This Lecture Part 2

April 5th, 2021

Exam

Same, Take Home, This Week, Tuesday-Monday...LMK 72 hours

Thinking about the final project...

Choosing Data

- Robust enough data to really dig into things and see how distributions of values in various features may/may not impact results.
- Enough to appropriately train a good/great model and be able to discuss success/failures.
- More features (columns) is not always better, but too few is probably not great.
    - Asterisks... images are really just a matrix of pixel values 28x28 for example. So flattening out an image to 28x28=784 columns is what it actually is. Same for text...one sentences is many words (features).
- Usually the more rows the better unless you want to show results in a constrained data problem setting.

Choosing the Problem:

- Most of the datasets I will send out have a clearly marked column for label (supervised learning). This usually comes with a task - classify types of animals or something. (You can do other things but this should be a consideration)
- We will discuss more next week but you want to start thinking about what is important - accuracy, precision, recall, F1 score - in other words, what are the what case scenarios if you are wrong.

Choosing Other Data:

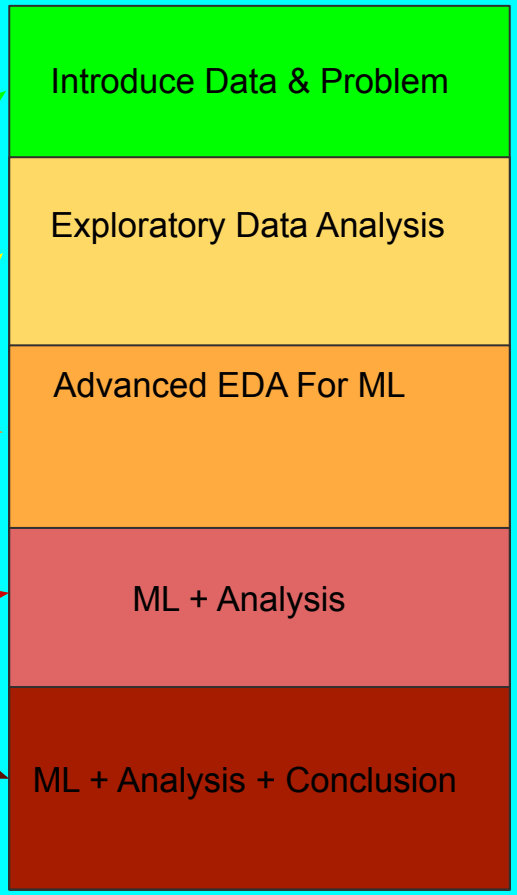
I highly recommend meeting with me or discussing via email why your dataset warrants this task and what problem (ML) you intend to solve with it. In form of HW or early/short 1-1.

Final Project Outline

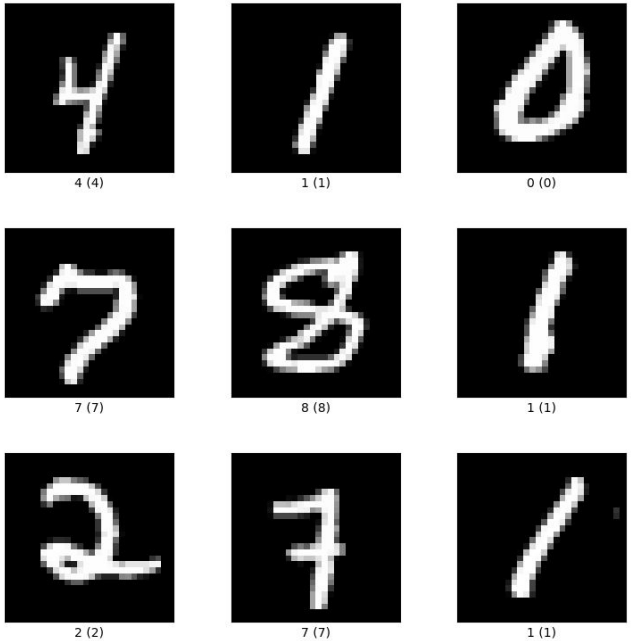
8 minute presentations to ensure completion on Final Day
HARD STOP...if you are well under...why...

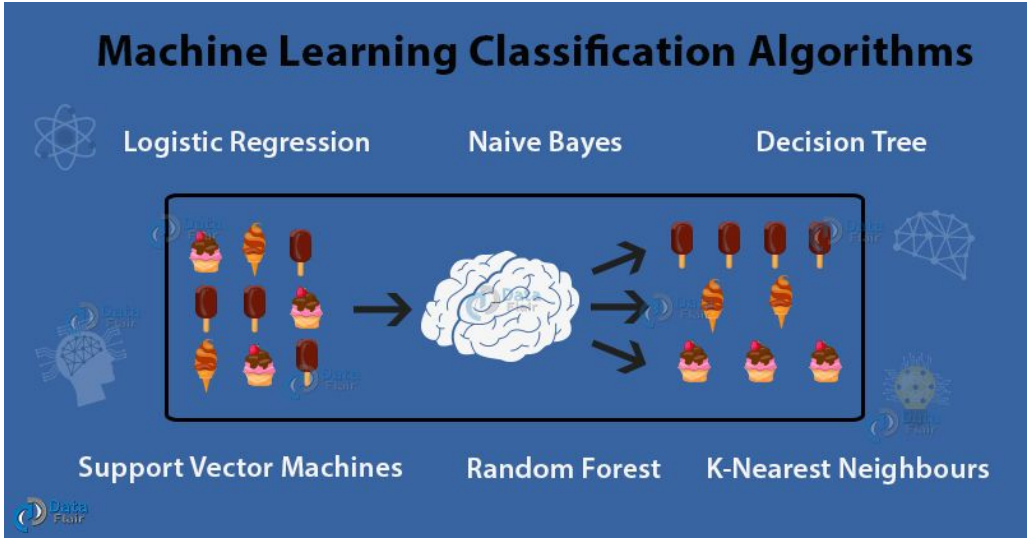
This is a suggested breakdown

- What is your data, what is your problem? 1min
- EDA - Midterm Presentations (what's in the data?) 1-2min
- EDA with a purpose for ML 1-2min
  - Distributions
  - Bootstrapping
  - Balancing
  - Train / Test / K Folds Splits?
- ML Baseline - what comes out of the box 1-2min
  - Describe the algorithm to us - teach me!
- ML Improvement - how did you improve a model? 1-2min
  - Comparisons from 'EDA with a purpose'
  - Explanations + teach me!
  - Wrap up - lessons learned

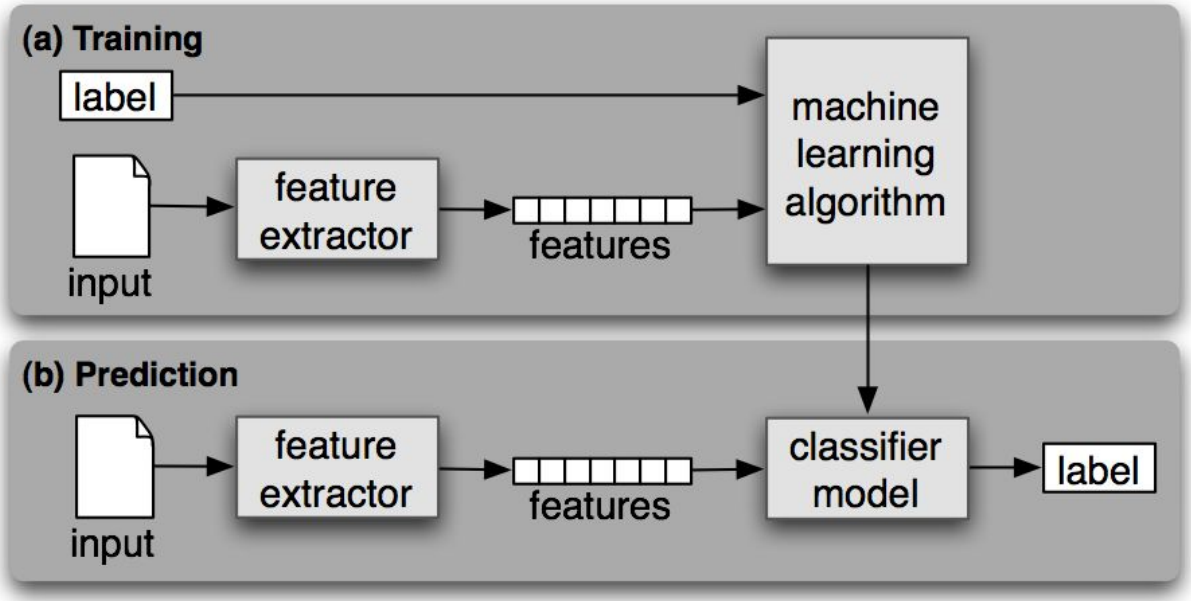


Machine Learning Notions





Recall our algorithm for algorithms



Remember - the algorithm for our algorithms is also an algorithm...or something like that...

In [24]:
```
from sklearn.svm import SVC

model_name = 'Kernel SVM Classifier'

svmClassifier = SVC(kernel='rbf', gamma= 'auto')

svm_model = Pipeline(steps=[('preprocessor', preprocessorForFeatures),('classifier', svmClassifier)])

svm_model.fit(X_train,y_train)

y_pred_svm = svm_model.predict(X_test)
```

In [18]:
```
from sklearn.linear_model import LogisticRegression

model_name = "Logistic Regression Classifier"

logisticRegressionClassifier = LogisticRegression(random_state=0,multi_class='auto',solver='lbfgs',max_iter=1000)

lrc_model = Pipeline(steps=[('preprocessor', preprocessorForCategoricalColumns),
                            ('classifier', logisticRegressionClassifier)])

lrc_model.fit(X_train,y_train)

y_pred_lrc = lrc_model.predict(X_test)
```

In [217]:
```
from sklearn.naive_bayes import GaussianNB
```

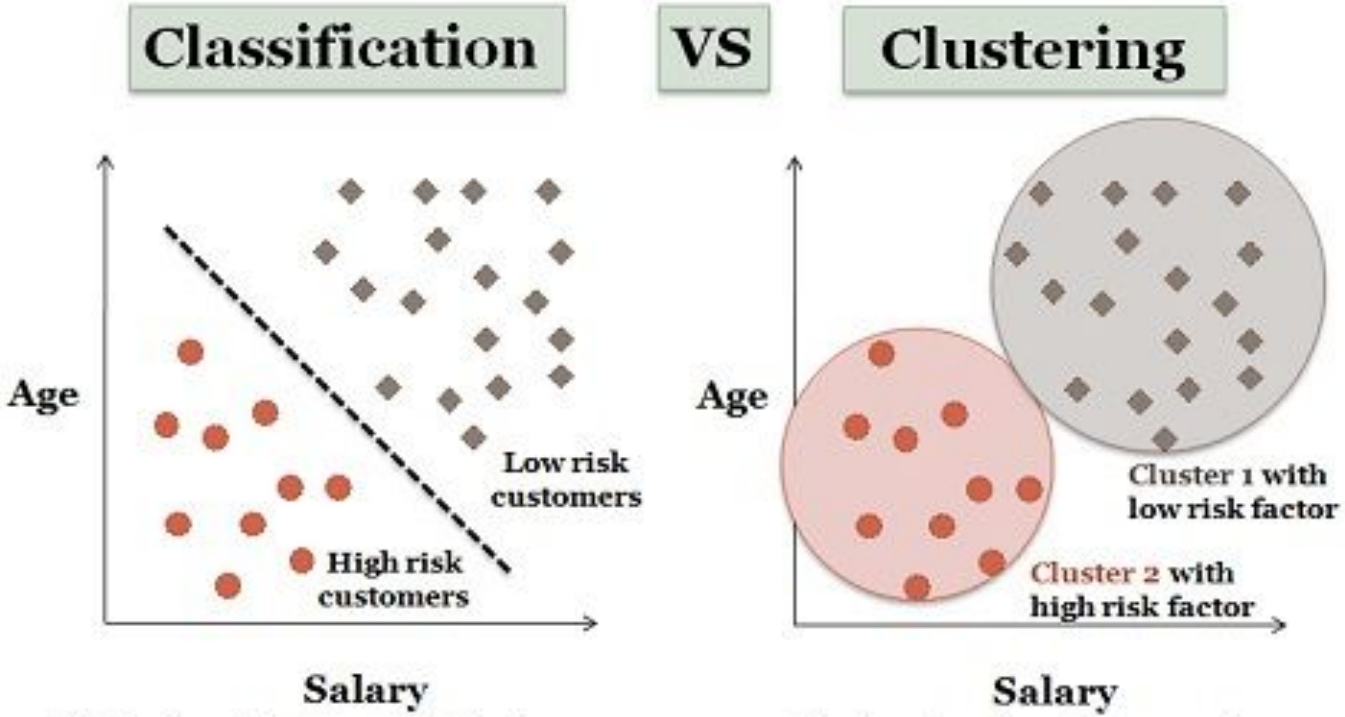
In [218]:
```
classifier = GaussianNB()
```

In [226]:
```
features = zip(data.W[:1600], data.ADJOE[:1600], data.WAB[:1600])
test = zip(data.W[1600:], data.ADJOE[1600:], data.WAB[1600:])
#classifier.fit(features, data.SEED[:1600])
classifier.fit(np.array(data.W[:1600]).reshape(-1,1), data.SEED[:1600].astype(int))
```

Out[226]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [222]:
```
preds = classifier.predict(test)
print(preds)
```

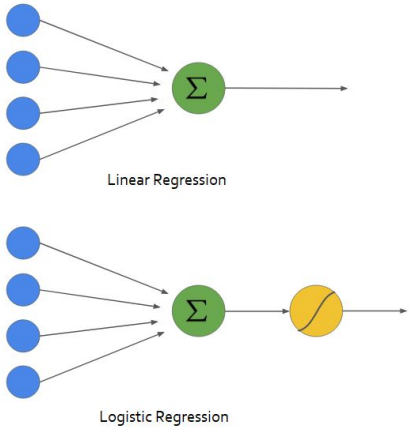
Classification vs. Clustering

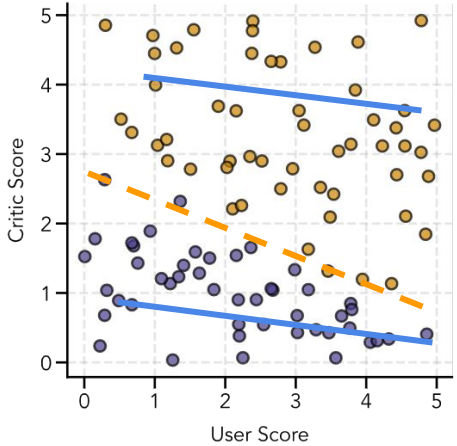


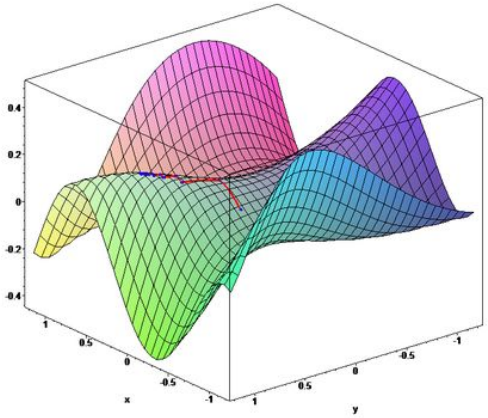
Risk classification for the loan payees on the basis of customer salary

Logistic Regression

$$\mathbf{0} = \beta 0 + \beta 1.x1 + \cdots + \beta n.xn$$







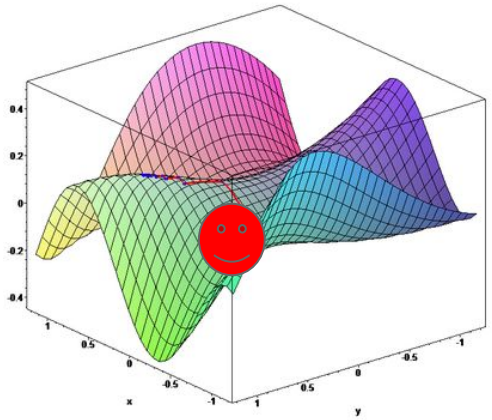
Gradient Descent - ML!

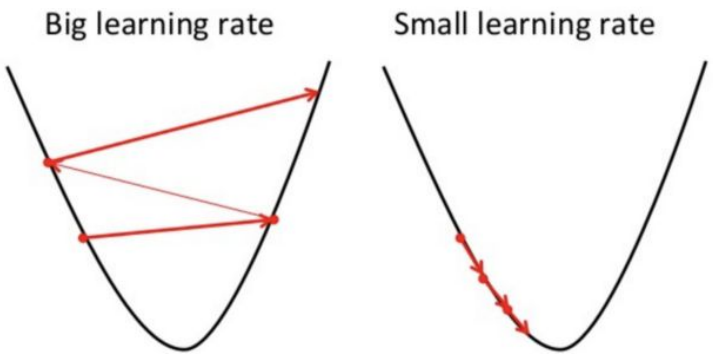
Baby Intro - Gradient Descent

$$J(w_0, w_1, w_2) = \frac{-1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(h_w(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_w(x^{(i)})) \right]$$

where our **hypothesis function** is:

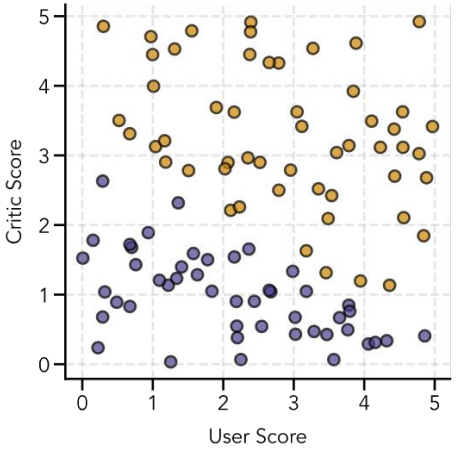
$$h_w(x_1^{(i)}, x_2^{(i)}) = \frac{1}{1 + \exp(-z)} \quad \text{where} \quad z = w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)}$$

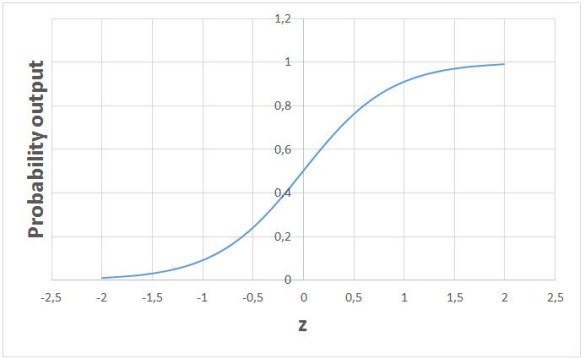


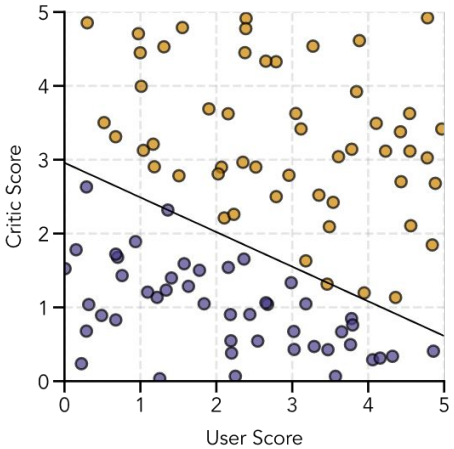


https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

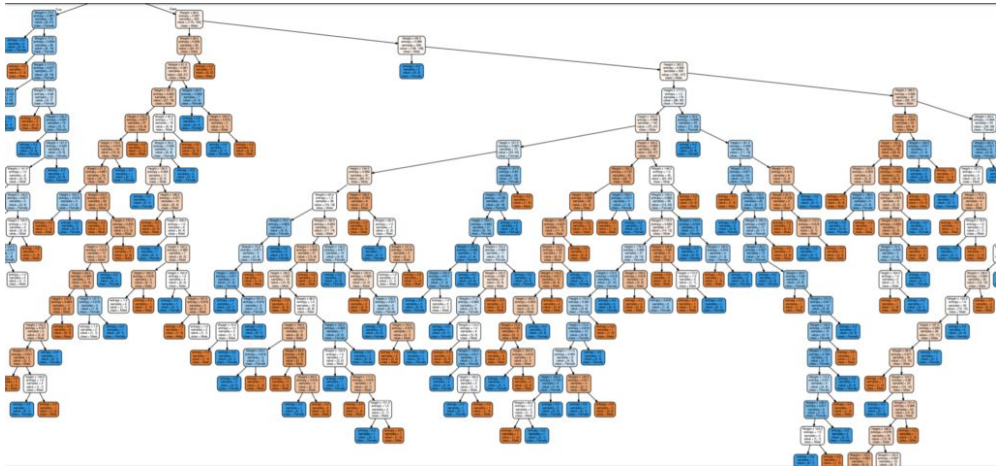
Logistic Regression Output

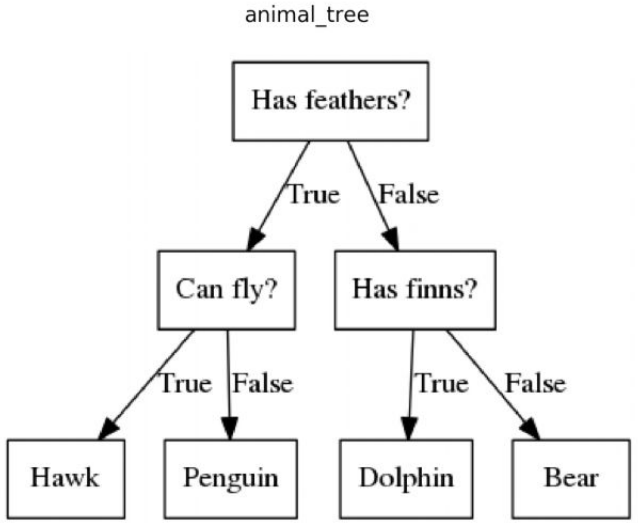






Decision Trees





Decision Tree Terminology

- **Parent node**: In any two connected nodes, the one which is higher hierarchically, is a parent node.
- **Child node**: In any two connected nodes, the one which is lower hierarchically, is a child node.
- **Root node**: The starting node from which the tree starts, It has only child nodes. The root node does not have a parent node. (dark blue node in the above image)
- **Leaf Node/leaf:** Nodes at the end of the tree, which do not have any children are leaf nodes or called simply leaf. (green nodes in the above image)
- **Internal nodes/nodes**: All the in-between the root node and the leaf nodes are internal nodes or simply called nodes. internal nodes have both a parent and at least one child. (red nodes in the above image)
- **Splitting**: Dividing a node into two or more sun-nodes or adding two or more children to a node.
- **Decision node**: when a parent splits into two or more children nodes then that node is called a decision node.
- **Pruning:** When we remove the sub-node of a decision node, it is called pruning. You can understand it as the opposite process of splitting.
- **Branch/Sub-tree**: a subsection of the entire tree is called a branch or sub-tree.

Gini Index - CART

Gini says, if we select two items from a population at random then they must be of the same class and the probability for this is 1 if the population is pure.

It works with the categorical target variable “Success” or “Failure”.

It performs only Binary splits

Higher the value of Gini higher the homogeneity.

CART (Classification and Regression Tree) uses the Gini method to create binary splits.

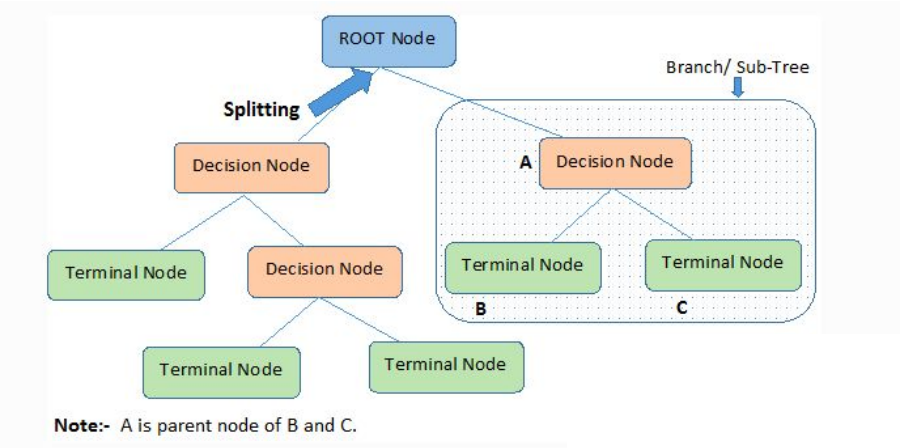
1. Calculate Gini impurity for sub-nodes, using the formula subtracting the sum of the square of probability for success and failure from one.
   $1-(p^2+q^2)$
   where p =P(Success) & q=P(Failure)
2. Calculate Gini for split using the weighted Gini score of each node of that split
3. Select the feature with the least Gini impurity for the split.

Information Gain - ID3

A less impure node requires less information to describe it and, a more impure node requires more information. Information theory is a measure to define this degree of disorganization in a system known as Entropy. If the sample is completely homogeneous, then the entropy is zero and if the sample is equally divided (50% — 50%), it has an entropy of one. Entropy is calculated as follows.

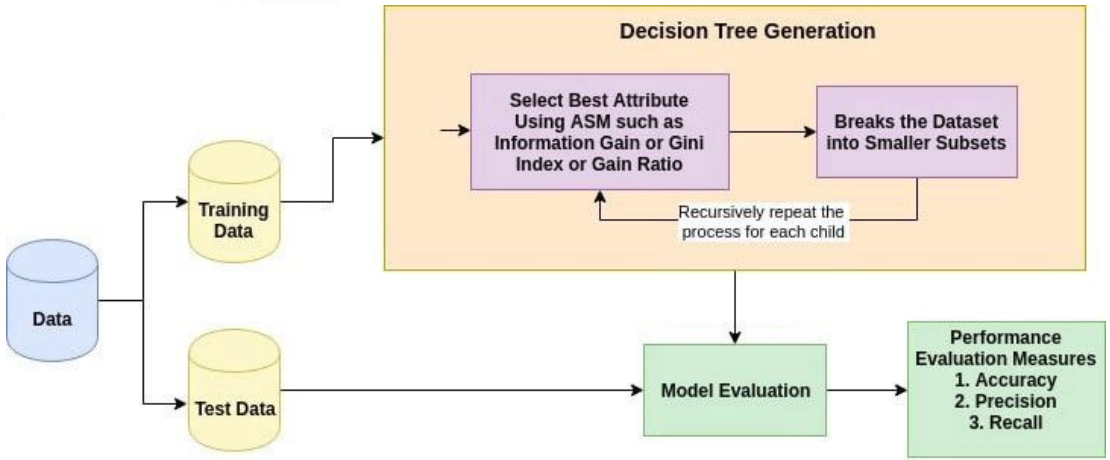
1. Calculate the entropy of the parent node
2. Calculate entropy of each individual node of split and calculate the weighted average of all sub-nodes available in the split. The lesser the entropy, the better it is.
3. calculate information gain as follows and chose the node with the highest information gain for splitting

Decision Tree Mechanics



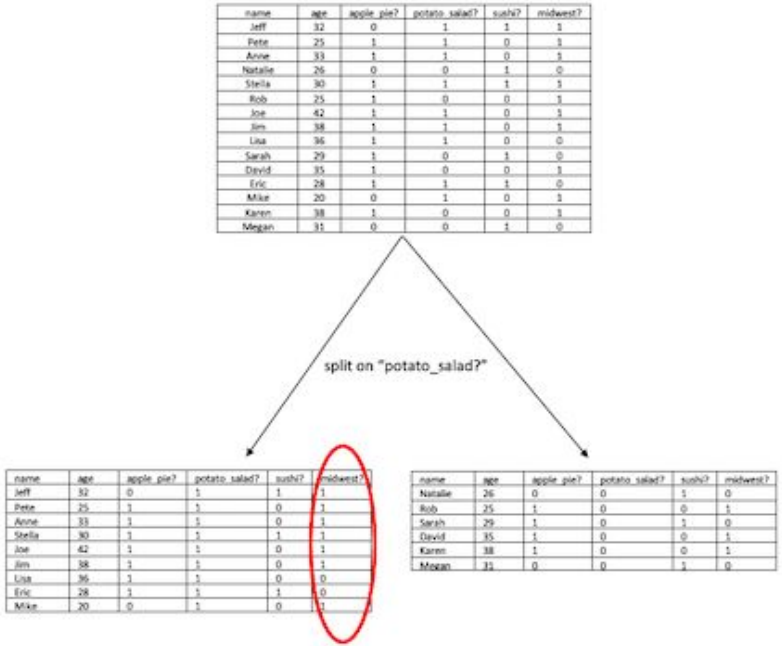
$$\text{Gini} = 1 - \sum_{i=1}^{n} (p_i)^2$$

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$



Quick Entropy Example

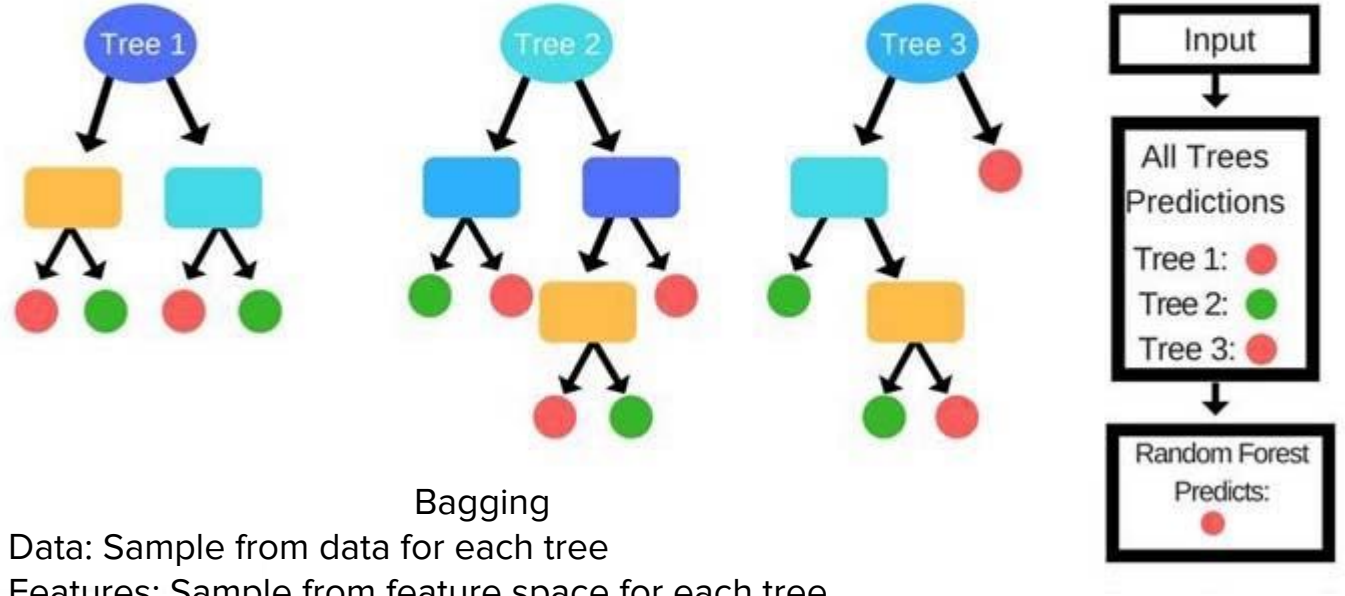
name	age	apple_pie?	potato_salad?	sushi?	midwest?
Jeff	32	0	1	1	1
Pete	25	1	1	0	1
Anne	33	1	1	0	1
Natalie	26	0	0	1	0
Stella	30	1	1	1	1
Rob	25	1	0	0	1
Joe	42	1	1	0	1
Jim	38	1	1	0	1
Lisa	36	1	1	0	0
Sarah	29	1	0	1	0
David	35	1	0	0	1
Eric	28	1	1	1	0
Mike	20	0	1	0	1
Karen	38	1	0	0	1
Megan	31	0	0	1	0



$$-(10/15 \cdot \log_2(10/15) + 5/15 \cdot \log_2(5/15))$$
$$-(-.389975 + -.528308)$$
$$-(-.918278)$$
$$.918278$$

For the left split(split on 1 for “potato_salad?”) we get **9/15 * .764204**.
For the right side of the split (split on 0 for “potato_salad?”) **we get 6/15 * 1.**

Decision Trees Become Forests?



Bagging

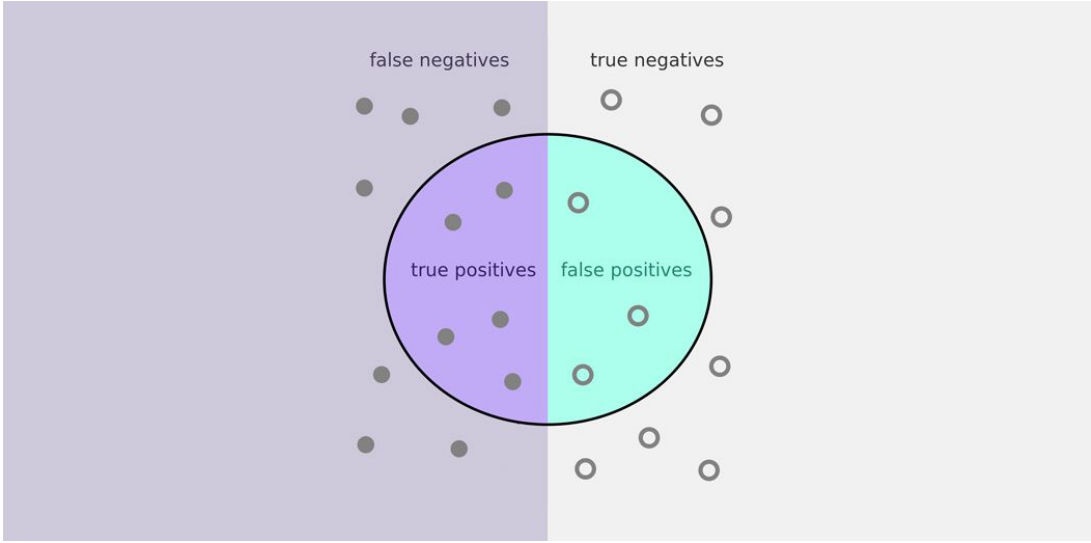
Data: Sample from data for each tree

Features: Sample from feature space for each tree

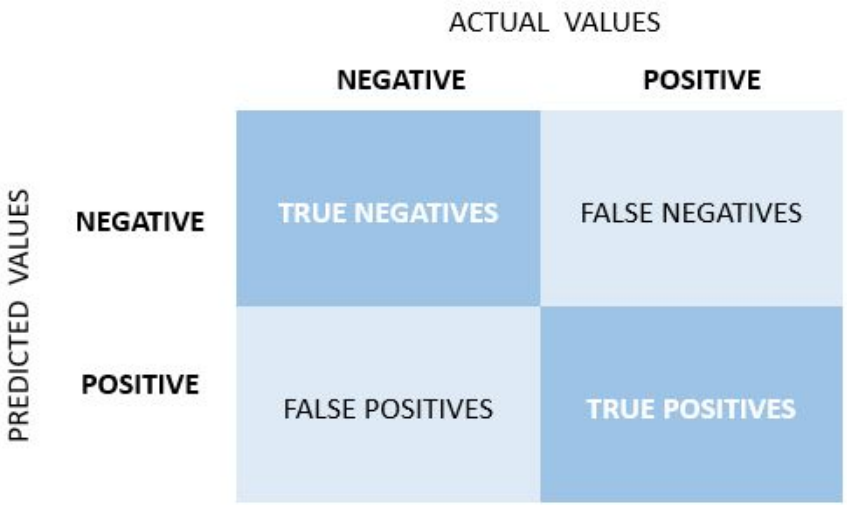
Measure of Goodness - Accuracy, Precision, Recall

Create a model entirely in their head to identify terrorists trying to board flights with greater than 99% accuracy?

Simply label every single person flying as not a terrorist. Given the 800 million average passengers on US flights per year and the 19 (confirmed) terrorists who boarded US flights from 2000–2017, this model achieves accuracy of 99.9999999%!



How Good Is A Model



Metric Name	Formula from Confusion Matrix
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
Precision	$\frac{TP}{TP+FP}$
Recall, Sensitivity, TPR	$\frac{TP}{TP+FN}$
Specificity, 1-FPR	$\frac{TN}{TN+FP}$
F1	$\frac{2 * precision * recall}{precision + recall}$