Cognixia

A Collabera LEARNING SOLUTIONS COMPANY

Supervised Learning - Classification

Agenda

1. Logistic Regression
2. Decision Trees
3. Random Forests
4. SVM
5. Naïve Bayes
6. Confusion Matrix

Business Scenario for Classification Models

- When the output or dependent variable has limited number of values i.e. Yes, No or True, False
- For example: voting, mortality, Churn output, and participation data is not continuous or distributed normally.
- Logistic regression is a type of regression analysis where the dependent variable is a dummy variable: coded 0 (did not vote) or 1(did vote)

Logistic Regression Model

- The "logit" model solves these problems:

  ln[p/(1-p)] = $\alpha + \beta X + e$

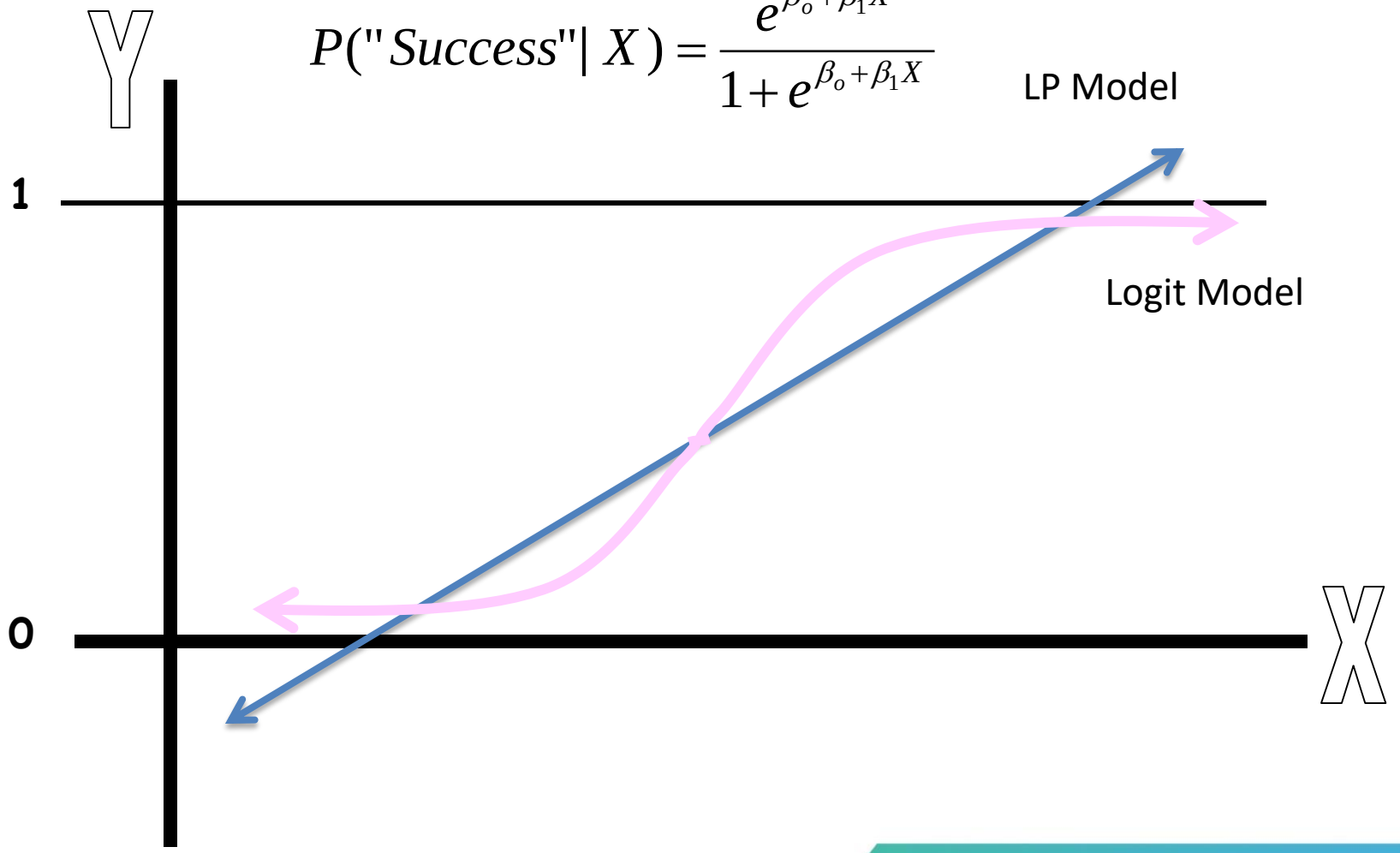
- p is the probability that the event Y occurs, p(Y=1)
- p/(1-p) is the "odds ratio"
- ln[p/(1-p)] is the log odds ratio, or "logit“
- The logistic distribution constrains the estimated probabilities to lie between 0 and 1.
- The estimated probability is:

  $p = 1/[1 + \exp(-\alpha - \beta X)]$

- if you let $\alpha + \beta X = 0$, then p = .50
- as $\alpha + \beta X$ gets really big, p approaches 1
- as $\alpha + \beta X$ gets really small, p approaches 0

Logistic Regression Model

$$P(\text{"Success"}|\ X) = \frac{e^{\beta_o + \beta_1 X}}{1 + e^{\beta_o + \beta_1 X}}$$



Logistic Regression Model

odds:

$$odds = \frac{p}{1-p}$$

The odds has a range of 0 to ∞ with values greater than 1 associated with an event being more likely to occur than to not occur and values less than 1 associated with an event that is less likely to occur than not occur.

The **logit** is defined as the log of the odds:

$$\ln(odds) = \ln\left(\frac{p}{1-p}\right) = \ln(p) - \ln(1-p)$$

This transformation is useful because it creates a variable with a range from -∞ to +∞. Hence, this transformation solves the problem we encountered in fitting a linear model to probabilities.

Logit Transformation

The logistic regression model is given by

$$P(Y \mid X) = \frac{e^{\beta_o + \beta_1 X}}{1 + e^{\beta_o + \beta_1 X}}$$

which is equivalent to

$$\ln\left(\frac{P(Y \mid X)}{1 - P(Y \mid X)}\right) = \beta_o + \beta_1 X$$

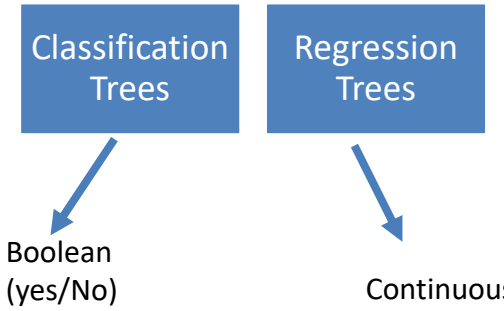
This is called the Logit Transformation

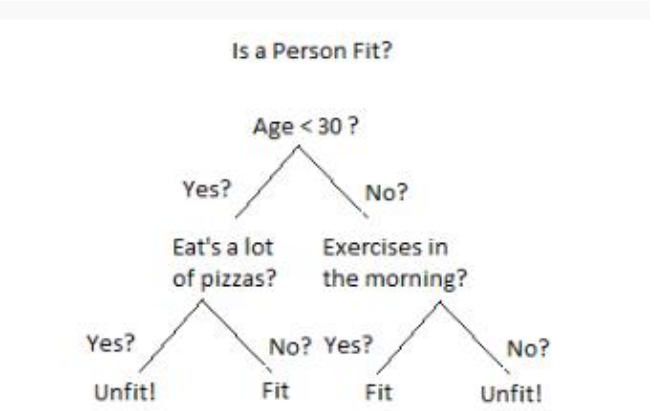
Logistic Regression

- In logistic regression the response (Y) is a dichotomous categorical variable.
- The parameter estimates give the odds ratio associated the variables in the model.
- These odds ratios are adjusted for the other variables in the model.
- One can also calculate P(Y|**X**) if that is of interest, e.g. given demographics of the mother what is the estimated probability of her having a child with low birth weight.

Decision Tree

- Decision Tree Analysis is another supervised machine learning predictive modelling tool. This is a non-parametric model used for both classification and regression tasks.
- The tree can be explained by two entities, namely decision nodes (where the data is split) and leaves (Final outcomes).
- Decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions
- *A decision tree is drawn upside down with its root at the top.*





A Simple Decision tree to analyse the fitness of a person based on age, eating habits and lifestyle.

Key requirements - Decision Tree

Attribute-value description: object or case must be expressible in terms of a fixed collection of properties or attributes (e.g., hot, mild, cold).

Predefined classes (Dependent Variable): The dependent variable has **discrete output values** (bollean or multiclass)

Sufficient data: enough training cases should be provided to learn the model.

An Example - Decision Tree

You want to guess the outcome of next week's game between the Team A and Team B.

Available data we have are

- was the game at Home or Away
- was the starting time 6 pm, 8 pm or 10 pm
- Did John play center, or forward.
- whether that opponent's center was tall or not.

An Example - Decision Tree

The data we have

Where	When	John Offense	John Defence	Opp Center	Outcome
Home	6:00 PM	Center	Forward	Tall	Won
Away	8:00 PM	Forward	Center	Tall	Lost
Away	8:00 PM	Forward	Center	Short	Won
Away	6:00 PM	Forward	Center	Short	Won
Home	10:00 PM	Center	Forward	Short	Won
Home	10:00 PM	Center	Forward	Short	Won
Home	8:00 PM	Center	Forward	Short	Lost
Home	8:00 PM	Center	Forward	Tall	Lost
Away	6:00 PM	Forward	Center	Tall	Lost
Away	6:00 PM	Forward	Center	Short	Won
Away	8:00 PM	Forward	Center	Short	Lost
Home	10:00 PM	Center	Forward	Short	Won

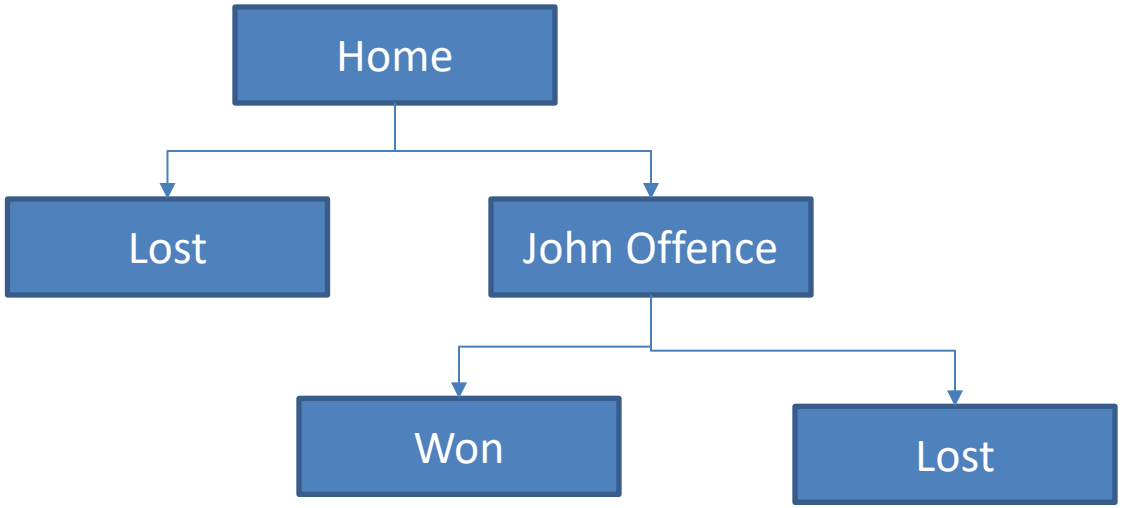
Need to predict

Where	When	John Offense	John Defence	Opp Center	Outcome
Home	6:00 PM	Center	Forward	Tall	??

An Example - Decision Tree

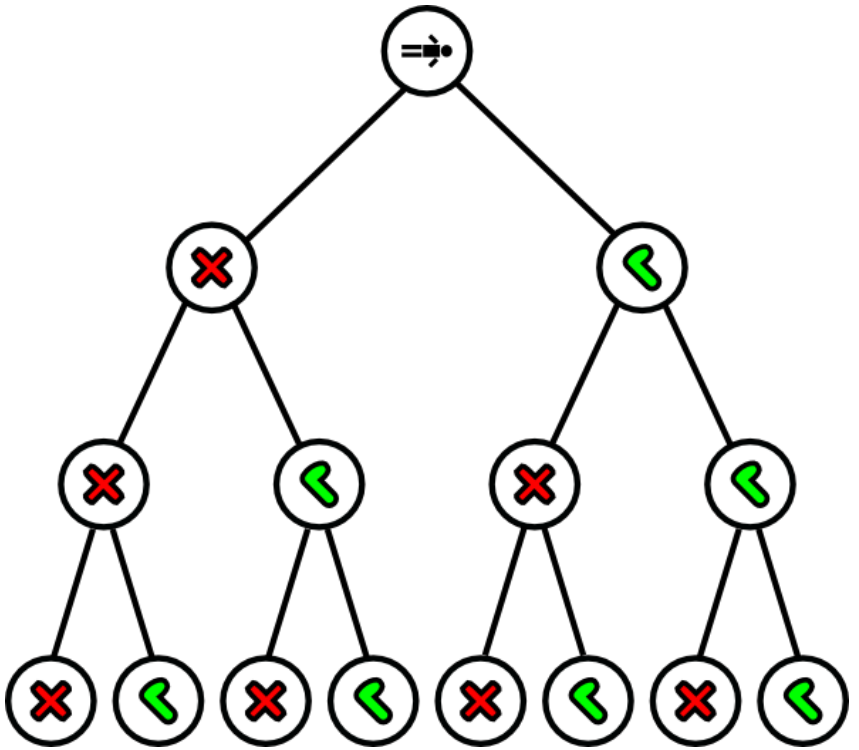
- Decision tree is a classifier in the form of a tree structure
  - Decision node: specifies a test on a single attribute
  - Leaf node: indicates the value of the target attribute
  - Arc/edge: split of one attribute
  - Path: a disjunction of test to make the final decision

- Decision trees classify instances or examples by starting at the root of the tree and moving through it until a leaf node.



An Example - Decision Tree

- The tree can grow huge
- These trees are hard to understand.
- Larger trees are typically less accurate than smaller trees.



Principle Criteria

Selection of an attribute to test at each node - choosing the most useful attribute for classifying examples.

Information gain

- measures how well a given attribute separates the training examples according to their target classification
- This measure is used to select among the candidate attributes at each step while growing the tree

Entropy

- A measure of homogeneity of the set of examples.
- Given a set S of positive and negative examples of some target concept (a 2-class problem), the entropy of set S relative to this binary classification is

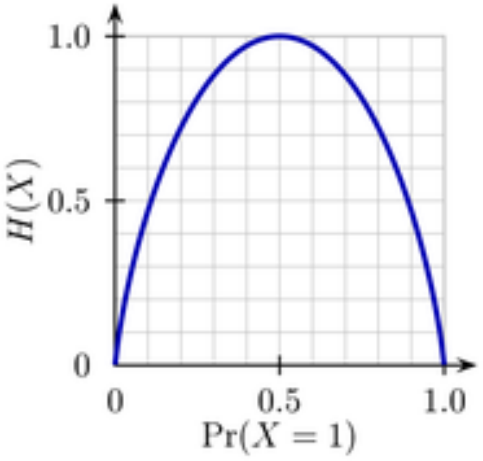
$$E(S) = - p(P)\log_2 p(P) – p(N)\log_2 p(N)$$

Entropy Calculation

- Suppose S has 25 examples, 12 positive and 13 negatives [12+, 13-]. Then the entropy of S relative to this classification is

$$E(S)=-(12/25)\log_2(12/25) - (13/25)\log_2(13/25)$$

- The entropy is 0 if the outcome is ``certain''.
- The entropy is maximum if we have no knowledge of the system (or any outcome is equally possible).



Information Gain

- Information gain measures the expected reduction in entropy, or uncertainty.

  - Values(A) is the set of all possible values for attribute A, and Sv the subset of S for which attribute A has value v Sv = {s in S | A(s) = v}.
  - the first term in the equation for *Gain* is just the entropy of the original collection *S*
  - the second term is the expected value of the entropy after S is partitioned using attribute A

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- It is simply the expected reduction in entropy caused by partitioning the examples according to this attribute.
- It is the number of bits saved when encoding the target value of an arbitrary member of *S*, by knowing the value of attribute *A*.

Decision Tree Code

```
# Decision Tree Classification

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

Decision Tree Code

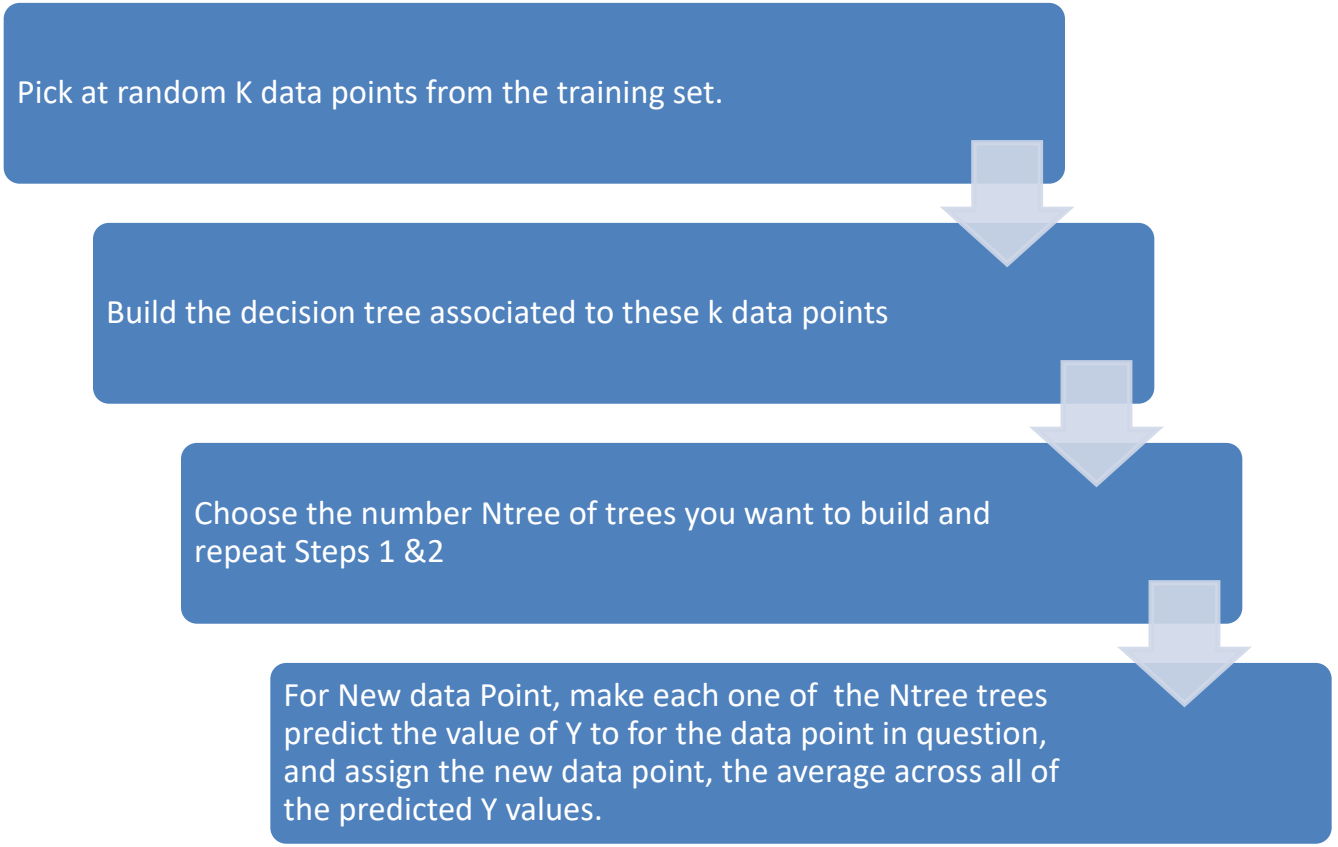
```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Random Forest Tree



Parameters of Random Forest function like: n_estimators *(no. of trees we want to build),* Criterion, max_features etc are used to improve the model as per the requirement. All these parameters have default values. So, if we don't define them, model would still run on the default values.

Random Forest Code

```
# Random Forest Classification

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

Random Forest Code

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

SVR: Support Vector Regression

- Support Vector Machine is a supervised machine learning algorithm which can be used for both classification or regression.
- Plot each data item as a point in n-dimensional space (where n is number of parameters) with the value of each parameter being the value of a particular coordinate.
- Classification is performed by hyper-plane that differentiate the two classes.
- SVR support linear and non-linear regression.

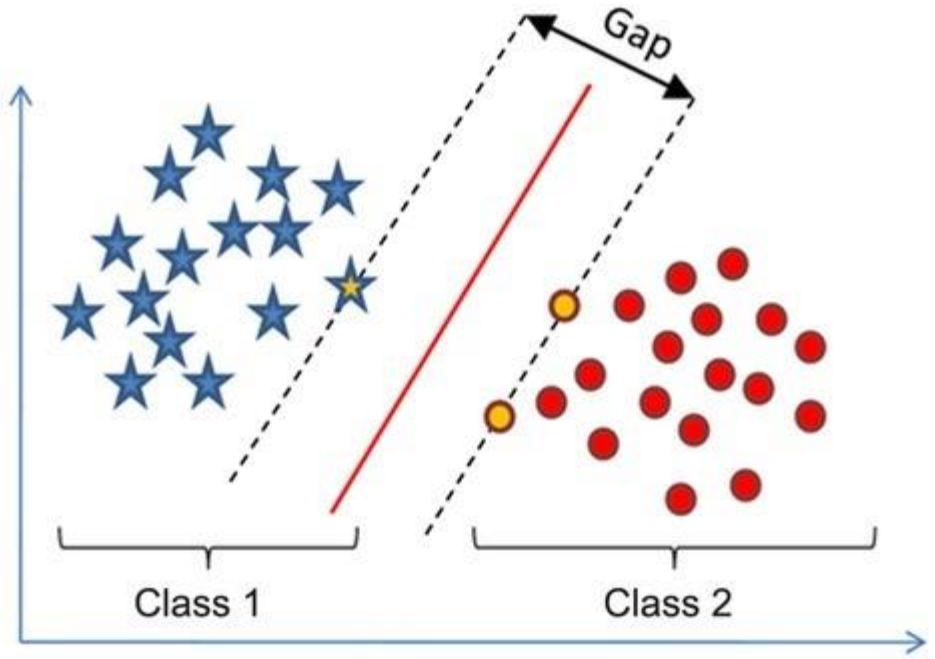
Why it is called “support vector”: It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

Linear SVR

$$y = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot \langle x_i, x \rangle + b$$

Non-Linear SVR

$$y = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot K(x_i, x) + b$$



SVM - Code

```
# Support Vector Machine (SVM)

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

SVM - Code

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting SVM to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

SVM – Kernel Function

The *kernel function* can be any of the following:

- linear: $\langle x, x' \rangle$.
- polynomial: $(\gamma \langle x, x' \rangle + r)^d$. d is specified by keyword `degree`, r by `coef0`.
- rbf: $\exp(-\gamma \|x - x'\|^2)$. γ is specified by keyword `gamma`, must be greater than 0.
- sigmoid ($\tanh(\gamma \langle x, x' \rangle + r)$), where r is specified by `coef0`.

Bayes’ Theorem and Naive Bayes Classifier

Bayes' theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event. For example, if cancer is related to age, then, using Bayes' theorem, a person's age can be used to more accurately assess the probability that they have cancer, compared to the assessment of the probability of cancer made without knowledge of the person's age. *(Source: Wikipedia)*

$$P(h|d) = (P(d|h) * P(h)) / P(d)$$

P(h|d) is the probability of hypothesis h given the data d. This is called the posterior probability.
P(d|h) is the probability of data d given that the hypothesis h was true.
P(h) is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.
P(d) is the probability of the data (regardless of the hypothesis).

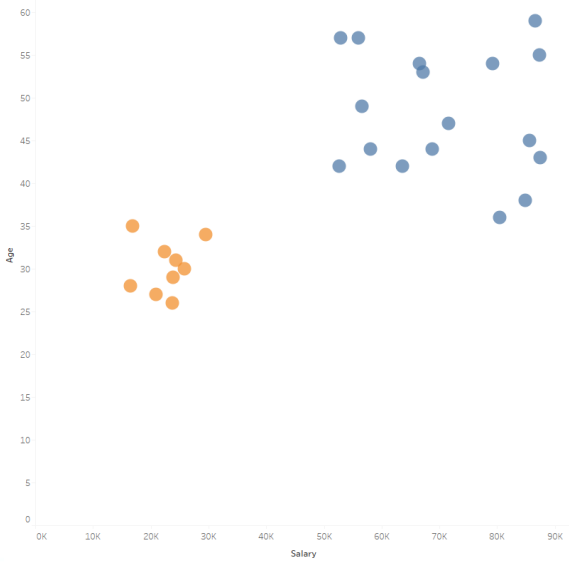
Naive Bayes Classifier is a supervised modelling tool which runs on the basis of Bayes’ theorem.
Let’s understand this with the help of an example:
Let’s say we have a data set of two categories of people :

1. Who drives to work
2. Who walks to work

The classification criteria:

a) Salary
b) Age

Refer the graph on Right



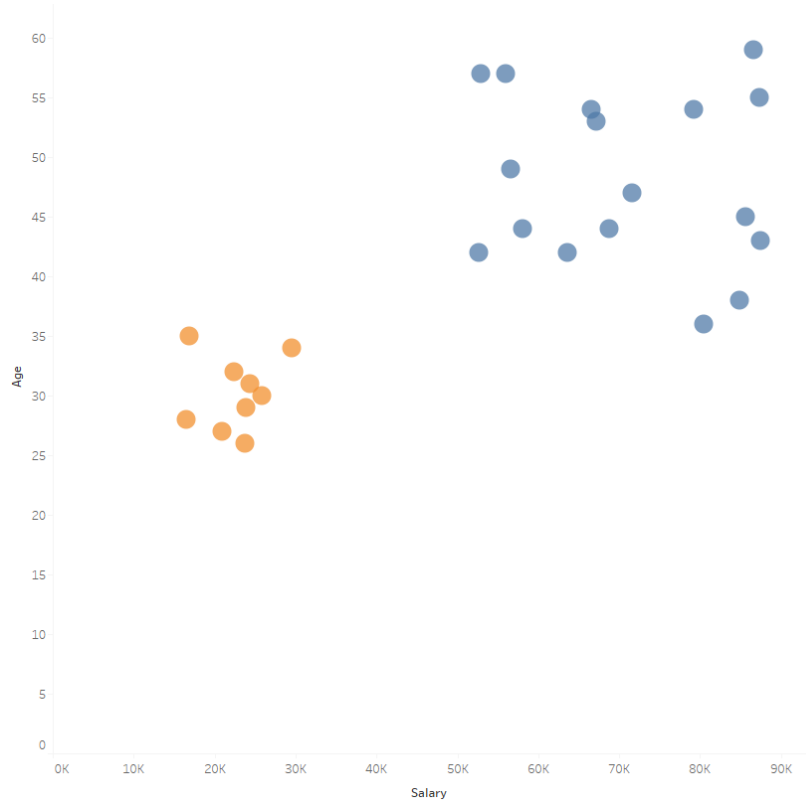
Naive Bayes Classifier: Example

Now, We have to find the likelihood of this person to drive to work or walks.

We can accomplish this by finding probability of this person to take the either option using Naïve Bayes classifier

$$P(c \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid c)P(c)}{P(\mathbf{x})}$$

$$Posterior = \frac{Likelihood \times Prior}{Evidence}$$



- Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification. There are two kind of probabilities we van find using this tool:
  - **Class Probabilities**: The probabilities of each class in the training dataset.
  - **Conditional Probabilities**: The conditional probabilities of each input value given each class value.

Naïve Bayes - Code

```
# Naive Bayes

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report,accuracy_score, confusion_matrix, auc ,roc_curve
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split
```

Naïve Bayes - Code

```
# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Naïve Bayes - Code

```
# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Model Building in Python

Step 1
- Import the Right class

Step 2
- Create the right object of the class (regression/Random Forest etc.)

Step 3:
- Fit the object in the data set using the fit method

- These are the basic steps in python while building a model.
- Object may be the simple linear regression, random forest etc.

Confusion Matrix

Visualization of the performance of an algorithm by running it on the test data for which true values are already known.
It is used for evaluating the accuracy of classification

	Predicted: NO	Predicted: YES
Actual: NO	TN = ??	FP = ??
Actual: YES	FN = ??	TP = ??

```
>>> from sklearn.metrics import confusion_matrix
>>> y_true = [2, 0, 2, 2, 0, 1]
>>> y_pred = [0, 0, 2, 2, 0, 2]
>>> confusion_matrix(y_true, y_pred)
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]])
```

Confusion Matrix: Example

- Consider a sample of 25 toys, e.g. 7 toy dolls, 8 toy cars, and 10 animals. (confusion matrix drawn below).

This confusion matrix predicted six of the eight actual toy cars, but in two cases it took a toy car for a toy doll.
The seven toy dolls were correctly recognized in six cases but in one case a toy doll was taken to be a toy car. And eight out of ten toy animals had been correctly recognized.

$$Precision_i = \frac{M_{ii}}{\sum_j M_{ji}}$$

$$Recall_i = \frac{M_{ii}}{\sum_j M_{ij}}$$

		Predicted		
		Toy Car	Toy Doll	Toy Animal
Actual	Toy Car	6	2	0
	Toy Doll	1	6	0
	Toy Animal	1	1	8

Precision(Toys):

Precision_toycar = 6/(6+1+1)=3/4=0.75
Precision_toydoll = 6/(2+6+1)=6/9=0.67
Precision_toyanimal = 8/(0+0+8)=1

Recall(Toys):
recall _toycar =6/(6+2+0)=3/4=0.75
recall _toydoll =6/(1+6+0)=6/7=0.86
recall _toyanimal =8/(1+1+8)=4/5=0.8

Python Code:

```
from sklearn.metrics import confusion_matrix
y_actu = [2, 0, 2, 2, 0, 1, 1, 2, 2, 0, 1, 2]
y_pred = [0, 0, 2, 1, 0, 2, 1, 0, 2, 0, 2, 2]
confusion_matrix(y_actu, y_pred)
```


Cognixia
A Collabera LEARNING SOLUTIONS COMPANY
THANK YOU
www.cognixia.com