Multi-Class Classification :

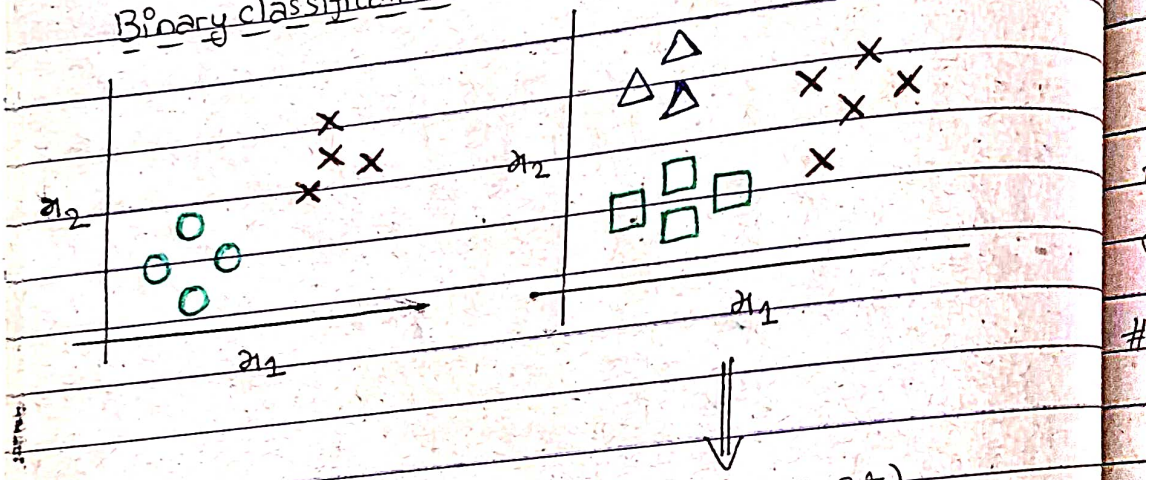
Email folderring / tagging : Works, Friends, Family, hobby

Medical diagrams : Not ill, cold, Flu

$y = 1, 2, 3$

Weather : Sunny, Cloudy, rainy, snow.

Binary classification Multi-class classification.

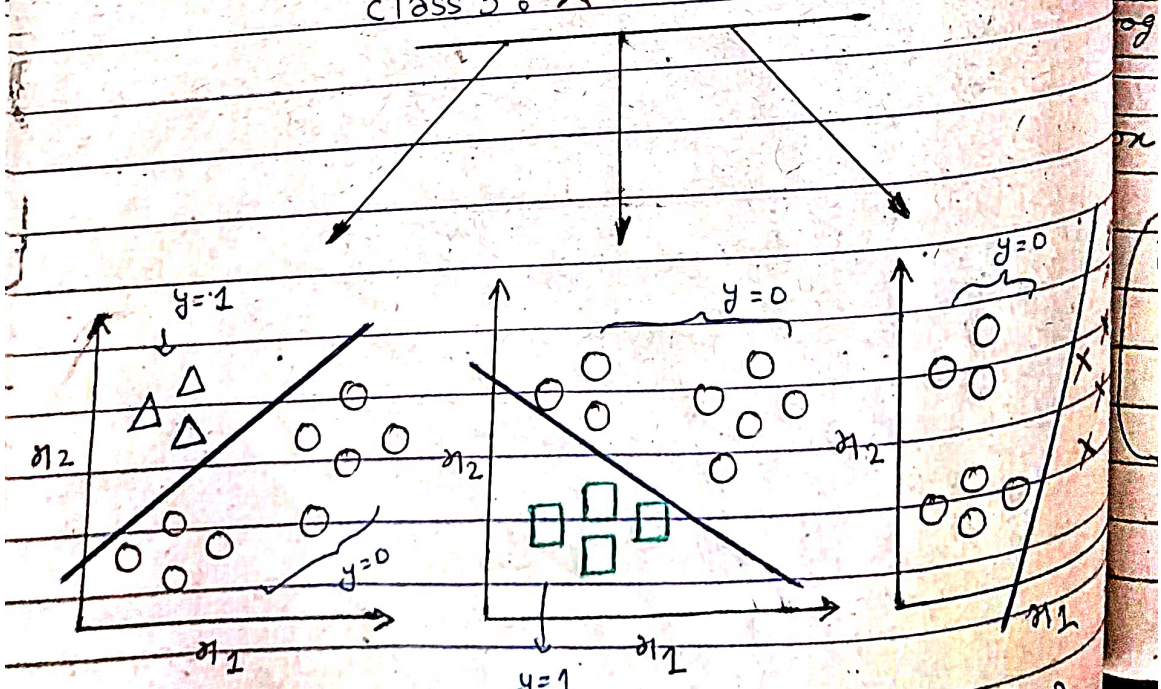


One-vs-all (One-vs-rest)

class 1 : △

class 2 : □

class 3 : ×



$h_\theta^{(1)}(x)$ or $P(y=1|x;\theta)$ $h_\theta^{(2)}(x)$ or $P(y=2|x;\theta)$ $h_\theta^{(3)}(x)$

$h_\theta^{(i)}(x) = P(y=i \mid x;\theta)$ $(i = 1,2,3,\ldots)$

Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y=i$

On a new input x to make a prediction, pick the class i that maximizes

$$\max_i h_\theta^{(i)}(x)$$

C: Cat
D: Dog
F: Fox

true values

y_true = [C, C, C, C, C, C, F, F, F, F, F, F, F, F, F, D, D, D, D, D, D, D, D, D, D, D]

predicted values

y_pred = [C, C, C, C, D, F, C, C, C, C, C, C, D, D, F, F, C, C, C, D, D, D, D, D, D, D]

print confusion matrix

metrics.confusion_matrix(y_true, y_pred)

	Cat	Dog	Fox
Cat	(4)	1	1
Dog	3	(6)	0
Fox	6	2	(2)

Actual

	Cat	Dog	Fox
Cat	4	1	1
Dog	3	6	0
Fox	6	2	2

Predicted

Precision → for each class

$P_{cat} \Rightarrow \frac{4}{4+3+6} \Rightarrow \frac{4}{13} \Rightarrow 0.30$

$P_{dog} \Rightarrow \frac{1}{1+6+2} \Rightarrow \frac{6}{9} \Rightarrow 0.66$

$P_{fox} \Rightarrow \frac{2}{2+0+1} \Rightarrow \frac{2}{3} \Rightarrow 0.66$

macro avg $\Rightarrow \frac{0.30+0.66+0.66}{3} \Rightarrow \frac{1.62}{3} \Rightarrow 0.54$

→ TP ⇒ 12

Accuracy ⇒ $\frac{12}{25}$

⇒ 0.480

Regularization:

```
x = dataset.drop('Price', axis=1)
y = dataset['Price']

from sklearn.model_selection import
                        train_test_split
train_x, test_x, train_y, test_y =
    train_test_split(x, y, test_size=0.3, random_st

from sklearn.model_selection import LinearRegressio
reg = LinearRegression().fit(train_x, train_y
```

reg.score(test_x, test_y)
↳ 0.1385

reg.score(train_x, train_y)
↳ 0.6827

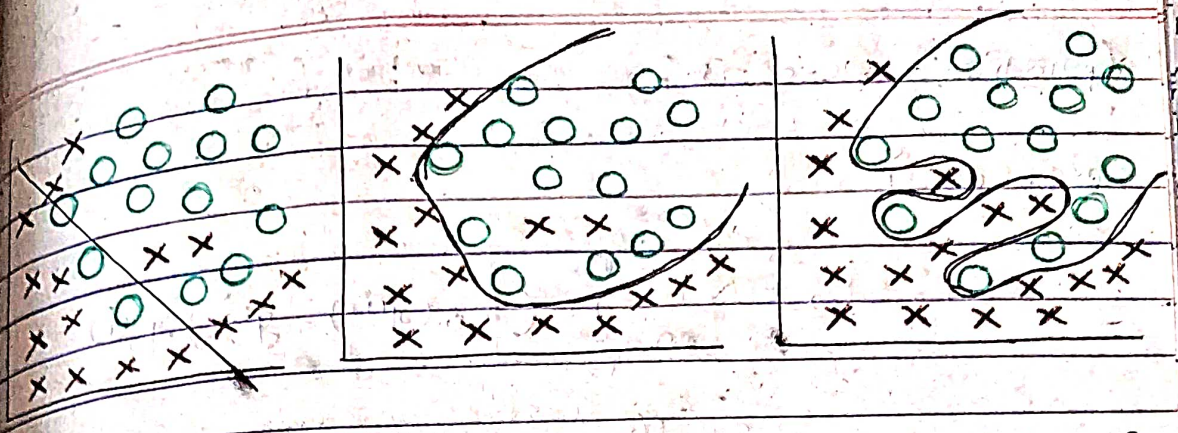
} better on train data but not g on testing
Overfitting

L1 reg: mse $\Rightarrow \frac{1}{n}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{i=1}^{n} |\theta_i|$

$\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \dots + \theta_n x_n$

```
from sklearn import linear_model

lasso_reg = linear_model.Lasso(alpha=50,
            max_iter=100, tol=0.1)
lasso_reg.fit(trainx, trainy)
    ↳ Lasso(max_iter=..., ...)
lass_reg.score(train_x, train_y)
    ↳ 0.176
lasso_reg.score(test_x, test_y)
    ↳ 0.163
```



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

↑ underfit

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$

↑ generalized fit

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_5 x_1^3 x_2 + \dots)$

↑ overfit

Cost function :

$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2 \quad (j = 1, 2, 3, \dots, n)$$

Gradient Descent :

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} - \frac{\lambda}{m} \theta_j \right]$$

$$(j = \{1, 2, 3, \ldots, n\})$$

}

Classification using Logistic Regression :

#Step 1. Importing Libraries

```
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

#Step 2. Generate the dataset

```
x, y = make_classification(
    n_samples = 100,
    n_features = 1,
    n_classes = 2,
    n_cluster_per_class = 1,
    flip_y = 0.03,
    n_informative = 1,
    n_redundant = 0,
    n_repeated = 0,
)
```

x → [[-1.1011]
[-1.001]
[.001]
]]

y → [0 0 0 1]

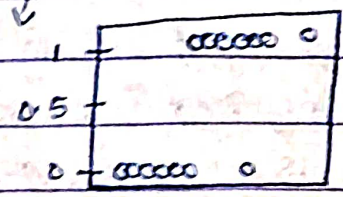
make_classification( ) : create synthetic dataset further to be used for training & analysis.

n_classes = 2 → binary classification

n_cluster_per_class → not for real world data

flip_y = 0.03 → 3% of samples in dataset has their class labels flipped, to add noise & check robustness of dataset to noise.

plt.scatter(x, y, c=y, cmap='rainbow')
plt.title('Scatter plot')
plt.show()



#step4. Split the dataset

X_train, X_test, y_train, y_test = train_test_split
(x, y, random_state = 1)

#Step5. perform logistic regression

log_reg = logisticRegression()
log_reg.fit(X_train, y_train)

#step6. make prediction using model

y_pred = log_reg.predict(X_test)

#Step7. Display the confusion matrix

confusion_matrix(y_test, y_pred)

↓

array([[11, 0]
[0, 14]], dtype = int64)