Group No 71

Group Member Names:

1. Saurabh Arunrao Dhande (2021fc04700)

1. Problem Statement

Students are expected to identify a classification / regression problem of your choice. You have to detail the problem under this heading which basically addresses the following questions.

1. What is the problem that you are trying to solve?
2. What kind of prediction (classification / regression) task are you performing?

ENSURE THAT YOU ARE USING NUMERICAL / CATEGORICAL DATA only.

DO NOT use images or textual data.

Score: 1 Mark in total (0.5 mark each)

1.What is the problem that you are trying to solve? -> Breast Cancer classification problem. From given data set user has to classify which type of cancer has been ocurred in a patient ie. (M = malignant, B = benign)

2.What kind of prediction (classification / regression) task are you performing? -> Classifcation Problem

2. Data Acquisition

For the problem identified by you, students have to find the data source themselves from any data source.

2.1 Download the data directly

In [136…

```
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
#from scikeras.wrappers import KerasClassifier
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
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn.metrics import confusion_matrix
```

```
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
# keeps the plots in one place. calls image as static pngs
%matplotlib inline
import matplotlib.pyplot as plt # side-stepping mpl backend
import matplotlib.gridspec as gridspec # subplots
from sklearn.model_selection import train_test_split
#import mpld3 as mpl
#
#Import models from scikit learn module:
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
#from sklearn.cross_validation import KFold   #For K-fold cross validation
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import metrics
```

In [137…

```
# load dataset
df = pd.read_csv("data_breast.csv",header = 0)
df.head()
```

Out[137]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	comp
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 32 columns

2.2 Code for converting the above downloaded data into a form suitable for DL

2.3 Write your observations from the above.

1. Size of the dataset
2. What type of data attributes are there?

Score: 2 Mark

In [138…

```
df.shape
```

Out[138]:

```
(569, 32)
```

In [139…

```
df.describe()
```

Out[139]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compa
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	

8 rows × 31 columns

In [140…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

3. Data Preparation

Perform the data prepracessing that is required for the data that you have downloaded.

3.1 Apply techiniques

- to remove duplicate data
- to impute or remove missing data
- to remove data inconsistencies

IF ANY

In [141...
```
df.drop('id',axis=1,inplace=True)
# size of the dataframe
len(df)
```

Out[141]: 569

In [142...
```
df.isnull().values.any()
```

Out[142]: False

In [143...
```
df.duplicated().values.any()
```

Out[143]: False

3.2 Encode categorical data

In [144...
```
df.diagnosis.unique()
```

Out[144]: array(['M', 'B'], dtype=object)

In [145...
```
df['diagnosis'] = df['diagnosis'].map({'M':1,'B':0})
df.head()
```

Out[145]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_me
0	1	17.99	10.38	122.80	1001.0	0.11840	0.277
1	1	20.57	17.77	132.90	1326.0	0.08474	0.078
2	1	19.69	21.25	130.00	1203.0	0.10960	0.159
3	1	11.42	20.38	77.58	386.1	0.14250	0.283
4	1	20.29	14.34	135.10	1297.0	0.10030	0.132

5 rows × 31 columns

3.3 Normalize the data

In [146...

```
df_min_max_norm = df.copy()

# apply normalization techniques
for column in df_min_max_norm.columns:
    df_min_max_norm[column] = df_min_max_norm[column]  / df_min_max_norm[column].abs().max

# view normalized data
display(df_min_max_norm)
```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_r
0	1.0	0.639986	0.264257	0.651459	0.400240	0.724602	0.80
1	1.0	0.731768	0.452393	0.705040	0.530188	0.518605	0.22
2	1.0	0.700462	0.540988	0.689655	0.481008	0.670747	0.46
3	1.0	0.406261	0.518839	0.411565	0.154378	0.872093	0.82
4	1.0	0.721807	0.365071	0.716711	0.518593	0.613831	0.38
...	...	...	...	...	...	...	
564	1.0	0.766987	0.570010	0.753316	0.591363	0.679315	0.33
565	1.0	0.716115	0.719196	0.696021	0.504198	0.598531	0.29
566	1.0	0.590537	0.714868	0.574536	0.343103	0.517442	0.29
567	1.0	0.732835	0.746690	0.743236	0.505798	0.720930	0.80
568	0.0	0.276058	0.624745	0.254218	0.072371	0.322093	0.12

569 rows × 31 columns

3.4 Feature Engineering

if any

In [1]:

```
##---------Type the code below this line------------------##
```

3.5 Identify the target variables.

- Separate the data front the target such that the dataset is in the form of (X,y) or (Features, Label)
- Discretize / Encode the target variable or perform one-hot encoding on the target or any other as and if required.

In [148...

```
df_target = df['diagnosis']
df_target.head
```

Out[148]:
```
<bound method NDFrame.head of 0      1
1      1
2      1
3      1
4      1
      ..
564    1
565    1
566    1
567    1
568    0
Name: diagnosis, Length: 569, dtype: int64>
```

In [149…
```
df_target.value_counts()
```

Out[149]:
```
0    357
1    212
Name: diagnosis, dtype: int64
```

3.6 Split the data into training set and testing set

In [150…
```
# split the dataset
X_train, X_test, y_train, y_test = train_test_split(
    df_min_max_norm, df_target, test_size=0.3, random_state=0)
```

In [151…
```
print("X_tran : ",X_train.shape)
print("y_tran : ",y_train.shape)
print("X_test : ",X_test.shape)
print("y_test : ",y_test.shape)
```

```
X_tran :  (398, 31)
y_tran :  (398,)
X_test :  (171, 31)
y_test :  (171,)
```

3.7 Report

Mention the method adopted and justify why the method was used

- to remove duplicate data, if present
- to impute or remove missing data, if present
- to remove data inconsistencies, if present
- to encode categorical data
- the normalization technique used

If the any of the above are not present, then also add in the report below.

Report the size of the training dataset and testing dataset

Score: 3 Marks

---------Type the answer below this line------------------

- to remove duplicate data, if present -> Duplcate data is not present

- to impute or remove missing data, if present -> No missing values
- to remove data inconsistencies, if present -> No inconsistencies present in the data
- to encode categorical data -> Categorical data is available and it has been encoded refer below code df['diagnosis'] = df['diagnosis'].map({'M':1,'B':0}) df.head()
- the normalization technique used -> We have used min max normaliation technique for normalization
- Report the size of the training dataset and testing dataset X_tran : (398, 31) y_tran : (398,) X_test : (171, 31) y_test : (171,)

4. Deep Neural Network Architecture

4.1 Design the architecture that you will be using to solve the prediction problem identified.

- Add dense layers, specifying the number of units in each layer and the activation function used in the layer.

In [152…

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

model = Sequential()
model.add(Dense(128, activation='relu', input_dim=31))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_16"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
dense_31 (Dense)            (None, 128)               4096

_________________________________________________________________
dense_32 (Dense)            (None, 1)                 129
=================================================================
Total params: 4,225
Trainable params: 4,225
Non-trainable params: 0
_________________________________________________________________
```

4.2 Report

Report the following and provide justification for the same.

- Number of layers
- Number of units in each layer
- Activation function used in each hidden layer
- Activation function used in the output layer
- Total number of trainable parameters

Score: 4 Marks

---------Type the answer below this line------------------

- Number of layers -> 2
- Number of units in each layer -> 1 st layer has 128 units 2nd layer has 1 unit
- Activation function used in each hidden layer -> Rectified Linear Activation Function or ReLU
- Activation function used in the output layer -> As this is a binary classifcation model so we have used "Sigmoid" as aactivation function
- Total number of trainable parameters -> 31

5. Training the model

5.1 Configure the training and Train the model

Configure the model for training, by using appropriate optimizers and regularizations

In [153…

```
hist = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size
```

```
Epoch 1/10
4/4 [==============================] - 0s 79ms/step - loss: 0.6633 - accuracy: 0.3869 - v
al_loss: 0.6344 - val_accuracy: 0.7076
Epoch 2/10
4/4 [==============================] - 0s 10ms/step - loss: 0.6129 - accuracy: 0.9146 - v
al_loss: 0.5904 - val_accuracy: 1.0000
Epoch 3/10
4/4 [==============================] - 0s 9ms/step - loss: 0.5731 - accuracy: 1.0000 - va
l_loss: 0.5551 - val_accuracy: 0.9942
Epoch 4/10
4/4 [==============================] - 0s 15ms/step - loss: 0.5384 - accuracy: 1.0000 - v
al_loss: 0.5252 - val_accuracy: 0.9942
Epoch 5/10
4/4 [==============================] - 0s 16ms/step - loss: 0.5093 - accuracy: 1.0000 - v
al_loss: 0.4970 - val_accuracy: 0.9942
Epoch 6/10
4/4 [==============================] - 0s 10ms/step - loss: 0.4806 - accuracy: 1.0000 - v
al_loss: 0.4690 - val_accuracy: 0.9883
Epoch 7/10
4/4 [==============================] - 0s 12ms/step - loss: 0.4529 - accuracy: 1.0000 - v
al_loss: 0.4426 - val_accuracy: 0.9883
Epoch 8/10
4/4 [==============================] - 0s 12ms/step - loss: 0.4273 - accuracy: 1.0000 - v
al_loss: 0.4177 - val_accuracy: 0.9942
Epoch 9/10
4/4 [==============================] - 0s 14ms/step - loss: 0.4031 - accuracy: 1.0000 - v
al_loss: 0.3944 - val_accuracy: 0.9942
Epoch 10/10
4/4 [==============================] - 0s 14ms/step - loss: 0.3801 - accuracy: 0.9975 - v
al_loss: 0.3722 - val_accuracy: 0.9942
```

Justify your choice of optimizers and regulizations used and the hyperparameters tuned

Score: 4 Marks

---------Type the answers below this line------------------

This is binary classifictaion probelem soto get better accuracy we have used Sigmoid activation function in output layer and relu activation fuction in the middle layer. As dataset size is low and contain around 600 rows so we have used 2 layer NN model.Adaptive Moment Estimation is an algorithm for optimization technique for gradient descent. The method is really efficient when working with large problem involving a lot of parameters.Here we have used binary_crossentropy loss fuction as this is a binary classification problem.

6. Test the model

Score: 2 Marks

In [154…

```
from sklearn.metrics import confusion_matrix

y_pred = model.predict(X_test) > 0.5
```

7. Conclusion

Plot the training and validation loss Report the testing accuracy and loss.

Report values for preformance study metrics like accuracy, precision, recall, F1 Score.

A proper comparision based on different metrics should be done and not just accuracy alone, only then the comparision becomes authentic. You may use Confusion matrix, classification report, MAE etc per the requirement of your application/problem.

Score 2 Marks

In [158…

```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()

acc = hist.history['accuracy']
val = hist.history['val_accuracy']
epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, '-', label='Training accuracy')
plt.plot(epochs, val, ':', label='Validation accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.plot()
```

Out[158]:

```
[]
```

In [156…

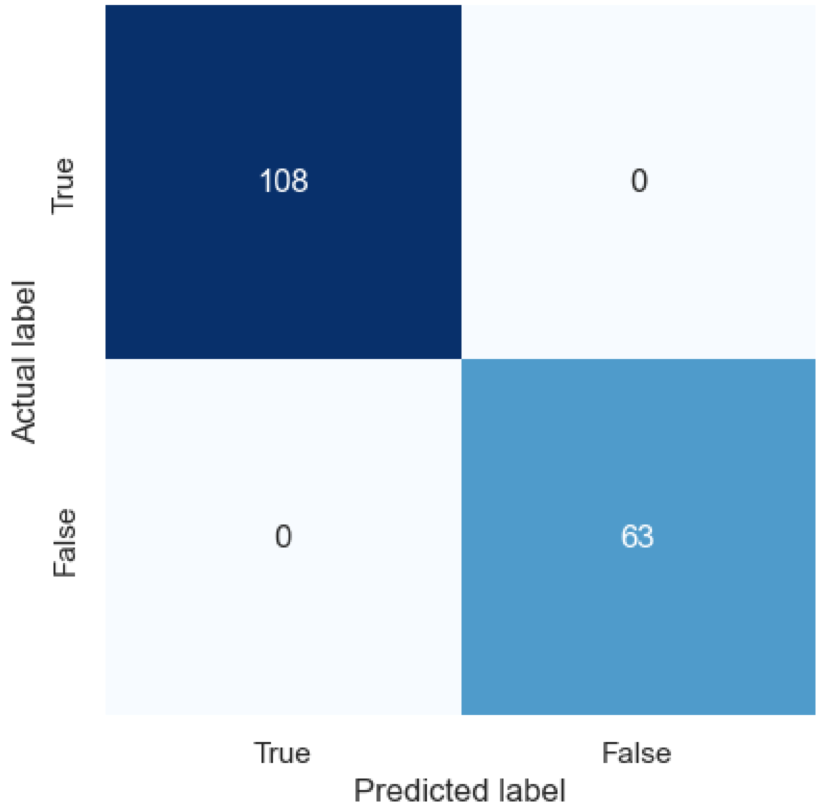
```
mat = confusion_matrix(y_test, y_predicted)
labels = ['True', 'False']

sns.heatmap(mat, square=True, annot=True, fmt='d', cbar=False, cmap='Blues',
            xticklabels=labels, yticklabels=labels)

plt.xlabel('Predicted label')
plt.ylabel('Actual label')
```

Out[156]:

```
Text(110.44999999999997, 0.5, 'Actual label')
```



In [157…

```
print('Precision: %.3f' % precision_score(y_test, y_pred))
print('Recall: %.3f' % recall_score(y_test, y_pred))
print('Accuracy: %.3f' % accuracy_score(y_test, y_pred))
print('F1 Score: %.3f' % f1_score(y_test, y_pred))
```

```
Precision: 1.000
Recall: 0.984
Accuracy: 0.994
F1 Score: 0.992
```

8. Solution

What is the solution that is proposed to solve the business problem discussed in Section 1. Also share your learnings while working through solving the problem in terms of challenges, observations, decisions made etc.

Score 2 Marks

---------Type the answers below this line------------------

1.One of the common challnege I have faced is that installation of libariries in to the system so that required code should run. 2.I was not able to download data form internet so I have downloaded file and used in the file. 3.I have god hands on eperience on Deep learning problem. 4.The problem which I have selected is a binary classification problem and as this is the binary classifiction problem then I have decided to use "sigmoid" activation function in the output layer.

NOTE

All Late Submissions will incur a penalty of -2 marks. So submit your assignments on time.

Good Luck