Artificial Neural Networks(ANN)

Artificial Neural Networks(ANN)

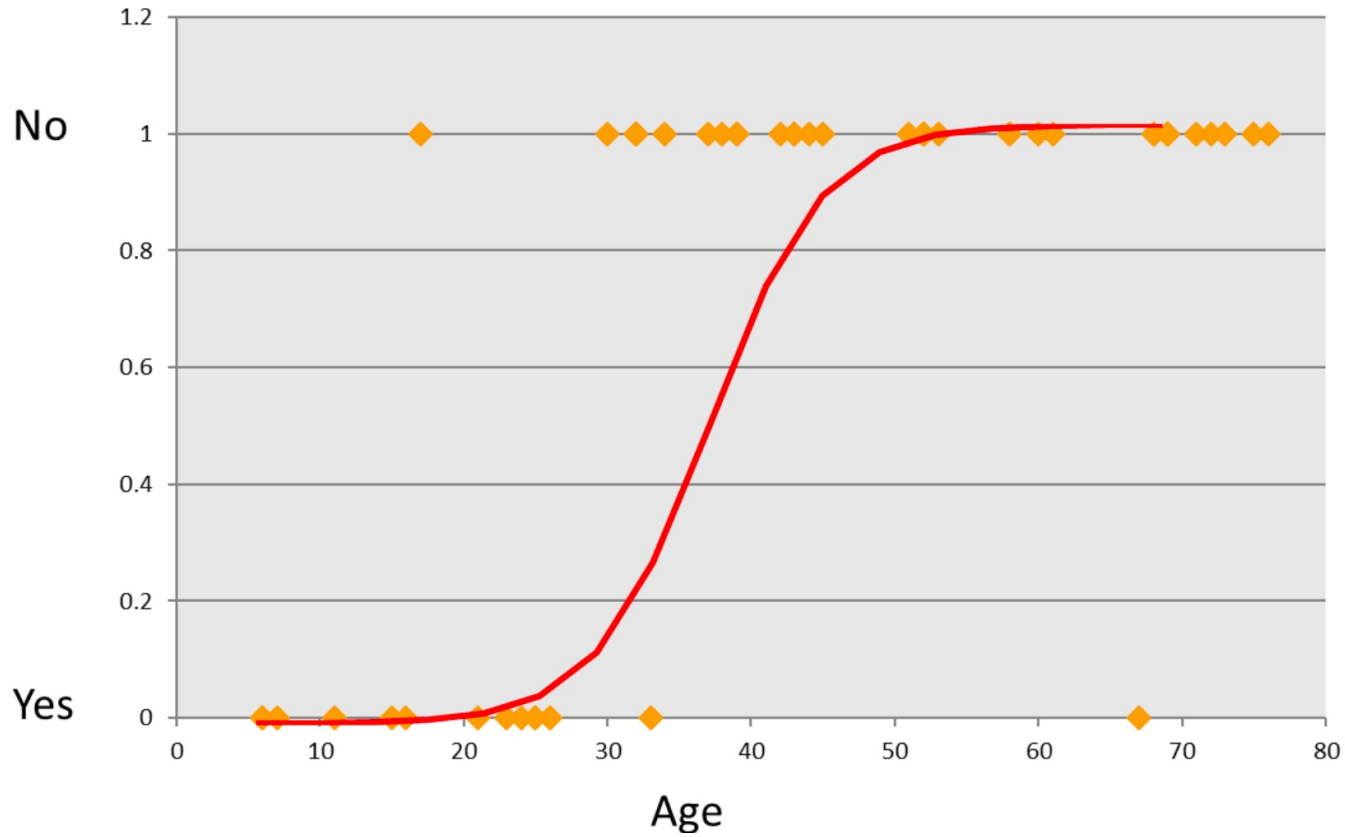
Contents

- Neural network Intuition
- Neural network and vocabulary
- Neural network algorithm
- Math behind neural network algorithm
- Building the neural networks
- Validating the neural network model
- Neural network applications
- Image recognition using neural networks

Recap of Logistic Regression

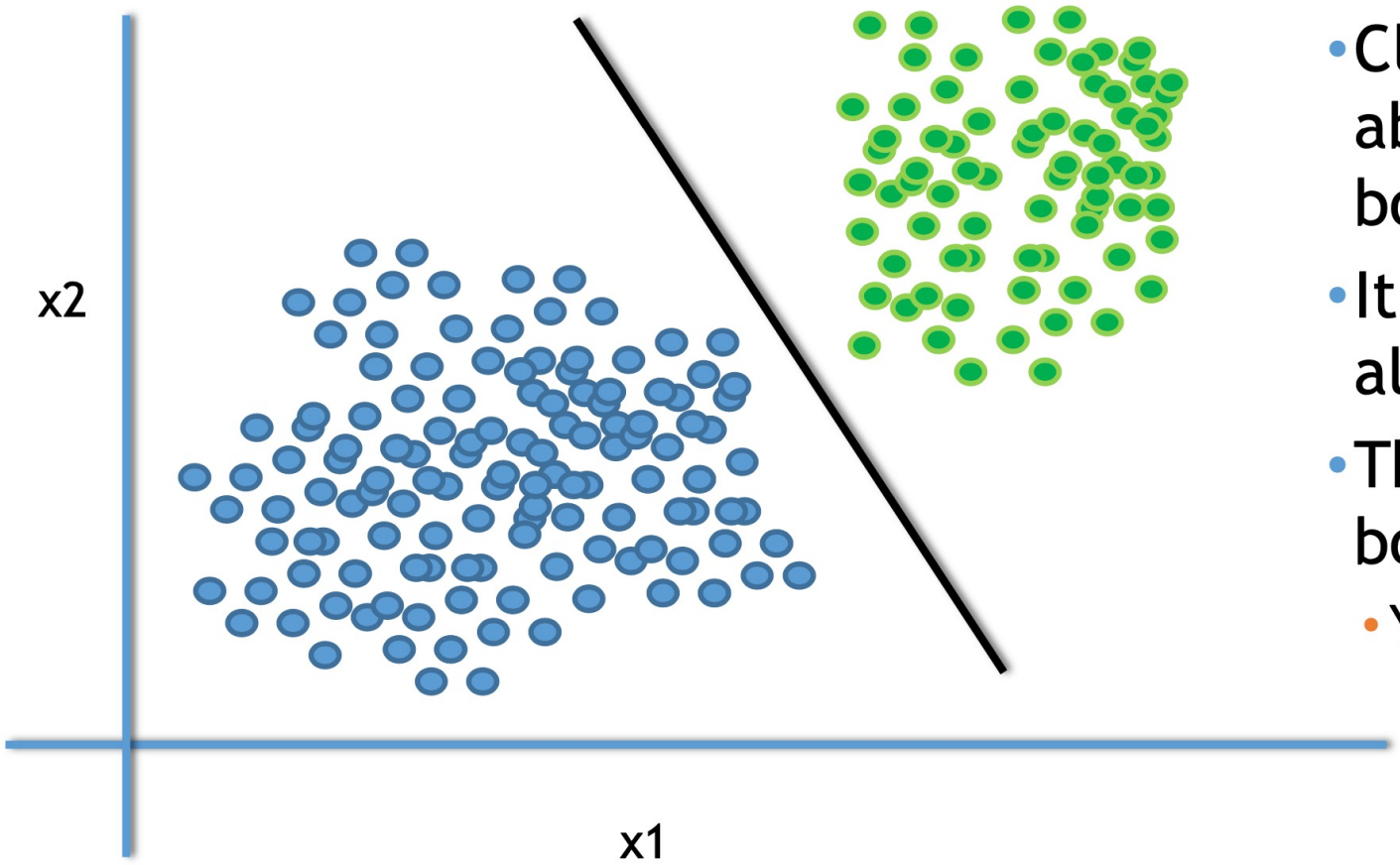
Recap of Logistic Regression

- Categorical output YES/NO type
- Using the predictor variables to predict the categorical output



Decision Boundary

Decision Boundary – Logistic Regression



- The line or margin that separates the classes
- Classification algorithms are all about finding the decision boundaries
- It need not be straight line always
- The final function of our decision boundary looks like
  - $Y=1$ if $w^T x+w_0>0$; else $Y=0$

Decision Boundary – Logistic Regression

- In logistic regression, Decision Boundary can be derived from the logistic regression coefficients and the threshold.
  - Imagine the logistic regression line $p(y)=e^{(b0+b1x1+b2x2)}/1+\exp^{(b0+b1x1+b2x2)}$
  - Suppose if $p(y)>0.5$ then class-1 or else class-0
    - $y=e^{(b1x1+b2x2)}/(1+e^{(b1x1+b2x2)})$
    - $\log(y/1-y)=b_0+b_1x_1+b_2x_2$
    - $\text{Log}(0.5/0.5)=b_0+b_1x_1+b_2x_2$
    - $0=b_0+b_1x_1+b_2x_2$
    - $b_0+b_1x_1+b_2x_2=0$ is the line

Decision Boundary – Logistic Regression

- Rewriting it in y=mx+c form
  - $X_2=(-b_1/b_2)X_1+(-b_0/b_2)$
- Anything above this line is class-1, below this line is class-0
  - $X_2>(-b_1/b_2)X_1+(-b_0/b_2)$is class-1
  - $X_2<(-b_1/b_2)X_1+(-b_0/b_2)$ is class-0
  - $X_2=(-b_1/b_2)X_1+(-b_0/b_2)$ tie probability of 0.5
- We can change the decision boundary by changing the threshold value(here 0.5)

LAB: Decision Boundary

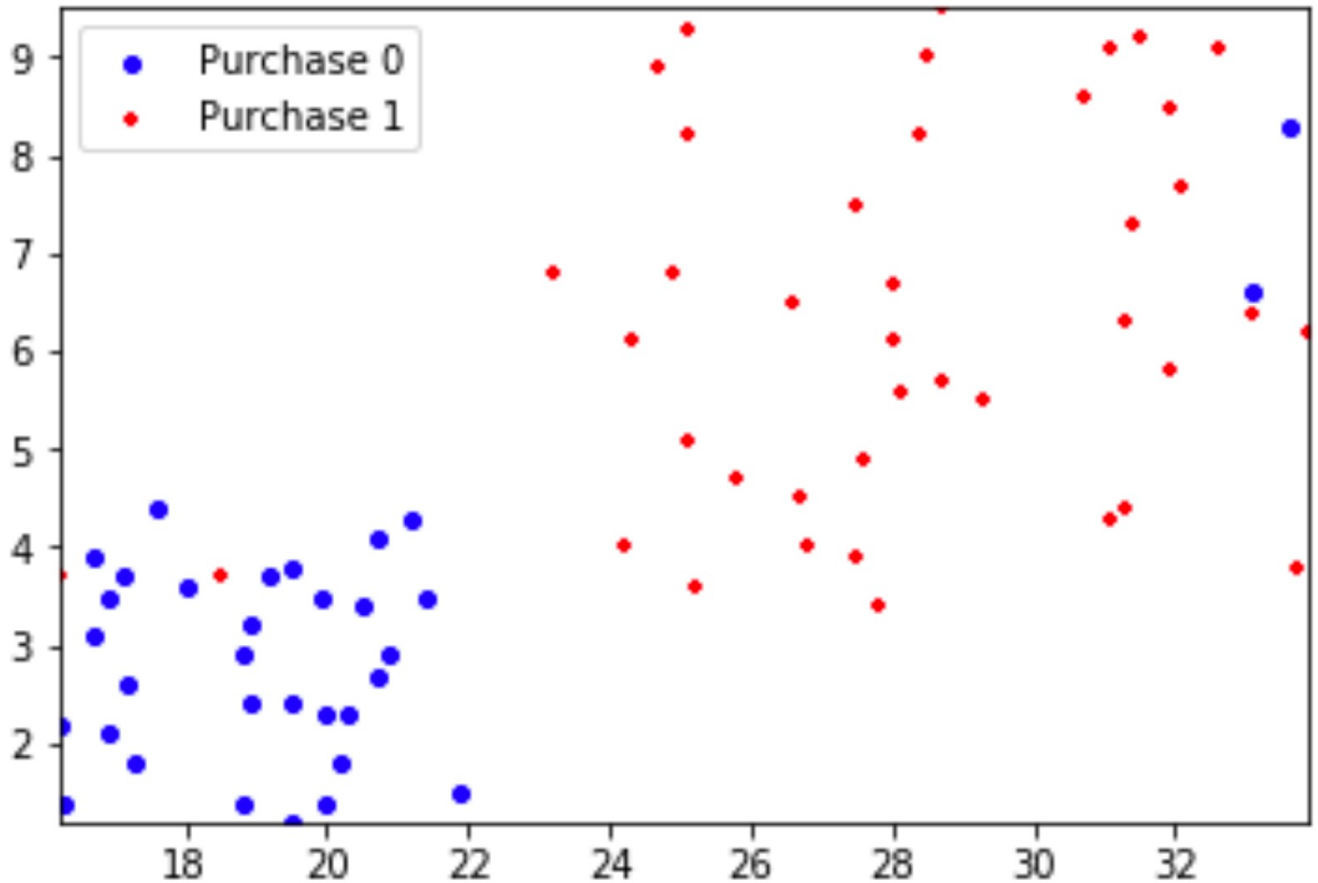
LAB: Logistic Regression

- Dataset: Emp_Purchase/Emp_Purchase.csv
- Filter the data and take a subset from above dataset . Filter condition is Sample_Set<3
- Draw a scatter plot that shows Age on X axis and Experience on Y-axis. Try to distinguish the two classes with colors or shapes (visualizing the classes)
- Build a logistic regression model to predict purchases using age and experience
- Create the confusion matrix
- Calculate the accuracy and error rates

LAB: Decision Boundary

- Draw a scatter plot that shows Age on X axis and Experience on Y-axis. Try to distinguish the two classes with colors or shapes (visualizing the classes)
- Build a logistic regression model to predict purchases using age and experience
- Finally draw the decision boundary for this logistic regression model

Code: Logistic Regression



Code: Logistic Regression

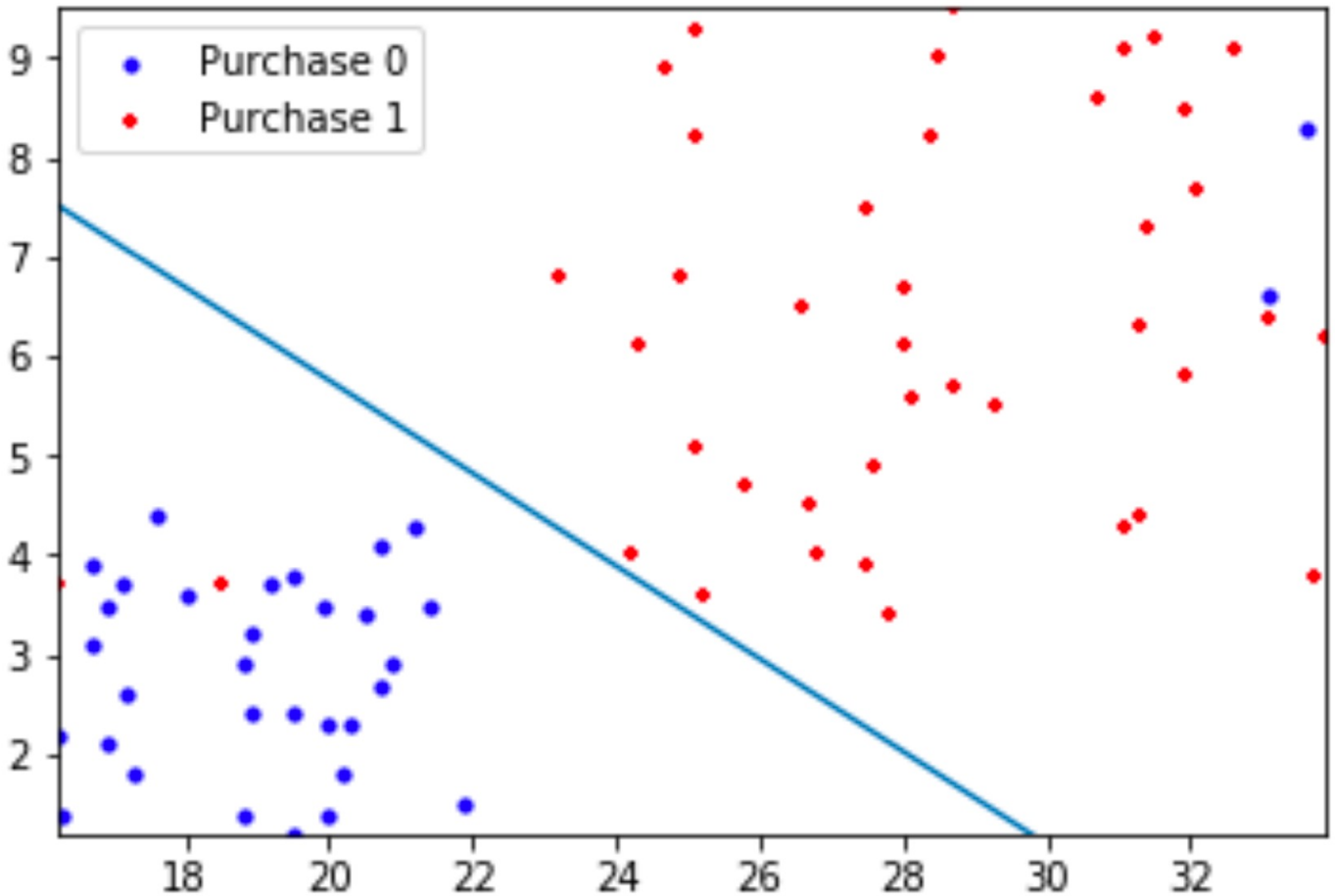
```
import statsmodels.formula.api as sm
model1 = sm.logit(formula='Purchase ~ Age+Experience', data=Emp_Purchase1)
fitted1 = model1.fit()
fitted1.summary2()

#######Accuracy and error of the model1
#Create the confusion matrix
predicted_values=fitted1.predict(Emp_Purchase1[["Age"]+["Experience"]])
predicted_values[1:10]
threshold=0.5

import numpy as np
predicted_class=np.zeros(predicted_values.shape)
predicted_class[predicted_values>threshold]=1

predicted_class

from sklearn.metrics import confusion_matrix as cm
ConfusionMatrix = cm(Emp_Purchase1[['Purchase']],predicted_class)
print(ConfusionMatrix)
accuracy=(ConfusionMatrix[0,0]+ConfusionMatrix[1,1])/sum(sum(ConfusionMatrix))
print('Accuracy : ',accuracy)
error=1-accuracy
print('Error: ',error)
```

```
[[31  2]
 [ 2 39]]
Accuracy :  0.945945945946
Error:  0.054054054054
```

Code: Logistic Regression

```
slope1=fitted1.params[1]/(-fitted1.params[2])
intercept1=fitted1.params[0]/(-fitted1.params[2])


#Finally draw the decision boundary for this logistic regression model

import matplotlib.pyplot as plt

fig = plt.figure()
ax1 = fig.add_subplot(111)

ax1.scatter(Emp_Purchase1.Age[Emp_Purchase1.Purchase==0],Emp_Purchase1.Experience[Emp_Purchase1.Purchase==0], s=10,
arker="o", label='Purchase 0')
ax1.scatter(Emp_Purchase1.Age[Emp_Purchase1.Purchase==1],Emp_Purchase1.Experience[Emp_Purchase1.Purchase==1], s=10,
arker="+", label='Purchase 1')

plt.xlim(min(Emp_Purchase1.Age), max(Emp_Purchase1.Age))
plt.ylim(min(Emp_Purchase1.Experience), max(Emp_Purchase1.Experience))
plt.legend(loc='upper left');

x_min, x_max = ax1.get_xlim()
ax1.plot([0, x_max], [intercept1, x_max*slope1+intercept1])
plt.show()
```

Code: Logistic Regression



4 Sub Theories

1. Every logistic regression line gives us a decision boundary. It looks like a straight line between class-0 and class-1

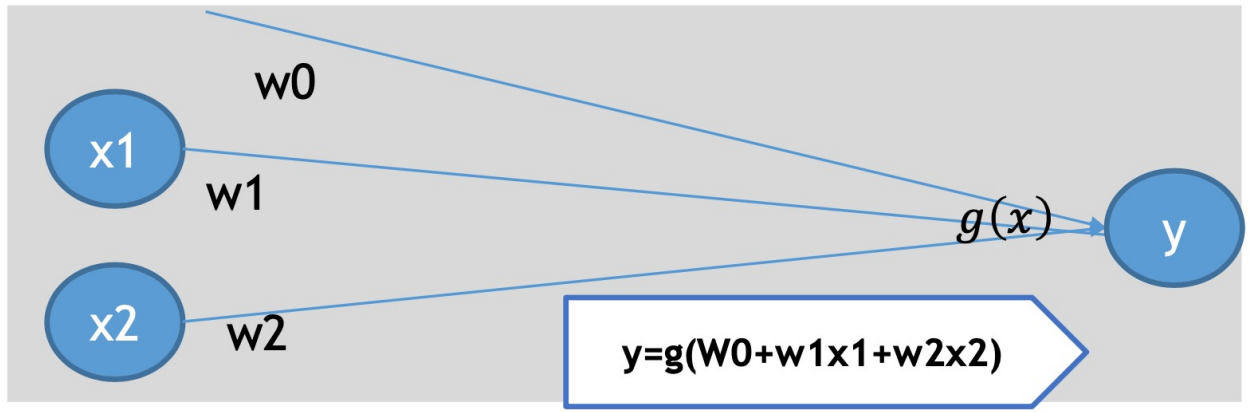
New representation for logistic regression

New representation for logistic regression

$$y = \frac{e^{(\beta_0+\beta_1 x_1+\beta_2 x_2)}}{1+e^{(\beta_0+\beta_1 x_1+\beta_2 x_2)}}$$

$$y = \frac{e^{(w_0+w_1 x_1+w_2 x_2)}}{1+e^{(w_0+w_1 x_1+w_2 x_2)}}$$

$$y = g(w_0 + w_1 x_1 + w_2 x_2) \quad Where\ g(x) = \frac{e^x}{1+e^x}$$



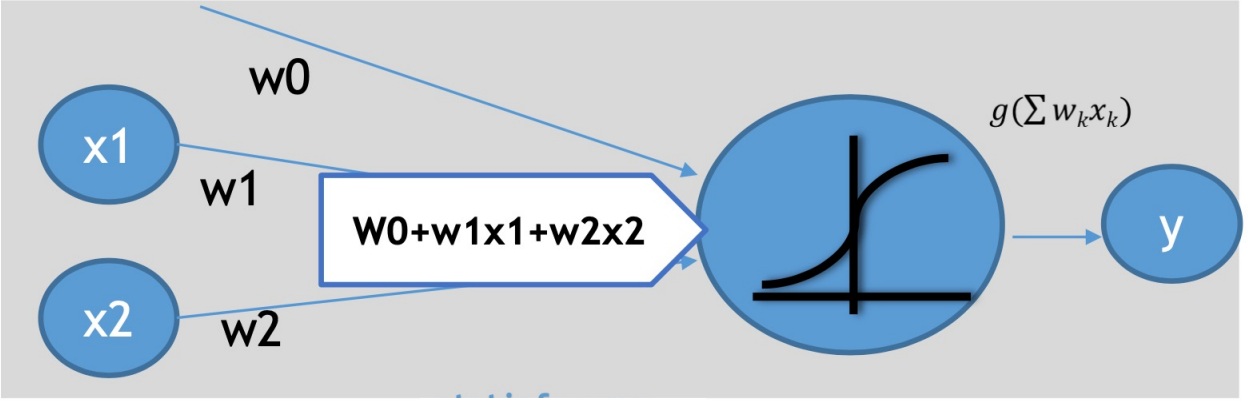
New representation for logistic regression

$$y = \frac{e^{\beta 0+\beta 1x1+\beta 2x2}}{1+e^{\beta 0+\beta 1x1+\beta 2x2}}$$

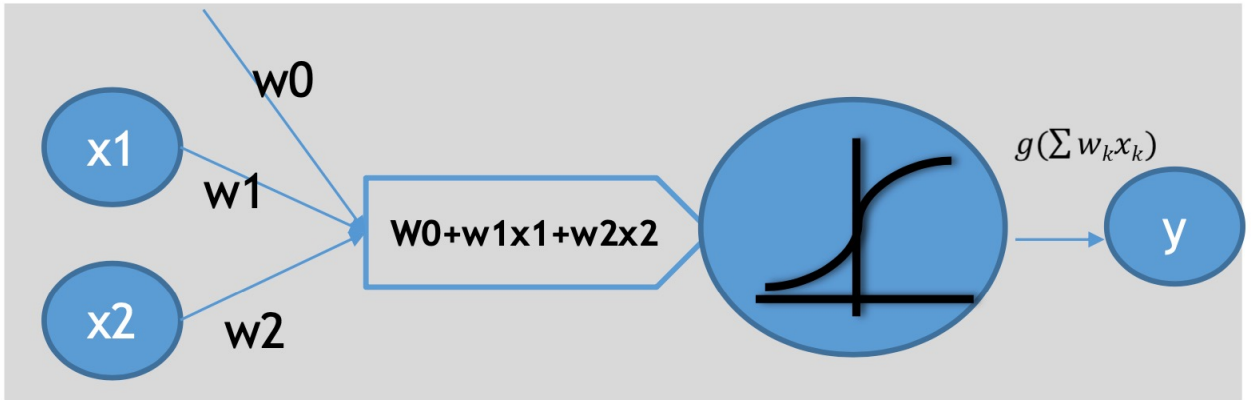
$$y = \frac{1}{1+e^{-(\beta 0+\beta 1x1+\beta 2x2)}}$$

$$y = g(w_0 + w_1x_1 + w_2x_2) \;\; where \;\; g(x) = \frac{1}{1+e^{-x}}$$

$$y = g(\sum w_k x_k)$$



Finding the weights in logistic regression



$$out(x) = g(\sum w_k x_k)$$

The above output is a non linear function of linear combination of inputs – A typical multiple logistic regression line

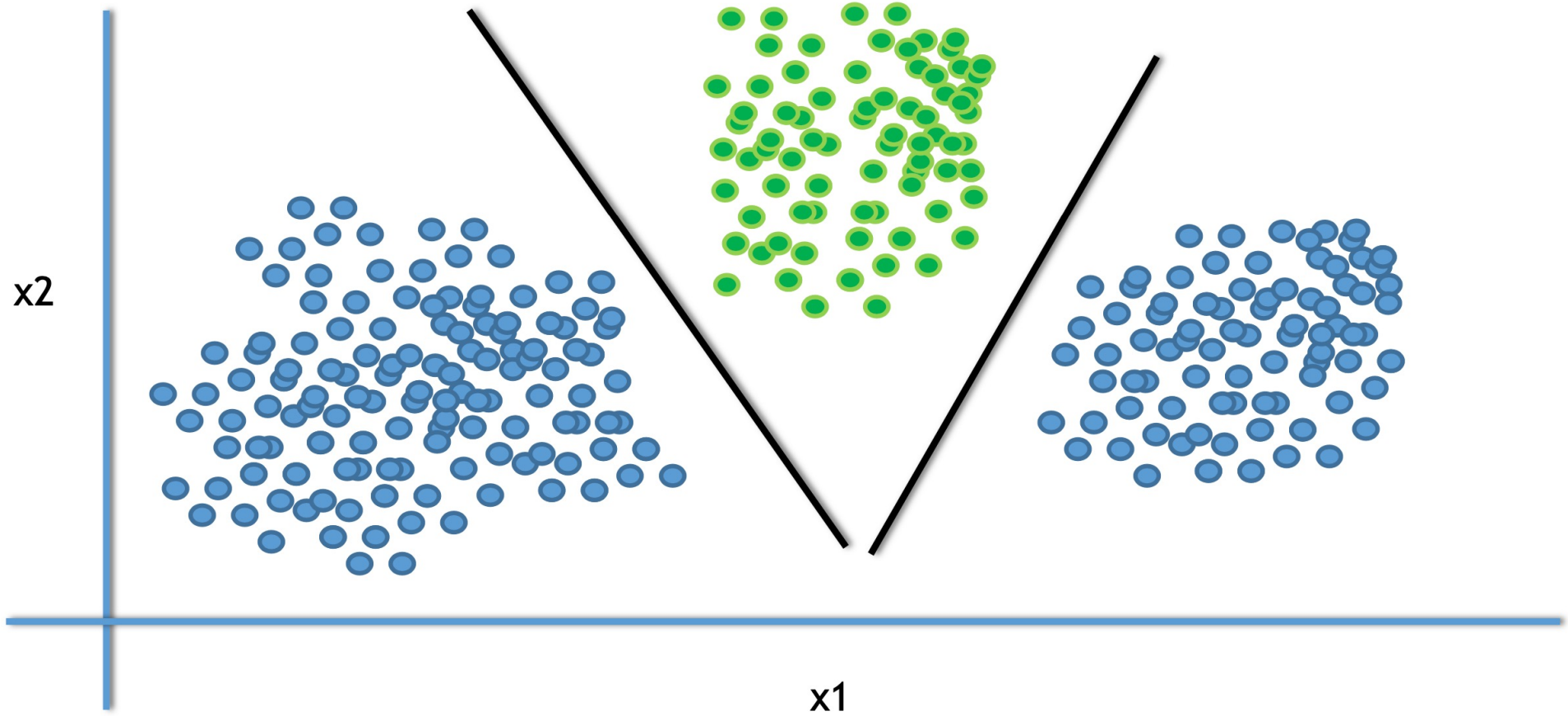
$$We\ find\ \ w\ to\ minimize\ \sum_{i=1}^{n}[y_i - g(\sum w_k x_k)]^2$$

4 Sub Theories

1. Every logistic regression line gives us a decision boundary. It looks like a straight line between class-0 and class-1
2. We are trying to find w's by minimizing error. While building any model

Multiple / Non-Linear Decision Boundaries-Issue

Multiple /Non-Linear Decision Boundaries



Multiple /Non-Linear Decision Boundaries issues

- Logistic Regression line doesn’t seam to be a good option when we have non-linear decision boundaries

LAB: Non-Linear Decision Boundaries

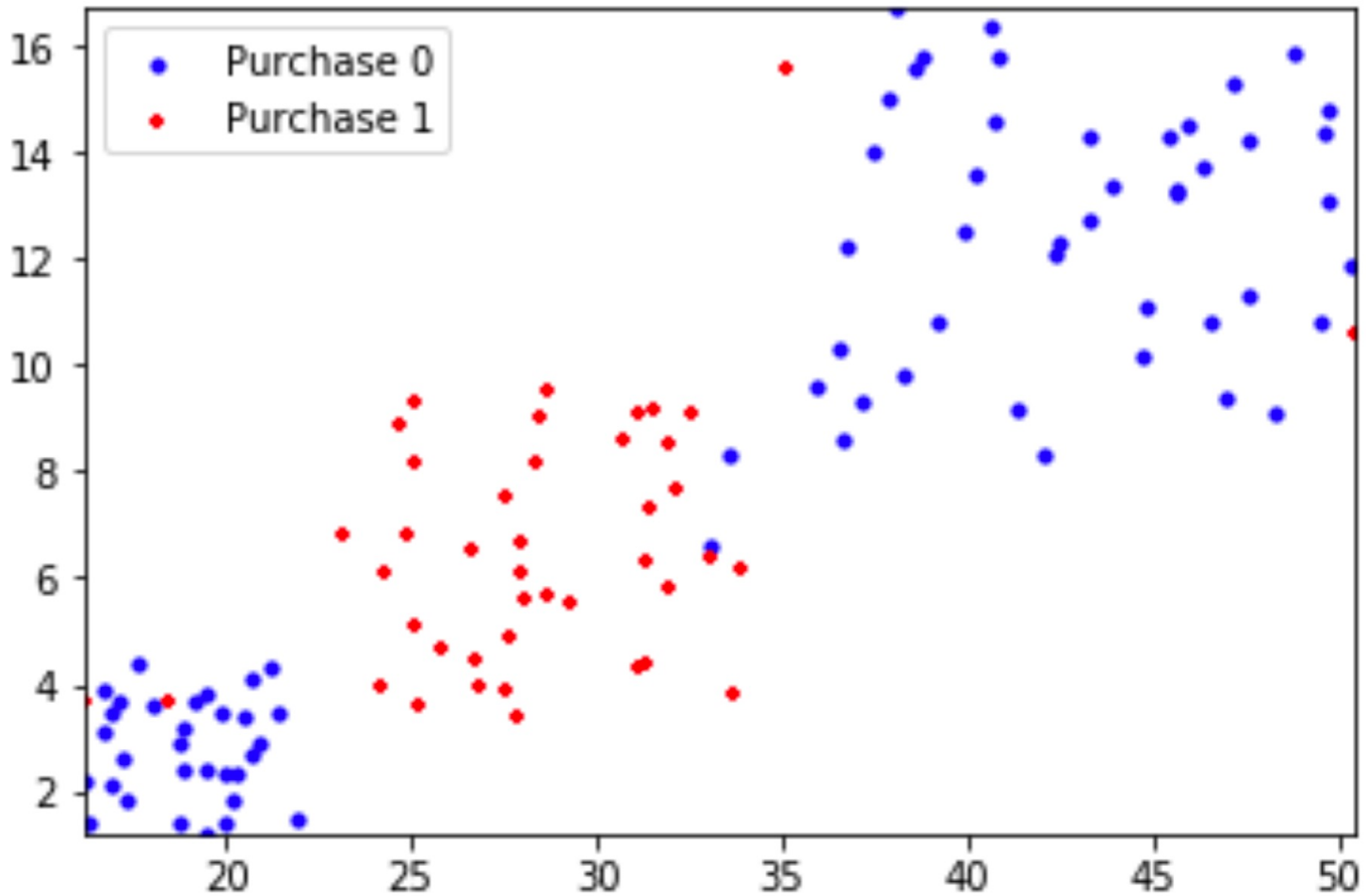
LAB: Non-Linear Decision Boundaries

- Dataset: “Emp_Purchase/ Emp_Purchase.csv”
- Draw a scatter plot that shows Age on X axis and Experience on Y-axis. Try to distinguish the two classes with colors or shapes (visualizing the classes)
- Build a logistic regression model to predict Purchases using age and experience
- Finally draw the decision boundary for this logistic regression model
- Create the confusion matrix
- Calculate the accuracy and error rates

LAB: Non-Linear Decision Boundaries

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(111)

ax.scatter(Emp_Purchase_raw.Age[Emp_Purchase_raw.Purchase==0],Emp_Purchase_raw.Experience[Emp_Purcha
se_raw.Purchase==0], s=10, c='b', marker="o", label='Purchase 0')
ax.scatter(Emp_Purchase_raw.Age[Emp_Purchase_raw.Purchase==1],Emp_Purchase_raw.Experience[Emp_Purcha
se_raw.Purchase==1], s=10, c='r', marker="+", label='Purchase 1')

plt.xlim(min(Emp_Purchase_raw.Age), max(Emp_Purchase_raw.Age))
plt.ylim(min(Emp_Purchase_raw.Experience), max(Emp_Purchase_raw.Experience))
plt.legend(loc='upper left');
plt.show()
```

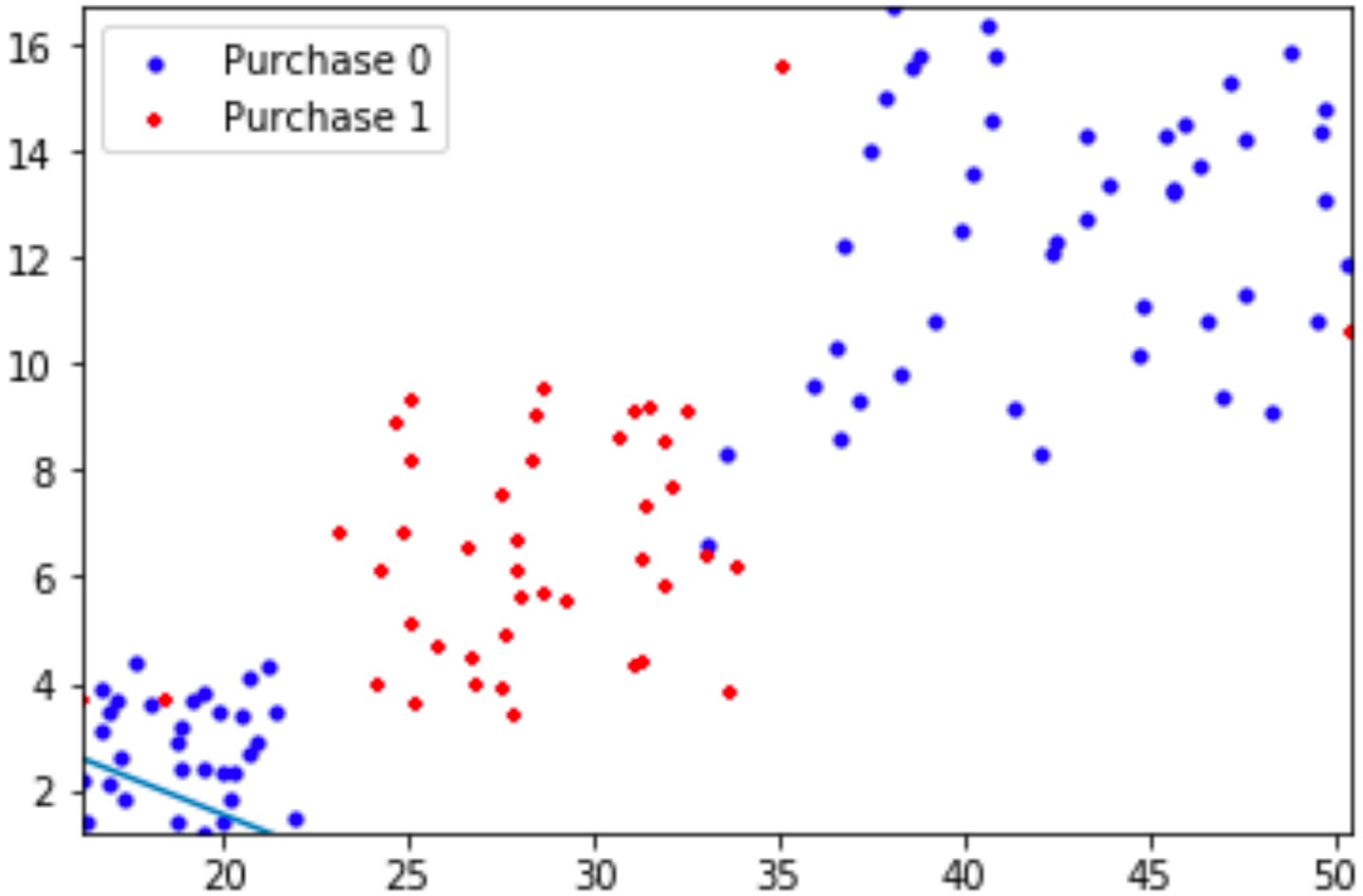
LAB: Non-Linear Decision Boundaries



Code: Non-Linear Decision Boundaries

```
###Logistic Regression model1
import statsmodels.formula.api as sm
model = sm.logit(formula='Purchase ~ Age+Experience', data=Emp_Purchase_raw)
fitted = model.fit()
fitted.summary2()

# getting slope and intercept of the line
slope=fitted.params[1]/(-fitted.params[2])
intercept=fitted.params[0]/(-fitted.params[2])

#Finally draw the decision boundary for this logistic regression model
fig = plt.figure()
ax = fig.add_subplot(111)

ax.scatter(Emp_Purchase_raw.Age[Emp_Purchase_raw.Purchase==0],Emp_Purchase_raw.Experience[Emp_Purchase_raw.Purchase==0], s=10,
c='b', marker="o", label='Purchase 0')
ax.scatter(Emp_Purchase_raw.Age[Emp_Purchase_raw.Purchase==1],Emp_Purchase_raw.Experience[Emp_Purchase_raw.Purchase==1], s=10,
c='r', marker="+", label='Purchase 1')
plt.xlim(min(Emp_Purchase_raw.Age), max(Emp_Purchase_raw.Age))
plt.ylim(min(Emp_Purchase_raw.Experience), max(Emp_Purchase_raw.Experience))
plt.legend(loc='upper left');

x_min, x_max = ax.get_xlim()
ax.plot([0, x_max], [intercept, x_max*slope+intercept])
plt.show()
```

Code: Non-Linear Decision Boundaries



Code: Non-Linear Decision Boundaries

```
#######Accuracy and error of the model1
#Create the confusion matrix
#predicting values
predicted_values=fitted.predict(Emp_Purchase_raw[["Age"]+["Experience"]])
predicted_values[1:10]

#Lets convert them to classes using a threshold
threshold=0.5
threshold

import numpy as np
predicted_class=np.zeros(predicted_values.shape)
predicted_class[predicted_values>threshold]=1

#Predcited Classes
predicted_class[1:10]

from sklearn.metrics import confusion_matrix as cm
ConfusionMatrix = cm(Emp_Purchase_raw[['Purchase']],predicted_class)
print(ConfusionMatrix)
accuracy=(ConfusionMatrix[0,0]+ConfusionMatrix[1,1])/sum(sum(ConfusionMatrix))
print(accuracy)
error=1-accuracy
```

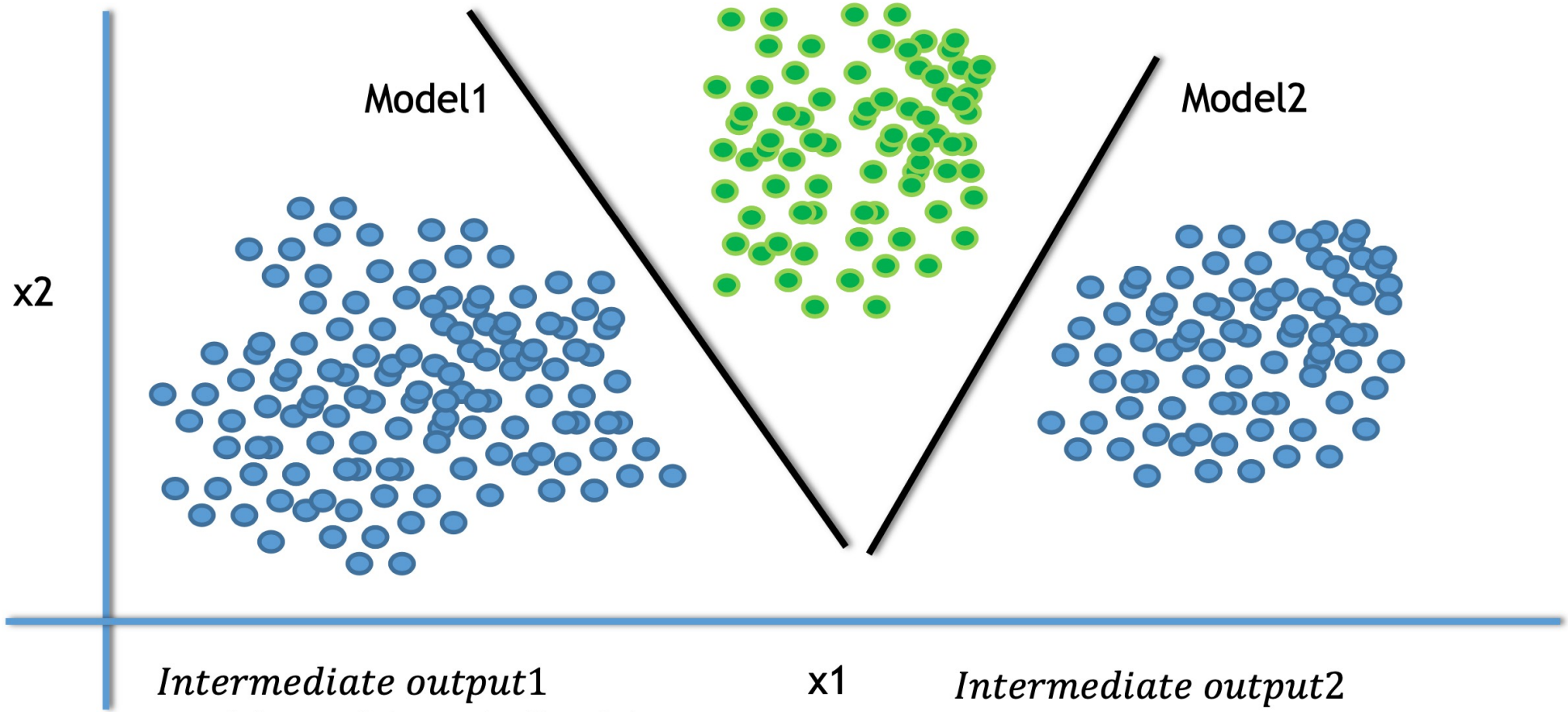
```
[[69  7]
 [43  0]]
0.579831932773
```

4 Sub Theories

1. Every logistic regression line gives us a decision boundary. It looks like a straight line between class-0 and class-1
2. We are trying to find w's by minimizing error. While building any model
3. Logistic Regression Line fails in case of multiple/ non linear decision boundaries

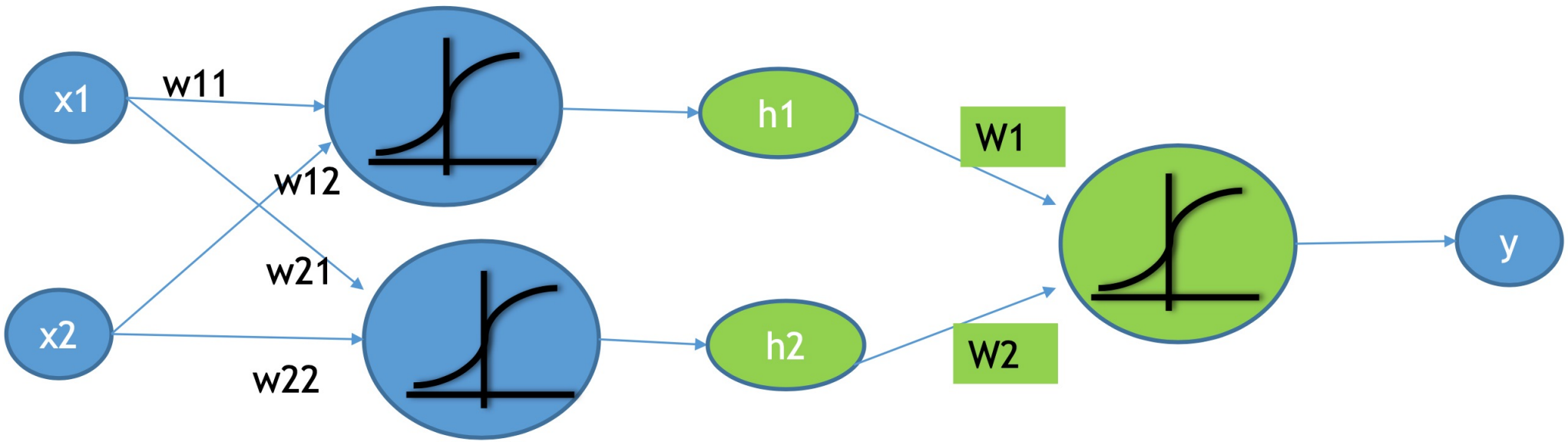
Non-Linear Decision Boundaries-Solution

Intermediate outputs



The Intermediate output

- Using the x's Directly predicting y is challenging.
- We can predict h, the intermediate output, which will indeed predict Y

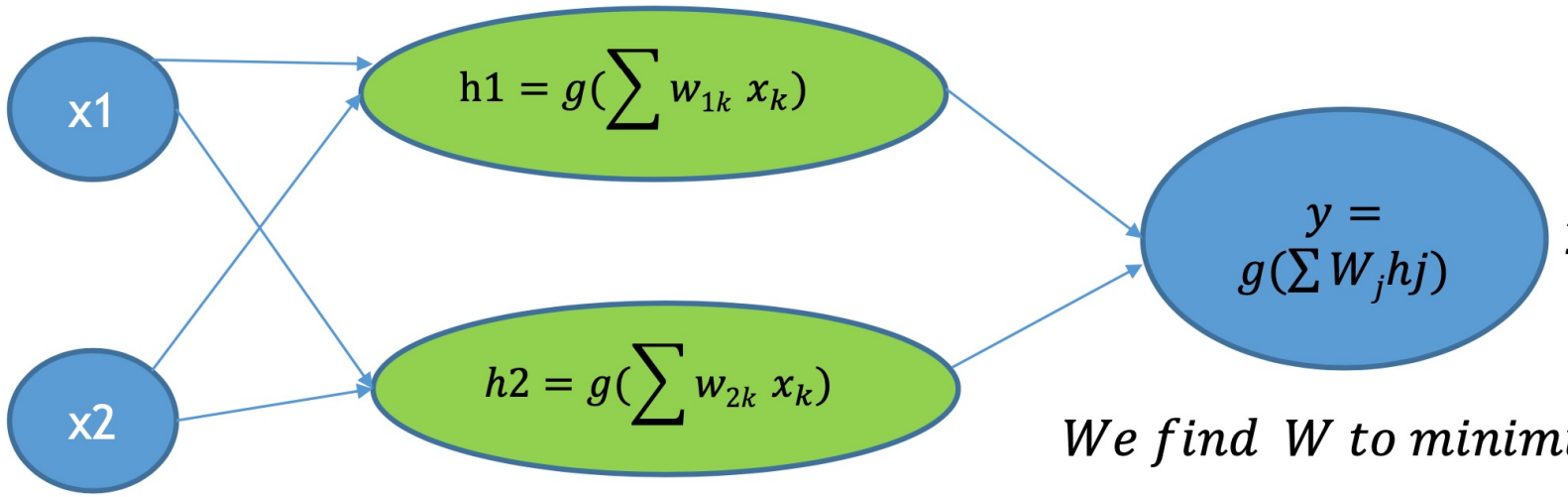


Finding the weights for intermediate outputs

*Intermediate output*1

$h1 = out(x) = g(\sum w_{1k}\, x_k)$

$We\ find\ \ w_1\ to\ minimize\ \sum_{i=1}^{n}[h_{1i} - g(\sum w_{1k}\, x_k)]^2$



Final output

$y = out(h) = g(\sum W_j hj)$

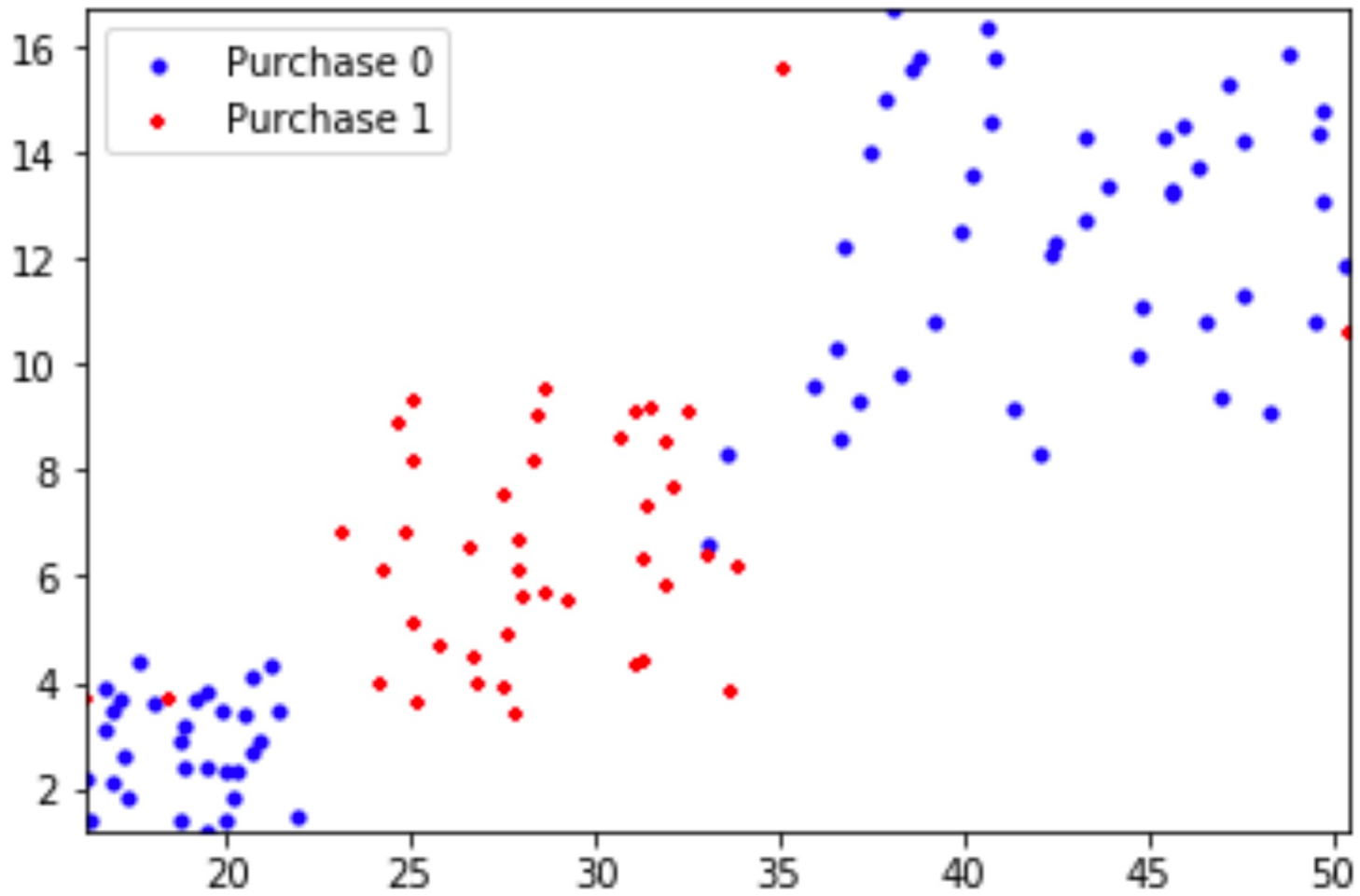
$We\ find\ \ W\ to\ minimize\ \sum_{i=1}^{n}[y_i - g(\sum W_j h_{ji})]^2$

*Intermediate output*2

$h2 = out(x) = g(\sum w_{2k}\, x_k)$

$We\ find\ \ w_2\ to\ minimize\ \sum_{i=1}^{n}[h_{2i} - g(\sum w_{1k}\, x_k)]^2$

Non-Linear Decision Boundaries



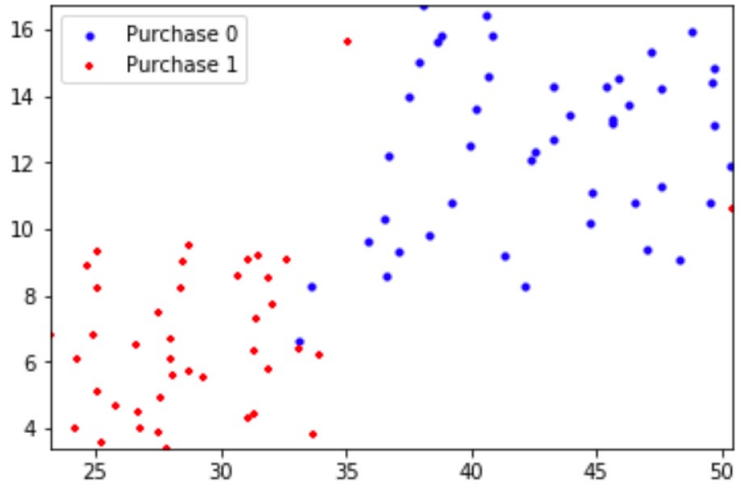
LAB: Intermediate output

LAB: Intermediate output

- Dataset: Emp_Purchase/ Emp_Purchase.csv
- Filter the data and take first 74 observations from above dataset .
- Build a logistic regression model to predict purchases using age and experience
- Calculate the prediction probabilities for all the inputs. Store the probabilities in inter1 variable
- Filter the data and take observations from row 34 onwards.
- Build a logistic regression model to predict purchases using age and experience
- Calculate the prediction probabilities for all the inputs. Store the probabilities in inter2 variable
- Build a consolidated model to predict purchases using inter-1 and inter-2 variables
- Create the confusion matrix and find the accuracy and error rates for the consolidated model

Code: Intermediate output

```
Emp_Purchase2=Emp_Purchase_raw[Emp_Purchase_raw.Sample_Set>1]
Emp_Purchase2.shape
Emp_Purchase2.columns.values
Emp_Purchase2.head(10)

#frequency table of Purchase variable
Emp_Purchase2.Purchase.value_counts()

####The clasification graph
#Draw a scatter plot that shows Age on X axis and Experience on Y-axis. Tr
with colors or shapes.
import matplotlib.pyplot as plt

fig = plt.figure()
ax2 = fig.add_subplot(111)

ax2.scatter(Emp_Purchase2.Age[Emp_Purchase2.Purchase==0],Emp_Purchase2.Experience[Emp_Purchase2.Purchase==0],
s=10, c='b', marker="o", label='Purchase 0')
ax2.scatter(Emp_Purchase2.Age[Emp_Purchase2.Purchase==1],Emp_Purchase2.Experience[Emp_Purchase2.Purchase==1],
s=10, c='r', marker="+", label='Purchase 1')
plt.xlim(min(Emp_Purchase2.Age), max(Emp_Purchase2.Age))
plt.ylim(min(Emp_Purchase2.Experience), max(Emp_Purchase2.Experience))
plt.legend(loc='upper left');
plt.show()
```



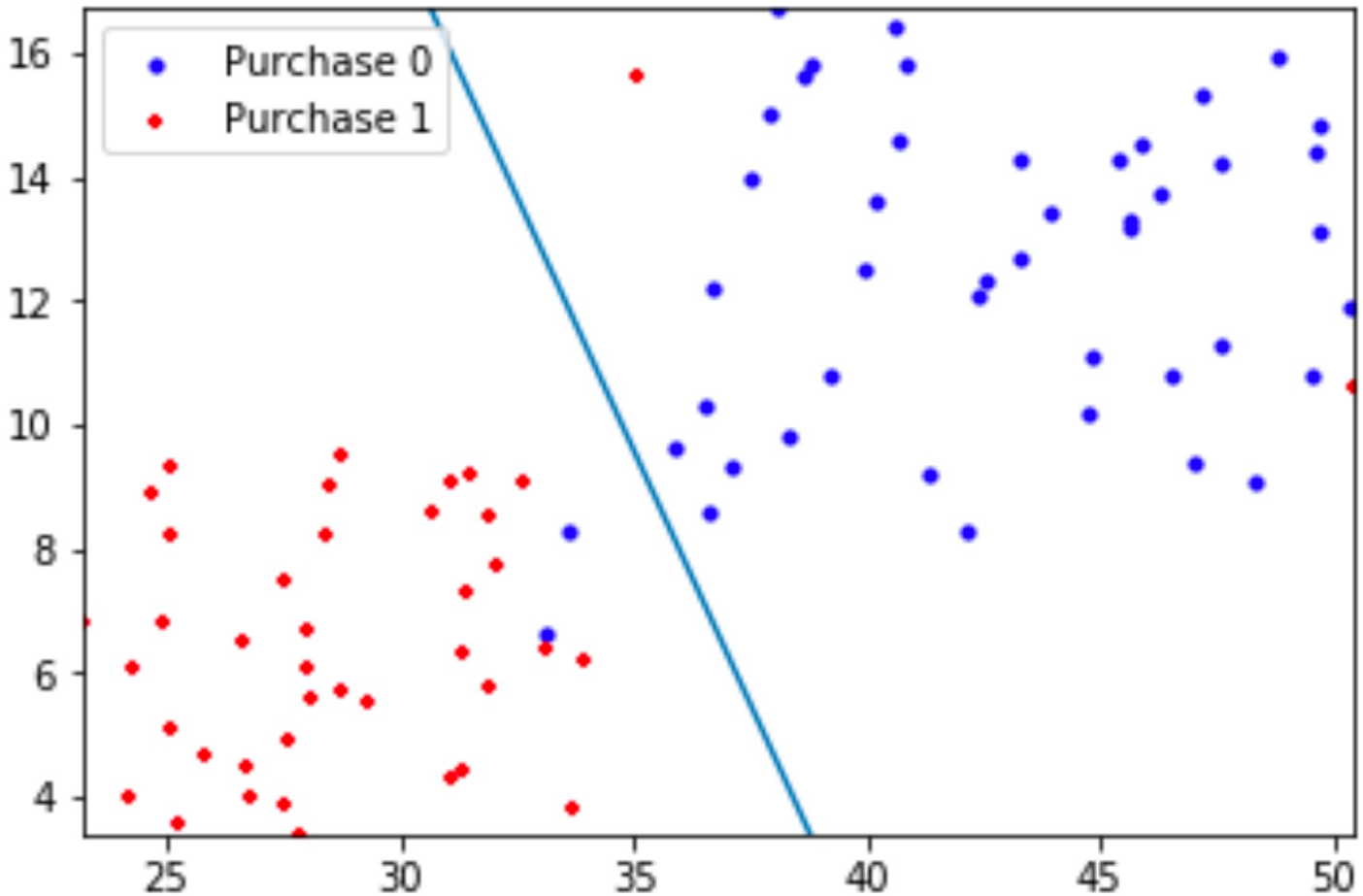
Code: Intermediate output

```
import statsmodels.formula.api as sm
model2 = sm.logit(formula='Purchase ~ Age+Experience', data=Emp_Purchase2)
fitted2 = model2.fit(method="bfgs")
fitted2.summary2()
```

Code: Intermediate output

```
# getting slope and intercept of the line
# getting slope and intercept of the line
slope2=fitted2.params[1]/(-fitted2.params[2])
intercept2=fitted2.params[0]/(-fitted2.params[2])

import matplotlib.pyplot as plt

fig = plt.figure()
ax2 = fig.add_subplot(111)

ax2.scatter(Emp_Purchase2.Age[Emp_Purchase2.Purchase==0],Emp_Purchase2.Experience[Emp_Purchase2.Purchase==0],
s=10, c='b', marker="o", label='Purchase 0')
ax2.scatter(Emp_Purchase2.Age[Emp_Purchase2.Purchase==1],Emp_Purchase2.Experience[Emp_Purchase2.Purchase==1],
s=10, c='r', marker="+", label='Purchase 1')
plt.xlim(min(Emp_Purchase2.Age), max(Emp_Purchase2.Age))
plt.ylim(min(Emp_Purchase2.Experience), max(Emp_Purchase2.Experience))
plt.legend(loc='upper left');

x_min, x_max = ax2.get_xlim()
y_min,y_max=ax2.get_ylim()
ax2.plot([x_min, x_max], [x_min*slope2+intercept2, x_max*slope2+intercept2])
plt.show()
```

Code: Intermediate output



Code: Intermediate output

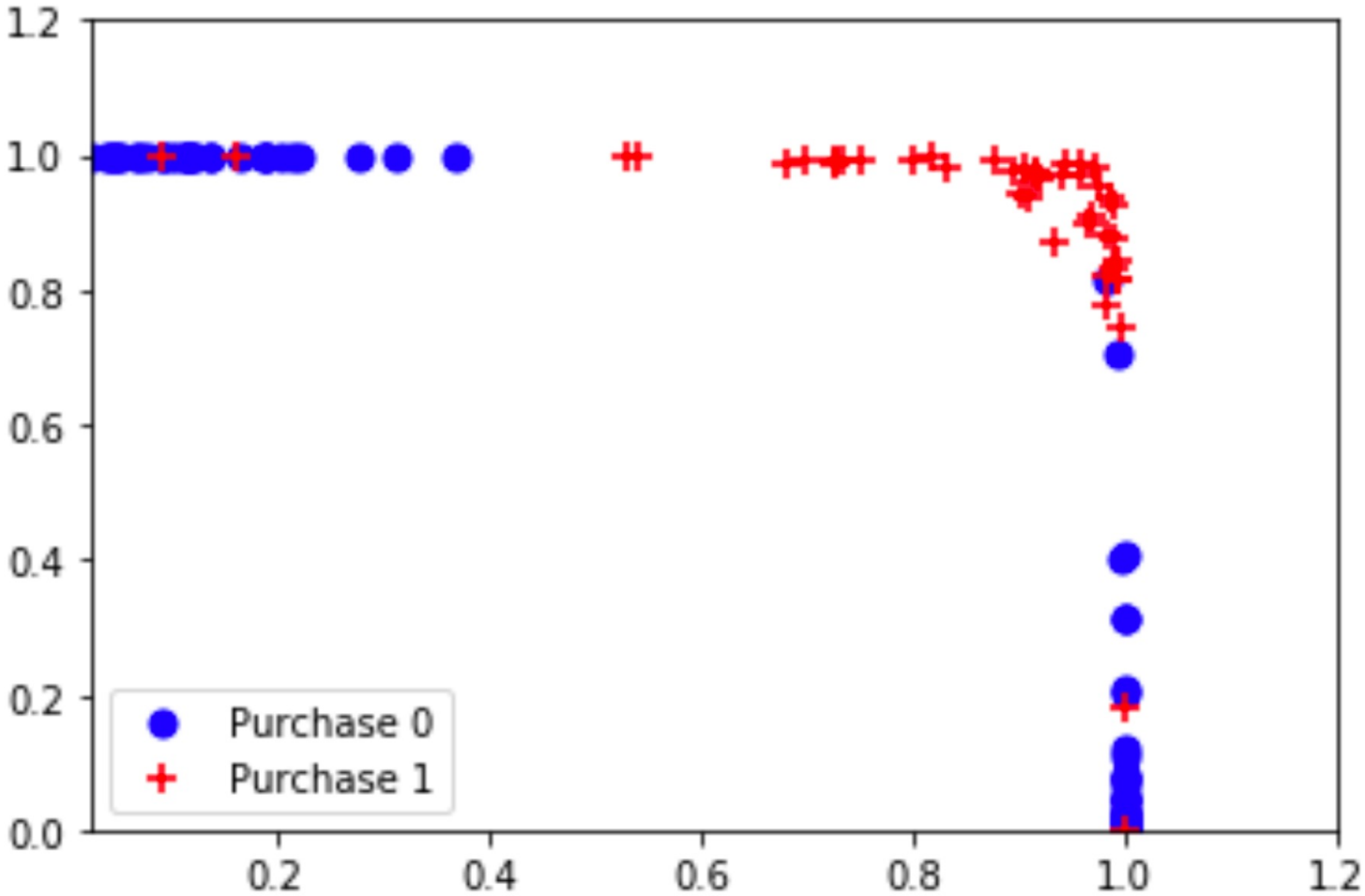
```
#######Accuracy and error of the model1
#Create the confusion matrix
#Predciting Values
predicted_values=fitted2.predict(Emp_Purchase2[["Age"]+["Experience"]])
predicted_values[1:10]

#Lets convert them to classes using a threshold
threshold=0.5
threshold

import numpy as np
predicted_class=np.zeros(predicted_values.shape)
predicted_class[predicted_values>threshold]=1

#Predcited Classes
predicted_class[1:10]

from sklearn.metrics import confusion_matrix as cm
ConfusionMatrix = cm(Emp_Purchase2[['Purchase']],predicted_class)
print(ConfusionMatrix)
accuracy=(ConfusionMatrix[0,0]+ConfusionMatrix[1,1])/sum(sum(ConfusionMatrix))
print(accuracy)

error=1-accuracy
error
```

```
[[43  2]
 [ 2 39]]
0.953488372093
```

Code: Intermediate output

```
fitted1.summary2()
fitted2.summary2()

#The two new coloumns
Emp_Purchase_raw['inter1']=fitted1.predict(Emp_Purchase_raw[["Age"]+["Experience"]])
Emp_Purchase_raw['inter2']=fitted2.predict(Emp_Purchase_raw[["Age"]+["Experience"]])

#plotting the new columns
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(111)

ax.scatter(Emp_Purchase_raw.inter1[Emp_Purchase_raw.Purchase==0],Emp_Purchase_raw.inter2[Emp_Purchase_raw.Pur
chase==0], s=50, c='b', marker="o", label='Purchase 0')
ax.scatter(Emp_Purchase_raw.inter1[Emp_Purchase_raw.Purchase==1],Emp_Purchase_raw.inter2[Emp_Purchase_raw.Pur
chase==1], s=50, c='r', marker="+", label='Purchase 1')

plt.xlim(min(Emp_Purchase_raw.inter1), max(Emp_Purchase_raw.inter1)+0.2)
plt.ylim(min(Emp_Purchase_raw.inter2), max(Emp_Purchase_raw.inter2)+0.2)

plt.legend(loc='lower left');
plt.show()
```

Code: Intermediate output



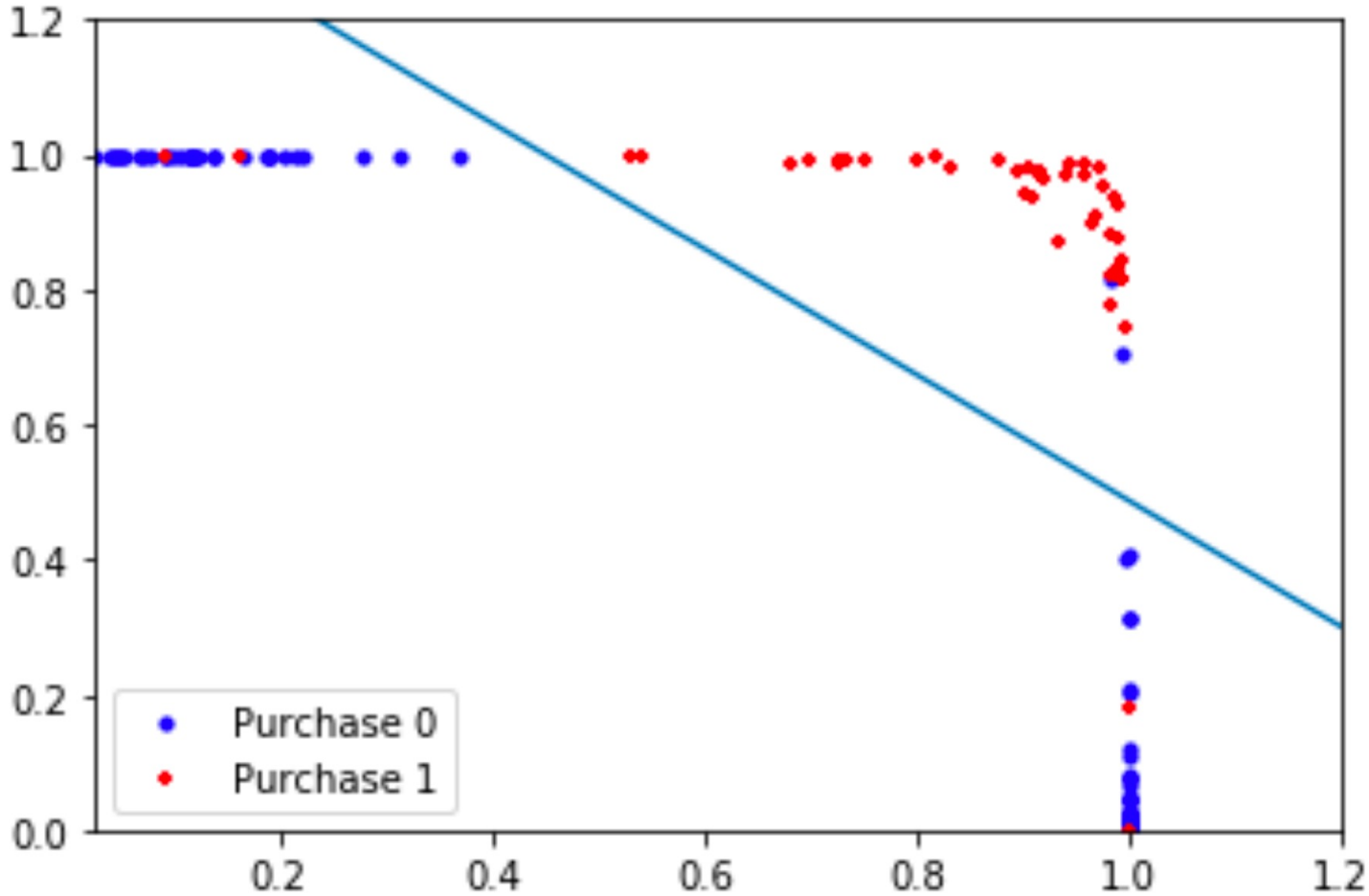
Code: Intermediate output

```
import statsmodels.formula.api as sm

model_combined = sm.logit(formula='Purchase ~ inter1+inter2', data=Emp_Purchase_raw)
fitted_combined = model_combined.fit(method="bfgs")
fitted_combined.summary()

# getting slope and intercept of the line
slope_combined=fitted_combined.params[1]/(-fitted_combined.params[2])
intercept_combined=fitted_combined.params[0]/(-fitted_combined.params[2])
```

Code: Intermediate output

```
#Finally draw the decision boundary for this logistic regression model
import matplotlib.pyplot as plt

fig = plt.figure()
ax2 = fig.add_subplot(111)

ax2.scatter(Emp_Purchase_raw.inter1[Emp_Purchase_raw.Purchase==0],Emp_Purchase_raw.inter2[Emp_Purcha
se_raw.Purchase==0], s=10, c='b', marker="o", label='Purchase 0')
ax2.scatter(Emp_Purchase_raw.inter1[Emp_Purchase_raw.Purchase==1],Emp_Purchase_raw.inter2[Emp_Purcha
se_raw.Purchase==1], s=10, c='r', marker="+", label='Purchase 1')

plt.xlim(min(Emp_Purchase_raw.inter1), max(Emp_Purchase_raw.inter1)+0.2)
plt.ylim(min(Emp_Purchase_raw.inter2), max(Emp_Purchase_raw.inter2)+0.2)

plt.legend(loc='lower left');

x_min, x_max = ax2.get_xlim()
y_min,y_max=ax2.get_ylim()
ax2.plot([x_min, x_max], [x_min*slope_combined+intercept_combined,
x_max*slope_combined+intercept_combined])
plt.show()
```

Code: Intermediate output



Code: Intermediate output

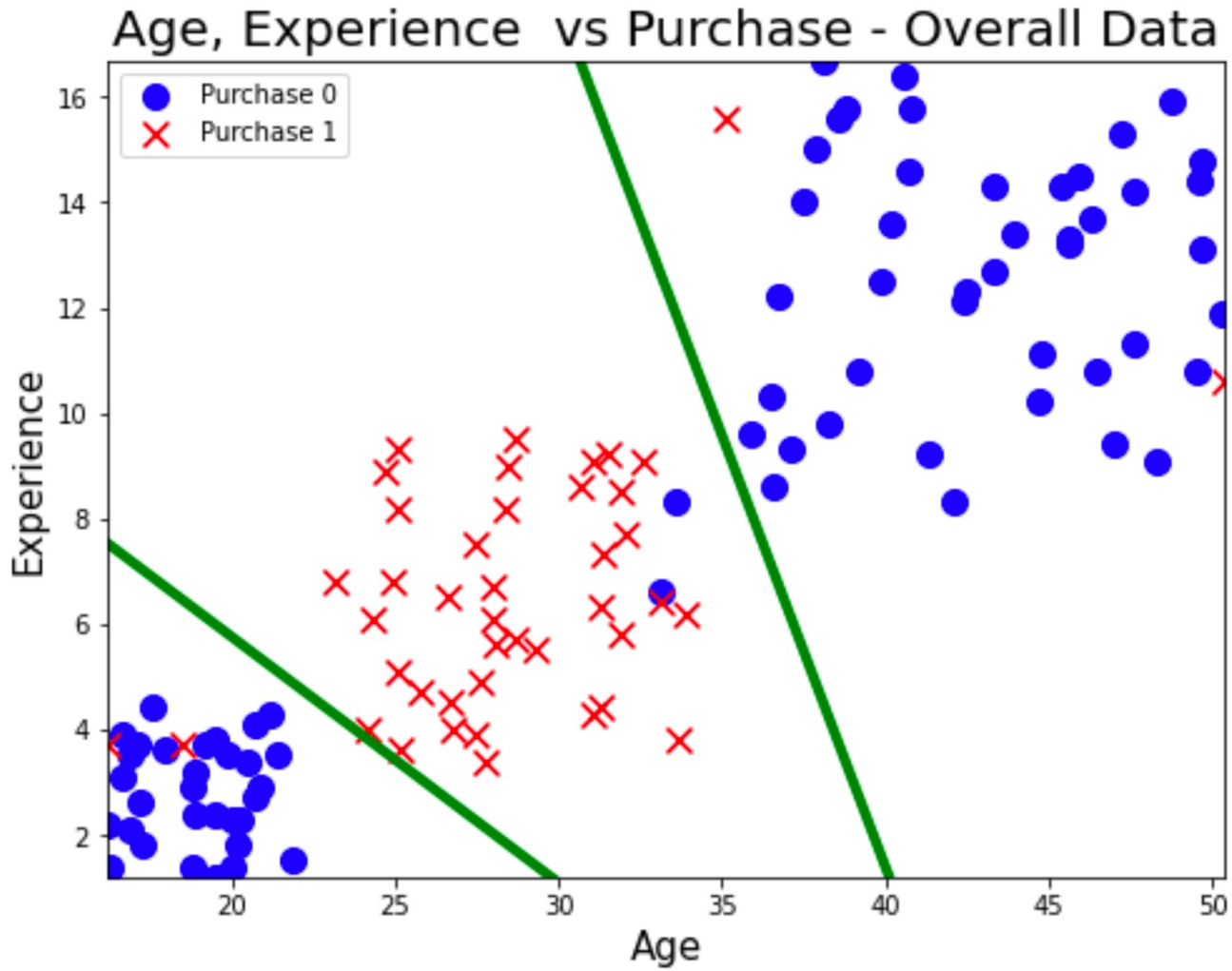
```
#######Accuracy and error of the model1
#Create the confusion matrix
#Predciting Values
predicted_values=fitted_combined.predict(Emp_Purchase_raw[["inter1"]+["inter2"]])
predicted_values[1:10]

#Lets convert them to classes using a threshold
threshold=0.5
threshold

import numpy as np
predicted_class=np.zeros(predicted_values.shape)
predicted_class[predicted_values>threshold]=1

#ConfusionMatrix
from sklearn.metrics import confusion_matrix as cm
ConfusionMatrix = cm(Emp_Purchase_raw[['Purchase']],predicted_class)
print(ConfusionMatrix)
accuracy=(ConfusionMatrix[0,0]+ConfusionMatrix[1,1])/sum(sum(ConfusionMatrix))
print(accuracy)
```

```
[[31  2]
 [ 2 39]]
Accuracy :  0.94594
```

```
[[43  2]
 [ 2 39]]
0.953488372093
```

```
[[74  2]
 [ 4 39]]
0.949579831933
```

Result from Two models



4 Sub Theories

1. Every logistic regression line gives us a decision boundary. It looks like a straight line between class-0 and class-1
2. We are trying to find w's by minimizing error. While building any model
3. Logistic Regression Line fails in case of non linear decision boundaries
4. We used intermediate outputs to solve the problem of non linear decision boundaries