UNIVERSIDAD MAYOR, REAL Y PONTIFICIA DE SAN FRANCISCO XAVIER DE CHUQUISACA

FACULTAD DE TECNOLOGÍA

INGENIERIA DE SISTEMAS

LABORATORIO 10

ESTUDIANTE: Calle Perez Edwin

CARRERA: INGENIERIA DE SISTEMAS

MATERIA: INTELIGENCIA ARTIFICIAL (SIS 420)

SUCRE – BOLIVIA

Laboratorio 10

Desarrollar lo siguiente:
Descargar un dataset para clasificacion de www.kaggle.com, aplicar el cuadernillo empleado en clase para regresion logistica (clasificacion) identificando, los mejores valores de thetha, el menor valor de la funcion de costo, el numero de iteraciones suficientes para alcanzar los valores ideales de theta, el mejor valor de alfa (coheficiente de aprendizaje) y realizar la grafica del comportamiento del costo en funcion al numero de iteraciones y verificar los resultado con la ecuacion de la normal.

Se debe incluir el codigo y la direccion del repositorio empleado de manera obligatoria.
El dataset de constar de al menos m > 200 y n > 5.

Adecuar Dataset

Se represento los datos de las columnas necesarias a números , en las columnas **Departamentos , location, education ,recruitment_type**:

Departamentos HR=1 Technology=2 Sales=3 Purchasing=4 Marketing=5	**Education** PG=1 UG=0
Location Suburb=1 City=0	**recruitment_type** Referral=1 Walk-in=2

Cargar Datos

calcule la probabilidad de admisión de un solicitante en función de los puntajes de esos dos exámenes.

La siguiente celda cargará los datos y las etiquetas correspondientes:

In [81]:
```
# Cargar datos
# Las columnas de X contienen caracteristicas de una persona en su empleo
#  Y contiene la etiqueta que indica si la persona esta satisfecho con su empleo.
data = np.loadtxt(os.path.join( 'Datasets','Employee Satisfaction Indexo.csv'), delimiter=',')
X, y = data[:, 0:11], data[:, 11]
print(X)
print(y)
```

```
[[2.8000e+01 1.0000e+00 1.0000e+00 ... 1.0000e+00 0.0000e+00 8.6750e+04]
 [5.0000e+01 2.0000e+00 1.0000e+00 ... 2.0000e+00 1.0000e+00 4.2419e+04]
 [4.3000e+01 2.0000e+00 1.0000e+00 ... 2.0000e+00 0.0000e+00 6.5715e+04]
 ...
 [3.4000e+01 5.0000e+00 0.0000e+00 ... 2.0000e+00 0.0000e+00 2.4076e+04]
 [2.6000e+01 2.0000e+00 0.0000e+00 ... 1.0000e+00 1.0000e+00 2.9805e+04]
 [2.6000e+01 2.0000e+00 0.0000e+00 ... 3.0000e+00 0.0000e+00 4.2419e+04]]
[1. 0. 0. 1. 1. 1. 0. 0. 1. 0. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1.
 0. 0. 1. 0. 0. 1. 0. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1. 0.
 1. 0. 1. 0. 0. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 0. 1. 0. 1. 1. 0. 0.
 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0.
 1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1. 1. 0. 1. 1. 0. 0. 1. 0. 0. 0. 0. 0.
 0. 0. 1. 0. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0. 0. 1. 1. 1. 0. 1. 0. 0. 0. 1. 1.
 1. 0. 0. 1. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 1. 0. 1. 1.
 1. 0. 1. 0. 0. 1. 0. 1. 0. 1. 1. 1. 1. 0. 1. 0. 1. 0. 0. 1. 1. 1. 1. 1. 0.
 1. 1. 0. 1. 1. 0. 1. 1. 1. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0.
 0. 0. 0. 1. 0. 1. 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 0. 1. 1. 0. 0. 1. 1. 0.
 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0. 0. 1. 1. 1. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 0.
 0. 1. 0. 1. 1. 1. 1. 0. 0. 1. 0. 0. 1. 1. 0. 1. 1. 1. 1. 0. 0. 0. 1. 0. 1. 0.
 0. 1. 0. 1. 0. 0. 1. 1. 1. 0. 0. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 0. 1. 0.
 0. 1. 1. 0. 1. 0. 1. 0. 0. 1. 0. 0. 1. 1. 0. 1. 1. 0. 0. 1. 1. 1. 1. 0. 0.]
```

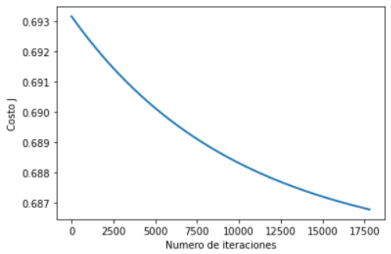
Resultado

```
# Muestra los resultados del descenso por el gradiente
print('theta calculado por el descenso por el gradiente: {:s}'.format(str(theta)))

# verificar si ingresa o no a la universidad

X_array = [1,33,3,0,1,3,4,4,1,3,1,65715]
aprueba = sigmoid(np.dot(X_array, theta))   # Se debe cambiar esto
print(f"resultado : {aprueba}")
#print(f"Un estudiante con nota del examen 1: {X_array[1]} y nota del examen 2: {X_array[2]} (usando el descenso por el gradiente
```

```
theta calculado por el descenso por el gradiente: [ 0.0613568   0.01085097 -0.01245871 -0.03489187  0.03655078  0.02809583
  0.00084621  0.10830895  0.03351339  0.0140236  -0.01298642  0.02433458]
resultado : 1.0
```



Valores de theta

In [71]:
```
# Calcula y muestra el costo y el gradiente con valores de theta diferentes a cero

test_theta = np.array([0.11, 0.03, 0.03, 0.07, 0.07, 0.05, -0.62, 0.19, 0.06, 0.01, -0.03, 0.65])
#test_theta = np.array([-11.74749157, 0.09927308, 0.09316497])
print(test_theta)
cost, grad = costFunction(test_theta, X, y)

print('Costo en theta prueba: {:.3f}'.format(cost))
#print('Costo esperado (aproximado): 0.218\n')

print('Gradiente en theta prueba:')
print('\t[{:.3f}, {:.3f}, {:.3f}]'.format(*grad))
#print('Gradiente esperado (aproximado):\n\t[0.043, 2.566, 2.647]')
```

```
[ 0.11  0.03  0.03  0.07  0.07  0.05 -0.62  0.19  0.06  0.01 -0.03  0.65]
Costo en theta prueba: 0.686
Gradiente en theta prueba:
        [0.001, 0.002, 0.017]
```

Los mejores valores para el theta son [0.11, 0.03, 0.03, 0.07, 0.07, 0.05, -0.62, 0.19, 0.06, 0.01, -0.03, 0.65].

El menor valor de la función costo es: 0.686.

Ecuacion de la normal

2.3 Ecuacion de la Normal

Una manera de calcular rapidamente el modelo de una regresion lineal es:

$$\theta = \left(X^T X\right)^{-1} X^T \vec{y}$$

Utilizando esta formula no requiere que se escale ninguna caracteristica, y se obtendra una solucion exacta con un solo calculo: no hay "bucles de convergencia" como en el descenso por el gradiente.

Primero se recargan los datos para garantizar que las variables no esten modificadas. Recordar que no es necesario escalar las caracteristicas, se debe agregar la columna de unos a la matriz X para tener el termino de intersección(θ_0).

In [72]:
```
# Cargar datos
data = np.loadtxt(os.path.join( 'Datasets','Employee Satisfaction Indexo.csv'), delimiter=',')
X = data[:, :11]
y = data[:, 11]
m = y.size
X = np.concatenate([np.ones((m, 1)), X], axis=1)
```

In [73]:
```
def normalEqn(X, y):

    theta = np.zeros(X.shape[1])

    theta = np.dot(np.dot(np.linalg.inv(np.dot(X.T,X)),X.T),y)

    return theta
```

Resultado con ecuación de la Normal

In [74]:
```
# Calcula los parametros con la ecuación de la normal
theta = normalEqn(X, y);

# Muestra los resultados optenidos a partir de la aplicación de la ecuación de la normal
print('Theta calculado a partir de la ecuación de la normal: {:s}'.format(str(theta)));


X_array = [1,33,3,0,1,3,4,4,1,3,1,65715]
price = np.dot(X_array, theta)

print('Resultado si el empleado esta satisfecho o no {:.0f}'.format(price))
```

```
Theta calculado a partir de la ecuación de la normal: [ 3.61671646e-01  7.28367352e-04 -5.52743844e-03 -3.50934734e-02
  3.65966232e-02  1.02598151e-02 -1.06855114e-01  3.23677976e-02
  3.14249641e-02  8.11918056e-04 -1.25173930e-02  6.71271871e-06]
Resultado si el empleado esta satisfecho o no 1
```

1.2.3 Parámetros de aprendizaje usando `scipy.optimize`

Los valores de theta que Saca la ecuación de la normal son diferentes a los valores ideales de theta que debería tener.

Parámetros de aprendizaje usando `scipy.optimize`¶

```
                      options=options)

# la propiedad fun del objeto devuelto por `OptimizeResult`
# contiene el valor del costFunction de un theta optimizado
cost = res.fun

# Theta optimizada esta en la propiedad x
theta = res.x

# Imprimir theta en la pantalla
print('Costo con un valor de theta encontrado por optimize.minimize: {:.3f}'.format(cost))
print('Costo esperado (aproximado): 0.203\n');

print('theta:')
print('\t[{:.4f}, {:.4f}, {:.4f}, {:.4f}, {:.4f}, {:.4f}, {:.4f}, {:.4f}, {:.4f}, {:.4f}, {:.4f}]'.format(*theta))
#print('Theta esperado (aproximado):\n\t[-25.161, 0.206, 0.201]')
```

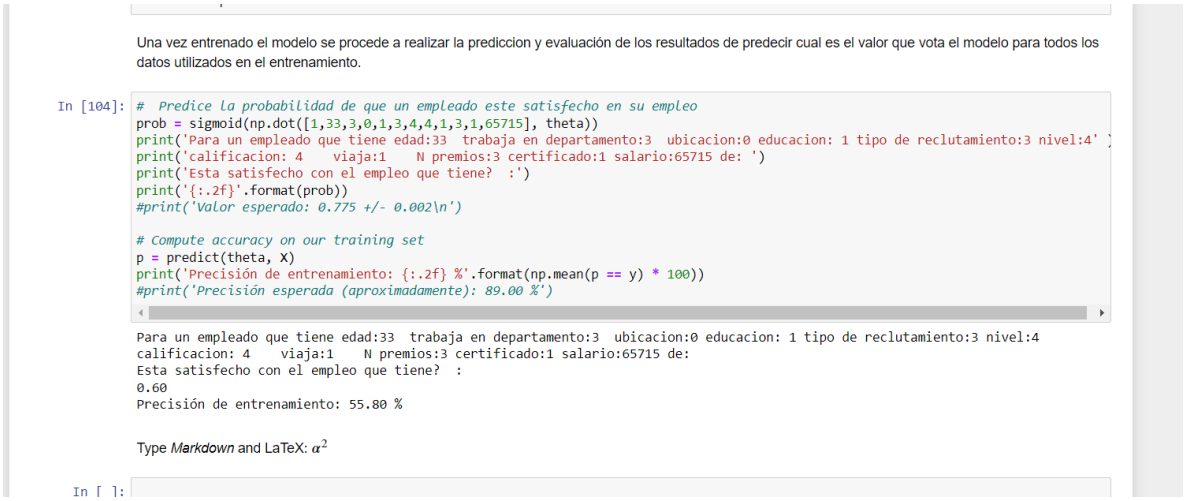
```
Costo con un valor de theta encontrado por optimize.minimize: 0.683
Costo esperado (aproximado): 0.203

theta:
        [-0.5577, 0.0029, -0.0230, -0.1427, 0.1493, 0.0420, -0.4345, 0.1310, 0.1281, 0.0034, -0.0515, 0.0000]
```

Una vez que se completa `optimize.minimize`, se usa el valor final de θ para visualizar el límite de decisión en los datos de entrenamiento.

Para hacerlo, se implementa la función `plotDecisionBoundary` para trazar el límite de decisión sobre los datos de entrenamiento.

Resultado



Una vez entrenado el modelo se procede a realizar la prediccion y evaluación de los resultados de predecir cual es el valor que vota el modelo para todos los datos utilizados en el entrenamiento.

In [104]:
```
#  Predice la probabilidad de que un empleado este satisfecho en su empleo
prob = sigmoid(np.dot([1,33,3,0,1,3,4,4,1,3,1,65715], theta))
print('Para un empleado que tiene edad:33  trabaja en departamento:3  ubicacion:0 educacion: 1 tipo de reclutamiento:3 nivel:4' )
print('calificacion: 4    viaja:1    N premios:3 certificado:1 salario:65715 de: ')
print('Esta satisfecho con el empleo que tiene?  :')
print('{:.2f}'.format(prob))
#print('Valor esperado: 0.775 +/- 0.002\n')

# Compute accuracy on our training set
p = predict(theta, X)
print('Precisión de entrenamiento: {:.2f} %'.format(np.mean(p == y) * 100))
#print('Precisión esperada (aproximadamente): 89.00 %')
```

```
Para un empleado que tiene edad:33  trabaja en departamento:3  ubicacion:0 educacion: 1 tipo de reclutamiento:3 nivel:4
calificacion: 4    viaja:1    N premios:3 certificado:1 salario:65715 de:
Esta satisfecho con el empleo que tiene?  :
0.60
Precisión de entrenamiento: 55.80 %
```

Type *Markdown* and LaTeX: α^2

In [ ]:

Se hizo la predicción con Descenso por el gradiente, ecuación de la Normal y Parámetros de aprendizaje usando scipy.optimize.

Predijeron de igual forma que el empleado esta satisfecho con el trabajo que tiene.