SONAR DATASET

In [ ]:

```
from pandas import read_csv
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from tensorflow.keras.optimizers import SGD

from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
```

Loading the dataset

In [ ]:

```
dataframe = read_csv("/content/sample_data/sonar data.csv", header=None)
dataset = dataframe.values
```

Splitting dataset into input and output variables

In [ ]:

```
X = dataset[:,0:60].astype(float)
Y = dataset[:,60]
```

Encoding

In [ ]:

```
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
```

Baseline model

In [ ]:

```
def create_baseline():
        # create model
        model = Sequential()
        model.add(Dense(60, input_dim=60, activation='relu'))
        model.add(Dense(30,  activation='relu'))
        model.add(Dense(1, activation='sigmoid'))
        # Compile model
        sgd = SGD(lr=0.01, momentum=0.8)
        model.compile(loss='binary_crossentropy', optimizer=sgd, metrics=['accuracy'])
        return model
```

In [ ]:

```
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasClassifier(build_fn=create_baseline, epochs=300, batch_s
ize=16, verbose=0)))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: Deprecatio
nWarning: KerasClassifier is deprecated, use Sci-Keras (https://github.co
m/adriangb/scikeras) instead. See https://www.adriangb.com/scikeras/stabl
e/migration.html for help migrating.
  This is separate from the ipykernel package so we can avoid doing import
s until
```

In [ ]:

```
pipeline = Pipeline(estimators)
kfold = StratifiedKFold(n_splits=10, shuffle=True)
results = cross_val_score(pipeline, X, encoded_Y, cv=kfold)
print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)

WARNING:tensorflow:5 out of the last 9 calls to <function Model.make_test_
function.<locals>.test_function at 0x7f4fb025ae60> triggered tf.function r
etracing. Tracing is expensive and the excessive number of tracings could
be due to (1) creating @tf.function repeatedly in a loop, (2) passing tens
ors with different shapes, (3) passing Python objects instead of tensors.
For (1), please define your @tf.function outside of the loop. For (2), @t
f.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to http
s://www.tensorflow.org/guide/function#controlling_retracing and https://ww
w.tensorflow.org/api_docs/python/tf/function for  more details.

/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)

WARNING:tensorflow:6 out of the last 11 calls to <function Model.make_test
_function.<locals>.test_function at 0x7f4fb03eb680> triggered tf.function
retracing. Tracing is expensive and the excessive number of tracings could
be due to (1) creating @tf.function repeatedly in a loop, (2) passing tens
ors with different shapes, (3) passing Python objects instead of tensors.
For (1), please define your @tf.function outside of the loop. For (2), @t
f.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to http
s://www.tensorflow.org/guide/function#controlling_retracing and https://ww
w.tensorflow.org/api_docs/python/tf/function for  more details.
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)

Baseline: 83.17% (6.49%)
```

classification accuracy is 83.17%

Using Dropout on the Visible Layer

Dropout in the input layer with weight constraint

In [ ]:

```
from keras.layers import Dropout
from keras.constraints import maxnorm
```

In [ ]:

```
def create_model():
        # create model
        model = Sequential()
        model.add(Dropout(0.2, input_shape=(60,)))
        model.add(Dense(60, activation='relu', kernel_constraint=maxnorm(3)))
        model.add(Dense(30, activation='relu', kernel_constraint=maxnorm(3)))
        model.add(Dense(1, activation='sigmoid'))
        # Compile model
        sgd = SGD(lr=0.1, momentum=0.9)
        model.compile(loss='binary_crossentropy', optimizer=sgd, metrics=['accuracy'])
        return model
```

In [ ]:

```
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasClassifier(build_fn=create_model, epochs=300, batch_size
=16, verbose=0)))
pipeline = Pipeline(estimators)
kfold = StratifiedKFold(n_splits=10, shuffle=True)
results = cross_val_score(pipeline, X, encoded_Y, cv=kfold)
print("Visible: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: Deprecatio
nWarning: KerasClassifier is deprecated, use Sci-Keras (https://github.co
m/adriangb/scikeras) instead. See https://www.adriangb.com/scikeras/stabl
e/migration.html for help migrating.
  This is separate from the ipykernel package so we can avoid doing import
s until
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)


Visible: 88.48% (5.32%)
```

Classification accuracy is 88.48%

Using Dropout on Hidden Layers

Dropout in hidden layers with weight constraint

In [ ]:

```
def create_model():
        # create model
        model = Sequential()
        model.add(Dense(60, input_dim=60, activation='relu', kernel_constraint=maxnorm(
3)))
        model.add(Dropout(0.2))
        model.add(Dense(30, activation='relu', kernel_constraint=maxnorm(3)))
        model.add(Dropout(0.2))
        model.add(Dense(1, activation='sigmoid'))
        # Compile model
        sgd = SGD(lr=0.1, momentum=0.9)
        model.compile(loss='binary_crossentropy', optimizer=sgd, metrics=['accuracy'])
        return model
```

In [ ]:

```
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasClassifier(build_fn=create_model, epochs=300, batch_size
=16, verbose=0)))
pipeline = Pipeline(estimators)
kfold = StratifiedKFold(n_splits=10, shuffle=True)
results = cross_val_score(pipeline, X, encoded_Y, cv=kfold)
print("Hidden: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: Deprecatio
nWarning: KerasClassifier is deprecated, use Sci-Keras (https://github.co
m/adriangb/scikeras) instead. See https://www.adriangb.com/scikeras/stabl
e/migration.html for help migrating.
  This is separate from the ipykernel package so we can avoid doing import
s until
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descen
t.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate
` instead.
  super(SGD, self).__init__(name, **kwargs)

Hidden: 83.55% (7.51%)
```

Classification accuracy is 83.60%

make_circles dtaset

In [ ]:

```
from sklearn.datasets import make_circles
from matplotlib import pyplot
from pandas import DataFrame
```

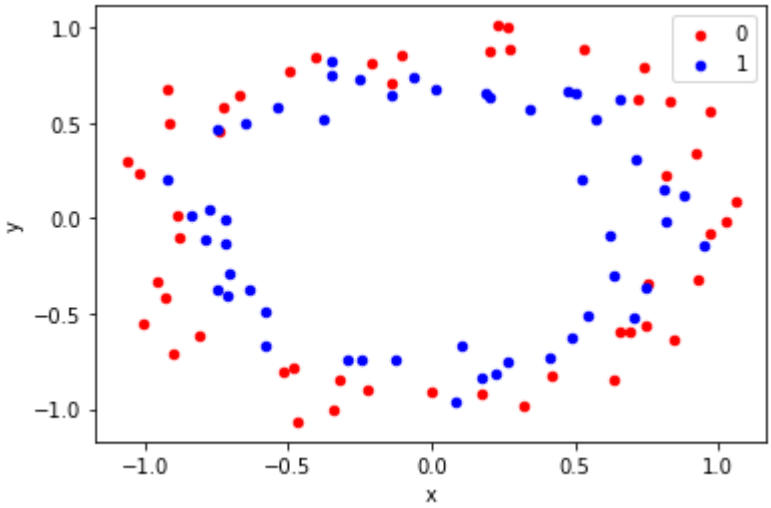
In [ ]:

```
from sklearn import datasets

# generate 2d classification dataset
X, y = make_circles(n_samples=100, noise=0.1, random_state=1)
```

In [ ]:

```
df = DataFrame(dict(x=X[:,0], y=X[:,1], label=y))
colors = {0:'red', 1:'blue'}
fig, ax = pyplot.subplots()
grouped = df.groupby('label')
for key, group in grouped:
    group.plot(ax=ax, kind='scatter', x='x', y='y', label=key, color=colors[key])
pyplot.show()
```



without dropouts

In [ ]:

```
X, y = make_circles(n_samples=100, noise=0.1, random_state=1)
# split into train and test
n_train = 30
trainX, testX = X[:n_train, :], X[n_train:, :]
trainy, testy = y[:n_train], y[n_train:]
```

In [ ]:

```
from keras import models
from keras import layers
```

Defining the Model

In [ ]:

```
model = models.Sequential()
model.add(layers.Dense(500, input_dim=2, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [ ]:

```
#fitting the model

from scipy.integrate import simps
```

In [ ]:

```
history = model.fit(trainX, trainy, validation_data=(testX, testy), epochs=400, verbose
=0)
```

In [ ]:

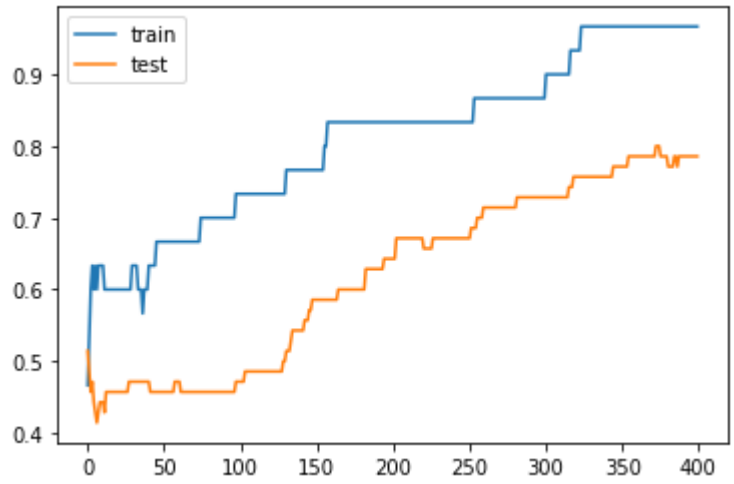
```
train_acc = model.evaluate(trainX, trainy, verbose=0)
test_acc = model.evaluate(testX, testy, verbose=0)
print('Train acc: %.3f' %train_acc[1])
print('Test acc: %.3f' %test_acc[1])
```

```
Train acc: 0.967
Test acc: 0.786
```

Plotting history

In [ ]:

```
pyplot.plot(history.history['accuracy'], label='train')
pyplot.plot(history.history['val_accuracy'], label='test')
pyplot.legend()
pyplot.show()
```



Without dropouts

In [ ]:

```
from keras.layers import Dropout
from keras.models import Sequential
from keras.layers import Dense
```

In [ ]:

```
#defining the model

model = Sequential()
model.add(Dense(500, input_dim=2, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Fitting model

In [ ]:

```
history = model.fit(trainX, trainy, validation_data=(testX, testy), epochs=400, verbose
=0)
```

Evaluating model

In [ ]:

```
train_acc = model.evaluate(trainX, trainy, verbose=0)
test_acc = model.evaluate(testX, testy, verbose=0)
print('Train acc: %.3f' %train_acc[1])
print('Test acc: %.3f' %test_acc[1])
```

```
Train acc: 0.767
Test acc: 0.571
```

In [ ]:

```
pyplot.plot(history.history['accuracy'], label='train')
pyplot.plot(history.history['val_accuracy'], label='test')
pyplot.legend()
pyplot.show()
```