UNIT 2

Supervised Learning: Regression: Introduction to Linear Regression and Multiple Linear Regression, KNN. Measuring regression model performance - R Square, Mean Square Error(MSE),Root Mean Square Error(RMSE), Mean Absolute Error(MAE)
Classification: Support vector machine- Characteristics of SVM, Linear SVM, Naive Bayes Classifier, KNN classifier, Logistic Regression. [TB-2]
Measuring Classifier Performance: Precision, Recall, Confusion Matrix. [TB1]

Supervised Learning: -

Supervised Machine Learning: It is an ML technique where models are trained on labeled data i.e output variable is provided in these types of problems. Here, the models find the mapping function to map input variables with the output variable or the labels. **Regression and Classification** problems are a part of Supervised Machine Learning.

Regression: -

Regression Analysis in Machine learning

Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables. More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed. It predicts continuous/real values such as **temperature, age, salary, price,** etc.

We can understand the concept of regression analysis using the below example:
Example: Suppose there is a marketing company A, who does various advertisement every year and get sales on that. The below list shows the advertisement made by the company in the last 5 years and the corresponding sales:

Advertisement	Sales
$90	$1000
$120	$1300
$150	$1800
$100	$1200
$130	$1380
$200	??

Now, the company wants to do the advertisement of $200 in the year 2023 **and wants to know the prediction about the sales for this year**. So to solve such type of prediction problems in machine learning, we need regression analysis.

Regression is a supervised learning technique which helps in finding the correlation between variables and enables us to predict the continuous output variable based on the one or more predictor variables. It is mainly used for **prediction, forecasting, time series modeling, and determining the causal-effect relationship between variables**.

In Regression, we plot a graph between the variables which best fits the given datapoints, using this plot, the machine learning model can make predictions about the data.

In simple words, ***"Regression shows a line or curve that passes through all the datapoints on target-predictor graph in such a way that the vertical distance between the datapoints and the regression line is minimum."*** The distance between datapoints and line tells whether a model has captured a strong relationship or not.

Some examples of regression can be as:

- Prediction of rain using temperature and other factors
- Determining Market trends
- Prediction of road accidents due to rash driving.

Terminologies Related to the Regression Analysis:

- **Dependent Variable:** The main factor in Regression analysis which we want to predict or understand is called the dependent variable. It is also called **target variable**.
- **Independent Variable:** The factors which affect the dependent variables or which are used to predict the values of the dependent variables are called independent variable, also called as a **predictor**.
- **Outliers:** Outlier is an observation which contains either very low value or very high value in comparison to other observed values. An outlier may hamper the result, so it should be avoided.

- **Multicollinearity:** If the independent variables are highly correlated with each other than other variables, then such condition is called Multicollinearity. It should not be present in the dataset, because it creates problem while ranking the most affecting variable.
- **Underfitting and Overfitting:** If our algorithm works well with the training dataset but not well with test dataset, then such problem is called **Overfitting**. And if our algorithm does not perform well even with training dataset, then such problem is called **underfitting**.

Why do we use Regression Analysis?

As mentioned above, Regression analysis helps in the prediction of a continuous variable. There are various scenarios in the real world where we need some future predictions such as weather condition, sales prediction, marketing trends, etc., for such case we need some technology which can make predictions more accurately. So for such case we need Regression analysis which is a statistical method and used in machine learning and data science. Below are some other reasons for using Regression analysis:

- Regression estimates the relationship between the target and the independent variable.
- It is used to find the trends in data.
- It helps to predict real/continuous values.
- By performing the regression, we can confidently determine the **most important factor, the least important factor, and how each factor is affecting the other factors**.

Types of Regression

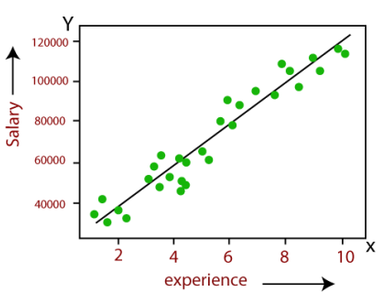
There are various types of regressions which are used in data science and machine learning. Each type has its own importance on different scenarios, but at the core, all the regression methods analyze the effect of the independent variable on dependent variables. Here we are discussing some important types of regression which are given below:

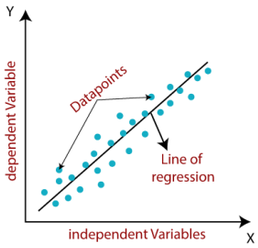
- **Linear Regression**
- **Logistic Regression**
- **Polynomial Regression**
- **Support Vector Regression**
- **Decision Tree Regression**
- **Random Forest Regression**
- **Ridge Regression**
- **Lasso Regression**

Linear Regression:

Linear regression is a statistical regression method which is used for predictive analysis. It is one of the very simple and easy algorithms which works on regression and shows the relationship between the continuous variables. It is used for solving the regression problem in machine learning. In the simplest words, **Linear Regression** is the supervised Machine Learning model in which the **model finds the best fit linear line between the independent and dependent variable** i.e it finds the linear relationship between the dependent and independent variable.

If there is only one input variable (x), then such linear regression is called **simple linear regression**. And if there is more than one input variable, then such linear regression is called **multiple linear regression**. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price,** etc. The relationship between variables in the linear regression model can be explained using the below image. Here we are predicting the salary of an employee on the basis of **the year of experience**.



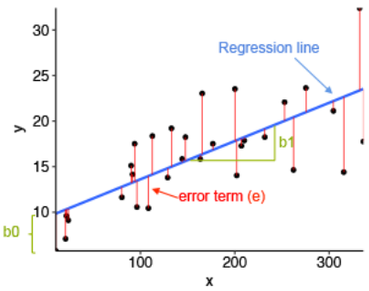


Below is the mathematical equation for Linear regression:

$$Y = aX + b$$

Here, Y = dependent variables (target variables),
X= Independent variables (predictor variables),
a and b are the linear coefficients

A Linear Regression model's main aim is to find the best fit linear line and the optimal values of intercept and coefficients such that the error is minimized. Error is the difference between the actual value and Predicted value and the goal is to reduce this difference.



Statistical tools for high-throughput data analysis

In the above diagram,

- x is our dependent variable which is plotted on the x-axis and y is the dependent variable which is plotted on the y-axis.
- Black dots are the data points i.e the actual values.
- b_o is the intercept which is 10 and b_1 is the slope of the x variable.
- The blue line is the best fit line predicted by the model i.e the predicted values lie on the blue line.

The vertical distance between the data point and the regression line is known as error or residual. Each data point has one residual and the sum of all the differences is known as **the Sum of Residuals/Errors.**

Mathematical Approach:

Residual/Error = Actual values – Predicted Values

Sum of Residuals/Errors = Sum(Actual- Predicted Values)

Square of Sum of Residuals/Errors = (Sum(Actual- Predicted Values))^2

i.e

$$\sum {e_i}^2 = \sum (Y_i - \hat{Y}_i)^2$$

Some popular applications of linear regression are:

- **Analyzing trends and sales estimates**
- **Salary forecasting**
- **Real estate prediction**
- **Arriving at ETAs in traffic.**

Types of Linear Regression

Linear regression can be further divided into two types of the algorithm:

- **Simple Linear Regression:** If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.
- **Multiple Linear regression:** If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

Equation of Simple Linear Regression, where b_o is the intercept, b_1 is coefficient or slope, x is the independent variable and y is the dependent variable.

$$y = b_o + b_1 x$$

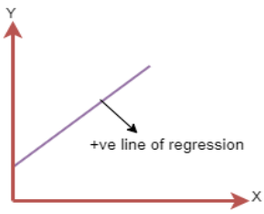
Equation of Multiple Linear Regression, where bo is the intercept, $b_1, b_2, b_3, b_4 \ldots, b_n$ are coefficients or slopes of the independent variables $x_1, x_2, x_3, x_4 \ldots, x_n$ and y is the dependent variable.

$$y = b_o + b_1 x_1 + b_2 x_2 + b_3 x_3 \ldots . + b_n x_n$$

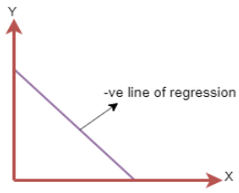
Linear Regression Line

A linear line showing the relationship between the dependent and independent variables is called a **regression line**. A regression line can show two types of relationship:

- **Positive Linear Relationship:**
  If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.
- **Negative Linear Relationship:**
  If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.



The line equation will be: $Y= a_0+a_1x$



The line of equation will be: $Y= -a_0+a_1x$

Finding the best fit line: -

When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.

Simple Linear Regression: -

Simple Linear Regression is a type of Regression algorithms that models the relationship between a dependent variable and a single independent variable. The relationship shown by a Simple Linear Regression model is linear or a sloped straight line, hence it is called Simple Linear Regression.

The key point in Simple Linear Regression is that the ***dependent variable must be a continuous/real value***. However, the independent variable can be measured on continuous or categorical values.

Simple Linear regression algorithm has mainly two objectives:

- **Model the relationship between the two variables.** Such as the relationship between Income and expenditure, experience and Salary, etc.
- **Forecasting new observations.** Such as Weather forecasting according to temperature, Revenue of a company according to the investments in a year, etc.

Simple Linear Regression Model:

The Simple Linear Regression model can be represented using the below equation:

$y= a+bx+ \varepsilon$

Where,

a= It is the intercept of the Regression line (can be obtained putting x=0)
b= It is the slope of the regression line, which tells whether the line is increasing or decreasing.
ε = The error term. (For a good model it will be negligible)

We will find the value of a and b by using the below formula

$$a= \frac{(\sum Y)(\sum X^2)-(\sum X)(\sum XY)}{n(\sum x^2)-(\sum x)^2}$$

$$b= \frac{n(\sum XY)-(\sum X)(\sum Y)}{n(\sum x^2)-(\sum x)^2}$$

2. How do you define Slope?

Slope tells you how much your target variable will change as the independent variable increases or decreases.

The formula of the slope is y=mx+b, where m is the slope.

Multiple Linear Regression: -

In, Simple Linear Regression, where a single Independent/Predictor(X) variable is used to model the response variable (Y). But there may be various cases in which the response variable is affected by more than one predictor variable; for such cases, the Multiple Linear Regression algorithm is used.

Moreover, Multiple Linear Regression is an extension of Simple Linear regression as it takes more than one predictor variable to predict the response variable. We can define it as:

Multiple Linear Regression is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one independent variable.

Example:

Prediction of CO_2 emission based on engine size and number of cylinders in a car.

Assumptions of multiple linear regression

Multiple linear regression makes all of the same assumptions as simple linear regression:

1. **Linearity: -** The relationship between dependent and independent variables should be linear.
2. **Lack of Multicollinearity: -** It is assumed that there is little or no multicollinearity in the data. Multicollinearity occurs when the features (or independent variables) are not independent of each other.

3. **Multivariate Normality: -** Multiple regression assumes that the residuals are normally distributed.

Multiple linear regression formula

The formula for a multiple linear regression is:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon$$

where, for $i = n$ observations:

y_i = dependent variable

x_i = explanatory variables

β_0 = y-intercept (constant term)

β_p = slope coefficients for each explanatory variable

ϵ = the model's error term (also known as the residuals)

Multiple linear regression is used to estimate the relationship between **two or more independent variables** and **one dependent variable**. You can use multiple linear regression when you want to know:

1. How strong the relationship is between two or more independent variables and one dependent variable (e.g. how rainfall, temperature, and amount of fertilizer added affect crop growth).
2. The value of the dependent variable at a certain value of the independent variables (e.g. the expected yield of a crop at certain levels of rainfall, temperature, and fertilizer addition).

Multiple linear regression example

You are a public health researcher interested in social factors that influence heart disease. You survey 500 towns and gather data on the percentage of people in each town who smoke, the percentage of people in each town who bike to work, and the percentage of people in each town who have heart disease.

Because you have two independent variables and one dependent variable, and all your variables are quantitative, you can use multiple linear regression to analyze the relationship between them.

Difference between Simple Linear Regression and Multi Linear Regression

Simple Linear Regression

Simple Linear Regression establishes the relationship between two variables using a straight line. It attempts to draw a line that comes closest to the data by finding the slope and intercept which define the line and minimize regression errors. Simple linear regression has only one x and one y variable.

Multi Linear Regression

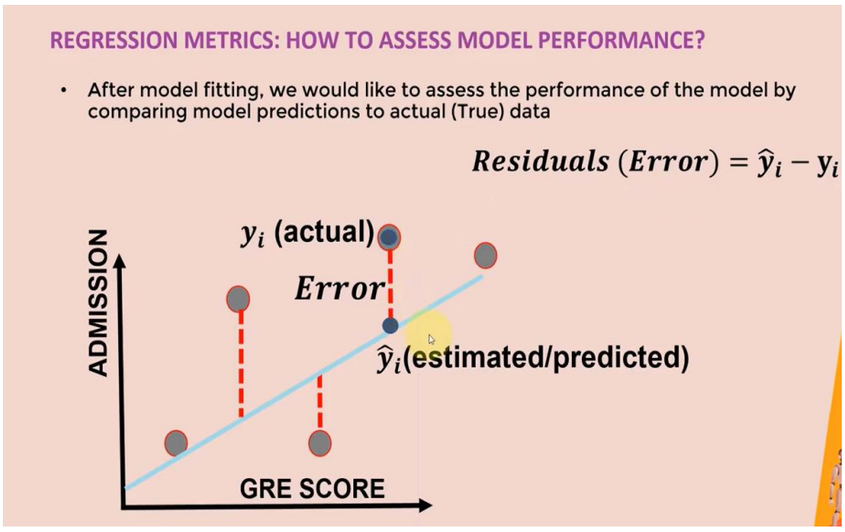
Multiple Linear regressions are based on the assumption that there is a linear relationship between both the dependent and independent variables or Predictor variable and Target variable. It also assumes that there is no major correlation between the independent variables. Multi Linear regressions can be linear and nonlinear. It has one y and two or more x variables or one dependent variable and two or more independent variables.

Measuring regression model performance –

Regression model evaluation metrics: -

The MSE, MAE, RMSE, and R-Squared metrics are mainly used to evaluate the prediction error rates and model performance in regression analysis.

- **MAE** (Mean absolute error) represents the difference between the original and predicted values extracted by averaged the absolute difference over the data set.
- **MSE** (Mean Squared Error) represents the difference between the original and predicted values extracted by squared the average difference over the data set.
- **RMSE** (Root Mean Squared Error) is the error rate by the square root of MSE.
- **R-squared** (Coefficient of determination) represents the coefficient of how well the values fit compared to the original values. The value from 0 to 1 interpreted as percentages. The higher the value is, the better the model is.



R Square: -

R squared or Coefficient of determination, or R^2 is a measure that provides information about the goodness of fit of the regression model. In simple terms, it is a statistical measure that tells how well the plotted regression line fits the actual data. R squared measures how much the variation is there in predicted and actual values in the regression model.

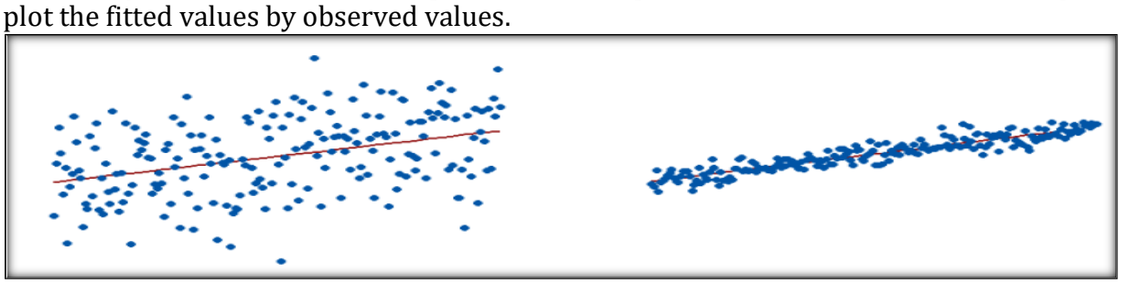
What is the significance of R squared

- R-squared values range from 0 to 1, usually expressed as a percentage from 0% to 100%.
- And this value of R square tells you how well the data fits the line you've drawn.
- The higher the model's R-Squared value, the better the regression line fits the data.
- So if the model value is close to 0, then the model is not a good fit
- A good model should have an R-squared greater than 0.8.

Note- As the R squared value increases, the difference between actual and predicted values decreases.

Visual Representation of R-squared

To visually demonstrate how R-squared values represent the scatter around the regression line, you can plot the fitted values by observed values.



The R-squared for the regression model on the left is 15%, and for the model on the right it is 85%. When a regression model accounts for more of the variance, the data points are closer to the regression line. In practice, you'll never see a regression model with an R^2 of 100%. In that case, the fitted values equal the data values and, consequently, all the observations fall exactly on the regression line.

How to calculate R squared in linear regression?

$$R^2 = 1 - \frac{\text{sum squared regression (SSR)}}{\text{total sum of squares (SST)}},$$
$$= 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}.$$

Mean Square Error(MSE)

While R Square is a relative measure of how well the model fits dependent variables, Mean Square Error is an absolute measure of the goodness for the fit.

Mean Squared Error represents the average of the squared difference between the original and predicted values in the data set. It measures the variance of the residuals.

A larger MSE indicates that the data points are dispersed widely around its central moment (mean), whereas a smaller MSE suggests the opposite. A smaller MSE is preferred because it indicates that your data points are dispersed closely around its central moment (mean). It reflects the centralized distribution of your data values, the fact that it is not skewed, and, most importantly, it has fewer errors (errors measured by the dispersion of the data points from its mean).

Lesser the MSE => Smaller is the error => Better the estimator.

The Mean Squared Error is calculated as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where:
Σ – a symbol that means "sum"
n – sample size
y_i – the actual data value
$\hat{}y_i$ – the predicted data value

The calculations for the mean squared error are similar to the variance. To find the MSE, take the observed value, subtract the predicted value, and square that difference. Repeat that for all observations. Then, sum all of those squared values and divide by the number of observations.

Root Mean Square Error(RMSE)

In statistics, regression analysis is a technique we use to understand the relationship between a predictor variable, x, and a response variable, y.

When we conduct regression analysis, we end up with a model that tells us the predicted value for the response variable based on the value of the predictor variable.

One way to assess how "good" our model fits a given dataset is to calculate the **root mean square error**, which is a metric that tells us how far apart our predicted values are from our observed values, on average.

The formula to find the root mean square error, more commonly referred to as **RMSE**, is as follows:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

where:

- Σ is a fancy symbol that means "sum"
- P_i is the predicted value for the i^{th} observation in the dataset
- O_i is the observed value for the i^{th} observation in the dataset
- n is the sample size

The root mean square error can be calculated for any type of model that produces predicted values, which can then be compared to the observed values of a dataset.

The root mean square error is also sometimes called the root mean square deviation, which is often abbreviated as RMSD.

How to Interpret RMSE

The larger the RMSE, the larger the difference between the predicted and observed values, which means the worse the regression model fits the data. Conversely, the smaller the RMSE, the better a model is able to fit the data.

It can be particularly useful to compare the RMSE of two different models with each other to see which model fits the data better.

Mean Absolute Error

Mean Absolute Error calculates the average difference between the calculated values and actual values. It is also known as scale-dependent accuracy as it calculates error in observations taken on the same scale. It is used as evaluation metrics for regression models in machine learning. It calculates errors between actual values and values predicted by the model. It is used to predict the accuracy of the machine learning model.

The **Mean Absolute Error**(MAE) is the average of all absolute errors. The formula is:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$$

Where:
n = the number of errors,
Σ = summation symbol (which means "add them all up"),
$|y_i - \hat{y}|$ = the absolute errors.
$\hat{y}$ – *predicted value of y*

Regression Model Accuracy (MAE, MSE, RMSE, R-squared)

Evaluating the model accuracy is an essential part of the process in creating machine learning models to describe how well the model is performing in its predictions. Evaluation metrics change according to the problem type.

The linear model (regression) can be an example of this type of problem, and the main characteristic of the regression problem is that the targets of a dataset contain the real numbers only. The errors represent how much the model is making mistakes in its prediction. The basic concept of accuracy evaluation is to compare the original target with the predicted one according to certain metrics.

Differences among these evaluation metrics

- Mean Squared Error (MSE) and Root Mean Square Error penalizes the large prediction errors vi-a-vis Mean Absolute Error (MAE). However, RMSE is widely used than MSE to evaluate the performance of the regression model with other random models as it has the same units as the dependent variable (Y-axis).
- MSE is a differentiable function that makes it easy to perform mathematical operations in comparison to a non-differentiable function like MAE. Therefore, in many models, RMSE is used as a default metric for calculating Loss Function despite being harder to interpret than MAE.
- The lower value of MAE, MSE, and RMSE implies higher accuracy of a regression model. However, a higher value of R square is considered desirable.

Conclusion

Both RMSE and R- Squared quantifies how well a linear regression model fits a dataset. The RMSE tells how well a regression model can predict the value of a response variable in absolute terms while R-Squared tells how well the predictor variables can explain the variation in the response variable.

Classification Algorithm in Machine Learning: -

The Supervised Machine Learning algorithm can be broadly classified into Regression and Classification Algorithms. In Regression algorithms, we have predicted the output for continuous values, but to predict the categorical values, we need Classification algorithms.

What is the Classification Algorithm?

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, **Yes or No, 0 or 1, Spam or Not Spam, cat or dog,** etc. Classes can be called as targets/labels or categories.

Unlike regression, the output variable of Classification is a category, not a value, such as "Green or Blue", "fruit or animal", etc. Since the Classification algorithm is a supervised learning technique, hence it takes labeled input data, which means it contains input with the corresponding output.

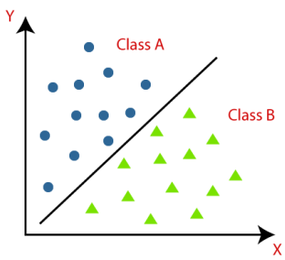
In classification algorithm, a discrete output function(y) is mapped to input variable(x).

y=f(x), where y = categorical output

The best example of an ML classification algorithm is **Email Spam Detector**.

The main goal of the Classification algorithm is to identify the category of a given dataset, and these algorithms are mainly used to predict the output for the categorical data.

Classification algorithms can be better understood using the below diagram. In the below diagram, there are two classes, class A and Class B. These classes have features that are similar to each other and dissimilar to other classes.



The algorithm which implements the classification on a dataset is known as a classifier. There are two types of Classifications:

- **Binary Classifier:** If the classification problem has only two possible outcomes, then it is called as Binary Classifier.
  Examples: YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.
- **Multi-class Classifier:** If a classification problem has more than two outcomes, then it is called as Multi-class Classifier.
  Example: Classifications of types of crops, Classification of types of music.

Learners in Classification Problems:

In the classification problems, there are two types of learners:

1. **Lazy Learners:** Lazy Learner firstly stores the training dataset and wait until it receives the test dataset. In Lazy learner case, classification is done on the basis of the most related data stored in the training dataset. It takes less time in training but more time for predictions.
   Example: K-NN algorithm, Case-based reasoning
2. **Eager Learners:** Eager Learners develop a classification model based on a training dataset before receiving a test dataset. Opposite to Lazy learners, Eager Learner takes more time in learning, and less time in prediction.
   Example: Decision Trees, Naïve Bayes, ANN.

Types of ML Classification Algorithms:

Classification Algorithms can be further divided into the Mainly two category:

- **Linear Models**
  - Logistic Regression
  - Support Vector Machines
- **Non-linear Models**
  - K-Nearest Neighbours
  - Naïve Bayes
  - Decision Tree Classification
  - Random Forest Classification

Support Vector Machine Algorithm

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

To illustrate the basic idea behind SVM, we first introduce the concept of a maximal margin hyperplane and explain the rationale of choosing such a hyperplane.

Maximum Margin Hyperplanes

Figure 5.21 shows a plot of a data set containing examples that belong to two different classes, represented as squares and circles. The data set is also linearly separable; i.e., we can find a hyperplane such that all the squares reside on one side of the hyperplane and all the circles reside on the other side. However, as shown in Figure 5.21, there are infinitely many such hyperplanes possible. Although their training errors are zero, there is no guarantee that the hyperplanes will perform equally well on previously unseen examples. The classifier must choose one of these hyperplanes to represent its decision boundary, based on how well they are expected to perform on test examples.

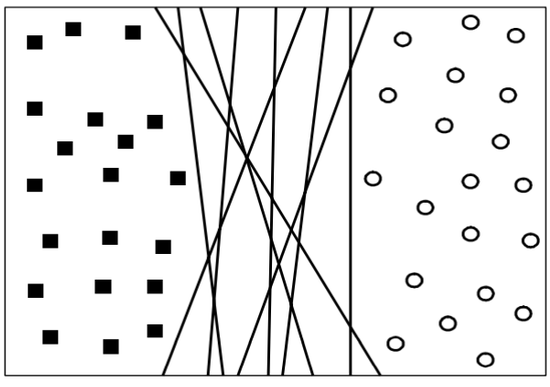


Figure 5.21. Possible decision boundaries for a linearly separable data set.

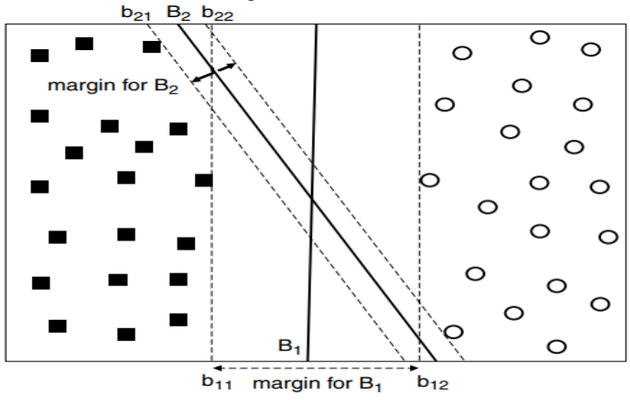


Figure 5.22. Margin of a decision boundary.

To get a clearer picture of how the different choices of hyperplanes affect the generalization errors, consider the two decision boundaries, B1 and B2, shown in Figure 5.22.

Both decision boundaries can separate the training examples into their respective classes without committing any misclassification errors.

Each decision boundary Bi is associated with a pair of hyperplanes, denoted as bi1 and bi2, respectively. bi1 is obtained by moving a parallel hyperplane away from the decision boundary until it touches the closest square(s), whereas bi2 is obtained by moving the hyperplane until it touches the closest circle(s).

The distance between these two hyperplanes is known as the margin of the classifier.

From the diagram shown in Figure 5.22, notice that the margin for B1 is considerably larger than that for B2. In this example, B1 turns out to be the maximum margin hyperplane of the training instances.

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

Rationale for Maximum Margin

Decision boundaries with large margins tend to have better generalization errors than those with small margins. Intuitively, if the margin is small, then any slight perturbations to the decision boundary can have quite a significant impact on its classification. Classifiers that produce decision boundaries with small margins are therefore more susceptible to model overfitting and tend to generalize poorly on previously unseen examples.

A more formal explanation relating the margin of a linear classifier to its generalization error is given by a statistical learning principle known as structural risk minimization (SRM). This principle provides an upper bound to the generalization error of a classifier (R) in terms of its training error (Re), the number of training examples (N), and the model complexity, otherwise known as its capacity (h). More specifically, with a probability of 1 − η, the generalization error of the classifier can be at worst

$$R \leq R_e + \varphi\left(\frac{h}{N}, \frac{\log(\eta)}{N}\right), \qquad (5.27)$$

where φ is a monotone increasing function of the capacity h.

SRM is another way to express generalization error as a tradeoff between training error and model complexity. The capacity of a linear model is inversely related to its margin. Models with small margins have higher capacities because they are more flexible and can fit many training sets, unlike models with large margins. However, according to the SRM principle, as the capacity increases, the generalization error bound will also increase. Therefore, it is desirable to design linear classifiers that maximize the margins of their decision boundaries in order to ensure that their worst-case generalization errors are minimized. One such classifier is the **linear SVM**.

Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Linear SVM: Separable Case

A linear SVM is a classifier that searches for a hyperplane with the largest margin, which is why it is often known as a maximal margin classifier.

Linear Decision Boundary: -

Consider a binary classification problem consisting of N training examples. Each example is denoted by a tuple (x_i, y_i) (i = 1, 2,...,N), where $x_i = (x_{i1}, x_{i2},...,x_{id})^T$ corresponds to the attribute set for the i^{th} example. By convention, let $y_i \in \{-1, 1\}$ denote its class label. The decision boundary of a linear classifier can be written in the following form:

$$\mathbf{w} \cdot \mathbf{x} + b = 0, \qquad (5.28)$$

where w and b are parameters of the model.

Figure 5.23 shows a two-dimensional training set consisting of squares and circles. A decision boundary that bisects the training examples into their respective classes is illustrated with a solid line. Any example located along the decision boundary must satisfy Equation 5.28. For example, if x_a and x_b are two points located on the decision boundary, then

$$\mathbf{w} \cdot \mathbf{x}_a + b = 0,$$

$$\mathbf{w} \cdot \mathbf{x}_b + b = 0.$$

Subtracting the two equations will yield the following:

$$\mathbf{w} \cdot (\mathbf{x}_b - \mathbf{x}_a) = 0,$$

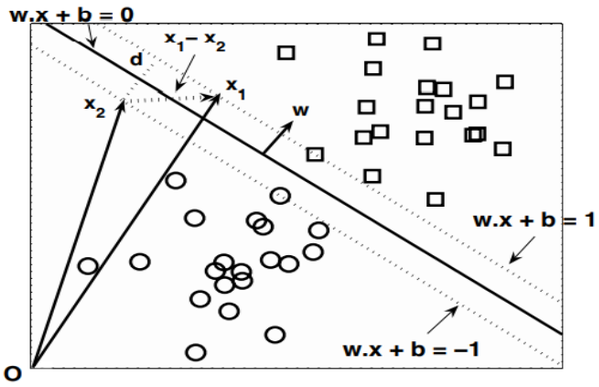


Figure 5.23. Decision boundary and margin of SVM.

where $x_b - x_a$ is a vector parallel to the decision boundary and is directed from x_a to x_b. Since the dot product is zero, the direction for w must be perpendicular to the decision boundary, as shown in Figure 5.23.

For any square $\mathbf{x}_s$ located above the decision boundary, we can show that

$$\mathbf{w} \cdot \mathbf{x}_s + b = k, \tag{5.29}$$

where $k > 0$. Similarly, for any circle $\mathbf{x}_c$ located below the decision boundary, we can show that

$$\mathbf{w} \cdot \mathbf{x}_c + b = k', \tag{5.30}$$

where $k' < 0$. If we label all the squares as class $+1$ and all the circles as class -1, then we can predict the class label y for any test example $\mathbf{z}$ in the following way:

$$y = \begin{cases} 1, & \text{if } \mathbf{w} \cdot \mathbf{z} + b > 0; \\ -1, & \text{if } \mathbf{w} \cdot \mathbf{z} + b < 0. \end{cases} \tag{5.31}$$

Margin of a Linear Classifier: -

Consider the square and the circle that are closest to the decision boundary. Since the square is located above the decision boundary, it must satisfy Equation 5.29 for some positive value k, whereas the circle must satisfy Equation 5.30 for some negative value k[1]. We can rescale the parameters w and b of the decision boundary so that the two parallel hyperplanes b_{i1} and b_{i2} can be expressed as follows:

$$b_{i1} : \ \mathbf{w} \cdot \mathbf{x} + b = 1, \tag{5.32}$$

$$b_{i2} : \ \mathbf{w} \cdot \mathbf{x} + b = -1. \tag{5.33}$$

The margin of the decision boundary is given by the distance between these two hyperplanes. To compute the margin, let $\mathbf{x}_1$ be a data point located on b_{i1} and $\mathbf{x}_2$ be a data point on b_{i2}, as shown in Figure 5.23. Upon substituting these points into Equations 5.32 and 5.33, the margin d can be computed by subtracting the second equation from the first equation:

$$\begin{aligned} &\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 2 \\ &\|\mathbf{w}\| \times d - 2 \\ &\therefore d = \frac{2}{\|\mathbf{w}\|}. \end{aligned} \tag{5.34}$$

Learning a Linear SVM Model: -

The training phase of SVM involves estimating the parameters w and b of the decision boundary from the training data. The parameters must be chosen in such a way that the following two conditions are met:

$$\begin{aligned} &\mathbf{w} \cdot \mathbf{x_i} + b \geq \ 1 \text{ if } y_i = 1, \\ &\mathbf{w} \cdot \mathbf{x_i} + b \leq -1 \text{ if } y_i = -1. \end{aligned} \tag{5.35}$$

These conditions impose the requirements that all training instances from class $y = 1$ (i.e., the squares) must be located on or above the hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 1$, while those instances from class $y = -1$ (i.e., the circles) must be located on or below the hyperplane $\mathbf{w} \cdot \mathbf{x} + b = -1$. Both inequalities can be summarized in a more compact form as follows:

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1, \quad i = 1, 2, \ldots, N. \tag{5.36}$$

Although the preceding conditions are also applicable to any linear classifiers (including perceptrons), SVM imposes an additional requirement that the margin of its decision boundary must be maximal. Maximizing the margin, however, is equivalent to minimizing the following objective function:

$$f(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2}. \tag{5.37}$$

Definition 5.1 (Linear SVM: Separable Case). The learning task in SVM can be formalized as the following constrained optimization problem:

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2}$$

$$\text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1, \quad i = 1, 2, \ldots, N.$$

Since the objective function is quadratic and the constraints are linear in the parameters w and b, this is known as a convex optimization problem, which can be solved using the standard **Lagrange multiplier** method. Following is a brief sketch of the main ideas for solving the optimization problem.
First, we must rewrite the objective function in a form that takes into account the constraints imposed on its solutions. The new objective function is known as the Lagrangian for the optimization problem:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \lambda_i \left(y_i(\mathbf{w} \cdot \mathbf{x_i} + b) - 1 \right), \tag{5.38}$$

Where the parameters λ_i are called the Lagrange multipliers.

The first term in the Lagrangian is the same as the original objective function, while the second term captures the inequality constraints.

It is easy to show that the function is minimized when w = 0, a null vector whose components are all zeros. Such a solution, however, violates the constraints given in Definition 5.1 because there is no feasible solution for b. The solutions for w and b are infeasible if they violate the inequality constraints; i.e., if $y_i(w \cdot x_i + b) - 1 < 0$. The Lagrangian given in Equation 5.38 incorporates this constraint by subtracting the term from its original objective function. Assuming that $\lambda_i \geq 0$, it is clear that any infeasible solution may only increase the value of the Lagrangian.

To minimize the Lagrangian, we must take the derivative of LP with respect to w and b and set them to zero:

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \Longrightarrow \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i, \tag{5.39}$$

$$\frac{\partial L_p}{\partial b} = 0 \Longrightarrow \sum_{i=1}^{N} \lambda_i y_i = 0. \tag{5.40}$$

Because the Lagrange multipliers are unknown, we still cannot solve for w and b. If Definition 5.1 contains only equality instead of inequality constraints, then we can use the N equations from equality constraints along with Equations 5.39 and 5.40 to find the feasible solutions for w, b, and λ_i. Note that the Lagrange multipliers for equality constraints are free parameters that can take any values.

One way to handle the inequality constraints is to transform them into a set of equality constraints. This is possible as long as the Lagrange multipliers are restricted to be non-negative. Such transformation leads to the following constraints on the Lagrange multipliers, which are known as the Karush-KuhnTucker (KKT) conditions:

$$\lambda_i \geq 0, \tag{5.41}$$

$$\lambda_i \left[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right] = 0. \tag{5.42}$$

Solving the preceding optimization problem is still quite a daunting task because it involves a large number of parameters: w, b, and λ_i. The problem can be simplified by transforming the Lagrangian into a function of the Lagrange multipliers only (this is known as the dual problem). To do this, we first substitute Equations 5.39 and 5.40 into Equation 5.38. This will lead to the following dual formulation of the optimization problem:

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j}. \tag{5.43}$$

The key differences between the dual and primary Lagrangians are as follows:

1. The dual Lagrangian involves only the Lagrange multipliers and the training data, while the primary Lagrangian involves the Lagrange multipliers as well as parameters of the decision boundary. Nevertheless, the solutions for both optimization problems are equivalent.
2. The quadratic term in Equation 5.43 has a negative sign, which means that the original minimization problem involving the primary Lagrangian, L_P , has turned into a maximization problem involving the dual Lagrangian, L_D.

Once the λ_i's are found, we can use Equations 5.39 and 5.42 to obtain the feasible solutions for w and b. The decision boundary can be expressed as follows:

$$\left(\sum_{i=1}^{N} \lambda_i y_i \mathbf{x_i} \cdot \mathbf{x}\right) + b = 0. \qquad (5.44)$$

b is obtained by solving Equation 5.42 for the support vectors. Because the λ_i's are calculated numerically and can have numerical errors, the value computed for b may not be unique. Instead it depends on the support vector used in Equation 5.42. In practice, the average value for b is chosen to be the parameter of the decision boundary.

Linear SVM: Nonseparable Case

Figure 5.25 shows a data set that is similar to Figure 5.22, except it has two new examples, P and Q. Although the decision boundary B1 misclassifies the new examples, while B_2 classifies them correctly, this does not mean that B_2 is a better decision boundary than B_1 because the new examples may correspond to noise in the training data. B_1 should still be preferred over B_2 because it has a wider margin, and thus, is less susceptible to overfitting.

This section examines how the formulation can be modified to learn a decision boundary that is tolerable to small training errors using a method known as the soft margin approach.

More importantly, the method presented in this section allows SVM to construct a linear decision boundary even in situations where the classes are not linearly separable. To do this, the learning algorithm in SVM must consider the trade-off between the width of the margin and the number of training errors committed by the linear decision boundary.

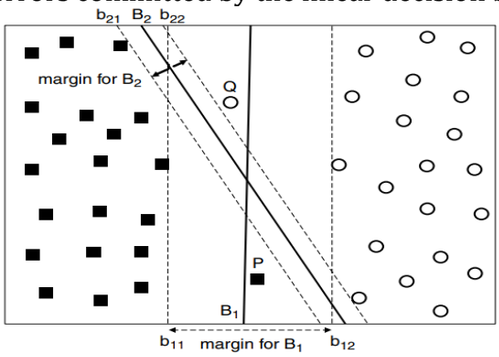


Figure 5.25. Decision boundary of SVM for the nonseparable case.

While the original objective function given in Equation 5.37 is still applicable, the decision boundary B_1 no longer satisfies all the constraints given in Equation 5.36. The inequality constraints must therefore be relaxed to accommodate the nonlinearly separable data. This can be done by introducing positive-valued slack variables (ξ) into the constraints of the optimization problem, as shown in the following equations:

$$\mathbf{w} \cdot \mathbf{x_i} + b \geq 1 - \xi_i \quad \text{if } y_i = 1,$$
$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1 + \xi_i \quad \text{if } y_i = -1, \qquad (5.45)$$

Where $\forall i: \xi_i > 0$.

To interpret the meaning of the slack variables ξ_i, consider the diagram shown in Figure 5.26. The circle **P** is one of the instances that violates the constraints given in Equation 5.35. Let $\mathbf{w} \cdot \mathbf{x} + b = -1 + \xi$ denote a line that is parallel to the decision boundary and passes through the point **P**. It can be shown that the distance between this line and the hyperplane $\mathbf{w} \cdot \mathbf{x} + b = -1$ is $\xi/\|\mathbf{w}\|$. Thus, ξ provides an estimate of the error of the decision boundary on the training example **P**.

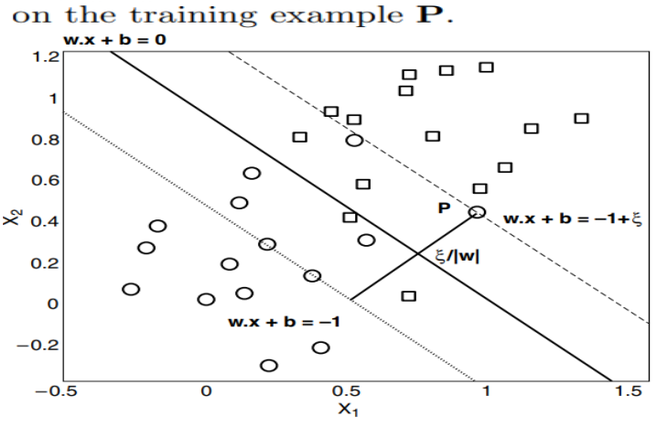


Figure 5.26. Slack variables for nonseparable data.

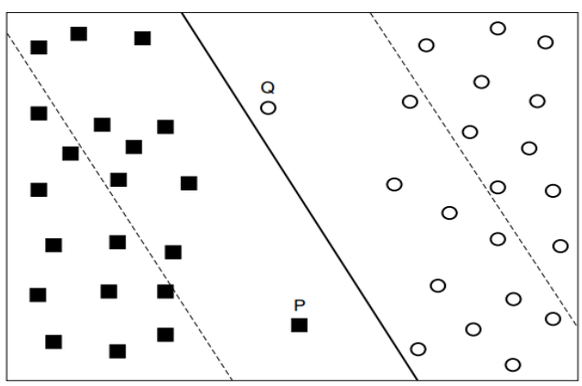


Figure 5.27. A decision boundary that has a wide margin but large training error.

In principle, we can apply the same objective function as before and impose the conditions given in Equation 5.45 to find the decision boundary. However, since there are no constraints on the number of mistakes the decision boundary can make, the learning algorithm may find a decision boundary with a very wide margin but misclassifies many of the training examples, as shown in Figure 5.27. To avoid this

problem, the objective function must be modified to penalize a decision boundary with large values of slack variables. The modified objective function is given by the following equation:

$$f(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2} + C(\sum_{i=1}^{N} \xi_i)^k,$$

Where C and k are user-specified parameters representing the penalty of misclassifying the training instances. For the remainder of this section, we assume k = 1 to simplify the problem. The parameter C can be chosen based on the model's performance on the validation set.

It follows that the Lagrangian for this constrained optimization problem can be written as follows:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\lambda_i\{y_i(\mathbf{w}\cdot\mathbf{x_i}+b) - 1 + \xi_i\} - \sum_{i=1}^{N}\mu_i\xi_i, \quad (5.46)$$

Where the first two terms are the objective function to be minimized, the third term represents the inequality constraints associated with the slack variables, and the last term is the result of the non-negativity requirements on the values of ξ_i's. Furthermore, the inequality constraints can be transformed into equality constraints using the following KKT conditions:

$$\xi_i \geq 0, \quad \lambda_i \geq 0, \quad \mu_i \geq 0, \quad (5.47)$$

$$\lambda_i\{y_i(\mathbf{w}\cdot\mathbf{x}_i+b) - 1 + \xi_i\} = 0, \quad (5.48)$$

$$\mu_i\xi_i = 0. \quad (5.49)$$

Note that the Lagrange multiplier λ_i given in Equation 5.48 is non-vanishing only if the training instance resides along the lines $\mathbf{w}\cdot\mathbf{x}_i + b = \pm 1$ or has $\xi_i > 0$. On the other hand, the Lagrange multipliers μ_i given in Equation 5.49 are zero for any training instances that are misclassified (i.e., having $\xi_i > 0$).

Setting the first-order derivative of L with respect to $\mathbf{w}$, b, and ξ_i to zero would result in the following equations:

$$\frac{\partial L}{\partial w_j} = w_j - \sum_{i=1}^{N}\lambda_i y_i x_{ij} = 0 \implies w_j = \sum_{i=1}^{N}\lambda_i y_i x_{ij}. \quad (5.50)$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{N}\lambda_i y_i = 0 \implies \sum_{i=1}^{N}\lambda_i y_i = 0. \quad (5.51)$$

$$\frac{\partial L}{\partial \xi_i} = C - \lambda_i - \mu_i = 0 \implies \lambda_i + \mu_i = C. \quad (5.52)$$

Substituting Equations 5.50, 5.51, and 5.52 into the Lagrangian will produce the following dual Lagrangian:

$$L_D = \sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j \mathbf{x_i}\cdot\mathbf{x_j}, \quad (5.53)$$

Which turns out to be identical to the dual Lagrangian for linearly separable data (see Equation 5.40). Nevertheless, the constraints imposed on the Lagrange multipliers λ_i's are slightly different those in the linearly separable case.

In the linearly separable case, the Lagrange multipliers must be non-negative, i.e., $\lambda_i \geq 0$. On the other hand, Equation 5.52 suggests that λ_i should not exceed C (since both μ_i and λ_i are non-negative). Therefore, the Lagrange multipliers for nonlinearly separable data are restricted to $0 \leq \lambda_i \leq C$.

Bayesian Classifiers: -

In many applications the relationship between the attribute set and the class variable is non-deterministic. In other words, the class label of a test record cannot be predicted with certainty even though its attribute set is identical to some of the training examples. This situation may arise because of noisy data or the presence of certain confounding factors that affect classification but are not included in the analysis. For example, consider the task of predicting whether a person is at risk for heart disease based on the person's diet and workout frequency. Although most people who eat healthily and exercise regularly have less chance of developing heart disease, they may still do so because of other factors such as heredity, excessive smoking, and alcohol abuse. Determining whether a person's diet is healthy or the workout frequency is sufficient is also subject to interpretation, which in turn may introduce uncertainties into the learning problem.

Bayes Theorem: -

Consider a football game between two rival teams: Team 0 and Team 1. Suppose Team 0 wins 65% of the time and Team 1 wins the remaining matches. Among the games won by Team 0, only 30% of them come from playing on Team 1's football field. On the other hand, 75% of the victories for Team 1 are obtained while playing at home. If Team 1 is to host the next match between the two teams, which team will most likely emerge as the winner?

This question can be answered by using the well-known Bayes theorem.

Let X and Y be a pair of random variables. Their joint probability, P(X = x, Y = y), refers to the probability that variable X will take on the value x and variable Y will take on the value y. A conditional probability is the probability that a random variable will take on a particular value given that the outcome for another random variable is known. For example, the conditional probability P(Y = y|X = x) refers to the probability that the variable Y will take on the value y, given that the variable X is observed to have the value x. The joint and conditional probabilities for X and Y are related in the following way:

$$P(X, Y) = P(Y|X) \times P(X) = P(X|Y) \times P(Y). \qquad (5.9)$$

Rearranging the last two expressions in Equation 5.9 leads to the following formula, known as the Bayes theorem:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}. \qquad (5.10)$$

The Bayes theorem can be used to solve the prediction problem stated at the beginning of this section. For notational convenience, let X be the random variable that represents the team hosting the match and Y be the random variable that represents the winner of the match. Both X and Y can take on values from the set {0, 1}. We can summarize the information given in the problem as follows:

Probability Team 0 wins is $P(Y = 0) = 0.65$.
Probability Team 1 wins is $P(Y = 1) = 1 - P(Y = 0) = 0.35$.
Probability Team 1 hosted the match it won is $P(X = 1|Y = 1) = 0.75$.
Probability Team 1 hosted the match won by Team 0 is $P(X = 1|Y = 0) = 0.3$.

Our objective is to compute P(Y = 1|X = 1), which is the conditional probability that Team 1 wins the next match it will be hosting, and compares it against P(Y = 0|X = 1). Using the Bayes theorem, we obtain

$$\begin{aligned} P(Y = 1|X = 1) &= \frac{P(X = 1|Y = 1) \times P(Y = 1)}{P(X = 1)} \\ &= \frac{P(X = 1|Y = 1) \times P(Y = 1)}{P(X = 1, Y = 1) + P(X = 1, Y = 0)} \\ &= \frac{P(X = 1|Y = 1) \times P(Y = 1)}{P(X = 1|Y = 1)P(Y = 1) + P(X = 1|Y = 0)P(Y = 0)} \\ &= \frac{0.75 \times 0.35}{0.75 \times 0.35 + 0.3 \times 0.65} \\ &= 0.5738, \end{aligned}$$

Where the law of total probability was applied in the second line. Furthermore, P(Y = 0|X = 1) = 1 – P(Y = 1|X = 1) = 0.4262. Since P(Y = 1|X = 1) > P(Y = 0|X = 1), Team 1 has a better chance than Team 0 of winning the next match.

Using the Bayes Theorem for Classification: -

Let X denotes the attribute set and Y denote the class variable. If the class variable has a non-deterministic relationship with the attributes, then we can treat X and Y as random variables and capture their relationship probabilistically using P(Y |X). This conditional probability is also known as the posterior probability for Y, as opposed to its prior probability, P(Y).

During the training phase, we need to learn the posterior probabilities P(Y |X) for every combination of X and Y based on information gathered from the training data. By knowing these probabilities, a test record X^l can be classified by finding the class Y^l that maximizes the posterior probability, $P(Y^l|X^l)$. To illustrate this approach, consider the task of predicting whether a loan borrower will default on their payments. Figure 5.9 shows a training set with the following attributes: Home Owner, Marital Status, and Annual Income. Loan borrowers who defaulted on their payments are classified as Yes, while those who repaid their loans are classified as No.

	binary	categorical	continuous	class
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Figure 5.9. Training set for predicting the loan default problem.

Suppose we are given a test record with the following attribute set: X = (Home Owner = No, Marital Status = Married, Annual Income = $120K). To classify the record, we need to compute the posterior probabilities P(Yes|X) and P(No|X) based on information available in the training data. If P(Yes|X) > P(No|X), then the record is classified as Yes; otherwise, it is classified as No.

Estimating the posterior probabilities accurately for every possible combination of class label and attribute value is a difficult problem because it requires a very large training set, even for a moderate number of attributes. The Bayes theorem is useful because it allows us to express the posterior probability in terms of the prior probability P(Y), the class-conditional probability P(X|Y), and the evidence, P(X):

$$P(Y|\mathbf{X}) = \frac{P(\mathbf{X}|Y) \times P(Y)}{P(\mathbf{X})}. \qquad (5.11)$$

When comparing the posterior probabilities for different values of Y, the denominator term, P(X), is always constant, and thus, can be ignored. The prior probability P(Y) can be easily estimated from the training set by computing the fraction of training records that belong to each class. To estimate the class-conditional probabilities P(X|Y), we present two implementations of Bayesian classification methods: the na¨ıve Bayes classifier and the Bayesian belief network.

Na¨ıve Bayes Classifier: -

A na¨ıve Bayes classifier estimates the class-conditional probability by assuming that the attributes are conditionally independent, given the class label y. The conditional independence assumption can be formally stated as follows:

$$P(\mathbf{X}|Y = y) = \prod_{i=1}^{d} P(X_i|Y = y), \qquad (5.12)$$

where each attribute set $\mathbf{X} = \{X_1, X_2, \ldots, X_d\}$ consists of d attributes.

Conditional Independence: -

Before delving into the details of how a na¨ıve Bayes classifier works, let us examine the notion of conditional independence. Let **X**, **Y**, and **Z** denote three sets of random variables. The variables in **X** are said to be conditionally independent of **Y**, given **Z**, if the following condition holds:

$$P(\mathbf{X}|\mathbf{Y}, \mathbf{Z}) = P(\mathbf{X}|\mathbf{Z}). \qquad (5.13)$$

An example of conditional independence is the relationship between a person's arm length and his or her reading skills. One might observe that people with longer arms tend to have higher levels of reading skills. This relationship can be explained by the presence of a confounding factor, which is age. A young child tends to have short arms and lacks the reading skills of an adult. If the age of a person is fixed, then the observed relationship between arm length and reading skills disappears. Thus, we can conclude that arm length and reading skills are conditionally independent when the age variable is fixed.

The conditional independence between X and Y can also be written into a form that looks similar to Equation 5.12:

$$\begin{aligned} P(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) &= \frac{P(\mathbf{X}, \mathbf{Y}, \mathbf{Z})}{P(\mathbf{Z})} \\ &= \frac{P(\mathbf{X}, \mathbf{Y}, \mathbf{Z})}{P(\mathbf{Y}, \mathbf{Z})} \times \frac{P(\mathbf{Y}, \mathbf{Z})}{P(\mathbf{Z})} \\ &= P(\mathbf{X}|\mathbf{Y}, \mathbf{Z}) \times P(\mathbf{Y}|\mathbf{Z}) \\ &= P(\mathbf{X}|\mathbf{Z}) \times P(\mathbf{Y}|\mathbf{Z}), \qquad (5.14) \end{aligned}$$

Where Equation 5.13 was used to obtain the last line of Equation 5.14.

How a Na¨ıve Bayes Classifier Works: -

With the conditional independence assumption, instead of computing the class-conditional probability for every combination of X, we only have to estimate the conditional probability of each X_i, given Y. The latter approach is more practical because it does not require a very large training set to obtain a good estimate of the probability.

To classify a test record, the na¨ıve Bayes classifier computes the posterior probability for each class Y:

$$P(Y|\mathbf{X}) = \frac{P(Y) \prod_{i=1}^{d} P(X_i|Y)}{P(\mathbf{X})}. \qquad (5.15)$$

Since $P(\mathbf{X})$ is fixed for every Y, it is sufficient to choose the class that maximizes the numerator term, $P(Y) \prod_{i=1}^{d} P(X_i|Y)$.

Estimating Conditional Probabilities for Categorical Attributes: -

For a categorical attribute X_i, the conditional probability $P(X_i = x_i|Y = y)$ is estimated according to the fraction of training instances in class y that take on a particular attribute value x_i. For example, in the training set given in Figure 5.9, three out of the seven people who repaid their loans also own a home. As a result, the conditional probability for P(Home Owner=Yes|No) is equal to 3/7. Similarly, the conditional probability for defaulted borrowers who are single is given by P(Marital Status = Single|Yes)=2/3.

Estimating Conditional Probabilities for Continuous Attributes: -

1. We can discretize each continuous attribute and then replace the continuous attribute value with its corresponding discrete interval. This approach transforms the continuous attributes into ordinal attributes. The conditional probability $P(X_i|Y = y)$ is estimated by computing the fraction of training records belonging to class y that falls within the corresponding interval for X_i.
2. We can assume a certain form of probability distribution for the continuous variable and estimate the parameters of the distribution using the training data. A Gaussian distribution is usually chosen to represent the class-conditional probability for continuous attributes. The distribution is characterized by two parameters, its mean, μ, and variance, σ^2. For each class y_j , the class-conditional probability for attribute X_i is

$$P(X_i = x_i|Y = y_j) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp^{-\frac{(x_i-\mu_{ij})^2}{2\sigma_{ij}^2}}. \qquad (5.16)$$

The parameter μ_{ij} can be estimated based on the sample mean of X_i ($\overline{x}$) for all training records that belong to the class y_j. Similarly, σ_{ij}^2 can be estimated from the sample variance (s^2) of such training records. For example, consider the annual income attribute shown in Figure 5.9. The sample mean and variance for this attribute with respect to the class No are

$$\overline{x} = \frac{125 + 100 + 70 + \ldots + 75}{7} = 110$$

$$s^2 = \frac{(125 - 110)^2 + (100 - 110)^2 + \ldots + (75 - 110)^2}{7(6)} = 2975$$

$$s = \sqrt{2975} = 54.54.$$

Given a test record with taxable income equal to \$120K, we can compute its class-conditional probability as follows:

$$P(\text{Income=120|No}) = \frac{1}{\sqrt{2\pi}(54.54)} \exp^{-\frac{(120-110)^2}{2\times 2975}} = 0.0072.$$

M-estimate of Conditional Probability: -

If the class-conditional probability for one of the attributes is zero, then the overall posterior probability for the class vanishes. This approach of estimating class-conditional probabilities using simple fractions may seem too brittle, especially when there are few training examples available and the number of attributes is large.

In a more extreme case, if the training examples do not cover many of the attribute values, we may not be able to classify some of the test records. For example, if P(Marital Status = Divorced|No) is zero instead of 1/7, then a record with attribute set X = (Home Owner = Yes, Marital Status = Divorced, Income = \$120K) has the following class-conditional probabilities:

$$P(\mathbf{X}|\text{No}) = 3/7 \times 0 \times 0.0072 = 0.$$

$$P(\mathbf{X}|\text{Yes}) = 0 \times 1/3 \times 1.2 \times 10^{-9} = 0.$$

The na¨ıve Bayes classifier will not be able to classify the record. This problem can be addressed by using the m-estimate approach for estimating the conditional probabilities:

$$P(x_i|y_j) = \frac{n_c + mp}{n + m}, \qquad (5.18)$$

where n is the total number of instances from class y_j , n_c is the number of training examples from class y_j that take on the value x_i, m is a parameter known as the equivalent sample size, and p is a user-specified parameter. If there is no training set available (i.e., n = 0), then $P(x_i|y_j) = p$.

The conditional probability P(Status = Married|Yes) = 0 because none of the training records for the class has the particular attribute value. Using the m-estimate approach with m = 3 and p = 1/3, the conditional probability is no longer zero:

$$P(\text{Marital Status} = \text{Married}|\text{Yes}) = (0 + 3 \times 1/3)/(3 + 3) = 1/6.$$

If we assume p = 1/3 for all attributes of class Yes and p = 2/3 for all attributes of class No, then

$$\begin{aligned} P(\mathbf{X}|\text{No}) &= P(\text{Home Owner} = \text{No}|\text{No}) \times P(\text{Status} = \text{Married}|\text{No}) \\ &\quad \times P(\text{Annual Income} = \$120\text{K}|\text{No}) \\ &= 6/10 \times 6/10 \times 0.0072 = 0.0026. \\ P(\mathbf{X}|\text{Yes}) &= P(\text{Home Owner} = \text{No}|\text{Yes}) \times P(\text{Status} = \text{Married}|\text{Yes}) \\ &\quad \times P(\text{Annual Income} = \$120\text{K}|\text{Yes}) \\ &= 4/6 \times 1/6 \times 1.2 \times 10^{-9} = 1.3 \times 10^{-10}. \end{aligned}$$

The posterior probability for class No is $P(\text{No}|\mathbf{X}) = \alpha \times 7/10 \times 0.0026 = 0.0018\alpha$, while the posterior probability for class Yes is $P(\text{Yes}|\mathbf{X}) = \alpha \times 3/10 \times 1.3 \times 10^{-10} = 4.0 \times 10^{-11}\alpha$. Although the classification decision has not changed, the m-estimate approach generally provides a more robust way for estimating probabilities when the number of training examples is small.

Characteristics of Na¨ıve Bayes Classifiers: -

Na¨ıve Bayes classifiers generally have the following characteristics:

- They are robust to isolated noise points because such points are averaged out when estimating conditional probabilities from data. Na¨ıve Bayes classifiers can also handle missing values by ignoring the example during model building and classification.
- They are robust to irrelevant attributes. If X_i is an irrelevant attribute, then $P(X_i|Y)$ becomes almost uniformly distributed. The classconditional probability for X_i has no impact on the overall computation of the posterior probability.

Nearest-Neighbor classifiers: -

The classification framework shown in Figure 4.3 involves a two-step process:
(1) an inductive step for constructing a classification model from data, and
(2) a deductive step for applying the model to test examples.

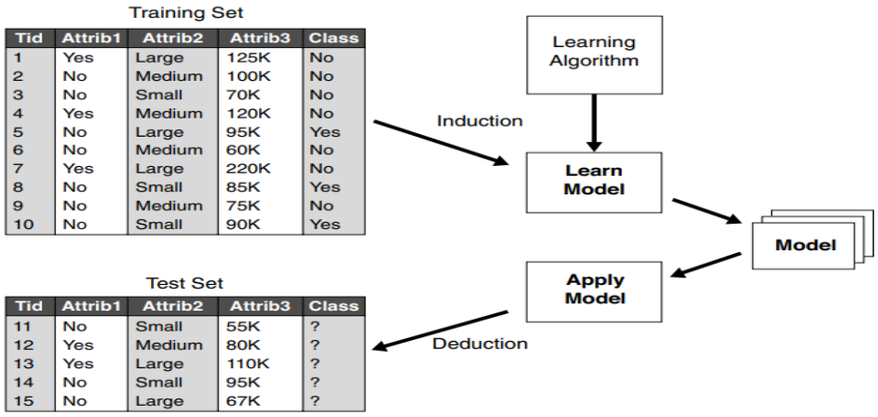


Figure 4.3. General approach for building a classification model.

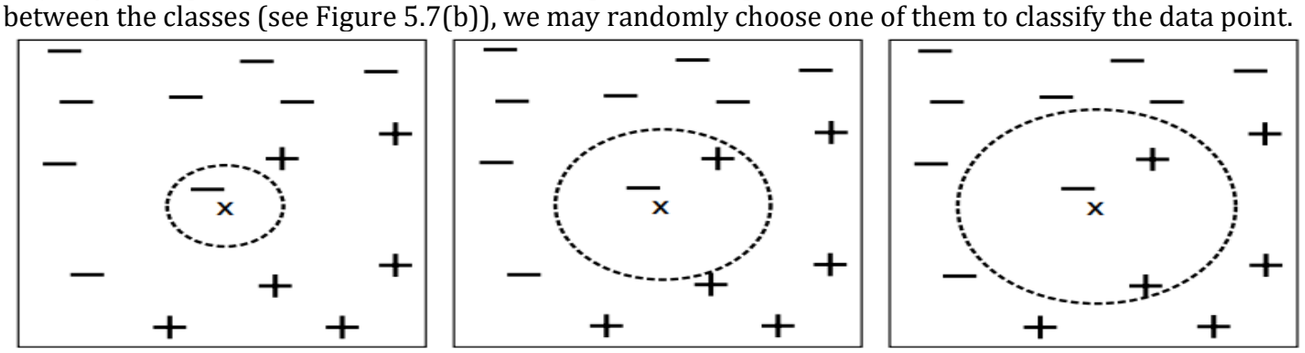
Decision tree and rule-based classifiers are examples of eager learners because they are designed to learn a model that maps the input attributes to the class label as soon as the training data becomes available.

An opposite strategy would be to delay the process of modeling the training data until it is needed to classify the test examples. Techniques that employ this strategy are known as lazy learners.

An example of a lazy learner is the Rote classifier, which memorizes the entire training data and performs classification only if the attributes of a test instance match one of the training examples exactly. An obvious drawback of this approach is that some test records may not be classified because they do not match any training example.

One way to make this approach more flexible is to find all the training examples that are relatively similar to the attributes of the test example. These examples, which are known as nearest neighbors, can be used to determine the class label of the test example. The justification for using nearest neighbors is best exemplified by the following saying: *"If it walks like a duck, quacks like a duck, and looks like a duck, then it's probably a duck."* A nearestneighbor classifier represents each example as a data point in a d-dimensional space, where d is the number of attributes. The k-nearest neighbors of a given example z refer to the k points that are closest to z.

Figure 5.7 illustrates the 1-, 2-, and 3-nearest neighbors of a data point located at the center of each circle. The data point is classified based on the class labels of its neighbors. In the case where the neighbors have more than one label, the data point is assigned to the majority class of its nearest neighbors. In Figure 5.7(a), the 1-nearest neighbor of the data point is a negative example. Therefore the data point is assigned to the negative class. If the number of nearest neighbors is three, as shown in Figure 5.7(c), then the neighborhood contains two positive examples and one negative example. Using the majority voting scheme, the data point is assigned to the positive class. In the case where there is a tie between the classes (see Figure 5.7(b)), we may randomly choose one of them to classify the data point.



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

Figure 5.7. The 1-, 2-, and 3-nearest neighbors of an instance.

The preceding discussion underscores the importance of choosing the right value for k. If k is too small, then the nearest-neighbor classifier may be susceptible to overfitting because of noise in the training data. On the other hand, if k is too large, the nearest-neighbor classifier may misclassify the test instance because its list of nearest neighbors may include data points that are located far away from its neighborhood (see Figure 5.8).

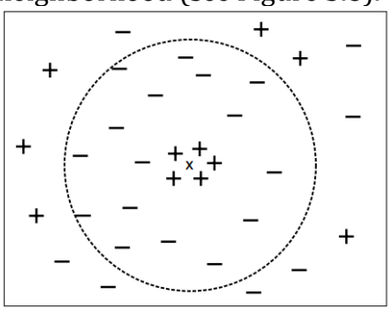


Figure 5.8. k-nearest neighbor classification with large k.

Algorithm: -

A high-level summary of the nearest-neighbor classification method is given in Algorithm 5.2. The algorithm computes the distance (or similarity) between each test example z = (x′, y′) and all the training examples (x, y) ∈ D to determine its nearest-neighbor list, D_z. Such computation can be costly if the number of training examples is large. However, efficient indexing techniques are available to reduce the amount of computations needed to find the nearest neighbors of a test example.

Algorithm 5.2 The k-nearest neighbor classification algorithm.

1: Let k be the number of nearest neighbors and D be the set of training examples.
2: **for** each test example $z = (\mathbf{x}', y')$ **do**
3: Compute $d(\mathbf{x}', \mathbf{x})$, the distance between z and every example, $(\mathbf{x}, y) \in D$.
4: Select $D_z \subseteq D$, the set of k closest training examples to z.
5: $y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$
6: **end for**

Once the nearest-neighbor list is obtained, the test example is classified based on the majority class of its nearest neighbors:

$$\text{Majority Voting:} \quad y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i), \qquad (5.7)$$

Where v is a class label, y_i is the class label for one of the nearest neighbors, and I(·) is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

In the majority voting approach, every neighbor has the same impact on the classification. This makes the algorithm sensitive to the choice of k, as shown in Figure 5.7. One way to reduce the impact of k is to weight the influence of each nearest neighbor $\mathbf{x}_i$ according to its distance: $w_i = 1/d(\mathbf{x}', \mathbf{x}_i)^2$. As a result, training examples that are located far away from z have a weaker impact on the classification compared to those that are located close to z. Using the distance-weighted voting scheme, the class label can be determined as follows:

$$\text{Distance-Weighted Voting:} \quad y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} w_i \times I(v = y_i). \qquad (5.8)$$

Characteristics of Nearest-Neighbor Classifiers: -

The characteristics of the nearest-neighbor classifier are summarized below:

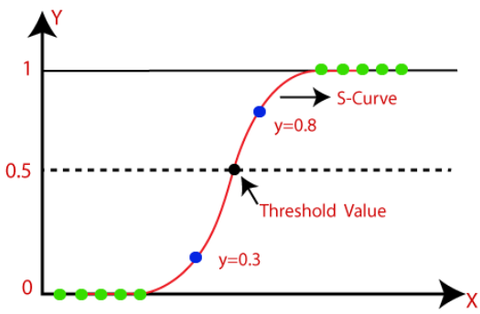
- Nearest-neighbor classification is part of a more general technique known as instance-based learning, which uses specific training instances to make predictions without having to maintain an abstraction (or model) derived from data. Instance-based learning algorithms require a proximity measure to determine the similarity or distance between instances and a classification function that returns the predicted class of a test instance based on its proximity to other instances.
- Lazy learners such as nearest-neighbor classifiers do not require model building. However, classifying a test example can be quite expensive because we need to compute the proximity values individually between the test and training examples. In contrast, eager learners often spend the bulk of their computing resources for model building. Once a model has been built, classifying a test example is extremely fast.
- Nearest-neighbor classifiers make their predictions based on local information, whereas decision tree and rule-based classifiers attempt to find a global model that fits the entire input space. Because the classification decisions are made locally, nearest-neighbor classifiers (with small values of k) are quite susceptible to noise.

- Nearest-neighbor classifiers can produce arbitrarily shaped decision boundaries. Such boundaries provide a more flexible model representation compared to decision tree and rule-based classifiers that are often constrained to rectilinear decision boundaries. The decision boundaries of nearest-neighbor classifiers also have high variability because they depend on the composition of training examples. Increasing the number of nearest neighbors may reduce such variability.
- Nearest-neighbor classifiers can produce wrong predictions unless the appropriate proximity measure and data preprocessing steps are taken. For example, suppose we want to classify a group of people based on attributes such as height (measured in meters) and weight (measured in pounds). The height attribute has a low variability, ranging from 1.5 m to 1.85 m, whereas the weight attribute may vary from 90 lb. to 250 lb. If the scale of the attributes is not taken into consideration, the proximity measure may be dominated by differences in the weights of a person.

Logistic Regression: -

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1). The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc. Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets. Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Note: - *Logistic regression uses the concept of predictive modeling as regression; therefore, it is called logistic regression, nut is used to classify samples; therefore, it falls under the classification algorithm.*

Logistic Function (Sigmoid Function):

The sigmoid function is a mathematical function used to map the predicted values to probabilities. It maps any real value into another value within a range of 0 and 1. The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function. In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- In a binary logistic regression, the dependent variable must be binary
- For a binary regression, the factor level one of the dependent variables should represent the desired outcome
- Only meaningful variables should be included
- The independent variables should be independent of each other. This means the model should have little or no multicollinearity
- The independent variables are linearly related to the log odds
- Logistic regression requires quite large sample sizes.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}\text{; 0 for y= 0, and infinity for y=1}$$

The equation of the sigmoid function is:

$$p(x) = \frac{1}{1 + e^{-(\beta_0+\beta_1 x)}}$$

The sigmoid curve obtained from the above equation is as follows:

Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

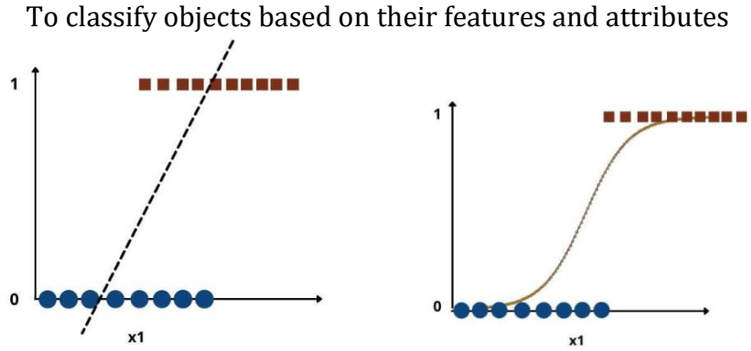
- **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
- **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

Linear Regression vs. Logistic Regression

Linear Regression	Logistic Regression
Used to solve regression problems	Used to solve classification problems
The response variables are continuous in nature	The response variable is categorical in nature
It helps estimate the dependent variable when there is a change in the independent variable	It helps to calculate the possibility of a particular event taking place
It is a straight line	It is an S-curve (S = Sigmoid)

Applications of Logistic Regression

- Using the logistic regression algorithm, banks can predict whether a customer would default on loans or not
- To predict the weather conditions of a certain place (sunny, windy, rainy, humid, etc.)
- Ecommerce companies can identify buyers if they are likely to purchase a certain product
- Companies can predict whether they will gain or lose money in the next quarter, year, or month based on their current performance
- To classify objects based on their features and attributes



Measuring Classifier Performance

The quality is classifier is measured in terms of True Positive, False Positive, True Negative and False Negative, Precision, Recall and F-Measure.

True Positive: -

Those instance where predicted class and actual class are both positive are called as true positive or a true positive is an outcome where the model correctly predicts the positive class.

For example in case of bird classifier, the birds that are correctly identified as birds are called as true positive.

True Negative: -
Those instances where predicted class and actual class are both negative are called as true negative or a true negative is an outcome where the model correctly predicts the negative class.
For example, in case of our bird classifier there are images that are not of birds which our classifier correctly identified as "not a bird" are called as true negatives.

False Positive: -
Those instances where predicted class is positive, but actually the instances are negative or a false positive is an outcome where the model incorrectly predicts the positive class.
For example, in case of our bird classifier there are some images that the classifier predicted as birds but they were something else. These are our false positives.

False Negative: -
Those instances where predicted class is negative, but actually the instances are positive or a false negative is an outcome where the model incorrectly predicts the negative class.
For example, in case of our bird classifier there are some images of birds that the classifier did not correctly recognize as birds. These are our false negatives.

EXAMPLE
A machine learning model is trained to predict tumor in patients. The test dataset consists of 100 people.

		ACTUAL	
		Negative	Positive
PREDICTION	Negative	60	8
	Positive	22	10

True Positive (TP) — model correctly predicts the positive class (prediction and actual both are positive). In the above example, **10 people** who have tumors are predicted positively by the model.

True Negative (TN) — model correctly predicts the negative class (prediction and actual both are negative). In the above example, **60 people** who don't have tumors are predicted negatively by the model.

False Positive (FP) — model gives the wrong prediction of the negative class (predicted-positive, actual-negative). In the above example, **22 people** are predicted as positive of having a tumor, although they don't have a tumor. FP is also called a **TYPE I** error.

False Negative (FN) — model wrongly predicts the positive class (predicted-negative, actual-positive). In the above example, **8 people** who have tumors are predicted as negative. FN is also called a **TYPE II** error.

With the help of these four values, we can calculate True Positive Rate (TPR), False Negative Rate (FPR), True Negative Rate (TNR), and False Negative Rate (FNR).

$$TPR = \frac{TP}{Actual\ Positive} = \frac{TP}{TP + FN}$$

$$FNR = \frac{FN}{Actual\ Positive} = \frac{FN}{TP + FN}$$

$$TNR = \frac{TN}{Actual\ Negative} = \frac{TN}{TN + FP}$$

$$FPR = \frac{FP}{Actual\ Negative} = \frac{FP}{TN + FP}$$

Even if data is imbalanced, we can figure out that our model is working well or not. For that, **the values of TPR and TNR should be high, and FPR and FNR should be as low as possible.**

Confusion Matrix: -
Confusion matrix is a N×N table that summarizes the accuracy of a classification model's predictions. Here, N represents the number of classes. In a binary classification problem, N=2.

In simple words, it is a correlation between the actual labels and the model's predicted labels. One axis of a confusion matrix is the label that the model predicted, and the other axis is the actual label.

Table 19.1 Confusion matrix for two classes

	Predicted class		
True class	Positive	Negative	Total
Positive	tp : true positive	fn : false negative	p
Negative	fp : false positive	tn : true negative	n
Total	p'	n'	N

Example: -
A machine learning model is trained to predict tumor in patients.

		Predicted	
		Tumor	**No Tumor**
Actual	**Tumor**	18 (TP)	1(FN)
	No Tumor	6(FP)	452(TN)

Basic measures derived from the confusion matrix
Various measures can be derived from a confusion matrix.
First two basic measures from the confusion matrix
Error rate (ERR) and accuracy (ACC) are the most common and intuitive measures derived from the confusion matrix.
Error rate
Error rate (ERR) is calculated as the number of all incorrect predictions divided by the total number of the dataset. The best error rate is 0.0, whereas the worst is 1.0.

$$\mathrm{ERR} = \frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FN} + \mathrm{FP}} = \frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{P} + \mathrm{N}}$$

Error rate is calculated as the total number of two incorrect predictions (FN + FP) divided by the total number of a dataset (P + N).

Accuracy
Accuracy (ACC) is calculated as the number of all correct predictions divided by the total number of the dataset. The best accuracy is 1.0, whereas the worst is 0.0. It can also be calculated by 1 – ERR.

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FN} + \mathrm{FP}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}}$$

Accuracy is calculated as the total number of two correct predictions (TP + TN) divided by the total number of a dataset (P + N).

Other basic measures from the confusion matrix
Error costs of positives and negatives are usually different. For instance, one wants to avoid false negatives more than false positives or vice versa. Other basic measures, such as sensitivity and specificity, are more informative than accuracy and error rate in such cases.

Sensitivity (Recall or True positive rate)
Sensitivity (SN) is calculated as the number of correct positive predictions divided by the total number of positives. It is also called recall (REC) or true positive rate (TPR). The best sensitivity is 1.0, whereas the worst is 0.0.

$$Recall = \frac{TP}{TP + FN}$$

Specificity (True negative rate)
Specificity (SP) is calculated as the number of correct negative predictions divided by the total number of negatives. It is also called true negative rate (TNR). The best specificity is 1.0, whereas the worst is 0.0.

$$\mathrm{SP} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

Precision (Positive predictive value)

Precision (PREC) is calculated as the number of correct positive predictions divided by the total number of positive predictions. It is also called positive predictive value (PPV). The best precision is 1.0, whereas the worst is 0.0.

$$Precision = \frac{TP}{TP + FP}$$

F1 Score

It is the harmonic mean of precision and recall. It takes both false positive and false negatives into account. Therefore, it performs well on an imbalanced dataset.

$$F1\ score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

F1 score gives the same weightage to recall and precision.

The F1 score punishes extreme values more. F1 Score could be an effective evaluation metric in the following cases:

- When FP and FN are equally costly.
- Adding more data doesn't effectively change the outcome
- True Negative is high

There is a **weighted F1 score** in which we can give different weightage to recall and precision. As discussed in the previous section, different problems give different weightage to recall and precision.

$$F_\beta = (1 + \beta^2) * \frac{(Precision * Recall)}{(\beta^2 * Precision) + Recall}$$

Beta represents how many times recall is more important than precision. If the recall is twice as important as precision, the value of Beta is 2.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\text{-}score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

What is a "good" classification model?

A good classifier should have high accuracy, precision, and recall.

In some use-cases, data scientists optimize their model to have higher precision or recall depending on the scenario.

A model with higher recall than precision often makes more positive predictions. A model like this comes with higher false positives and low false negatives.

In scenarios like disease prediction, models should always be optimized for recall. False positives are better than false negatives in the healthcare industry.

On the other hand, a model with higher precision will have fewer false positives and more false negatives. If you were to build a bot detection machine learning model for an online store, you may want to optimize for higher precision, since banning legitimate users from the website will lead to a decline in sales.

It is important to properly understand this concept, as data science interviewers often present use-cases like the ones above and ask candidates whether to optimize for precision or recall.

UNIT WISE IMPORTANT QUESTIONS: -

1. What is Supervised Learning Illustrate with an example?
2. What is Regression? Explain Linear Regression with example?
3. Make use of example discuss Multiple Linear Regression
4. Distinguish between Linear Regression and Multiple Linear Regression

5. With the help of example explain KNN.
6. Compare Root Mean Square Error (RMSE), Mean Absolute Error (MAE) with example.
7. Identify and explain regression model performance measures.
8. What is Classification Apply KNN classifier with an example
9. Illustrate basic idea behind SVM
10. Explain how Support Vector Machine can be used for classification of linearly separable data.
11. Derive the dual Lagrangian for the linear SVM with separable data where the objective function is

$$f(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2}$$

12. Explain how Support Vector Machine can be used for classification of linearly Non separable data.
13. Derive the dual Lagrangian for the linear SVM with nonseparable data where the objective function is

$$f(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^{N} \xi_i$$

14. Consider the training data in the following table where Play is a class attribute. In the table, the Humidity attribute has values "L" (for low) or "H" (for high), Sunny has values "Y" (for yes) or "N" (for no), Wind has values "S" (for strong) or "W" (for weak), and Play has values "Yes" or "No".

Humidity	Sunny	Wind	Play
L	N	S	No
H	N	W	Yes
H	Y	S	Yes
H	N	W	Yes
L	Y	S	No

Apply Naïve Bayes classifier to predict class label for the following day (Humidity=L, Sunny=N, Wind=W).

15. With the help of example illustrate Logistic Regression.
16. Identify Classifier Performance Measures and explain each measure with example.
17. Suppose 10000 patients get tested for flu; out of them, 9000 are actually healthy and 1000 are actually sick. For the sick people, a test was positive for 620 and negative for 380. For the healthy people, the same test was positive for 180 and negative for 8820.
Construct a confusion matrix for the data and compute the precision and recall for the data.
18.
    (a) Suppose the fraction of undergraduate students who smoke is 15% and the fraction of graduate students who smoke is 23%. If one-fifth of the college students are graduate students and the rest are undergraduates, what is the probability that a student who smokes is a graduate student?
    (b) Given the information in part (a), is a randomly chosen college student more likely to be a graduate or undergraduate student?
    (c) Repeat part (b) assuming that the student is a smoker.
    (d) Suppose 30% of the graduate students live in a dorm but only 10% of the undergraduate students live in a dorm. If a student smokes and lives in the dorm, is he or she more likely to be a graduate or undergraduate student? You can assume independence between students who live in a dorm and those who smoke.
19. Consider the data set shown in Table 5.10

Table 5.10. Data set for Exercise 7.

Record	A	B	C	Class
1	0	0	0	+
2	0	0	1	−
3	0	1	1	−
4	0	1	1	−
5	0	0	1	+
6	1	0	1	+
7	1	0	1	−
8	1	0	1	−
9	1	1	1	+
10	1	0	1	+

    (e) Estimate the conditional probabilities for P(A|+), P(B|+), P(C|+), P(A|−), P(B|−), and P(C|−).
    (f) Use the estimate of conditional probabilities given in the previous question to predict the class label for a test sample (A = 0, B = 1, C = 0) using the na¨ıve Bayes approach.
    (g) Estimate the conditional probabilities using the m-estimate approach, with p = 1/2 and m = 4.
    (h) Repeat part (b) using the conditional probabilities given in part (c).
    (i) Compare the two methods for estimating probabilities. Which method is better and why?
20. Consider the data set shown in Table 5.11.
    (a) Estimate the conditional probabilities for P(A = 1|+), P(B = 1|+), P(C = 1|+), P(A = 1|−), P(B = 1|−), and P(C = 1|−) using the same approach as in the previous problem.
    (b) Use the conditional probabilities in part (a) to predict the class label for a test sample (A = 1, B = 1, C = 1) using the na¨ıve Bayes approach.
    (c) Compare P(A = 1), P(B = 1), and P(A = 1, B = 1). State the relationships between A and B.

Table 5.11. Data set for Exercise 8.

Instance	A	B	C	Class
1	0	0	1	−
2	1	0	1	+
3	0	1	0	−
4	1	0	0	−
5	1	0	1	+
6	0	0	1	+
7	1	1	0	−
8	0	0	0	−
9	0	1	0	+
10	1	1	1	+

(d) Repeat the analysis in part (c) using P(A = 1), P(B = 0), and P(A = 1, B = 0).

(e) Compare P(A = 1, B = 1|Class = +) against P(A = 1|Class = +) and P(B = 1|Class = +). Are the variables conditionally independent given the class?

21.

(a) Explain how naïve Bayes performs on the data set shown in Figure 5.46.

(b) If each class is further divided such that there are four classes (A1, A2, B1, and B2), will naïve Bayes perform better?

(c) How will a decision tree perform on this data set (for the two-class problem)? What if there are four classes?

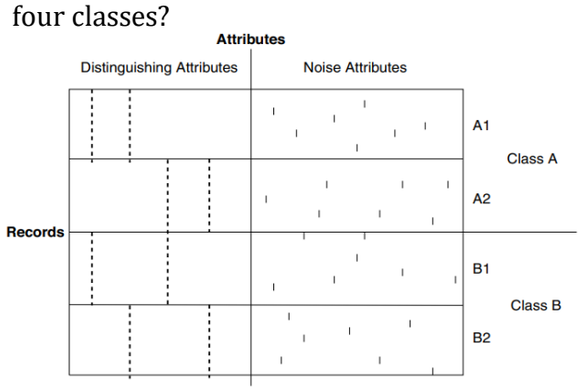


Figure 5.46. Data set for Exercise 9.

22. Consider the one-dimensional data set shown in Table 5.13.

Table 5.13. Data set for Exercise 13.

x	0.5	3.0	4.5	4.6	4.9	5.2	5.3	5.5	7.0	9.5
y	−	−	+	+	+	−	−	+	−	−

(a) Classify the data point x = 5.0 according to its 1-, 3-, 5-, and 9-nearest neighbors (using majority vote).

(b) Repeat the previous analysis using the distance-weighted voting approach.

23. The nearest-neighbor algorithm can be extended to handle nominal attributes. A variant of the algorithm called PEBLS (Parallel Examplar-Based Learning System) by Cost and Salzberg [171] measures the distance between two values of a nominal attribute using the modified value difference metric (MVDM). Given a pair of nominal attribute values, V_1 and V_2, the distance between them is defined as follows:

$$d(V_1, V_2) = \sum_{i=1}^{k} \left| \frac{n_{i1}}{n_1} - \frac{n_{i2}}{n_2} \right|, \qquad (5.84)$$

where n_{ij} is the number of examples from class i with attribute value V_j and n_j is the number of examples with attribute value V_j . Consider the training set for the loan classification problem shown in Figure 5.9. Use the MVDM measure to compute the distance between every pair of attribute values for the Home Owner and Marital Status attributes.

	binary	categorical	continuous	class
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Figure 5.9. Training set for predicting the loan default problem.

24. Derive the dual Lagrangian for the linear SVM with nonseparable data where the objective function is

$$f(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i\right)^2.$$

25. Given the data sets shown in Figures 5.49, explain how the decision tree, naïve Bayes, and k-nearest neighbor classifiers would perform on these data sets.

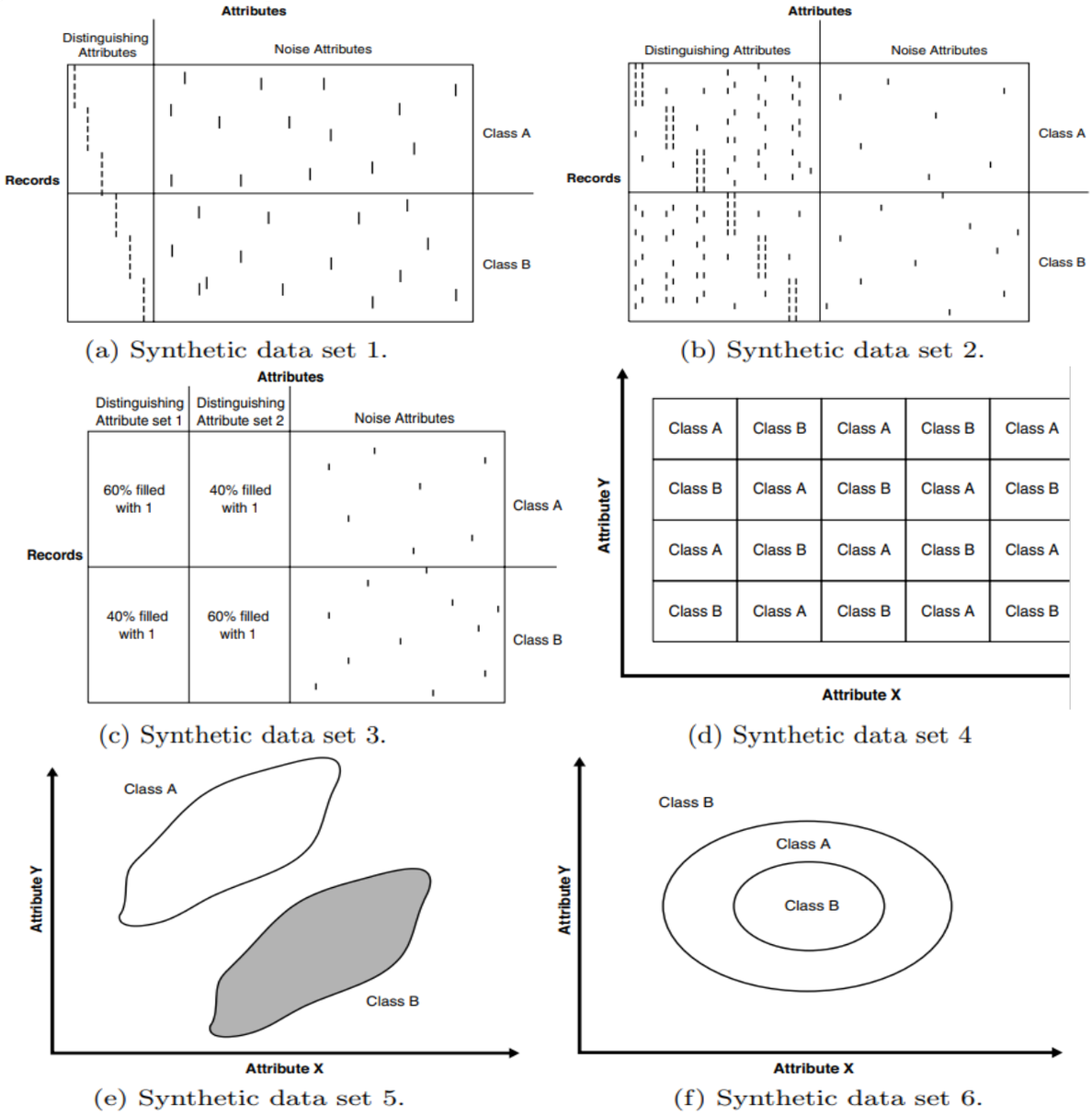


Figure 5.49. Data set

Answer: -

a) Both decision tree and NB will do well on this data set because the distinguishing attributes have better discriminating power than noise attributes in terms of entropy gain and conditional probability. K-NN will not do as well due to relatively large number of noise attributes.

b) NB will not work at all with this data set due to attribute dependency. Other schemes will do better than NB.

c) NB will do very well in this data set, because each discriminating attribute has higher conditional probability in one class over the other and the overall classification is done by multiplying these individual conditional probabilities. Decision tree will not do as well, due to the relatively large number of distinguishing attributes. It will have an over fitting problem. K-NN will do reasonably well.

d) K-NN will do well on this data set. Decision trees will also work, but will result in a fairly large decision tree. The first few splits will be quite random, because it may not find a good initial split at the beginning. NB will not perform quite as well due to the attribute dependency.

e) K-NN will do well on this data set. Decision trees will also work, but will result in a large decision tree. If decision tree uses an oblique split instead of just vertical and horizontal splits, then the resulting decision tree will be more compact and highly accurate. NB will not perform quite as well due to attribute dependency.

f) K-NN works the best. NB does not work well for this data set due to attribute dependency. Decision tree will have a large tree in order to capture the circular decision boundaries.