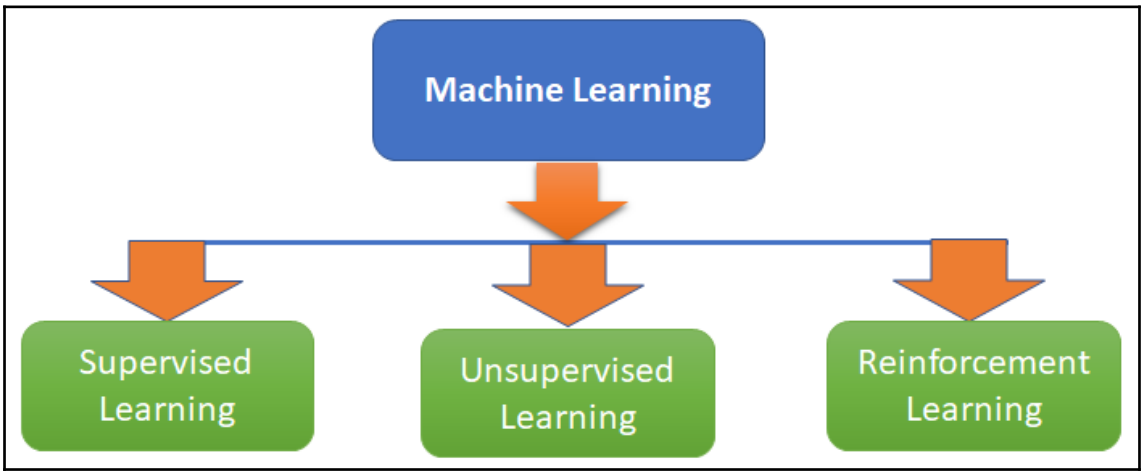
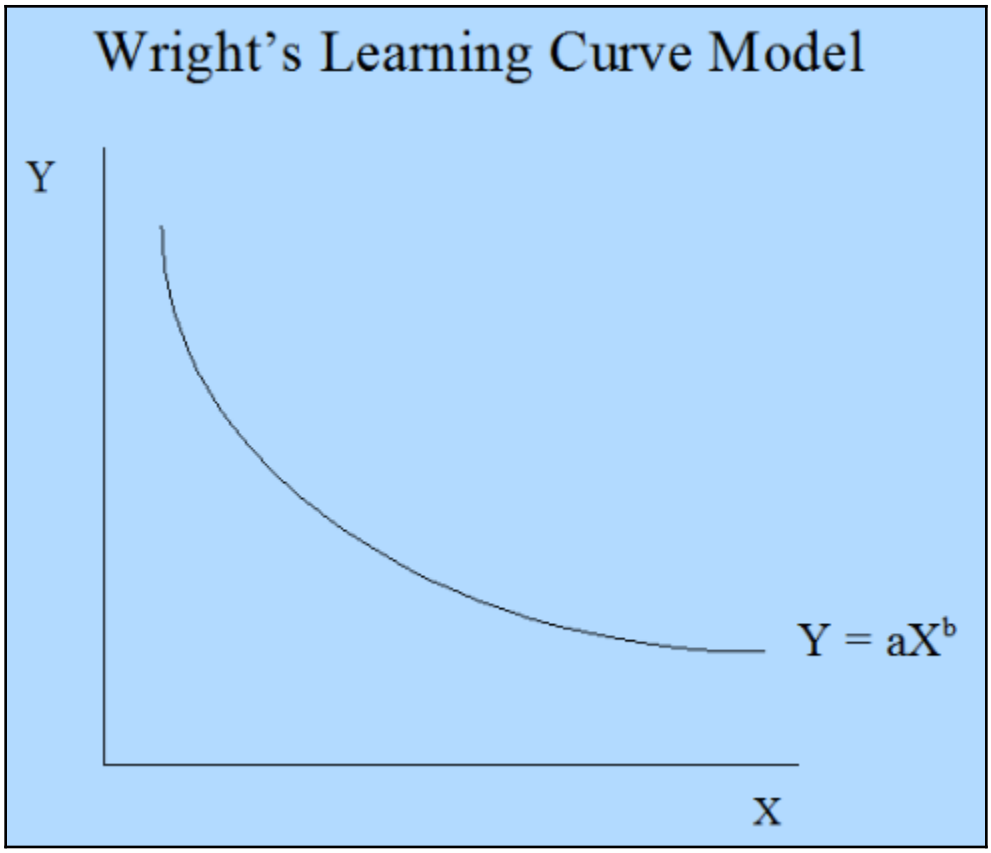
Table of Contents

1

Index 122

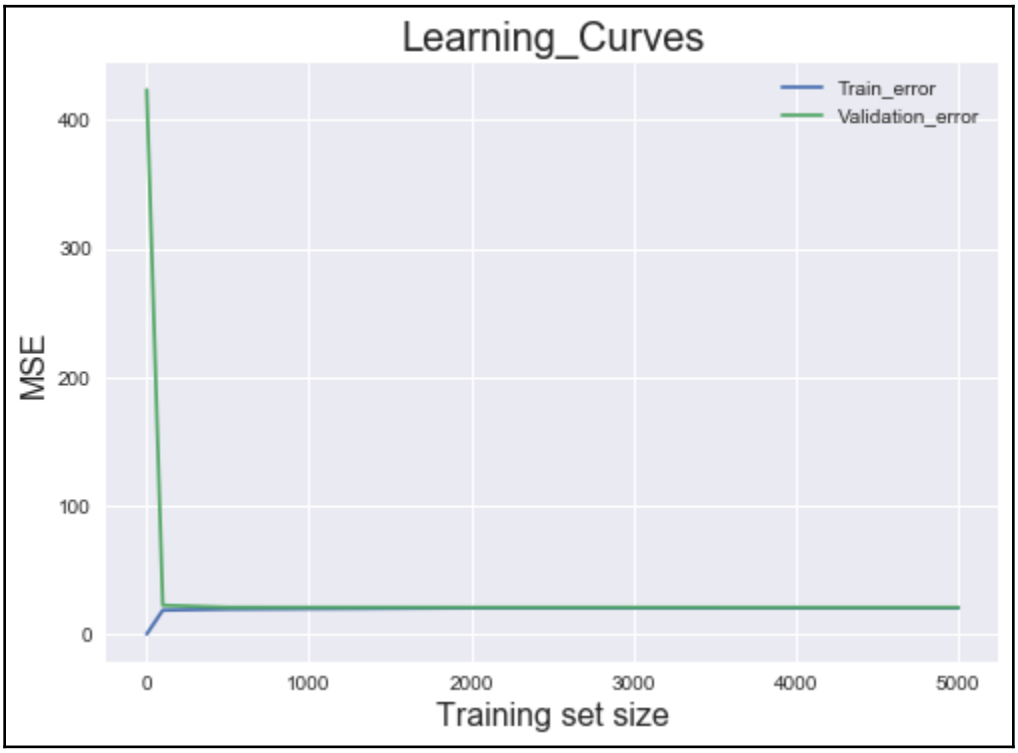
Chapter 1: Quantifying Learning Algorithms

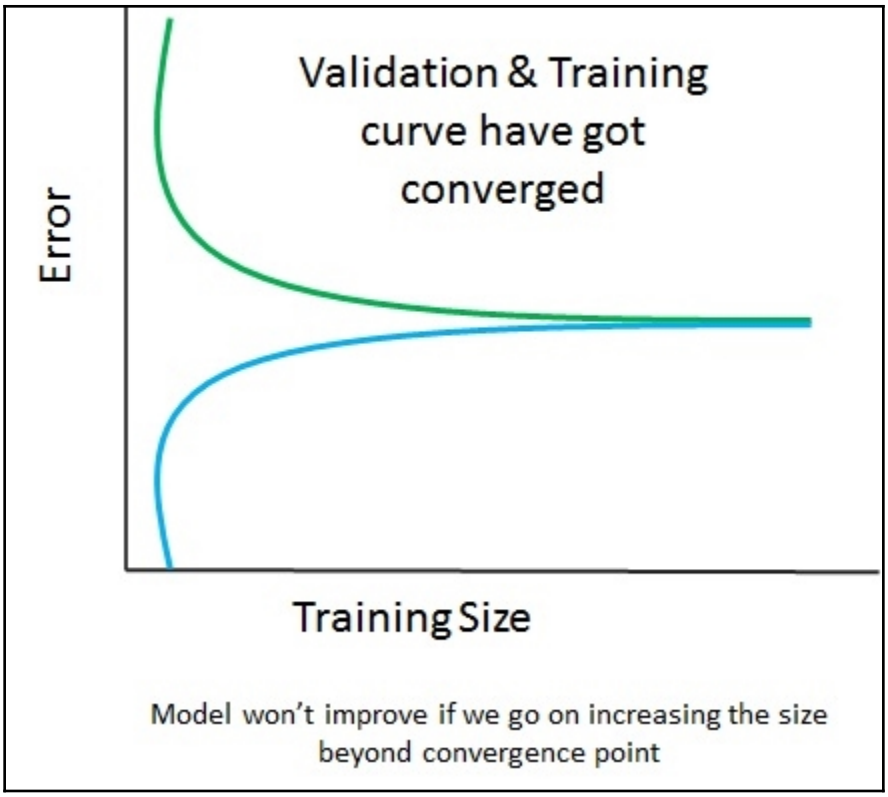


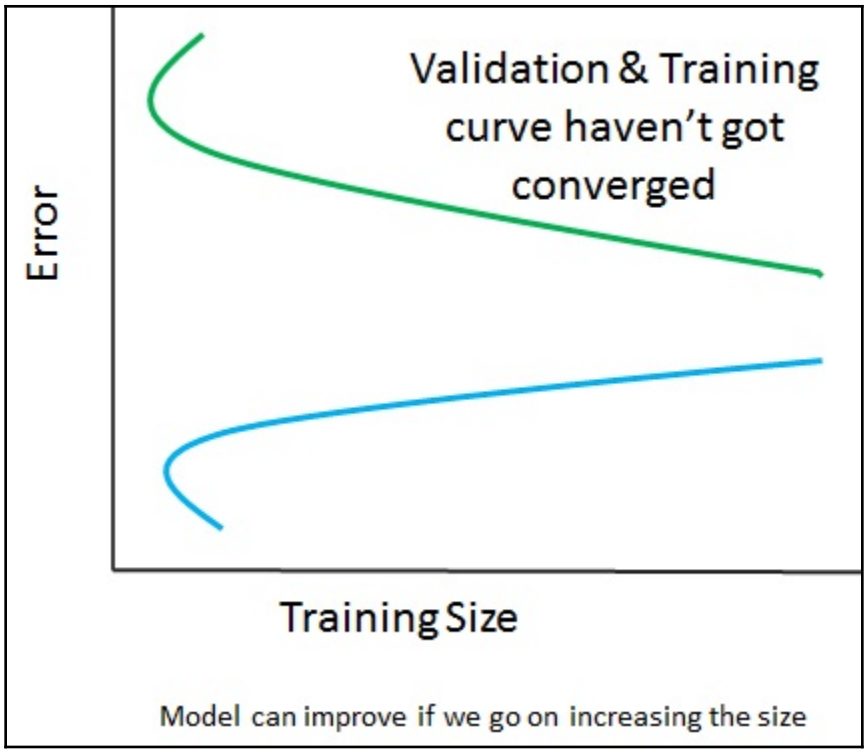


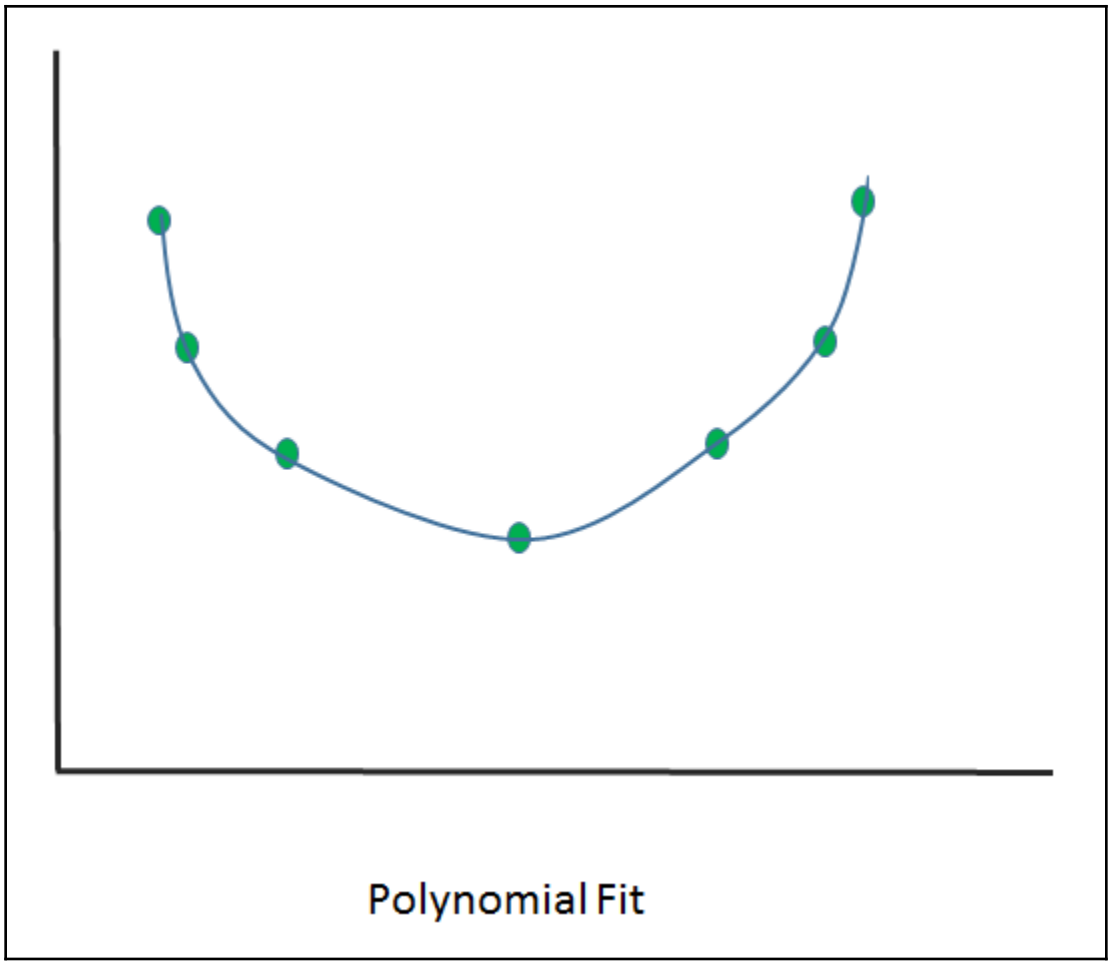

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9568 entries, 0 to 9567
Data columns (total 5 columns):
AT    9568 non-null float64
V     9568 non-null float64
AP    9568 non-null float64
RH    9568 non-null float64
PE    9568 non-null float64
dtypes: float64(5)
memory usage: 373.8 KB
None
```

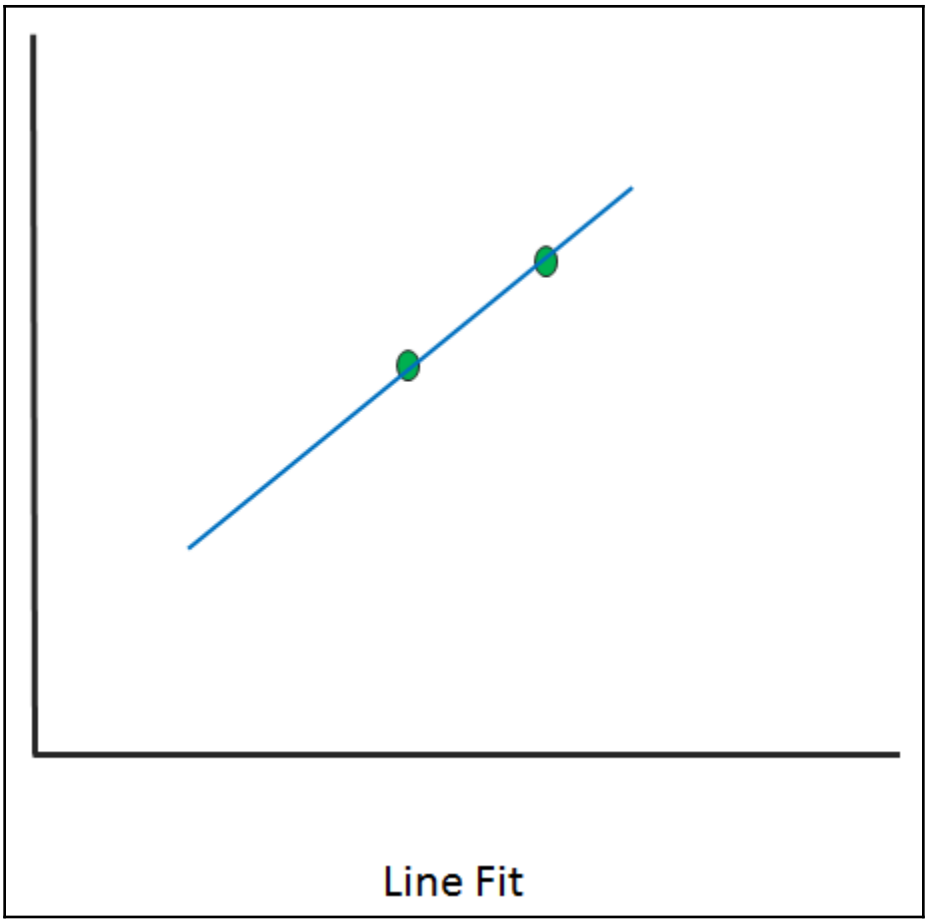
	AT	V	AP	RH	PE
0	14.96	41.76	1024.07	73.17	463.26
1	25.18	62.96	1020.04	59.08	444.37
2	5.11	39.40	1012.16	92.14	488.56
3	20.86	57.32	1010.24	76.64	446.48
4	10.82	37.50	1009.23	96.62	473.90

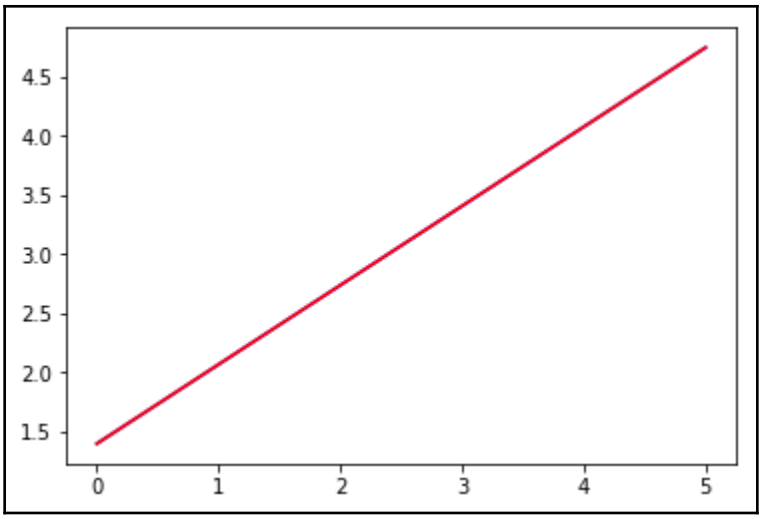


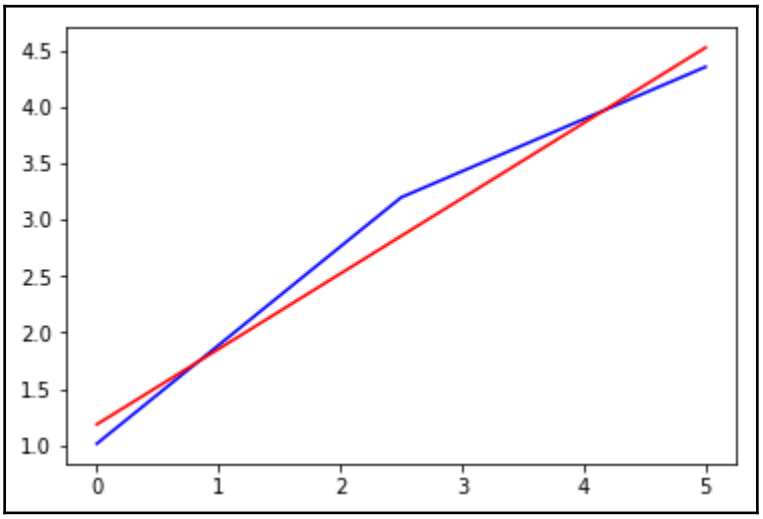

Validation & Training
curve have got
converged
Error
Training Size
Model won't improve if we go on increasing the size
beyond convergence point

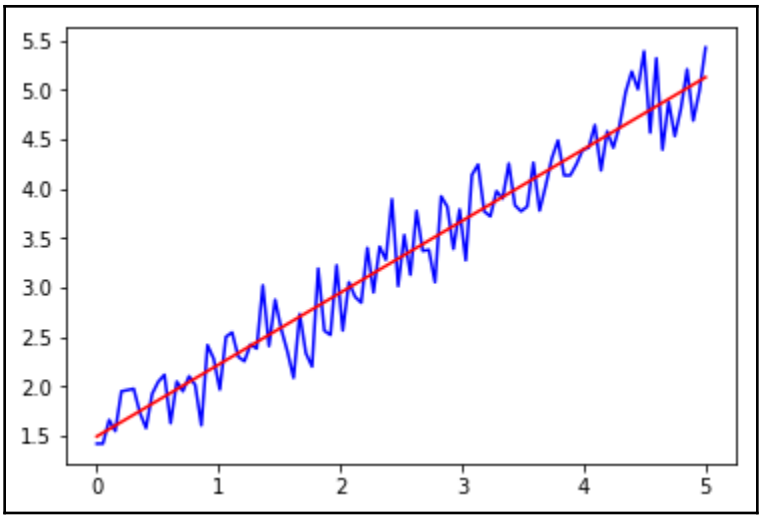

Validation & Training curve haven't got converged
Error
Training Size
Model can improve if we go on increasing the size

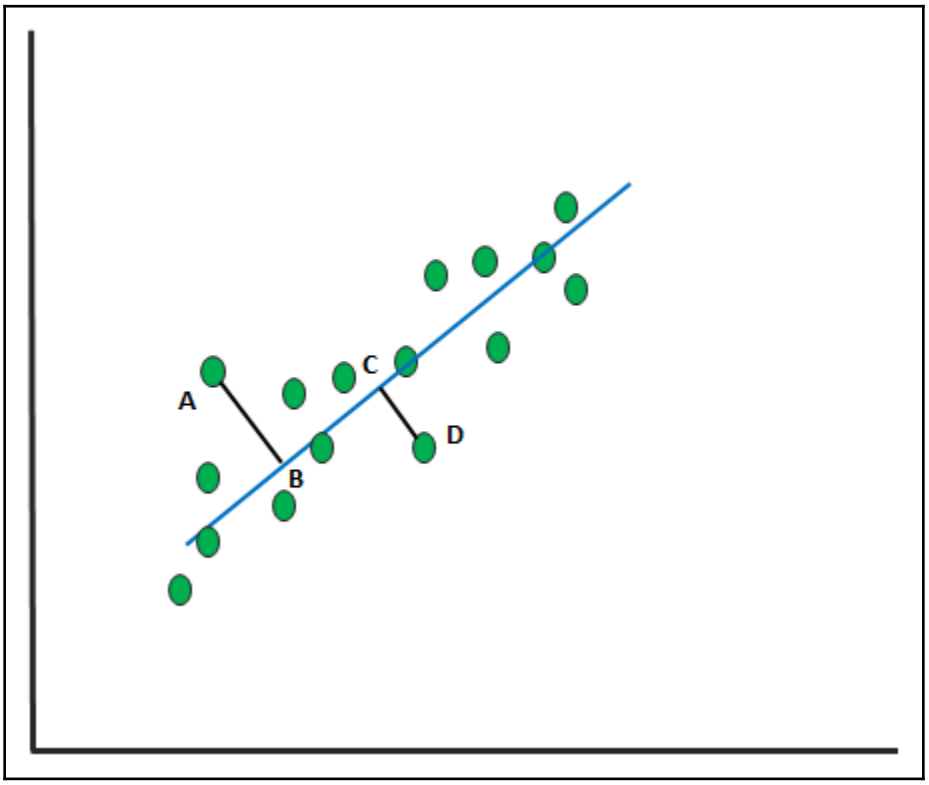

Polynomial Fit

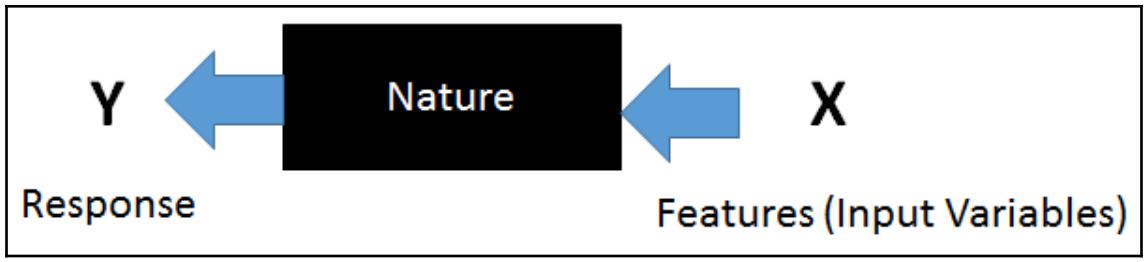

Line Fit


4.5
4.0
3.5
3.0
2.5
2.0
1.5
0
1
2
3
4
5


4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
1
2
3
4
5


5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
0
1
2
3
4
5

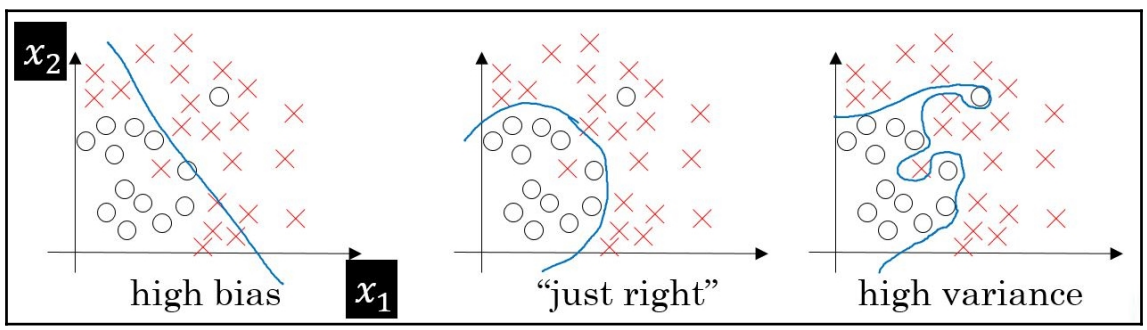

A
B
C
D

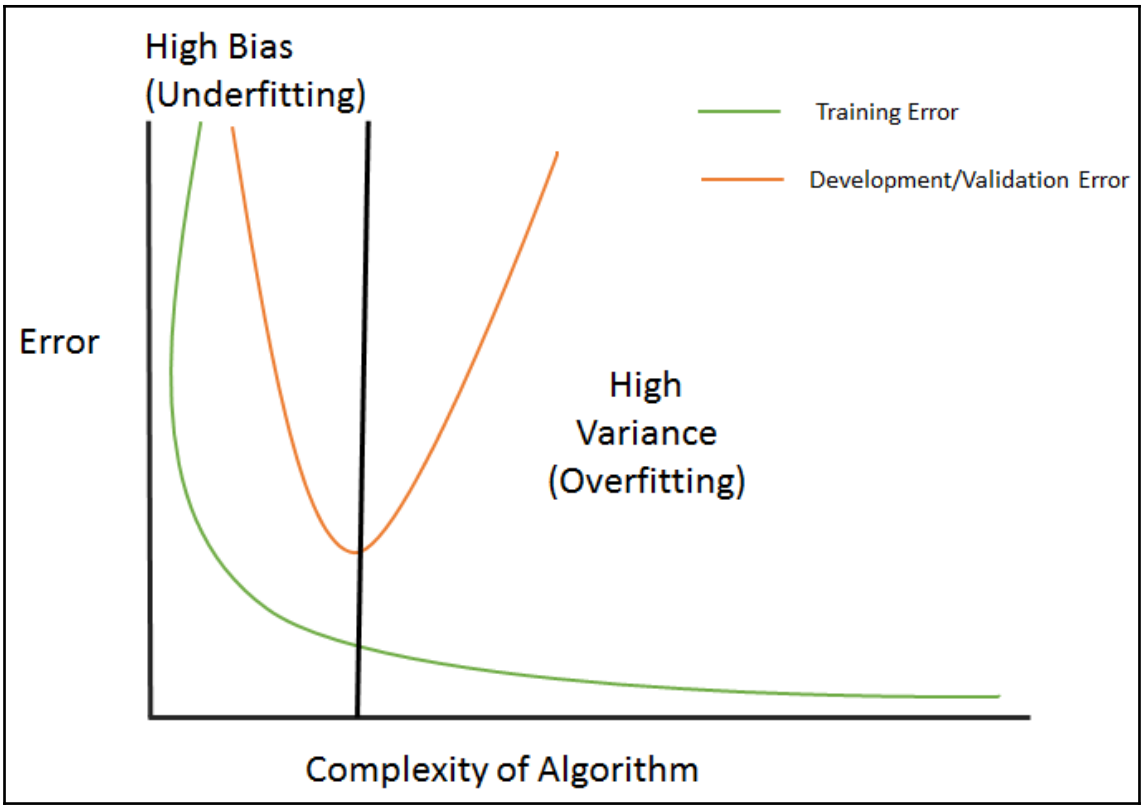


Training Set	Development Set	Test Set

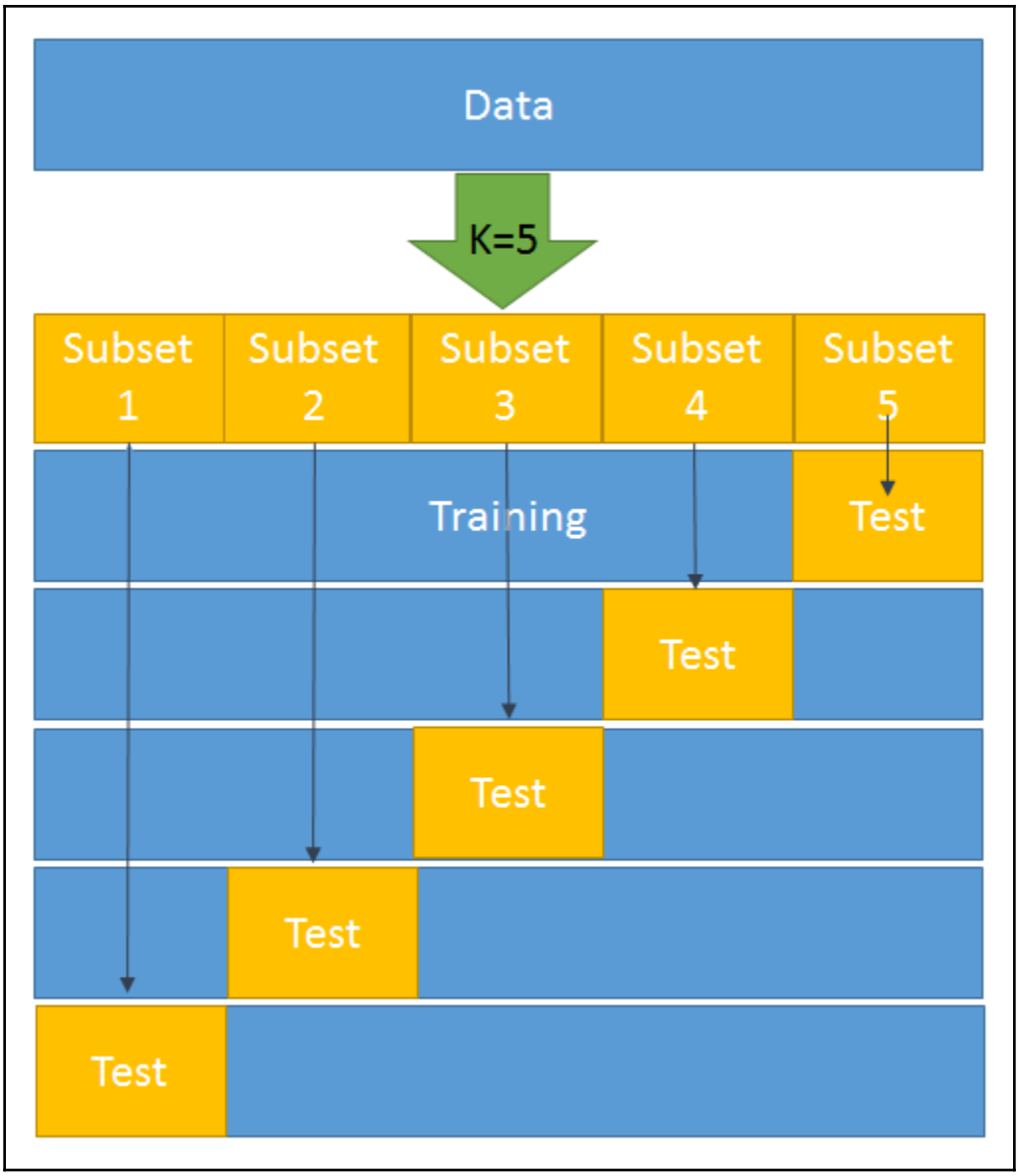
Training Set	Development Set	Test Set
60%	20%	20%

Training Set	Development Set	Test Set
70%	15%	15%





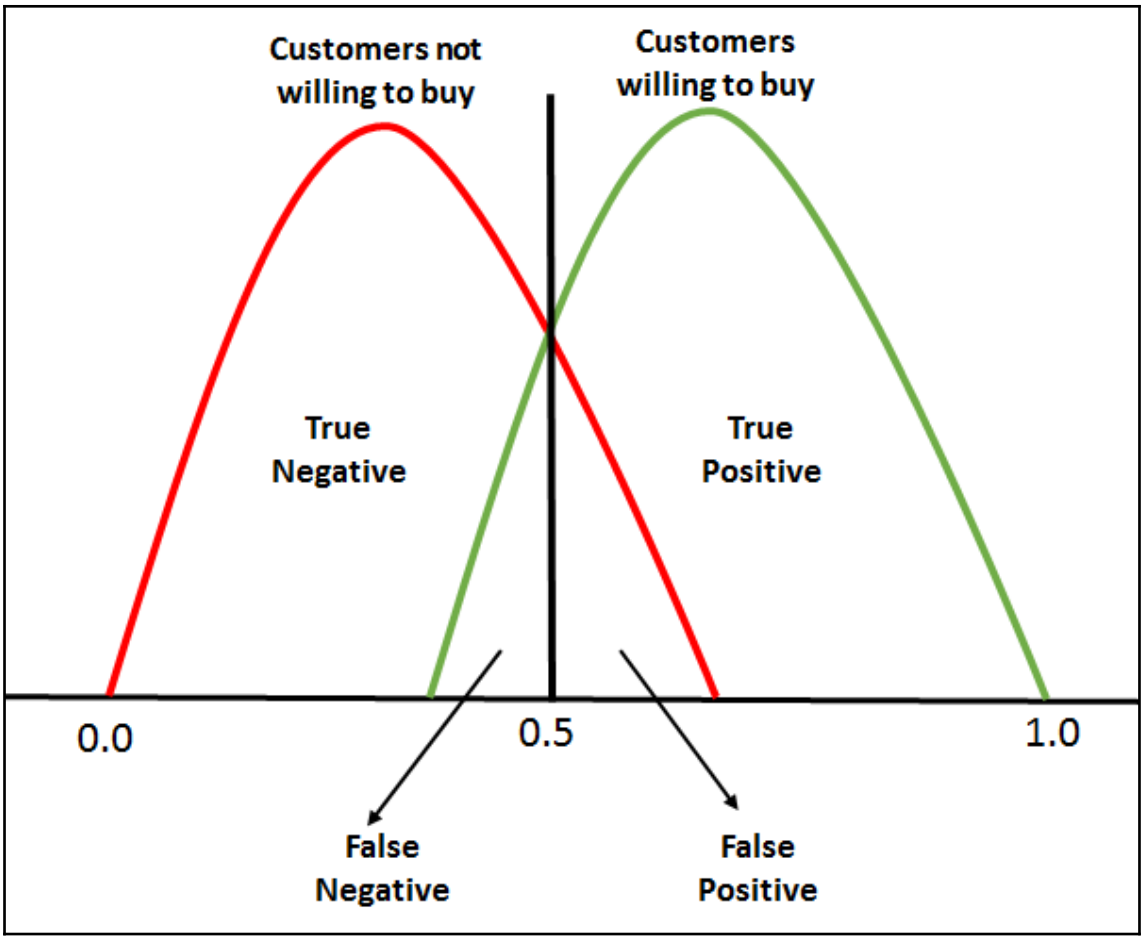
Training Error	1%	16%	16%	0.8%
Development Error	10%	17%	30%	1%
	High Variance	High Bias	High Bias High Variance	Low Bias Low Variance

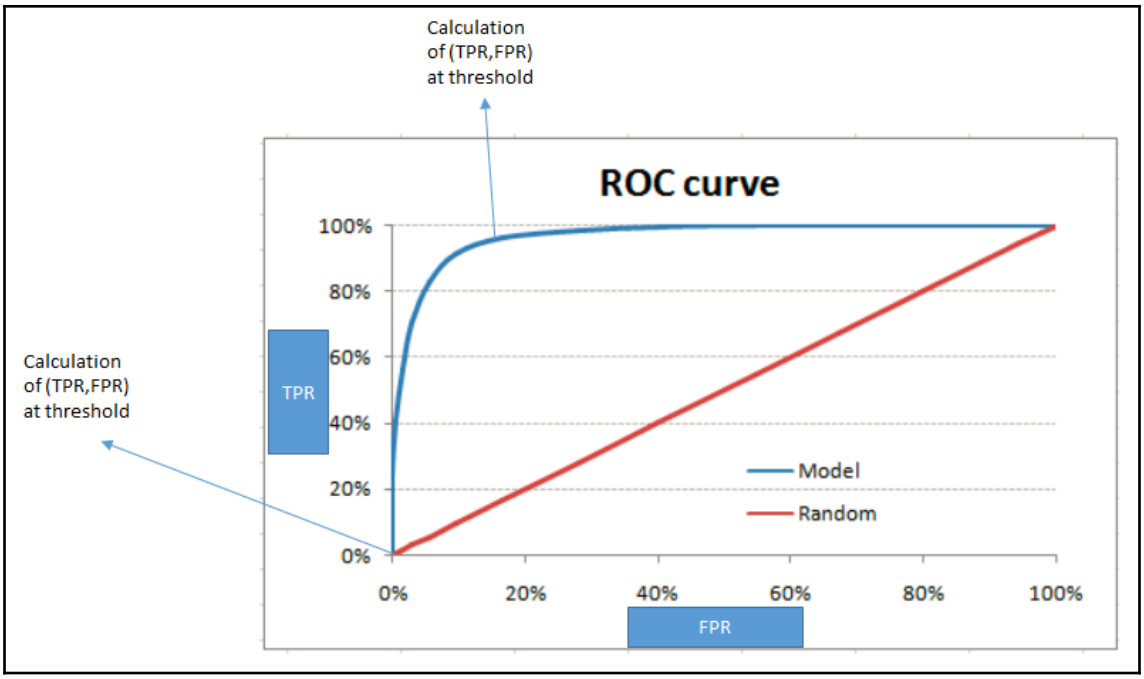

Data
K=5
Subset 1
Subset 2
Subset 3
Subset 4
Subset 5
Training
Test
Test
Test
Test
Test

N=80	Actual:Yes	Actual:No
Predicted: Yes	50	6
Predicted: No	4	20

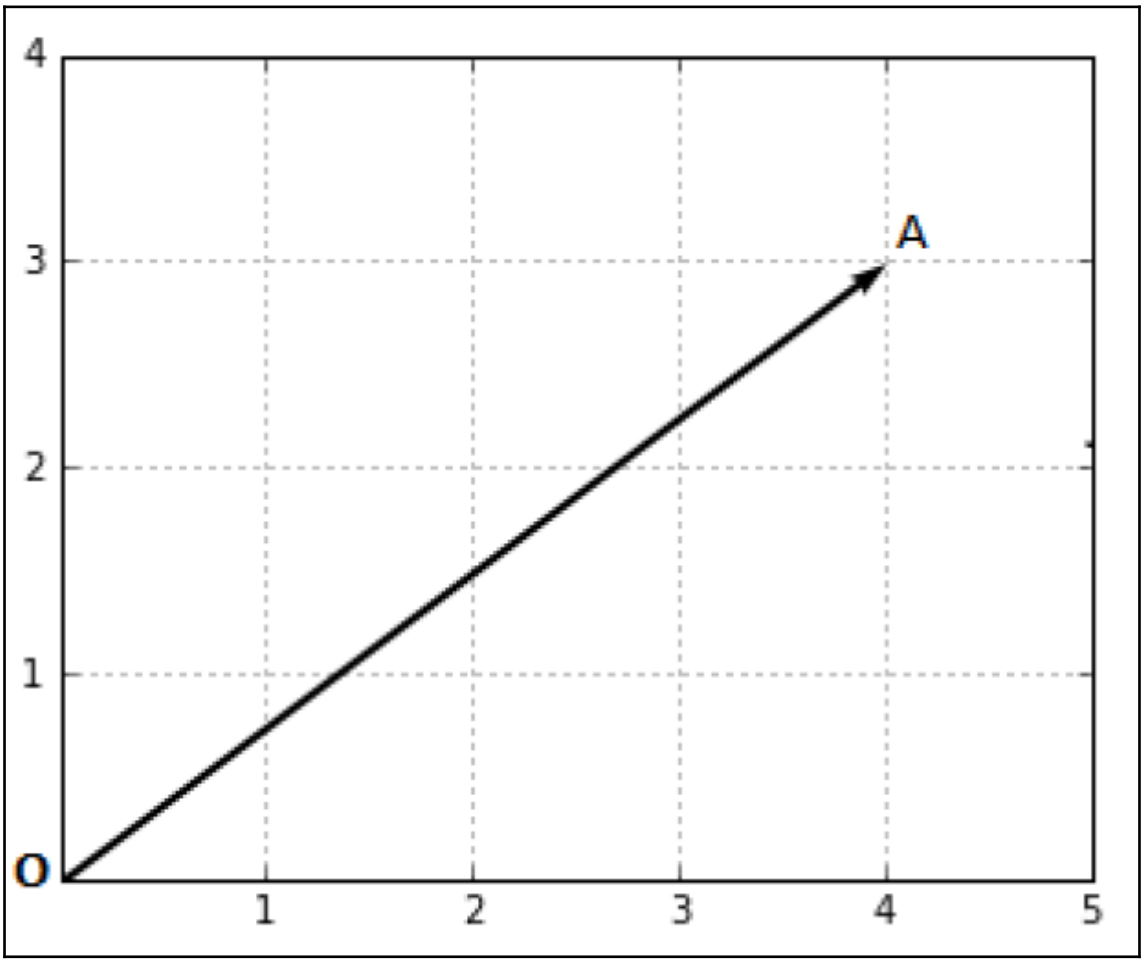
N=80	Actual:Yes	Actual:No	
Predicted: Yes	TP= 50	FP=6	56
Predicted: No	FN=4	TN= 20	24
	54	26	

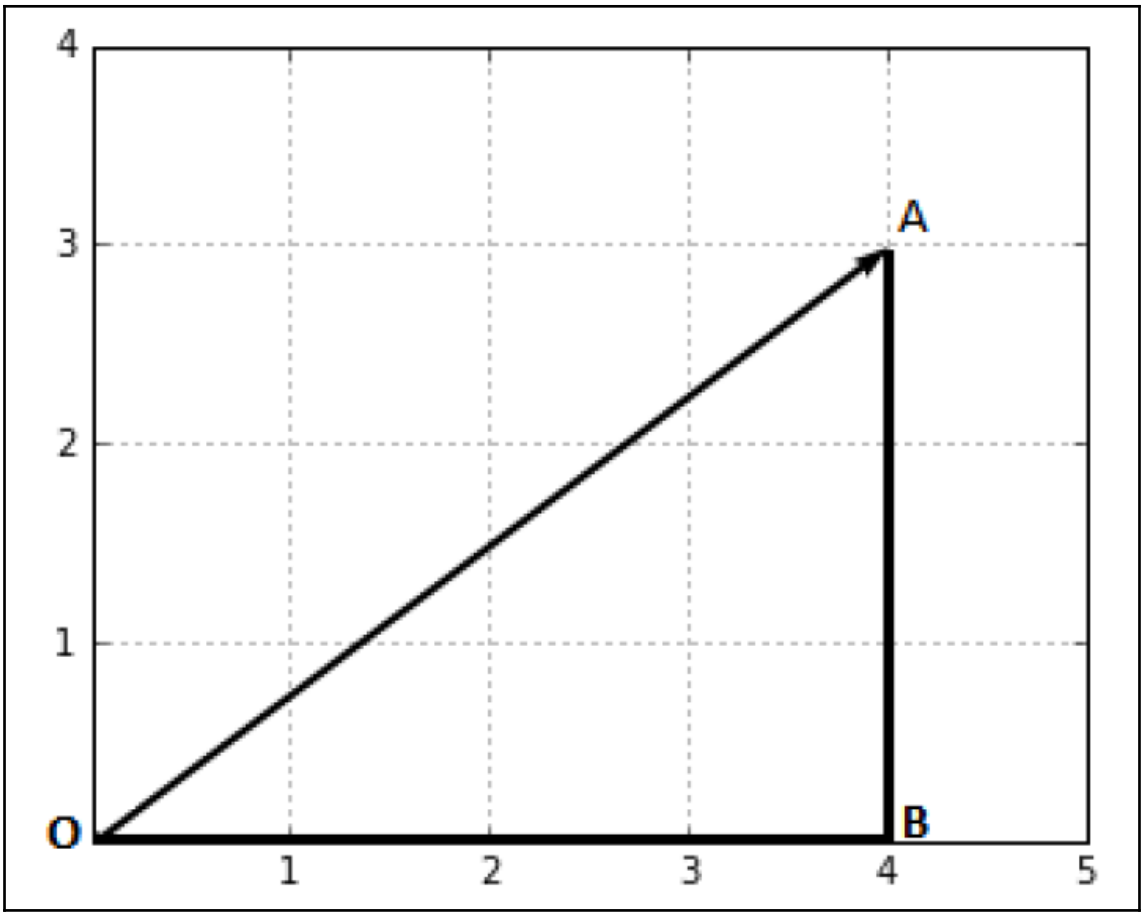
		True condition				
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$	F_1 score = $\frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$		


Customers not willing to buy
Customers willing to buy
True Negative
True Positive
0.0
0.5
1.0
False Negative
False Positive

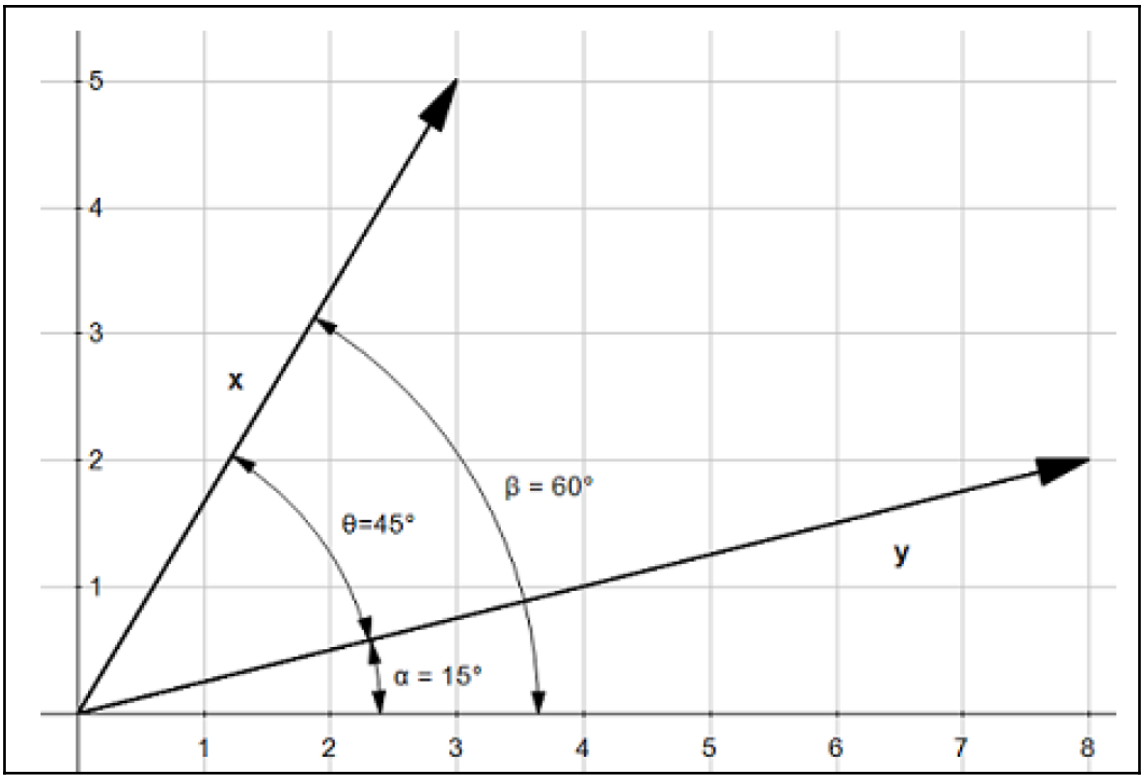

Calculation
of (TPR,FPR)
at threshold
ROC curve
100%
80%
60%
40%
20%
0%
TPR
Model
Random
0%
20%
40%
60%
80%
100%
FPR
Calculation
of (TPR,FPR)
at threshold

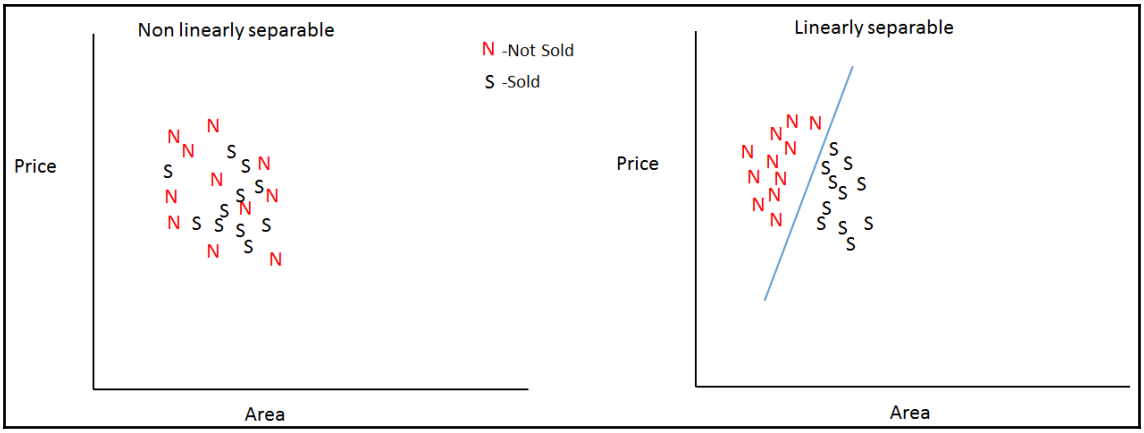
Chapter 2: Evaluating Kernel Learning

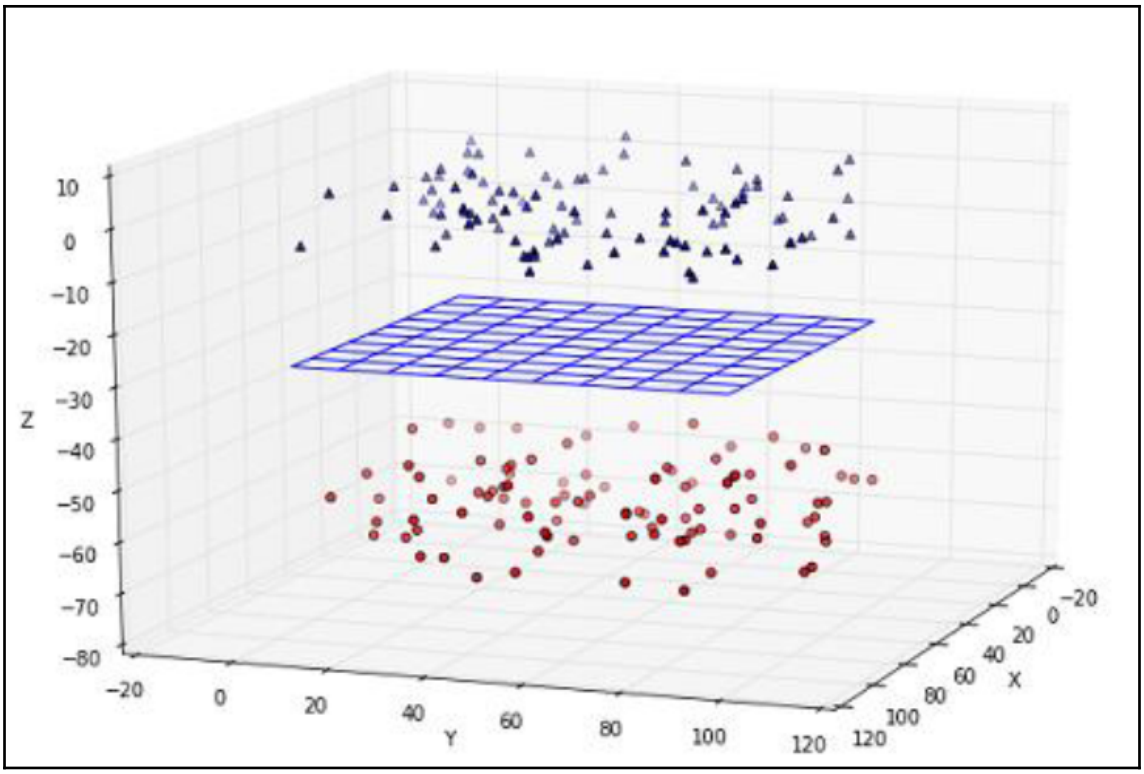


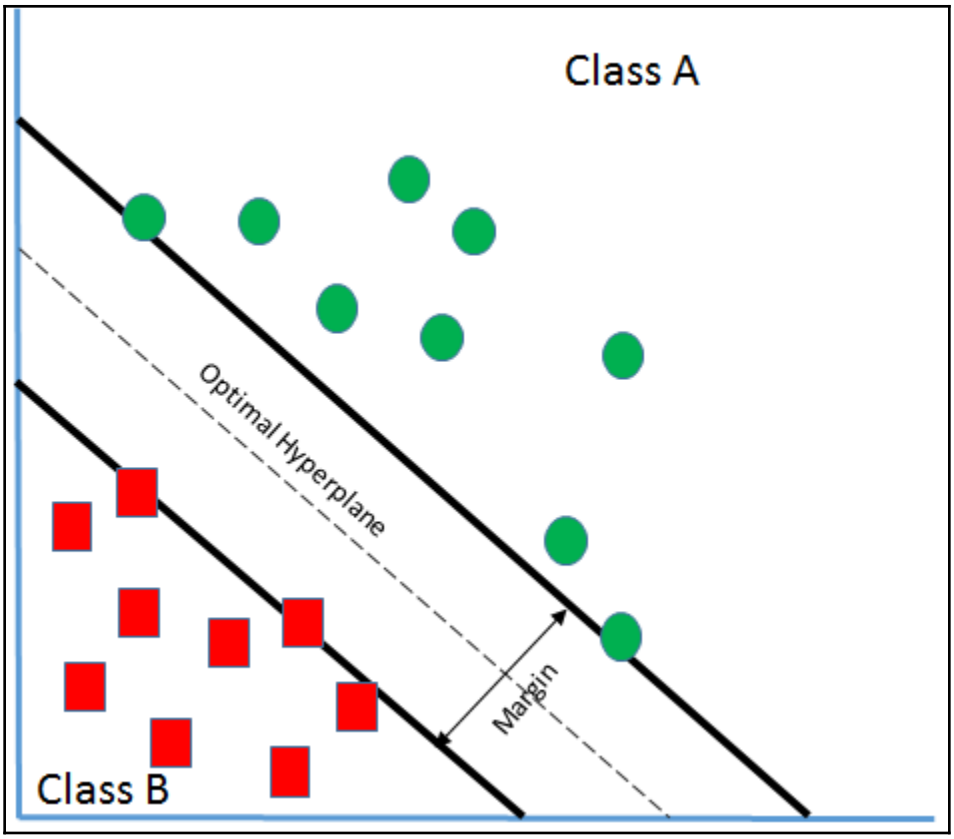


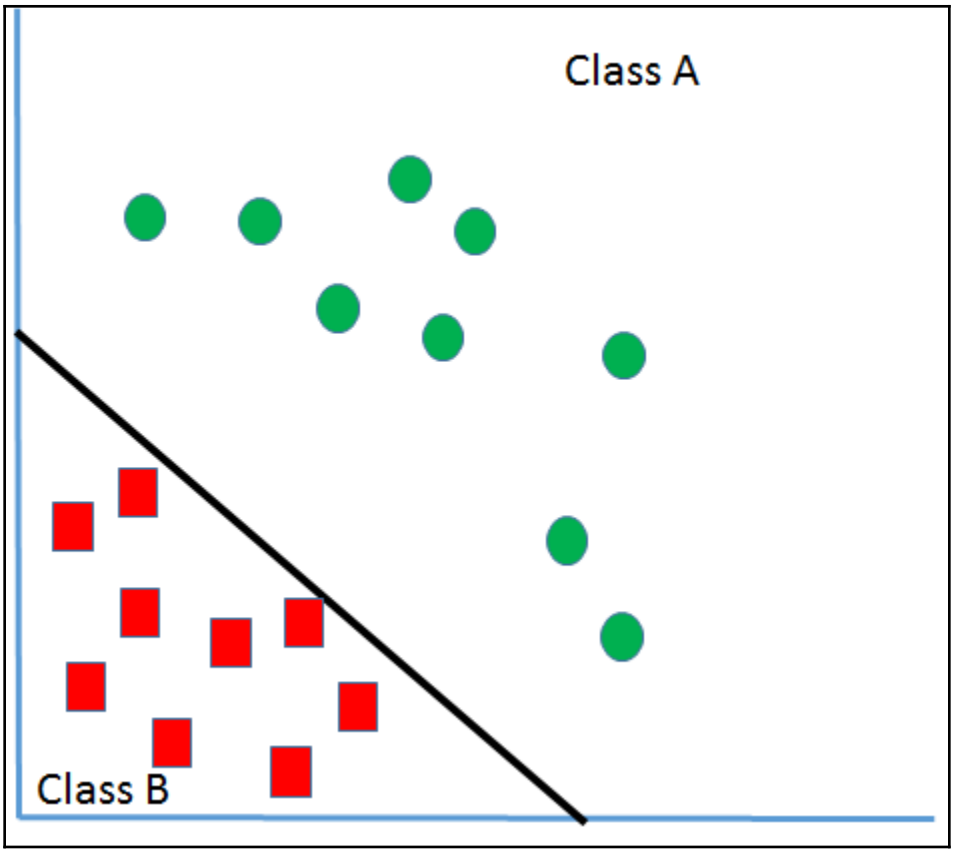
$$x = \left(\frac{x_1}{\|x\|}, \frac{x_2}{\|x\|}, \ldots, \frac{x_n}{\|x\|} \right)$$

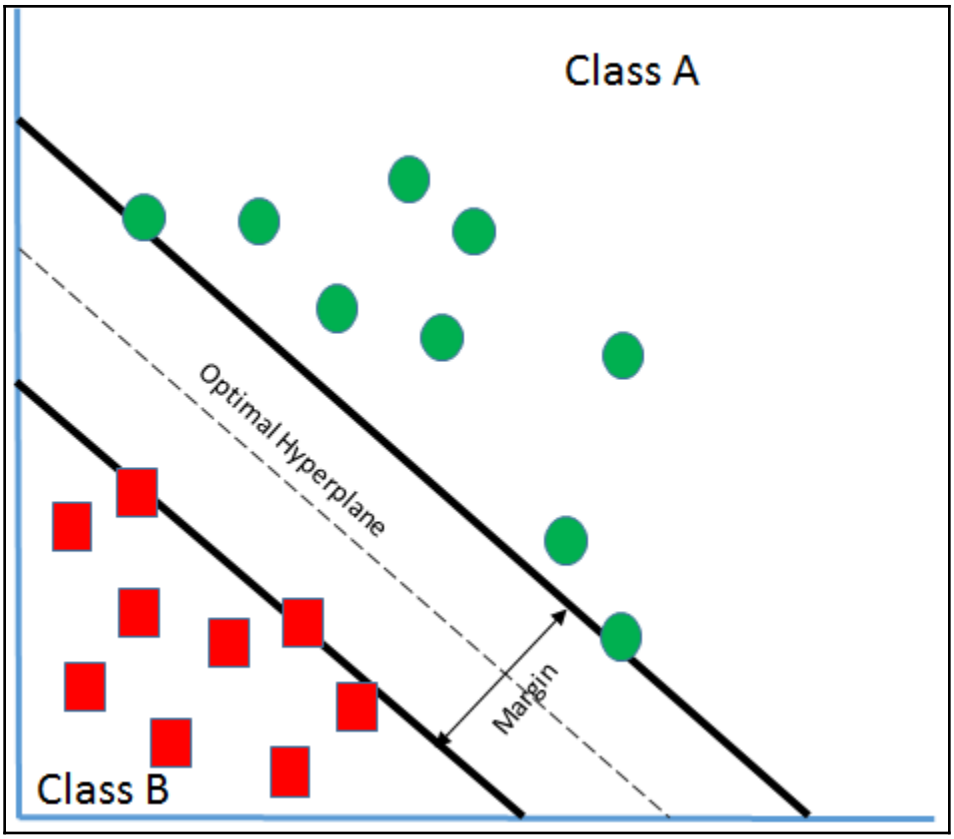

5
4
3
2
1
x
β = 60°
θ=45°
y
α = 15°
1
2
3
4
5
6
7
8

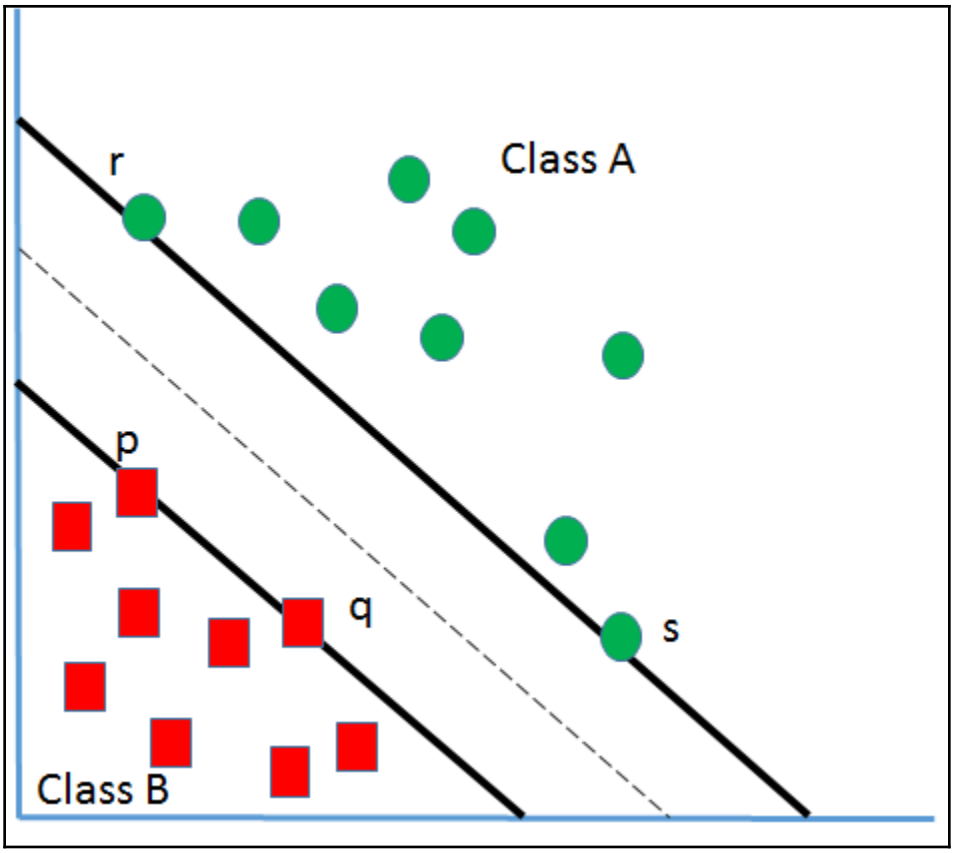

Non linearly separable
N -Not Sold
S -Sold
Price
Area
Linearly separable
Price
Area


10
0
−10
−20
−30
Z
−40
−50
−60
−70
−80
−20
0
20
40
Y
60
80
100
120
−20
0
20
40
60
X
80
100
120


Class A
Optimal Hyperplane
Margin
Class B


Class A
Class B


Class A
Optimal Hyperplane
Margin
Class B


r
Class A
p
q
s
Class B

Loading Libraries

In [1]:
```
from sklearn.svm import SVC
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
import numpy as np
```

Loading the data set

In [2]:
```
data= datasets.load_iris()
```

In [3]:
```
X=data.data
y= data.target
```

Standardization

In [4]:
```
scaler = StandardScaler()
X_std = scaler.fit_transform(X)
```

Model Building

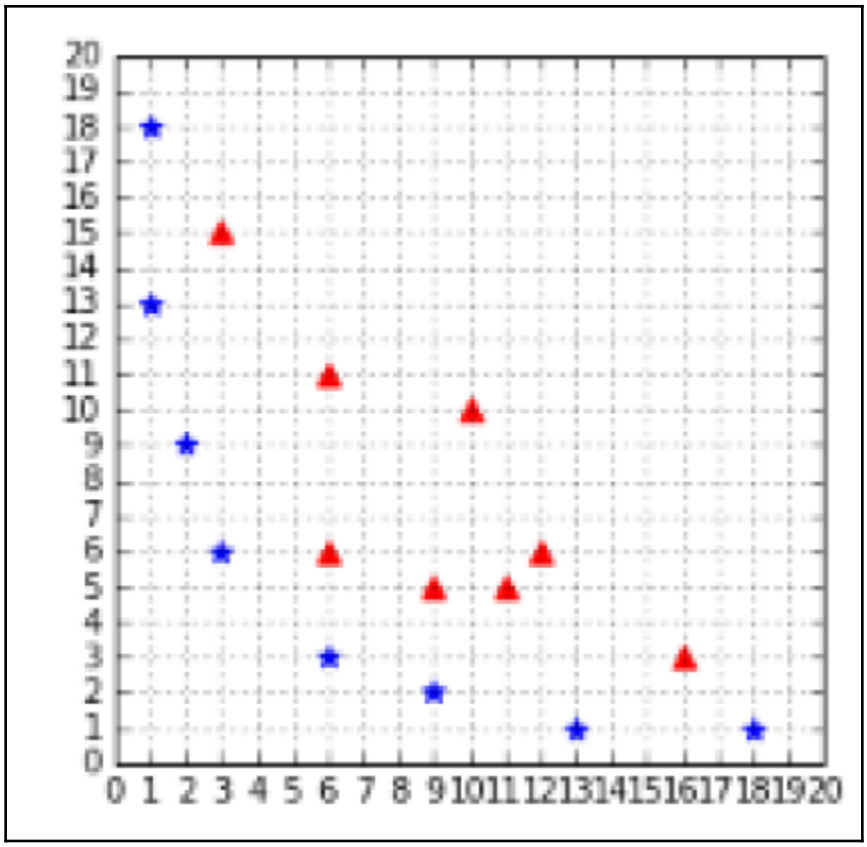
In [6]:
```
sv= SVC(kernel="linear",probability= True,random_state=0)

#training

model = sv.fit(X_std, y)
```

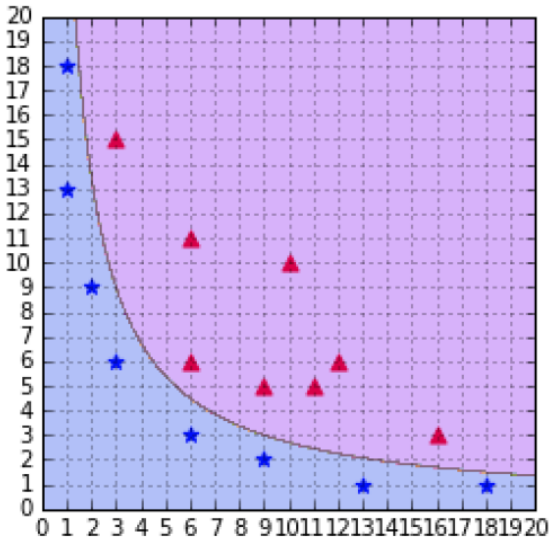
Probability for a new data

In [8]:
```
new_observation = [[.3, .4, .5, .6]]
```

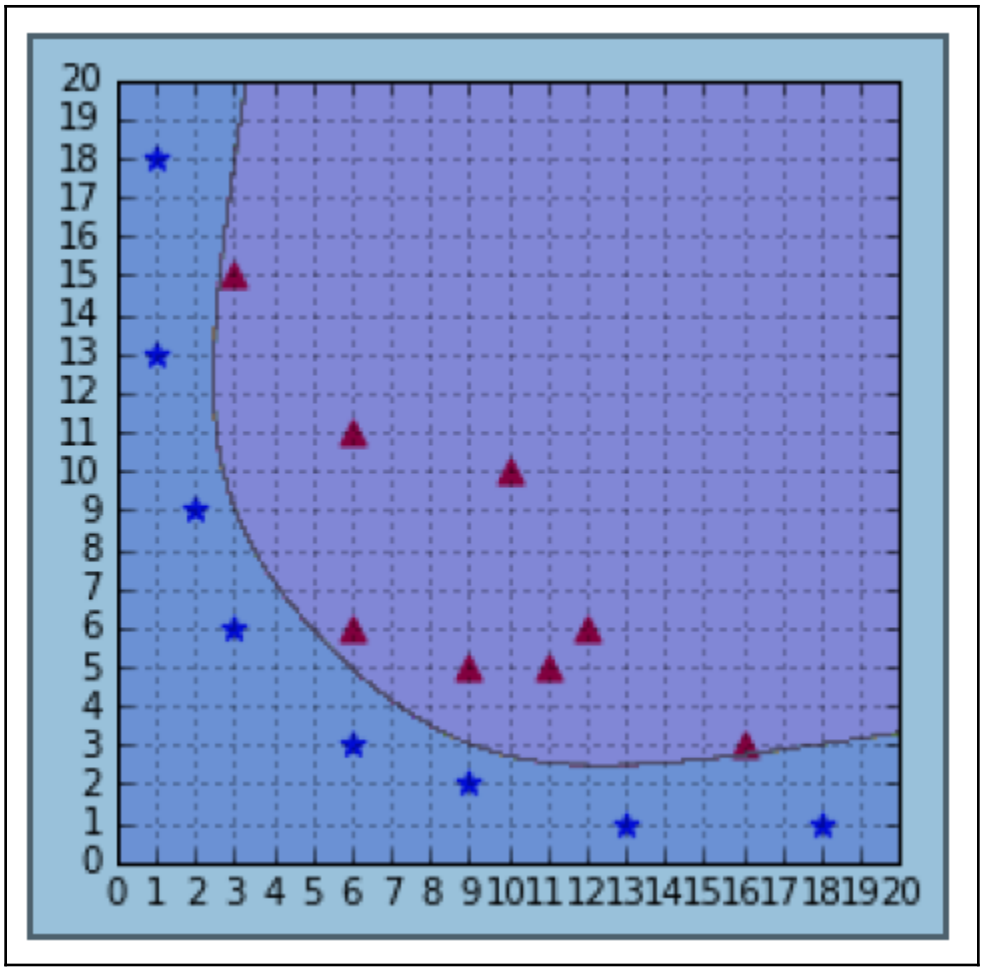
In [9]:
```
model.predict_proba(new_observation)
```

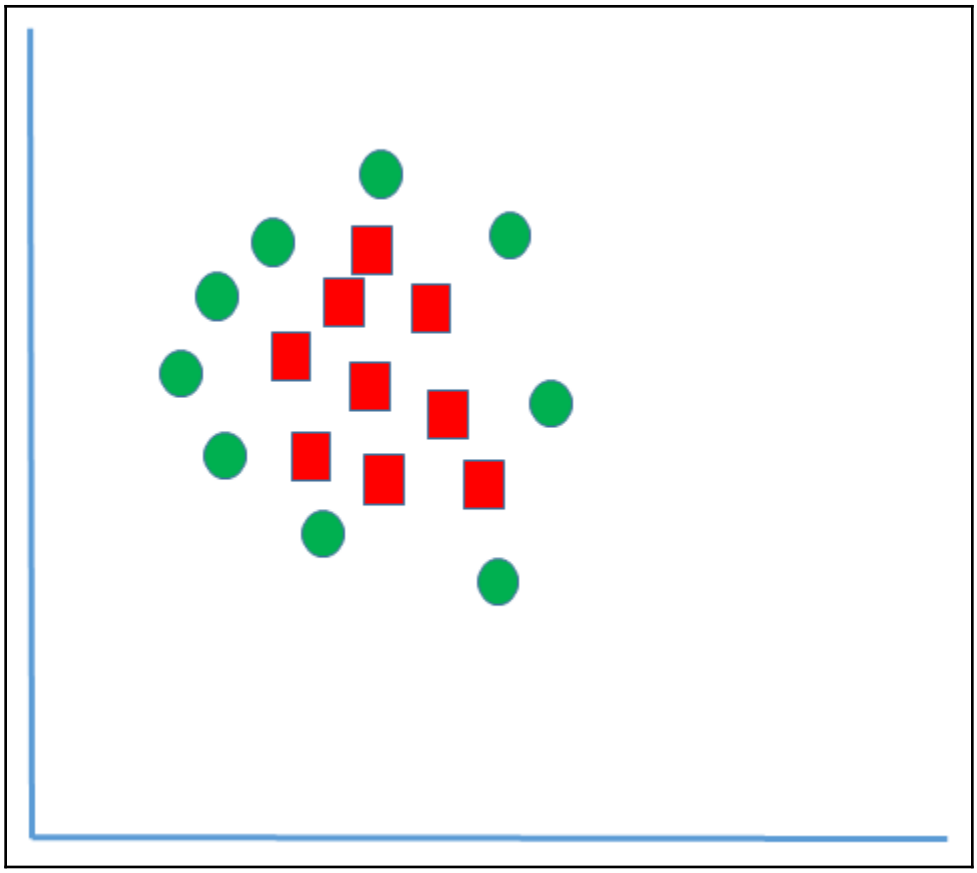
Out[9]:
```
array([[ 0.00832731,  0.85200651,  0.13966618]])
```


20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20



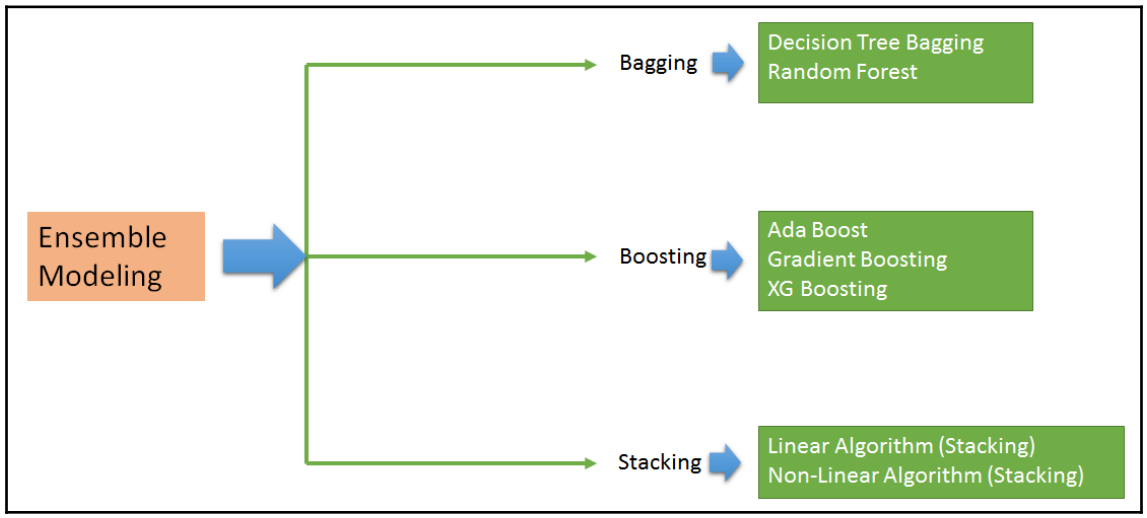
A SVM using a polynomial kernel is able to separate the data (degree=2)

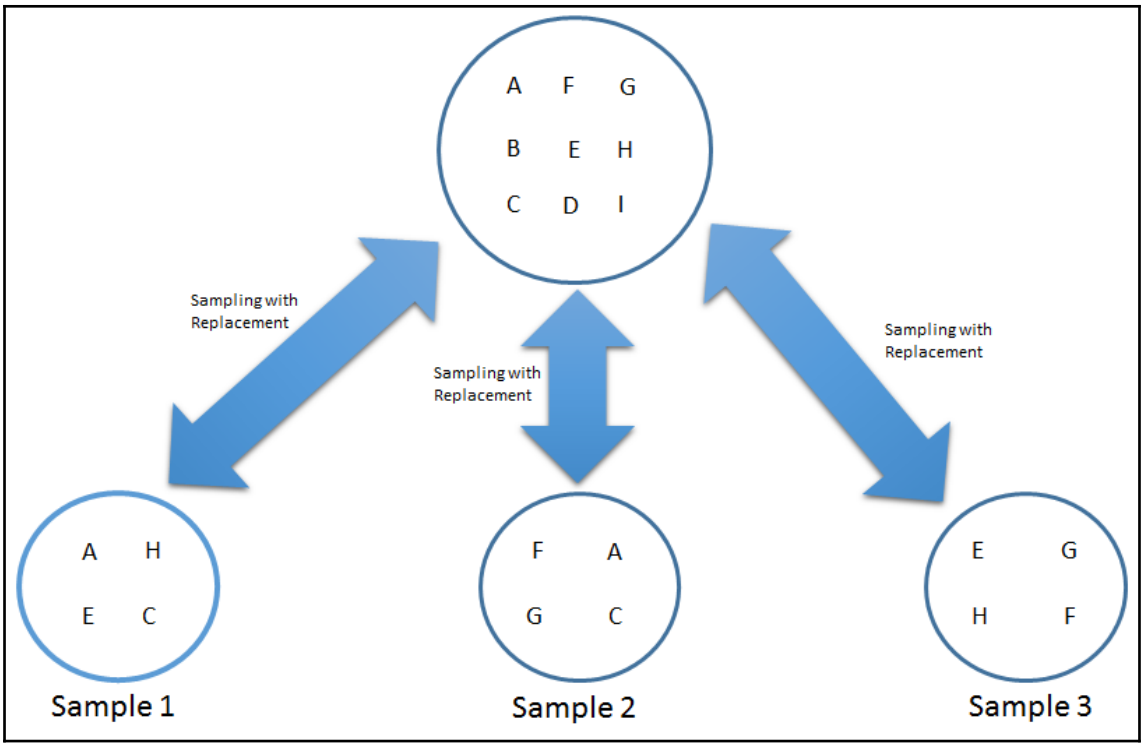

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

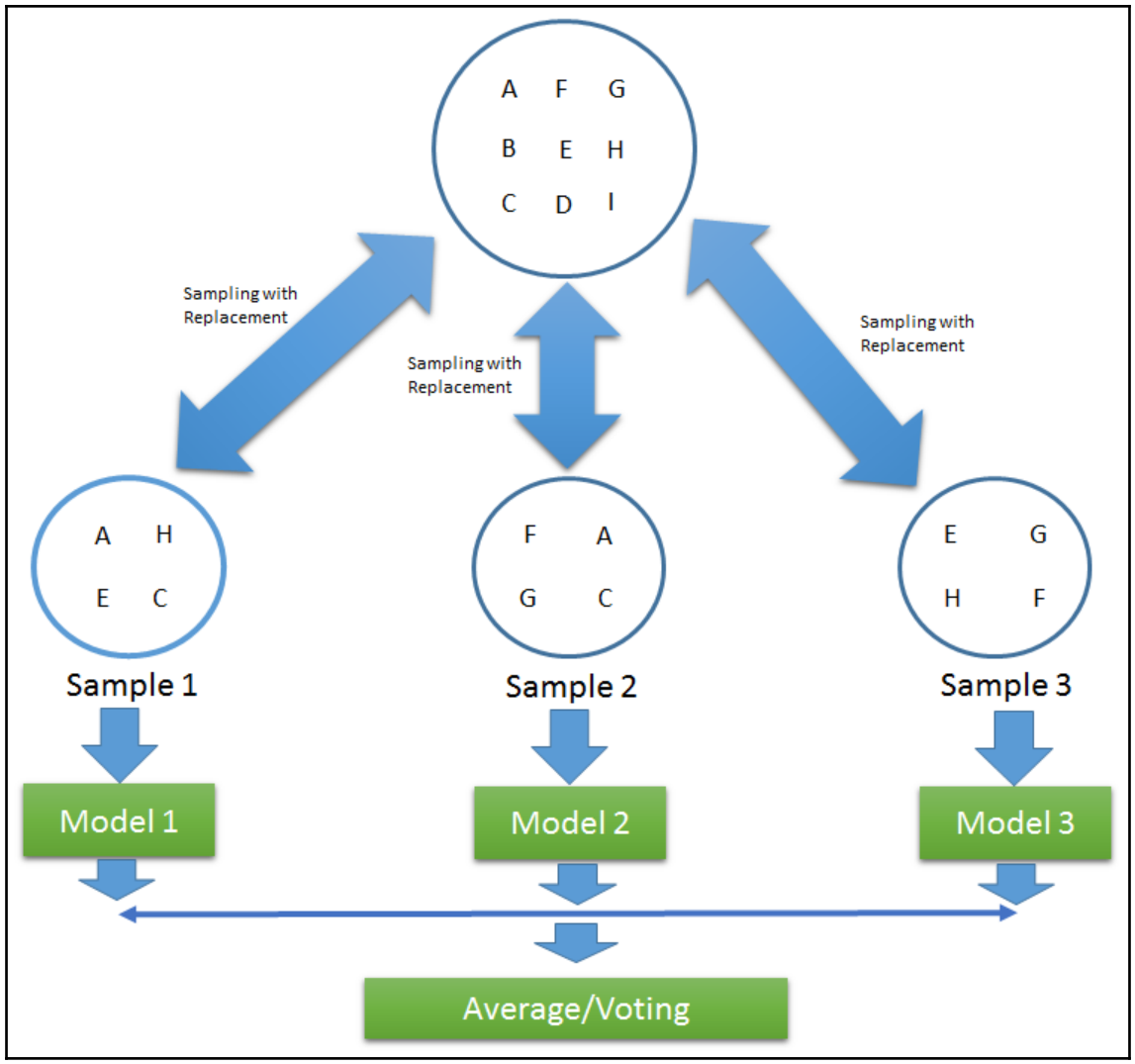


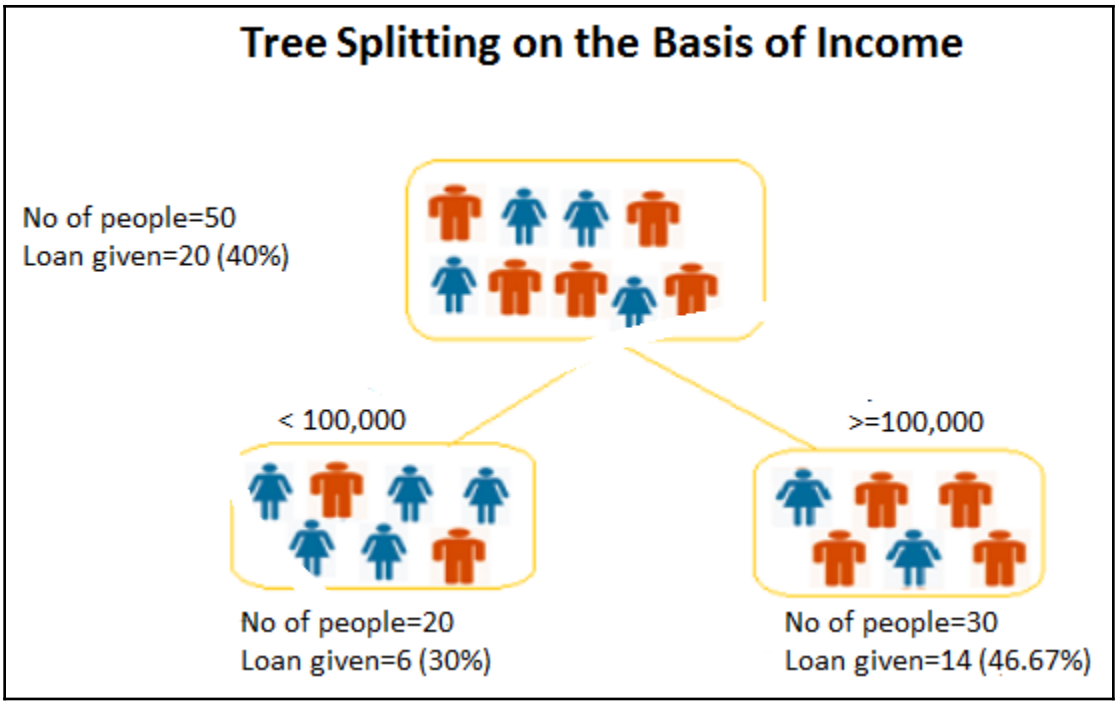
```
Best parameters set found on development set:

{'kernel': 'linear', 'C': 1}

Grid scores on training set:

0.937 (+/-0.057) for {'gamma': 0.0001, 'kernel': 'rbf', 'C': 1}
0.923 (+/-0.071) for {'gamma': 0.001, 'kernel': 'rbf', 'C': 1}
0.627 (+/-0.006) for {'gamma': 0.01, 'kernel': 'rbf', 'C': 1}
0.627 (+/-0.006) for {'gamma': 0.1, 'kernel': 'rbf', 'C': 1}
0.627 (+/-0.006) for {'gamma': 0.2, 'kernel': 'rbf', 'C': 1}
0.627 (+/-0.006) for {'gamma': 0.5, 'kernel': 'rbf', 'C': 1}
0.937 (+/-0.044) for {'gamma': 0.0001, 'kernel': 'rbf', 'C': 10}
0.918 (+/-0.047) for {'gamma': 0.001, 'kernel': 'rbf', 'C': 10}
0.629 (+/-0.015) for {'gamma': 0.01, 'kernel': 'rbf', 'C': 10}
0.627 (+/-0.006) for {'gamma': 0.1, 'kernel': 'rbf', 'C': 10}
0.627 (+/-0.006) for {'gamma': 0.2, 'kernel': 'rbf', 'C': 10}
0.627 (+/-0.006) for {'gamma': 0.5, 'kernel': 'rbf', 'C': 10}
0.934 (+/-0.031) for {'gamma': 0.0001, 'kernel': 'rbf', 'C': 100}
0.918 (+/-0.047) for {'gamma': 0.001, 'kernel': 'rbf', 'C': 100}
0.629 (+/-0.015) for {'gamma': 0.01, 'kernel': 'rbf', 'C': 100}
0.627 (+/-0.006) for {'gamma': 0.1, 'kernel': 'rbf', 'C': 100}
0.627 (+/-0.006) for {'gamma': 0.2, 'kernel': 'rbf', 'C': 100}
0.627 (+/-0.006) for {'gamma': 0.5, 'kernel': 'rbf', 'C': 100}
0.930 (+/-0.040) for {'gamma': 0.0001, 'kernel': 'rbf', 'C': 1000}
0.918 (+/-0.047) for {'gamma': 0.001, 'kernel': 'rbf', 'C': 1000}
0.629 (+/-0.015) for {'gamma': 0.01, 'kernel': 'rbf', 'C': 1000}
0.627 (+/-0.006) for {'gamma': 0.1, 'kernel': 'rbf', 'C': 1000}
0.627 (+/-0.006) for {'gamma': 0.2, 'kernel': 'rbf', 'C': 1000}
0.627 (+/-0.006) for {'gamma': 0.5, 'kernel': 'rbf', 'C': 1000}
0.960 (+/-0.048) for {'kernel': 'linear', 'C': 1}
0.946 (+/-0.048) for {'kernel': 'linear', 'C': 10}
0.953 (+/-0.039) for {'kernel': 'linear', 'C': 100}
0.953 (+/-0.039) for {'kernel': 'linear', 'C': 1000}
```

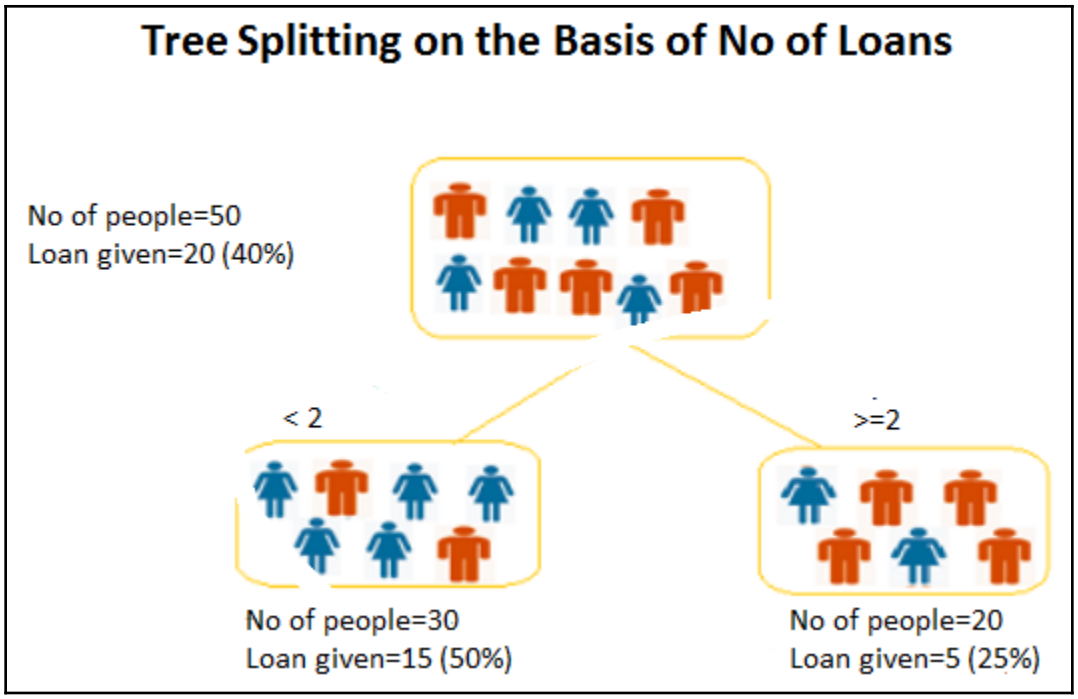
Chapter 3: Performance in Ensemble Learning

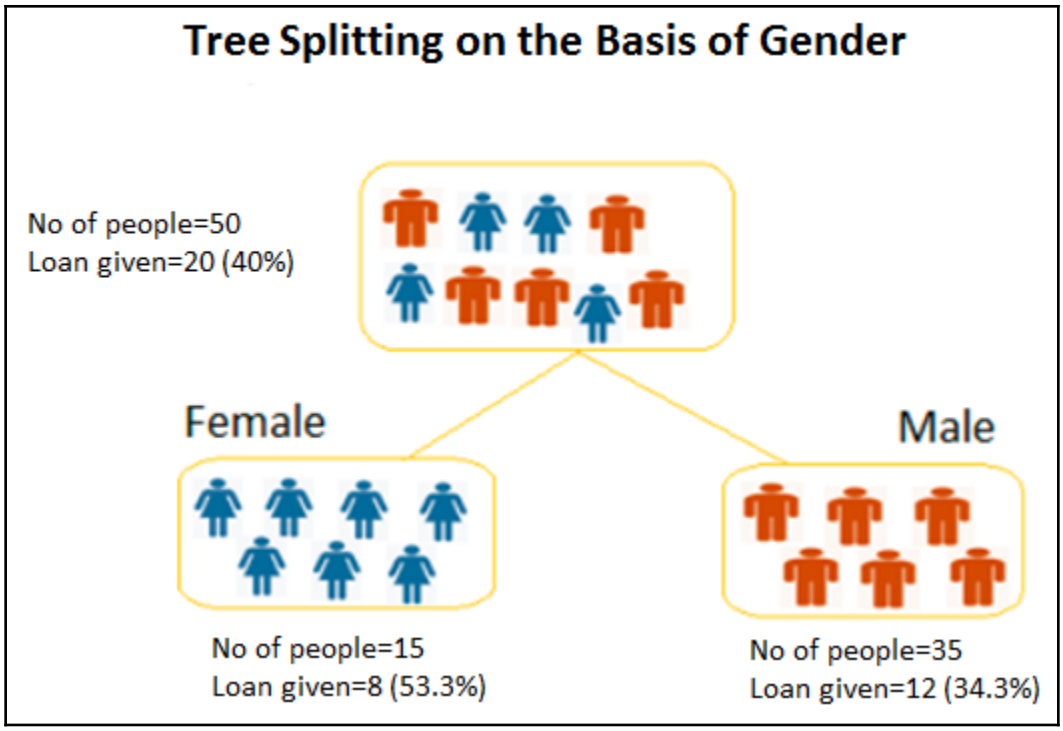


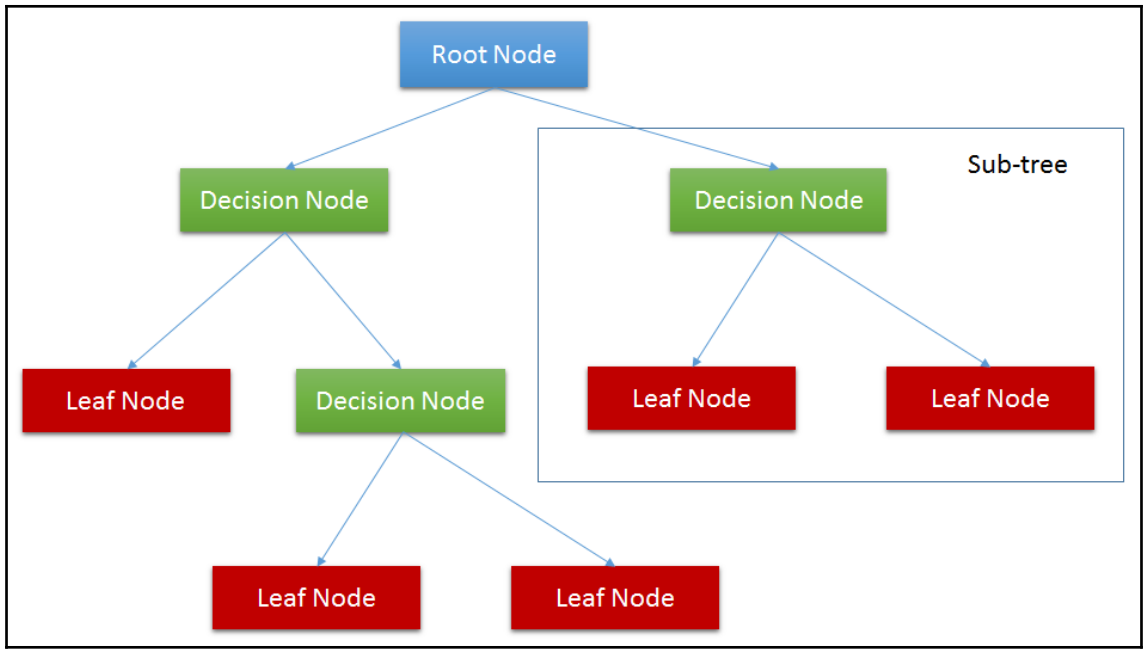

A F G
B E H
C D I
Sampling with Replacement
Sampling with Replacement
Sampling with Replacement
A H
E C
F A
G C
E G
H F
Sample 1
Sample 2
Sample 3

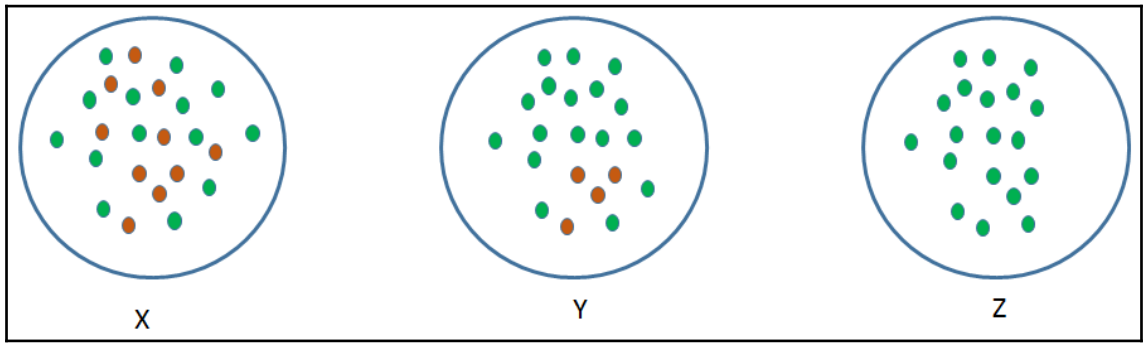

A F G
B E H
C D I
Sampling with Replacement
Sampling with Replacement
Sampling with Replacement
A H
E C
F A
G C
E G
H F
Sample 1
Sample 2
Sample 3
Model 1
Model 2
Model 3
Average/Voting


Tree Splitting on the Basis of Income
No of people=50
Loan given=20 (40%)
< 100,000
>=100,000
No of people=20
Loan given=6 (30%)
No of people=30
Loan given=14 (46.67%)


Tree Splitting on the Basis of No of Loans
No of people=50
Loan given=20 (40%)
< 2
>=2
No of people=30
Loan given=15 (50%)
No of people=20
Loan given=5 (25%)


Tree Splitting on the Basis of Gender
No of people=50
Loan given=20 (40%)
Female
Male
No of people=15
Loan given=8 (53.3%)
No of people=35
Loan given=12 (34.3%)


Root Node
Decision Node
Decision Node
Sub-tree
Leaf Node
Decision Node
Leaf Node
Leaf Node
Leaf Node
Leaf Node


X
Y
Z


All Data
random subset
random subset
random subset
random subset
tree
tree
tree
tree
tree
At each node:
choose some ballsubset of variables at random
find a variable (and a value for that variable) which optimizes the split

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
id                         569 non-null int64
diagnosis                  569 non-null object
radius_mean                569 non-null float64
texture_mean               569 non-null float64
perimeter_mean             569 non-null float64
area_mean                  569 non-null float64
smoothness_mean            569 non-null float64
compactness_mean           569 non-null float64
concavity_mean             569 non-null float64
concave points_mean        569 non-null float64
symmetry_mean              569 non-null float64
fractal_dimension_mean     569 non-null float64
radius_se                  569 non-null float64
texture_se                 569 non-null float64
perimeter_se               569 non-null float64
area_se                    569 non-null float64
smoothness_se              569 non-null float64
compactness_se             569 non-null float64
concavity_se               569 non-null float64
concave points_se          569 non-null float64
symmetry_se                569 non-null float64
fractal_dimension_se       569 non-null float64
radius_worst               569 non-null float64
texture_worst              569 non-null float64
perimeter_worst            569 non-null float64
area_worst                 569 non-null float64
smoothness_worst           569 non-null float64
compactness_worst          569 non-null float64
concavity_worst            569 non-null float64
concave points_worst       569 non-null float64
symmetry_worst             569 non-null float64
fractal_dimension_worst    569 non-null float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.3+ KB
```

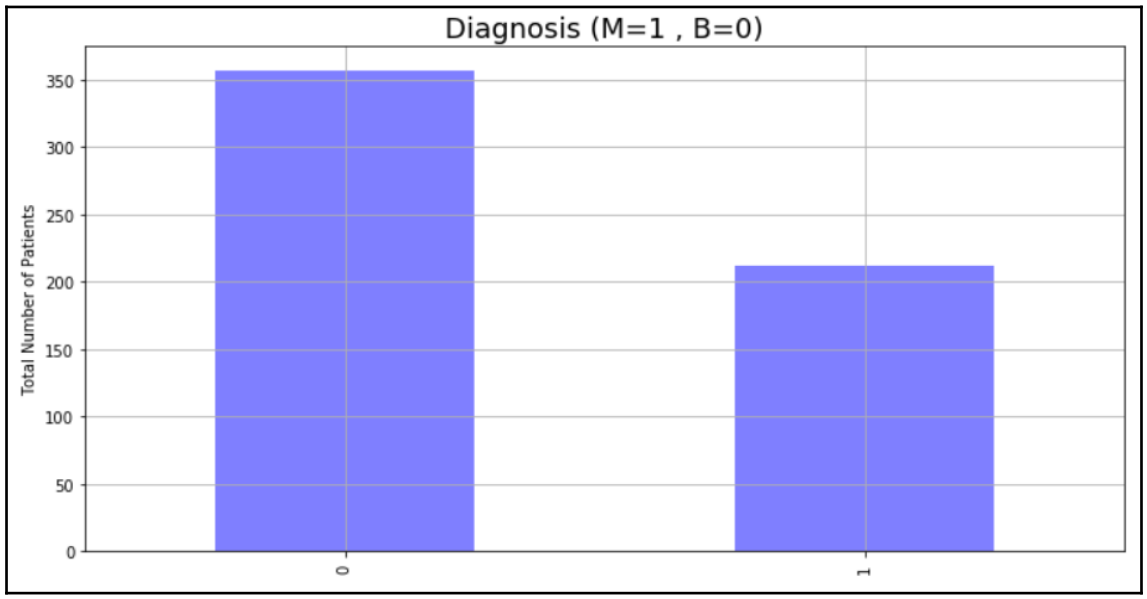
	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	ra
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	

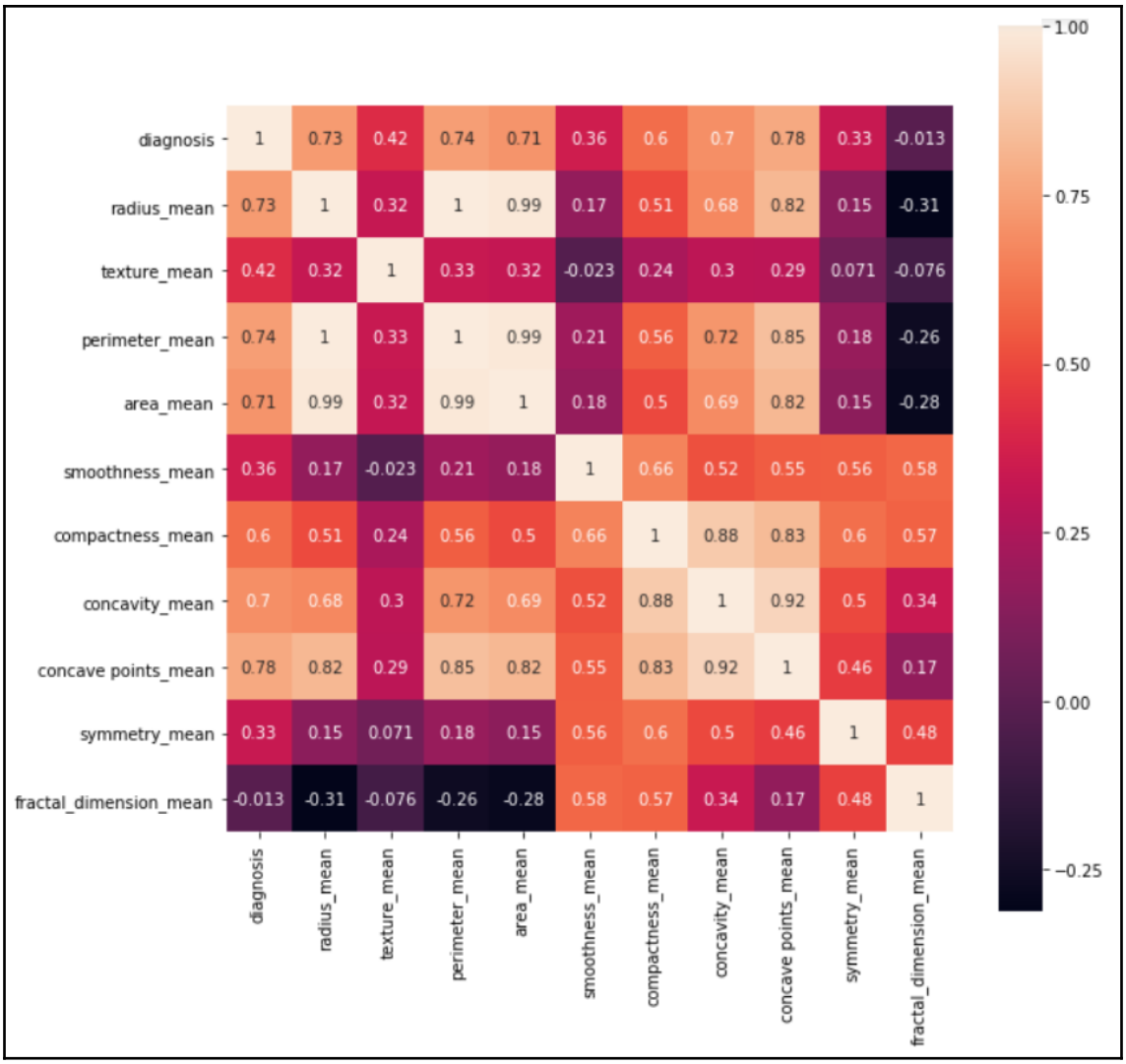
5 rows × 32 columns

```
array(['M', 'B'], dtype=object)
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200

8 rows × 31 columns






```
Shape of training set : 455 & Shape of test set : 114
There are very few data points so 10-fold cross validation should give us a better estimate
```

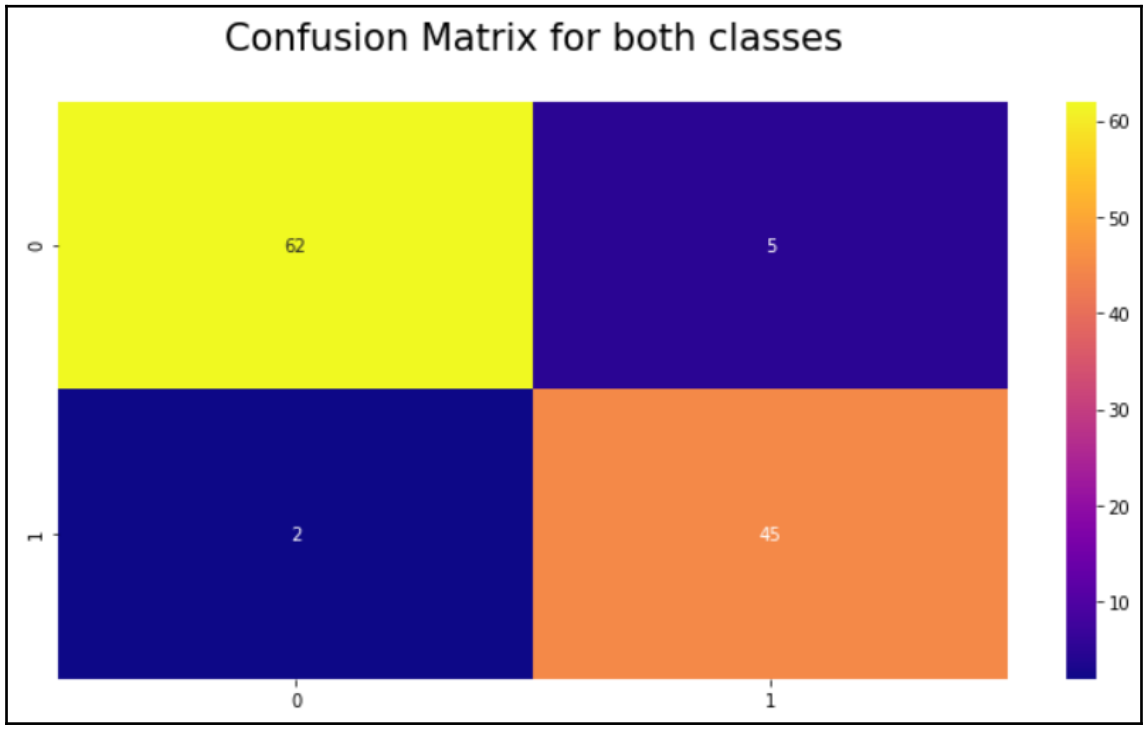
```
{'mean_fit_time': array([ 0.03339021,  0.06928167,  0.12723982,  0.18830116,  0.37480223,
         0.62295649,  0.03198514,  0.06366966,  0.13215754,  0.19233065,
         0.37359273,  0.6272675 ,  0.03098235,  0.06176491,  0.12673779,
         0.18218436,  0.37519989,  0.62526178,  0.03318925,  0.06437261,
         0.13265123,  0.19200687,  0.37980766,  0.64130707,  0.03298304,
         0.06457033,  0.1324533 ,  0.19311185,  0.37760334,  0.64290974,
         0.03268728,  0.06437161,  0.13275001,  0.19471939,  0.38081119,
         0.62576702,  0.03238645,  0.06457105,  0.13024404,  0.19472077,
         0.38281748,  0.63689423,  0.03278713,  0.06437168,  0.12603452,
         0.19481535,  0.38311908,  0.63247719,  0.03178444,  0.06337047,
         0.12833807,  0.19582243,  0.37660072,  0.63127663,  0.03188739,
         0.06417322,  0.12814078,  0.19571686,  0.381515  ,  0.62877288,
         0.03278725,  0.06447268,  0.12914252,  0.19521923,  0.3822166 ,
         0.62776752,  0.03258617,  0.06427264,  0.12723784,  0.19631989,
         0.38231587,  0.63749213]),
```

```
GridSearhCV best model:

The best score:  0.949033391916
The best parameter: {'max_depth': 8, 'max_features': 'sqrt', 'n_estimators': 150}
The best model estimator: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=8, max_features='sqrt', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=150, n_jobs=1,
            oob_score=True, random_state=None, verbose=0, warm_start=False)
```

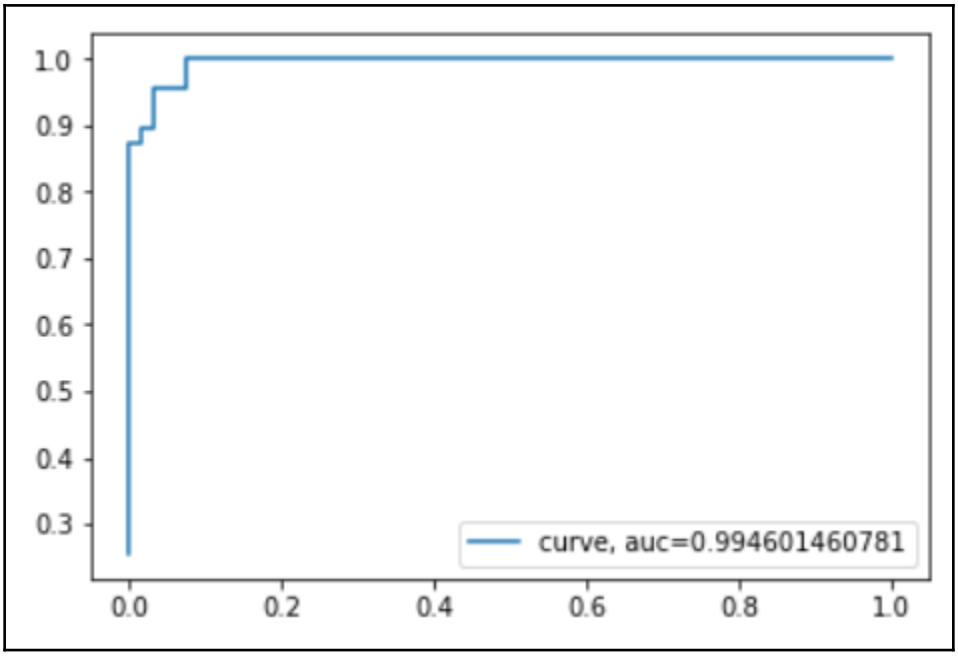
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
             max_depth=8, max_features='sqrt', max_leaf_nodes=None,
             min_impurity_decrease=0.0, min_impurity_split=None,
             min_samples_leaf=1, min_samples_split=2,
             min_weight_fraction_leaf=0.0, n_estimators=150, n_jobs=1,
             oob_score=True, random_state=None, verbose=0, warm_start=False)
```

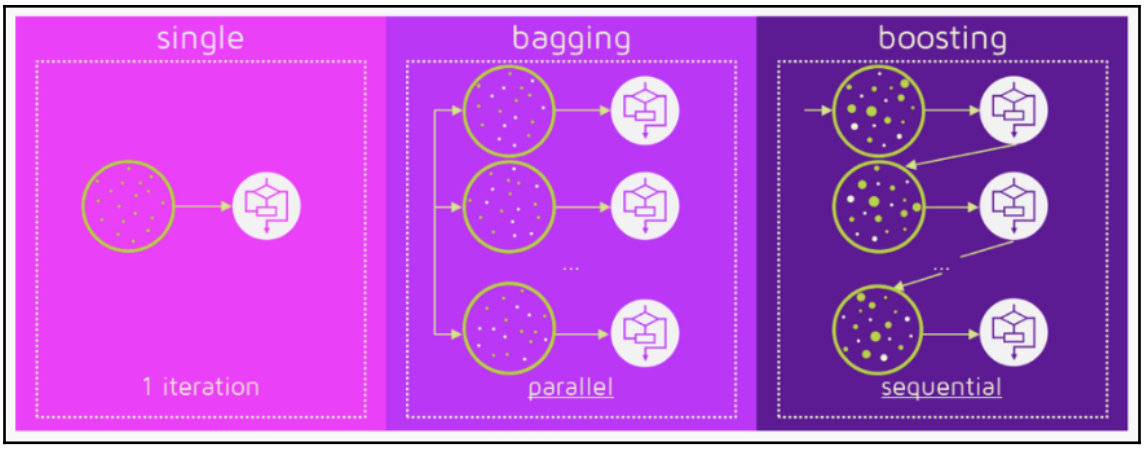
(114, 2)

	Truth	Predictions
0	1	1
1	0	0
2	0	0
3	0	0
4	0	0

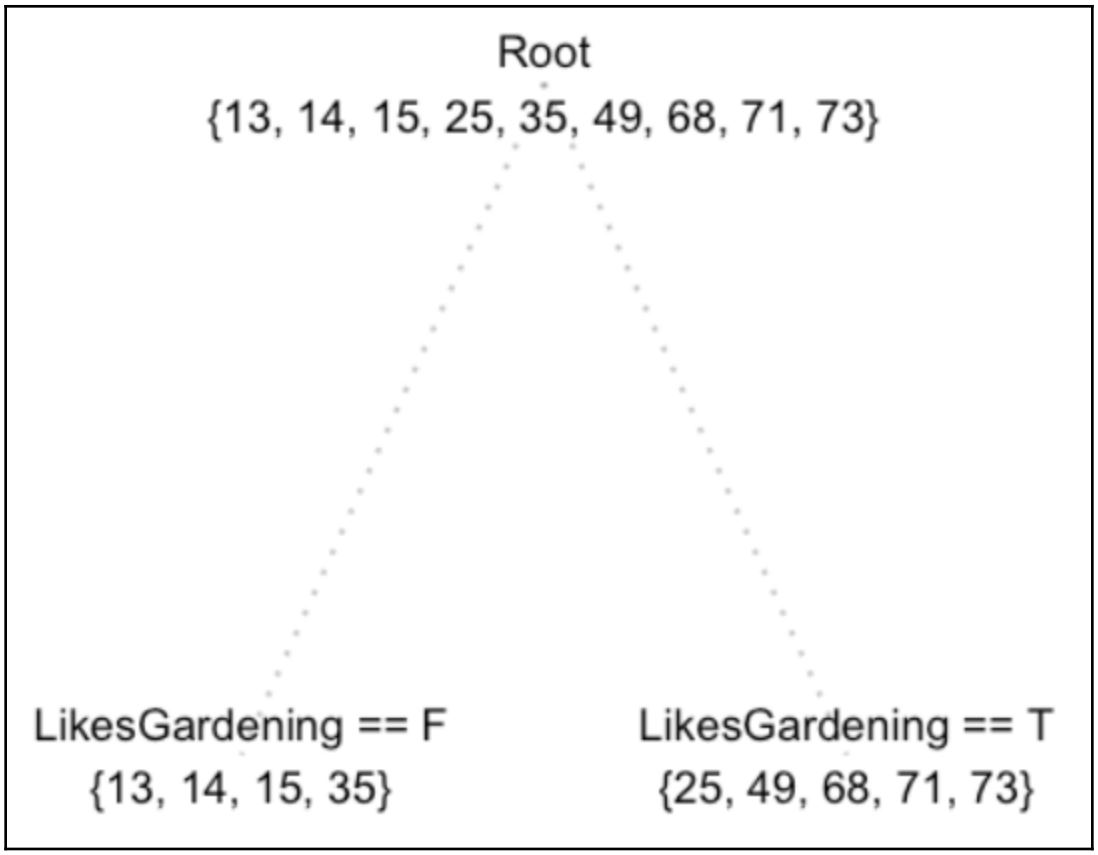


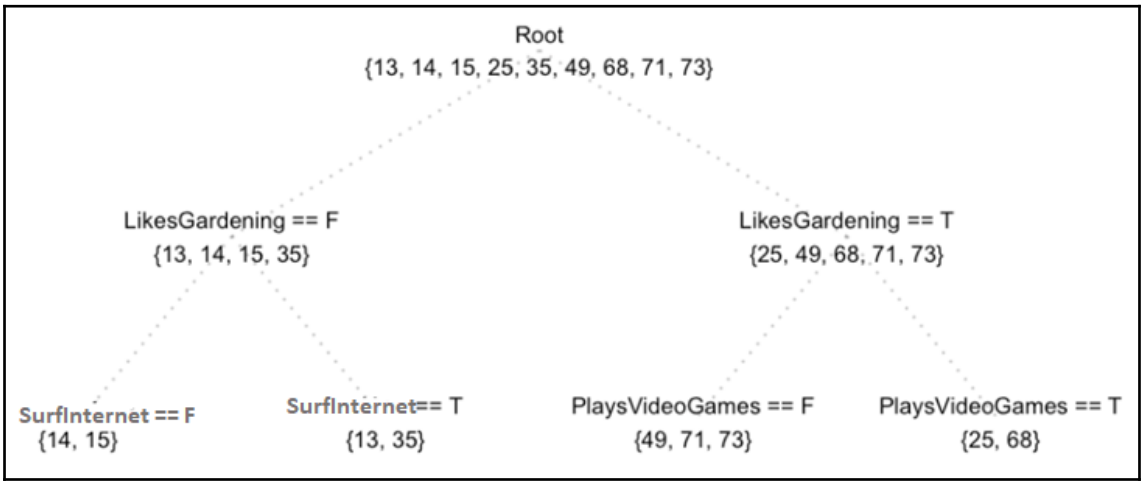
	precision	recall	f1-score	support
Benign [Class 0]	0.97	0.93	0.95	67
Malignant[Class 1]	0.90	0.96	0.93	47
avg / total	0.94	0.94	0.94	114



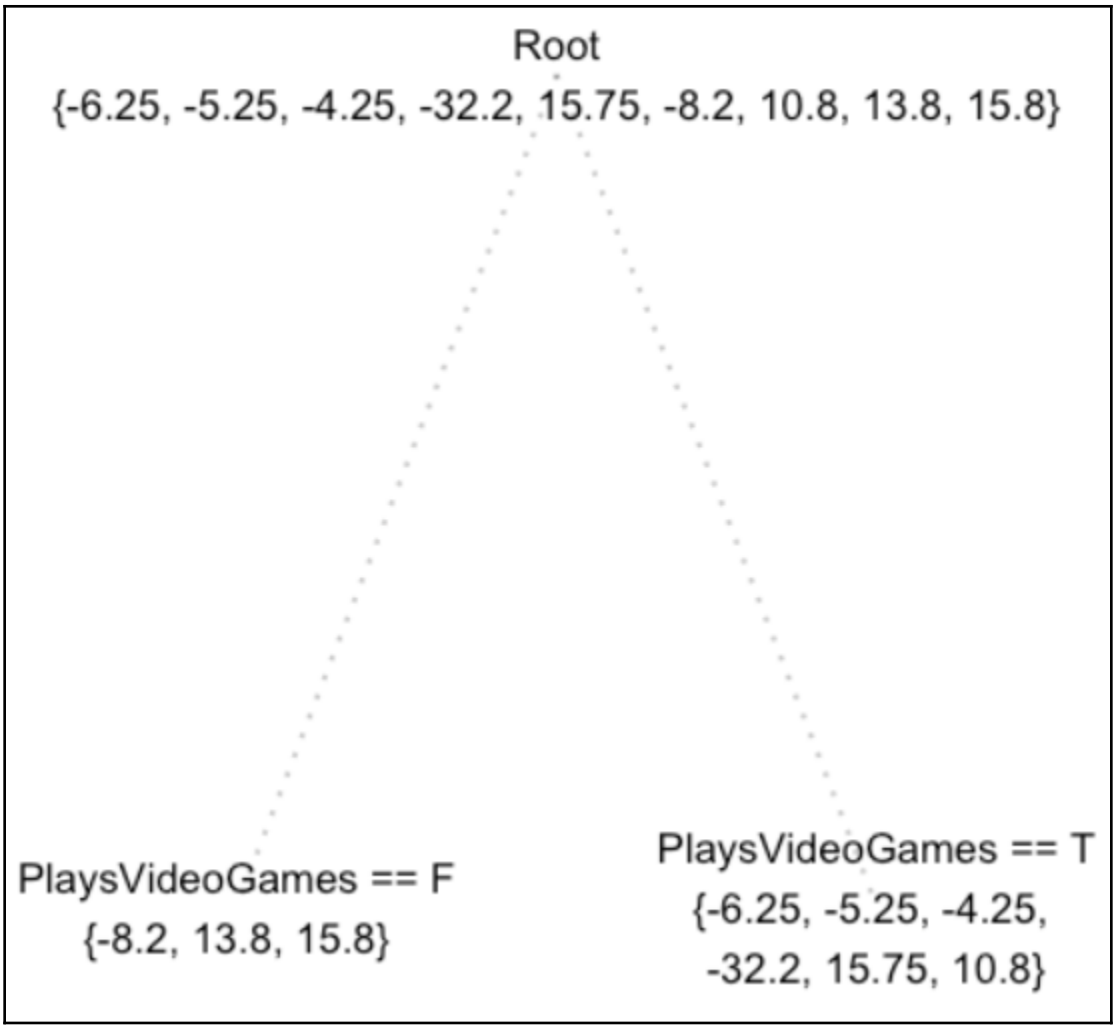


Person ID	Age	LikesGardening	PlaysVideoGames	LikesSurfingNet
1	13	FALSE	TRUE	TRUE
2	14	FALSE	TRUE	FALSE
3	15	FALSE	TRUE	FALSE
4	25	TRUE	TRUE	TRUE
5	35	FALSE	TRUE	TRUE
6	49	TRUE	FALSE	FALSE
7	68	TRUE	TRUE	TRUE
8	71	TRUE	FALSE	FALSE
9	73	TRUE	FALSE	TRUE


Root
{13, 14, 15, 25, 35, 49, 68, 71, 73}
LikesGardening == F
{13, 14, 15, 35}
LikesGardening == T
{25, 49, 68, 71, 73}


Root
{13, 14, 15, 25, 35, 49, 68, 71, 73}
LikesGardening == F
{13, 14, 15, 35}
LikesGardening == T
{25, 49, 68, 71, 73}
SurfInternet == F
{14, 15}
SurfInternet== T
{13, 35}
PlaysVideoGames == F
{49, 71, 73}
PlaysVideoGames == T
{25, 68}

Person ID	Age	Prediction	Residual
1	13	19.25	-6.25
2	14	19.25	-5.25
3	15	19.25	-4.25
4	25	57.2	-32.2
5	35	19.25	15.75
6	49	57.2	-8.2
7	68	57.2	10.8
8	71	57.2	13.8
9	73	57.2	15.8


Root
{-6.25, -5.25, -4.25, -32.2, 15.75, -8.2, 10.8, 13.8, 15.8}
PlaysVideoGames == F
{-8.2, 13.8, 15.8}
PlaysVideoGames == T
{-6.25, -5.25, -4.25,
-32.2, 15.75, 10.8}

Person ID	Age	Prediction	Residual	Prediction on residuals	Combined Prediction	Final Residuals
1	13	19.25	-6.25	-3.567	15.683	2.683
2	14	19.25	-5.25	-3.567	15.683	1.683
3	15	19.25	-4.25	-3.567	15.683	0.683
4	25	57.2	-32.2	-3.567	53.633	28.633
5	35	19.25	15.75	-3.567	15.683	-19.317
6	49	57.2	-8.2	7.133	64.333	15.333
7	68	57.2	10.8	-3.567	53.633	-14.367
8	71	57.2	13.8	7.133	64.333	-6.667
9	73	57.2	15.8	7.133	64.333	-8.667

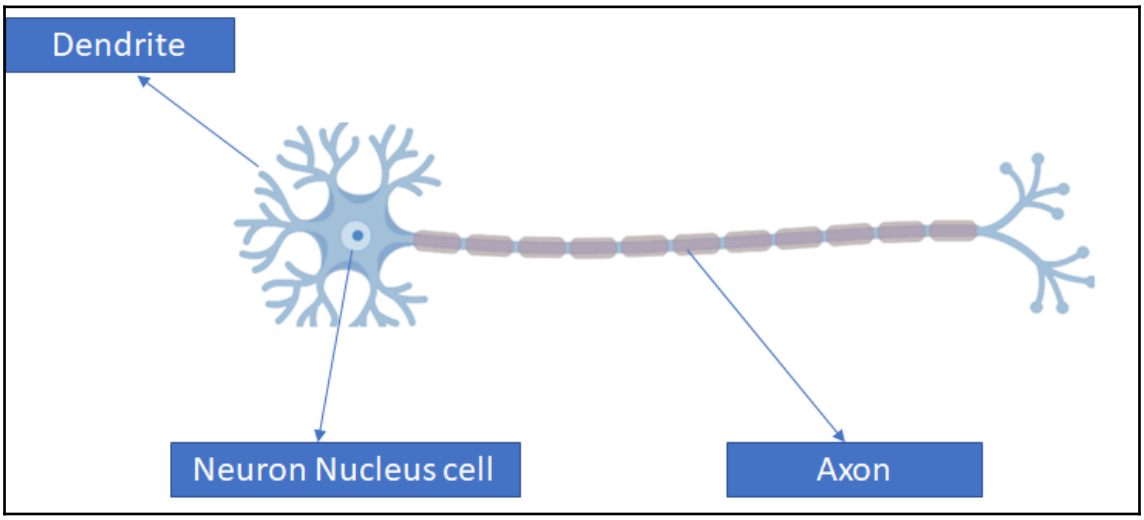
```
{'mean_fit_time': array([ 0.03494983,  0.06627471,  0.12813728,  0.18599195,  0.24799919,
         0.33799727,  0.02897744,  0.06652975,  0.12413988,  0.1881336 ,
         0.23819361,  0.29800813,  0.03043046,  0.05891035,  0.11034069,
         0.18427889,  0.221032  ,  0.30923851,  0.06096241,  0.12604637,
         0.20514412,  0.2363739 ,  0.28559666,  0.33619659,  0.06156273,
         0.11972167,  0.2156975 ,  0.21138661,  0.26119628,  0.31739309,
         0.05794506,  0.12085245,  0.19251099,  0.20506291,  0.2632467 ,
         0.31910994,  0.06727653,  0.15261815,  0.25119522,  0.26912684,
         0.30465198,  0.38809872,  0.07574708,  0.15285401,  0.2602922 ,
         0.27513788,  0.29188688,  0.35694766,  0.06843309,  0.14909971,
         0.22379372,  0.23729031,  0.28556411,  0.34478147,  0.06652713,
         0.14558716,  0.23933291,  0.25528142,  0.29443026,  0.35715394,
         0.06782725,  0.1500011 ,  0.22931345,  0.24915528,  0.28485856,
         0.35234025,  0.06879389,  0.1524158 ,  0.23522427,  0.24537499,
         0.28084641,  0.34170115]),
 'mean_score_time': array([ 0.00045459,  0.00025072,  0.00040183,  0.00055137,  0.00049999,
         0.0014482 ,  0.0003505 ,  0.00030062,  0.00044746,  0.00094807,
```

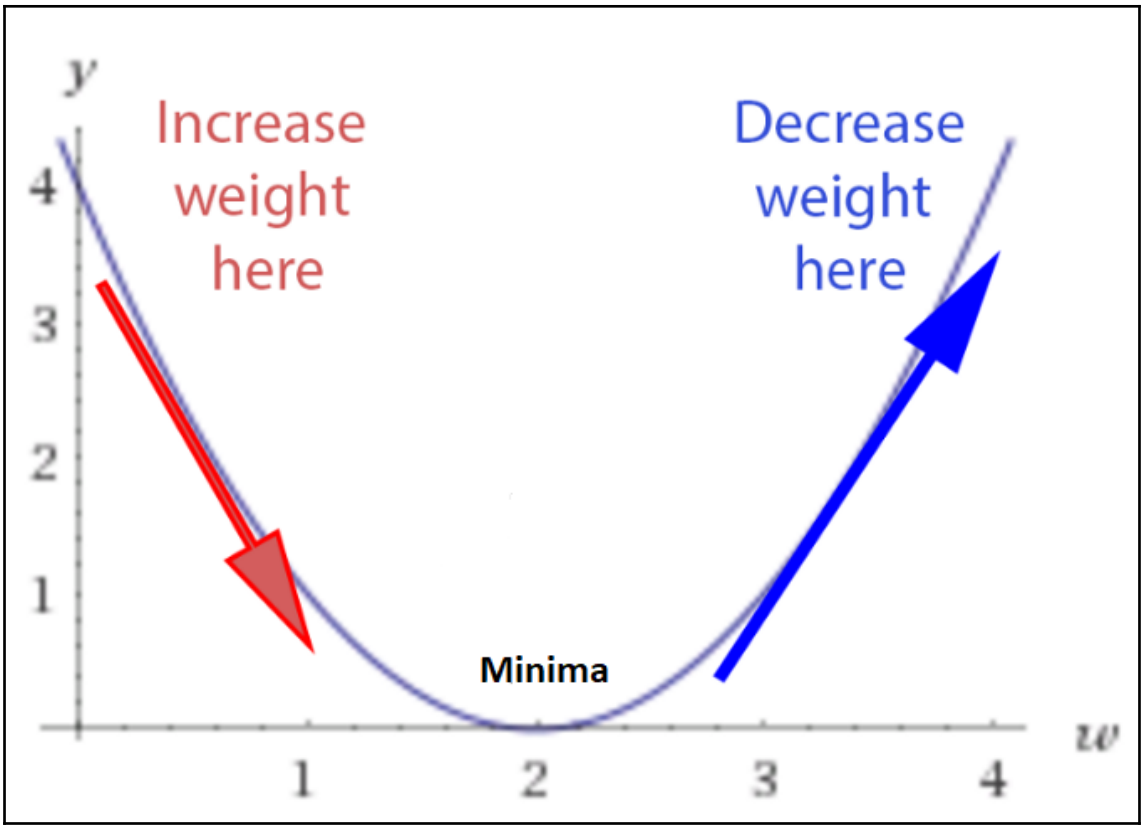
```
GridSearhCV best model:

The best score:  0.952548330404
The best parameter: {'n_estimators': 150, 'max_depth': 5, 'max_features': 'sqrt'}
The best model estimator: GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=5,
              max_features='sqrt', max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=150,
              presort='auto', random_state=10, subsample=0.8, verbose=0,
              warm_start=False)
```

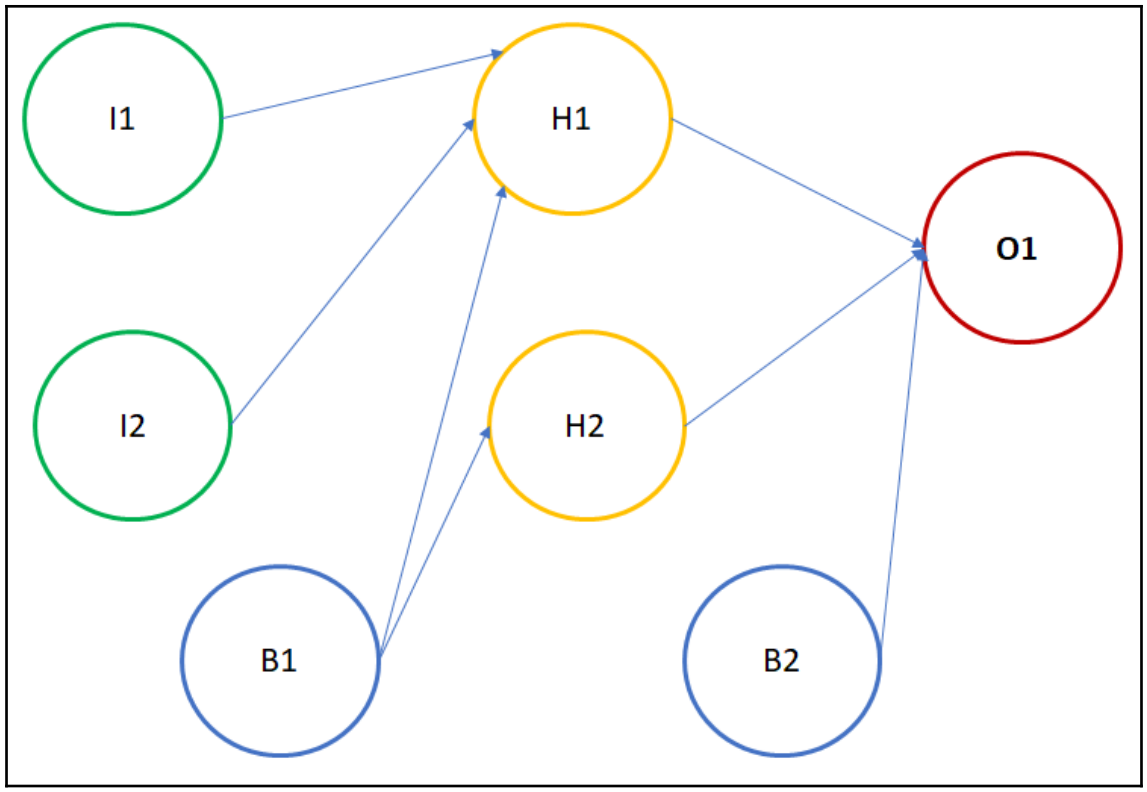
(114, 2)

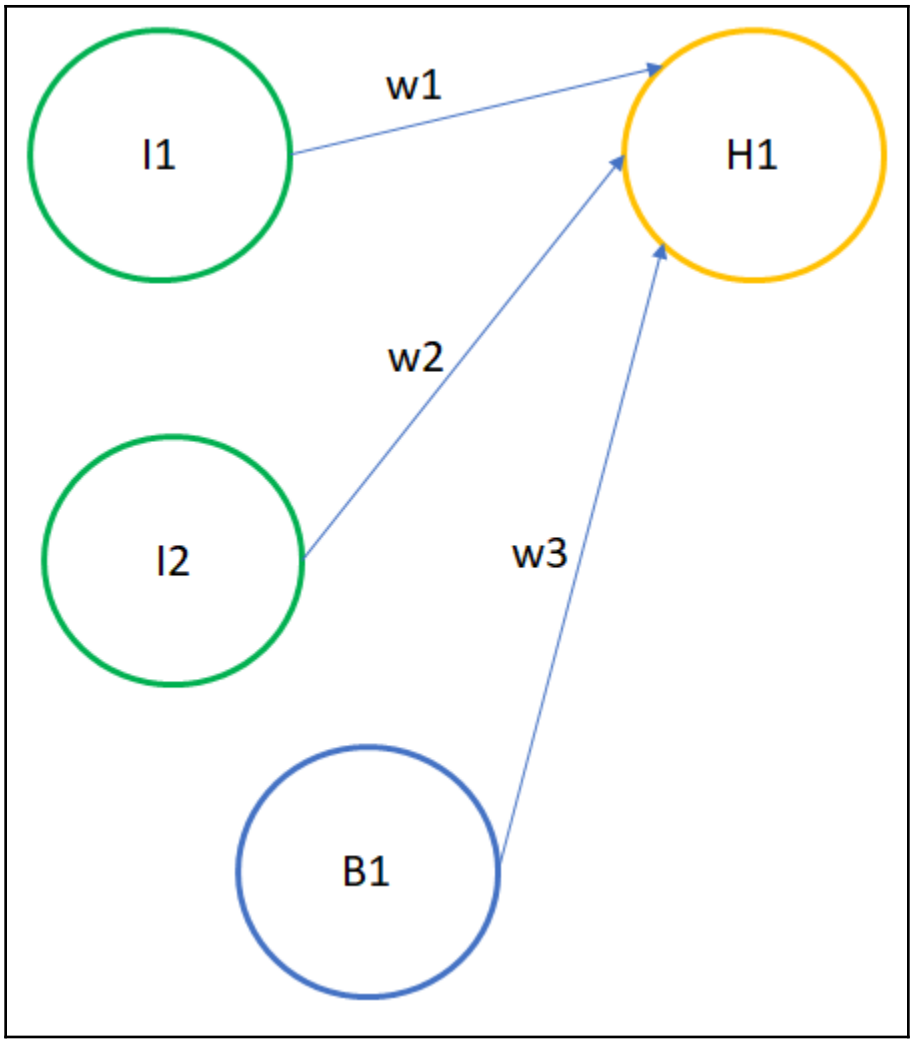
	Truth	Predictions
0	1	1
1	0	0
2	0	0
3	0	0
4	0	0

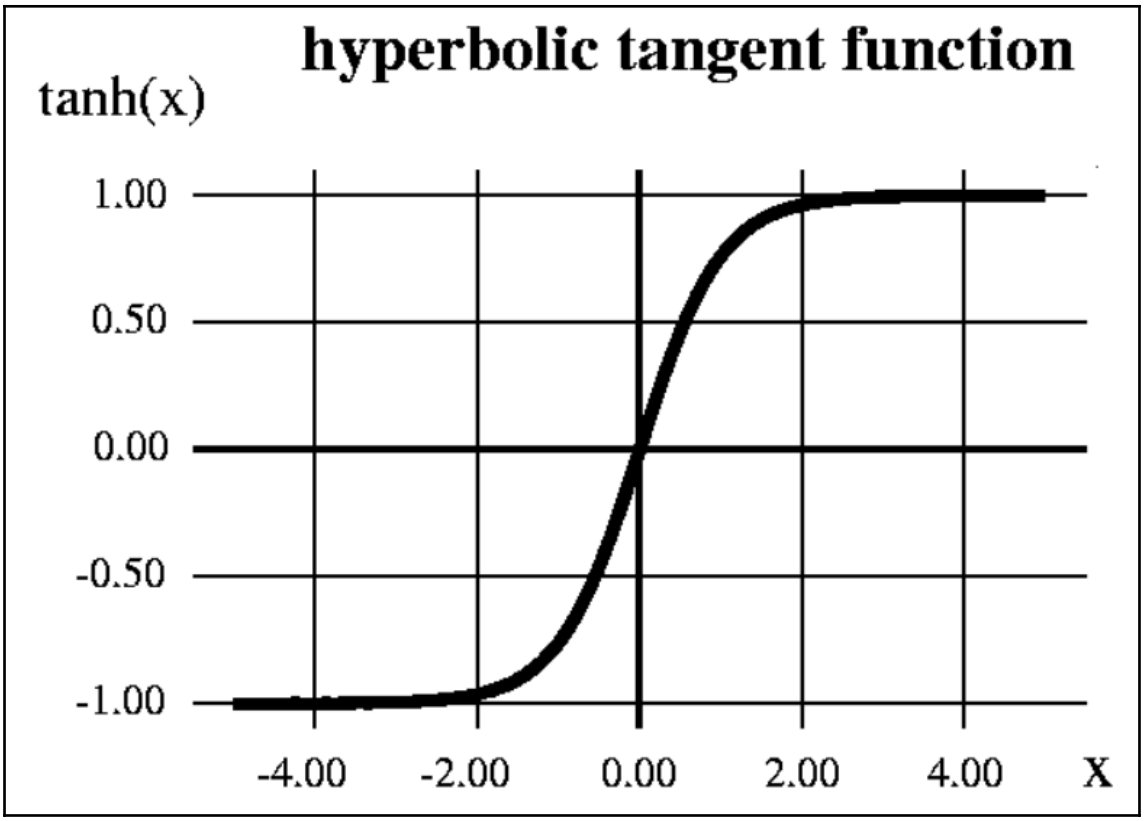
Chapter 4: Training Neural Networks

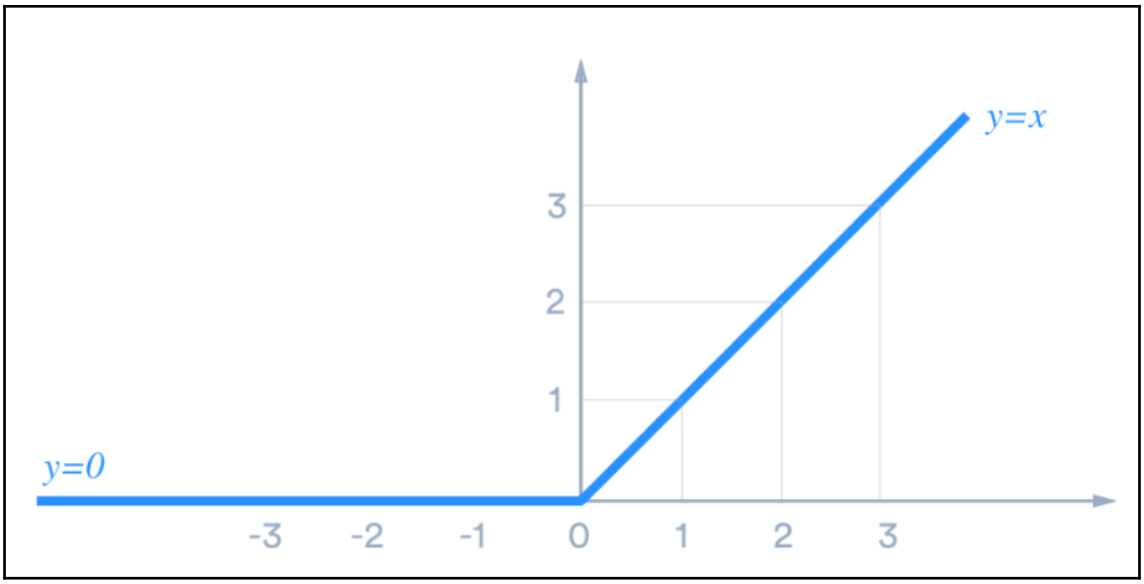


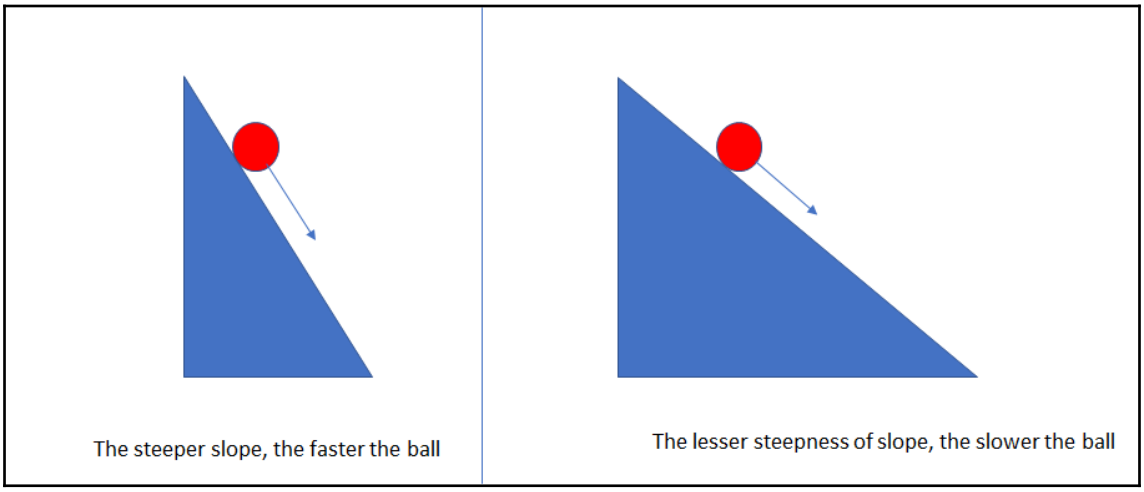

y
Increase
weight
here
Decrease
weight
here
4
3
2
1
Minima
1
2
3
4
w


I1
H1
O1
I2
H2
B1
B2


I1
w1
H1
w2
I2
w3
B1

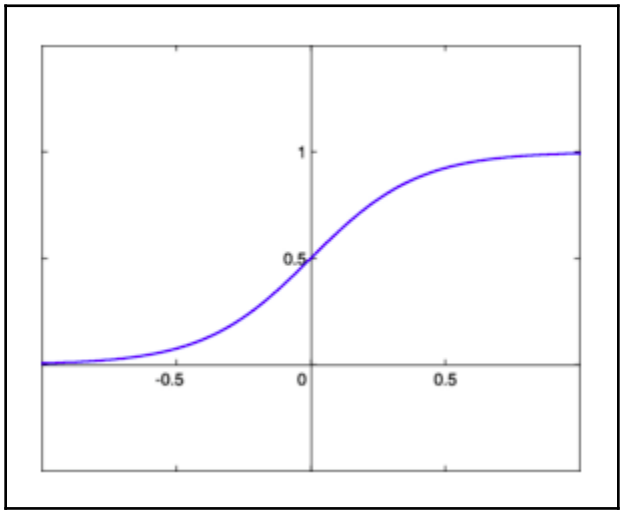

hyperbolic tangent function
tanh(x)
1.00
0.50
0.00
-0.50
-1.00
-4.00
-2.00
0.00
2.00
4.00
X

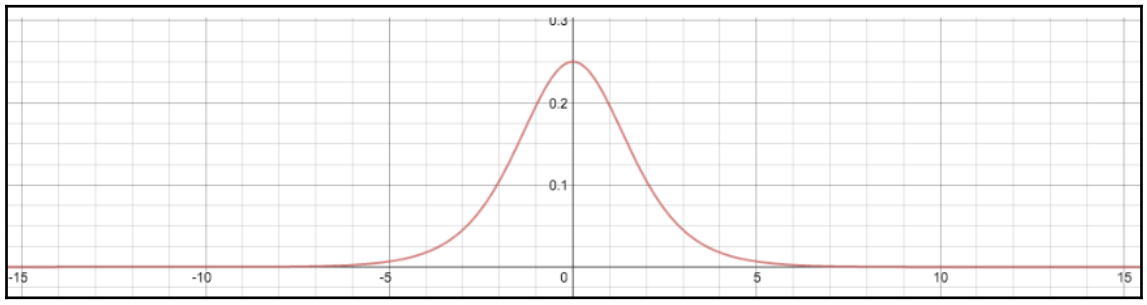

y=x
3
2
1
y=0
-3
-2
-1
0
1
2
3

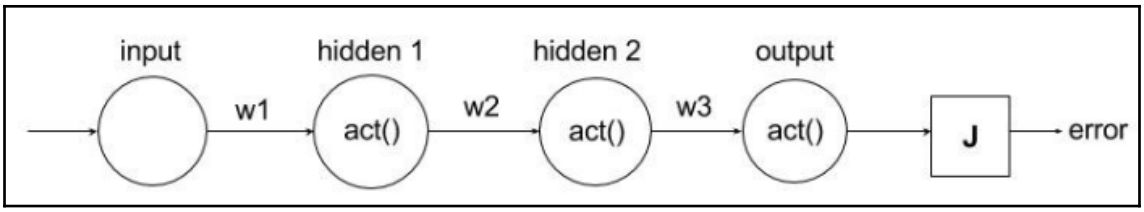

The steeper slope, the faster the ball
The lesser steepness of slope, the slower the ball

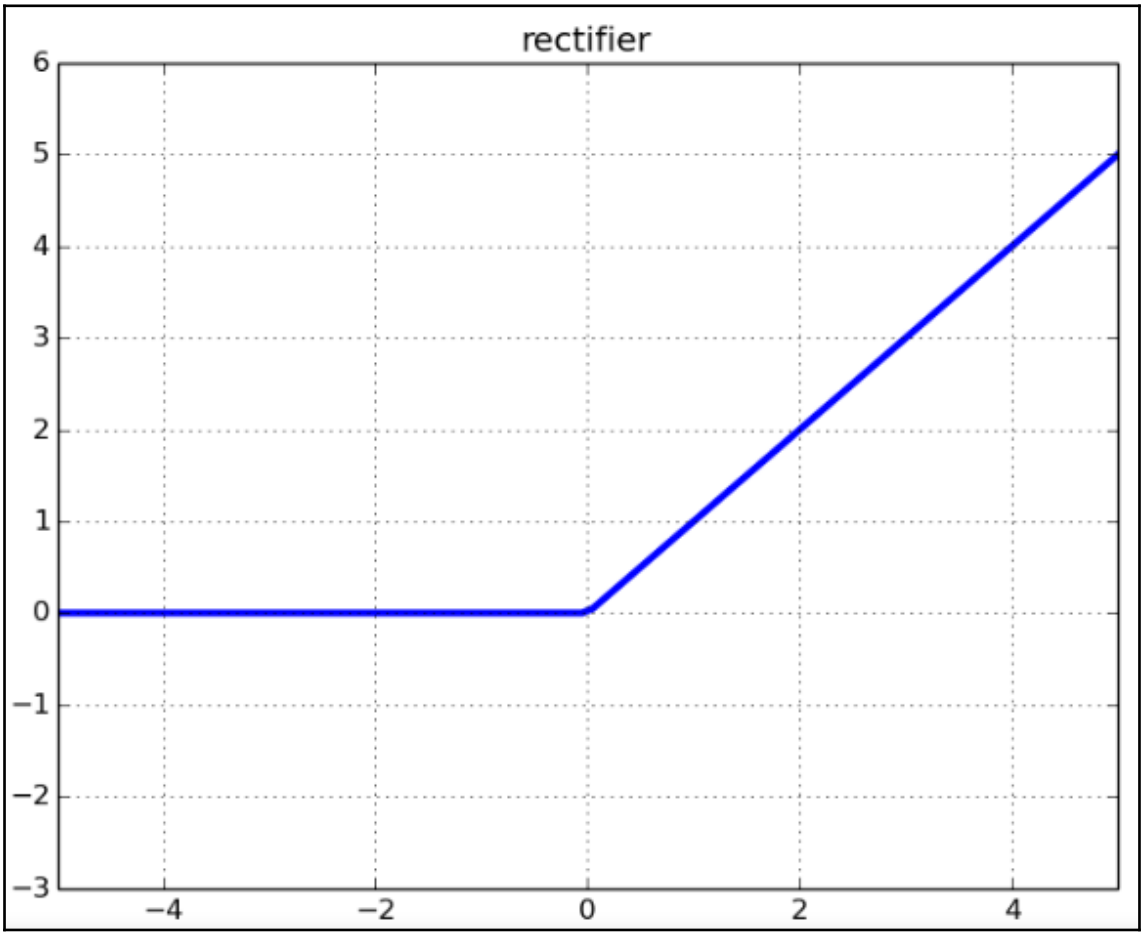
$$repeat\ \ until\ \ \frac{\partial J}{\partial W^{layer}_{ij}} \rightarrow 0:$$

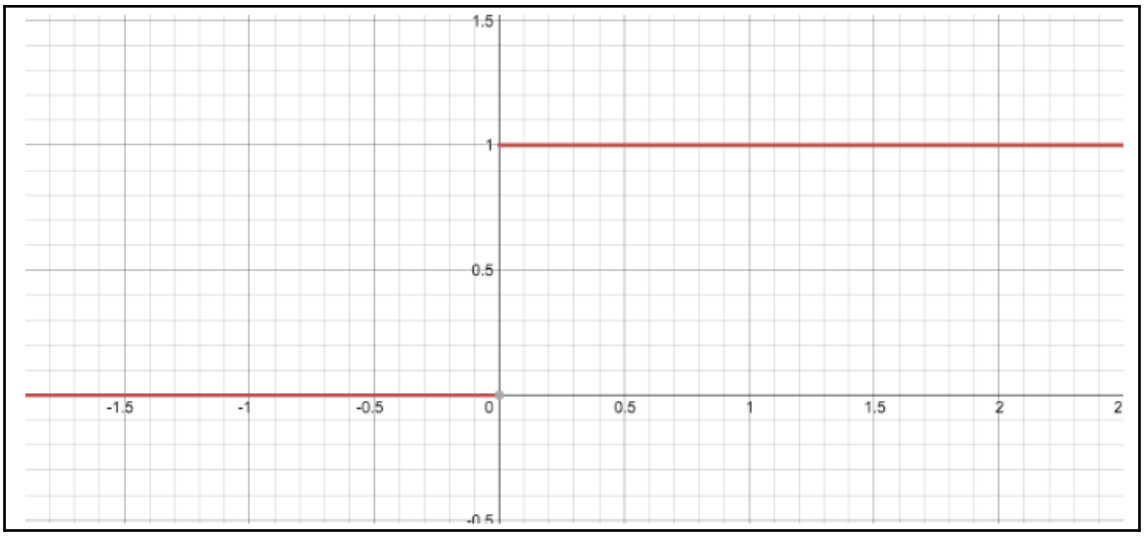
$$\{\quad W^{layer}_{ij} := W^{layer}_{ij} - \alpha \frac{\partial J}{\partial W^{layer}_{ij}}$$

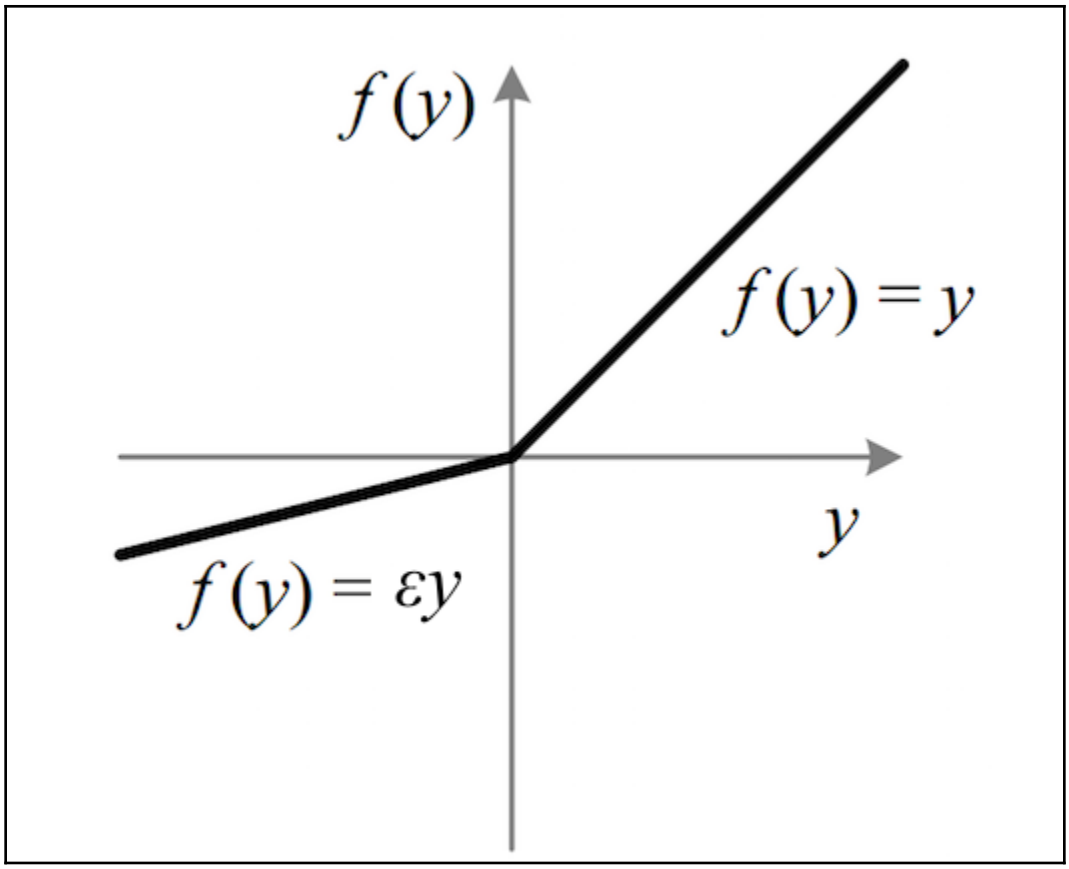


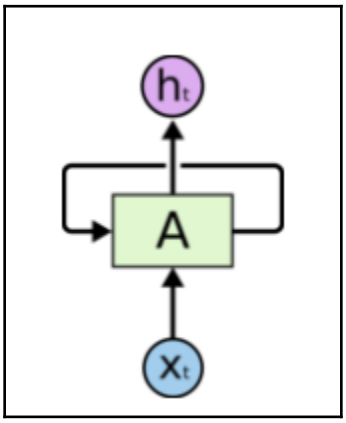


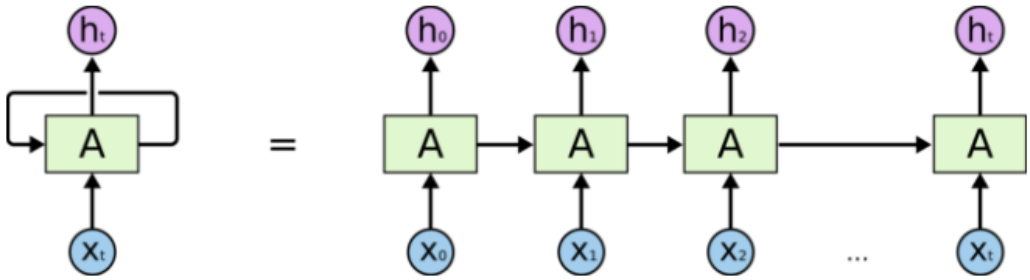

input
hidden 1
hidden 2
output
w1
act()
w2
act()
w3
act()
J
error


rectifier
6
5
4
3
2
1
0
−1
−2
−3
−4
−2
0
2
4

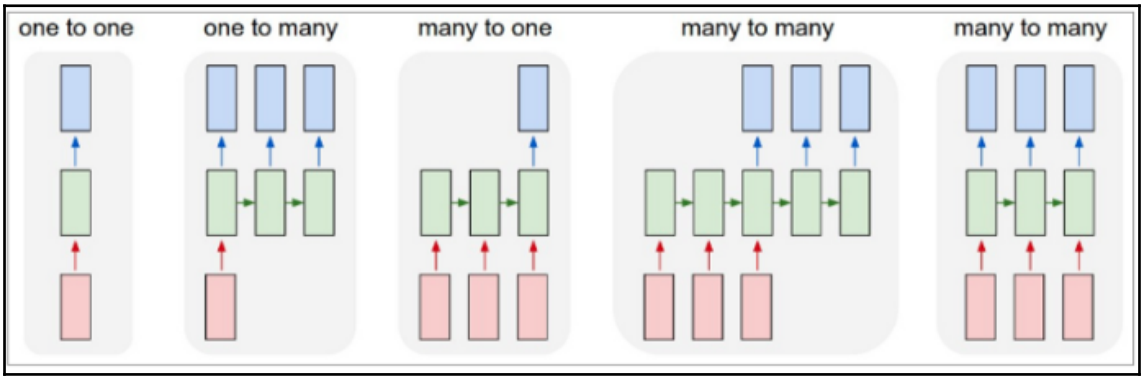

1.5
1
0.5
-1.5
-1
-0.5
0
0.5
1
1.5
2
2


f(y)
f(y) = y
y
f(y) = εy


ht
A
Xt

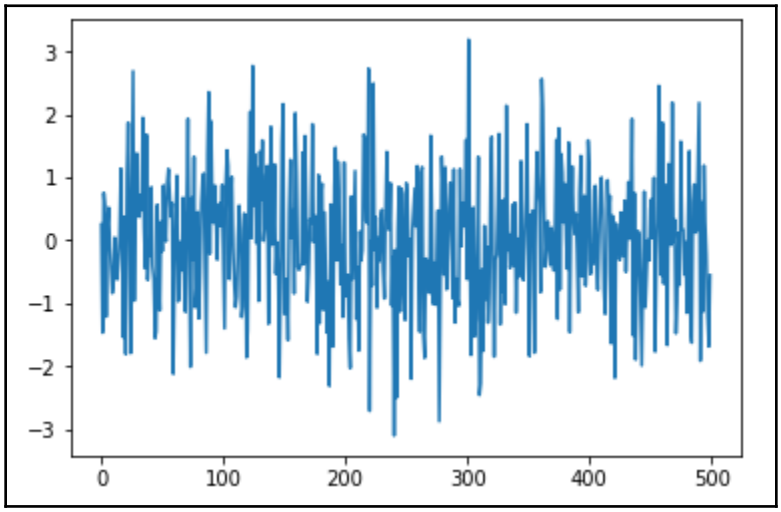


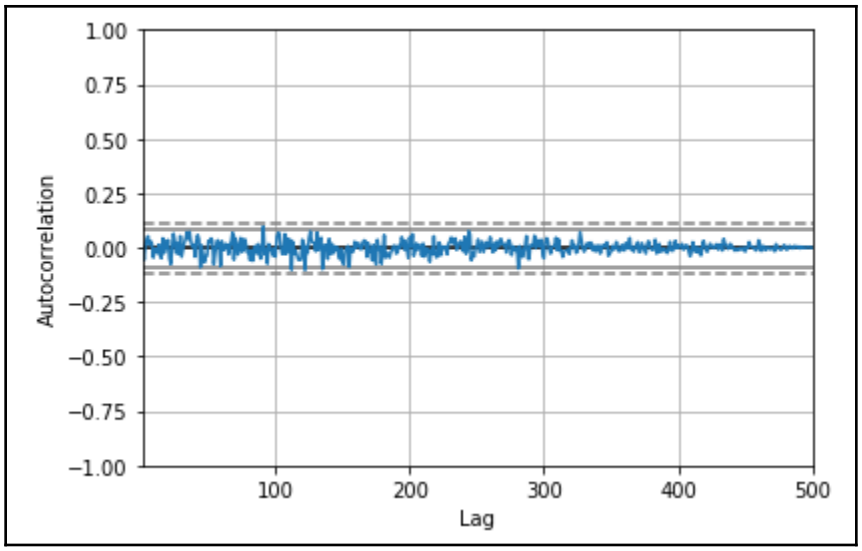
An unrolled recurrent neural network.

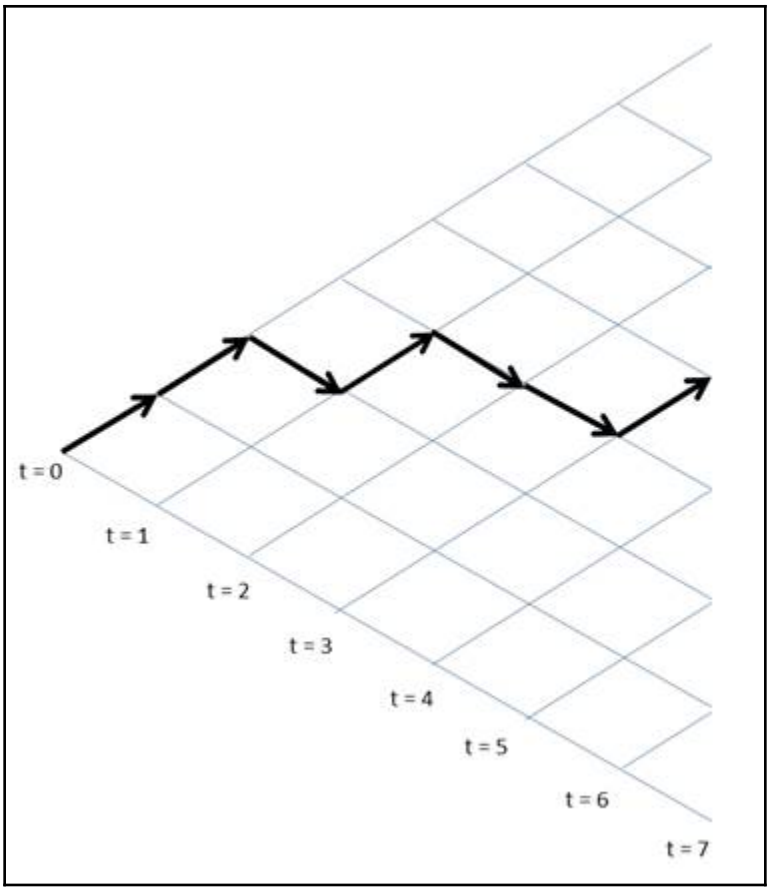


Chapter 5: Time Series Analysis

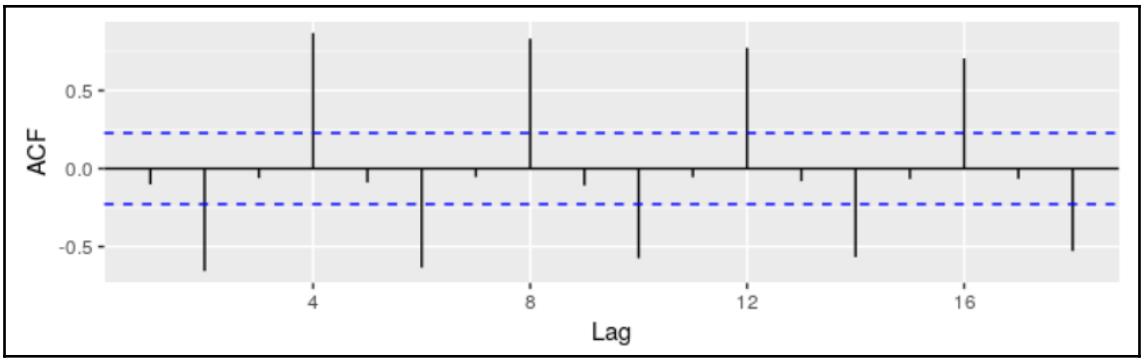
```
count    500.000000
mean       0.026039
std        1.050492
min       -3.102231
25%       -0.622977
50%        0.047703
75%        0.694352
max        3.192393
dtype: float64
```

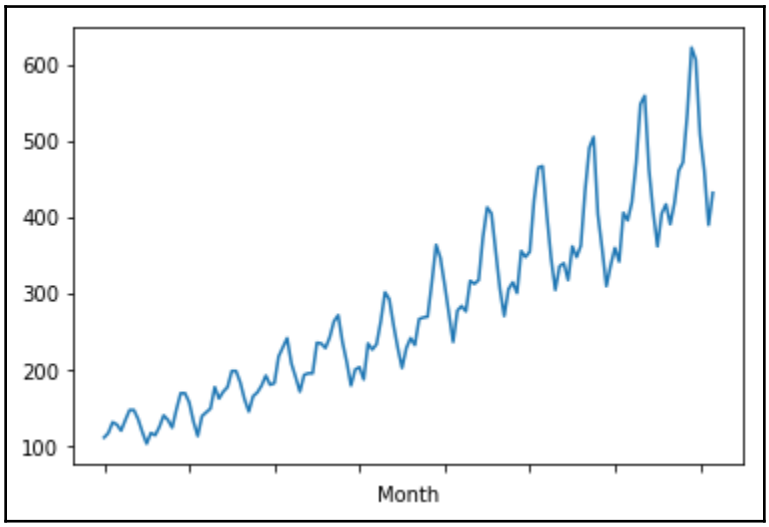



1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Autocorrelation
100
200
300
400
500
Lag



r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9
-0.102	-0.657	-0.060	0.869	-0.089	-0.635	-0.054	0.832	-0.108

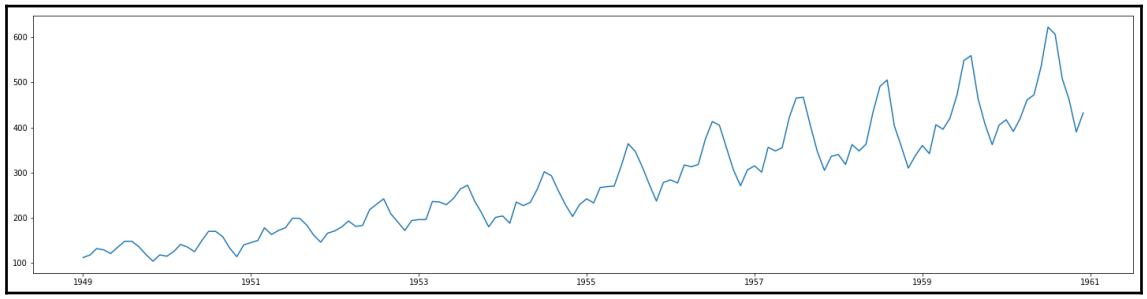




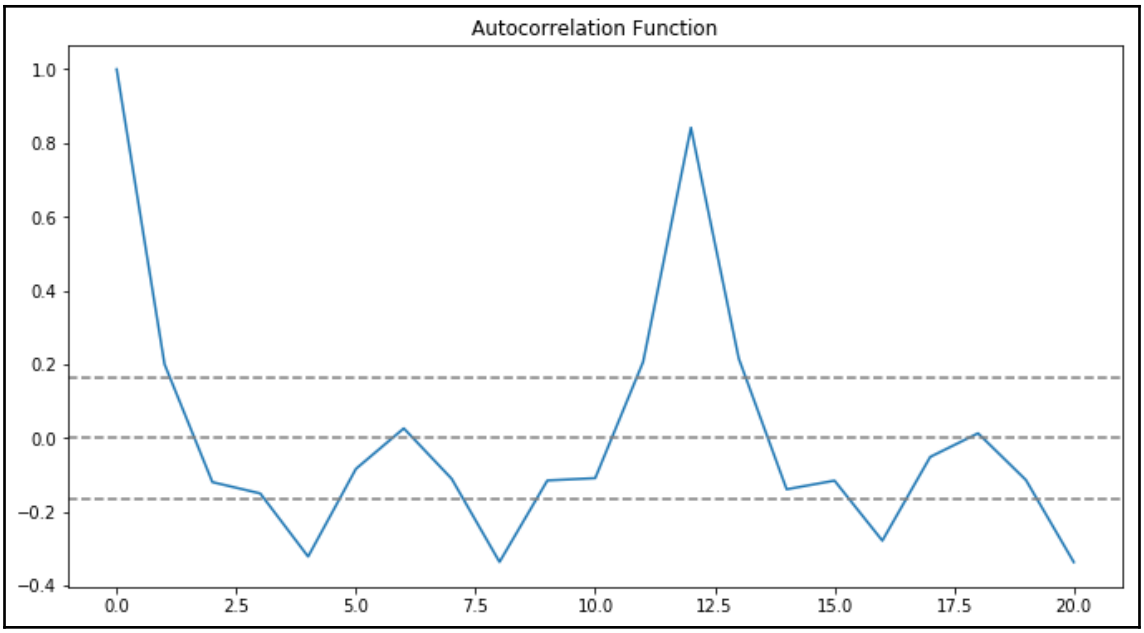

```
     Month  #Passengers
0  1949-01          112
1  1949-02          118
2  1949-03          132
3  1949-04          129
4  1949-05          121

 Data Types:
Month          object
#Passengers     int64
dtype: object
```

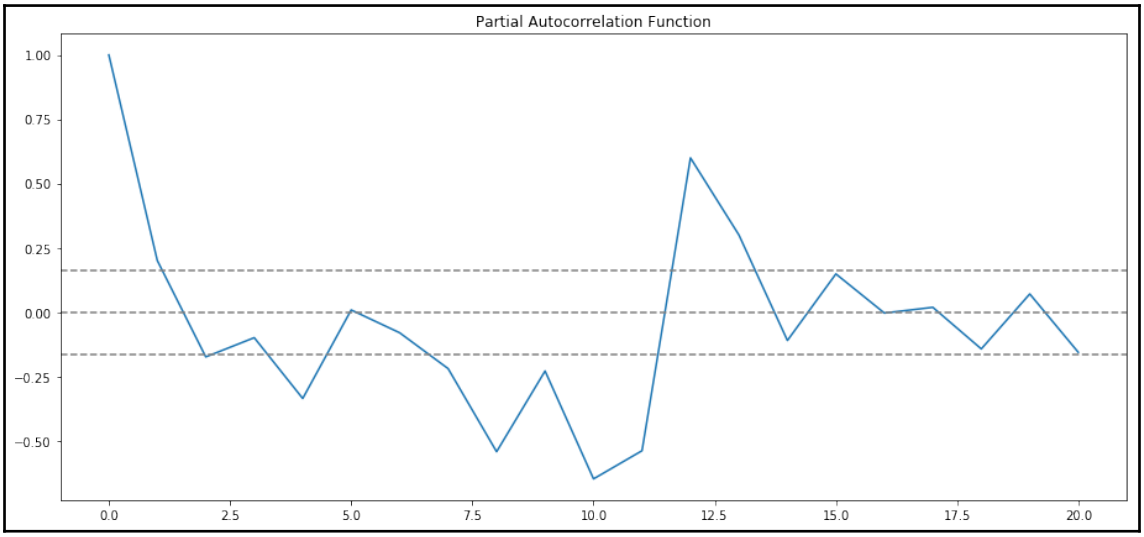
Month	#Passengers
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

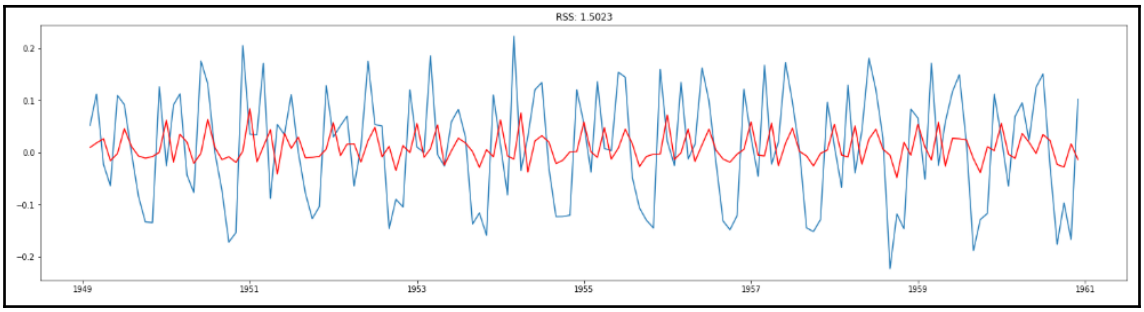
```
Month
1949-01-01    112
1949-02-01    118
1949-03-01    132
1949-04-01    129
1949-05-01    121
Name: #Passengers, dtype: int64
```




```
Test Statistic                  0.815369
p-value                         0.991880
#Lags Used                     13.000000
Number of Observations Used    130.000000
Critical Value (10%)            -2.578770
Critical Value (5%)             -2.884042
Critical Value (1%)             -3.481682
dtype: float64
```


Autocorrelation Function
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Partial Autocorrelation Function
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

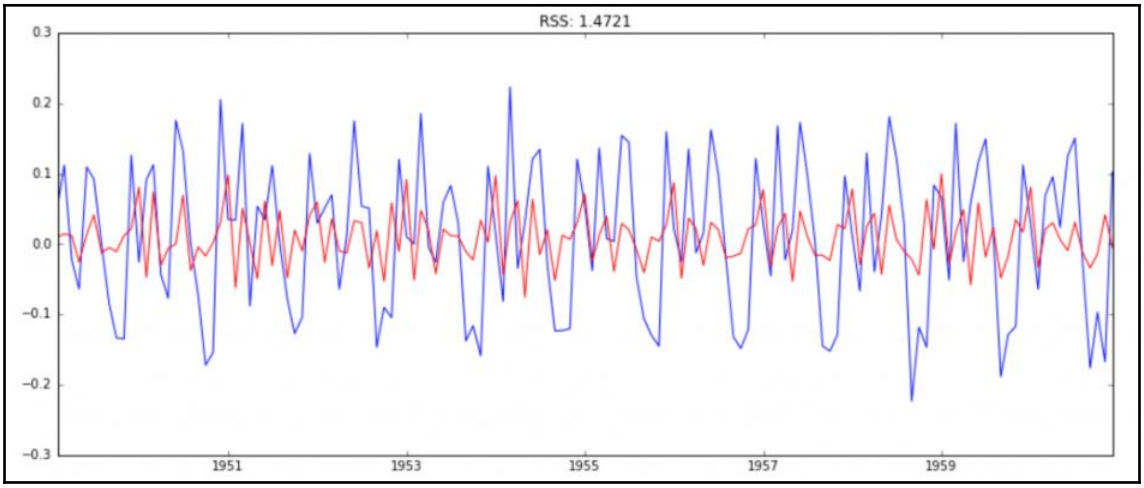


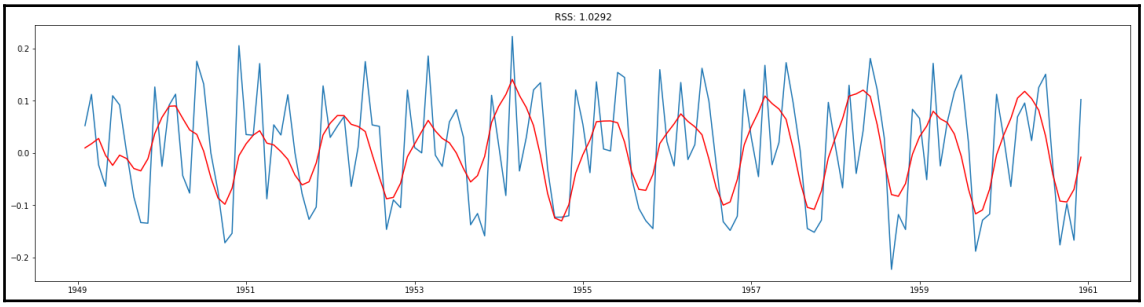
ARIMA Model Results

Dep. Variable:	D.#Passengers	**No. Observations:**	143
Model:	ARIMA(2, 1, 0)	**Log Likelihood**	122.802
Method:	css-mle	**S.D. of innovations**	0.102
Date:	Fri, 12 Oct 2018	**AIC**	-237.605
Time:	11:17:16	**BIC**	-225.753
Sample:	02-01-1949	**HQIC**	-232.789
	- 12-01-1960		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0096	0.009	1.048	0.296	-0.008	0.028
ar.L1.D.#Passengers	0.2359	0.083	2.855	0.005	0.074	0.398
ar.L2.D.#Passengers	-0.1725	0.083	-2.070	0.040	-0.336	-0.009

0.6838 -2.3088j 2.4079 -0.2042 0.6838 +2.3088j 2.4079 0.2042


RSS: 1.4721
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
1951
1953
1955
1957
1959


RSS: 1.0292
0.2
0.1
0.0
−0.1
−0.2
1949
1951
1953
1955
1957
1959
1961

ARIMA Model Results

Dep. Variable:	D.#Passengers	**No. Observations:**	143
Model:	ARIMA(2, 1, 2)	**Log Likelihood**	149.640
Method:	css-mle	**S.D. of innovations**	0.084
Date:	Sat, 13 Oct 2018	**AIC**	-287.281
Time:	07:05:22	**BIC**	-269.504
Sample:	02-01-1949	**HQIC**	-280.057
	- 12-01-1960		

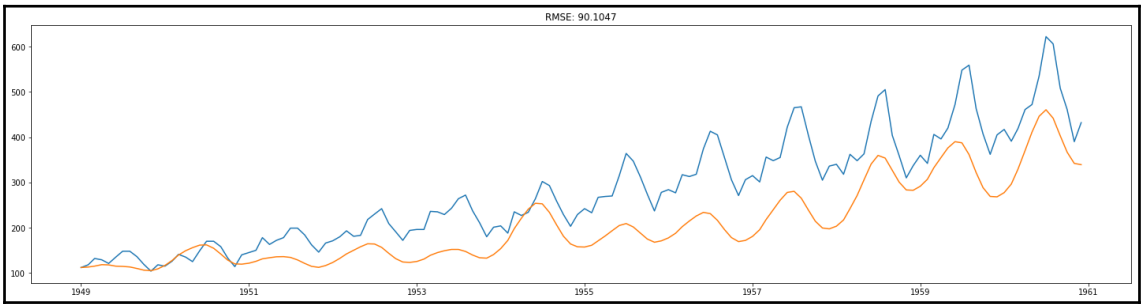
	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0096	0.003	3.697	0.000	0.005	0.015
ar.L1.D.#Passengers	1.6293	0.039	41.868	0.000	1.553	1.706
ar.L2.D.#Passengers	-0.8946	0.039	-23.127	0.000	-0.970	-0.819
ma.L1.D.#Passengers	-1.8270	0.036	-51.303	0.000	-1.897	-1.757
ma.L2.D.#Passengers	0.9245	0.036	25.568	0.000	0.854	0.995

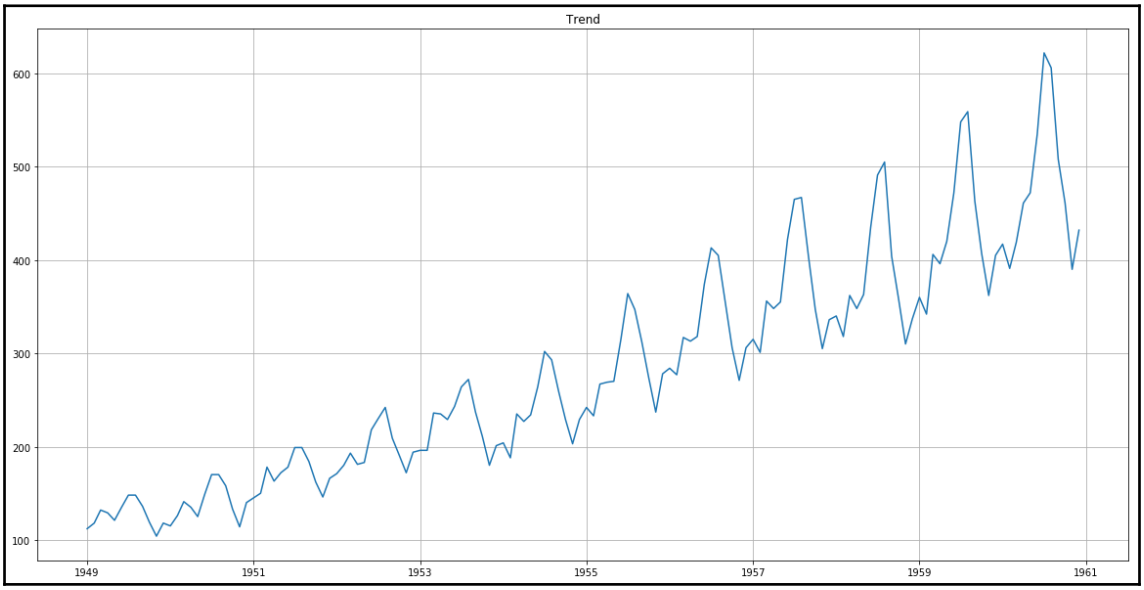
0.9106 -0.5372j 1.0573 -0.0848 0.9106 +0.5372j 1.0573 0.0848 0.9881 -0.3245j 1.0400 -0.0505 0.9881 +0.3245j 1.0400 0.0505

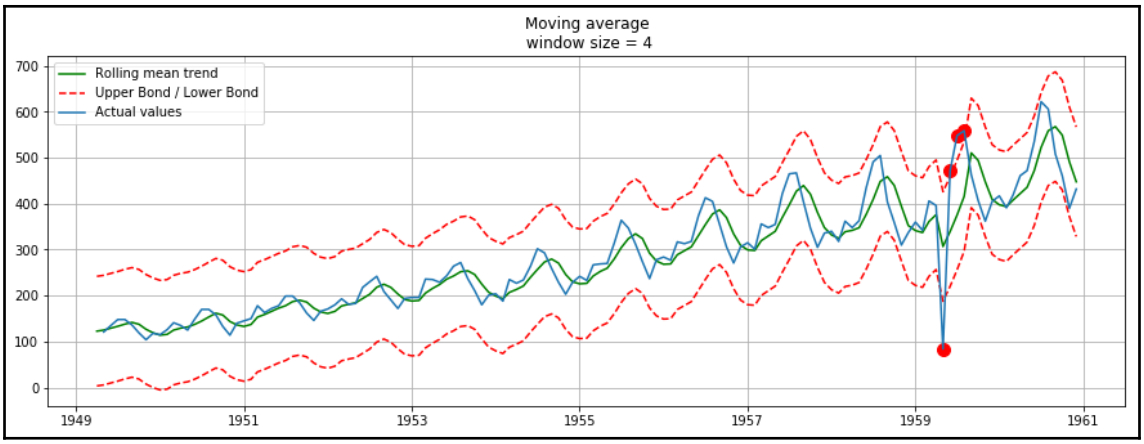
```
Month
1949-02-01    0.009580
1949-03-01    0.017491
1949-04-01    0.027670
1949-05-01   -0.004521
1949-06-01   -0.023890
dtype: float64
```

```
Month
1949-02-01    0.009580
1949-03-01    0.027071
1949-04-01    0.054742
1949-05-01    0.050221
1949-06-01    0.026331
dtype: float64
```

```
Month
1949-01-01    4.718499
1949-02-01    4.728079
1949-03-01    4.745570
1949-04-01    4.773241
1949-05-01    4.768720
dtype: float64
```






Moving average
window size = 4
Rolling mean trend
Upper Bond / Lower Bond
Actual values
700
600
500
400
300
200
100
0
1949
1951
1953
1955
1957
1959
1961

Chapter 6: Natural Language Processing

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=1,
        ngram_range=(1, 1), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

```
{'translation': 10, 'automatically': 1, 'another': 0, 'one': 6, 'machine': 5, 'human': 3, 'from': 2, 'language': 4, 'translat
e': 9, 'text': 7, 'to': 8}
```

```
<class 'scipy.sparse.csr.csr_matrix'>
[[1 1 1 1 1 1 1 2 1 1 1]]
```

```
['and', 'document', 'first', 'second', 'third']
(4, 5)
```

```
array([[ 0.        ,  0.55193942,  0.83388421,  0.        ,  0.        ],
       [ 0.        ,  0.55193942,  0.        ,  0.83388421,  0.        ],
       [ 0.        ,  0.46263733,  0.        ,  0.        ,  0.88654763],
       [ 0.6305035 ,  0.32902288,  0.4970962 ,  0.4970962 ,  0.        ]])
```

```
   Unnamed: 0  type                                             review label  \
0           0  test  Once again Mr. Costner has dragged out a movie...   neg
1           1  test  This is an example of why the majority of acti...   neg
2           2  test  First of all I hate those moronic rappers, who...   neg
3           3  test  Not even the Beatles could write songs everyon...   neg
4           4  test  Brass pictures (movies is not a fitting word f...   neg

          file
0      0_2.txt
1  10000_4.txt
2  10001_1.txt
3  10002_3.txt
4  10003_3.txt
(100000, 5)
```

```
(100000, 2)
```

label	review
neg	25000
pos	25000
unsup	50000

	review	label
0	Once again Mr. Costner has dragged out a movie...	neg
1	This is an example of why the majority of acti...	neg
2	First of all I hate those moronic rappers, who...	neg
3	Not even the Beatles could write songs everyon...	neg
4	Brass pictures (movies is not a fitting word f...	neg

	review	label
0	Once again Mr. Costner has dragged out a movie...	neg
1	This is an example of why the majority of acti...	neg
2	First of all I hate those moronic rappers, who...	neg
3	Not even the Beatles could write songs everyon...	neg
4	Brass pictures (movies is not a fitting word f...	neg

	review	label	Clean_review
0	Once again Mr. Costner has dragged out a movie...	0	Once again Mr Costner has dragged out a movie...
1	This is an example of why the majority of acti...	0	This is an example of why the majority of acti...
2	First of all I hate those moronic rappers, who...	0	First of all I hate those moronic rappers who...
3	Not even the Beatles could write songs everyon...	0	Not even the Beatles could write songs everyon...
4	Brass pictures (movies is not a fitting word f...	0	Brass pictures movies is not a fitting word f...

```
(50000, 3)
```

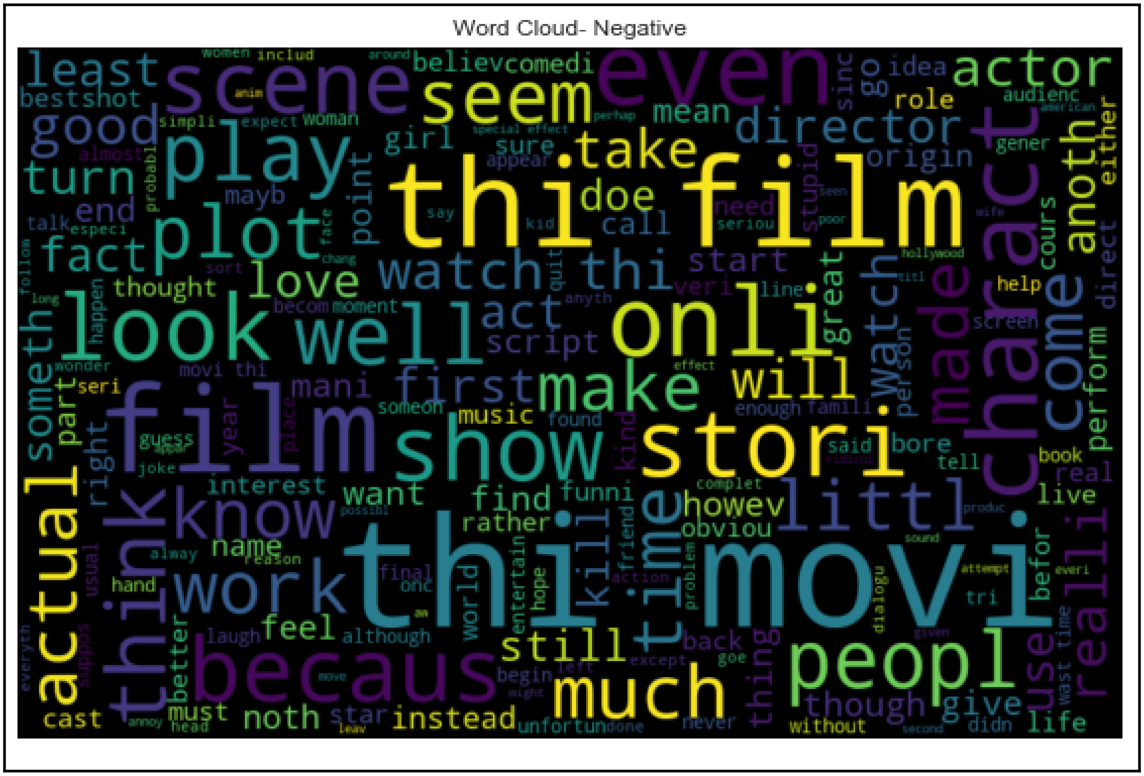
```
(50000,)
```

```
0    [onc, again, costner, drag, movi, longer, than...
1    [thi, exampl, major, action, film, same, gener...
2    [first, hate, those, moron, rapper, could, the...
3    [even, beatl, could, write, song, everyon, lik...
4    [brass, pictur, movi, fit, word, them, realli,...
Name: Clean_review, dtype: object
```

```
0    onc again costner drag movi longer than necess...
1    thi exampl major action film same gener bore t...
2    first hate those moron rapper could they press...
3    even beatl could write song everyon like altho...
4    brass pictur movi fit word them realli somewha...
Name: Clean_review, dtype: object
```

	review	label	Clean_review	Clean_review2
0	Once again Mr. Costner has dragged out a movie...	0	Once again Costner dragged movie longer than n...	onc again costner drag movi longer than necess...
1	This is an example of why the majority of acti...	0	This example majority action films same Generi...	thi exampl major action film same gener bore t...
2	First of all I hate those moronic rappers, who...	0	First hate those moronic rappers could they pr...	first hate those moron rapper could they press...
3	Not even the Beatles could write songs everyon...	0	even Beatles could write songs everyone liked ...	even beatl could write song everyon like altho...
4	Brass pictures (movies is not a fitting word f...	0	Brass pictures movies fitting word them really...	brass pictur movi fit word them realli somewha...


thi movi
thi film
charact
show
scene
work
onli
look
play
stori
even
well
peopl
becau
think
come
time
littl
film
much
actual
watch thi
love
seem
plot
first
made
know
howev
life
use
make
anoth
fact
turn
director
music
world
start
name
interest
direct
find
end
mani
good
act
real
will
take
believ
part
shot
tell
perform
point
wonder
still
though
call
give
cours
thing
right
place
appear
great
doe
origin
friend
found
someth
least
entertain
instead
comedi
young
watch
feel
befor
year
person
back
final
actor
need
girl
father
effect
usual
idea
famili
sure
audienc
happen
possibl
sort
thi show
realli
face
role
best
onc
becom
moment
woman
funni
screen
must
rather
although
includ
kill
either
said
star
noth
follow
want
seri
alway
begin
almost
done
kind
script
better
live
mean
movi thi
complet
especi
interest


Word Cloud- Negative
thi movi
thi film
film
even
scene
charact
onli
stori
show
look
well
plot
play
seem
make
think
becaus
much
peopl
actual
littl
know
work
time
watch thi
made
take
director
actor
good
least
turn
fact
love
first
act
will
come
anoth
realli
still
script
want
find
feel
instead
noth
though
give
use
interest
mani
doe
call
start
mean
origin
idea
role
best
shot
girl
point
watch
great
end
someth
right
part
better
must
cast
name
kill
world
music
rather
funni
howev
obviou
bore
said
tell
live
book
real
befor
perform
direct
cours
either
help
screen
life
without
never
back
thing
begin
although
laugh
hand
guess
year
seri
thought
maybe
sure
appear
believ
comedi
includ
woman
simpli
expect
audienc
gener
sinc
go
stupid
need
seriou
veri
line
enough
famili
person
someon
found
tri
wast time
didn


Word Cloud-Positive
thi film
thi movi
show
scene
work
onli
charact
stori
even
well
peopl
know
look
play
love
life
made
watch thi
much
actual
come
think
becau
time
plot
first
littl
film
use
make
anoth
fact
origin
howev
though
seem
turn
perform
still
director
music
world
start
interest
direct
part
good
take
will
believ
real
realli
act
becom
face
role
best
onc
friend
found
wonder
point
script
great
right
place
appear
call
cours
give
mani
end
find
want
instead
comedi
watch
feel
befor
actor
year
person
back
name
need
go
idea
famili
sure
audienc
happen
possibl

```
{'mind': 565,
 'self': 764,
 'receiv': 702,
 'festiv': 343,
 'univers': 932,
 'deserv': 235,
 'remind': 715,
 'viewer': 945,
 'certainli': 128,
 'break': 101,
 'boy': 98,
 'moral': 575,
 'battl': 74,
 'book': 94,
 'produc': 675,
 'mood': 574,
 'befor': 80,
 'shot': 781,
 'doubt': 261,
```

```
F1 Score- 0.855267194306
Accuracy- 0.842733333333
```

```
F1 Score- 0.853920386007
Accuracy- 0.838533333333
```

	W1	W2	Wn
D1	0	1	2
D2	1	1	0
D3	2	0	1
Dm	1	1	0

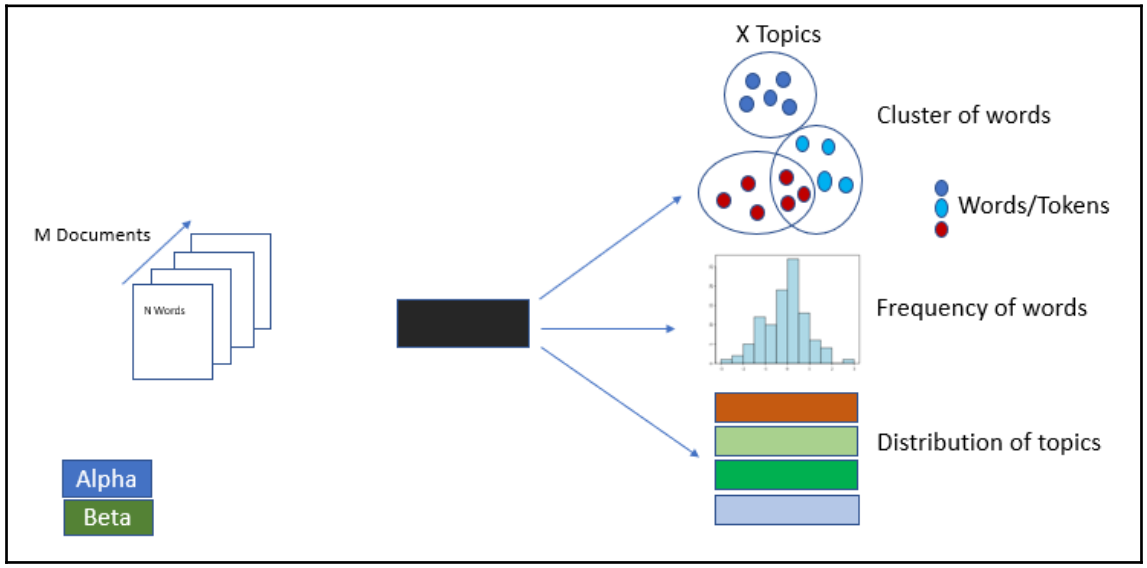
M x N matrix

	T1	T2	Tx
D1	1	1	1
D2	0	1	1
D3	2	0	0
Dm	0	1	1

M x X matrix of Documents & Topics

	W1	W2	Wn
T1	0	1	1
T2	1	0	1
T3	2	1	0
Tx	0	1	1

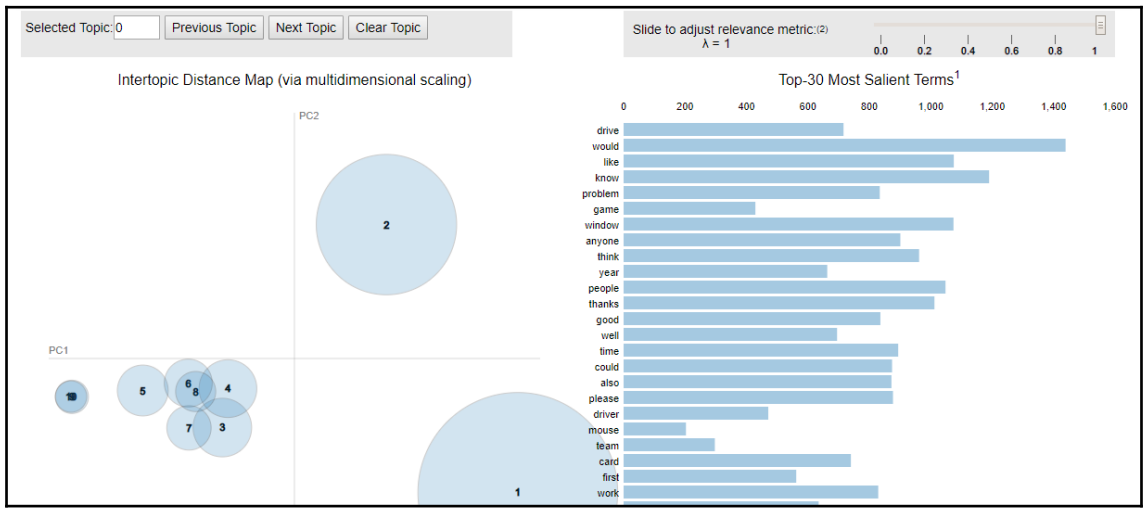
X x N matrix of Topics & Words

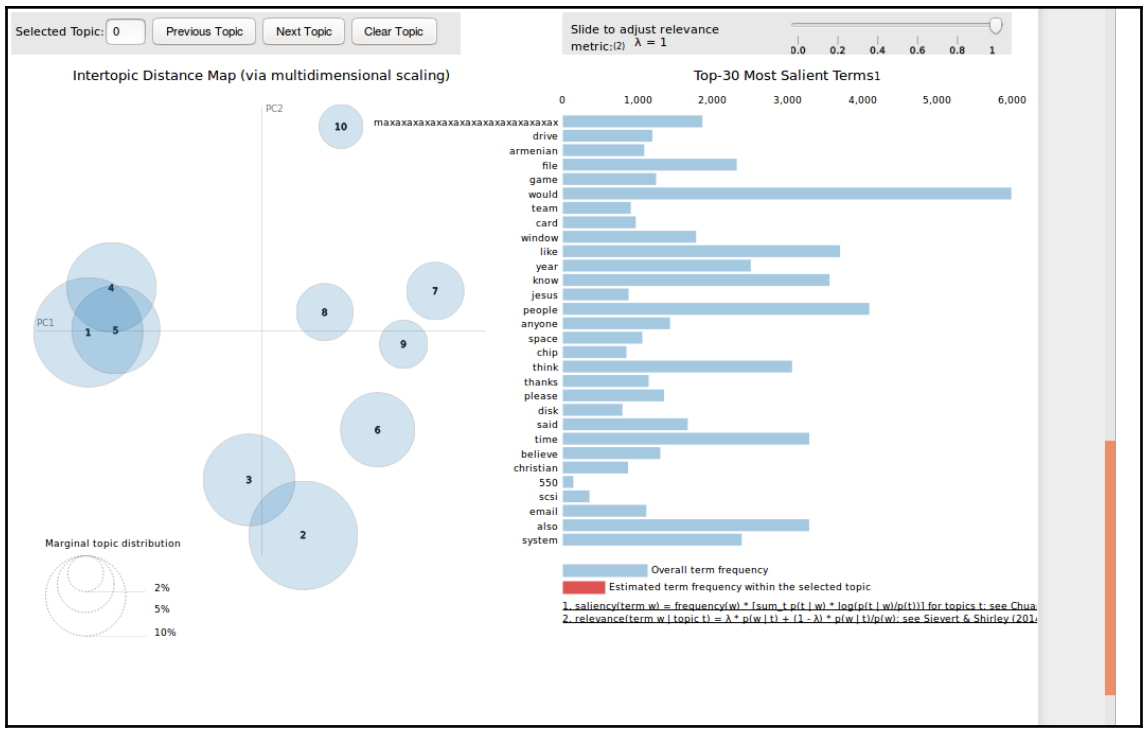



```
[(1, '0.001*"game" + 0.001*"team" + 0.001*"player" + 0.001*"play" + 0.001*"season"'), (3, '0.001*"people" + 0.001*"would" + 0.001*"think" + 0.001*"know" + 0.001*"thing"'),
(4, '0.001*"drive" + 0.001*"window" + 0.001*"problem" + 0.001*"maxaxaxaxaxaxaxaxaxaxaxaxax" + 0.001*"game"'), (2, '0.001*"mouse" + 0.001*"anyone" + 0.001*"know" + 0.001
*"would" + 0.001*"thanks"'), (6, '0.001*"would" + 0.000*"game" + 0.000*"year" + 0.000*"like" + 0.000*"think"')]
```

```
[(8, '0.007*"would" + 0.005*"think" + 0.004*"people" + 0.003*"like" + 0.003*"file"'), (7, '0.008*"file" + 0.006*"entry" + 0.003*"program" + 0.003*"time" + 0.002*"email"'),
(9, '0.034*"maxaxaxaxaxaxaxaxaxaxaxaxaxax" + 0.003*"year" + 0.003*"also" + 0.003*"would" + 0.002*"space"'), (6, '0.005*"would" + 0.004*"window" + 0.004*"time" + 0.004*"lik
e" + 0.003*"file"'), (5, '0.006*"game" + 0.006*"team" + 0.004*"would" + 0.003*"play" + 0.003*"year"')]
```

```
lda_model 1- Perplexity:- -12.8535943147
lda_model 2- Perplexity:- -9.33637689865
```

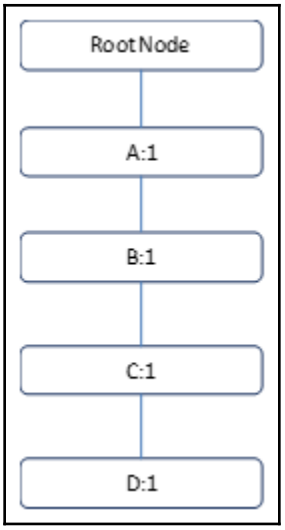


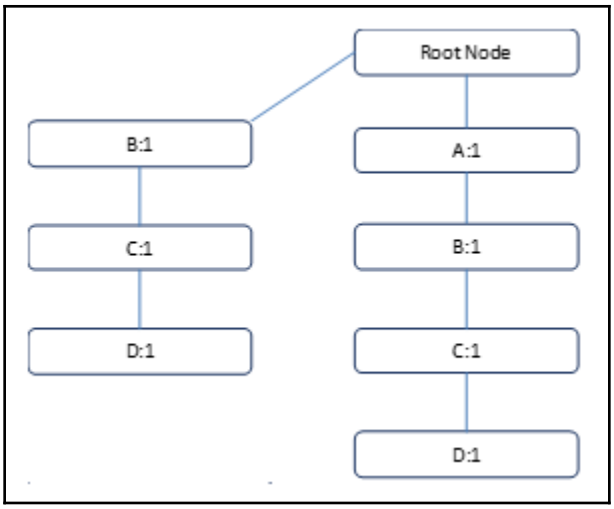


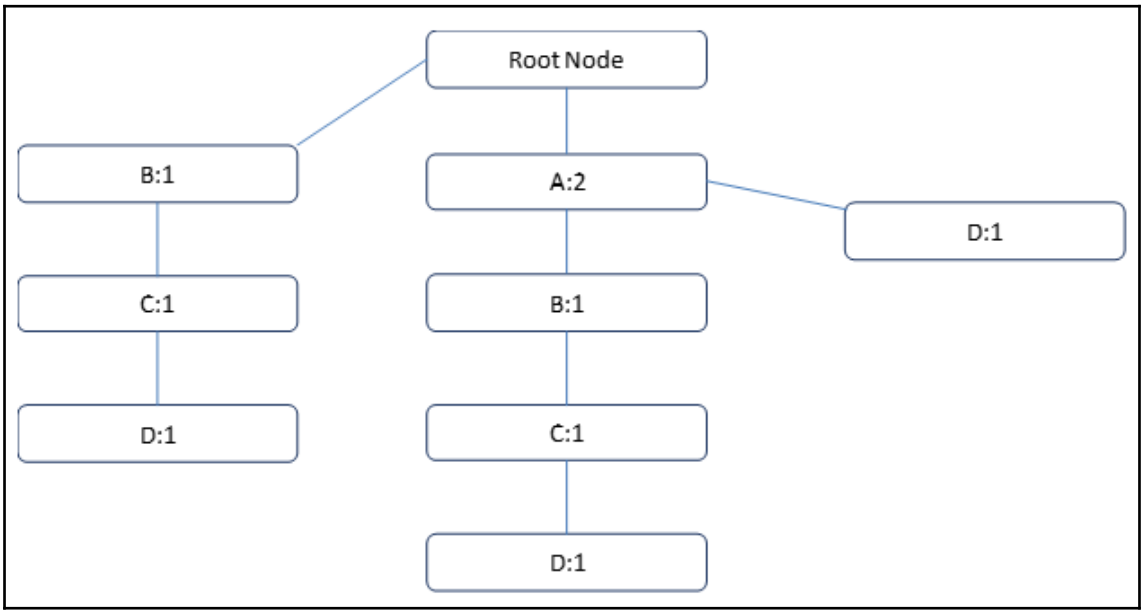
```
F1 Score- 0.771008403361
Accuracy- 0.709333333333
```

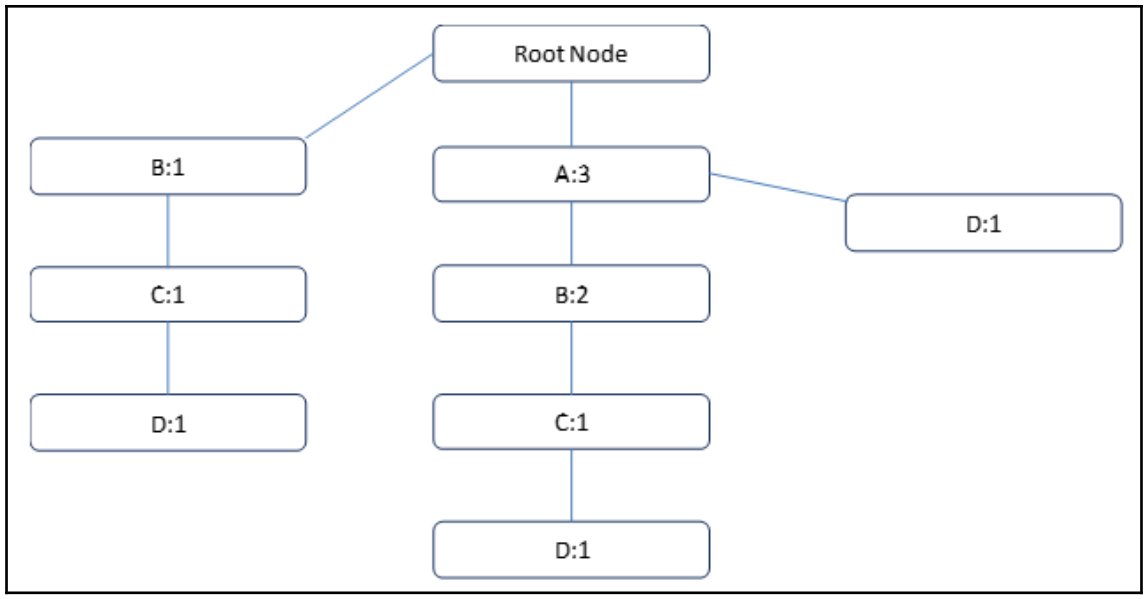
Chapter 7: Temporal and Sequential Pattern Discovery

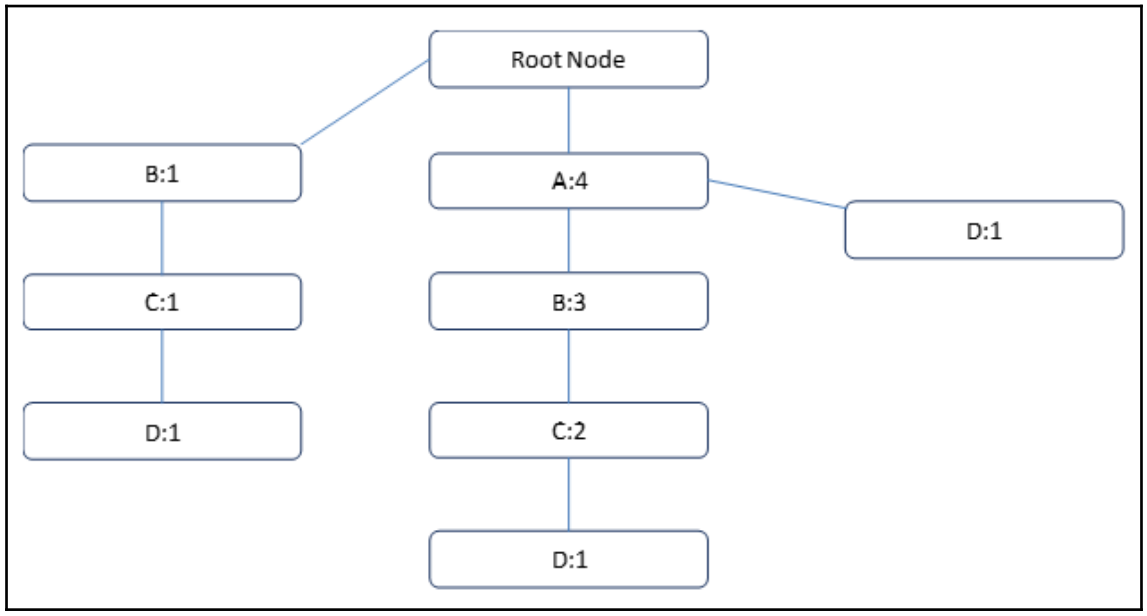
```
[RelationRecord(items=frozenset({'chicken', 'light cream'}), support=0.004532728969470737, ordered_statistics=[OrderedS
tatistic(items_base=frozenset({'light cream'}), items_add=frozenset({'chicken'}), confidence=0.29059829059829057, lift=
4.84395061728395)]),
 RelationRecord(items=frozenset({'escalope', 'mushroom cream sauce'}), support=0.005732568990801226, ordered_statistics
=[OrderedStatistic(items_base=frozenset({'mushroom cream sauce'}), items_add=frozenset({'escalope'}), confidence=0.3006
993006993007, lift=3.790832696715049)]),
 RelationRecord(items=frozenset({'escalope', 'pasta'}), support=0.005865884548726837, ordered_statistics=[OrderedStatis
tic(items_base=frozenset({'pasta'}), items_add=frozenset({'escalope'}), confidence=0.3728813559322034, lift=4.700811850
163794)]),
 RelationRecord(items=frozenset({'fromage blanc', 'honey'}), support=0.003332888948140248, ordered_statistics=[OrderedS
tatistic(items_base=frozenset({'fromage blanc'}), items_add=frozenset({'honey'}), confidence=0.2450980392156863, lift=
5.164270764485569)]),
 RelationRecord(items=frozenset({'ground beef', 'herb & pepper'}), support=0.015997866951073192, ordered_statistics=[Or
deredStatistic(items_base=frozenset({'herb & pepper'}), items_add=frozenset({'ground beef'}), confidence=0.323450134770
8895, lift=3.2919938411349285)]),
 RelationRecord(items=frozenset({'ground beef', 'tomato sauce'}), support=0.005332622317024397, ordered_statistics=[Ord
eredStatistic(items_base=frozenset({'tomato sauce'}), items_add=frozenset({'ground beef'}), confidence=0.37735849056603
77, lift=3.840659481324083)]),
 RelationRecord(items=frozenset({'olive oil', 'light cream'}), support=0.003199573390214638, ordered_statistics=[Ordere
dStatistic(items_base=frozenset({'light cream'}), items_add=frozenset({'olive oil'}), confidence=0.20512820512820515, l
ift=3.1147098515519573)]),
 RelationRecord(items=frozenset({'whole wheat pasta', 'olive oil'}), support=0.007998933475536596, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'whole wheat pasta'}), items_add=frozenset({'olive oil'}), confidence=0.2714932
126696833, lift=4.122410097642296)]),
 RelationRecord(items=frozenset({'pasta', 'shrimp'}), support=0.005065991201173177, ordered_statistics=[OrderedStatisti
c(items_base=frozenset({'pasta'}), items_add=frozenset({'shrimp'}), confidence=0.3220338983050847, lift=4.5066721477358
96)]),
 RelationRecord(items=frozenset({'milk', 'spaghetti', 'avocado'}), support=0.003332888948140248, ordered_statistics=[Or
deredStatistic(items_base=frozenset({'spaghetti', 'avocado'}), items_add=frozenset({'milk'}), confidence=0.416666666666
66663, lift=3.215449245541838)]),
```

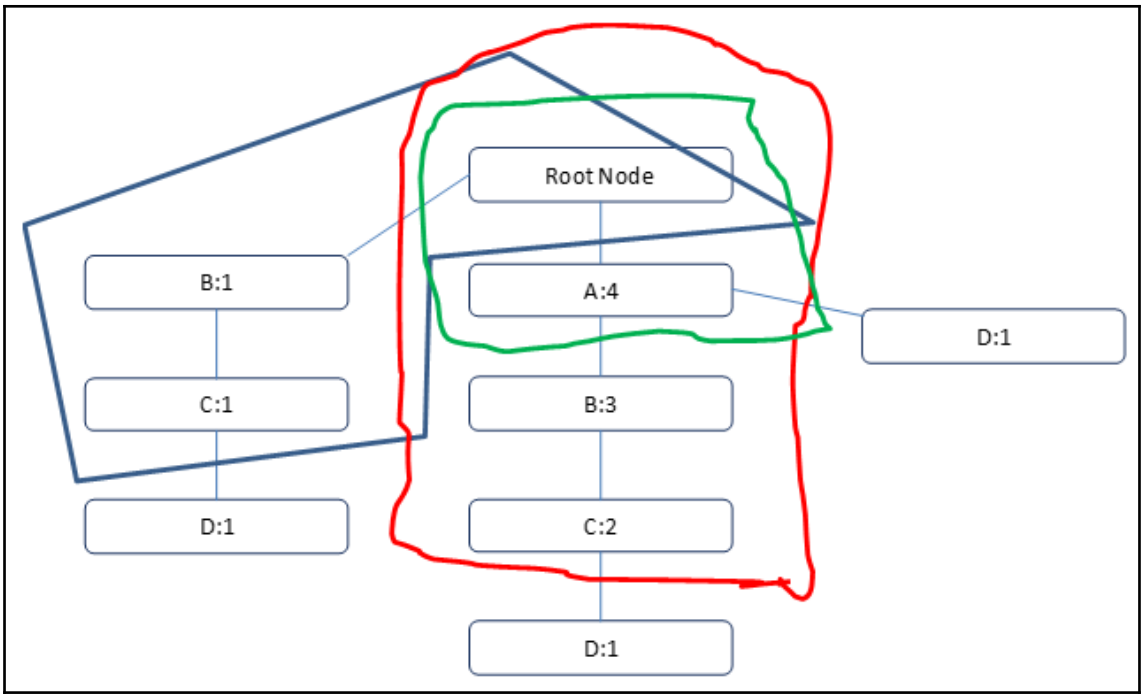


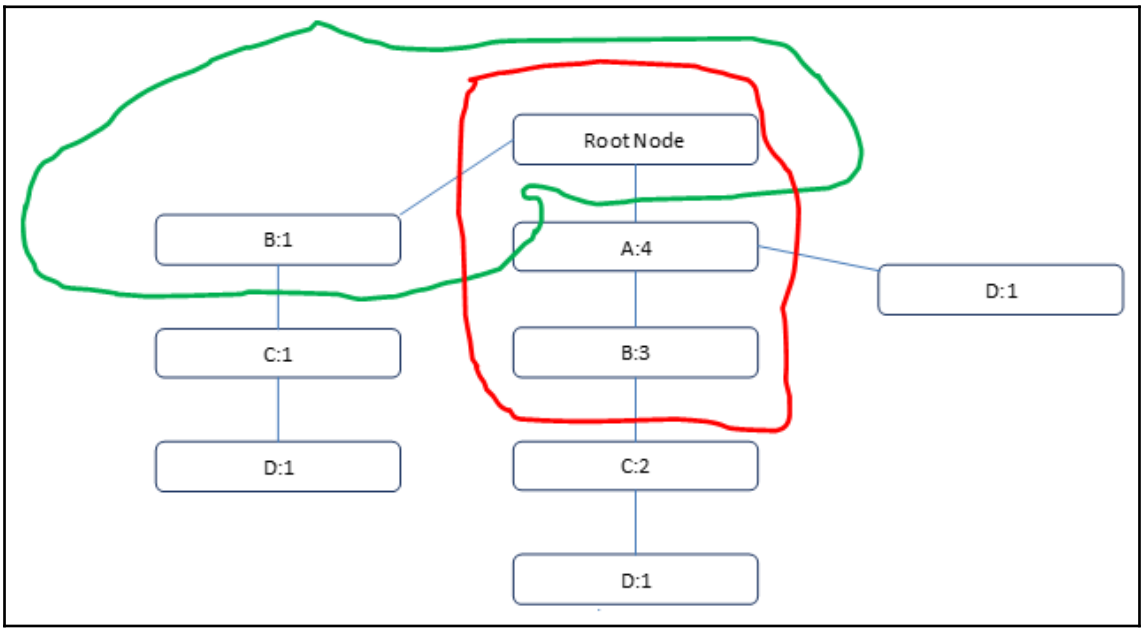

Root Node
B:1
A:1
C:1
B:1
D:1
C:1
D:1

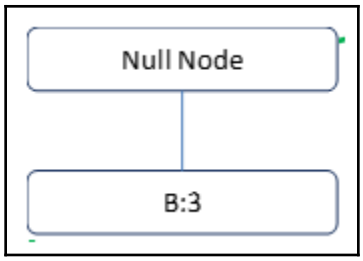

Root Node
B:1
A:2
D:1
C:1
B:1
D:1
C:1
D:1


Root Node
B:1
A:3
D:1
C:1
B:2
D:1
C:1
D:1


Root Node
B:1
A:4
D:1
C:1
B:3
D:1
C:2
D:1

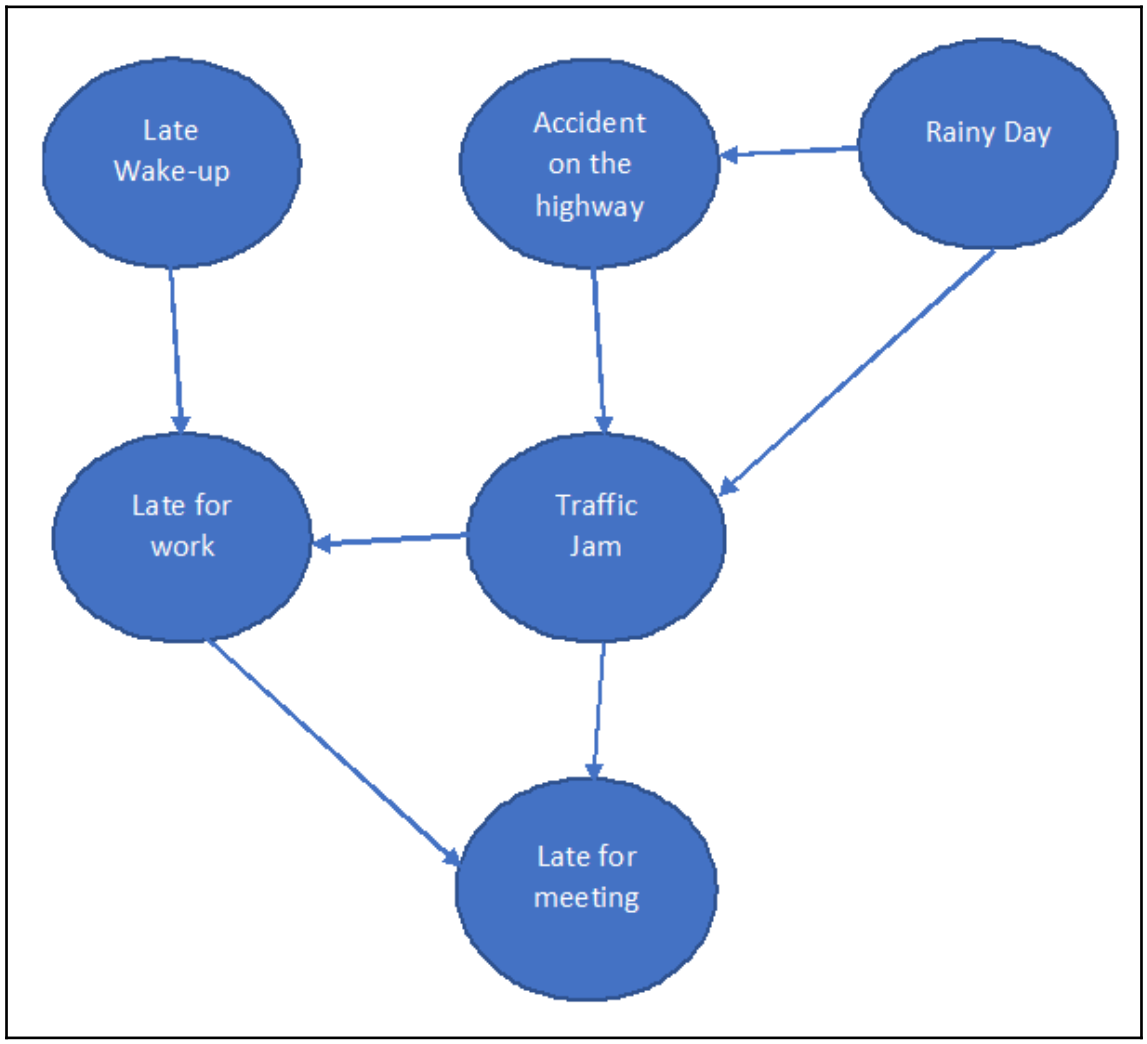

Root Node
B:1
A:4
D:1
C:1
B:3
D:1
C:2
D:1

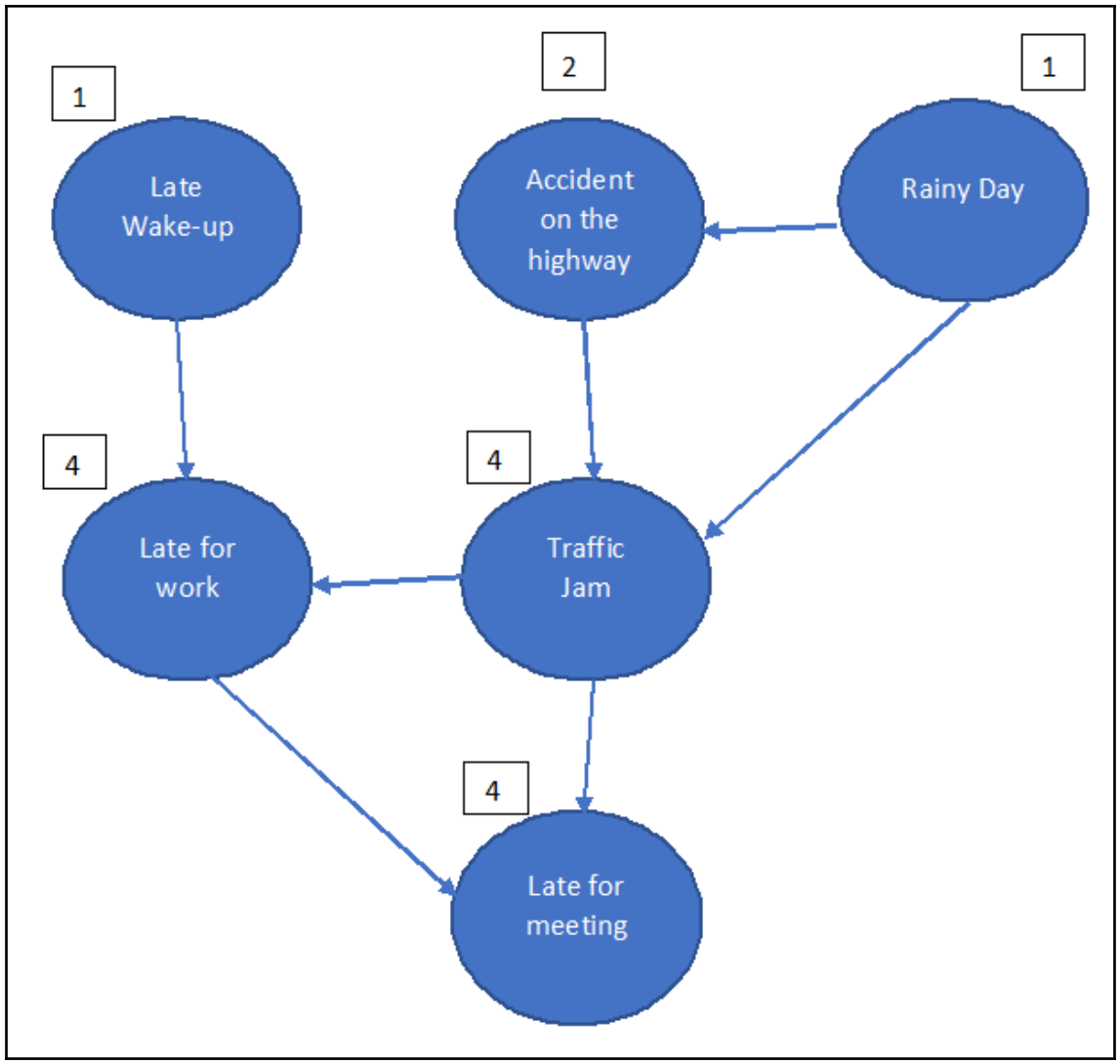

Root Node
B:1
A:4
D:1
C:1
B:3
D:1
C:2
D:1

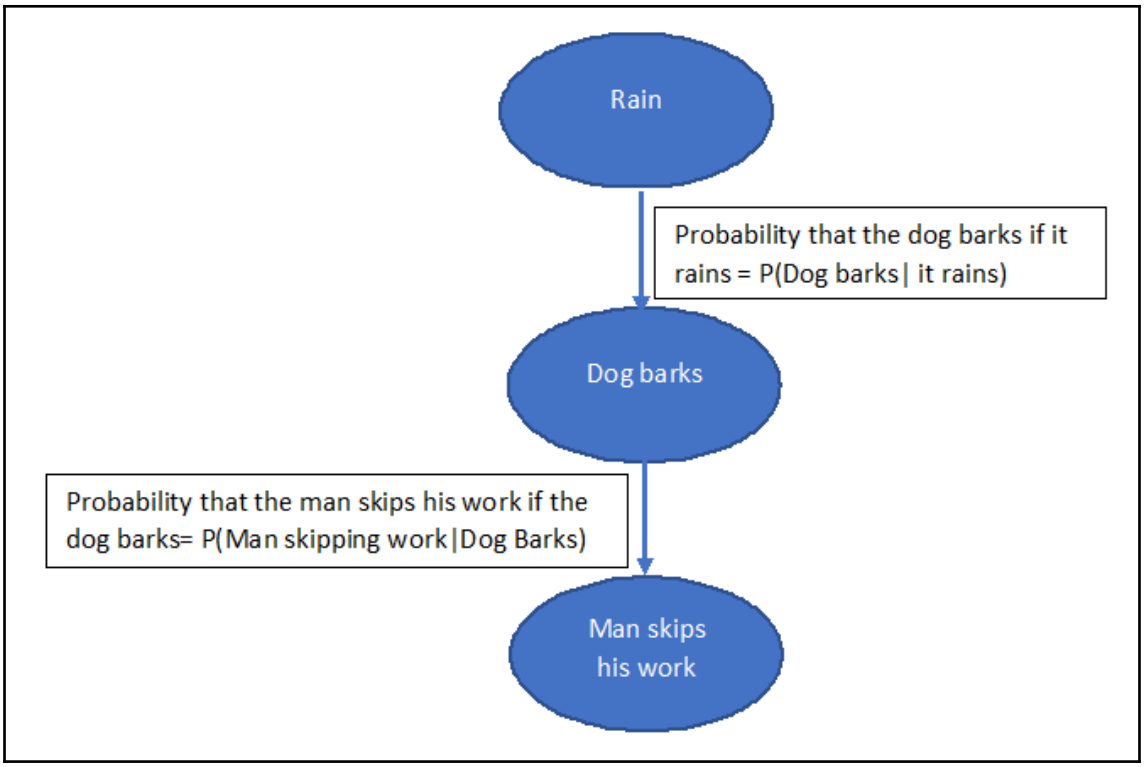


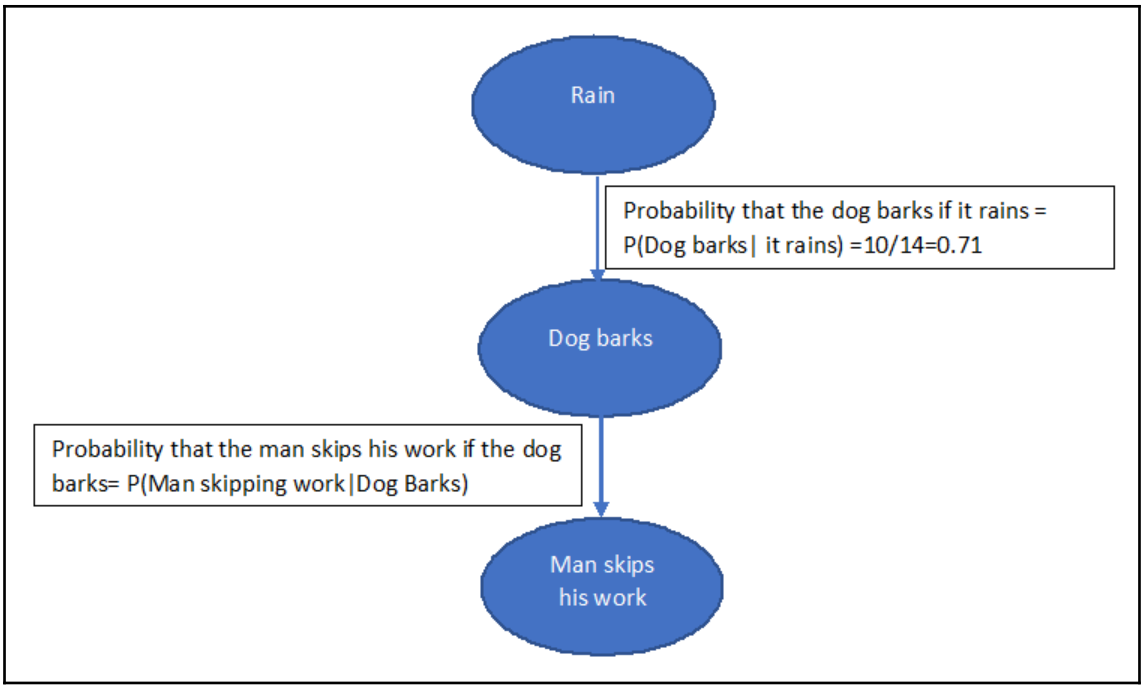

```
{('bread',): (('butter',), 0.6),
 ('bread', 'butter'): (('cereal',), 0.6666666666666666),
 ('bread', 'cereal'): (('butter',), 1.0),
 ('butter',): (('bread',), 0.5),
 ('butter', 'cereal'): (('bread',), 0.5),
 ('cereal',): (('butter',), 0.8)}
```

Chapter 8: Probabilistic Graphical Models




1
Late Wake-up
2
Accident on the highway
1
Rainy Day
4
Late for work
4
Traffic Jam
4
Late for meeting


Rain
Probability that the dog barks if it rains = P(Dog barks| it rains)
Dog barks
Probability that the man skips his work if the dog barks= P(Man skipping work|Dog Barks)
Man skips his work

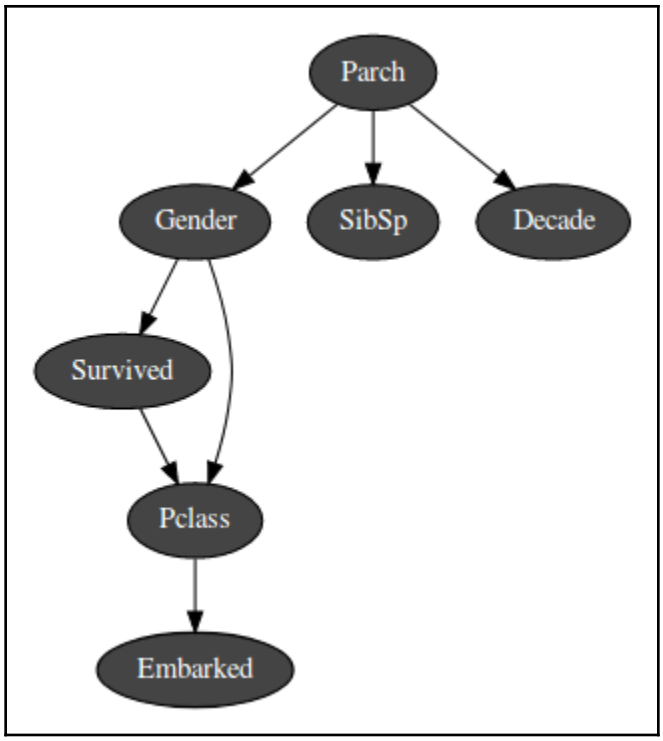


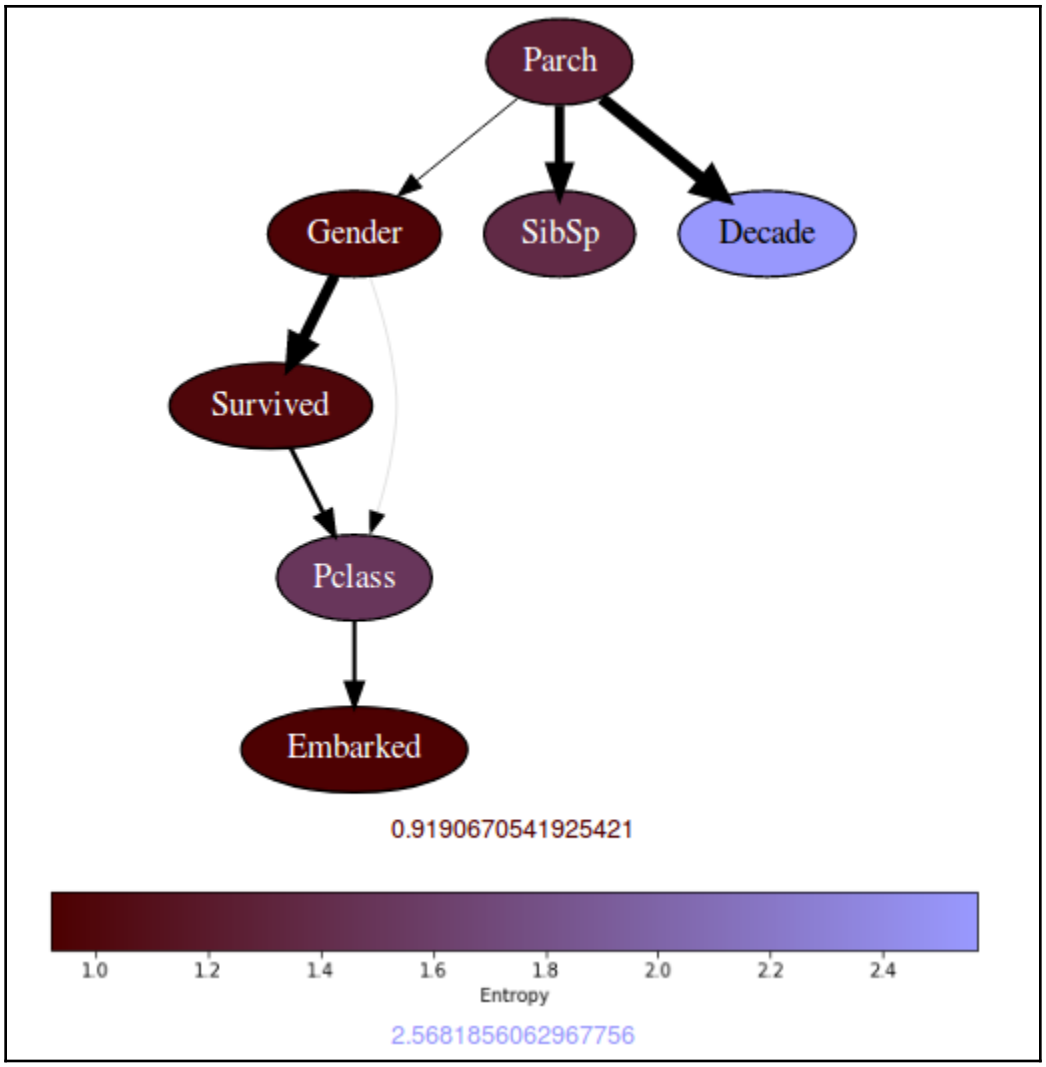

```
PassengerId: 891
Survived: 2
Pclass: 3
Name: 891
Gender: 2
Age: 89
SibSp: 7
Parch: 7
Ticket: 681
Fare: 248
Cabin: 148
Embarked: 4
```

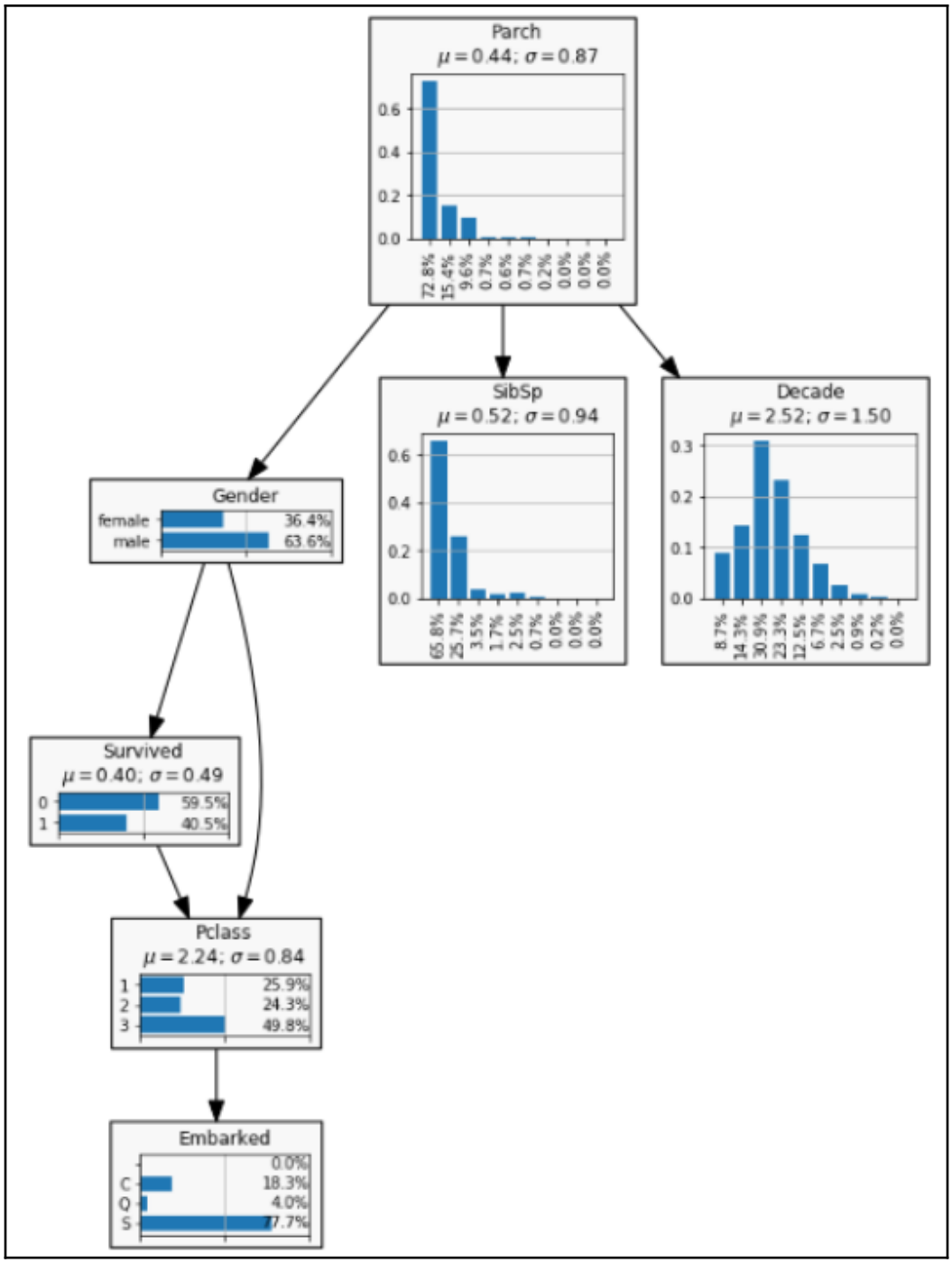
```
Survived
Pclass
Gender
SibSp
Parch
Embarked
```

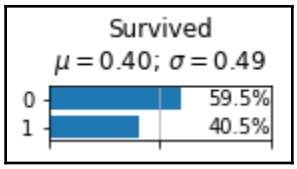
```
Decade: 10
```

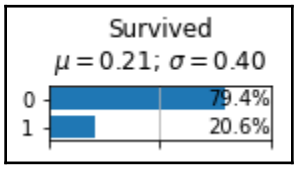
```
Survived : [0 1]
Pclass : [3 1 2]
Gender : ['male' 'female']
SibSp : [1 0 3 4 2 5]
Parch : [0 1 2 5 3 4 6]
Embarked : ['S' 'C' 'Q']
Decade : [2 3 5 0 1 4 6 7 8]
```

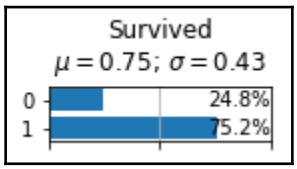

Parch
Gender
SibSp
Decade
Survived
Pclass
Embarked

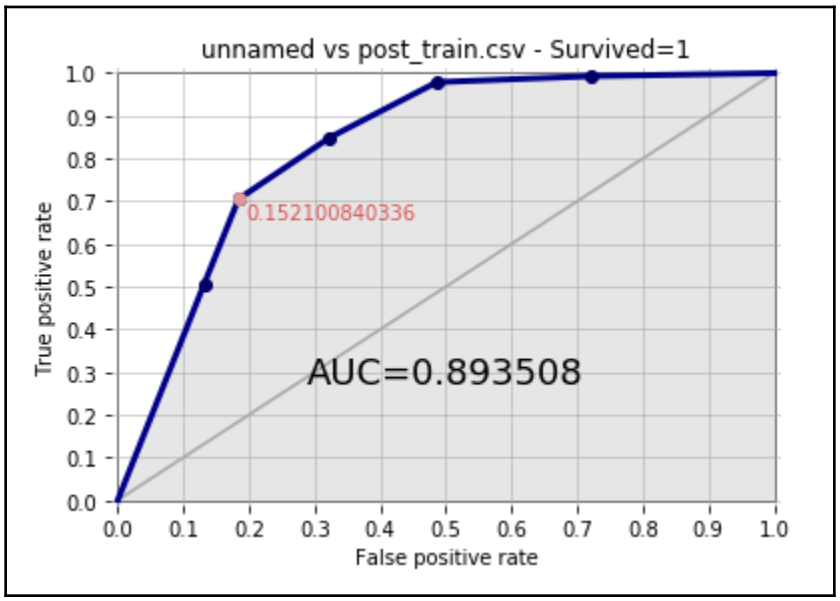

Parch
Gender
SibSp
Decade
Survived
Pclass
Embarked
0.9190670541925421
1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
Entropy
2.5681856062967756


Parch
$\mu = 0.44; \sigma = 0.87$
0.6
0.4
0.2
0.0
72.8%
15.4%
9.6%
0.7%
0.6%
0.7%
0.2%
0.0%
0.0%
0.0%
SibSp
$\mu = 0.52; \sigma = 0.94$
0.6
0.4
0.2
0.0
65.8%
25.7%
3.5%
1.7%
2.5%
0.7%
0.0%
0.0%
0.0%
Decade
$\mu = 2.52; \sigma = 1.50$
0.3
0.2
0.1
0.0
8.7%
14.3%
30.9%
23.3%
12.5%
6.7%
2.5%
0.9%
0.2%
0.0%
Gender
female
male
36.4%
63.6%
Survived
$\mu = 0.40; \sigma = 0.49$
0
1
59.5%
40.5%
Pclass
$\mu = 2.24; \sigma = 0.84$
1
2
3
25.9%
24.3%
49.8%
Embarked
0.0%
C
18.3%
Q
4.0%
S
77.7%

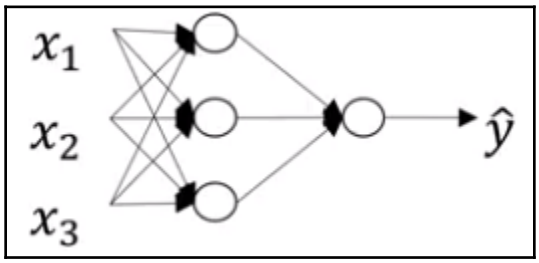

Survived
$\mu = 0.40; \sigma = 0.49$
0
1
59.5%
40.5%

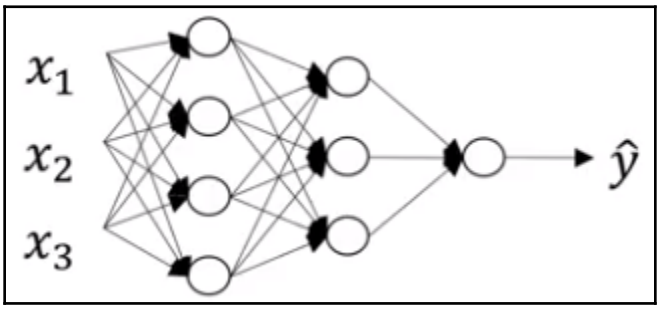

Survived
$\mu = 0.21; \sigma = 0.40$
0
1
79.4%
20.6%

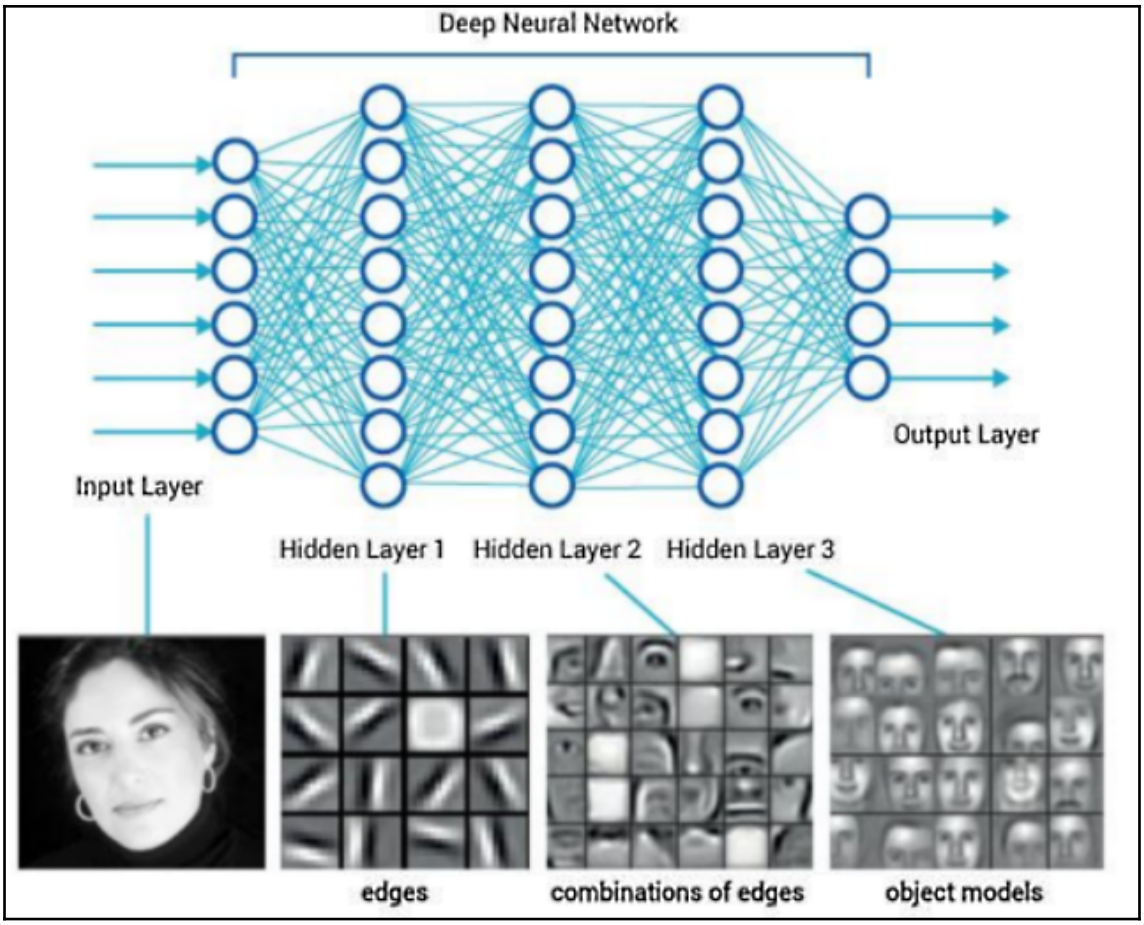

Survived
$\mu = 0.75; \sigma = 0.43$
0
1
24.8%
75.2%

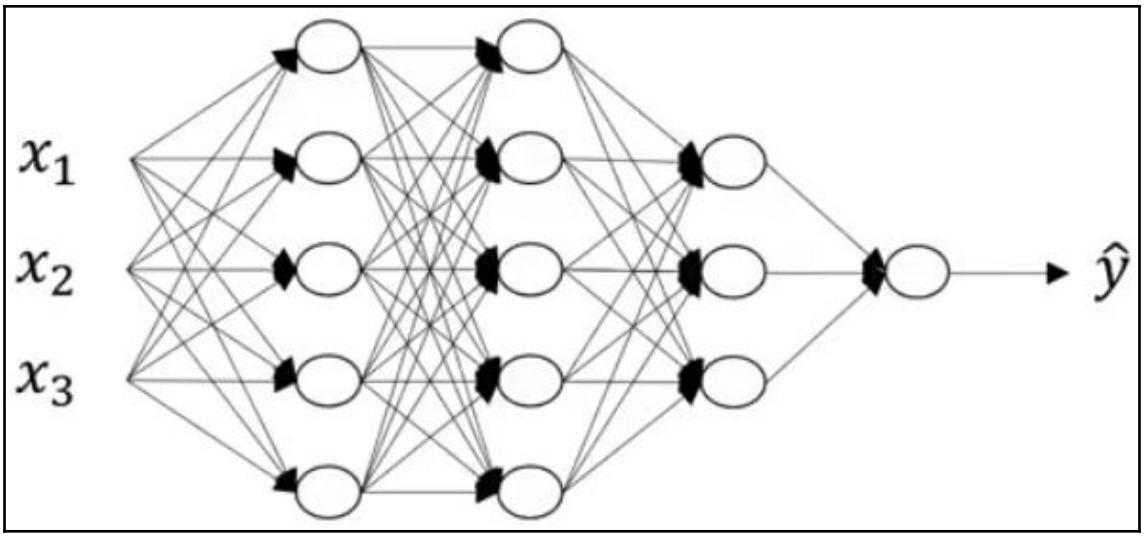

unnamed vs post_train.csv - Survived=1
True positive rate
False positive rate
0.152100840336
AUC=0.893508
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

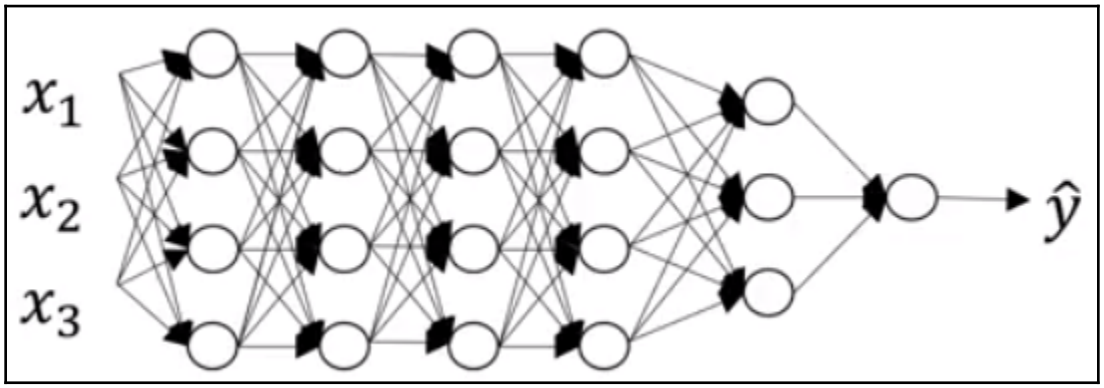
Chapter 9: Selected Topics in Deep Learning

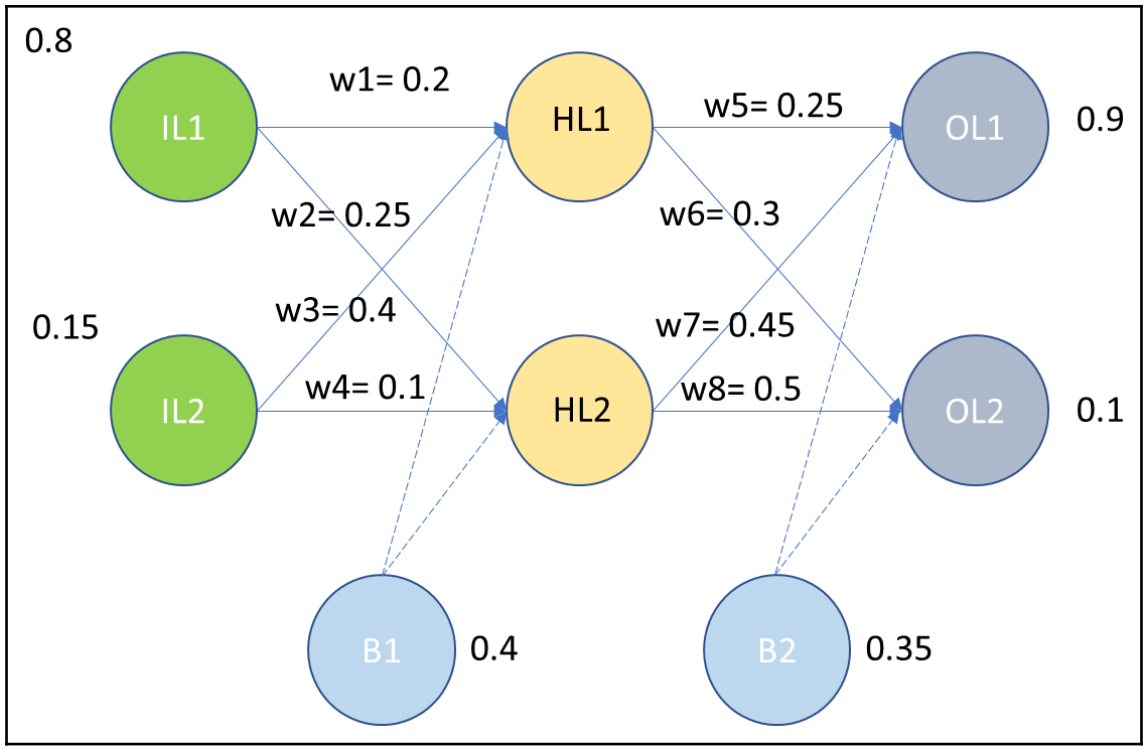


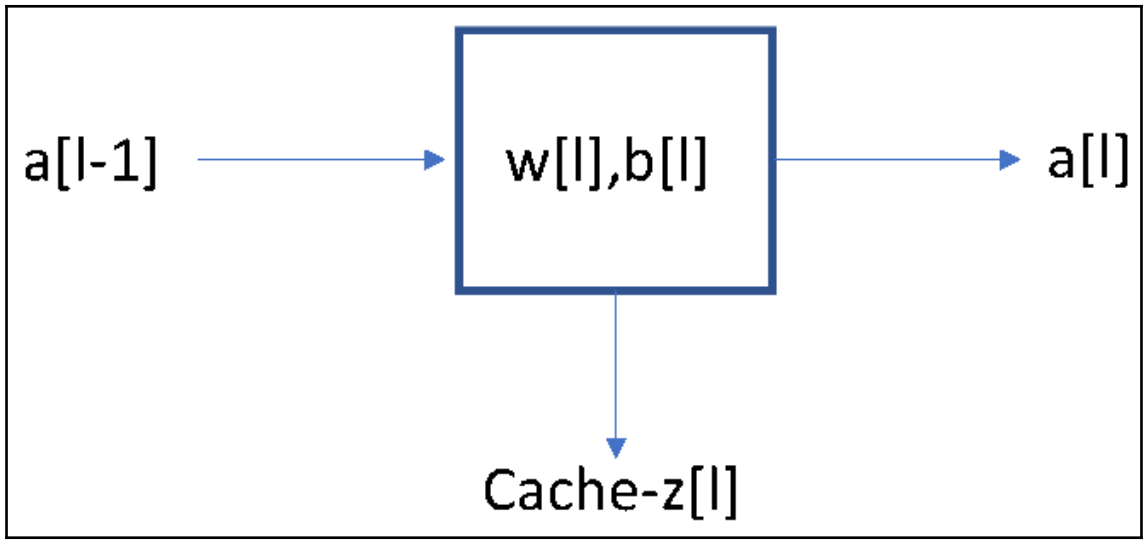


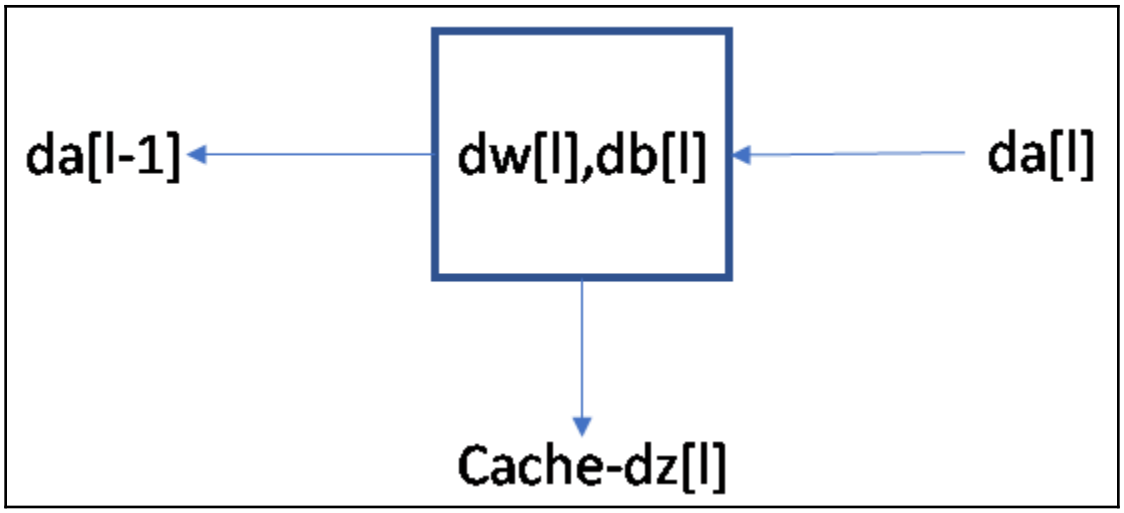

Deep Neural Network
Input Layer
Hidden Layer 1
Hidden Layer 2
Hidden Layer 3
Output Layer
edges
combinations of edges
object models


x_1
x_2
x_3
$\hat{y}$


x_1
x_2
x_3
$\hat{y}$


0.8
IL1
w1= 0.2
HL1
w5= 0.25
OL1
0.9
w2= 0.25
w6= 0.3
w3= 0.4
0.15
w7= 0.45
IL2
w4= 0.1
HL2
w8= 0.5
OL2
0.1
B1
0.4
B2
0.35

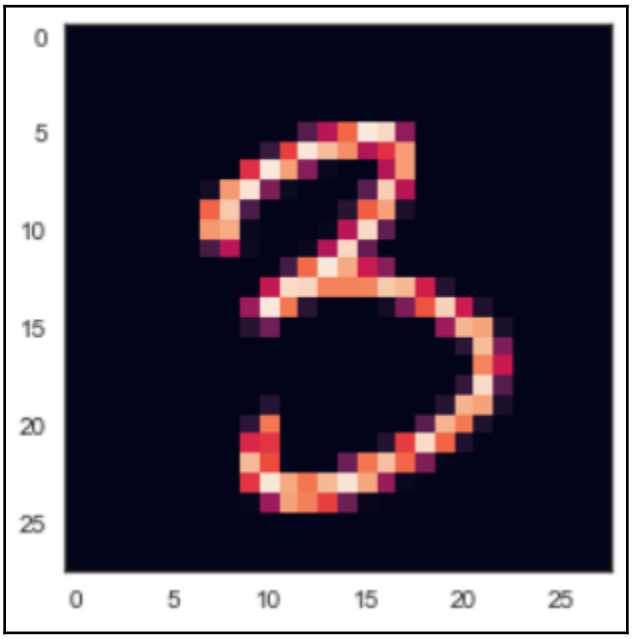

a[l-1]
w[l],b[l]
a[l]
Cache-z[l]




```
1    4684
7    4401
3    4351
9    4188
2    4177
6    4137
0    4132
4    4072
8    4063
5    3795
Name: label, dtype: int64
```

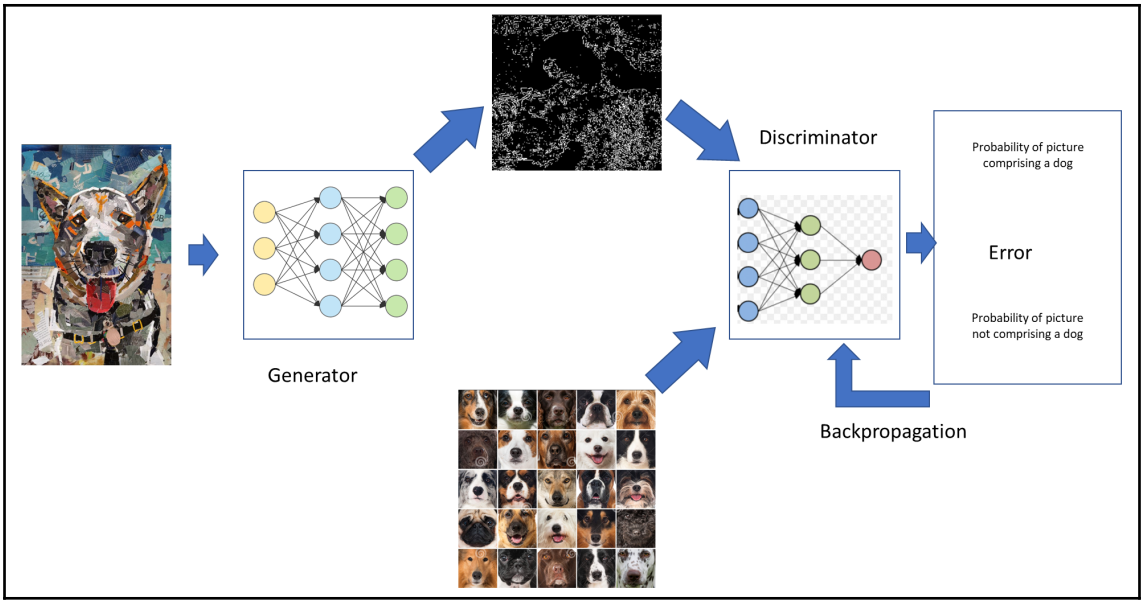
```
count       784
unique        1
top       False
freq        784
dtype: object
```

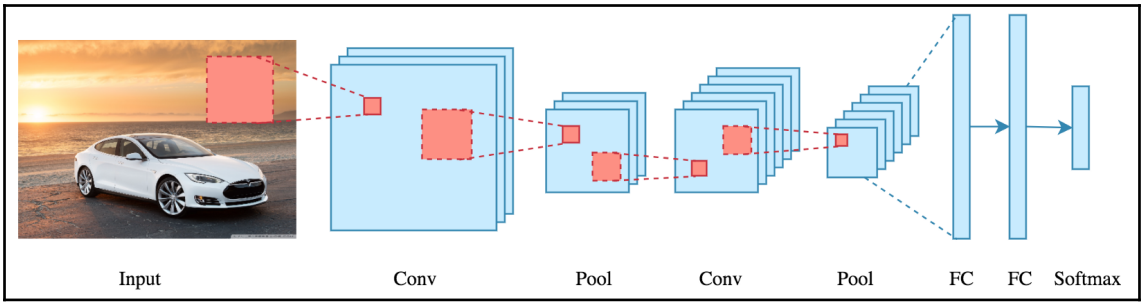
```
count        784
unique         1
top        False
freq         784
dtype: object
```

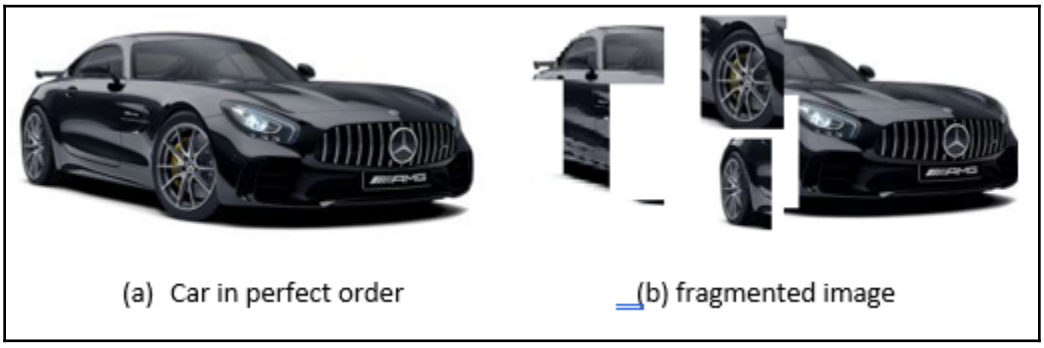



```
Epoch 1/5
 - 217s - loss: 1.8641 - acc: 0.3410 - val_loss: 0.5560 - val_acc: 0.8583
Epoch 2/5
 - 218s - loss: 0.7747 - acc: 0.7465 - val_loss: 0.1898 - val_acc: 0.9481
Epoch 3/5
 - 224s - loss: 0.4380 - acc: 0.8618 - val_loss: 0.1323 - val_acc: 0.9605
Epoch 4/5
 - 219s - loss: 0.3300 - acc: 0.8962 - val_loss: 0.1068 - val_acc: 0.9700
Epoch 5/5
 - 233s - loss: 0.2802 - acc: 0.9143 - val_loss: 0.0915 - val_acc: 0.9726
```

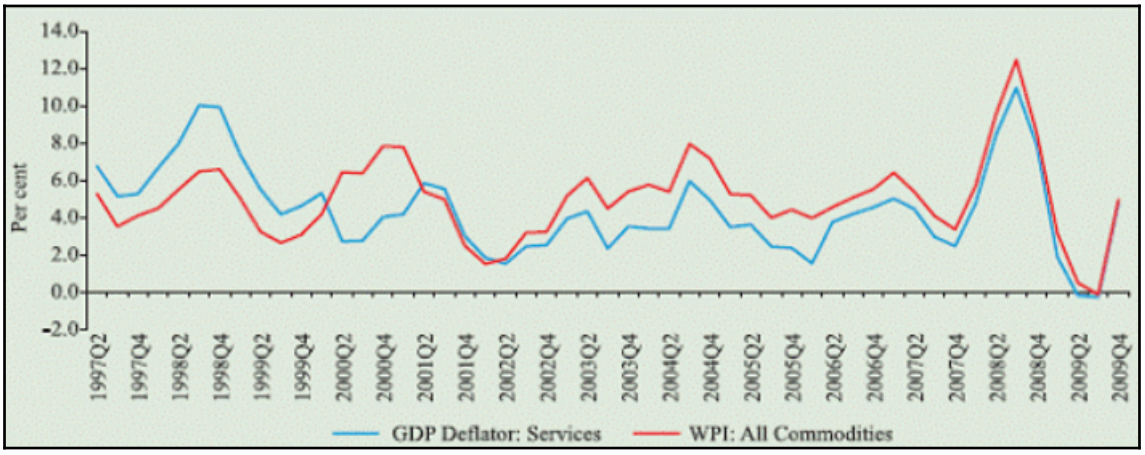
0	2
1	0
2	9
3	0
4	3
5	9
6	0
7	3
8	0
9	3
10	5
11	7
12	4
13	0
14	4
15	3
16	3
17	1
18	9
19	0
20	9
21	1
22	1
23	5
24	7
25	4
26	2
27	7
28	4
29	7

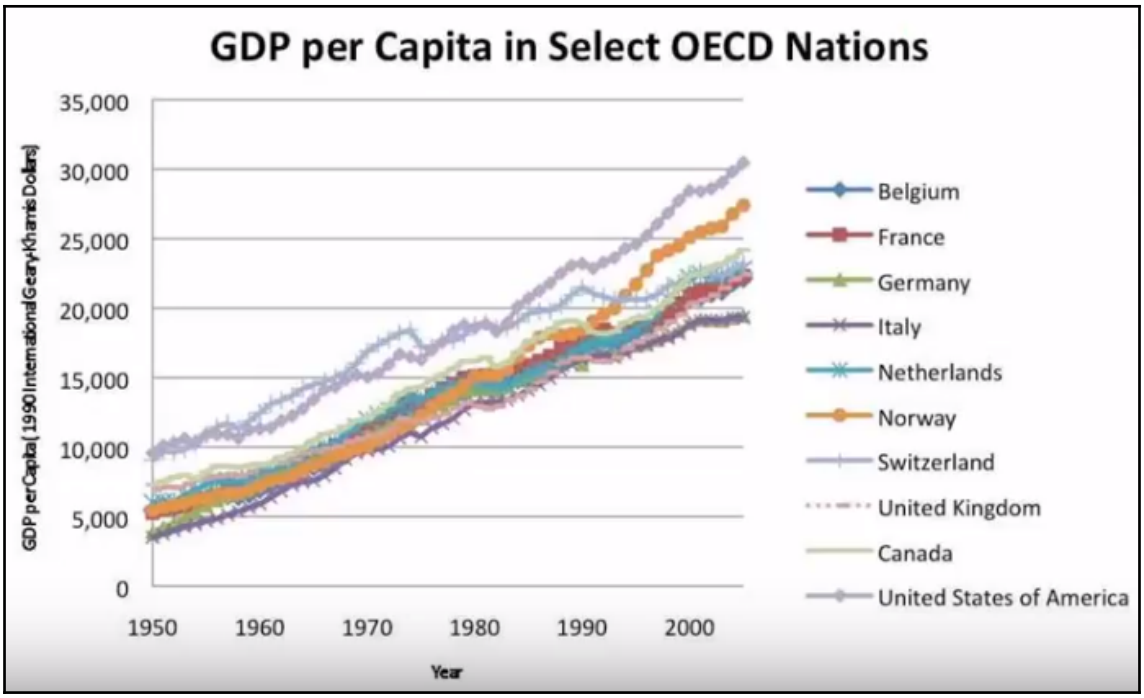

Discriminator
Probability of picture comprising a dog
Error
Probability of picture not comprising a dog
Generator
Backpropagation


Input
Conv
Pool
Conv
Pool
FC
FC
Softmax

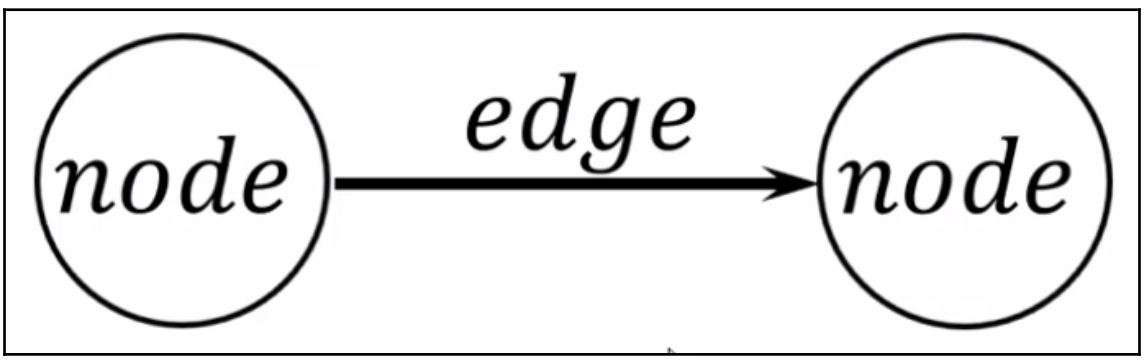

(a) Car in perfect order
(b) fragmented image

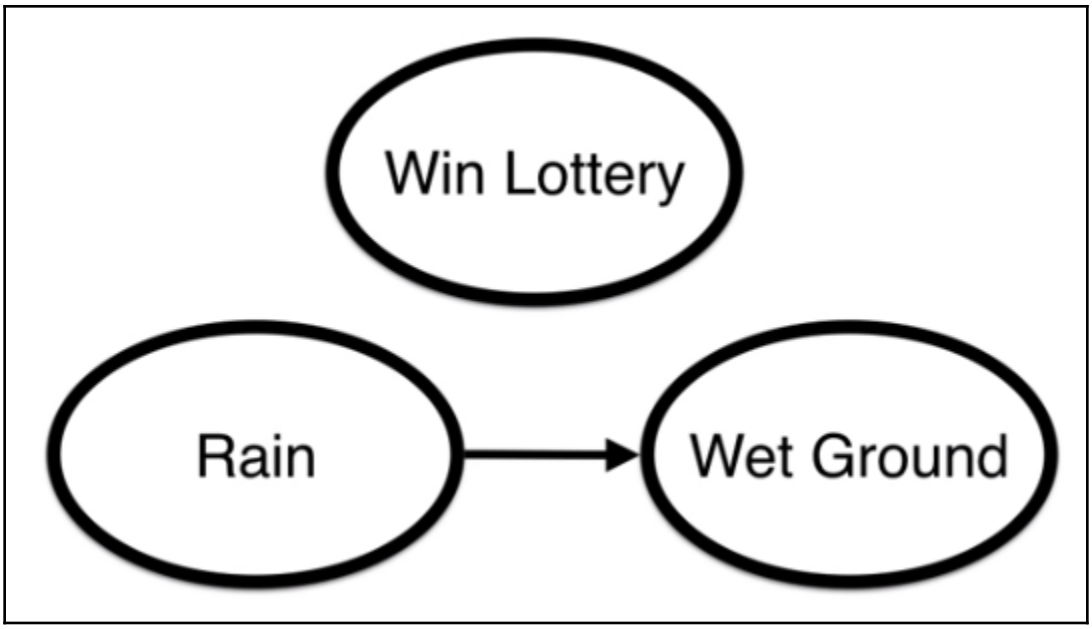
Chapter 10: Causal Inference

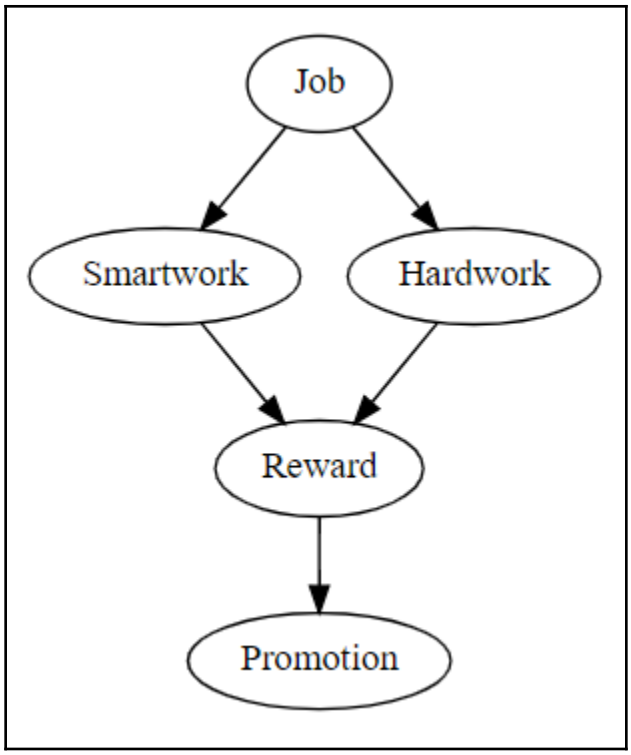





```
Granger causality test

Model 1: NOx.GT. ~ Lags(NOx.GT., 1:3) + Lags(NO2.GT., 1:3)
Model 2: NOx.GT. ~ Lags(NOx.GT., 1:3)
  Res.Df Df      F    Pr(>F)
1   9347
2   9350 -3 282.01 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

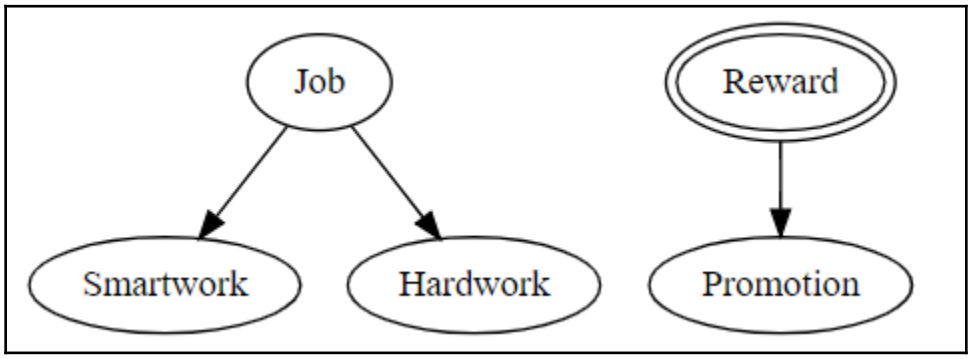



Win Lottery
Rain
Wet Ground

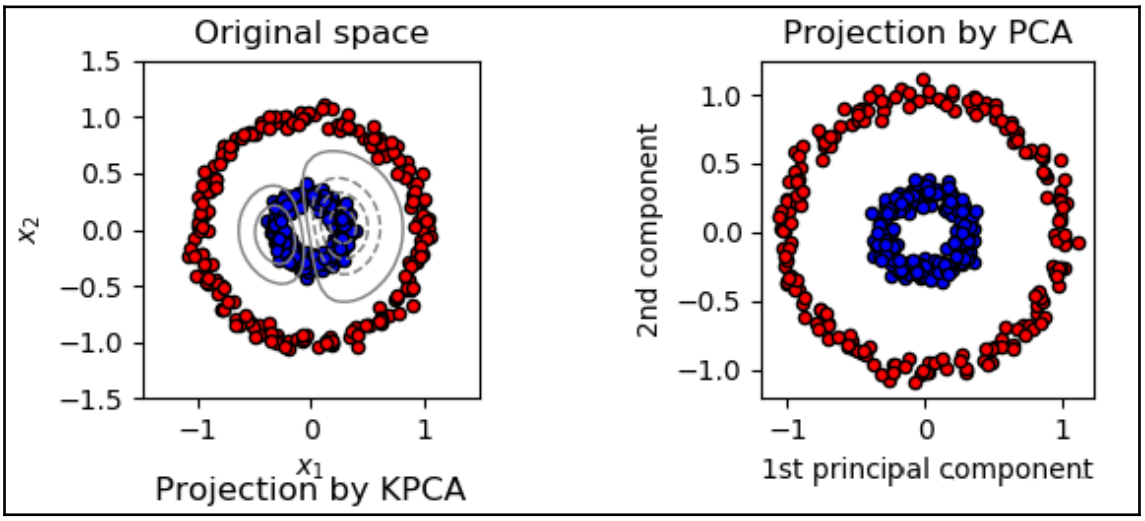


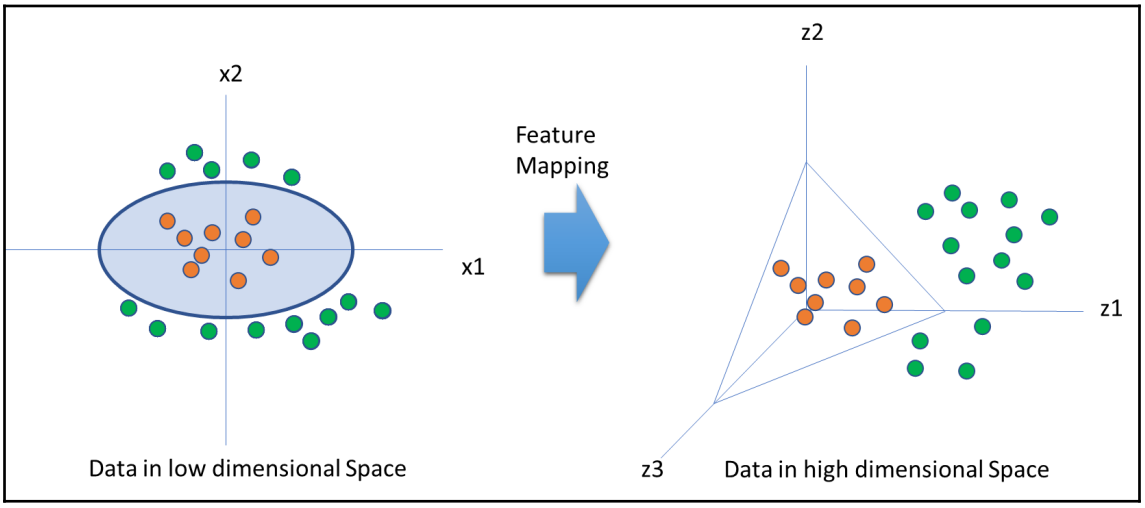

```
P(Job)P(Hardwork|Job)P(Smartwork|Job)P(Reward|Smartwork,Hardwork)P(Promotion|Reward)
```

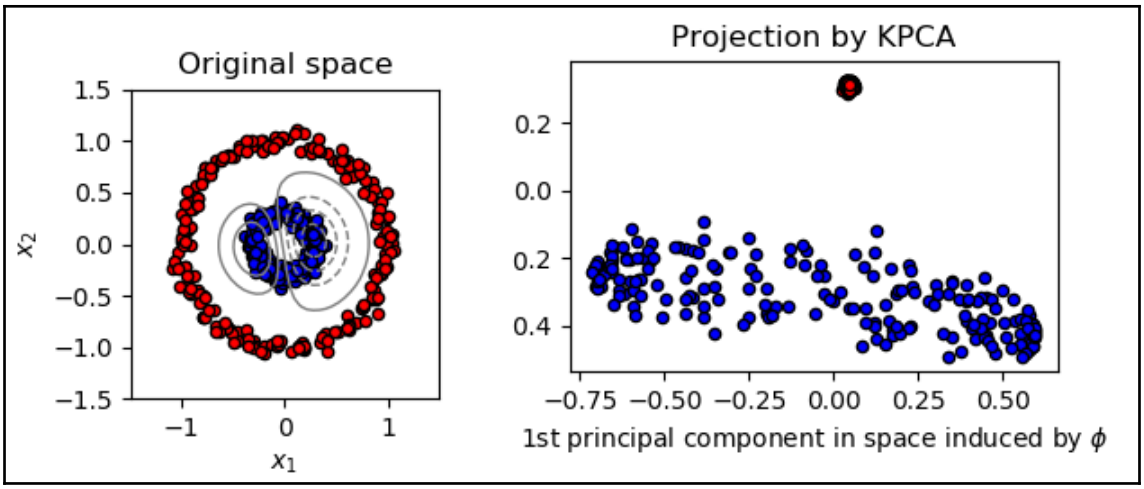
```
[('Job', 'Promotion', {'Reward'}),
 ('Job', 'Promotion', {'Reward', 'Smartwork'}),
 ('Job', 'Promotion', {'Hardwork', 'Smartwork'}),
 ('Job', 'Promotion', {'Hardwork', 'Reward'}),
 ('Job', 'Promotion', {'Hardwork', 'Reward', 'Smartwork'}),
 ('Job', 'Reward', {'Hardwork', 'Smartwork'}),
 ('Job', 'Reward', {'Hardwork', 'Promotion', 'Smartwork'}),
 ('Promotion', 'Smartwork', {'Reward'}),
 ('Promotion', 'Smartwork', {'Job', 'Reward'}),
 ('Promotion', 'Smartwork', {'Hardwork', 'Reward'}),
 ('Promotion', 'Smartwork', {'Hardwork', 'Job', 'Reward'}),
 ('Promotion', 'Hardwork', {'Reward'}),
 ('Promotion', 'Hardwork', {'Job', 'Reward'}),
 ('Promotion', 'Hardwork', {'Reward', 'Smartwork'}),
 ('Promotion', 'Hardwork', {'Job', 'Reward', 'Smartwork'}),
 ('Smartwork', 'Hardwork', {'Job'})]
```



Chapter 11: Advanced Methods



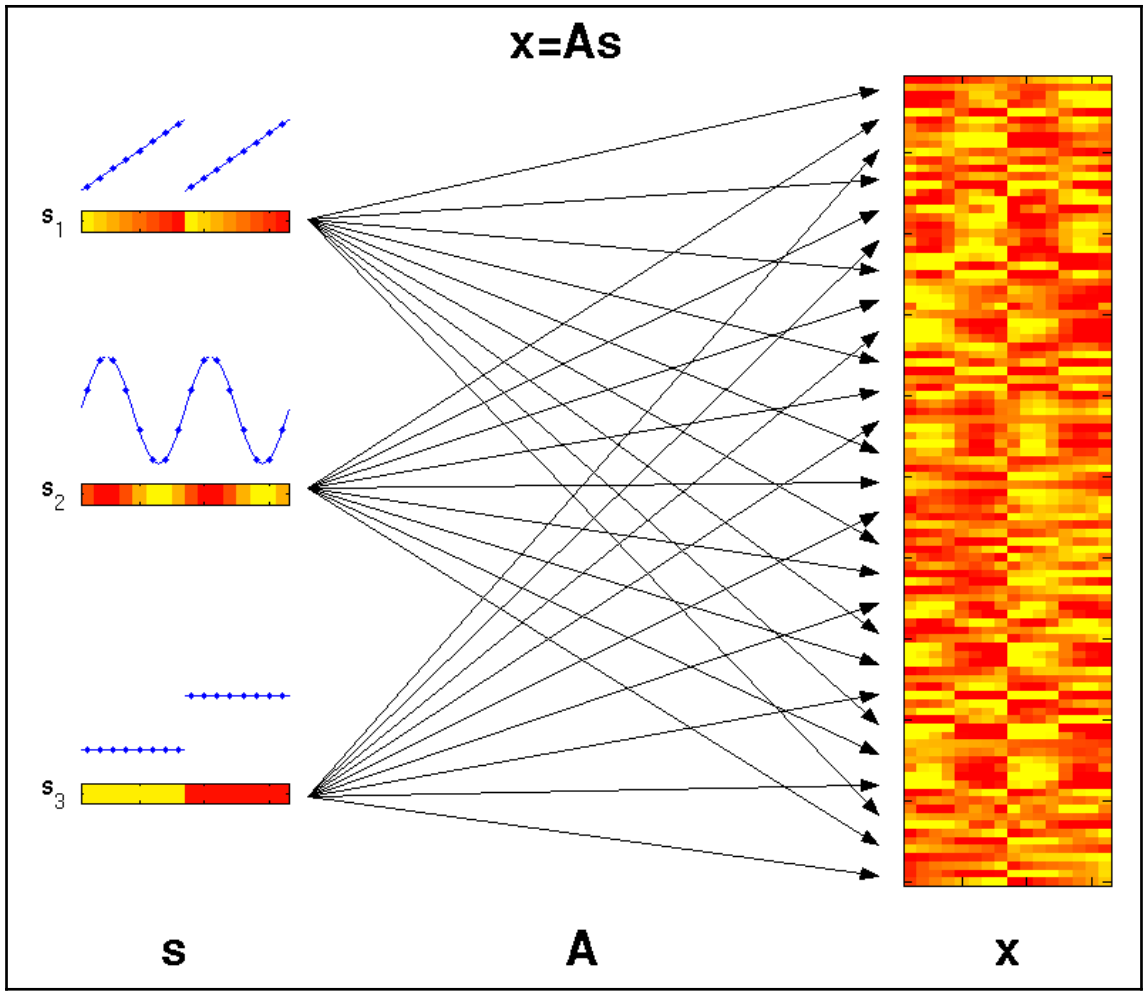


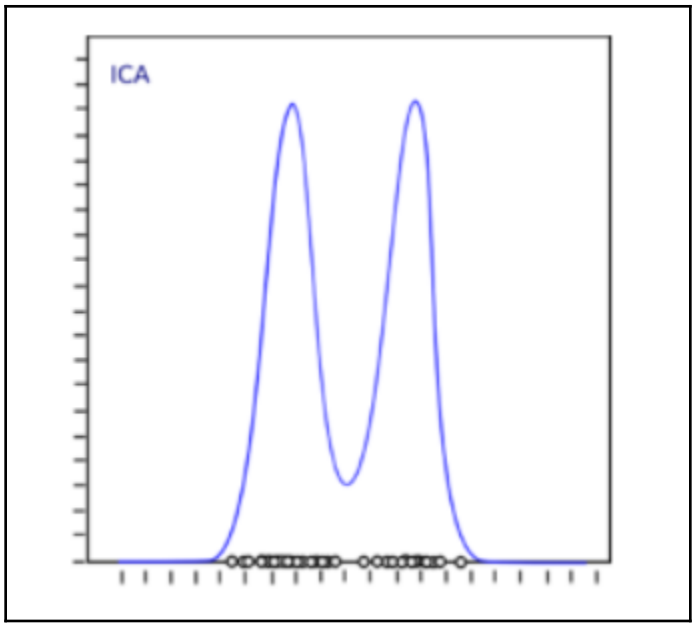


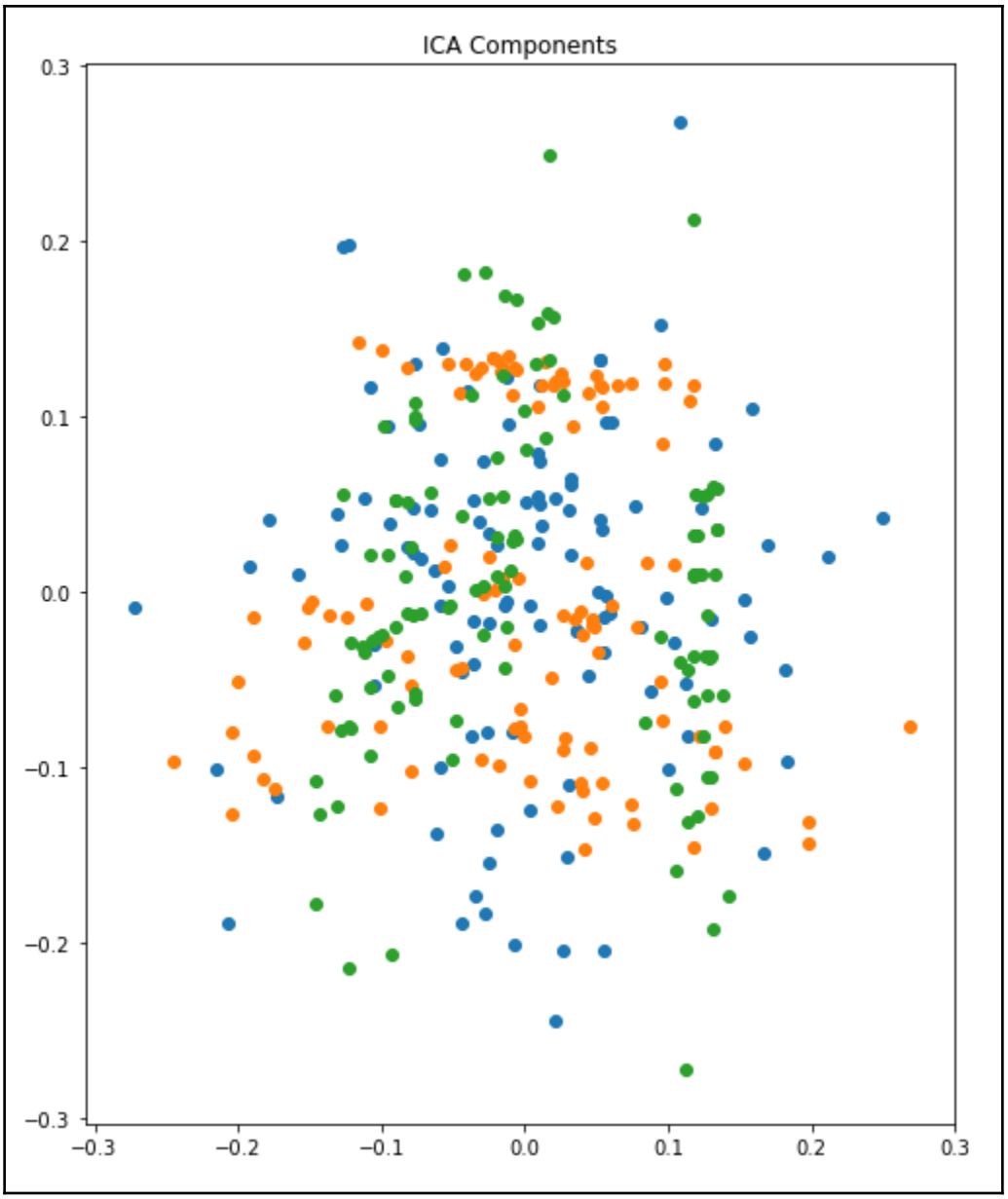

```
array([[ 5.1,  3.5,  1.4,  0.2],
       [ 4.9,  3. ,  1.4,  0.2],
       [ 4.7,  3.2,  1.3,  0.2],
       [ 4.6,  3.1,  1.5,  0.2],
       [ 5. ,  3.6,  1.4,  0.2],
       [ 5.4,  3.9,  1.7,  0.4],
       [ 4.6,  3.4,  1.4,  0.3],
       [ 5. ,  3.4,  1.5,  0.2],
       [ 4.4,  2.9,  1.4,  0.2],
       [ 4.9,  3.1,  1.5,  0.1],
       [ 5.4,  3.7,  1.5,  0.2],
       [ 4.8,  3.4,  1.6,  0.2],
       [ 4.8,  3. ,  1.4,  0.1],
       [ 4.3,  3. ,  1.1,  0.1],
       [ 5.8,  4. ,  1.2,  0.2],
       [ 5.7,  4.4,  1.5,  0.4],
       [ 5.4,  3.9,  1.3,  0.4],
       [ 5.1,  3.5,  1.4,  0.3],
       [ 5.7,  3.8,  1.7,  0.3],
```

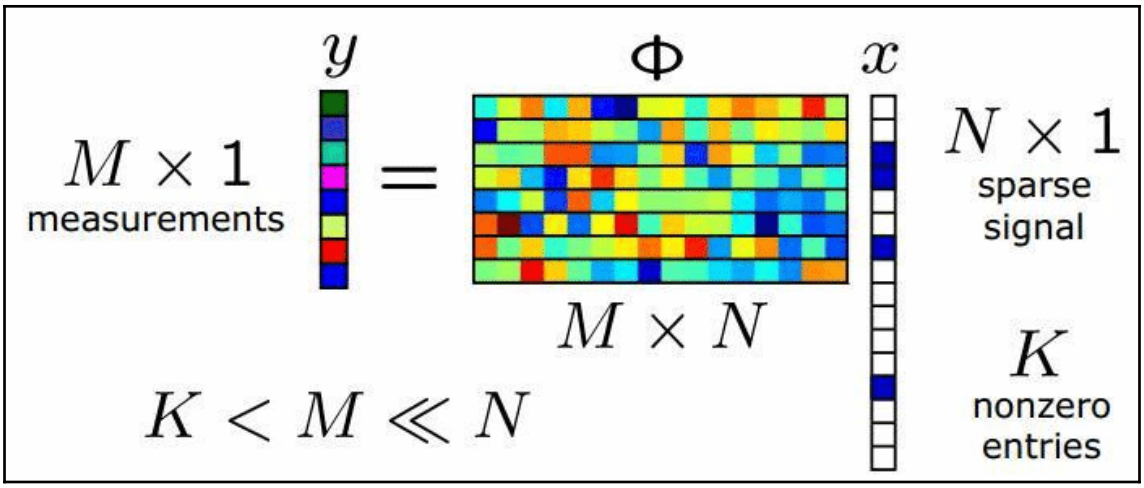
```
array([[  1.54399532e-02,  -1.19254753e-01,   2.25126850e-01,
          3.56381749e-01],
       [ -9.98450310e-02,  -1.04039491e+00,   1.13559562e-01,
         -2.86480506e-02],
       [  1.05300481e+00,  -1.19254753e-01,   9.50314227e-01,
          1.12644135e+00],
       [ -1.36797986e+00,   3.41315328e-01,  -1.39259884e+00,
         -1.31208072e+00],
       [  1.16828980e+00,   1.11030287e-01,   7.27179649e-01,
          1.38312788e+00],
       [ -1.02212490e+00,   1.03217045e+00,  -1.22524790e+00,
         -7.98707650e-01],
       [ -5.60984968e-01,   1.49274053e+00,  -1.28103155e+00,
         -1.31208072e+00],
       [ -1.02212490e+00,  -2.42210516e+00,  -1.65358660e-01,
         -2.85334584e-01],
       [  7.07149859e-01,  -1.19254753e-01,   9.50314227e-01,
          7.41411549e-01],
       [  9.37719827e-01,   5.71600368e-01,   1.06188152e+00,
```

```
array([[ -3.37725246e-01,  -2.67929399e-01],
       [ -2.28159618e-01,  -6.42964823e-01],
       [ -5.42657970e-01,   4.00470428e-01],
       [  7.68893086e-01,   4.93146060e-02],
       [ -4.94431455e-01,   4.74938667e-01],
       [  7.26835183e-01,   1.08946020e-01],
       [  6.93412746e-01,   1.93039625e-01],
       [  2.71518785e-02,  -4.56605659e-01],
       [ -5.56008764e-01,   2.08040974e-01],
       [ -4.15727350e-01,   5.35959232e-01],
       [ -2.33820327e-01,  -4.91624713e-01],
       [ -4.12615994e-01,  -2.79420859e-02],
       [ -1.42346326e-01,  -6.98291047e-01],
       [ -3.19744482e-01,   5.23774154e-01],
       [ -3.91565822e-01,   1.76538163e-01],
       [ -1.06099524e-01,  -7.01543477e-01],
       [ -5.02349236e-01,  -1.80419722e-01],
       [ -4.93355128e-01,  -1.85187916e-01],
       [ -1.27684052e-01,  -5.40426177e-01],
```

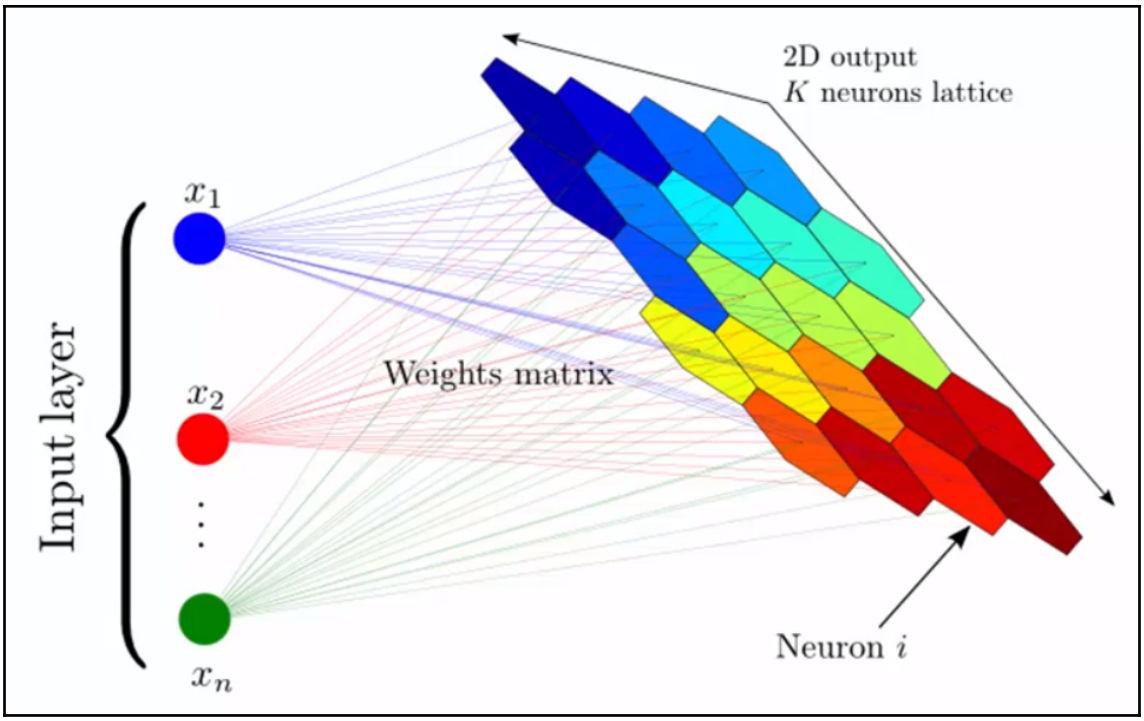

x=As
s1
s2
s3
s
A
x

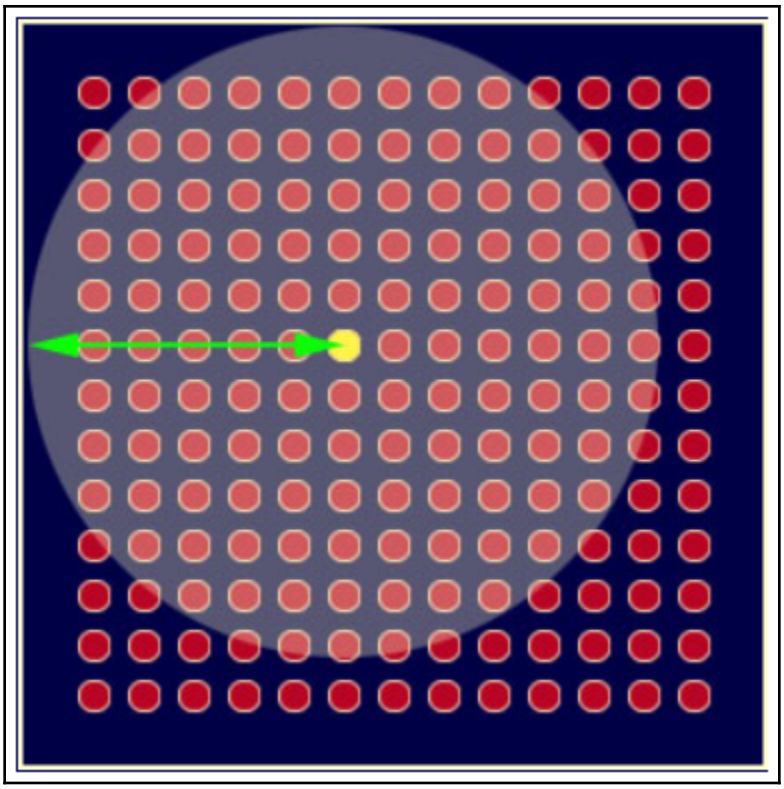

ICA

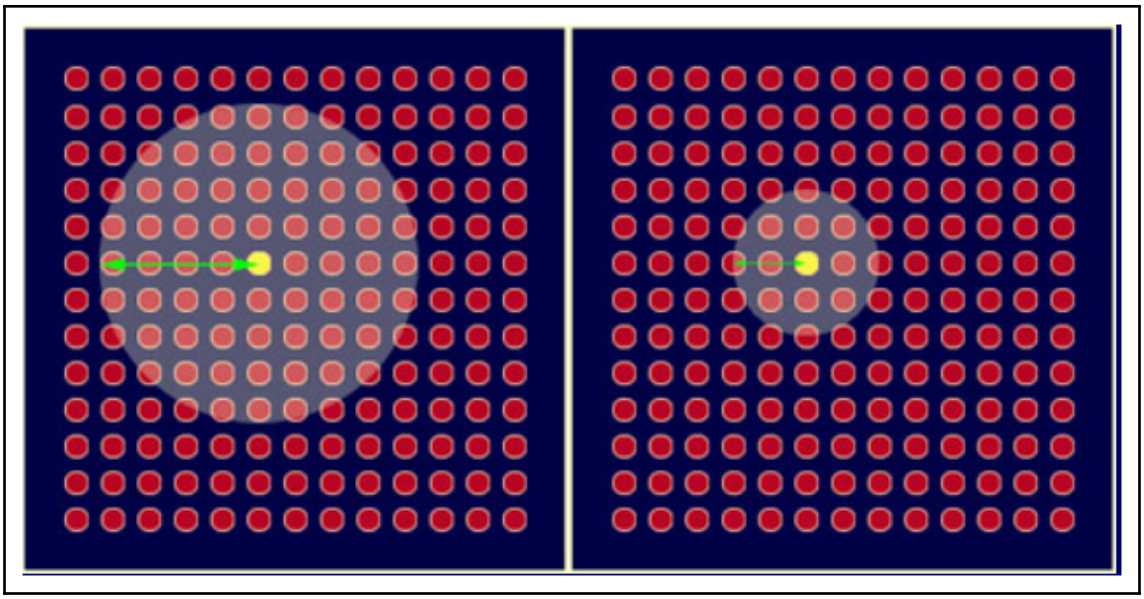

ICA Components
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3

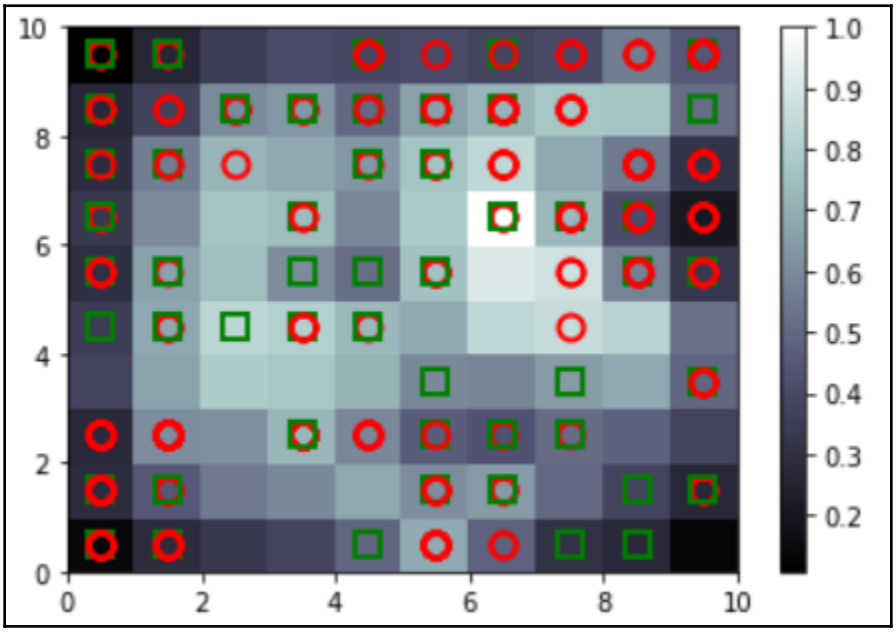



```
ECOS 2.0.4 - (C) embotech GmbH, Zurich Switzerland, 2012-15. Web: www.embotech.com/ECOS

It     pcost       dcost      gap   pres   dres    k/t    mu      step    sigma      IR     |   BT
 0  +0.000e+00  +4.986e-22  +6e+04  1e+00  1e-02  1e+00  3e+00    ---     ---     1  1  -  |  -  -
 1  +1.843e+02  +1.844e+02  +3e+04  9e-01  5e-03  6e-01  1e+00  0.6016  1e-01   1  1  1  |  0  0
 2  +2.368e+02  +2.371e+02  +2e+04  8e-01  4e-03  7e-01  1e+00  0.3928  6e-01   2  1  0  |  0  0
 3  +3.080e+02  +3.083e+02  +2e+04  6e-01  3e-03  5e-01  8e-01  0.5850  4e-01   1  1  1  |  0  0
 4  +3.660e+02  +3.662e+02  +9e+03  4e-01  2e-03  3e-01  5e-01  0.5546  3e-01   1  1  1  |  0  0
 5  +4.180e+02  +4.181e+02  +5e+03  2e-01  1e-03  2e-01  3e-01  0.5174  2e-01   1  1  1  |  0  0
 6  +4.659e+02  +4.660e+02  +3e+03  1e-01  5e-04  1e-01  2e-01  0.5998  3e-01   1  1  1  |  0  0
 7  +5.209e+02  +5.209e+02  +2e+03  6e-02  3e-04  7e-02  8e-02  0.6191  2e-01   1  1  1  |  0  0
 8  +5.643e+02  +5.643e+02  +7e+02  3e-02  1e-04  3e-02  3e-02  0.7787  3e-01   1  1  1  |  0  0
 9  +5.851e+02  +5.851e+02  +3e+02  1e-02  6e-05  2e-02  2e-02  0.7366  3e-01   1  1  1  |  0  0
10  +5.956e+02  +5.956e+02  +2e+02  7e-03  3e-05  8e-03  8e-03  0.7063  3e-01   1  1  1  |  0  0
11  +6.001e+02  +6.001e+02  +9e+01  4e-03  1e-05  4e-03  4e-03  0.5765  2e-01   1  1  1  |  0  0
12  +6.027e+02  +6.027e+02  +4e+01  2e-03  7e-06  2e-03  2e-03  0.7119  3e-01   1  1  1  |  0  0
13  +6.042e+02  +6.042e+02  +2e+01  8e-04  3e-06  9e-04  1e-03  0.7316  2e-01   1  1  1  |  0  0
14  +6.048e+02  +6.048e+02  +1e+01  4e-04  2e-06  5e-04  5e-04  0.6762  3e-01   1  1  1  |  0  0
15  +6.051e+02  +6.051e+02  +5e+00  2e-04  8e-07  2e-04  2e-04  0.7416  2e-01   1  1  1  |  0  0
16  +6.053e+02  +6.053e+02  +2e+00  8e-05  3e-07  1e-04  1e-04  0.6391  1e-01   1  1  1  |  0  0
17  +6.054e+02  +6.054e+02  +9e-01  4e-05  2e-07  4e-05  4e-05  0.7088  2e-01   1  1  1  |  0  0
18  +6.054e+02  +6.054e+02  +4e-01  2e-05  7e-08  2e-05  2e-05  0.9890  5e-01   1  1  1  |  0  0
19  +6.054e+02  +6.054e+02  +1e-01  5e-06  2e-08  6e-06  6e-06  0.7531  5e-02   1  1  1  |  0  0
20  +6.054e+02  +6.054e+02  +6e-02  2e-06  1e-08  3e-06  3e-06  0.6106  2e-01   1  1  1  |  0  0
21  +6.054e+02  +6.054e+02  +2e-02  7e-07  3e-09  9e-07  9e-07  0.8765  2e-01   1  1  1  |  0  0
22  +6.054e+02  +6.054e+02  +5e-03  2e-07  8e-10  2e-07  2e-07  0.7680  4e-02   1  1  1  |  0  0
23  +6.054e+02  +6.054e+02  +1e-03  6e-08  2e-10  6e-08  7e-08  0.7542  5e-02   1  1  1  |  0  0
24  +6.054e+02  +6.054e+02  +3e-04  1e-08  6e-11  2e-08  2e-08  0.8159  7e-02   1  1  1  |  0  0
25  +6.054e+02  +6.054e+02  +9e-06  4e-10  2e-12  5e-10  5e-10  0.9854  1e-02   1  1  1  |  0  0
26  +6.054e+02  +6.054e+02  +1e-07  4e-12  5e-14  5e-12  5e-12  0.9890  1e-04   1  1  1  |  0  0

OPTIMAL (within feastol=4.3e-12, reltol=1.7e-10, abstol=1.0e-07).
```


2D output
K neurons lattice
x_1
x_2
x_n
Input layer
Weights matrix
Neuron i






10
8
6
4
2
0
0
2
4
6
8
10
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

Graphics Bundle Ends Here

Index