Reading Assignment 4

**1. Naive Baye’s algorithm:**

It is a type of classification algorithm is based on baye’s theorem and assumes to have particular feature present in a class is not related to any other feature. It is easy to build for large datasets.  Baye’s theorem calculates posterior probability as P(c|x) = (P(x|c) P(c))/ P(x) where P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes). P(c) is the prior probability of class. P(x|c) is the likelihood which is the probability of predictor given class. P(x) is the prior probability of predictor.

Steps followed to perform it. Firstly, convert the data set into a frequency table. Then, create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64. Then use baye’s formula to find posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Advantages:

* It is easy and fast to predict class of test data set. It can also perform well in multiple class prediction
* When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
* It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Disadvantages:

* If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
* On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict\_proba are not to be taken too seriously.
* Another limitation is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Applications of Naïve baye’s include Real time prediction, Multi class prediction, Text classification and Recommendation system.

**2. An Introduction to Bayes’ Rule**

Bayes’ rule transforms probabilities that look useful (but are often not) into probabilities that are useful. In the poxy diseases example, in patient’s perspective , the doctor used Bayes’ rule to transform the uninformative probability of your symptoms given that you have smallpox into the informative probability that you have smallpox given your symptoms.

The ‘weighted likelihood’ p(smallpox|spots) is also a conditional probability, but it is the probability of the disease smallpox given the symptoms observed, as shown in Figure 1.4. So, by making use of prior experience, we have transformed the conditional probability of the observed symptoms given a specific disease (the likelihood, which is based only on the available evidence) into a more useful conditional probability: the probability that the patient has a particular disease (smallpox) given that he has particular symptoms (spots). In fact, we have just made use of Bayes’ rule to convert one conditional probability, the likelihood p(spots|smallpox) into a more useful conditional probability, which we have been calling a ‘weighted likelihood’, but is formally known as the posterior probability p(smallpox|spots). As noted above, both p(smallpox|spots) and p(spots|smallpox) are conditional probabilities, which have the same status from a mathematical viewpoint. However, for Bayes’ rule, we treat them very differently.

**3. Bayes’ Rule**: Hypothesis and Data If we consider a putative disease to represent a specific hypothesis, and the symptoms to be some observed data then Bayes’ rule becomes p(hypothesis|data) = p(data|hypothesis) ⇥ p(hypothesis) p(data) , where the word “hypothesis” should be interpreted as, “hypothesis is true”.

Specifically, the probability that the proposed hypothesis is true given some data that were actually observed is the posterior probability p(hypothesis|data), whereas the probability of observing the data given that the hypothesis is true is the likelihood p(data|hypothesis).

**4. Model selection** is the most probable model which involves a comparison using a ratio of posterior probabilities.

The posterior ratio, which is also known as the posterior odds between the hypotheses θc and θs, is Rpost = p(θc|x) p(θs|x) .

Baye’s factor B = p(x|θc) p(x|θs) ,

posterior odds = Bayes factor \* prior odds

A forward probability, which involves calculating the probability of each of a number of different consequences (eg obtaining two heads) given some known cause or fact,

The converse of reasoning forwards from a given physical parameter or scenario involves a harder problem, also illustrated in Figure 1.12. Reasoning backwards from measurements (eg coin flips or images) amounts to finding the posterior or inverse probability of the value of an unobserved variable (eg coin bias, 3D shape), which is usually the cause of the observed measurement.

Thus this rule says that not only that prior experience is crucial for interpreting evidence, but also that Bayes’ rule provides a rigorous method for doing so.

**5. Splitting a dataframe into train and test in Python:**

1. Load the data by t=giving the path of the file where the data is stored.
2. Selecting the data for splitting by selecting the required columns.
3. We can use train\_test\_split function of sklearn library to split data that can handle pandas data frames and arrays.

test\_size: This parameter represents the proportion of dataset that is to be considered in the test split portion. The default value for the test split size is set to 0.25.i.e. 25% of data for testing and 75% of data for training.

Random\_state: The shuffling applied to the data before the split is controlled by this parameter. By using random\_state, we can split the data across multiple function calls.

Shuffle: indicates whether the data needs to be shuffled before splitting.

|  |
| --- |
| train, test = train\_test\_split(df\_text\_genre, test\_size=0.2, random\_state =42, shuffle=True) |
|  |
|  |

After splitting the data, we use the directory path variable to define a file path for saving the train and the test data.

**6. Logistic regression**

Logistic regression is used when the dependent variable is categorical. Ex prediction of a mail whether it is spam or not.

**Types of Logistic Regression**

1. Binary Logistic Regression: The categorical response has only two 2 possible outcomes. Example: Spam or Not

2. Multinomial Logistic Regression: Three or more categories without ordering. Example: Predicting which food is preferred more (Veg, Non-Veg, Vegan)

3. Ordinal Logistic Regression: Three or more categories with ordering. Example: Movie rating from 1 to 5

**Decision Boundary**

To predict which class a data belongs, a threshold can be set. Based upon this threshold, the obtained estimated probability is classified into classes.Say, if predicted\_value ≥ 0.5, then classify email as spam else as not spam.Decision boundary can be linear or non-linear. Polynomial order can be increased to get complex decision boundary.

**Cost Function**

For linear regression, we use mean square error as cost function. For logistic regression, we have hΘ(x) = sigmoid (Z)

Cost(hΘ (x), Y actual) = -log(hΘ (x)) if Y = 1

log(hΘ (x)) if Y = 1

**7. Decision Tree:**

A decision tree is an algorithm which employs a top-down, greedy search through the space of possible branches with no backtracking.A decision tree can be used to visually and explicitly represent decisions and decision making. Decision trees are building blocks of random forest model.

the bold text that represents a condition is **internal node**, based on which the tree splits into branches or **edges**. The end of the branch that doesn’t split anymore is the decision or **leaf**. Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

Recursive binary splitting: In this procedure all the features are considered and different split points are tried and tested using a cost function. The split with the best cost (or lowest cost) is selected. We have the features and splits and then calculate how much accuracy each split will cost us, using a function. The accuracy of the model can be improved by making a more accurate estimate of f(X) and therefore reducing the reducible error.

Cost of split

**The cost functions used for classification and regression**. In both cases the cost functions try to **find most homogeneous branches, or branches having groups with similar responses**. This makes sense we can be more sure that a test data input will follow a certain path.

Regression : sum(y-prediction)2

A Gini score gives an idea of how good a split is by how mixed the response classes are in the groups created by the split.

Classification: G = sum(pk \*(1-pk))

We should stop splitting after a certain stage to avoid trees are complexity and can lead to overfitting.

One way of doing this is to **set a minimum number of training inputs to use on each leaf.**  Another way is to set **maximum depth** of your model. **Maximum depth refers to the the length of the longest path from a root to a leaf.**

Pruning : Pruning involves removing the branches that make use of features having low importance. This way, we reduce the complexity of tree, and thus increasing its predictive power by reducing overfitting. The performance can be increased using this method.

Advantages of CART

* Simple to understand, interpret, visualize.
* Decision trees implicitly perform variable screening or feature selection.
* Can handle both numerical and categorical data. Can also handle multi-output problems.
* Decision trees require relatively little effort from users for data preparation.
* Nonlinear relationships between parameters do not affect tree performance.

**Disadvantages of CART**

* Decision trees can be unstable because small variations in the data might result in a completely different tree being generated.
* Decision-tree learners can create over-complex trees that do not generalize the data well. This is called overfitting.
* Greedy algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees, where the features and samples are randomly sampled with replacement.

**8. Random Forests :**

As its name implies, it consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction.

**A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.** the prerequisites for random forest to perform well are:

1. There needs to be some actual signal in our features so that models built using those features do better than random guessing.
2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.
3. **Bagging (Bootstrap Aggregation) — Decisions trees are very sensitive to the data they are trained on — small changes to the training set can result in significantly different tree structures.**
4. Random forest takes advantage of this by allowing each individual tree to randomly sample from the dataset with replacement, resulting in different trees. This process is known as bagging.

**Feature Randomness - T**his forces even more variation amongst the trees in the model and ultimately results in lower correlation across trees and more diversification.

The random forest is a classification algorithm consisting of many decisions trees. **It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees** whose prediction by committee is more accurate than that of any individual tree.

**9. Principal component analysis(PCA):**

PCA is used for dimensionality reduction for large dataset. Reducing the number of components or features costs some accuracy and  on the other hand, it makes the large data set simpler, easy to explore and visualize. It also reduces the computational complexity of the model which makes machine learning algorithms run faster.

1. Standardization

Standardization is the process of bringing all the features spaces that are present with different scale of measurements to mean =0 and variance =1. It is recommended before performing PCA as PCA is very sensitive to variances.

2. Eigen vectors and eigen values: The Eigenvectors (principal components) determine the directions of the new feature space, and the eigenvalues determine their magnitude.  They explain the variance of the data along the new feature axes. It means the corresponding eigenvalue tells us that how much variance is included in that new transformed feature. To get eigenvalues and Eigenvectors we need to compute the covariance matrix.

**Covariance Matrix** : This classic approach of PCA is to perform the Eigen decomposition on the covariance matrix Σ, which is a d×d matrix where each element represents the covariance between two features, where d is the number of original dimensions of the data set

Eigen vectors can be calculated as cov\_mat = np.cov(X\_std.T)   
eig\_vals, eig\_vecs = np.linalg.eig(cov\_mat)

Selecting Principal components:The typical goal of a PCA is to reduce the dimensionality of the original feature space by projecting it onto a smaller subspace, where the eigenvectors will form the axes. the eigenvectors only define the directions of the new axis, since they have all the same unit length 1. The rule behind selecting a new set of Principal components is that

we sort the Eigenvalues in descending order and then choose the top k features concerning top k Eigenvalues.

**Sorting eigenvalues**

Based on the some conditions we can decide which eigen values can be dropped. They are, The Eigenvectors with the lowest eigenvalues bear the least information about the distribution of the data; those are the ones can be dropped. To do so, the common approach is to rank the eigenvalues from highest to lowest to choose the top k Eigenvectors.

**Explained Variance**: This can be calculated from eigen values. The explained variance tells us how much information (variance) can be attributed to each of the principal components.

**Projection matrix**: Projection matrix will be used to transform the Iris data onto the new feature subspace or we say newly transformed data set with reduced dimensions. It is a matrix of our concatenated top k Eigenvectors.

**Predictive analysis** is a part of data mining that helps in finding predictions and trend. Prediction analysis for dummies book helps to explore different ways of valuable predictions in business, or in advertising, politics etc.

We can load data using the below code. This loads the iris data

from sklearn.datasets import load\_iris

iris = load\_iris()

We can create an instance in the classifier using the below lines of code. First line imports logistic regression library. Second line creates instance of logistic regression. The parameter C in the constructor is regularization parameter, it is default set to 1.

sklearn import linear\_model

logClassifier = linear\_model.LogisticRegression(C=1, random\_state=111)

To run training data, we need to split the data into train and test sets and then create a logistic regression classifier.

from sklearn import cross\_validation

X\_train, X\_test, y\_train, y\_test = cross\_validation.train\_test\_split(iris.data, iris.target, test\_size=0.10, random\_state=111)

logClassifier.fit(X\_train, y\_train)

To run the test data, we use the following lines of code

predicted = logClassifier.predict(X\_test)

predictedarray([0, 0, 2, 2, 1, 0, 0, 2, 2, 1, 2, 0, 2, 2, 2])

To evaluate a model, You can cross-reference the output from the prediction against the y\_testarray. As a result, you can see that it predicted all the test data points correctly. Here’s the code:

from sklearn import metrics

predictedarray([0, 0, 2, 2, 1, 0, 0, 2, 2, 1, 2, 0, 2, 2, 2])

y\_testarray([0, 0, 2, 2, 1, 0, 0, 2, 2, 1, 2, 0, 2, 2, 2])

metrics.accuracy\_score(y\_test, predicted)1.0 # 1.0 is 100 percent accuracy

predicted == y\_testarray([ True, True, True, True, True, True, True, True, True, True, True, True, True, True, True], dtype=bool)

the below code to create and evaluate the logistic classifier with C=150:

logClassifier\_2 = linear\_model.LogisticRegression( C=150, random\_state=111)

logClassifier\_2.fit(X\_train, y\_train)

predicted = logClassifier\_2.predict(X\_test)

metrics.accuracy\_score(y\_test, predicted)0.93333333333333335

metrics.confusion\_matrix(y\_test, predicted)array([[5, 0, 0], [0, 2, 0], [0, 1, 7]])

We expected better, but it was actually worse. There was one error in the predictions. The result is the same as that of the Support Vector Machine (SVM) model.

**AUC-ROC curve**

It is a performance measurement for classification problem at various threshold settings

AUC describes degree or measure of separability while ROC is probability.

**Train/Test splitting and cross validation in python**

In machine learning and statistics, we usually split our data into two subsets: training and testing data (or sometimes as train, validate and test) to fit our model on train data, in order to make predictions on the test data.

**Overfitting:** This model will be very accurate on the training data but will probably be very not accurate on untrained or new data. Overfitting means that model we trained has trained “too well” and is now, well, fit too closely to the training dataset. This usually happens when the model is too complex (i.e. too many features/variables compared to the number of observations).

**Underfitting:** It also means the model cannot be generalized to new data. In contrast to overfitting, when a model is underfitted, it means that the model does not fit the training data and therefore misses the trends in the data.