Assignment 3 (Regularization) by vipin_2011MT22

Implement L2, L1, early stopping and drop out regularization for the previous assignment and write a report comparing the results (Accuracy, Precision, Recall and F1).

```
#importing the libraries
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from keras.constraints import maxnorm
from tensorflow.keras import regularizers
from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score
```

```
#importing the dataset
dataframe = pd.read_csv("diabetes.csv")
dataframe.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
#spilting of dataset into feature and label
df_label = dataframe['Outcome']
df_features = dataframe.drop('Outcome', 1)
df_features.replace('?', -99999, inplace=True)
print(df_label.head())
print(df_features.head())
```

```
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
   Pregnancies  Glucose  BloodPressure  ...   BMI  DiabetesPedigreeFunction  Age
0            6      148             72  ...  33.6                     0.627   50
1            1       85             66  ...  26.6                     0.351   31
2            8      183             64  ...  23.3                     0.672   32
3            1       89             66  ...  28.1                     0.167   21
4            0      137             40  ...  43.1                     2.288   33

[5 rows x 8 columns]
```

```
#hot encoding the label dataset
label = []
for lab in df_label:
    if lab == 1:
        label.append([1, 0])  # class 1
    elif lab == 0:
        label.append([0, 1])  # class 0
```

```
data = np.array(df_features)
label = np.array(label)
print(data.shape,label.shape)
```

```
(768, 8) (768, 2)
```

```
#spilting dataset into testing and training
x_train, x_test, y_train, y_test = train_test_split(data, label, test_size=0.2, random_state=2)
x_train.shape
```

```
(614, 8)
```

For l2

```
#building our Neural Network
model = Sequential()
model.add(Dense(500, input_dim=8, activation='sigmoid',kernel_regularizer=regularizers.l2(0.01), bias_regularizer=regularizer
                activity_regularizer=regularizers.l2(0.01)))
model.add(Dense(100, activation='sigmoid',kernel_regularizer=regularizers.l2(0.01), bias_regularizer=regularizers.l2(0.01),
                activity_regularizer=regularizers.l2(0.01)))
model.add(Dense(2, activation='softmax'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(x_train,y_train, epochs=1000, batch_size=70, validation_data=(x_test, y_test))
```

```
Epoch 1/1000
9/9 [==============================] - 1s 62ms/step - loss: 4.9969 - accuracy: 0.5040 - val_loss: 4.1902 - val_accura
Epoch 2/1000
9/9 [==============================] - 0s 9ms/step - loss: 4.0711 - accuracy: 0.6400 - val_loss: 3.4837 - val_accurac
Epoch 3/1000
9/9 [==============================] - 0s 7ms/step - loss: 3.3932 - accuracy: 0.6151 - val_loss: 2.8637 - val_accurac
Epoch 4/1000
9/9 [==============================] - 0s 7ms/step - loss: 2.7929 - accuracy: 0.6133 - val_loss: 2.3187 - val_accurac
Epoch 5/1000
9/9 [==============================] - 0s 7ms/step - loss: 2.2719 - accuracy: 0.6292 - val_loss: 1.8990 - val_accurac
Epoch 6/1000
9/9 [==============================] - 0s 7ms/step - loss: 1.8678 - accuracy: 0.6500 - val_loss: 1.5995 - val_accurac
Epoch 7/1000
9/9 [==============================] - 0s 9ms/step - loss: 1.5873 - accuracy: 0.6369 - val_loss: 1.4041 - val_accurac
Epoch 8/1000
9/9 [==============================] - 0s 8ms/step - loss: 1.4131 - accuracy: 0.6175 - val_loss: 1.2765 - val_accurac
Epoch 9/1000
9/9 [==============================] - 0s 8ms/step - loss: 1.3075 - accuracy: 0.6078 - val_loss: 1.1833 - val_accurac
Epoch 10/1000
9/9 [==============================] - 0s 10ms/step - loss: 1.2178 - accuracy: 0.6157 - val_loss: 1.1186 - val_accura
Epoch 11/1000
9/9 [==============================] - 0s 9ms/step - loss: 1.1543 - accuracy: 0.6302 - val_loss: 1.0695 - val_accurac
Epoch 12/1000
9/9 [==============================] - 0s 8ms/step - loss: 1.0905 - accuracy: 0.6431 - val_loss: 1.0298 - val_accurac
Epoch 13/1000
9/9 [==============================] - 0s 8ms/step - loss: 1.0566 - accuracy: 0.6345 - val_loss: 1.0007 - val_accurac
Epoch 14/1000
9/9 [==============================] - 0s 8ms/step - loss: 1.0359 - accuracy: 0.6206 - val_loss: 0.9777 - val_accurac
Epoch 15/1000
9/9 [==============================] - 0s 8ms/step - loss: 1.0048 - accuracy: 0.6379 - val_loss: 0.9599 - val_accurac
Epoch 16/1000
9/9 [==============================] - 0s 9ms/step - loss: 0.9877 - accuracy: 0.6364 - val_loss: 0.9434 - val_accurac
Epoch 17/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.9702 - accuracy: 0.6410 - val_loss: 0.9324 - val_accurac
Epoch 18/1000
9/9 [==============================] - 0s 9ms/step - loss: 0.9589 - accuracy: 0.6355 - val_loss: 0.9345 - val_accurac
Epoch 19/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.9515 - accuracy: 0.6431 - val_loss: 0.9143 - val_accurac
Epoch 20/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.9486 - accuracy: 0.6278 - val_loss: 0.9083 - val_accurac
Epoch 21/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.9288 - accuracy: 0.6485 - val_loss: 0.9158 - val_accurac
Epoch 22/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.9348 - accuracy: 0.6380 - val_loss: 0.8960 - val_accurac
Epoch 23/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.9137 - accuracy: 0.6632 - val_loss: 0.9011 - val_accurac
Epoch 24/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.9217 - accuracy: 0.6200 - val_loss: 0.8896 - val_accurac
Epoch 25/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.9012 - accuracy: 0.6578 - val_loss: 0.8955 - val_accurac
Epoch 26/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.9006 - accuracy: 0.6633 - val_loss: 0.8850 - val_accurac
Epoch 27/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.9010 - accuracy: 0.6418 - val_loss: 0.8826 - val_accurac
Epoch 28/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.9141 - accuracy: 0.6244 - val_loss: 0.8802 - val_accurac
Epoch 29/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.9019 - accuracy: 0.6428 - val_loss: 0.8752 - val_accurac
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 500)               4500
_________________________________________________________________
dense_1 (Dense)              (None, 100)               50100
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 202
=================================================================
Total params: 54,802
Trainable params: 54,802
Non-trainable params: 0
_________________________________________________________________
```

```
#predictating whether diabetic or not
Y_pred = model.predict(x_test)
Y_pred =(Y_pred>0.5)
```

```
#Calculating accuracy, f1 score, precision, recall

accuracyL2 = accuracy_score(y_test, Y_pred)
scoreL2 = f1_score(y_test, Y_pred, average='weighted')
precisionL2 = precision_score(y_test, Y_pred, labels=[0,1], average='weighted')
recallL2 = recall_score(y_test, Y_pred, average='weighted')

print('accuracy: ',accuracyL2)
print('F1-score: ',scoreL2)
print('Precision: ', precisionL2)
print("recall: ", recallL2)
```

```
accuracy:  0.7662337662337663
F1-score:  0.7589679907832293
Precision:  0.7563457563457563
recall:  0.7662337662337663
```

For l1

```
#building our Neural Network
model = Sequential()
model.add(Dense(500, input_dim=8, activation='sigmoid',kernel_regularizer=regularizers.l1(0.0001), bias_regularizer=regulari
                activity_regularizer=regularizers.l1(0.0001)))
model.add(Dense(100, activation='sigmoid',kernel_regularizer=regularizers.l1(0.0001), bias_regularizer=regularizers.l1(0.000
                activity_regularizer=regularizers.l1(0.0001)))
model.add(Dense(2, activation='softmax'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(x_train,y_train, epochs=1000, batch_size=70, validation_data=(x_test, y_test))
```

```
Epoch 1/1000
9/9 [==============================] - 1s 29ms/step - loss: 1.0600 - accuracy: 0.5612 - val_loss: 0.9176 - val_accura
Epoch 2/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.9787 - accuracy: 0.6179 - val_loss: 0.8942 - val_accurac
Epoch 3/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.9042 - accuracy: 0.6539 - val_loss: 0.8453 - val_accurac
Epoch 4/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.8635 - accuracy: 0.6711 - val_loss: 0.8345 - val_accurac
Epoch 5/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.8565 - accuracy: 0.6731 - val_loss: 0.8156 - val_accurac
Epoch 6/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.8343 - accuracy: 0.6860 - val_loss: 0.7975 - val_accurac
Epoch 7/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.8215 - accuracy: 0.6943 - val_loss: 0.7809 - val_accurac
Epoch 8/1000
9/9 [==============================] - 0s 9ms/step - loss: 0.8143 - accuracy: 0.7033 - val_loss: 0.7662 - val_accurac
Epoch 9/1000
9/9 [==============================] - 0s 9ms/step - loss: 0.8068 - accuracy: 0.7045 - val_loss: 0.7644 - val_accurac
Epoch 10/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.7747 - accuracy: 0.7137 - val_loss: 0.7473 - val_accurac
Epoch 11/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.7582 - accuracy: 0.7369 - val_loss: 0.7686 - val_accurac
Epoch 12/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.7373 - accuracy: 0.7479 - val_loss: 0.7323 - val_accurac
Epoch 13/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.7367 - accuracy: 0.7280 - val_loss: 0.7340 - val_accurac
Epoch 14/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.7391 - accuracy: 0.7368 - val_loss: 0.7247 - val_accurac
Epoch 15/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.7191 - accuracy: 0.7461 - val_loss: 0.7163 - val_accurac
Epoch 16/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.7182 - accuracy: 0.7630 - val_loss: 0.7379 - val_accurac
Epoch 17/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.7052 - accuracy: 0.7467 - val_loss: 0.7176 - val_accurac
Epoch 18/1000
9/9 [==============================] - 0s 10ms/step - loss: 0.6985 - accuracy: 0.7490 - val_loss: 0.6953 - val_accura
Epoch 19/1000
9/9 [==============================] - 0s 9ms/step - loss: 0.6904 - accuracy: 0.7667 - val_loss: 0.6999 - val_accurac
Epoch 20/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.6912 - accuracy: 0.7518 - val_loss: 0.7041 - val_accurac
Epoch 21/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.6894 - accuracy: 0.7462 - val_loss: 0.6843 - val_accurac
Epoch 22/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.6813 - accuracy: 0.7644 - val_loss: 0.6943 - val_accurac
Epoch 23/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.6714 - accuracy: 0.7593 - val_loss: 0.7148 - val_accurac
Epoch 24/1000
```

```
9/9 [==============================] - 0s 7ms/step - loss: 0.7014 - accuracy: 0.7237 - val_loss: 0.6805 - val_accurac
Epoch 25/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.6752 - accuracy: 0.7502 - val_loss: 0.6648 - val_accurac
Epoch 26/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.6654 - accuracy: 0.7594 - val_loss: 0.7324 - val_accurac
Epoch 27/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.6574 - accuracy: 0.7617 - val_loss: 0.6671 - val_accurac
Epoch 28/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.6464 - accuracy: 0.7605 - val_loss: 0.6639 - val_accurac
Epoch 29/1000
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 500)               4500
_________________________________________________________________
dense_4 (Dense)              (None, 100)               50100
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 202
=================================================================
Total params: 54,802
Trainable params: 54,802
Non-trainable params: 0
_________________________________________________________________
```

```
#predictating whether diabetic or not
Y_pred = model.predict(x_test)
Y_pred =(Y_pred>0.5)
```

```
#Calculating accuracy, f1 score, precision, recall

accuracyL1 = accuracy_score(y_test, Y_pred)
scoreL1 = f1_score(y_test, Y_pred, average='weighted')
precisionL1 = precision_score(y_test, Y_pred, labels=[0,1], average='weighted')
recallL1 = recall_score(y_test, Y_pred, average='weighted')

print('accuracy: ',accuracyL1)
print('F1-score: ',scoreL1)
print('Precision: ', precisionL1)
print("recall: ", recallL1)
```

```
accuracy:  0.7077922077922078
F1-score:  0.708724455268234
Precision:  0.7097162097162096
recall:  0.7077922077922078
```

For Early Stopping

```
#building our Neural Network
model = Sequential()
model.add(Dense(500, input_dim=8, activation='sigmoid'))
model.add(Dense(100, activation='sigmoid'))
model.add(Dense(2, activation='softmax'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
es = EarlyStopping(monitor='val_loss',patience=300)
model.fit(x_train,y_train, epochs=1000, batch_size=70, validation_data=(x_test,y_test),callbacks=[es])
```

```
9/9 [==============================] - 0s 9ms/step - loss: 0.2341 - accuracy: 0.9007 - val_loss: 0.7982 - val_accurac
Epoch 346/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.2364 - accuracy: 0.8915 - val_loss: 0.7903 - val_accurac
Epoch 347/1000
9/9 [==============================] - 0s 9ms/step - loss: 0.2240 - accuracy: 0.8957 - val_loss: 0.8117 - val_accurac
Epoch 348/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.2397 - accuracy: 0.8932 - val_loss: 0.8049 - val_accurac
Epoch 349/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.2334 - accuracy: 0.9035 - val_loss: 0.9362 - val_accurac
Epoch 350/1000
9/9 [==============================] - 0s 9ms/step - loss: 0.2353 - accuracy: 0.9094 - val_loss: 0.8247 - val_accurac
Epoch 351/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.2275 - accuracy: 0.9045 - val_loss: 0.8382 - val_accurac
Epoch 352/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.2331 - accuracy: 0.9018 - val_loss: 0.7882 - val_accurac
Epoch 353/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.2006 - accuracy: 0.9240 - val_loss: 0.7753 - val_accurac
Epoch 354/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.2167 - accuracy: 0.9035 - val_loss: 0.8099 - val_accurac
Epoch 355/1000
```

```
9/9 [==============================] - 0s 7ms/step - loss: 0.2490 - accuracy: 0.8927 - val_loss: 0.8004 - val_accurac
Epoch 356/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.2357 - accuracy: 0.9090 - val_loss: 0.8150 - val_accurac
Epoch 357/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.2309 - accuracy: 0.9043 - val_loss: 0.8188 - val_accurac
Epoch 358/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.1891 - accuracy: 0.9276 - val_loss: 0.8116 - val_accurac
Epoch 359/1000
9/9 [==============================] - 0s 9ms/step - loss: 0.2212 - accuracy: 0.9106 - val_loss: 0.8707 - val_accurac
Epoch 360/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.2220 - accuracy: 0.9107 - val_loss: 0.9720 - val_accurac
Epoch 361/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.2702 - accuracy: 0.8761 - val_loss: 0.9762 - val_accurac
Epoch 362/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.2567 - accuracy: 0.8849 - val_loss: 1.0803 - val_accurac
Epoch 363/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.3259 - accuracy: 0.8672 - val_loss: 0.8595 - val_accurac
Epoch 364/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.2495 - accuracy: 0.8837 - val_loss: 0.8244 - val_accurac
Epoch 365/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.2265 - accuracy: 0.9077 - val_loss: 0.8118 - val_accurac
Epoch 366/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.2268 - accuracy: 0.8958 - val_loss: 0.7841 - val_accurac
Epoch 367/1000
9/9 [==============================] - 0s 9ms/step - loss: 0.1996 - accuracy: 0.9216 - val_loss: 0.8105 - val_accurac
Epoch 368/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.2114 - accuracy: 0.9119 - val_loss: 0.8484 - val_accurac
Epoch 369/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.1854 - accuracy: 0.9367 - val_loss: 0.8963 - val_accurac
Epoch 370/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.2048 - accuracy: 0.9291 - val_loss: 0.8741 - val_accurac
Epoch 371/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.2100 - accuracy: 0.9130 - val_loss: 0.8298 - val_accurac
Epoch 372/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.2459 - accuracy: 0.8966 - val_loss: 0.9344 - val_accurac
Epoch 373/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.2932 - accuracy: 0.8613 - val_loss: 0.9151 - val_accurac
<tensorflow.python.keras.callbacks.History at 0x7f9024080550>
```

```
model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_21 (Dense)             (None, 500)               4500
_________________________________________________________________
dense_22 (Dense)             (None, 100)               50100
_________________________________________________________________
dense_23 (Dense)             (None, 2)                 202
=================================================================
Total params: 54,802
Trainable params: 54,802
Non-trainable params: 0
_________________________________________________________________
```

```
#predictating whether diabetic or not
Y_pred = model.predict(x_test)
Y_pred =(Y_pred>0.5)
```

```
#Calculating accuracy, f1 score, precision, recall

accuracyES = accuracy_score(y_test, Y_pred)
scoreES = f1_score(y_test, Y_pred, average='weighted')
precisionES = precision_score(y_test, Y_pred, labels=[0,1], average='weighted')
recallES = recall_score(y_test, Y_pred, average='weighted')

print('accuracy: ',accuracyES)
print('F1-score: ',scoreES)
print('Precision: ', precisionES)
print("recall: ", recallES)
```

```
accuracy:  0.7402597402597403
F1-score:  0.724173285879275
Precision:  0.7220922272063107
recall:  0.7402597402597403
```

For Dropout

```
#building our Neural Network
```

```
#building our Neural Network
model = Sequential()
model.add(Dense(500, input_dim=8, activation='sigmoid',kernel_constraint=maxnorm(3)))
model.add(Dropout(0.3))

model.add(Dense(100, activation='sigmoid',kernel_constraint=maxnorm(3)))
model.add(Dropout(0.3))

model.add(Dense(2, activation='softmax'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(x_train,y_train, epochs=1000, batch_size=70, validation_data=(x_test, y_test))
```

```
Epoch 1/1000
9/9 [==============================] - 1s 23ms/step - loss: 0.6998 - accuracy: 0.5995 - val_loss: 0.5902 - val_accura
Epoch 2/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.6881 - accuracy: 0.6176 - val_loss: 0.5792 - val_accurac
Epoch 3/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.6518 - accuracy: 0.6068 - val_loss: 0.5746 - val_accurac
Epoch 4/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.6517 - accuracy: 0.6242 - val_loss: 0.5681 - val_accurac
Epoch 5/1000
9/9 [==============================] - 0s 9ms/step - loss: 0.6012 - accuracy: 0.6751 - val_loss: 0.5571 - val_accurac
Epoch 6/1000
9/9 [==============================] - 0s 9ms/step - loss: 0.6231 - accuracy: 0.6482 - val_loss: 0.5566 - val_accurac
Epoch 7/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.6039 - accuracy: 0.6847 - val_loss: 0.5504 - val_accurac
Epoch 8/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.5850 - accuracy: 0.6991 - val_loss: 0.5463 - val_accurac
Epoch 9/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.5824 - accuracy: 0.6975 - val_loss: 0.5572 - val_accurac
Epoch 10/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.5762 - accuracy: 0.7182 - val_loss: 0.5374 - val_accurac
Epoch 11/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.5976 - accuracy: 0.6782 - val_loss: 0.5479 - val_accurac
Epoch 12/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.5646 - accuracy: 0.6917 - val_loss: 0.5371 - val_accurac
Epoch 13/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.5878 - accuracy: 0.6747 - val_loss: 0.5353 - val_accurac
Epoch 14/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.5468 - accuracy: 0.7158 - val_loss: 0.5537 - val_accurac
Epoch 15/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.5689 - accuracy: 0.7125 - val_loss: 0.5325 - val_accurac
Epoch 16/1000
9/9 [==============================] - 0s 10ms/step - loss: 0.5440 - accuracy: 0.7412 - val_loss: 0.5311 - val_accura
Epoch 17/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.5440 - accuracy: 0.7321 - val_loss: 0.5186 - val_accurac
Epoch 18/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.5248 - accuracy: 0.7637 - val_loss: 0.5213 - val_accurac
Epoch 19/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.5450 - accuracy: 0.7438 - val_loss: 0.5175 - val_accurac
Epoch 20/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.5159 - accuracy: 0.7626 - val_loss: 0.5364 - val_accurac
Epoch 21/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.5564 - accuracy: 0.7509 - val_loss: 0.5197 - val_accurac
Epoch 22/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.5570 - accuracy: 0.7269 - val_loss: 0.5166 - val_accurac
Epoch 23/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.5191 - accuracy: 0.7437 - val_loss: 0.5577 - val_accurac
Epoch 24/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.5401 - accuracy: 0.7234 - val_loss: 0.5150 - val_accurac
Epoch 25/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.5066 - accuracy: 0.7769 - val_loss: 0.5169 - val_accurac
Epoch 26/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.5141 - accuracy: 0.7661 - val_loss: 0.5251 - val_accurac
Epoch 27/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.5287 - accuracy: 0.7308 - val_loss: 0.5107 - val_accurac
Epoch 28/1000
9/9 [==============================] - 0s 8ms/step - loss: 0.4971 - accuracy: 0.7694 - val_loss: 0.5280 - val_accurac
Epoch 29/1000
9/9 [==============================] - 0s 7ms/step - loss: 0.5251 - accuracy: 0.7344 - val_loss: 0.5218 - val_accurac
```

```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_9 (Dense)              (None, 500)               4500
_________________________________________________________________
dropout (Dropout)            (None, 500)               0
_________________________________________________________________
dense_10 (Dense)             (None, 100)               50100
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_11 (Dense)             (None, 2)                 202
=================================================================
Total params: 54,802
```

```
Trainable params: 54,802
Non-trainable params: 0
_________________________________________________________________
```

```
#predictating whether diabetic or not
Y_pred = model.predict(x_test)
Y_pred =(Y_pred>0.5)
```

```
#Calculating accuracy, f1 score, precision, recall

accuracyDO = accuracy_score(y_test, Y_pred)
scoreDO = f1_score(y_test, Y_pred, average='weighted')
precisionDO = precision_score(y_test, Y_pred, labels=[0,1], average='weighted')
recallDO = recall_score(y_test, Y_pred, average='weighted')

print('accuracy: ',accuracyDO)
print('F1-score: ',scoreDO)
print('Precision: ', precisionDO)
print("recall: ", recallDO)
```

```
accuracy:  0.7077922077922078
F1-score:  0.7178683385579938
Precision:  0.7409505388228792
recall:  0.7077922077922078
```

Report

```
print('For L2')
print('accuracy: ',accuracyL2)
print('F1-score: ',scoreL2)
print('Precision: ', precisionL2)
print("recall: ", recallL2, '\n')

print('For L1')
print('accuracy: ',accuracyL1)
print('F1-score: ',scoreL1)
print('Precision: ', precisionL1)
print("recall: ", recallL1, '\n')

print('For Early Stopping')
print('accuracy: ',accuracyES)
print('F1-score: ',scoreES)
print('Precision: ', precisionES)
print("recall: ", recallES, '\n')

print('For Dropout')
print('accuracy: ',accuracyDO)
print('F1-score: ',scoreDO)
print('Precision: ', precisionDO)
print("recall: ", recallDO, '\n')
```

```
For L2
accuracy:  0.7662337662337663
F1-score:  0.7589679907832293
Precision:  0.7563457563457563
recall:  0.7662337662337663

For L1
accuracy:  0.7077922077922078
F1-score:  0.708724455268234
Precision:  0.7097162097162096
recall:  0.7077922077922078

For Early Stopping
accuracy:  0.7402597402597403
F1-score:  0.724173285879275
Precision:  0.7220922272063107
recall:  0.7402597402597403

For Dropout
accuracy:  0.7077922077922078
F1-score:  0.7178683385579938
Precision:  0.7409505388228792
recall:  0.7077922077922078
```

L2 regularization model is giving maximum accuracy followed by EarlyStopping. Dropout and L1 regularization model is giving same accuracy.

F1-score of L2 regularization model is maximum followed by Dropout regularization model, EarlyStopping regularization model and L1 regularization model.

Pricision of L2 regularization model is maximum followed by Dropout regularization model, L1 regularization model and EarlyStopping regularization model.

L2 regularization model is giving maximum recall followed by EarlyStopping regularization model, Dropout regularization model and L1 regularization model

for early stopping, the patience parameter plays a huge role, for patience = 300, model stopped at 373 epochs with accuracy of around 74% on test data. This gives a hint that epochs = 1000 is bound to give an overfitted model

comparing between L1 and L2 when for the same models we used a parameter of 0.01 for both the regularisations, L2 had better results and L1 seemed overfitted

overall the metrics are comparable for all models however the L2 model seems to perform slightly better than other models