**Practical 6**

**6 a)Aim:** **Implementing regularization to avoid overfitting in binary**

**classification.**

**Description:**

Regularization is a technique used to prevent overfitting in binary classification models by adding a penalty to the loss function, discouraging the model from fitting the noise in the training data. Common regularization methods include L1 (Lasso) and L2 (Ridge) regularization, which add the absolute values or squared values of the weights to the loss function, respectively. Another effective method is dropout, where randomly selected neurons are ignored during training, preventing the model from becoming overly reliant on specific neurons. Additionally, early stopping can be used, which halts training once the model's performance on a validation set stops improving, preventing overfitting by limiting the number of training epochs. These techniques help improve the generalization capability of the model, ensuring better performance on unseen data.

**Code:**

from matplotlib import pyplot

from sklearn.datasets import make\_moons

from keras.models import Sequential

from keras.layers import Dense

X,Y=make\_moons(n\_samples=100,noise=0.2,random\_state=1)

n\_train=30

trainX,testX=X[:n\_train,:],X[n\_train:]

trainY,testY=Y[:n\_train],Y[n\_train:]

#print(trainX)

#print(trainY)

#print(testX)

#print(testY)

model=Sequential()

model.add(Dense(500,input\_dim=2,activation='relu'))

model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['accuracy'])

history=model.fit(trainX,trainY,validation\_data=(testX,testY),epochs=1000)

pyplot.plot(history.history['accuracy'],label='train')

pyplot.plot(history.history['val\_accuracy'],label='test')

pyplot.legend()

pyplot.show()

pyplot.plot(history.history['accuracy'],label='train')

pyplot.plot(history.history['val\_accuracy'],label='test')

pyplot.legend()

pyplot.show()

**Output:**

A graph with numbers and lines

Description automatically generated

**6 b)**

The above code and resultant graph demonstrate overfitting with accuracy of testing data less than accuracy of training data also the accuracy of testing data increases once and then start decreases gradually.

to solve this problem, we can use regularization Hence, we will add two lines in the above code as highlighted below to implement l2 regularization with alpha=0.001

**Code:**

from matplotlib import pyplot

from sklearn.datasets import make\_moons

from keras.models import Sequential

from keras.layers import Dense

from keras.regularizers import l2

X,Y=make\_moons(n\_samples=100,noise=0.2,random\_state=1)

n\_train=30

trainX,testX=X[:n\_train,:],X[n\_train:]

trainY,testY=Y[:n\_train],Y[n\_train:]

#print(trainX)

#print(trainY)

#print(testX)

#print(testY)

model=Sequential()

model.add(Dense(500,input\_dim=2,activation='relu',kernel\_regularizer=l2(0.001)))

model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['accuracy'])

history=model.fit(trainX,trainY,validation\_data=(testX,testY),epochs=1000)

pyplot.plot(history.history['accuracy'],label='train')

pyplot.plot(history.history['val\_accuracy'],label='test')

pyplot.legend()

pyplot.show()

**Output:**

A screen shot of a computer

Description automatically generated

**6 c)**

By applying l1 and l2 regularizer we can observe the following changes in accuracy of both

trainig and testing data. The changes in code are also highlighted.

**Code:**

# !pip install pandas

# !pip install matplotlib

# !pip install keras

# !pip install tensorflow

from matplotlib import pyplot

from sklearn.datasets import make\_moons

from keras.models import Sequential

from keras.layers import Dense

from keras.regularizers import l1\_l2

X,Y=make\_moons(n\_samples=100,noise=0.2,random\_state=1)

n\_train=30

trainX,testX=X[:n\_train,:],X[n\_train:]

trainY,testY=Y[:n\_train],Y[n\_train:]

#print(trainX)

#print(trainY)

#print(testX)

#print(testY)

model=Sequential()

model.add(Dense(500,input\_dim=2,activation='relu',kernel\_regularizer=l1\_l2(l1=0.001,l2=0.001)))

model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['accuracy'])

history=model.fit(trainX,trainY,validation\_data=(testX,testY),epochs=400)

pyplot.plot(history.history['accuracy'],label='train')

pyplot.plot(history.history['val\_accuracy'],label='test')

pyplot.legend()

pyplot.show()

**Output:**

A screenshot of a computer

Description automatically generated

**Learning:**  
This code demonstrates the creation and training of a neural network using the Keras library on a synthetic dataset generated by the **make\_moons** function. The dataset is split into training and testing sets, with the neural network consisting of a single hidden layer with 500 neurons and a ReLU activation function, followed by an output layer with a sigmoid activation function for binary classification. The model is compiled using binary cross-entropy loss and the Adam optimizer. The training process is run for 1000 epochs, and the accuracy for both the training and validation sets is plotted to visualize the model's performance over time.