Which Classifier to Pick?

Based on Sci-Kit guide, use Naïve Bayes

A diagram of a flowchart

Description automatically generated

**NAÏVE BAYES:**

Based on Bayes’ Theorem with assumption that features are independent of each other. The reality is that features sometimes determine each other, but this is ignored by “Naïve” Bayes. This makes it work well (since we’re able to ignore it).

Works well for spam and other NLP applications:

How it works:

* Bayes’ Theorem: computer posterior probability based on likelihood and prior probabilities.
* Naïve Bayes assumes features are independent, so when given a (thing, message, paragraph, other), it breaks it down into its constituent parts and calculates probability this way.
  + Ex. “Hello Friend” becomes “Hello” “Friend”
* For textual applications, some words may have a frequency (or some other thing) of 0 (Zero-Frequency Problem). This would greatly skew results. As such, we use Laplace Smoothing to regularize Naïve Bayes.

Pros:

* Requires small amount of training data (less time)
* Handles continuous and discrete data, and not sensitive to irrelevant features (collinearity?)
* Both binary and multi-class classification problems (synthetic/authentic or male/female, but also twain/Austen/bronte/…)
* Scalable
* When features are not independent, Naïve Bayes can learn decision boundary faster than discriminative models like log regression because it treats them as independent. (log regression would maximize a likelihood function, this doesn’t).

Cons:

* In reality, features are not usually independent.
* The zero frequency problem, solved with smoothing techniques.
* Pure Probability, not learning anything.

Applications:

* Real-time Prediction
* Multi-class Prediction
* Text Classification/Sentiment Analysis
* Recommendation Systems

First Mention (I think): Comparing Decision Trees and Naïve Bayes Classifiers in four medical diagnostic problems.

<https://scholar.google.com/citations?view_op=view_citation&hl=en&user=b2CpVeoAAAAJ&cstart=200&pagesize=100&sortby=pubdate&citation_for_view=b2CpVeoAAAAJ:SP6oXDckpogC>

Most used for: NLP tasks because of independence assumptions.

Gaussian Naïve Bayes requires normal distribution, but normal Naïve Bayes does not.

Normal Naïve Bayes allows for multi-nomial distributions

**DECISION TREE:**

What is it:

* Flowchart where a node represents a feature (attribute) and an edge represents a decision. The outcome of the decision tree is held in the lead nodes.
* It shares internal decision-making process (white box), unlike other black box models like neural networks (we know that).
* They can handle high-dimensional data with good accuracy
* Distribution-free and non-parametric: they don’t require any specific distribution of data (normal, skewed, etc..), and they don’t assume homoskedasticity (homogeneity of variances in different groups being compared). This makes them versatile.

How does it work:

1. Select the best attribute using Attribute Selection Measures (ASM) to split the records (what are records?)
2. Make that attribute a decision node and break the dataset into smaller subset
3. Build the tree by doing the above recursive until there are either no more instances (?), there are no more remaining attributes, or that all of them belong to the same attribute (what does this even mean).

Attribute Selection Measures: selecting the splitting criterion that splits the data in the best manner possible. ASM will rank each feature by explaining the given dataset. The best score attribute will be selected as a splitting attribute.

Most popular selection measures:

* Information Gain ([Kullback–Leibler divergence](https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence)), good source on this (<https://homes.cs.washington.edu/~shapiro/EE596/notes/InfoGain.pdf>) :
  + Computes the difference between the entropy of the the dataset before the split (by some feature) and the average entropy after the split of the dataset based on some attribute. It decides which training feature vectors is most useful for discriminating between classes
    - Entropy is measure of randomness (impurity or noise) of the dataset
      * Impurity is a measure of the homogeneity of the labels
    - Information gain = a decrease in entropy = a more homogenous split = a better split
  + It can work with both [continuous and discrete](https://en.wikipedia.org/wiki/Continuous_or_discrete_variable) variables.
  + Cons I guess: ignores feature interactions and biased towards features with many categories (more distinct values for an attribute) or multiple attributes with many distinct values
    - Example from Wikipedia: if your feature is a member ID where each person has distinct one
    - This is why gain ratio came about
* Gain Ratio:
  + Handles the issue of bias by normalizing information (a ratio of information gain to the intrinsic information). It takes the number and size of branches into account when choosing an attribute.
    - Information gain/split information
      * where split information is a positive number that describes the worth of splitting a branch from the node
      * split information penalizes attribute that result in a large number of distinct splits (lots of complexity,
    - member id example (based on my understanding): if we split by id, there is high information gain. However, there’s also high split information, since we’re generating branches equal to the number of ids (since each is distinct). Information gain ratio considers both values, and the overall importance of this split would be reduced, guiding the decision tree to some other meaningul feature.
  + Will always favor an attribute that has a lower number of distinct values.

A screenshot of a phone

Description automatically generated

“learn the training set too well” -> overfitting: where is captures both underlying patterns AND noise outliers or irrelevant data. This means that the model performs well on training set but not on unseen data (poor accuracy and validation)

* **Gini Index (default, what we’re using):** 
  + Measures how often a randomly chosen element of set would be incorecctly labeled and if it were labelled randomly and independently according to the distribution of labels in a set. Range 0-1 [0 pure, 1 max impurity]
    - An attribute with a lower gini index should be preferred, since they would create a more pure split.
  + Can be sensitive to noise, not as effective when classes are imbalanced (one class has more instances that the other

Pros:

**RANDOM FORESTS:**

What is it:

* Ensemble algorithm that builds multiple decision trees and merges them together to get a more accurate and stable prediction.
  + Differs from decision trees in that they add randomness to models while growing trees by picking for the best feature among a random subset of features (random subspace selection, which usually reduces correlation between estimators and ends up using a subset of features rather than entire set ). Wide diversity that results in a generally better model.
    - Excerpt from Wiki: “The random subspace method is similar to bagging except that the [features](https://en.wikipedia.org/wiki/Feature_(machine_learning)) ("attributes", "predictors", "independent variables") are randomly sampled, with replacement, for each learner. Informally, this causes individual learners to not over-focus on features that appear highly predictive/descriptive in the training set, but fail to be as predictive for points outside that set. For this reason, random subspaces are an attractive choice for high-dimensional problems where the number of features is much larger than the number of training points, such as learning from fMRI data[[3]](https://en.wikipedia.org/wiki/Random_subspace_method#cite_note-4) or gene expression data.[[4]](https://en.wikipedia.org/wiki/Random_subspace_method#cite_note-5)”
  + Deep Decision trees suffer from overfitting (they constrain themselves to the “rules” they’ve created). The random subsets of features in random forest classification and smaller trees (bootstrap sample -> bootstrap aggregation?) prevent this from happening as often (slower computation, doesn’t work every time but is a good strategy against overfitting).
  + Feature importance can drop ones that aren’t indicative of prediction process.

Pros:

* Doesn’t overfit with enough tress
* Liked by data scientists because its good. Also is flexible and handles regression and classification tasks well.
  + Less influenced by outliers according to this [ds stack overflow](https://datascience.stackexchange.com/questions/54751/when-to-use-random-forest)
  + Don’t make assumptions about distribution of data and can implicitly handle collinearity
  + Good for feature selection, although keep an eye out for highly correlated features.
* Handle high dimensionality well

Cons:

* Not good for finding degree of influence, in which case use log regression (not necessary for us I don’t think)
* Performs poorly for extrapolation

[This paper](https://dl.acm.org/doi/10.1145/3357384.3357891#:~:text=The%20Random%20Forest%20(RF)%20classifiers,using%20random%20subsets%20of%20features.) says that its suitable for dealing with high dimensional noisy data in text classification, which would theoretically make it suitable for our needs. There’s a lot of other papers that propose an improve text classification specific RF that has been shown to perform better than RF.

**LOG REGRESSION:**

What is it:

* Handles binary classification tasks, hinging on the sigmoid (log) function that takes input and outputs out 0-1 probability.
* Sigmoid-S shaped function, allows for 0 and 1 max values
  + Takes the best of log odds (probability of success over the probability of failure. The classifier uses different coeffecients, calculates log odd, and chooses the best one.
  + Trying to interpret it within our context: odds that this paragraph belongs in synthetic (1) or authentic (0). Maybe.
* Types:
  + Binomial: 2 class, multinomial: 3+ unordered types, Ordinal: 3+ ordered
* Assumptions:
  + Independence (like Naïve Bayes)
  + Binary Dependent variables.
  + Linear relationship between independent variables and log odds (ratio of occurring to not occurring) of the dependent variable.
    - No multicollinearity:
  + Large sample
  + No outliers
* Setting the threshold value is important and is dependent on the classification problem itself.
  + Low Precision/High Recall: Reducing false negative without necessarily reducing false positive, decision value with low Prevision or high recalls (Ex. Cancer—want don’t want people to be classified as non-affected, opposite we don’t really care)
  + High Precision/Low Recall: reducing false positives without necessarily reducing false negatives.
* For multi-class classification, SoftMax function used.

OCR, Fraud, banking and others

**SUPPORT VECTOR MACHINE (SVM):**

What is it:

* Can do binary and multi-class classification
* Linear and nonlinear classification
* Effective because they focus on finding max separating hyperplane between different classes in target feature.
  + Dimension of hyperplane depends on number of features.
  + Maximizes separation margin between two classes (between hyperplane and closest points from both classes
  + If I understand it correctly, during optimization, an SVM plays (not plays, calculates) around with the weight (coefficient) of each decision boundary until the highest margin is achieved. The highest margin is taken as the best separator, and its weight correlates to its feature importance. Higher weight = more important feature.
    - Only the support vectors (the closest data points to the decision boundary) influence the weights. These data points determine the orientation and position of the hyperplane, so their features are most influential in setting the feature weights

Pros:

* Effective in high dimension spaces
* Robust to outliers
* Effective in cases where number of dimensions is greater than the number of samples
* Uses subset of training points in decision function (support vectors)

Cons:

* If features great than # of samples, overfitting. Choose kernel functions and regularization terms.

Used for text classification, image classification, spam detection, handwriting identification, gene expression analysis, face detection, and anomaly detection.

**NEAREST SHRUNKEN CENTROID:**

First used for gene expression

What is it:

* Calculates a centroid for each class, then test samples are classified to the class with the nearest centroid (obviously).
  + Shrunken centroid has a shrink-threshold where value for each feature of each centroid is divided by within class variance (standardized).
    - Removes noisy features.
  + With our data, this means that there are 2 cetroids (synthetic/authentic), and we’ve calculated the average value for each feature. Then we take the overall mean (over all sample regardless of class) and we subtract the class-specific mean from that. Whatever results from that shows us how much a feature differs from class to overall. A higher difference means more weight. When shrunken (mean multiplied by shrink threshold), features with significant deviation are more weighted. Less important features shrink closer to 0.

Cons:

* Suffers when classes have a lot of variance (do ours?), since equal variance in all dimensions is assumed.
  + LDA and Quadratic Discriminant Analysis are the same but they don’t use this assumption.