Practical 1

|  |  |  |
| --- | --- | --- |
| * 1. **Numpy Date: / /2022** | | |
|  | | |
| **Aim:**   1. **Creating blank array, with predefined data, with pattern specific data**   **Code:**  import numpy as np  #initalize 16 elements in a 1-D array  a = np.arange(16)  a  #type of array  a.dtype  **Output:** | | |
| 1. **Slicing and Updating elements**   **Code:**  #slicing  #Basic slicing  c = a[1:4]  c  #Reverse slicing  d= a[::-1]  d  #updating elements  g = g\*10 - 10  g  **Output:** | | |
| 1. **Shape manipulations**   **Code:**  # Shape manipulation  # 1D to 2D  a = np.array([0,5,10,15,20,25,30,35,40,45,50,55])  # A = a.reshape(3,4)  print(a.reshape(3,4))  # 2D to 1D  # A = A.ravel()  print(a.ravel())  **Output:** | | |
| 1. **Looping over arrays**   **Code:**  #Looping  #Print all elements  for x in a:  print(x, end = ' ')  #Loop, print only even elements  for x in a:  if (x%2==0):  print(x, end = ' ')  **Output:** | | |
| 1. **Reading files in numpy**   **Code:**  #file  #after uploading file  with open("numpyread.txt", "r") as f:  a = f.read()  b = np.array(list(a.replace(" ", "")), dtype=int)  print(b)  **Output:** | | |
| 1. **Use numpy vs list for matrix multiplication of 1000 X 1000 array and evaluate computing performance.**   **Code:**  #System Module  import sys  #Declaring 2 lists of 1000 elements  print("Declaring 2 lists of 1000 elements")  list1 = range(1000)  list2 = range(1000,2000)  print(list1)  print(list2)  print("Size of each element of list1 in bytes: ", sys.getsizeof(list1))  print("Size of whole list1 in bytes: ", sys.getsizeof(list1)\*len(list1))  print("Size of each element of list2 in bytes: ", sys.getsizeof(list2))  print("Size of whole list2 in bytes: ", sys.getsizeof(list1)\*len(list2))  #Declaring 2 arrays of 1000 elements  print("\n\nDeclaring 2 arrays of 1000 elements")  arr1 = np.arange(1000)  arr2 = np.arange(1000,2000)  print(arr1)  print(arr2)  print("Size of each element of the Numpy Array1 in bytes: ", arr1.itemsize)  print("Size of the whole Numpy array in bytes: ", arr1.size\*arr1.itemsize)  print("Size of each element of the Numpy Array2 in bytes: ", arr2.itemsize)  print("Size of the whole Numpy Array2 in bytes: ", arr2.size\*arr2.itemsize)  import time  #Capturing time before multiplication of Python Lists  initialTime1 = time.time()  list3 = [(a\*b) for a,b in zip(list1,list2)]  #Calculating execution time  print("Time taken by 2 Lists to perform multiplication: ", (time.time() - initialTime1), "seconds")  #Capturing time before multiplication of Numpy Arrays  initialTime2 = time.time()  arr3 = arr1\*arr2  print("Time taken by 2 Arrays to perform multiplication: ", (time.time() - initialTime2), "seconds")  **Output:** | | |
| * 1. **Pandas** | | |
| **Aim:**   1. **Creating data frame**   **Code:**  import pandas as pd  data = [10,20,30,40,50,60]  df = pd.DataFrame(data, columns=['Numbers'])  df  **Output:** | | |
| 1. **Reading files**   **Code:**  data = pd.read\_csv('/content/iris.data.csv')  data.head()  **Output:** | | |
| 1. **Slicing manipulations**   **Code:**  student = pd.DataFrame({'Name': ['A', 'B', 'C','D', 'E', 'F','G', 'H'],  'Score': [65, 70, 75, 80, 85, 90, 95, 100]})  print(student.loc[0:5, 'Name'])  print(student.iloc[0:5, 0:2])  **Output:** | | |
| 1. **Exporting data files**   **Code:**  # First: create your Data Frames  student = pd.DataFrame({'Name': ['Maaz', 'Krish', 'Riya','Kunal', 'Kartik', 'Rohan','Frenny', 'Sahil'],                          'Score': [96, 69, 70, 88, 79, 64, 62, 57]})  student  # Second: exporting/saving our DataFrame 'student' into CSV file  student\_data\_csv = student.to\_csv('Student\_Score.csv', index=True)  df = pd.read\_csv("Student\_Score.csv")  df  **Output:** | | |
| 1. **Columns and row manipulations with loops**   **Code:**  #IterTuples  for i in country.itertuples():  print(i)  **Output:** | | |
| 1. **Use pandas for masking data and reading if in Boolean format.**   **Code:**  df = pd.DataFrame({"A": [1,None,3,4,5],                     "B": [7,4,1,2,8],                     "C": [9,6,3,2,1],                     "D": [8,7,4,None,3]})  df  **Output:** | | |
| * 1. **Matplotlib** | | |
| **Aim:**   1. **Importing matplotlib**   **Code:**    import matplotlib.pyplot as plt | | |
| 1. **Importing matplotlib**   **Code:**  #simple line chart  xpoints = np.array([0,2,5, 15,30])  ypoints = np.array([1,9,14, 20, 10])  plt.plot(xpoints,ypoints)  plt.show  **Output:** | | |
| 1. **Correlation chart**   **Code:**  #correlation chart  data = np.array([12.0, 24.0, 7., 20.0,  7.0, 22.0, 18.0,22.0,  6.0, 7.0, 20.0, 13.0,  8.0, 5.0, 8,10.0,15.0,25.0])  plt.title("Auto Correlation Plot")  plt.acorr(data, maxlags = 9)  **Output:** | | |
| 1. **Histogram**   **Code:**  #histogram  x = np.random.normal(150,200,300)  plt.hist(x)  **Output:** | | |
| 1. **Plotting of Multivariate data**   **Code:**  #Multivariate data  plt.rcParams['figure.figsize'] = [15, 6.5]  plt.rcParams['figure.autolayout'] = True  def func(x, y):  return 3 \* x + 4 \* y - 2 + np.random.randn(30)  x, y = np.random.randn(2, 30)  y \*= 50  z = func(x, y)  fig, ax = plt.subplots()  s = ax.scatter(x,y, c=z, s=100, marker ='\*', cmap = 'plasma')  fig.colorbar(s)  plt.show()    **Output:** | | |
| 1. **Plot Pi Chart**   **Code:**  #pi chart  y = np.array([20,15,15,60,10, 50])  mylabels = ["Eat", "Fun", "Study", "Sleep", "Play", "Walk"]  plt.pie(y, labels = mylabels)  **Output:** | | |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

Practical 2

|  |
| --- |
| **2. Linear Regression Date: / /2022** |
| **Aim:**  **Select the Dataset of your choice and respond to following questions.**   * **Why do you want to apply regression on selected dataset? Discuss the full story behind the dataset.** * **How many total observations in data?** * **How many independent variables?** * **Which is dependent variable?** * **Which are most useful variable in estimation? Prove using correlation.** * **Implement linear regression using OLS method.** * **Implement linear regression using Gradient Descent from scratch.** * **Implement linear regression using sklearn API.** * **Quantify goodness of your model and discuss steps taken for improvement (RMSE, SSE, R2Score).** * **Discuss comparison of different methods.** |
| **Solution:**  **1) Why do you want to apply regression on selected dataset? Discuss the full story behind the dataset?**  **Answer:** Consider you own an ice cream business and you would like to create a model that could predict the daily revenue in dollars based on the outside air temperature (degC). So to make this kind of prediction where we want have an input parameter aka outside temperate(DegC) and revenue that can be generated as our output it is best choice to use a linear regression model to extrapolate the results and cater them to our needs.  Independent variable X: Outside Air Temperature  Dependant variable Y: Overall daily revenue generated in dollars  **2) How many total observations in data?**  **Answer:** There are total 500 observations in the data set  **Code:**  data = pd.read\_csv("IceCreamData.csv")  data  **Output:**    **3) How many independent variables?**  **Answer:** There is one independent variable as visible which is Independent variable X: Outside Air Temperature  **4) Which are most useful variable in estimation? Prove using correlation.**  **Answer:** The most useful variable from our dataset is Temperature.  **5) Implement linear regression using sklearn API.**  **Answer:** In sklearn library in Python linear regression is implemented using OLS method.  **Code:**  #Split 80% for training and 20% for testing  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size =0.2, random\_state=0)  #Create Linear Regressor and fit data  reg = LinearRegression(fit\_intercept = True)  reg.fit(X\_train.values,y\_train.values)  #Obtaining best-fit Line  print('Linear coefficient is=' , reg.coef\_)  print('Intercept is=' , reg.intercept\_)  #prediction  y\_predict = reg.predict(X\_test)  print(y\_predict)  **Output:**    **Prediction**    **6) Quantify goodness of your model and discuss steps taken for improvement (RMSE, SSE, R2Score)**  **Answer:**  **Code:**  pred\_values = reg.predict(X\_test.values)  mae = metrics.mean\_absolute\_error(y\_test, pred\_values)  rmse = np.sqrt(mse)  r2 = metrics.r2\_score(y\_test, pred\_values)  print('Results')  print("MAE:",mae)  print("MSE:", mse)  print("RMSE:", rmse)  print("R-Squared:", r2)  **Output:** |

|  |  |  |
| --- | --- | --- |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

Practical 3

|  |
| --- |
| **3. Logistic Regression Date: / /2023** |
| **Aim:**  **Select Dataset of your choice and respond to following questions.**   * **Why you want to apply classification on selected dataset? Discuss full story behind dataset.** * **How many total observations in data?** * **How many independent variables?** * **Which is dependent variable?** * **Which are most useful variable in classification? Prove using correlation.** * **Imlement Logistic regression using sklearn** |

|  |  |  |
| --- | --- | --- |
| **Solution:**   1. **Why do you want to apply regression on selected dataset? Discuss the full story behind the Dataset.**   **Answer:** Consider you create a model that could predict that the Person has been placed or not. We have input parameters like ssc\_p, hsc\_p, degree\_p, hsc\_s, specialization for building the model.   1. **How many total observations in data?**   **Answer:** There are total 215 observations in the data set   1. **How many independent variables? Answer:** There are 12 independent variables 2. **Which is dependent variable? Answer:** Status is dependent variable.   **Which is dependent variable? Answer:** Status is dependent variable.  **5) Implement logistic regression using sklearn.**  Text  Description automatically generated  Text  Description automatically generated          **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

**Practical 4**

|  |
| --- |
| **4.KNN Date: / /2023** |
| **Aim:**  **Multi Class Classification (KNN)**  **Select Dataset of your choice and respond to following questions.**   * **Why you want to apply classification on selected dataset? Discuss full story behind dataset.** * **How many total observations in data?** * **How many independent variables?** * **Which is dependent variable?** * **Which is the most useful variable in classification? Prove using correlation.** * **Implement KNN using sklearn api.** * **Implement code to find best value of k by splitting data in train and test.** * **Quantify goodness of your model and discuss steps taken for improvement.** * **Can we use KNN for regression also? Why / Why not?** * **Discuss drawbacks of algorithms such as KNN.** |
| **Code:**   1. **Why you want to apply classification on selected dataset? Discuss full story behind dataset. Answer:** Suppose you own a wine making & exporting business, then depending on the attributes you want to classify the wine in class 0/1/2. Number of instances are 178. Class distribution is - class\_0 (59), class\_1 (71), class\_2 (48).   It has attributes - 13 numeric, predictive attributes and the class. 13 numeric attributes as follows:   * + Alcohol   + Malic acid   + Ash   + Alcalinity of ash   + Magnesium   + Total phenols   + Flavanoids   + Nonflavanoid phenols   + Proanthocyanins   + Color intensity   + Hue   + OD280/OD315 of diluted wines   + Proline  1. **How many total observations in data?**   Answer: There are total 178 observations in the data set   1. **How many independent variables?**   Answer: There are 13 independent variable which are all numeric attributes.   1. **Which is dependent variable?**   Answer: There are 2 dependent variables which are predictive attributes & class. |

|  |
| --- |
| 1. **Which are most useful variable in estimation? Prove using correlation.**   Answer: The most useful variable from our dataset are numeric attributes.   1. **Implement KNN using sklearn api.**   **Code: Knowing our data**  Table  Description automatically generated  Graphical user interface, table  Description automatically generated |

|  |
| --- |
| Graphical user interface  Description automatically generated with low confidence   1. **Implement code to find best value of k by splitting data in train and test Code: Training and testing**   Graphical user interface, text, application, email  Description automatically generated   1. **Quantify goodness of your model and discuss steps taken for improvement.**   **Code: Score and accuracy** |

|  |
| --- |
| Chart, Teams  Description automatically generated   1. **Can we use KNN for regression also? Why / Why not?**   Answer: Yes, KNN can also be used for regression in the same way we do it for classification as KNN works best for numeric values.   1. **Discuss drawbacks of algorithms such as KNN.**   Answer: Disadvantages of KNN   * 1. Does not work well with large dataset: In large datasets, the cost of calculating the distance between the new point and each existing point is huge which degrades the performance of the algorithm.   2. Does not work well with high dimensions: The KNN algorithm doesn’t work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate the distance in each dimension.   3. Need feature scaling: We need to do feature scaling (standardization and normalization) before applying KNN algorithm to any dataset. If we don’t do so, KNN may generate wrong predictions.   4. KNN is sensitive to noise in the dataset. We need to manually impute missing values and remove outliers. |

|  |  |  |
| --- | --- | --- |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

**Practical 5**

|  |  |  |
| --- | --- | --- |
| **Date: / /2023** | | |
| **Aim:** Find a dataset with number of samples smaller than number of features. Apply principle component analysis to select K best features.  Use Support Vector Machines/Naïve Bayes to train predictive model.  Compare model accuracy and time required for training with full dataset and with selected K features. (use Sci-kit-learn library) | | |
| Solution:  "Wine Quality" dataset from the UCI Machine Learning Repository is being used. This dataset contains 1599 instances and 11 attributes describing various properties of different wines. The goal is to predict the quality of the wine on a scale of 0 to 10.  from sklearn.datasets import load\_wine  from sklearn.decomposition import PCA  from sklearn.model\_selection import train\_test\_split  from sklearn.svm import SVC  from sklearn.naive\_bayes import GaussianNB  from sklearn.metrics import accuracy\_score  import time  data = load\_wine()  X = data.data  y = data.target    K = 5  pca = PCA(n\_components=K)  X\_pca = pca.fit\_transform(X)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  X\_train\_pca, X\_test\_pca, y\_train\_pca, y\_test\_pca = train\_test\_split(X\_pca, y, test\_size=0.2, random\_state=42)  svm = SVC()  start\_time = time.time()  svm.fit(X\_train, y\_train)  train\_time = time.time() - start\_time  y\_pred = svm.predict(X\_test)  acc = accuracy\_score(y\_test, y\_pred)  print("Accuracy with full dataset:", acc)  print("Training time with full dataset:", train\_time)  Accuracy with full dataset: 0.8055555555555556  Training time with full dataset: 0.016330480575561523  svm\_pca = SVC()  start\_time = time.time()  svm\_pca.fit(X\_train\_pca, y\_train\_pca)  train\_time\_pca = time.time() - start\_time  y\_pred\_pca = svm\_pca.predict(X\_test\_pca)  acc\_pca = accuracy\_score(y\_test\_pca, y\_pred\_pca)  print("Accuracy with PCA-selected features:", acc\_pca)  print("Training time with PCA-selected features:", train\_time\_pca)  Accuracy with PCA-selected features: 0.7777777777777778  Training time with PCA-selected features: 0.002498626708984375  nb = GaussianNB()  start\_time = time.time()  nb.fit(X\_train, y\_train)  train\_time = time.time() - start\_time  y\_pred = nb.predict(X\_test)  acc = accuracy\_score(y\_test, y\_pred)  print("Accuracy with full dataset:", acc)  print("Training time with full dataset:", train\_time)  Accuracy with full dataset: 1.0  Training time with full dataset: 0.012969017028808594  nb\_pca = GaussianNB()  start\_time = time.time()  nb\_pca.fit(X\_train\_pca, y\_train\_pca)  train\_time\_pca = time.time() - start\_time  y\_pred\_pca = nb\_pca.predict(X\_test\_pca)  acc\_pca = accuracy\_score(y\_test\_pca, y\_pred\_pca)  print("Accuracy with PCA-selected features:", acc\_pca)  print("Training time with PCA-selected features:", train\_time\_pca)  Accuracy with PCA-selected features: 1.0  Training time with PCA-selected features: 0.0018205642700195312 | | |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

**Practical 6**

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **6. Decision Tree Date: / /2023** | | **Aim:** Write a program to demonstrate the working of the decision tree based ID3  algorithm. Use an appropriate data set for building the decision tree and apply  this knowledge to classify a new sample. |   **Solution:**  **Importing Necessary Libraries.**    **To Read Data from CSV File.**    **Creating a node class which can be used to create a tree-like structure where each node represents an attribute or decision point.**    **Splitting the data recursively based on different features to construct the decision tree.**    **Selecting the best feature to split the data at each node of the decision tree, feature with lowest entropy.**    **Selecting the best feature to split the data at each node of the decision tree, feature with highest gain ratio.**    **Creating a function which recursively creates a decision tree by dividing the data into sub tables based on the highest gain ratio.**    **Defining two functions that can be used to print a decision tree in a readable format.**    **This code reads in a dataset using the read\_data function and creates a decision tree using the create\_node function. It then prints the decision tree using the print\_tree function.** |

|  |  |  |
| --- | --- | --- |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

Practical 7

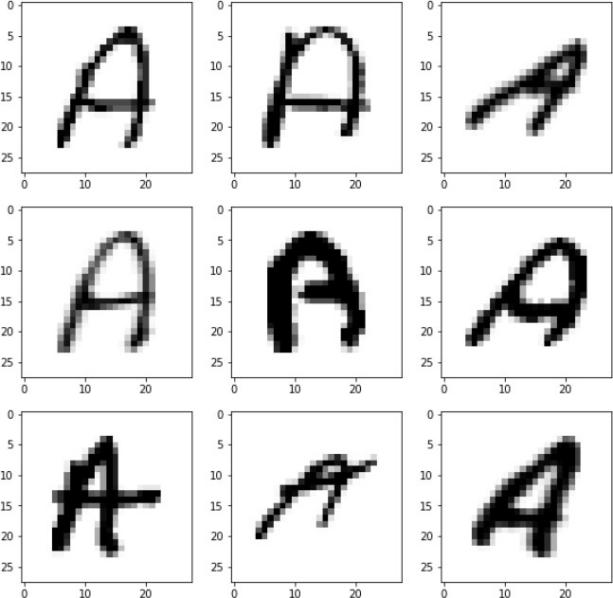
|  |
| --- |
| **7. Principle Component Analysis Date: / /2023** |
| **Aim: Practical Implementation of Principle Component Analysis(PCA).** |

|  |  |  |  |
| --- | --- | --- | --- |
| **We are using twitter US airlines for text classification using RNN and LSTM according to sentiments of text**  **Solution:** | | | |
| **Conclusion/Summary:** | | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

# Practical 8

|  |  |
| --- | --- |
| **8. CNN Date: / /2023** | |
| **Aim: Implement a Convolutional Neural Network (CNN) using Keras library.**   1. **Implement a Convolutional Neural Network (CNN) for a handwritten Character Recognition. Use MNIST dataset to train the model. Generate test images by yourself.** 2. **Case Study to build a CNN model using python.** 3. **Build a dataset on home appliances (available at your home/ can take help from internet). Also use data augmentation technique to increase dataset.** 4. **Preprocess the image to fit into the model** 5. **Apply the CNN model and train over the preprocess data.** 6. **Evaluate the model using confusion matrix.** | |
|  | **First step will be to import Required libraries:**  import numpy as np import pandas as pd  import matplotlib.pyplot as plt  **Read Dataset:**  data = pd.read\_csv('A\_Z Handwritten Data.csv').astype('float32') data.head(10) | |
|  |  | |
|  | **Splitting of Data:**  X = data.drop('0',axis = 1) y = data['0'] | |

**Performing Split Using Sklearn:**



from sklearn.model\_selection import train\_test\_split from sklearn.utils import shuffle

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2)

# Reshaping the Training and Testing data:

x\_train = np.reshape(x\_train.values, (x\_train.shape[0], 28,28)) x\_test = np.reshape(x\_test.values, (x\_test.shape[0], 28,28)) print("Shape of Training data: ", x\_train.shape)

print("Shape of Testing data: ", x\_test.shape)

# Shuffle the training data:

shuffle\_data = shuffle(x\_train)

# Visualize our training data:

import cv2

fig, axes = plt.subplots(3,3, figsize = (10,10)) axes = axes.flatten()

for i in range(9):

\_, shu = cv2.threshold(shuffle\_data[i], 30, 200, cv2.THRESH\_BINARY) axes[i].imshow(np.reshape(shuffle\_data[i], (28,28)), cmap="Greys") plt.show()



# Again Reshaping Data:

training of our model.

x\_train = x\_train.reshape(x\_train.shape[0],x\_train.shape[1],x\_train.shape[2],1) x\_test = x\_test.reshape(x\_test.shape[0], x\_test.shape[1], x\_test.shape[2],1) print("New shape of training data: ", x\_train.shape)

print("New shape of testing data: ", x\_test.shape)

# Model Creation:

import tensorflow

from tensorflow.keras.utils import to\_categorical

y\_training = to\_categorical(y\_train, num\_classes = 26, dtype='int') y\_testing = to\_categorical(y\_test, num\_classes = 26, dtype='int') from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout from tensorflow.keras.optimizers import SGD, Adam

from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping

model = Sequential()

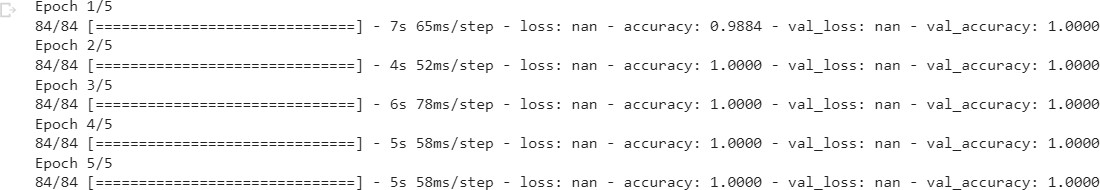
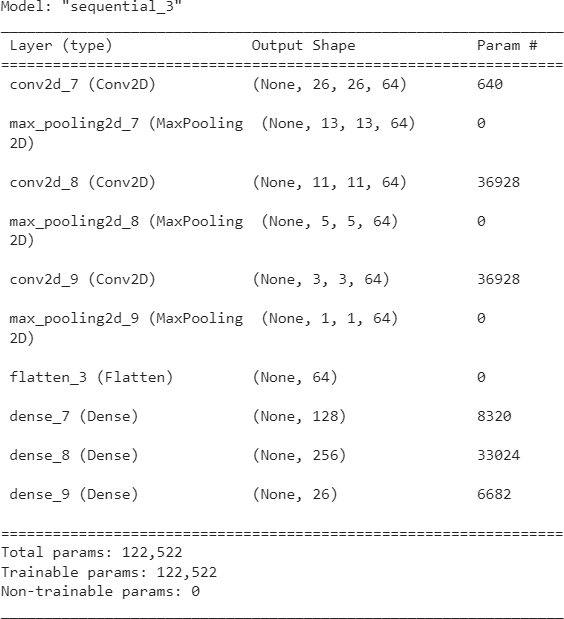
model.add(Conv2D(64 , (3, 3), activation='relu', input\_shape=(28,28,1)))

model.add(MaxPool2D(2, 2))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPool2D(2, 2))

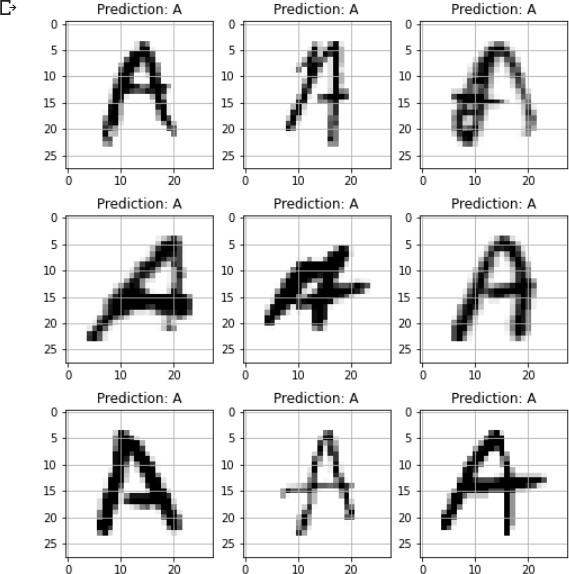
model.add(Conv2D(64, (3, 3), activation='relu')) model.add(MaxPool2D(2,2)) model.add(Flatten()) model.add(Dense(128,activation ="relu")) model.add(Dense(256,activation ="relu")) model.add(Dense(26,activation ="softmax")) model.summary()



# Compile and Fit our model:

model.compile(optimizer = Adam(learning\_rate=0.001), loss='categorical\_crossentropy', metrics=['accuracy'])

history = model.fit(x\_train, y\_training, epochs=5, validation\_data = (x\_test,y\_testing))



# Prediction Process:

fig, axes = plt.subplots(3,3, figsize=(8,9)) axes = axes.flatten()

for i,ax in enumerate(axes):

image = np.reshape(x\_test[i], (28,28)) ax.imshow(image, cmap="Greys") pred = words[np.argmax(y\_testing[i])] ax.set\_title("Prediction: "+pred) ax.grid()

|  |  |  |
| --- | --- | --- |
| **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

Practical 9

|  |
| --- |
| **9. RNN for Text Classification Date: / /2023** |
| **Aim: Implement a RNN/LSTM to classify Text into categories according to the**  **sentiment of the text.** |

|  |  |  |
| --- | --- | --- |
| **We are using twitter US airlines for text classification using RNN and LSTM according to sentiments of text**  **Solution:**  **Importing necessary libraries like numpy, pandas, keras, matplotlib, sklearn and nltk**  import numpy as np  import pandas as pd  import re  from keras.models import Model  from keras.layers import Dense, Input, Dropout, LSTM, Activation  from keras.layers.embeddings import Embedding  from keras.preprocessing import sequence  from keras.initializers import glorot\_uniform  from keras.utils import np\_utils  from keras.callbacks import EarlyStopping  from nltk.corpus import stopwords  np.random.seed(1)  from sklearn.model\_selection import train\_test\_split  #from emo\_utils import \*  import matplotlib.pyplot as plt  %matplotlib inline  **Function to remove English stopwords from a pandas series**  def remove\_stopwords(input\_text):  '''  Function to remove English stopwords from a Pandas Series.    Parameters:  input\_text : text to clean  Output:  cleaned Pandas Series  '''  stopwords\_list = stopwords.words('english')  # Some words which might indicate a certain sentiment are kept via a whitelist  whitelist = ["n't", "not", "no"]  words = input\_text.split()  clean\_words = [word for word in words if (word not in stopwords\_list or word in whitelist) and len(word) > 1]  return " ".join(clean\_words)  def remove\_mentions(input\_text):  '''  Function to remove mentions, preceded by @, in a Pandas Series    Parameters:  input\_text : text to clean  Output:  cleaned Pandas Series  '''  return re.sub(r'@\w+', '', input\_text)  **Read Dataset**  train\_df = pd.read\_csv("../input/twitter-airline-sentiment/Tweets.csv")  train\_df.head()      **Check labels and plott their sentiment graph**  Mood = train\_df['airline\_sentiment'].value\_counts()  index = [1,2,3]  plt.bar(index,Mood,color=['r','b','g'])  plt.xticks(index,['Negative','Neutral','Positive'])  plt.xlabel('Mood')  plt.ylabel('Mood Count')  plt.title('Mood Distribution')    **Cleaning Data**  train\_df = train\_df[['text', 'airline\_sentiment']]  train\_df.text = train\_df.text.apply(remove\_mentions)  train\_df.loc[:,'sentiment'] = train\_df.airline\_sentiment.map({'negative':0,'neutral':1,'positive':2})  train\_df = train\_df.drop(['airline\_sentiment'], axis=1)  train\_df.head()    **Split Dataset**  X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(raw\_docs\_train, sentiment\_train,  stratify=sentiment\_train,  random\_state=42,  test\_size=0.1, shuffle=True)  print('# Train data samples:', X\_train.shape)  print('# Test data samples:', X\_test.shape)  assert X\_train.shape[0] == Y\_train.shape[0]  assert X\_test.shape[0] == Y\_test.shape[0]    **Converting to hot encoding vector for softmax for neural network**  num\_labels = len(np.unique(sentiment\_train))  Y\_oh\_train = np\_utils.to\_categorical(Y\_train, num\_labels)  Y\_oh\_test = np\_utils.to\_categorical(Y\_test, num\_labels)  print(Y\_oh\_train.shape)    **Create Keras Embedding Layer**  def pretrained\_embedding\_layer(word\_to\_vec\_map, word\_to\_index):  vocab\_len = len(word\_to\_index) + 1  emb\_dim = word\_to\_vec\_map["cucumber"].shape[0] word vectors (= 50)    emb\_matrix = np.zeros((vocab\_len,emb\_dim))    for word, index in word\_to\_index.items():  emb\_matrix[index, :] = word\_to\_vec\_map[word]    embedding\_layer = Embedding(vocab\_len, emb\_dim, trainable = False)  embedding\_layer.build((None,))    embedding\_layer.set\_weights([emb\_matrix])    return embedding\_layer  **LSTM Model**  def ltsm\_model(input\_shape, word\_to\_vec\_map, word\_to\_index):    sentence\_indices = Input(shape=input\_shape, dtype='int32')  embedding\_layer = pretrained\_embedding\_layer(word\_to\_vec\_map, word\_to\_index)  embeddings = embedding\_layer(sentence\_indices)    X = LSTM(128, return\_sequences=True)(embeddings)  X = Dropout(0.5)(X)  X = LSTM(128, return\_sequences=False)(X)  X = Dropout(0.5)(X)  X = Dense(3, activation=None)(X)  X = Activation('softmax')(X)    model = Model(inputs=[sentence\_indices], outputs=X)  return model  model = ltsm\_model((maxLen,), word\_to\_vec\_map, word\_to\_index)  model.summary()    **Optimising Parameters using epochs and earlystopping**  earlystop = EarlyStopping(monitor='val\_loss', min\_delta=0, patience=3, verbose=0, mode='auto')  model.fit(X\_train\_indices, y=Y\_oh\_train, batch\_size=512, epochs=20,  verbose=1, validation\_data=(X\_test\_indices, Y\_oh\_test), callbacks=[earlystop])          **Conclusion/Summary:** | | |
| **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** |

**Practical 10**

|  |
| --- |
| **10. K-means Clustering Date: / /2023** |
| **Aim: Use K-Means Clustering and Hierarchical Clustering algorithm for following datasets.** |
| **Solution:**      **KMeans Clustering**      **Plotting**      **Hierarchical Clustering**    **Plotting**     |  |  |  | | --- | --- | --- | | **Conclusion/Summary:** | | | | **Student Signature & Date** | **Marks:** | **Evaluator Signature & Date** | |