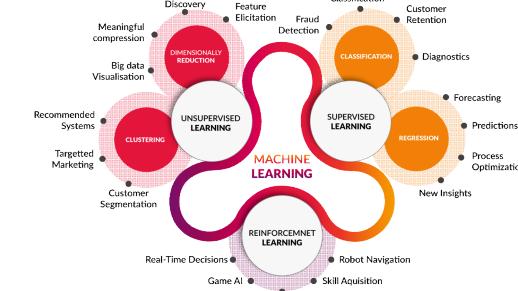


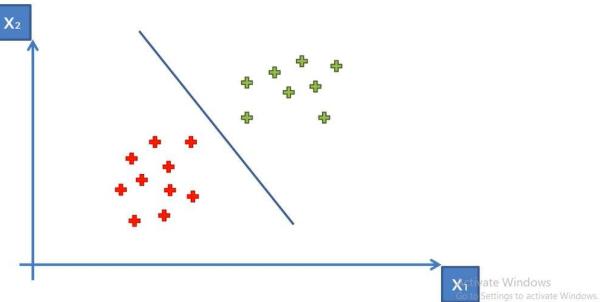
SVM, Decision Tree, Random Forest Classifiers

Machine Learning
Dr. Adnan Abid

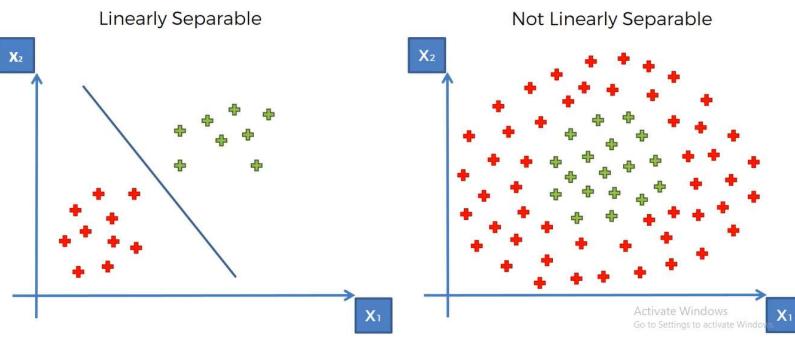


Support Vector Machine Classifier

SVM separates well these points



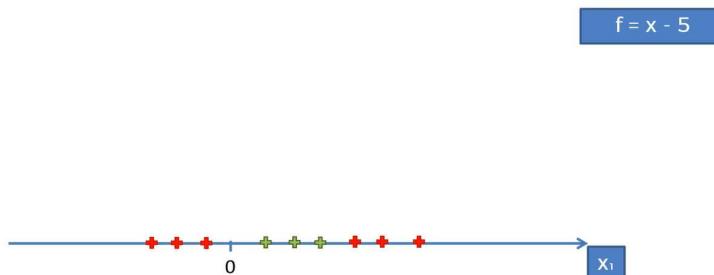
Linear Separability



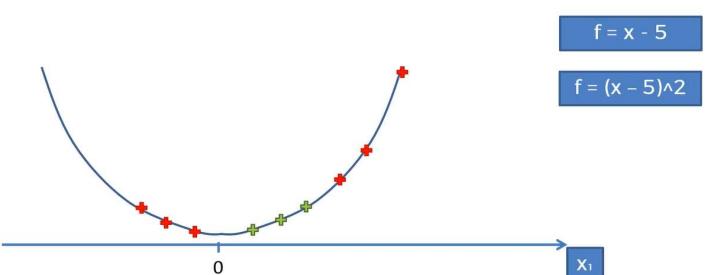
Mapping to a Higher Dimension



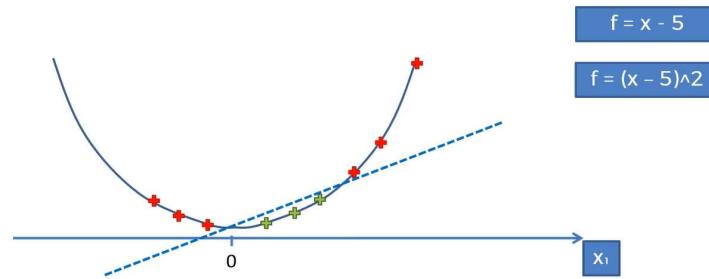
Mapping to a Higher Dimension



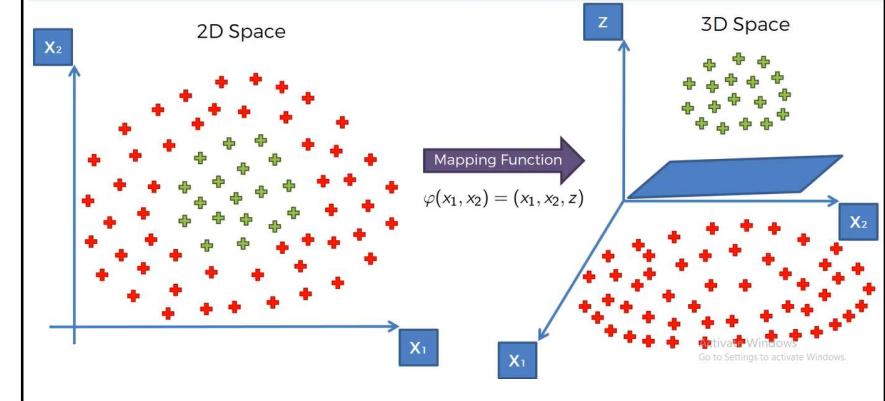
Mapping to a Higher Dimension



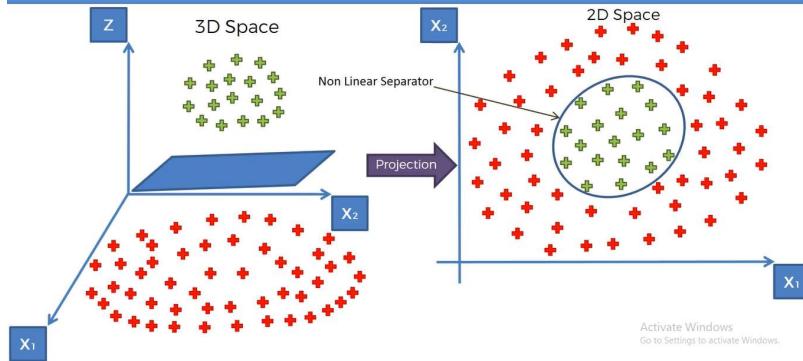
Mapping to a Higher Dimension



Mapping to a Higher Dimension



Projecting back to 2D Space



But there is a catch...

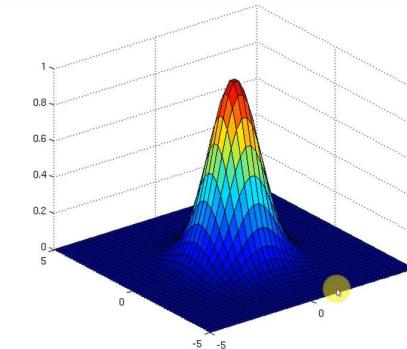
Mapping to a Higher Dimensional Space
can be highly compute-intensive

The Kernel Trick

The Gaussian RBF Kernel

$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

The Gaussian RBF Kernel

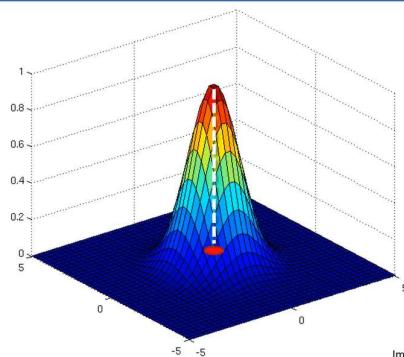


$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

Activate Windows

Image source: <http://www.cs.toronto.edu/~duvenaud/cookbook/index.html>

The Gaussian RBF Kernel



$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

Activate Windows

Image source: <http://www.cs.toronto.edu/~duvenaud/cookbook/index.html>

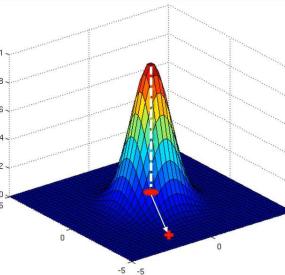
Calculate Distance of each point from a designated landmark.

$\|\vec{x} - \vec{l}^i\|^2$ is the distance calculation

Bigger distance will result into bigger numerator and e raise to the power $-\|\vec{x} - \vec{l}^i\|^2$... Will be near to 0.

While, with smaller values it will be big

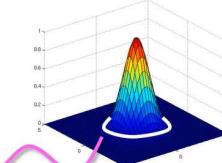
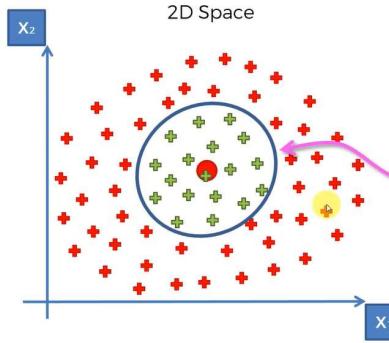
The Gaussian RBF Kernel



$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

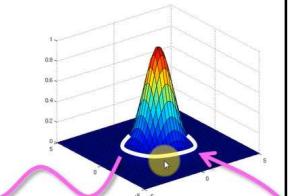
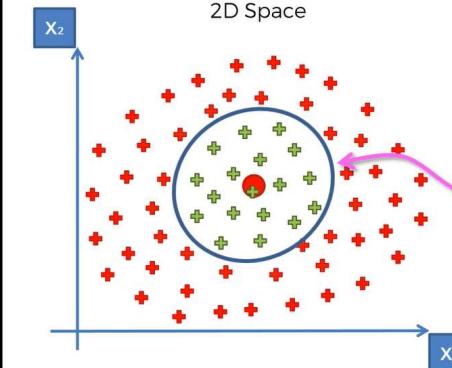
Activate Windows
Get to Seminars to activate WindowsImage source: <http://www.cs.toronto.edu/~duvenaud/cookbook/index.html>

The Gaussian RBF Kernel



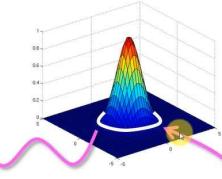
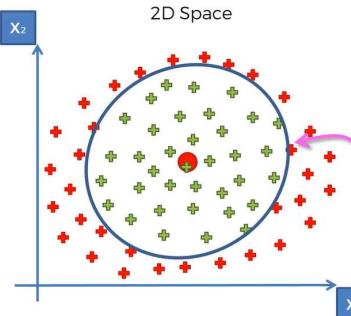
$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

The Gaussian RBF Kernel



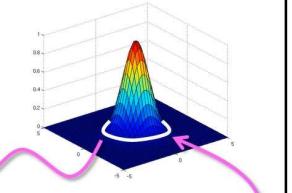
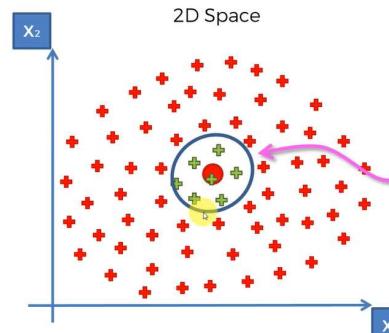
$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

The Gaussian RBF Kernel



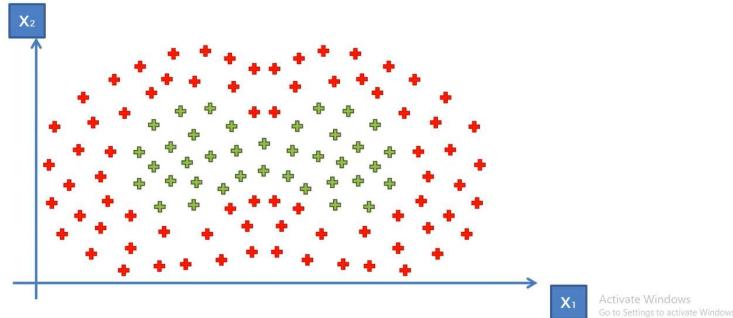
$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

The Gaussian RBF Kernel



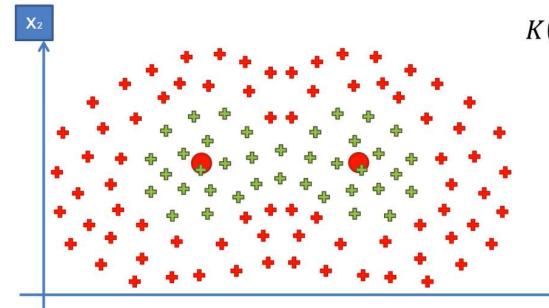
$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

The Gaussian RBF Kernel



Activate Windows
Go to Settings to activate Windows.

The Gaussian RBF Kernel



You may use two kernel functions.

Each landmark is being captured by one kernel.

This way we can cover such complex situation too using more than one kernel, without going to higher dimensions.

Activate Windows
Go to Settings to activate Windows.

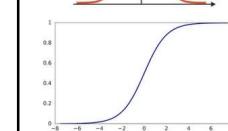
Types of Kernel Functions

Types of Kernel Functions



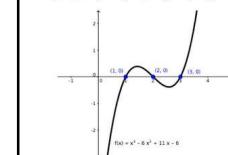
Gaussian RBF Kernel

$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x} - \vec{l}^i\|^2}{2\sigma^2}}$$



Sigmoid Kernel

$$K(X, Y) = \tanh(\gamma \cdot X^T Y + b)$$



Polynomial Kernel

$$K(X, Y) = (\gamma \cdot X^T Y + r)^d, \gamma > 0$$

Activate Windows
Go to Settings to activate Windows.

mikernels.readthedocs.io/en/latest/kernelfunctions.html

Kernel	Constructor	Merger	Negative Definite
Scalar Product Kernel	ScalarProductKernel()	✓	✗
Squared Distance Kernel	SquaredDistanceKernel(t)	✗	✓
Sine Squared Kernel	SineSquaredKernel()	✗	✓
Chi-Squared Kernel	ChiSquaredKernel()	✗	✓
Gaussian Kernel	GaussianKernel()	✗	✗
Laplacian Kernel	LaplacianKernel()	✓	✗
Periodic Kernel	PeriodicKernel()	✓	✗
Matern Kernel	MaternKernel()	✓	✗
Linear & Polynomial Kernel	PolynomialKernel(a, d)	✓	✗
Sigmoid Kernel	SigmoidKernel()	✗	✗

Scalar Product Kernel

```
ScalarProductKernel() # Construct 64 bit kernel (default)
ScalarProductKernel{Float32}() # Construct 32 bit kernel.
```

The scalar product kernel, or linear kernel, is the dot product of two vectors:

$$k(\mathbf{x}, \mathbf{y}) = \mathbf{x}'\mathbf{y}$$

The scalar product is a **Merger** kernel [berg]. This kernel is provided primarily for constructing more kernels (e.g., polynomial) kernel since it is a kernel-based algorithm.

Periodic Kernel

```
PeriodicKernel{Float32}(a, p, b)
```

The periodic kernel is given by:

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\alpha \sum_{i=1}^n \sin(p(x_i - y_i))^2\right) \quad p > 0, \alpha > 0$$

where \mathbf{x} and \mathbf{y} are n -dimensional vectors. The parameters p and α are scaling parameters for the periodicity and the magnitude, respectively. This kernel is useful when data has periodicity to it. The first three components of kernel PCA over an ellipse in \mathbb{R}^3 with a periodic kernel ($\alpha = 1$, $p = \pi$) are visualized below:

Rational-Quadratic Kernel

```
RationalQuadraticKernel{Float32}(a, b)
```

The rational-quadratic kernel is given by:

$$k(\mathbf{x}, \mathbf{y}) = (1 + \alpha ||\mathbf{x}, \mathbf{y}||^2)^{-\beta} \quad \alpha > 0, \beta > 0$$

where α is a scaling parameter and β is a shape parameter. This kernel can be seen as an infinite sum of Gaussian kernels. If one sets $\alpha = \alpha_0/\beta$, then taking the limit $\beta \rightarrow \infty$ results in the Gaussian kernel with scaling parameter α_0 . The first three components of kernel PCA over an ellipse in \mathbb{R}^3 with a rational-quadratic kernel ($\alpha = 1$, $\beta = 1$, $\gamma = 1$) are visualized below:

Periodic Kernel

Rational-Quadratic Kernel

Matern Kernel

```
MaternKernel{Float32}(a, l)
```

SquaredDistanceKernel{t}()

The squared distance kernel is a modification of the squared Euclidean distance with an additional shape parameter:

$$k(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n (x_i - y_i)^{2t} \quad 0 < t \leq 1$$

The squared distance is a **negative definite** stationary kernel [berg]. The first three components of kernel PCA over an ellipse in \mathbb{R}^3 with a squared distance kernel ($t = 0.5$) are visualized below:

Squared-Distance Kernel

Sine-Squared Kernel

Gaussian Kernel

Over a larger range, the projected manifold can be seen to fold in on itself and repeat the shape. Other common kernels include the radial basis kernel and the squared exponential covariance function (Gaussian process). The first three components of kernel PCA over an ellipse in \mathbb{R}^3 with a Gaussian kernel are visualized below:

```

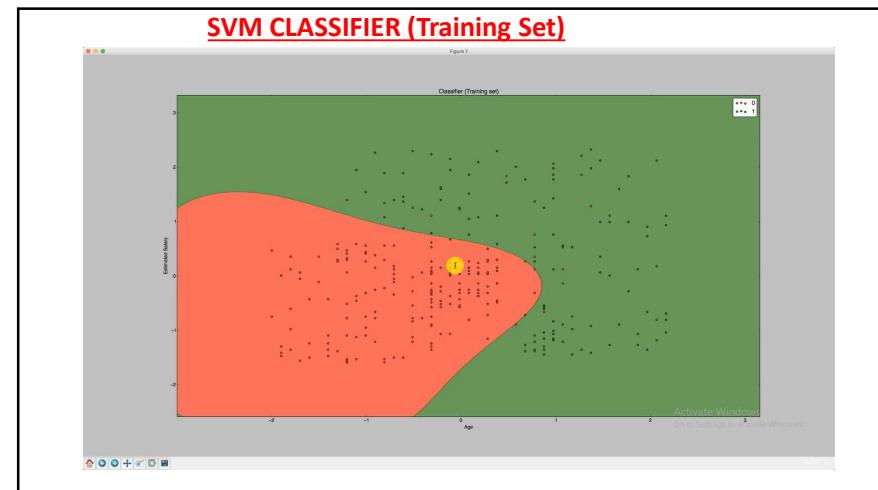
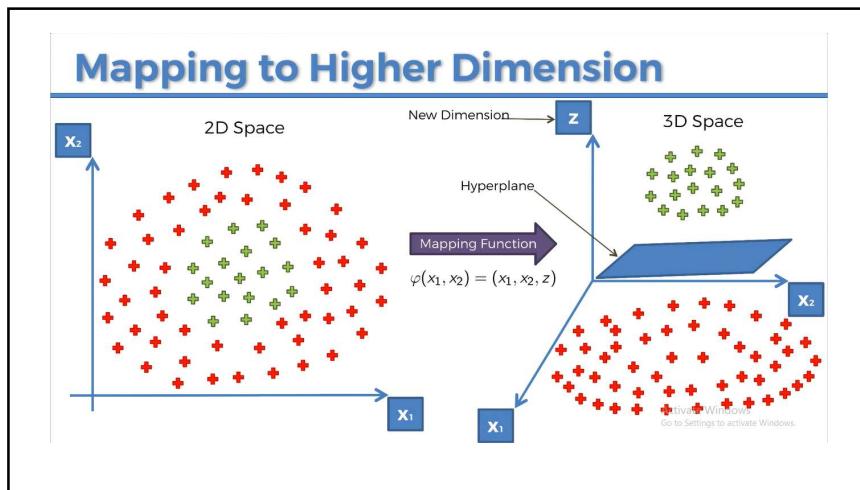
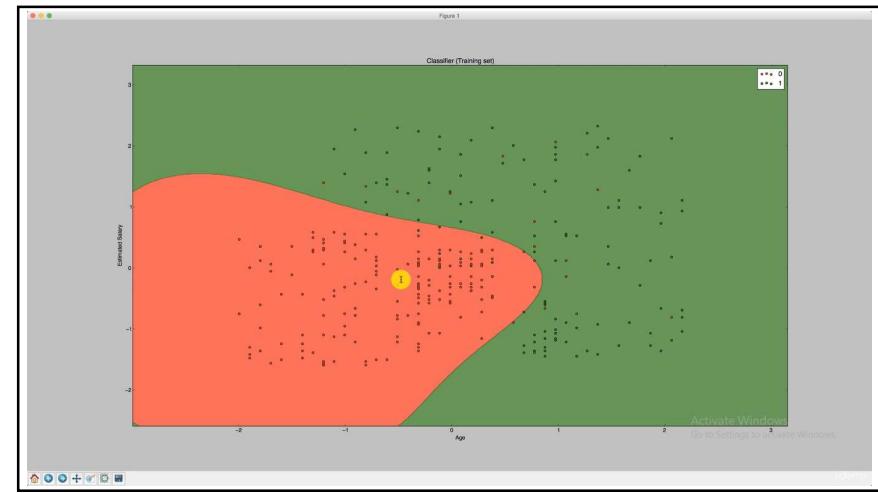
1 # Kernel SVM
2
3 # Importing the libraries
4 import numpy as np
5 import pandas as pd
6 import matplotlib.pyplot as plt
7 import pandas as pd
8
9 # Importing the dataset
10 dataset = pd.read_csv('Social_Network_Ads.csv')
11 X = dataset.iloc[:, [2, 3]].values
12 y = dataset.iloc[:, 4].values
13
14 # Splitting the dataset into the Training set and Test set
15 from sklearn.cross_validation import train_test_split
16 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
17
18 # Feature Scaling
19 from sklearn.preprocessing import StandardScaler
20 sc = StandardScaler()
21 X_train = sc.fit_transform(X_train)
22 X_test = sc.transform(X_test)
23
24 # Fitting classifier to the Training set
25 from sklearn.svm import SVC
26 classifier = SVC(kernel = 'rbf', random_state = 0)
27 classifier.fit(X_train, y_train)
28
29 # Predicting the Test set results
30 y_pred = classifier.predict(X_test)
31
32 # Making the Confusion Matrix
33 from sklearn.metrics import confusion_matrix
34 cm = confusion_matrix(y_test, y_pred)
35
36 # Visualising the Training set results
37 X_set, y_set = X_train, y_train
38 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
39                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
40 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape[0], X2.shape[1]), alpha = 0.75, cmap = ListedColormap(['red', 'green']))
41
42

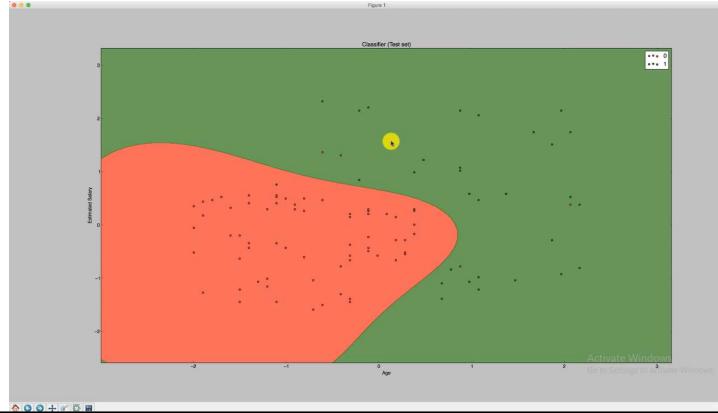
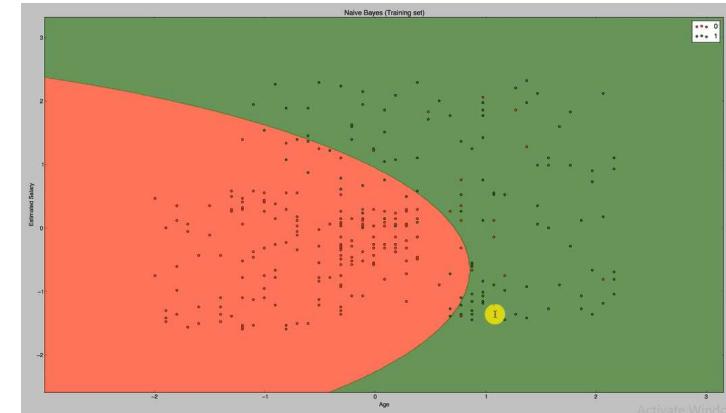
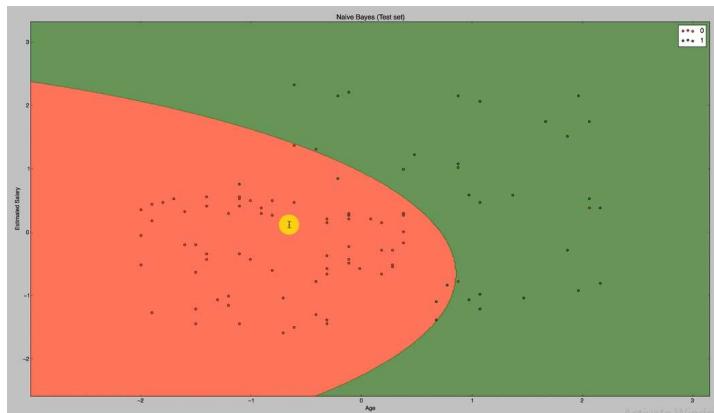
```

```

1 # Kernel: empyt
2
3 # Importing the libraries
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import pandas as pd
7
8 # Importing the dataset
9 dataset = pd.read_csv('Social_Network_Ads.csv')
10 X = dataset.iloc[:, [2, 3]].values
11 y = dataset.iloc[:, 4].values
12
13 # Splitting the dataset into the Training set and Test set
14 from sklearn.cross_validation import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
16
17 # Feature Scaling
18 from sklearn.preprocessing import StandardScaler
19 sc = StandardScaler()
20 X_train = sc.fit_transform(X_train)
21 X_test = sc.transform(X_test)
22
23 # Fitting classifier to the Training set
24 from sklearn.svm import SVC
25 classifier = SVC(kernel = 'rbf', random_state = 0)
26 classifier.fit(X_train, y_train)
27
28 # Predicting the Test set results
29 y_pred = classifier.predict(X_test)
30
31 # Making the Confusion Matrix
32 from sklearn.metrics import confusion_matrix
33 cm = confusion_matrix(y_test, y_pred)
34
35 # Visualising the Training set results
36 from matplotlib.colors import ListedColormap
37 X_set, y_set = X_train, y_train
38 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
39 np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
40 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape[0], X2.shape[1]),
41 alpha = 0.75, cmap = ListedColormap(['red', 'green']))
42
43 # Visualising the Test set results
44 from matplotlib.colors import ListedColormap
45 X_set, y_set = X_test, y_pred
46 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
47 np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
48 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape[0], X2.shape[1]),
49 alpha = 0.75, cmap = ListedColormap(['red', 'green']))
50
51

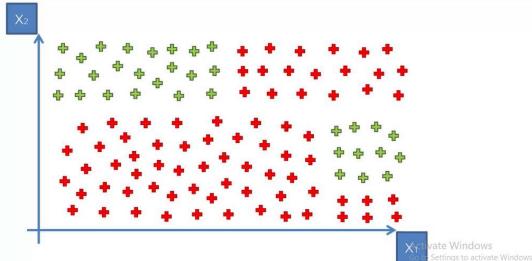
```



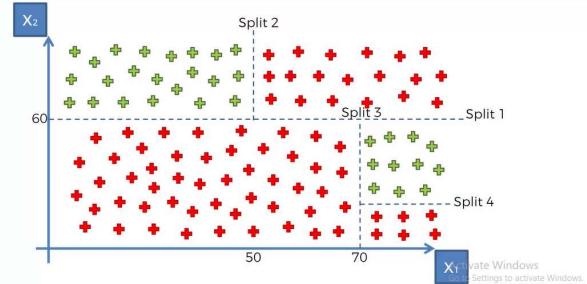
SVM CLASSIFIER (Test Set)**NAIVE BAYES CLASSIFIER (Training Set)****NAIVE BAYES CLASSIFIER (Test Set)**

Decision Tree Classifier

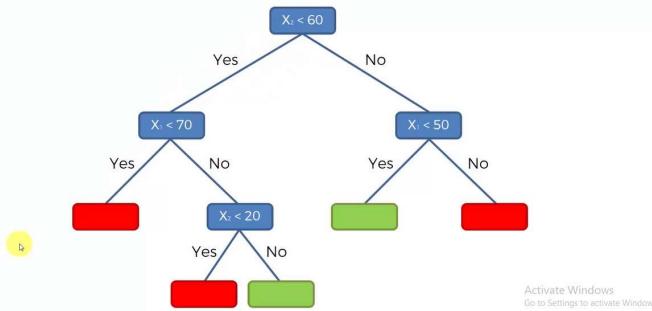
Decision Tree Intuition



Decision Tree Intuition



Split 4



```

11 y = dataset.iloc[:, 4].values
12
13 # Splitting the dataset into the Training set and Test set
14 from sklearn.cross_validation import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
16
17 # Feature Scaling
18 from sklearn.preprocessing import StandardScaler
19 sc = StandardScaler()
20 X_train = sc.fit_transform(X_train)
21 X_test = sc.transform(X_test)
22
23 # Fitting classifier to the Training set
24 from sklearn.tree import DecisionTreeClassifier
25 classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
26 classifier.fit(X_train, y_train)
27
28 # Predicting the Test set results
29 y_pred = classifier.predict(X_test)
30
31 # Making the Confusion Matrix
32 from sklearn.metrics import confusion_matrix
33 cm = confusion_matrix(y_test, y_pred)
34
35 # Visualizing the Training set results
36 from matplotlib.colors import ListedColormap
37 X_set, y_set = X_train, y_train
38 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max(), step = 0.01),
39                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max(), step = 0.01))
40 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T), alpha = 0.75, cmap = ListedColormap(['red', 'green']))
41 plt.xlim(X1.min(), X1.max())
42 plt.ylim(X2.min(), X2.max())
43 plt.scatter(X_set[y_set == 0], X_set[y_set == 1], c = ListedColormap(['red', 'green'])(y_set))
44 for i, j in enumerate(np.unique(y_set)):
45     plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(['red', 'green'])(j), label = j)
46 plt.title('Decision Tree (Training set)')
47 plt.xlabel('Age')
48 plt.ylabel('Estimated Salary')
49 plt.legend()
50 plt.show()

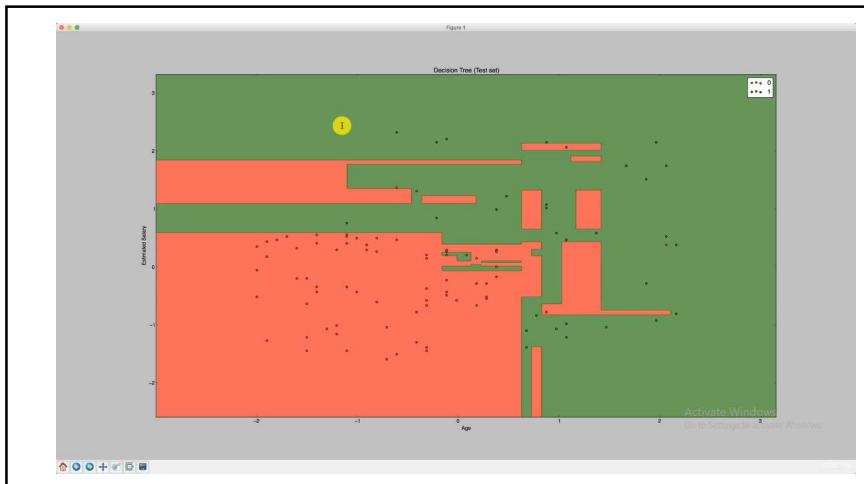
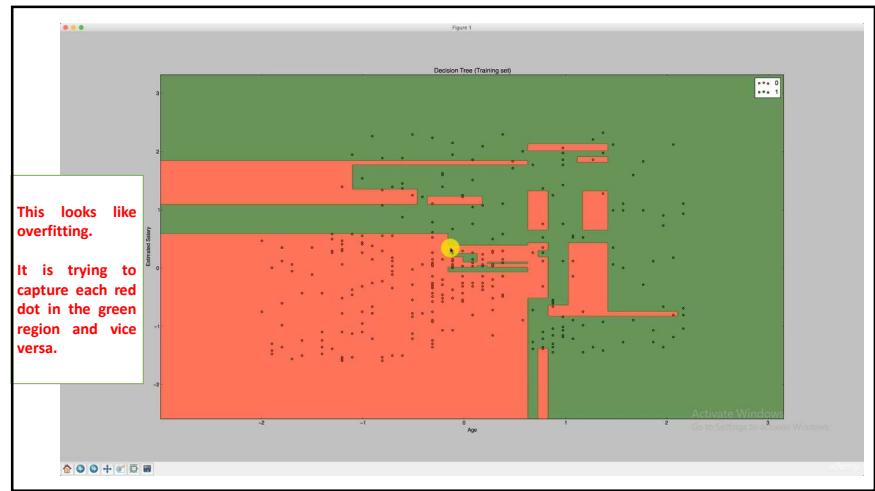
```

The screenshot shows a Jupyter Notebook interface with two panes. The left pane contains Python code for decision tree classification, including importing libraries, reading a dataset, splitting it into training and testing sets, scaling features, fitting a classifier, predicting test results, and visualizing the training set. The right pane shows the output of the code, including a variable explorer showing arrays for X_train, X_test, y_train, y_test, and classifier parameters, and a command-line history showing the execution of the code.

```

1 # Decision Tree Classification
2
3 # Importing the libraries
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import pandas as pd
7
8 # Importing the dataset
9 dataset = pd.read_csv('Social_Network_Ads.csv')
10 X = dataset.iloc[:, [2, 3]].values
11 y = dataset.iloc[:, 4].values
12
13 # Splitting the dataset into the Training set and Test set
14 from sklearn.cross_validation import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
16
17 # Feature Scaling
18 from sklearn.preprocessing import StandardScaler
19 sc = StandardScaler()
20 X_train = sc.fit_transform(X_train)
21 X_test = sc.transform(X_test)
22
23 # Fitting classifier to the training set
24 from sklearn.tree import DecisionTreeClassifier
25 classifier = DecisionTreeClassifier(criterion = 'entropy', max_depth = 3)
26 classifier.fit(X_train, y_train)
27
28 # Predicting the Test set results
29 y_pred = classifier.predict(X_test)
30
31 # Making the Confusion Matrix
32 from sklearn.metrics import confusion_matrix
33 cm = confusion_matrix(y_test, y_pred)
34
35 # Visualizing the Training set results
36 from matplotlib.colors import ListedColormap
37 X_set, y_set = X_train, y_train
38 X1, X2 = np.meshgrid(np.arange(X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.05),
39                      np.arange(X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.05))
40 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape[0], X1.shape[1]), alpha = 0.75, cmap = ListedColormap(['red', 'green']))
41

```



Random Forest Classifier

Random Forest Intuition

Ensemble Learning

In ensemble learning we build many models (different or same) and then make a collective decision in the end.

Alternatively, we may use same model, e.g. many decision tree models built on different subsets of data points, and take the decision made by the majority.

Random Forest Intuition

STEP 1: Pick at random K data points from the Training set.



STEP 2: Build the Decision Tree associated to these K data points.



STEP 3: Choose the number Ntree of trees you want to build and repeat STEPS 1 & 2



STEP 4: For a new data point, make each one of your Ntree trees predict the value of Y to for the data point in question, and assign the new data point the average across all of the predicted Y values.

Activate Windows

```

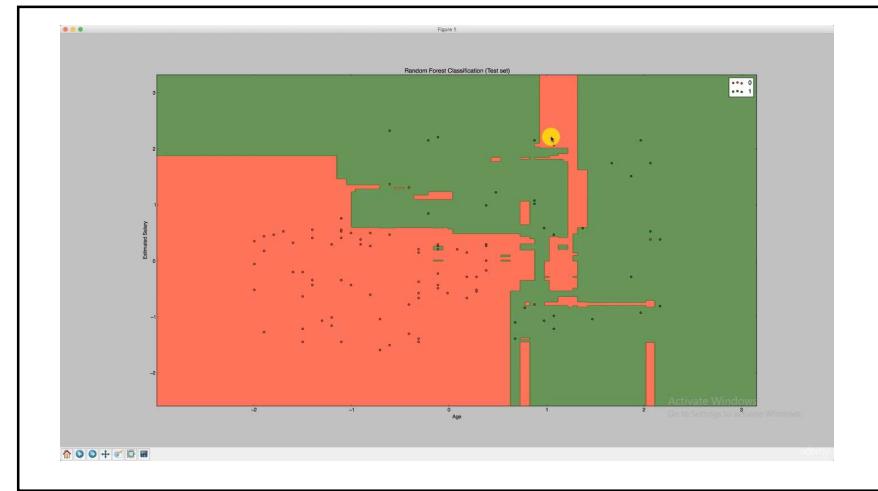
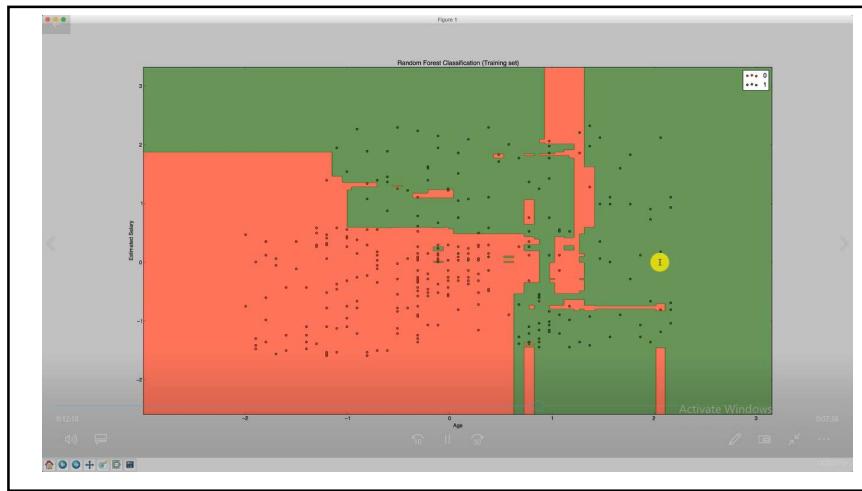
1 # Random Forest Classification
2
3 # Importing the Libraries
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import pandas as pd
7
8 # Importing the dataset
9 dataset = pd.read_csv('Social_Network_Ads.csv')
10 X = dataset.iloc[:, [2, 3]].values
11 y = dataset.iloc[:, 4].values
12
13 # Splitting the dataset into the Training set and Test set
14 from sklearn.cross_validation import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
16
17 # Feature Scaling
18 from sklearn.preprocessing import StandardScaler
19 sc = StandardScaler()
20 X_train = sc.fit_transform(X_train)
21 X_test = sc.transform(X_test)
22
23 # Fitting the Random Forest Classification to the Training set
24 from sklearn.ensemble import RandomForestClassifier
25 classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
26 classifier.fit(X_train, y_train)
27
28 # Predicting the Test set results
29 y_pred = classifier.predict(X_test)
30
31 # Making the Confusion Matrix
32 from sklearn.metrics import confusion_matrix
33 cm = confusion_matrix(y_test, y_pred) [1]
34
35 # Visualizing the Training set results
36 from matplotlib.colors import ListedColormap
37 X_set, y_set = X_train, y_train
38 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.6),
39 np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.6))
40 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape[0], X1.shape[1]), alpha = 0.75, cmap = ListedColormap(['red', 'green']))
41

```

```

1 # Random Forest Classification
2
3 # Importing the Libraries
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import pandas as pd
7
8 # Importing the dataset
9 dataset = pd.read_csv('Social_Network_Ads.csv')
10 X = dataset.iloc[:, [2, 3]].values
11 y = dataset.iloc[:, 4].values
12
13 # Splitting the dataset into the Training set and Test set
14 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
15
16 # Feature Scaling
17 from sklearn.preprocessing import StandardScaler
18 sc = StandardScaler()
19 X_train = sc.fit_transform(X_train)
20 X_test = sc.transform(X_test)
21
22 # Fitting the Random Forest Classification to the Training set
23 classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
24 classifier.fit(X_train, y_train)
25
26 # Predicting the Test set results
27 y_pred = classifier.predict(X_test)
28
29 # Making the Confusion Matrix
30 cm = confusion_matrix(y_test, y_pred)
31
32 # Visualizing the Training set results
33 X_set, y_set = X_train, y_train
34 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.6),
35 np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.6))
36 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape[0], X1.shape[1]), alpha = 0.75, cmap = ListedColormap(['red', 'green']))
37

```



Decision Tree/Random Forest Theory (Entropy and Information Gain)

DECISION TREE – ID3 ALGORITHM NUMERICAL EXAMPLE

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Find the attribute with maximum information gain.

Calculate information gain for each attribute.

Attribute with max information gain will be considered as root node of the tree.

For this, we need to calculate entropy of entire data set as well as for each attribute.

Entropy is measure of randomness or disorder
0 means fully random, 1 means no randomness

Day	Outlook	Temp	Humidity	Wind	Play Tennis	Attribute: Outlook
<i>Values (Outlook) = Sunny, Overcast, Rain</i>						
D1	Sunny	Hot	High	Weak	No	$S = [9+, 5-,]$
D2	Sunny	Hot	High	Strong	No	$Entropy(S) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.94$
D3	Overcast	Hot	High	Weak	Yes	$S_{Sunny} \leftarrow [2+, 3-,]$
D4	Rain	Mild	High	Weak	Yes	$Entropy(S_{Sunny}) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.971$
D5	Rain	Cool	Normal	Weak	Yes	$S_{Overcast} \leftarrow [4+, 0-,]$
D6	Rain	Cool	Normal	Strong	No	$Entropy(S_{Overcast}) = -\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} = 0$
D7	Overcast	Cool	Normal	Strong	Yes	$S_{Rain} \leftarrow [3+, 2-,]$
D8	Sunny	Mild	High	Weak	No	$Entropy(S_{Rain}) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.971$
D9	Sunny	Cool	Normal	Weak	Yes	
D10	Rain	Mild	Normal	Weak	Yes	
D11	Sunny	Mild	Normal	Strong	Yes	
D12	Overcast	Mild	High	Strong	Yes	
D13	Overcast	Hot	Normal	Weak	Yes	
D14	Rain	Mild	High	Strong	No	

$Entropy = -(+ve/total) \log (+ve/total) - (-ve/total) \log (-ve/total)$

Day	Outlook	Temp	Humidity	Wind	Play Tennis	Attribute: Outlook
<i>Values (Outlook) = Sunny, Overcast, Rain</i>						
D1	Sunny	Hot	High	Weak	No	$S = [9+, 5-,]$
D2	Sunny	Hot	High	Strong	No	$Entropy(S) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.94$
D3	Overcast	Hot	High	Weak	Yes	$S_{Sunny} \leftarrow [2+, 3-,]$
D4	Rain	Mild	High	Weak	Yes	$Entropy(S_{Sunny}) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.971$
D5	Rain	Cool	Normal	Weak	Yes	$S_{Overcast} \leftarrow [4+, 0-,]$
D6	Rain	Cool	Normal	Strong	No	$Entropy(S_{Overcast}) = -\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} = 0$
D7	Overcast	Cool	Normal	Strong	Yes	$S_{Rain} \leftarrow [3+, 2-,]$
D8	Sunny	Mild	High	Weak	No	$Entropy(S_{Rain}) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.971$
D9	Sunny	Cool	Normal	Weak	Yes	
D10	Rain	Mild	Normal	Weak	Yes	
D11	Sunny	Mild	Normal	Strong	Yes	
D12	Overcast	Mild	High	Strong	Yes	
D13	Overcast	Hot	Normal	Weak	Yes	
D14	Rain	Mild	High	Strong	No	

$Gain(S, Outlook) = Entropy(S) - \sum_{v \in \{Sunny, Overcast, Rain\}} \frac{|S_v|}{|S|} Entropy(S_v)$

$Gain(S, Outlook) = 0.94 - \frac{5}{14} 0.971 - \frac{4}{14} 0.971 - \frac{5}{14} 0.971 = 0.2464$

Day	Outlook	Temp	Humidity	Wind	Play Tennis	Attribute: Temp
<i>Values (Temp) = Hot, Mild, Cool</i>						
D1	Sunny	Hot	High	Weak	No	$S = [9+, 5-,]$
D2	Sunny	Hot	High	Strong	No	$Entropy(S) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.94$
D3	Overcast	Hot	High	Weak	Yes	$S_{Hot} \leftarrow [2+, 2-,]$
D4	Rain	Mild	High	Weak	Yes	$Entropy(S_{Hot}) = -\frac{2}{4} \log_2 \frac{2}{4} - \frac{2}{4} \log_2 \frac{2}{4} = 1.0$
D5	Rain	Cool	Normal	Weak	Yes	$S_{Mild} \leftarrow [4+, 2-,]$
D6	Rain	Cool	Normal	Strong	No	$Entropy(S_{Mild}) = -\frac{4}{6} \log_2 \frac{4}{6} - \frac{2}{6} \log_2 \frac{2}{6} = 0.9183$
D7	Overcast	Cool	Normal	Strong	Yes	$S_{Cool} \leftarrow [3+, 1-,]$
D8	Sunny	Mild	High	Weak	No	$Entropy(S_{Cool}) = -\frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4} = 0.8113$
D9	Sunny	Cool	Normal	Weak	Yes	
D10	Rain	Mild	Normal	Weak	Yes	
D11	Sunny	Mild	Normal	Strong	Yes	
D12	Overcast	Mild	High	Strong	Yes	
D13	Overcast	Hot	Normal	Weak	Yes	
D14	Rain	Mild	High	Strong	No	

$Gain(S, Temp) = Entropy(S) - \sum_{v \in \{Hot, Mild, Cool\}} \frac{|S_v|}{|S|} Entropy(S_v)$

$Gain(S, Temp) = 0.94 - \frac{4}{14} 1.0 - \frac{6}{14} 0.9183 - \frac{4}{14} 0.8113 = 0.0289$

Day	Outlook	Temp	Humidity	Wind	Play Tennis	Attribute: Humidity
<i>Values (Humidity) = High, Normal</i>						
D1	Sunny	Hot	High	Weak	No	$S = [9+, 5-,]$
D2	Sunny	Hot	High	Strong	No	$Entropy(S) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.94$
D3	Overcast	Hot	High	Weak	Yes	$S_{High} \leftarrow [3+, 4-,]$
D4	Rain	Mild	High	Weak	Yes	$Entropy(S_{High}) = -\frac{3}{7} \log_2 \frac{3}{7} - \frac{4}{7} \log_2 \frac{4}{7} = 0.9852$
D5	Rain	Cool	Normal	Weak	Yes	$S_{Normal} \leftarrow [6+, 1-,]$
D6	Rain	Cool	Normal	Strong	No	$Entropy(S_{Normal}) = -\frac{6}{7} \log_2 \frac{6}{7} - \frac{1}{7} \log_2 \frac{1}{7} = 0.5916$
D7	Overcast	Cool	Normal	Strong	Yes	
D8	Sunny	Mild	High	Weak	No	
D9	Sunny	Cool	Normal	Weak	Yes	
D10	Rain	Mild	Normal	Weak	Yes	
D11	Sunny	Mild	Normal	Strong	Yes	
D12	Overcast	Mild	High	Strong	Yes	
D13	Overcast	Hot	Normal	Weak	Yes	
D14	Rain	Mild	High	Strong	No	

$Gain(S, Humidity) = Entropy(S) - \sum_{v \in \{High, Normal\}} \frac{|S_v|}{|S|} Entropy(S_v)$

$Gain(S, Humidity) = 0.94 - \frac{7}{14} 0.9852 - \frac{7}{14} 0.5916 = 0.1516$

Day	Outlook	Temp	Humidity	Wind	Play Tennis	Attribute: Wind
D1	Sunny	Hot	High	Weak	No	Values (Wind) = Strong, Weak
D2	Sunny	Hot	High	Strong	No	
D3	Overcast	Hot	High	Weak	Yes	
D4	Rain	Mild	High	Weak	Yes	
D5	Rain	Cool	Normal	Weak	Yes	
D6	Rain	Cool	Normal	Strong	No	
D7	Overcast	Cool	Normal	Strong	Yes	
D8	Sunny	Mild	High	Weak	No	
D9	Sunny	Cool	Normal	Weak	Yes	
D10	Rain	Mild	Normal	Weak	Yes	
D11	Sunny	Mild	Normal	Strong	Yes	
D12	Overcast	Mild	High	Strong	Yes	
D13	Overcast	Hot	Normal	Weak	Yes	
D14	Rain	Mild	High	Strong	No	

$$S = [9+, 5 -]$$

$$\text{Entropy}(S) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.94$$

$$S_{Strong} \leftarrow [3+, 3 -] \quad \text{Entropy}(S_{Strong}) = 1.0$$

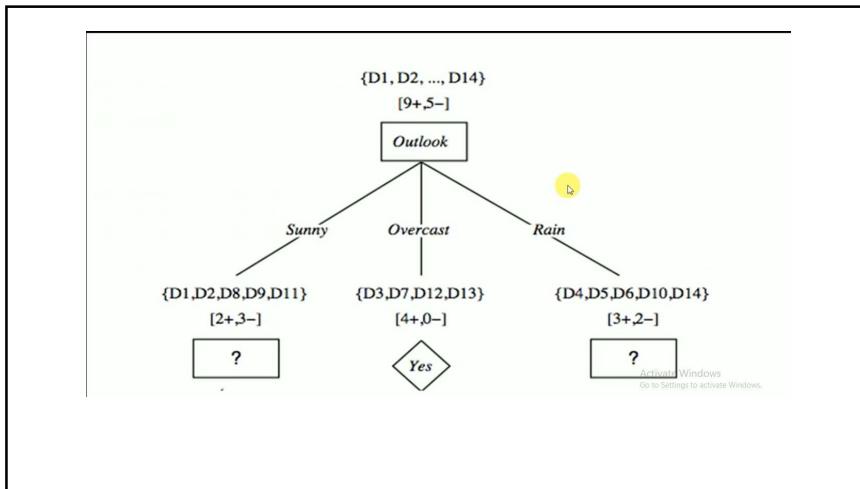
$$S_{Weak} \leftarrow [6+, 2 -] \quad \text{Entropy}(S_{Weak}) = -\frac{6}{8} \log_2 \frac{6}{8} - \frac{2}{8} \log_2 \frac{2}{8} = 0.8113$$

$$\text{Gain}(S, \text{Wind}) = \text{Entropy}(S) - \sum_{v \in \{\text{Strong}, \text{Weak}\}} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

$$\text{Gain}(S, \text{Wind}) = \text{Entropy}(S) - \frac{6}{14} \text{Entropy}(S_{Strong}) - \frac{8}{14} \text{Entropy}(S_{Weak})$$

$$\text{Gain}(S, \text{Wind}) = 0.94 - \frac{6}{14} \cdot 1.0 - \frac{8}{14} \cdot 0.8113 = 0.0478$$

Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



Attribute: Temp

Values (Temp) = Hot, Mild, Cool

$$S_{Sunny} = [2+, 3 -]$$

$$\text{Entropy}(S_{Sunny}) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.97$$

$$S_{Hot} \leftarrow [0+, 2 -]$$

$$\text{Entropy}(S_{Hot}) = 0.0$$

$$S_{Mild} \leftarrow [1+, 1 -]$$

$$\text{Entropy}(S_{Mild}) = 1.0$$

$$S_{Cool} \leftarrow [1+, 0 -]$$

$$\text{Entropy}(S_{Cool}) = 0.0$$

$$\text{Gain}(S_{Sunny}, \text{Temp}) = \text{Entropy}(S) - \sum_{v \in \{\text{Hot}, \text{Mild}, \text{Cool}\}} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

$$= \text{Entropy}(S) - \frac{2}{5} \text{Entropy}(S_{Hot}) - \frac{2}{5} \text{Entropy}(S_{Mild})$$

$$- \frac{1}{5} \text{Entropy}(S_{Cool})$$

$$\text{Gain}(S_{Sunny}, \text{Temp}) = 0.97 - \frac{2}{5} \cdot 0.0 - \frac{2}{5} \cdot 1 - \frac{1}{5} \cdot 0.0 = 0.570$$

Go to settings to activate Windows.

Day	Temp	Humidity	Wind	Play Tennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

Attribute: Humidity

Values (Humidity) = High, Normal

$$S_{Sunny} = [2+, 3-] \quad Entropy(S) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.97$$

$$S_{High} \leftarrow [0+, 3-] \quad Entropy(S_{High}) = 0.0$$

$$S_{Normal} \leftarrow [2+, 0-] \quad Entropy(S_{Normal}) = 0.0$$

$$Gain(S_{Sunny}, \text{Humidity}) = Entropy(S) - \sum_{v \in \{\text{High, Normal}\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Sunny}, \text{Humidity}) = Entropy(S) - \frac{3}{5} Entropy(S_{High}) - \frac{2}{5} Entropy(S_{Normal})$$

$$Gain(S_{Sunny}, \text{Humidity}) = 0.97 - \frac{3}{5} 0.0 - \frac{2}{5} 0.0 = 0.97$$

Day	Temp	Humidity	Wind	Play Tennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

Attribute: Wind

Values (Wind) = Strong, Weak

$$S_{Sunny} = [2+, 3-] \quad Entropy(S) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.97$$

$$S_{Strong} \leftarrow [1+, 1-] \quad Entropy(S_{Strong}) = 1.0$$

$$S_{Weak} \leftarrow [1+, 2-] \quad Entropy(S_{Weak}) = -\frac{1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} = 0.9183$$

$$Gain(S_{Sunny}, \text{Wind}) = Entropy(S) - \sum_{v \in \{\text{Strong, Weak}\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Sunny}, \text{Wind}) = Entropy(S) - \frac{2}{5} Entropy(S_{Strong}) - \frac{3}{5} Entropy(S_{Weak})$$

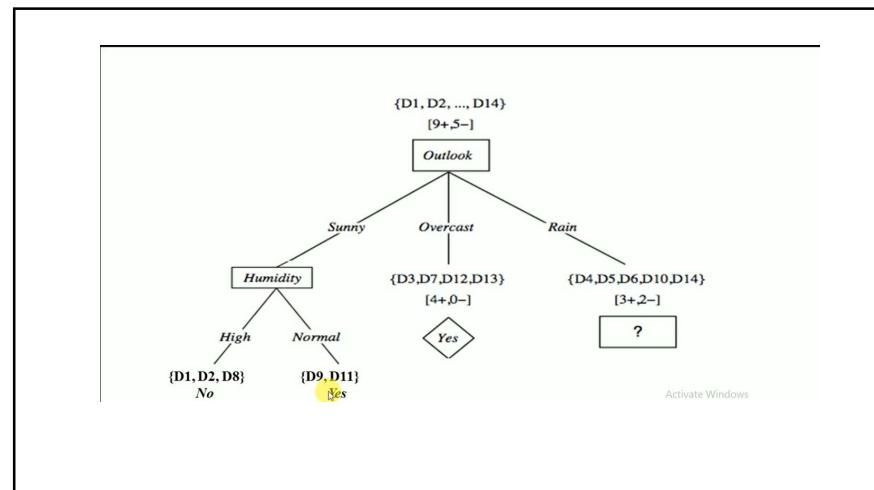
$$Gain(S_{Sunny}, \text{Wind}) = 0.97 - \frac{2}{5} 1.0 - \frac{3}{5} 0.918 = 0.0192$$

Day	Temp	Humidity	Wind	Play Tennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

$Gain(S_{Sunny}, \text{Temp}) = 0.570$

$Gain(S_{Sunny}, \text{Humidity}) = 0.97$

$Gain(S_{Sunny}, \text{Wind}) = 0.0192$



Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
D10	Mild	Normal	Weak	Yes
D14	Mild	High	Strong	No

Attribute: Temp

Values (Temp) = Hot, Mild, Cool

$S_{Rain} = [3+, 2-]$ $Entropy(S_{Sunny}) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.97$

$S_{Hot} \leftarrow [0+, 0-]$ $Entropy(S_{Hot}) = 0.0$

$S_{Mild} \leftarrow [2+, 1-]$ $Entropy(S_{Mild}) = -\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3} = 0.9183$

$S_{Cool} \leftarrow [1+, 1-]$ $Entropy(S_{Cool}) = 1.0$

$Gain(S_{Rain}, Temp) = Entropy(S) - \sum_{v \in \{Hot, Mild, Cool\}} \frac{|S_v|}{|S|} Entropy(S_v)$

$= Entropy(S) - \frac{0}{5} Entropy(S_{Hot}) - \frac{3}{5} Entropy(S_{Mild})$

$- \frac{2}{5} Entropy(S_{Cool})$

$Gain(S_{Rain}, Temp) = 0.97 - \frac{0}{5} 0.0 - \frac{3}{5} 0.9183 - \frac{2}{5} 1.0 = 0.0192$

Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
D10	Mild	Normal	Weak	Yes
D14	Mild	High	Strong	No

Attribute: Humidity

Values (Humidity) = High, Normal

$S_{Rain} = [3+, 2-]$ $Entropy(S_{Sunny}) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.97$

$S_{High} \leftarrow [1+, 1-]$ $Entropy(S_{High}) = 1.0$

$S_{Normal} \leftarrow [2+, 1-]$ $Entropy(S_{Normal}) = -\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3} = 0.9183$

$Gain(S_{Rain}, Humidity) = Entropy(S) - \sum_{v \in \{High, Normal\}} \frac{|S_v|}{|S|} Entropy(S_v)$

$= Entropy(S) - \frac{2}{5} Entropy(S_{High}) - \frac{3}{5} Entropy(S_{Normal})$

$Gain(S_{Rain}, Humidity) = 0.97 - \frac{2}{5} 1.0 - \frac{3}{5} 0.9183 = 0.0192$

Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
D10	Mild	Normal	Weak	Yes
D14	Mild	High	Strong	No

Attribute: Wind

Values (wind) = Strong, Weak

$S_{Rain} = [3+, 2-]$ $Entropy(S_{Sunny}) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.97$

$S_{Strong} \leftarrow [0+, 2-]$ $Entropy(S_{Strong}) = 0.0$

$S_{Weak} \leftarrow [3+, 0-]$ $Entropy(S_{Weak}) = 0.0$

$Gain(S_{Rain}, Wind) = Entropy(S) - \sum_{v \in \{Strong, Weak\}} \frac{|S_v|}{|S|} Entropy(S_v)$

$= Entropy(S) - \frac{2}{5} Entropy(S_{Strong}) - \frac{3}{5} Entropy(S_{Weak})$

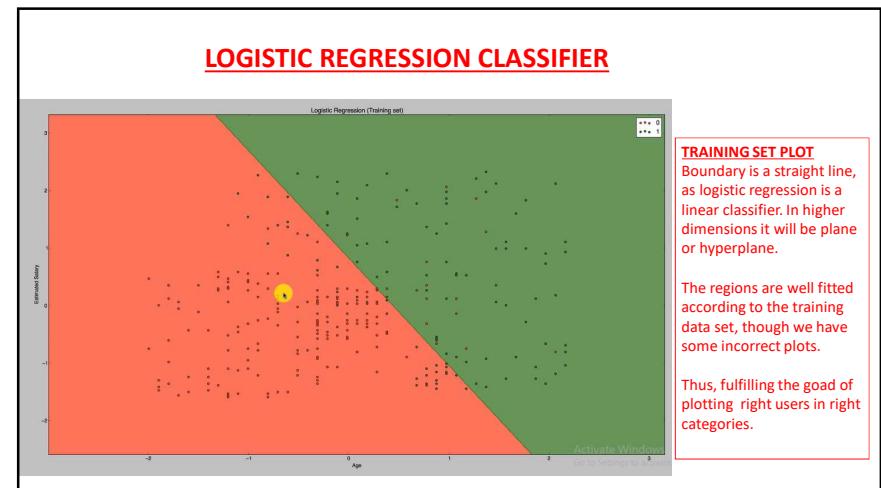
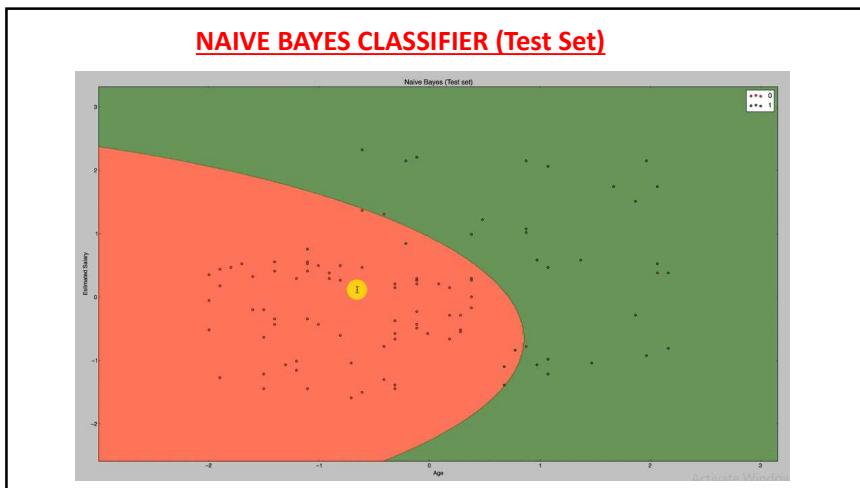
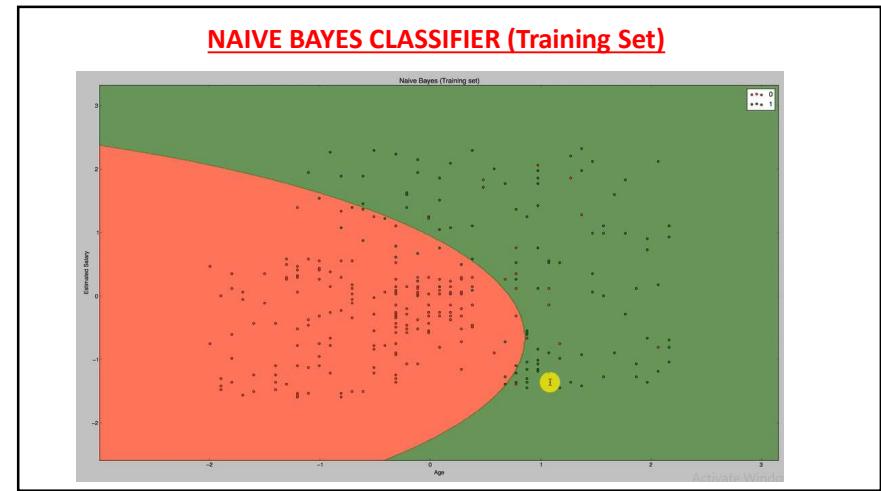
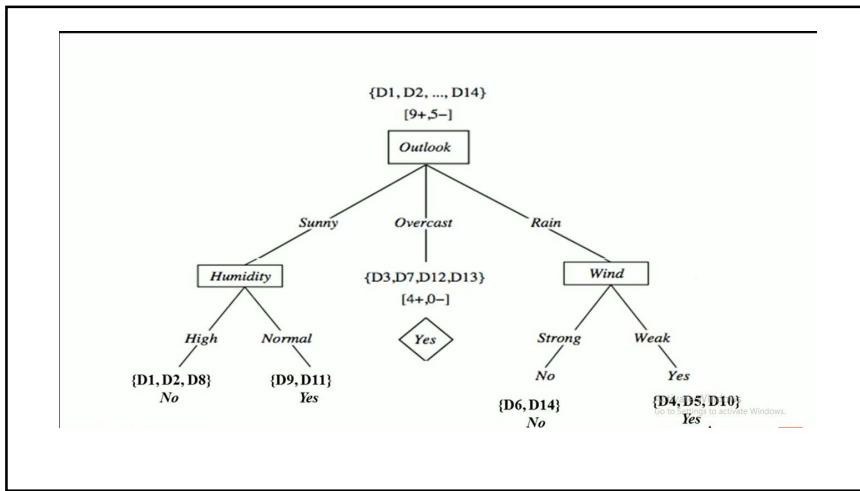
$Gain(S_{Rain}, Wind) = 0.97 - \frac{2}{5} 0.0 - \frac{3}{5} 0.0 = 0.97$

Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
D10	Mild	Normal	Weak	Yes
D14	Mild	High	Strong	No

$Gain(S_{Rain}, Temp) = 0.0192$

$Gain(S_{Rain}, Humidity) = 0.0192$

$Gain(S_{Rain}, Wind) = 0.97$



LOGISTIC REGRESSION CLASSIFIER

