Idea: Estimate probabilities $\hat{P}(y|x)$ and use those for prediction

**Probability Classifier**

Input $x$: Sentence from review

Estimate class probability $\hat{P}(y = +1|x)$

If $\hat{P}(y = +1|x) > 0.5$:
- $\hat{y} = +1$

Else:
- $\hat{y} = -1$

**Notes:**

Estimating the probability improves *interpretability*
**Interpreting Score**

\[
Score(x_i) = w^T h(x_i)
\]

Very sure
\[
\hat{y}_i = -1
\]

\[
\hat{P}(y_i = +1 | x_i) = 0
\]

Very sure
\[
\hat{y}_i = +1
\]

\[
\hat{P}(y = +1 | x) = 1
\]

Not sure if
\[
\hat{y}_i = -1 \text{ or } \hat{y}_i = +1
\]

\[
\hat{P}(y_i = +1 | x_i) = 0.5
\]
\[ \hat{P}(y = +1|x, \hat{w}) = \text{sigmoid} \left( \hat{w}^T h(x) \right) = \frac{1}{1 + e^{-\hat{w}^T h(x)}} \]
Naïve Bayes
Idea: Naïve Bayes

**Idea:** Select the class that is the most likely!

**Bayes Rule:**

\[
P(y = +1|x) = \frac{P(x|y = +1)P(y = +1)}{P(x)}
\]

**Example**

\[
P(\text{“The sushi & everything else was awesome!”} | y = +1) \frac{P(y = +1)}{P(\text{“The sushi & everything else was awesome!”})}
\]

Since we’re just trying to find out which class has the greater probability, we can discard the divisor.
Naïve Assumption

Idea: Select the class with the highest probability!

Problem: We have not seen the sentence before.

Assumption: Words are independent from each other.

\[ x = \text{“The sushi & everything else was awesome!”} \]

\[
P(\text{“The sushi & everything else was awesome!”} | y = +1) \cdot P(y = +1) \]

\[
P(\text{“The sushi & everything else was awesome!”})
\]

\[
P(\text{“The sushi & everything else was awesome!”} | y = +1) = P(\text{The} | y = +1) \cdot P(\text{sushi} | y = +1) \cdot P(\& | y = +1) \cdot P(\text{everything} | y = +1) \cdot P(\text{else} | y = +1) \cdot P(\text{was} | y = +1) \cdot P(\text{awesome} | y = +1) \]
Compute Probabilities

How do we compute something like $P(y = +1)$?

How do we compute something like $P(\text{“awesome”} | y = +1)$?
Zeros

If a feature is missing in a class everything becomes zero.

\[
P(\text{"The sushi & everything else was awesome!"} \mid y = +1) \\
= P(\text{The} \mid y = +1) \times P(\text{sushi} \mid y = +1) \times P(\& \mid y = +1) \\
\times P(\text{everything} \mid y = +1) \times P(\text{else} \mid y = +1) \times P(\text{was} \mid y = +1) \\
\times P(\text{awesome} \mid y = +1)
\]

Solutions?

Take the log (product becomes a sum).
- Generally define \( \log(0) = 0 \) in these contexts

Laplacian Smoothing (adding a constant to avoid multiplying by zero)
Logistic Regression:

\[
P(y = +1|x, w) = \frac{1}{1 + e^{-w^T h(x)}}
\]

Naïve Bayes:

\[
P(y|x_1, x_2, ..., x_d) = \prod_{j=1}^{d} P(x_j|y) \cdot P(y)
\]
Compare Models

**Generative:** defines a distribution for generating $x$ (e.g. Naïve Bayes)

**Discriminative:** only cares about defining and optimizing a decision boundary (e.g. Logistic Regression)
How do we make decisions?

Non-linear decision boundaries

A line might not always support our decisions.
What makes a loan risky?

I want to buy a new house!

Credit History ★★★★★
Income ★★★
Term ★★★★★★
Personal Info ★★★

Loan Application
Credit history explained

Did I pay previous loans on time?

**Example:** excellent, good, or fair
What’s my income?

Example:
$80K per year

Credit History ★★★★★
Income ★★★
Term ★★★★★★
Personal Info ★★★
How soon do I need to pay the loan?

Example: 3 years, 5 years,...
Personal information

Age, reason for the loan, marital status,…

Example: Home loan for a married couple
Intelligent application
Classifier review

Loan Application

Input: $x_i$

Classifier MODEL

Output: $\hat{y}$
Predicted class

$\hat{y}_i = +1$
Safe

$\hat{y}_i = -1$
Risky
Setup

Data (N observations, 3 features)

<table>
<thead>
<tr>
<th>Credit</th>
<th>Term</th>
<th>Income</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>excellent</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>5 yrs</td>
<td>low</td>
<td>risky</td>
</tr>
<tr>
<td>fair</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
<tr>
<td>poor</td>
<td>5 yrs</td>
<td>high</td>
<td>risky</td>
</tr>
<tr>
<td>excellent</td>
<td>3 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>5 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>poor</td>
<td>3 yrs</td>
<td>high</td>
<td>risky</td>
</tr>
<tr>
<td>poor</td>
<td>5 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
</tbody>
</table>

Evaluation: classification error
Many possible decisions: number of trees grows exponentially!
Decision Trees

- **Branch/Internal node:** splits into possible values of a feature
- **Leaf node:** final decision (the class value)

Start

- Credit?
  - excellent: Safe
  - fair: Term?
    - 3 years: Risky
    - 5 years: Safe
  - poor: Income?
    - high: Term?
      - 3 years: Risky
      - 5 years: Safe
    - Low: Risky
Growing Trees

- Grow the trees using a greedy approach
- What do we need?
Visual Notation

Loan status: Safe Risky

Root

6 3 # of Risky loans

# of Safe loans

N = 9 examples
Decision stump: 1 level

Loan status:
Safe Risky

Split on Credit

<table>
<thead>
<tr>
<th>Credit</th>
<th>Term</th>
<th>Income</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>excellent</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>5 yrs</td>
<td>low</td>
<td>risky</td>
</tr>
<tr>
<td>fair</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
<tr>
<td>poor</td>
<td>5 yrs</td>
<td>high</td>
<td>risky</td>
</tr>
<tr>
<td>excellent</td>
<td>3 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>5 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>poor</td>
<td>3 yrs</td>
<td>high</td>
<td>risky</td>
</tr>
<tr>
<td>poor</td>
<td>5 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
</tbody>
</table>

Subset of data with Credit = excellent
Subset of data with Credit = fair
Subset of data with Credit = poor
Making predictions

For each leaf node, set $\hat{y} = \text{majority value}$

**Loan status:**
- Safe
- Risky

### Loan Status Decision Tree

- **Root**
  - $6 \ 3$
  - **credit?**

- **excellent**
  - $2 \ 0$
  - **Safe**

- **fair**
  - $3 \ 1$
  - **Safe**

- **poor**
  - $1 \ 2$
  - **Risky**
How do we select the best feature?

* Select the split with lowest classification error

**Choice 1: Split on Credit**

- **Loan status:** Safe, Risky

- **Root:**
  - Credit? (6 3)

- **Branches:**
  - *excellent*
    - 2 0
  - *fair*
    - 3 1
  - *poor*
    - 1 2

**Choice 2: Split on Term**

- **Loan status:** Safe, Risky

- **Root:**
  - Term? (6 3)

- **Branches:**
  - **3 years**
  - **5 years**
Calculate the node values.

<table>
<thead>
<tr>
<th>Credit</th>
<th>Term</th>
<th>Income</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>excellent</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>5 yrs</td>
<td>low</td>
<td>risky</td>
</tr>
<tr>
<td>fair</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
<tr>
<td>poor</td>
<td>5 yrs</td>
<td>high</td>
<td>risky</td>
</tr>
<tr>
<td>excellent</td>
<td>3 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>5 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>poor</td>
<td>3 yrs</td>
<td>high</td>
<td>risky</td>
</tr>
<tr>
<td>poor</td>
<td>5 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
</tbody>
</table>

Choice 2: Split on Term

Loan status:
Safe  Risky

Root
6 3

Term?

3 years
5 years
How do we select the best feature?

Select the split with lowest classification error

**Choice 1: Split on Credit**

- **Loan status:** Safe  Risky
- **Root:** 6 3
- **Credit?**
  - **excellent:** 2 0
  - **fair:** 3 1
  - **poor:** 1 2

**Choice 2: Split on Term**

- **Loan status:** Safe  Risky
- **Root:** 6 3
- **Term?**
  - **3 years:** 4 1
  - **5 years:** 2 2
How do we measure effectiveness of a split?

Idea: Calculate classification error of this decision stump

Error at a node = \# mistakes in each child node / \# data points at the node

Loan status:
Safe  Risky

Credit?

excellent 2 0
fair 3 1
poor 1 2
Calculating classification error

Step 1: $\hat{y} =$ class of majority of data in node

Step 2: Calculate classification error of predicting $\hat{y}$ for this data

Loan status: Safe  Risky

<table>
<thead>
<tr>
<th>Tree</th>
<th>Classification error</th>
</tr>
</thead>
<tbody>
<tr>
<td>(root)</td>
<td>0.33</td>
</tr>
</tbody>
</table>

Root

6 correct

3 mistakes

$\hat{y} =$ majority class
Choice 1: Split on Credit history?

Does a split on Credit reduce classification error below 0.33?

Choice 1: Split on Credit

Loan status:
- Safe
- Risky

Root:
- 6
- 3

Credit?

- excellent (2, 0)
- fair (3, 1)
- poor (1, 2)
Split on Credit: Classification error

Choice 1: Split on Credit

Loan status:
Safe  Risky

Root
6 3

Credit?

excellent
2 0
Safe
0 mistakes

fair
3 1
Safe
1 mistake

Poor
1 2
Risky
1 mistake

Error = \frac{0+1+1}{9} = \frac{2}{9} = 0.22

Tree | Classification error
--- | ---
(root) | 0.33
Split on credit | 0.22
Choice 2: Split on Term

Loan status: Safe  Risky

Root
6
3

Term?

3 years
4
1
Safe

5 years
2
2
Risky
Evaluating the split on Term

Choice 2: Split on Term

Loan status: Safe Risky

Root
6 3

Term?

3 years
4 1

Safe
1 mistake

5 years
2 2

Risky
2 mistakes

Error = \frac{1 + 2}{9} = \frac{3}{9} = 0.33

Tree | Classification error
---|---
(root) | 0.33
Split on credit | 0.22
Split on term | 0.33
Choice 1 vs Choice 2: Comparing split on credit vs term

### Choice 1: Split on Credit

**Loan status:**
- Safe
- Risky

**Root:**
- 6 3

**Credit?**
- Excellent: 2 0
- Poor: 1 2

**Tree Classification error**
- (root): 0.33
- Split on credit: 0.22
- Split on loan term: 0.33

**WINNER**
- Choice 1: Split on Credit

### Choice 2: Split on Term

**Loan status:**
- Safe
- Risky

**Root:**
- 6 3

**Term?**
- 3 years: 4 1
- 5 years: 2 2

**Tree Classification error**
- (root): 0.33
- Split on credit: 0.22
- Split on loan term: 0.33

**WINNER**
- Choice 2: Split on Term
**Split (node)**

- Given a subset of data $M$ in node
- For each feature $h_i$:
  - Compute classification error for a split of $M$ according to feature $h_i$
  - Chose feature $h^*(x)$ with lowest classification error and expand the tree to include the children of current node after the split
Greedy & Recursive Algorithm

**BuildTree(node)**

- If the number of datapoints at the current node or the classification error is within a certain threshold:
  - Stop
- Else:
  - Split(node)
  - For child in node:
    - BuildTree(child)

- Decision Tree algorithm is **greedy**: It aims to optimize the classification error at each node. As a result, the final result won’t be globally optimal, but it guarantees computational efficiency
- Decision Tree algorithm is **recursive**: From the current node, if we decide to further expanding the tree, we will repeat the same operations in the child nodes.
Decision tree expansion: 1 level

**Loan status:**
- Safe
- Risky

**Credit?**

- **excellent**
  - Subset of data with Credit = excellent
  - 2 0

- **fair**
  - Subset of data with Credit = fair
  - 3 1

- **poor**
  - Subset of data with Credit = poor
  - 1 2

**Root**
- 6 3

**Split on Credit**
Stopping

- Stop if all points are in one class

**Loan status:**

- Safe
- Risky

**Root**

- 6
- 3

**Credit?**

- excellent
  - 2
  - 0
  - Safe

- fair
  - 3
  - 1

- poor
  - 2
  - 1

All data points are Safe nothing else to do with this subset of data
Expanding the trees by recursing on children nodes

Loan status:
Safe Risky

Root
6 3

Credit?

efficient
2 0

Safe

fair
3 1

Build a decision subtree with subset of data where Credit = fair

poor
2 1

Build a decision subtree with subset of data where Credit = poor
Loan status:
Safe Risky

Credit?
- excellent: Safe
- fair: Safe
- poor: Risky

Term?
- 3 years: Safe
- 5 years: Risky

Build another subtree for these subset data points
Different data types that decision trees support

<table>
<thead>
<tr>
<th>Income</th>
<th>Credit</th>
<th>Home ownership</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>$105 K</td>
<td>excellent</td>
<td>Rent</td>
<td>Safe</td>
</tr>
<tr>
<td>$112 K</td>
<td>good</td>
<td>Own</td>
<td>Risky</td>
</tr>
<tr>
<td>$73 K</td>
<td>fair</td>
<td>Rent</td>
<td>Safe</td>
</tr>
<tr>
<td>$69 K</td>
<td>excellent</td>
<td>Mortgage</td>
<td>Safe</td>
</tr>
<tr>
<td>$217 K</td>
<td>excellent</td>
<td>Own</td>
<td>Risky</td>
</tr>
<tr>
<td>$120 K</td>
<td>good</td>
<td>Mortgage</td>
<td>Safe</td>
</tr>
<tr>
<td>$64 K</td>
<td>fair</td>
<td>Own</td>
<td>Risky</td>
</tr>
<tr>
<td>$340 K</td>
<td>excellent</td>
<td>Own</td>
<td>Safe</td>
</tr>
<tr>
<td>$60 K</td>
<td>good</td>
<td>Other</td>
<td>Risky</td>
</tr>
</tbody>
</table>
We have been used to numerical features (such as number of bedrooms / bathrooms, income, are).

However, in practice, data comes from different forms.
- There are three main data types:
  - **Numeric**
  - **Categorical**: data that takes on a number of fixed possible values
    - **Ordinal**: data that have ordered categories
      E.g: credit quality (good / fair / bad)
    - **Nominal**: data that doesn’t have ordered categories
      E.g: home ownership (rent / own / mortgage)
- Reminder about the extra credit: Zip code is a categorically nominal variable
Transforming categorical features

Depending on implementations, decision trees might not need you to transform data of categorical types into numeric values. However, in models that use differentiable loss functions (like Linear Regression / Logistic Regression, some forms of decision trees), you need to transform categorical data:

- **Ordinal**: Transform into numbers of a certain order
  
  *e.g.* For 3 categories *(bad / fair / good)*
  
  
  \[
  \text{bad} = 0, \quad \text{fair} = 1, \quad \text{good} = 2
  \]

- **Nominal**: Using one-hot encoding

<table>
<thead>
<tr>
<th>id</th>
<th>color</th>
<th>color_red</th>
<th>color_blue</th>
<th>color_green</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>blue</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>green</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>blue</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Threshold split for numeric features

Loan status:
Safe Risky

Split on Income

Income?

< $60K
8 13

>= $60K
14 5

Subset of data with Income >= $60K
Best threshold?

Infinite possible values of $t$

Income < $t^*$

Income = $t^*$

Income >= $t^*$

Income

$10K$

$120K$

Safe

Risky
Threshold between points

Same classification error for any threshold split between \( v_A \) and \( v_B \)
Only need to consider mid-points

Finite number of splits to consider

Income

$10K $120K

Safe
Risky
Splitting for numeric features

**Step 1:** Sort the values of a feature $h_j(x)$:

Let $\{v_1, v_2, v_3, \ldots v_N\}$ denote sorted values

**Step 2:**
- For $i = 1 \ldots N-1$
  - Consider split $t_i = (v_i + v_{i+1}) / 2$
  - Compute classification error
- Chose the $t^*$ with the lowest classification error
Earlier, we learn how to learn decision trees based on classification error. However, this metric is not sensitive enough if two different splits give the same classification error.

In practice, people prefer using continuous loss functions for splitting decision (two algorithms like ID3 – entropy loss or CART – Gini impurity loss).
Visualizing the threshold split

Threshold split is the line $\text{Age} = 38$
Split on Age

>= 38

Income

age < 38

age >= 38

Predict Safe

Predict Risky
Each split partitions the 2-D space
Depth 1: Split on $x[1]$
Depth 2
Same feature can be used to split multiple times

For threshold splits, same feature can be used multiple times
Decision boundaries at different depths

Decision boundaries can be complex!
Advantages of Decision Tree

Advantages:
- Easy to interpret
- Can handle both continuous and categorical variables without preprocessing
- No normalization required as it uses rule-based approach
- Can create non-linear decision boundaries
- Can handle missing values
Disadvantages of Decision Tree

Disadvantages:
- Deep decision trees are prone to overfitting.
  - Decision boundaries are interpretable but not stable, because adding new datapoints will cause the trees to be regenerated.
  - Not suitable for large datasets due to the growing complexity
- Only allows axis-parallel rectangular decision boundaries
Overfitting prevention

Overcoming Overfitting:
- Set minimum number of data points in a node to split
- Early stopping
  - Fixed length depth
  - Maximum number of nodes
  - Stop if error does not considerably decrease
- Pruning

Tree Pruning Example
What you can do now:

- Define a decision tree classifier
- Interpret the output of a decision trees
- Learn a decision tree classifier using greedy & recursive algorithm
- Advantages and Disadvantages of a decision tree
- Understand different data types and necessary preprocessing steps
- Ways to overcome overfitting in decision trees