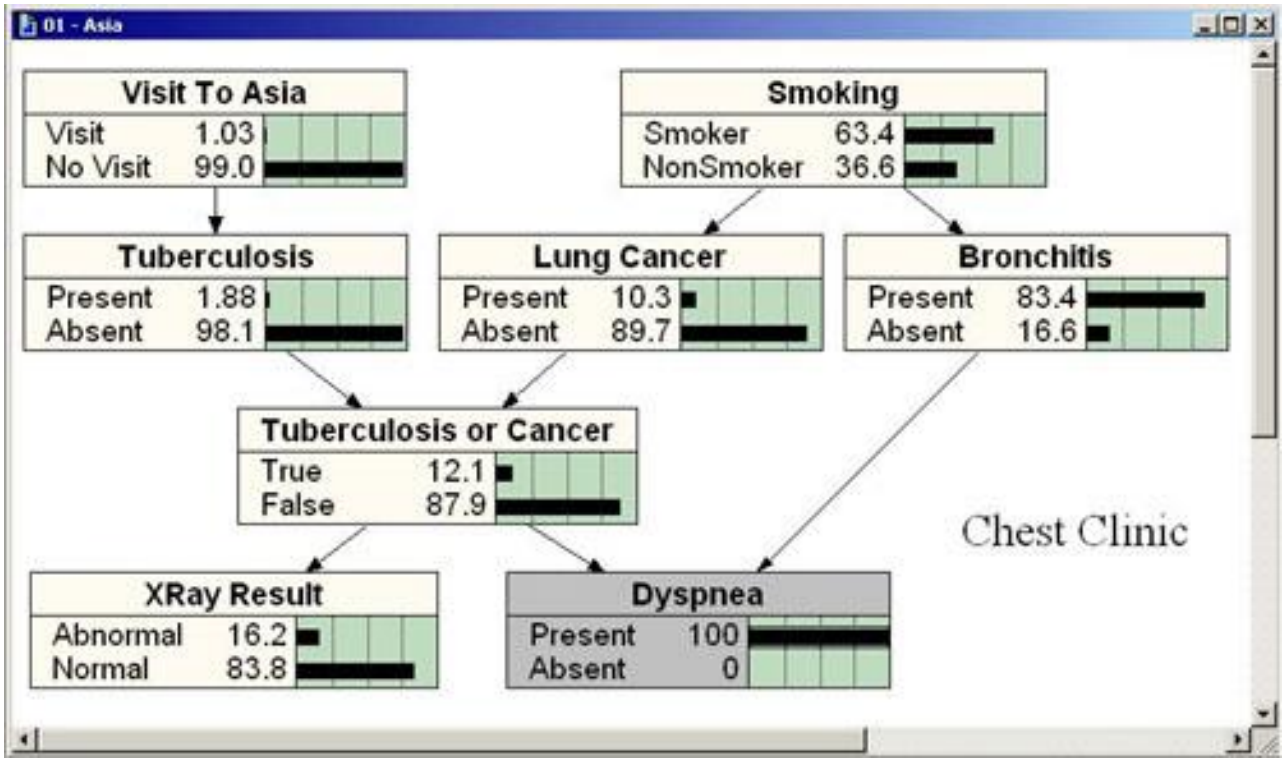
Continued…

50% of your patients smoke. 1% have TB. 5.5% have lung cancer.

45% have some form of mild or chronic bronchitis.

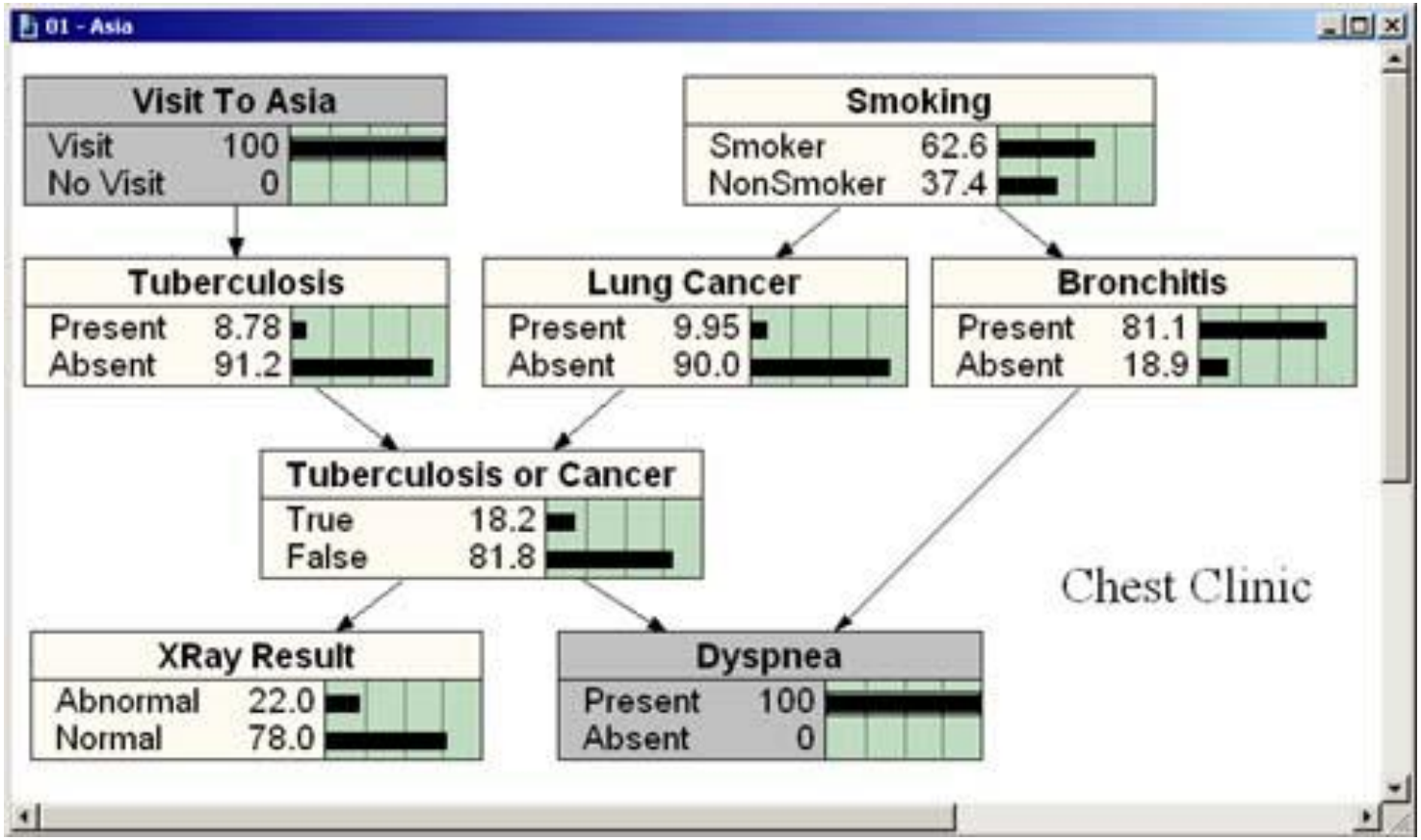


New patient with short of breath



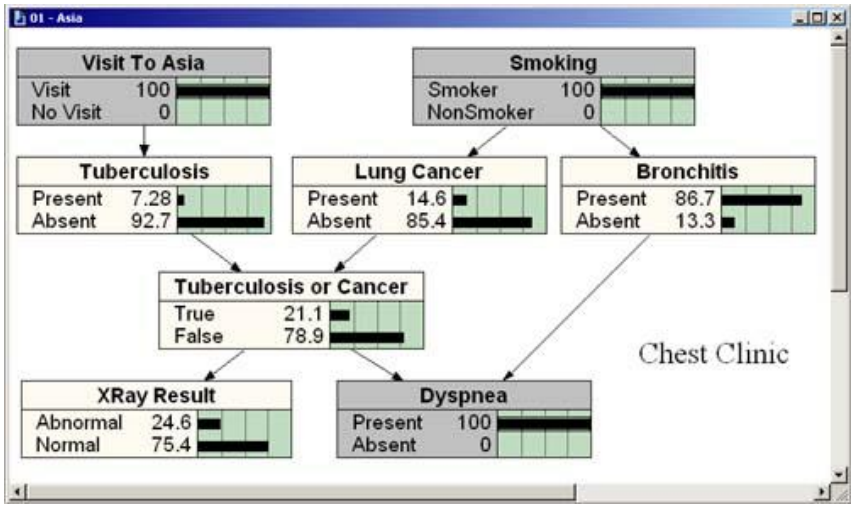
- bronchitis is far more common than cancer or TB
- Some of our beliefs are increased substantially, others hardly at all.
- And the beauty of it is that the amounts are precisely quantified.

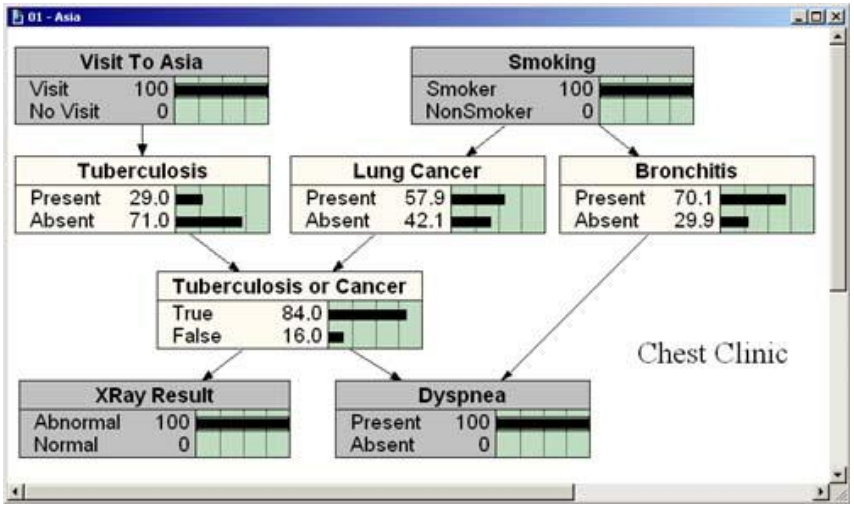
If she has been to Asia recently?



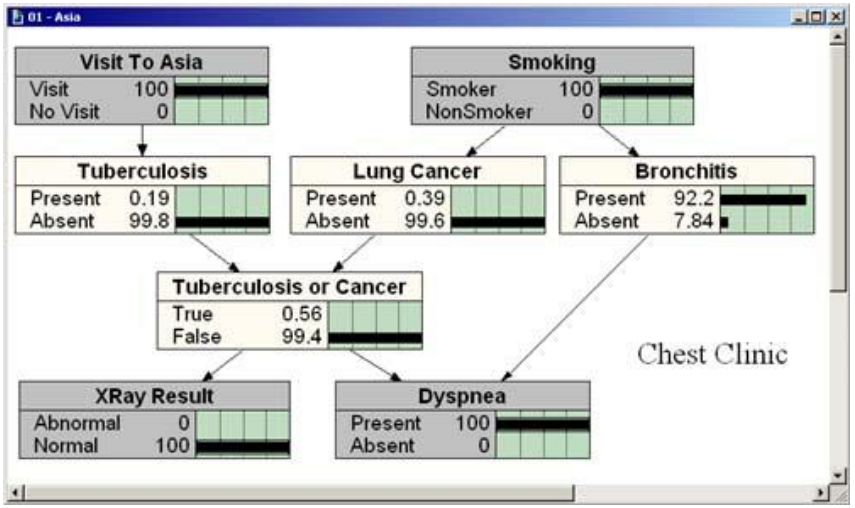
Explaining away: lung cancer, bronchitis, smoking

If patient is indeed a smoker?



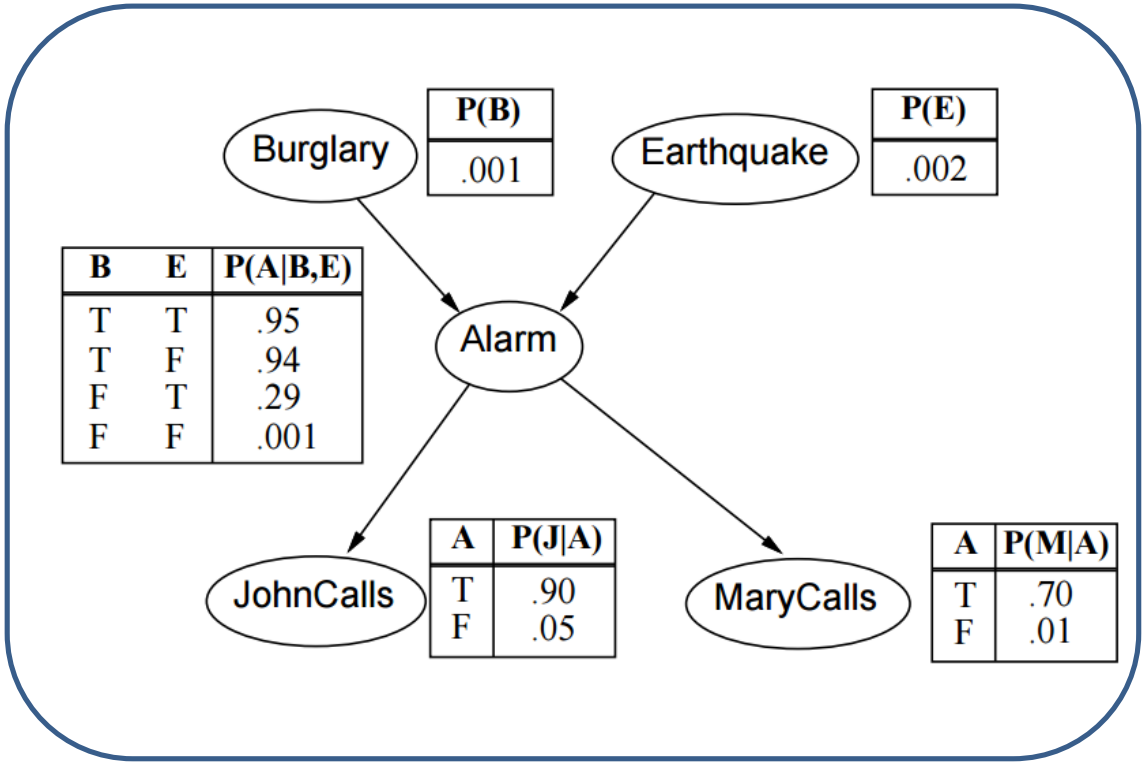


order a diagnostic X-Ray



Order more tests like blood test, long tissue biopsies,…

Bayesian Networks: Burglary-Alarm Ex.



Joint probability distribution:

$$P(X_1, X_2, \ldots) = \prod_{i=1,..n} P(X_i \mid pa(X_i))$$

P(J, ~M, A, ~B, ~E)
=
P(J|A).P(~M|A).P(A|~B,~E).
P(~B).P(~E)

= .9 X .3 X .001 X .999 X .998

= 0.00027

Find out P(J, M, A, ~B, ~E)? → 0.00063

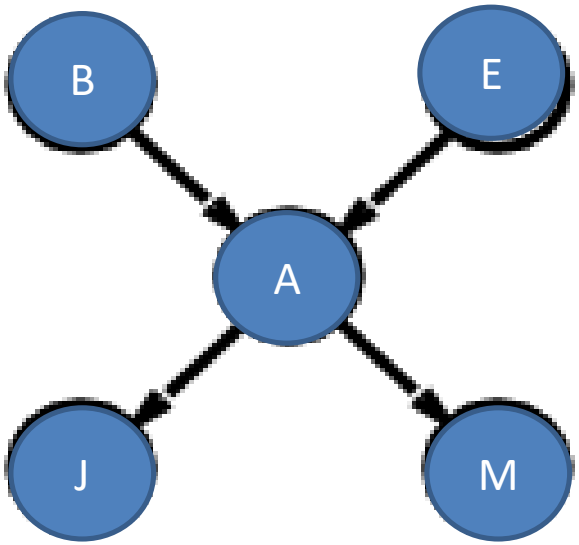
P(J,~M, A, B, ~E) =
P(J|A).P(~M|A).P(A|B,~E).P(B).P(~E) = .00025

Each variable is conditionally independent of all its non-descendants in the graph given the value of all its parents.

Independence relations in BBN

- If X and Y are connected by an edge, then X and Y are dependent
- Burglary and Alarm are what?

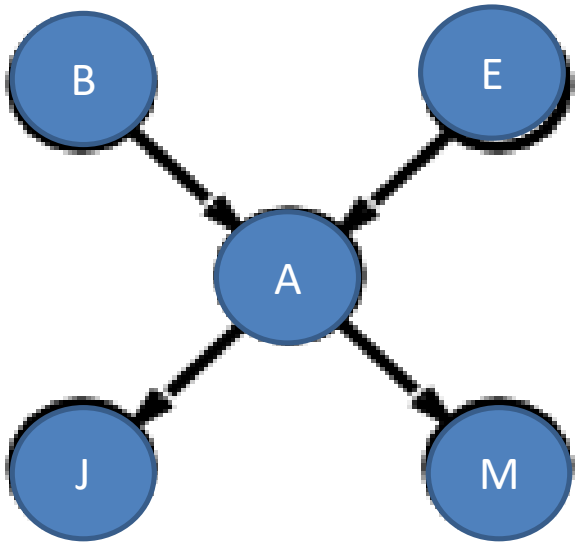
(direct connections)

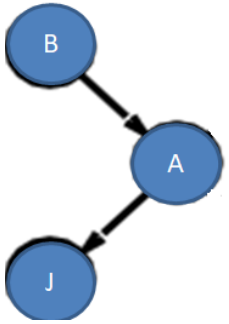


- Knowing B has taken place, increases the belief that the A has gone off. (Vice versa)

Serial Connections: BBN

- If A is not observed, then how are B and M related?
- Knowing that B has taken place, will you not believe more on M. (vice versa)





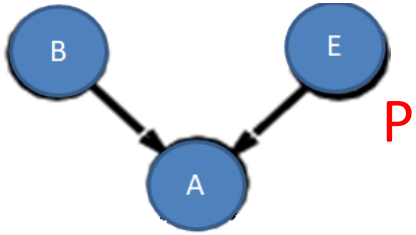
$P(J \mid A, B) = ?$ $P(J|A)$

$P(J, B \mid A) = ?$ $P(J|A)\ P(B|A)$

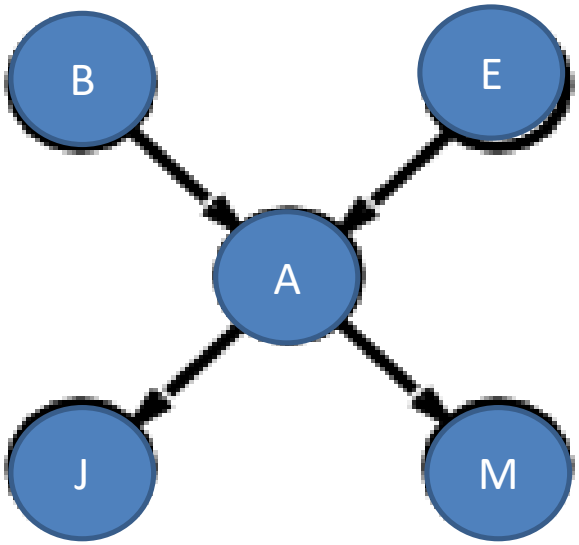
- If A is observed, then how are B and M related?
- If you know that A went off, will further knowing that B has taken place increase the belief on M? (vice versa)

Converging Connections: BBN

- If A is observed, then B and E are conditionally dependent.
- Knowing that A has taken place, will you believe less or more on E. (vice versa)



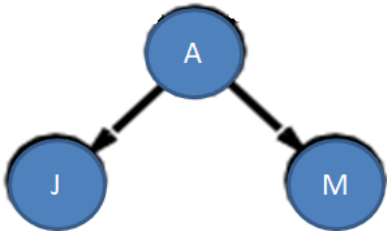
P(B,E) = ? P(B)XP(E)



- If M is observed, then also B and E are conditionally related.
- If A, J, and M are not observed, B and E are marginally independent.

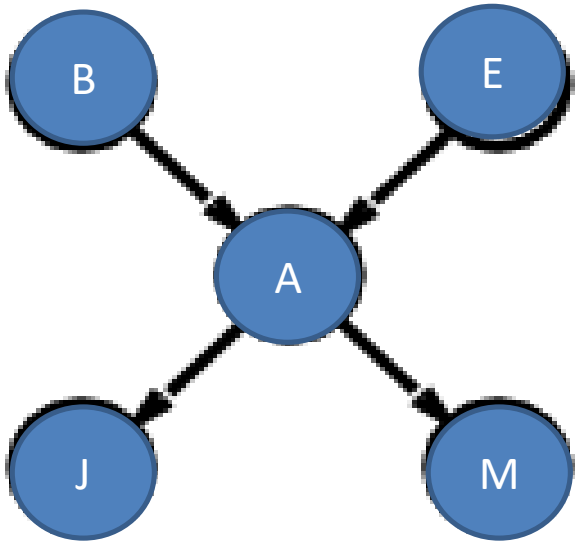
Diverging Connections: BBN

- If A is not observed, then how are say, J and M related?
- Knowing that J has taken place, will you not believe more on M. (vice versa)



$P(J \mid A,M) = ?$ $P(J|A)$

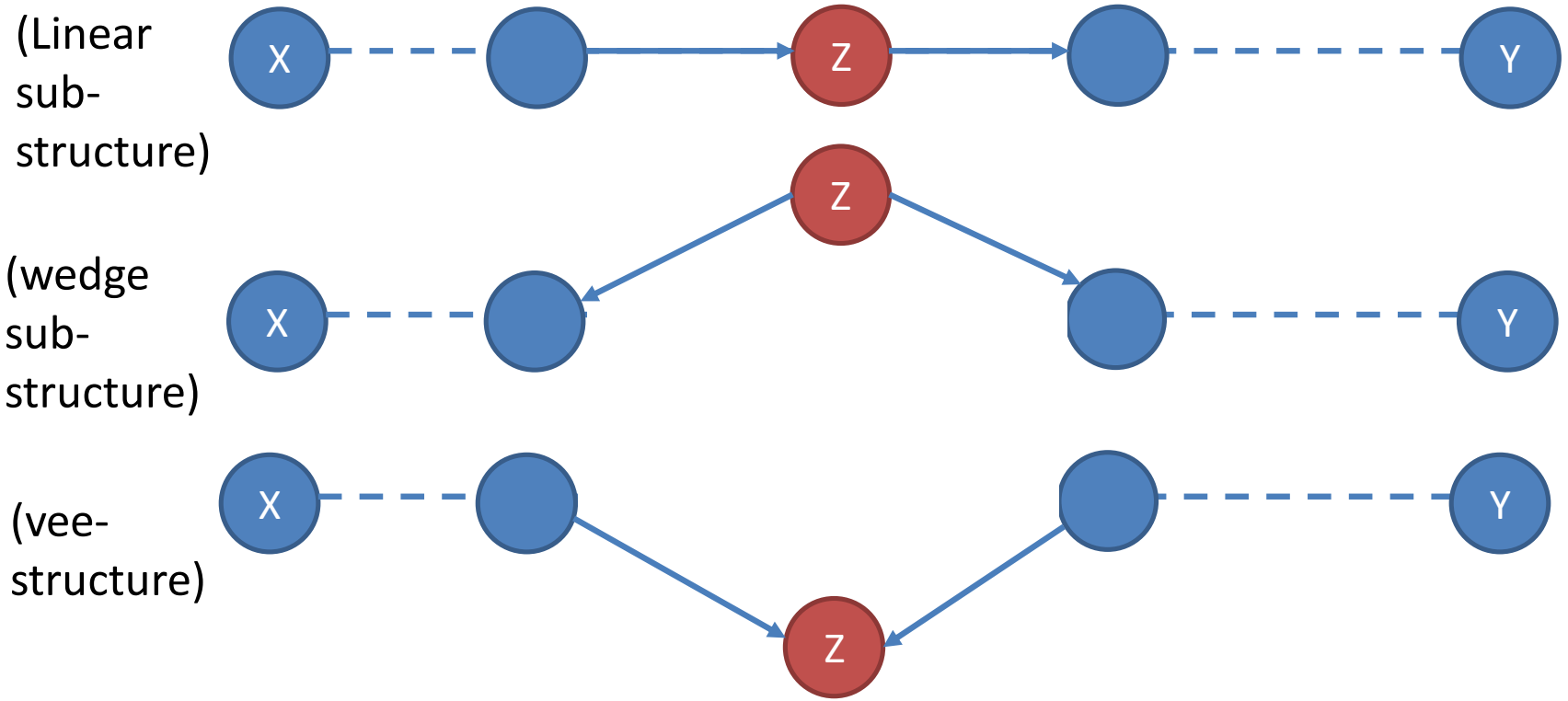
$P(J, M \mid A) = ?$ $P(J|A)\,P(M|A)$



- If A is observed, then how are J and M related?
- If you know that A went off, will further knowing that J has called increase the belief on M? (vice versa)

D-separation in Bayesian Belief Networks

- Conditional independence relations amongst different variables are defined in terms of graphical criteria, called d-separation.
- X is d-separated from Y given Z if every un-directed path between them is blocked by Z.



D-separation in Bayesian Belief Networks

1. Draw the “ancestral graph” (reduced version of original network consisting of parent, parent’s parent, …).
2. “Moralize” the ancestral graph by “marrying” the parents.
3. "Disorient" the graph by replacing the directed edges (arrows) with undirected edges (lines).
4. Delete the givens and their edges.

If the variables are **disconnected** in this graph, they are guaranteed to be independent

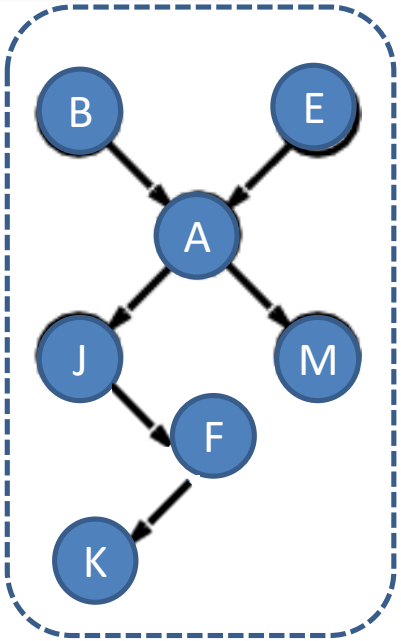
If the variables are **connected** in this graph, they are not guaranteed to be independent.

If one or both of the variables are missing (because they were givens, and were therefore deleted), they are independent.

An example

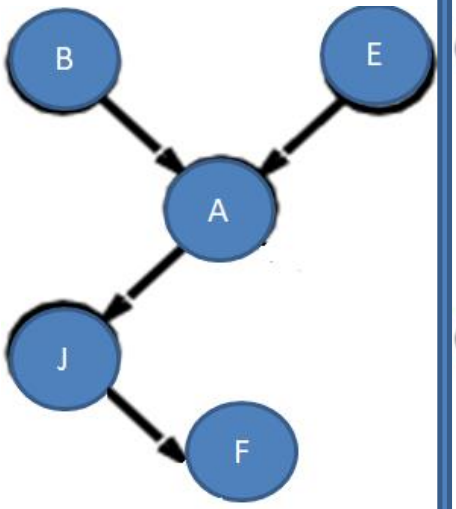
Are B and E marginally independent? P(B|E) = P(B), P(E|B)=P(E)

Yes

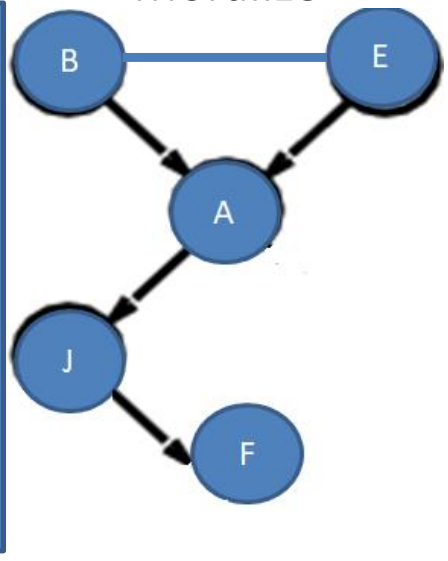


Question: Are B and E conditionally independent, given J and F?

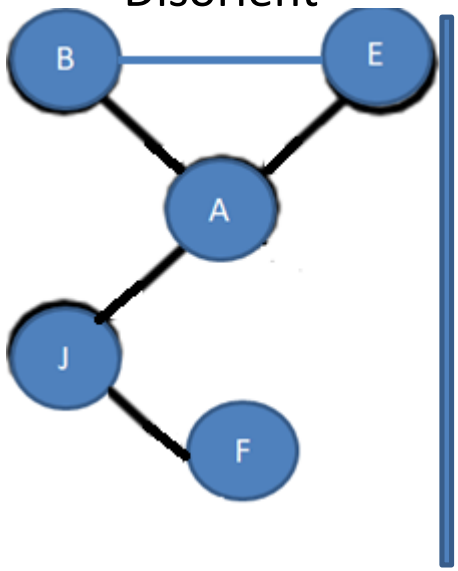
Ancestral graph



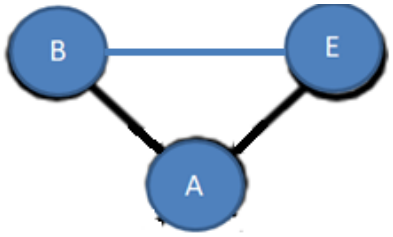
Moralize



Disorient



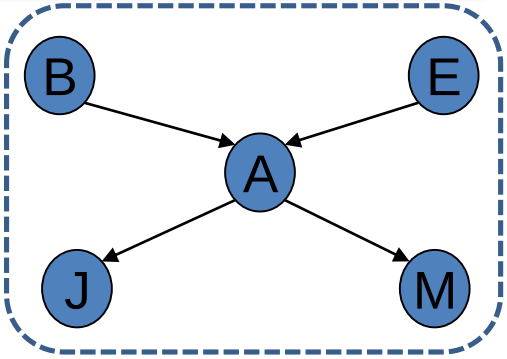
Delete givens



As B and E are connected they are not conditionally independent given J and F.

Bayes Network Construction

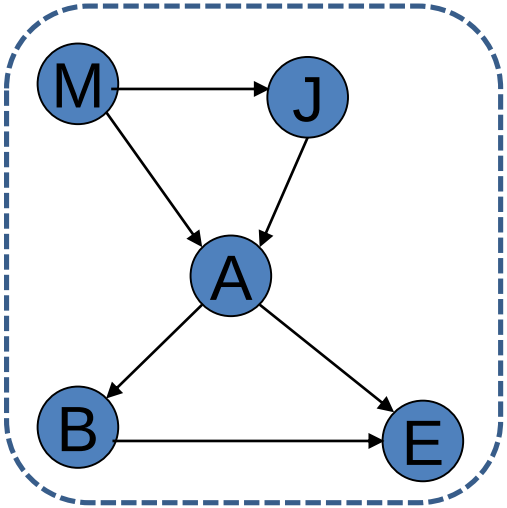
- Choose a set of variables describing the application domain
- Choose an ordering of variables
- Start with empty network and add variables to the network one by one according to the ordering



Order: B, E, A, J, M
pa(B)=pa(E)={}, pa(A)={B,E},
pa(J)={A}, pa{M}={A}

- To add i-th variable X_i:
  - Determine $pa(X_i)$ of variables already in the network $(X_1, \ldots, X_{i-1})$ such that:
    $P(X_i \mid X_1, \ldots, X_{i-1}) = P(X_i \mid pa(X_i))$
    (domain knowledge is needed here)
  - Draw an arc from each variable in $pa(X_i)$ to X_i

Order: M, J, A, B, E
pa{M}={}, pa{J}={M}, pa{A}={M,J}, pa{B}={A}, pa{E}={A,B}



TensorFlow (Bayesian Network)

```python
import tensorflow_probability as tfp
import tensorflow as tf

# Alias for distributions
tfd = tfp.distributions

# Define a Bayesian Network using JointDistributionCoroutine
def bayesian_network():
    # Node A: Prior distribution (no parents)
    A = yield tfd.Bernoulli(probs=0.3, dtype=tf.int32, name="A")  # A is 1 with probability 0.3

    # Node B: Conditional on A
    B = yield tfd.Bernoulli(probs=tf.where(A == 1, 0.8, 0.2), dtype=tf.int32, name="B")  # B depends on A

    # Node C: Conditional on both A and B
    C = yield tfd.Bernoulli(probs=tf.where(A == 1, tf.where(B == 1, 0.9, 0.4), 0.1), dtype=tf.int32, name="C")

# Create the joint distribution for the Bayesian Network
joint_dist = tfd.JointDistributionCoroutine(bayesian_network)

# Sample from the Bayesian Network
samples = joint_dist.sample(10)  # Sample 10 instances

# Display the results
print("Samples of A, B, C:")
print(samples)
```

```python
import tensorflow_probability as tfp
import tensorflow as tf

# Alias for distributions
tfd = tfp.distributions

# Define the Bayesian Network structure
def bayesian_network():
    A = yield tfd.Bernoulli(probs=0.3, dtype=tf.int32, name="A")  # Prior for A
    B = yield tfd.Bernoulli(probs=tf.where(A == 1, 0.8, 0.2), dtype=tf.int32, name="B")  # B conditional on A
    C = yield tfd.Bernoulli(probs=tf.where(A == 1, tf.where(B == 1, 0.9, 0.4), 0.1), dtype=tf.int32, name="C")
```

Other methods: Markov Chain Monte Carlo

```python
# Convert A and B to tf.Variables for optimization
A = tf.Variable(initial_value=0, dtype=tf.int32)
B = tf.Variable(initial_value=0, dtype=tf.int32)

# Function to compute the log probability of A, B given C
def log_prob_fn():
    return joint_dist.log_prob(a=A, b=B, c=observed_C)

# Use gradient ascent to maximize the log probability
optimizer = tf.keras.optimizers.Adam(learning_rate=0.1)

# Run optimization
for step in range(100):  # Perform 100 steps of optimization
    with tf.GradientTape() as tape:
        # Compute the negative log-probability (since we're minimizing)
        neg_log_prob = -log_prob_fn()

    # Compute the gradients of the log probability w.r.t. A and B
    grads = tape.gradient(neg_log_prob, [A, B])

    # Apply the gradients to A and B
    optimizer.apply_gradients(zip(grads, [A, B]))

    if step % 10 == 0:
        print(f"Step {step}: A={A.numpy()}, B={B.numpy()}, log_prob={-neg_log_prob.numpy()}")
```

Minimizing the negative log probability

Bayes to Naïve Bayes: Why?

How to choose a hypothesis?

(Recap)

Maximum A Posteriori = $\underset{h \in H}{\operatorname{argmax}}\ P(D|h).\ P(h)$

- Given training data, we can learn/estimate P(D|h) and P(h). Estimating P(h) is easy as we need to count the number of times 'h' appears in the training data.
- However, finding P(D|h) i.e the Likelihood function becomes too complex in real-world data with high dimensions because of large dependencies.
- Naive Bayes classifiers make the "naive" assumption that the features are conditionally independent given the class label. Mathematically, this can be expressed as:

$$P(D|h) = P(x_1|h) \times P(x_2|h) \times \ \ldots \ \times P(x_n|h) = \prod_{i=1}^{n} P(x_i|h)$$

Maximum A Posteriori = $\underset{h \in H}{\operatorname{argmax}}\ P(h) \prod_{i=1}^{n} P(x_i|h)$

Feature values are Independent given the target value: Strong assumption and unrealistic for real data. It is Naïve because it is (almost) never true.

Example Naïve Bayes: Play Tennis

Temperature	Play=Yes	Play=No
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Outlook	Play=Yes	Play=No
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Wind	Play=Yes	Play=No
Strong	3/9	3/5
Weak	6/9	2/5

Humidity	Play=Yes	Play=No
High	3/9	4/5
Normal	6/9	1/5

P(Play=Yes) = 9/14 P(Play=No) = 5/14

(Learning) (Testing)

Given: x'= (Outlook=Sunny, temperature=Cool, Humidity=High, Wind=Strong)

P(Yes|x'): [P(Sunny|Yes)P(Cool|Yes)P(High|Yes)P(Strong|Yes)] P(Play=*Yes*) = 0.0053 **No**

P(No|x'): [P(Sunny|No) P(Cool|No)P(High|No)P(Strong|No)] P(Play=No) = 0.0206 **Play**

Example Naïve Bayes: Play Tennis

Temperature	Play=Yes	Play=No
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Outlook	Play=Yes	Play=No
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Wind	Play=Yes	Play=No
Strong	3/9	3/5
Weak	6/9	2/5

Humidity	Play=Yes	Play=No
High	3/9	4/5
Normal	6/9	1/5

P(Play=Yes) = 9/14 P(Play=No) = 5/14

(Learning) (Testing)

Given: x'= (Outlook=Sunny, temperature=Cool, Humidity=High, Wind=Strong)

P(Yes|x'): [P(Sunny|Yes)P(Cool|Yes)P(High|Yes)P(Strong|Yes)] P(Play=*Yes*) = 0.0053 **No**

P(No|x'): [P(Sunny|No) P(Cool|No)P(High|No)P(Strong|No)] P(Play=No) = 0.0206 **Play**

Types of Naïve Bayes

- There are different types of Naive Bayes classifiers because of variations in the probability distribution assumptions made about the data.
- Gaussian Naïve Bayes: Assumes that features follow a normal (Gaussian) distribution. This is suitable for continuous data.
- Ex: Predicting diabetes using age, glucose level, etc

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
```

$$P(x|C) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- Multinomial Naïve Bayes: Features represent different frequencies of events. Used for text classification tasks.

$$P(x|C) = \frac{n!}{x_1!x_2!...x_k!} \prod_{i=1}^{k} P(x_i|C)^{x_i}$$

```
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
```

- Example: classifying news articles into categories or determining if a message is spam.
- Bernoulli Naïve Bayes: Document classification with binary features. BernoulliNB()
- Complement Naive Bayes: Multinomial for Imbalanced datasets. ComplementNB()
- Categorical Naive Bayes: Customer behaviour (Ex). CategoricalNB()

Spam Classification Using Multinomial Naive Bayes

```
import pandas as pd
from sklearn.feature_
from sklearn.model_s
from sklearn.naive_b
from sklearn.metrics

# Sample dataset (ex
data = pd.DataFrame(
    'message': ["Fre

    'label': ['spam'
})

# Map labels to 1 (s
data['label'] = data

# Initialize CountVe
vectorizer = CountVe

# Transform the text
X = vectorizer.fit_t

# Target variable
y = data['label']
```

```
# Split the data into train and test sets
# Adjust the test_size to ensure enough data for training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
                                                    random_state=42)

# Initialize the Multinomial Naive Bayes classifier
nb = MultinomialNB()

# Fit the model on the training data
nb.fit(X_train, y_train)

# Predict labels for the test set
y_pred = nb.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Detailed classification report
print(classification_report(y_test, y_pred))
```

```
Accuracy: 50.00%
```

CountVectorizer: “The cat sat on the mat”→[2, 1, 1, 1, 1, 0] “The dog sat on the mat” [2, 0, 1, 1, 1, 1]

Examples using GaussianNB(): Assignment 4

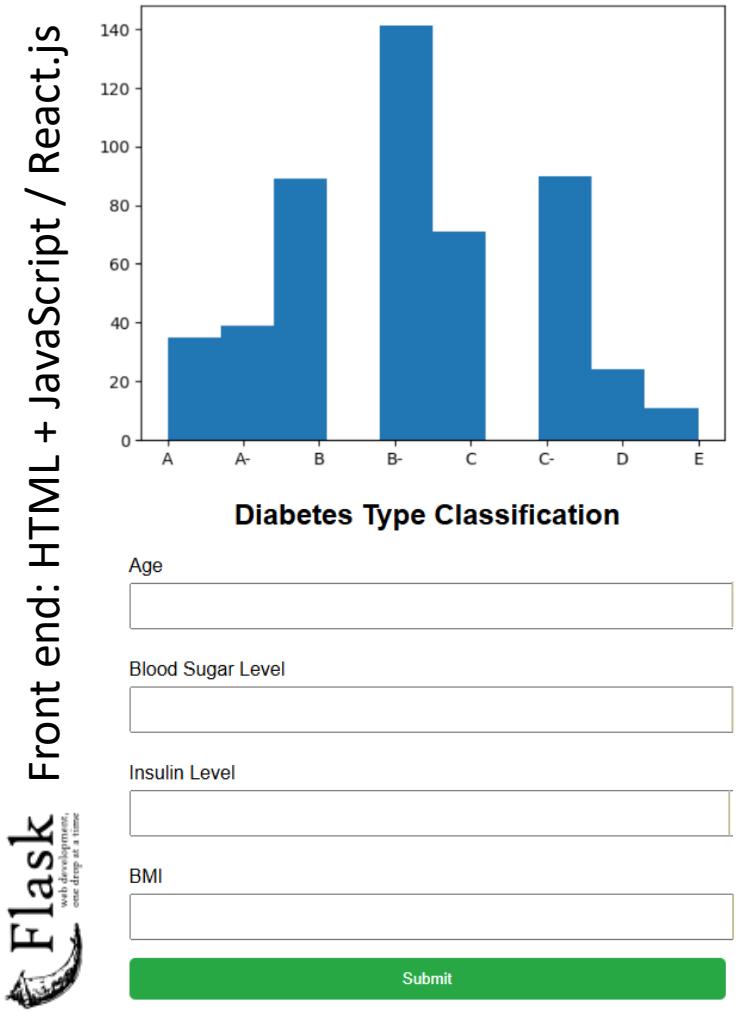
	Lab-Test1(30)	Lab-Test2(24)	Midsem Test (90)	Gender	Attendance	Grade
0	13.00	24	66.0	Male	High	A
1	15.00	24	67.0	Female	High	A
2	5.25	24	45.0	Male	High	B-
3	2.75	19	34.0	Male	High	C-
4	7.25	24	30.0	Male	High	C-

```
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
y_pred = nb_classifier.predict(X_test)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2
1	1.00	1.00	1.00	4
2	0.89	0.94	0.92	18
3	0.92	0.64	0.75	36
4	0.37	0.78	0.50	9
5	0.91	0.95	0.93	22
6	0.86	0.75	0.80	8
7	1.00	1.00	1.00	1
accuracy			0.81	100
macro avg	0.87	0.88	0.86	100
weighted avg	0.86	0.81	0.82	100

Age, blood sugar level, insulin level, and BMI

Front end: HTML + JavaScript / React.js



Thank You!