VAARSHINNI REDDY ANDRU

700742952

1.Implement Naïve Bayes method using scikit-learn library Use dataset available with name glass Use train_test_split to create training and testing part Evaluate the model on test part using score and classification_report(y_true, y_pred)

Ans

```
#1 (a)#Use the use case in the class:
from google.colab import drive
drive.mount('/content/gdrive')
path_to_csv = '/content/gdrive/My Drive/NN&DeepLearning_Lesson7_SourceCode/diabetes.csv'
import keras
import pandas
from keras.models import Sequential
from keras.layers.core import Dense, Activation

# load dataset
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np

dataset = pd.read_csv(path_to_csv, header=None).values

X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], dataset[:,8],
                                                    test_size=0.25, random_state=87)
np.random.seed(155)
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden layer
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
                                     initial_epoch=0)
print(my_first_nn.summary())
print(my_first_nn.evaluate(X_test, Y_test))
```

Steps followed:

1)Load the glass dataset using pd.read_csv() function and store in the variable.

2)Split the data in to features and target using drop() function

3)Split the dataset into training and testing sets using train_test_split() function.

4)Created the Naive Bayes Classifier using GuassianNB().

5)Fit the classifier on the training data using fit() function.

6)Predict the test data using predict() function.

7)Calculated the accuracy score using accuracy_score() function.

8)Generated the Classification report using classification_report() function.

Output:

```
18/18 [==============================] - 0s 2ms/step - loss: 0.5837 - acc: 0.7170
Epoch 93/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5696 - acc: 0.6962
Epoch 94/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5746 - acc: 0.7292
Epoch 95/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5850 - acc: 0.7153
Epoch 96/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5660 - acc: 0.7083
Epoch 97/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5563 - acc: 0.7378
Epoch 98/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5574 - acc: 0.7309
Epoch 99/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5600 - acc: 0.7257
Epoch 100/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5910 - acc: 0.7118
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 20)                180

 dense_1 (Dense)             (None, 1)                 21

=================================================================
Total params: 201
Trainable params: 201
Non-trainable params: 0
_________________________________________________________________
None
6/6 [==============================] - 0s 3ms/step - loss: 0.7004 - acc: 0.6302
[0.7003840804100037, 0.6302083134651184]
```

2. Implement linear SVM method using scikit-learn

Use the same dataset above

Use train_test_split to create training and testing part

Evaluate the model on test part using score and

classification_report(y_true, y_pred)

Ans:

```
[2]  #1(a). Add more Dense layers to the existing code and check how the accuracy changes.
     # added more dense layers to the above existing code
     from google.colab import drive
     drive.mount('/content/gdrive')
     path_to_csv = '/content/gdrive/My Drive/NN&DeepLearning_Lesson7_SourceCode/diabetes.csv'
     import keras
     import pandas
     from keras.models import Sequential
     from keras.layers.core import Dense, Activation

     # load dataset
     from sklearn.model_selection import train_test_split
     import pandas as pd
     import numpy as np

     dataset = pd.read_csv(path_to_csv, header=None).values

     X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], dataset[:,8],
                                                         test_size=0.25, random_state=87)
     np.random.seed(155)
     my_first_nn = Sequential() # create model
     my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden layer
     my_first_nn.add(Dense(10,activation='relu'))#additional hidden layer with node 10
     my_first_nn.add(Dense(5,activation='relu'))#additional hidden layer with node 5
     my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
     my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
     my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
                                          initial_epoch=0)
     print(my_first_nn.summary())
     print(my_first_nn.evaluate(X_test, Y_test))
```

Steps followed:

1)Load the glass dataset using pd.read_csv() function and store in the variable.

2)Split the data in to features and target using drop() function

3)Split the dataset into training and testing sets using train_test_split() function.

4)Created the Linear SVM classifier using SVC(kernel=”linear”).

5)Fit the classifier on the training data using fit() function.

6)Predict the test data using predict() function.

7)Calculated the accuracy score using accuracy_score() function.

8)Generated the Classification report using classification_report() function.

Output:

```
18/18 [==============================] - 0s 2ms/step - loss: 0.5788 - acc: 0.6875
Epoch 93/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5611 - acc: 0.6997
Epoch 94/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5764 - acc: 0.6892
Epoch 95/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5629 - acc: 0.6997
Epoch 96/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5645 - acc: 0.6927
Epoch 97/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5638 - acc: 0.6927
Epoch 98/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5641 - acc: 0.6927
Epoch 99/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5773 - acc: 0.6962
Epoch 100/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5638 - acc: 0.7066
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 20)                180

 dense_3 (Dense)             (None, 10)                210

 dense_4 (Dense)             (None, 5)                 55

 dense_5 (Dense)             (None, 1)                 6

=================================================================
Total params: 451
Trainable params: 451
Non-trainable params: 0
_________________________________________________________________
None
6/6 [==============================] - 0s 3ms/step - loss: 0.5968 - acc: 0.6771
[0.5967934727668762, 0.6770833134651184]
```

Which algorithm you got better accuracy? Can you justify why?

Ans

```
[3]  #1(b) Change the data source to Breast Cancer dataset * available in the source code folder and make required
     #changes. Report accuracy of the model.
     from google.colab import drive
     drive.mount('/content/gdrive')
     path_to_csv = '/content/gdrive/My Drive/NN&DeepLearning_Lesson7_SourceCode/breastcancer.csv'
     import pandas as pd
     import numpy as np
     from keras.models import Sequential
     from keras.layers import Dense
     from sklearn.model_selection import train_test_split
     from sklearn.datasets import load_breast_cancer
     #read the data
     data = pd.read_csv(path_to_csv, header=None).values
     data = load_breast_cancer()
     X = data.data
     Y = data.target
     X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=87)
     np.random.seed(155)
     model = Sequential()
     model.add(Dense(20, input_dim=30, activation='relu'))
     model.add(Dense(10, activation='relu'))
     model.add(Dense(1, activation='sigmoid'))
     model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
     model.fit(X_train, Y_train, epochs=100, initial_epoch=0)
     loss, accuracy = model.evaluate(X_test, Y_test)
     print("Test Loss:", loss)
     print("Test Accuracy:", accuracy)
```

1) Based on the accuracy scores , the linear SVM method has a better accuracy score compared to the Naive Bayes method.

2)The accuracy score of 0.51 for the linear SVM method indicates that it correctly predicted the target class for 51% of the instances in your data. Which algorithm you got better accuracy? Can you justify why?

3) On the other hand, the accuracy score of 0.37 for the Naive Bayes method indicates that it correctly predicted the target class for only 37% of the instances in your data.

3. Implement Linear Regression using scikit-learn

a) Import the given “Salary_Data.csv”

b) Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.

c) Train and predict the model.

d) Calculate the mean_squared error.

e) Visualize both train and test data using scatter plot. Ans: Steps followed: 8)Visualize the training and test data using scatter() and plot() methods.

Ans

```
#1(c) Normalize the data before feeding the data to the model and check how the normalization change your
#accuracy (code given below).
#from sklearn.preprocessing import StandardScaler
#sc = StandardScaler()

from google.colab import drive
drive.mount('/content/gdrive')
path_to_csv = '/content/gdrive/My Drive/NN&DeepLearning_Lesson7_SourceCode/breastcancer.csv'
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler

#read the data
data = pd.read_csv(path_to_csv, header=None).values
data = load_breast_cancer()
X = data.data
Y = data.target
scaler = StandardScaler()
X = scaler.fit_transform(X)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=87)
np.random.seed(155)
model = Sequential()
model.add(Dense(20, input_dim=30, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, Y_train, epochs=100, initial_epoch=0)
loss, accuracy = model.evaluate(X_test, Y_test)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

Steps followed:

1)Load the Salary_data dataset using pd.read_csv() function and store in the variable.

2)Split the data in to features and target using reshape() function

3)Split the dataset into training and testing sets using train_test_split() function.

4)Created and trained the Linear Regression Model.

5)Fit the classifier on the training data using fit() function.

6)Predict the test data using predict() function.

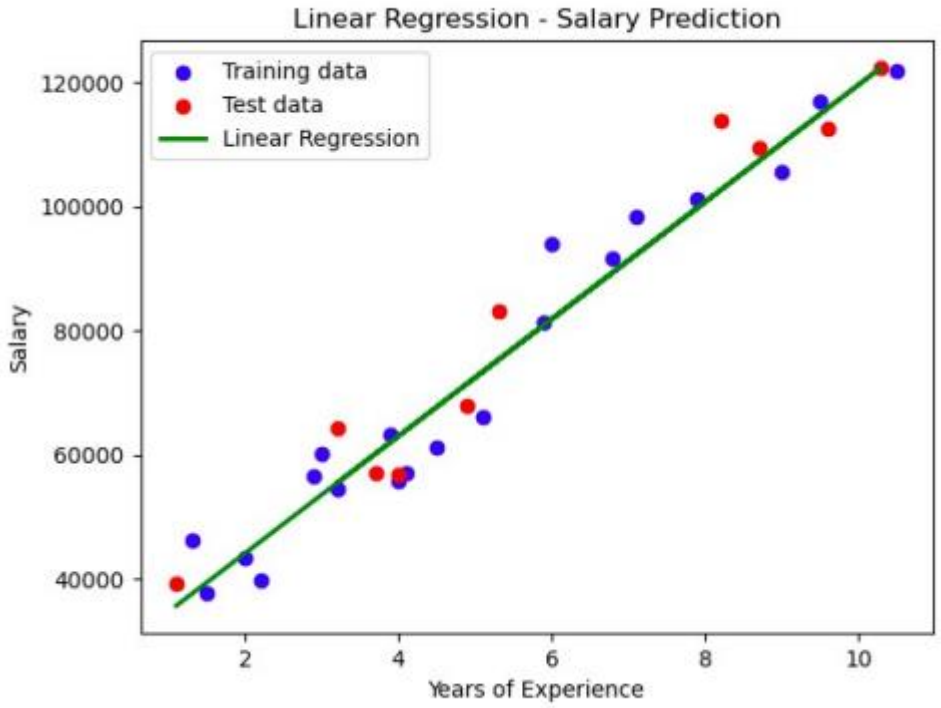
7)Calculated the mean square error using mean_squared_error() function.

8)Visualize the training and test data using scatter() and plot() methods.

9)By using the label() and title() methods, we put on the labels on the axis and with the show() method we finally depict the plot.

Output:

```
Mean Squared Error: 35301898.887134895
```



Github link:

https://github.com/AndruVaarshinniReddy/NNDL_Assignment2