Lab 3

Import Libraries

```
import pandas as pd
import sklearn
```

```
from sklearn.datasets import load_digits
```

Load Digits in-built library

```
digits=load_digits()
```

```
df=pd.DataFrame(digits['data'],columns=digits['feature_names'])
```

```
print(digits.DESCR)
```

```
.. _digits_dataset:

Optical recognition of handwritten digits dataset
--------------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 1797
    :Number of Attributes: 64
    :Attribute Information: 8x8 image of integer pixels in the range 0..16.
    :Missing Attribute Values: None
    :Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)
    :Date: July; 1998

This is a copy of the test set of the UCI ML hand-written digits datasets
https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Di

The data set contains images of hand-written digits: 10 classes where
each class refers to a digit.

Preprocessing programs made available by NIST were used to extract
normalized bitmaps of handwritten digits from a preprinted form. From a
total of 43 people, 30 contributed to the training set and different 13
to the test set. 32x32 bitmaps are divided into nonoverlapping blocks of
4x4 and the number of on pixels are counted in each block. This generates
an input matrix of 8x8 where each element is an integer in the range
0..16. This reduces dimensionality and gives invariance to small
distortions.

For info on NIST preprocessing routines, see M. D. Garris, J. L. Blue, G.
T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C.
```

```
L. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469,
1994.

.. topic:: References

  - C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their
    Applications to Handwritten Digit Recognition, MSc Thesis, Institute of
    Graduate Studies in Science and Engineering, Bogazici University.
  - E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.
  - Ken Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin.
    Linear dimensionalityreduction using relevance weighted LDA. School of
    Electrical and Electronic Engineering Nanyang Technological University.
    2005.
  - Claudio Gentile. A New Approximate Maximal Margin Classification
    Algorithm. NIPS. 2000.
```

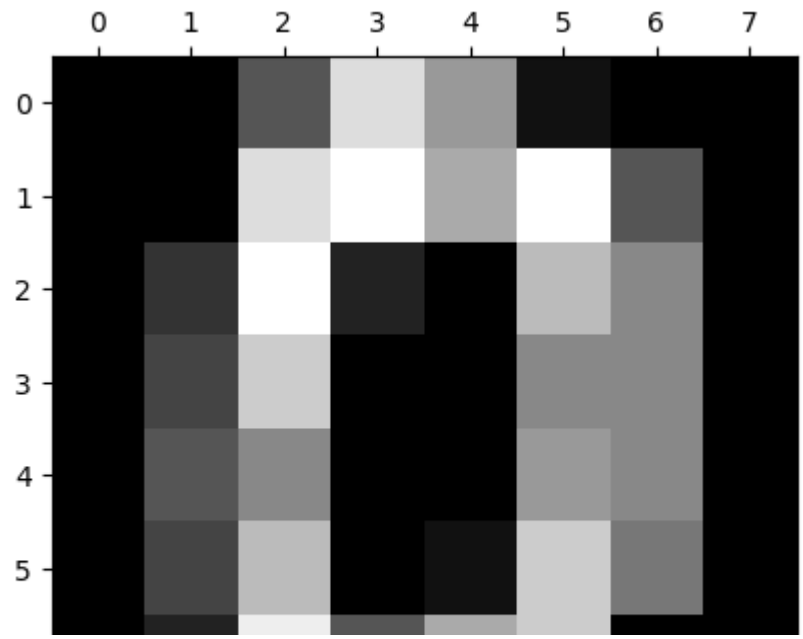
```
digits.data
```

```
array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ..., 10.,  0.,  0.],
       [ 0.,  0.,  0., ..., 16.,  9.,  0.],
       ...,
       [ 0.,  0.,  1., ...,  6.,  0.,  0.],
       [ 0.,  0.,  2., ..., 12.,  0.,  0.],
       [ 0.,  0., 10., ..., 12.,  1.,  0.]])
```

```
import matplotlib.pyplot as plt
plt.gray()
plt.matshow(digits.images[0])

plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



Split Dataset

```
# separating outcome and features
x=digits.data
y=digits.target

# split the data into train and test
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0)
```

Design ANN

```
#!pip install livelossplot
```

```
Requirement already satisfied: livelossplot in /usr/local/lib/python3.10/dist
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: bokeh in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: Jinja2>=2.9 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: contourpy>=1 in /usr/local/lib/python3.10/dist
Requirement already satisfied: numpy>=1.16 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: packaging>=16.8 in /usr/local/lib/python3.10/d
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: pillow>=7.1.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.10/dist
Requirement already satisfied: tornado>=5.1 in /usr/local/lib/python3.10/dist
Requirement already satisfied: xyzservices>=2021.09.1 in /usr/local/lib/pytho
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-pac
```

```
# Import the libraries
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Dense
from livelossplot import PlotLossesKerasTF
from tensorflow.keras.optimizers import SGD
```

Model Building

```
x_train.shape
```

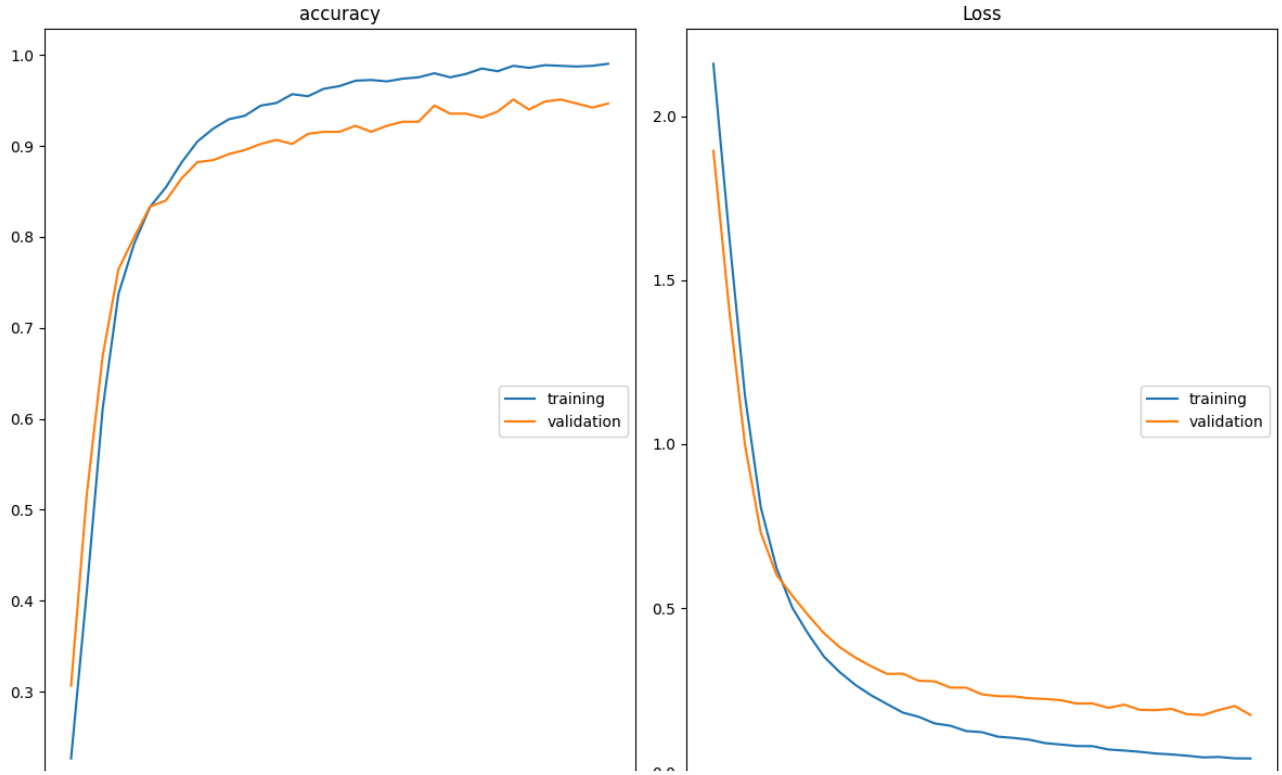
```
(1347, 64)
```

```
model=Sequential()

model.add(Input(shape=(64,),name="Input Layer"))
model.add(Dense(15,activation='relu',name="Hidden1"))
model.add(Dense(20,activation='relu',name="Hidden2"))
model.add(Dense(10,activation='softmax',name='OutputLayer'))

opt=tf.keras.optimizers.Adam()

model.compile(loss='sparse_categorical_crossentropy',optimizer=opt,metrics=['accuracy']
model.fit(x_train,y_train,epochs=35,batch_size=24,validation_data=(x_test,y_test),callb
```



Apply PCA

```
from sklearn.decomposition import PCA


pca=PCA(n_components=12)


x_pca=pca.fit_transform(df)


df.shape
```

```
(1797, 64)
```

```
#Standardise data
from sklearn.preprocessing import StandardScaler
st=StandardScaler()
# st.fit(x_train)
# st.fit(x_test)

x_train_std=st.fit_transform(x_train)
x_test_std=st.fit_transform(x_test)


x_pca.shape
```

```
(1797, 12)
```

```
x_pca
```

```
array([[ -1.2594598 ,  21.27487346,  -9.46302992, ...,   3.6318848 ,
          2.59268234,   1.56898285],
       [  7.95760062, -20.76870067,   4.43947514, ...,  -1.08605024,
         -5.3532776 ,  -2.14957598],
       [  6.99189934,  -9.95584305,   2.95852849, ...,   4.21649804,
         -1.2845012 ,  -0.4479114 ],
       ...,
       [ 10.80128513,  -6.96029852,   5.59956453, ..., -13.05460734,
          8.13381419,   3.82294325],
       [ -4.87210201,  12.4240051 , -10.17089547, ..., -13.15950514,
          3.24712337,   1.17686976],
       [ -0.34439037,   6.36557295,  10.77367739, ..., -12.56876039,
         11.0669261 ,   6.1465601 ]])
```

```
import numpy as np
explained_variance = np.var(x_pca, axis=0)
print(explained_variance)
```

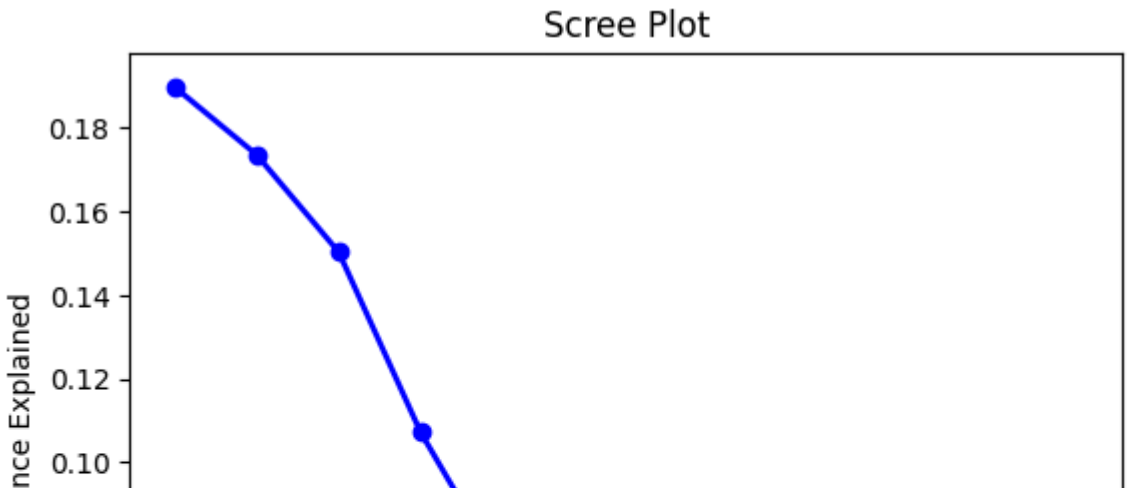
```
[178.90731578 163.62664073 141.70953623 101.04411455  69.47448051
  59.07563165  51.85566463  43.99058385  40.2885398   36.99097948
```

```
        28.50258086  27.30489759]
```

```
explained_variance_ratio = explained_variance / np.sum(explained_variance)
```

```
import matplotlib.pyplot as plt
import numpy as np

PC_values = np.arange(pca.n_components) + 1
plt.plot(PC_values, explained_variance_ratio, 'o-', linewidth=2, color='blue')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()
```




```
target=digits.target
X_train_std, X_test_std, y_train, y_test = train_test_split(x_pca, target, train_size =
print(X_train_std.shape)
print(X_test_std.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1257, 12)
(540, 12)
(1257,)
(540,)
```

```
model=Sequential()

model.add(Input(shape=(12,),name="Input Layer"))

model.add(Dense(15,activation='relu',name="Hidden1"))
```

```
model.add(Dense(20,activation='relu',name="Hidden2"))

model.add(Dense(10,activation='softmax',name='OutputLayer'))

opt = tf.keras.optimizers.Adam()

model.compile(loss='sparse_categorical_crossentropy',optimizer=opt,metrics=['accuracy']

model.fit(X_train_std,y_train,epochs=35,batch_size=24, validation_data=(X_test_std,y_te
```