Diabetes prediction

Using the optimal neural network technique (feed forward network), this challenge tries to predict whether patients will get diabetes in the following five years based on their medical information.

Data Cleaning

After importing the essential libraries and analysing the train data and some null values were observed as shown (Figure 01).

```
df.isnull().sum()
```

```
Id         0
A1       102
A2         5
A3        32
A4       196
A5       322
A6         9
A7         0
A8         0
Class      0
dtype: int64
```

Figure 1 (Null Values)

The null values were replaced with the median values to clean up the missing data.

```
df['A1'].fillna(df['A1'].median(), inplace =True)

df['A2'].fillna(df['A2'].median(), inplace =True)

df['A3'].fillna(df['A3'].median(), inplace =True)

df['A6'].fillna(df['A6'].median(), inplace =True)
```

```
df['A5'].fillna(df['A5'].median(), inplace =True)
```

```
df['A4'].fillna(df['A4'].median(), inplace =True)
```

Figure 2 Data Cleaning.

```
[ ]  df.isnull().sum()
```

```
Id       0
A1       0
A2       0
A3       0
A4       0
A5       0
A6       0
A7       0
A8       0
Class    0
dtype: int64
```

Figure 3 After Cleaning the dataset.

The training data is divided into X and y, with 70% of the data being used to train, 30% being used to validate, and with 100 random states.

```
[ ]  # extract labels
     y = df['Class']
     y = pd.Series(y)
     print(y.value_counts())

     # remove unnecessary columns
     X = df.drop(['Id', 'Class'], axis=1)
     print(X.info())
     X.head()
```

Figure 4 Splitting the input and output column.

Building the model

```
# split data to train and validation sets
X_train1, X_val1, y_train1, y_val1 = train_test_split(X, y, test_size=0.3, random_state=100)
print(f'training data set size: {len(X_train1)}')
print(f'validation data set size: {len(X_val1)}')
```

```
training data set size: 467
validation data set size: 201
```

Figure 5 Splitting the Data for Training and Validation.

A sequential model is used to develop a neural network (deep learning model). The "Keras" library was used to import this.

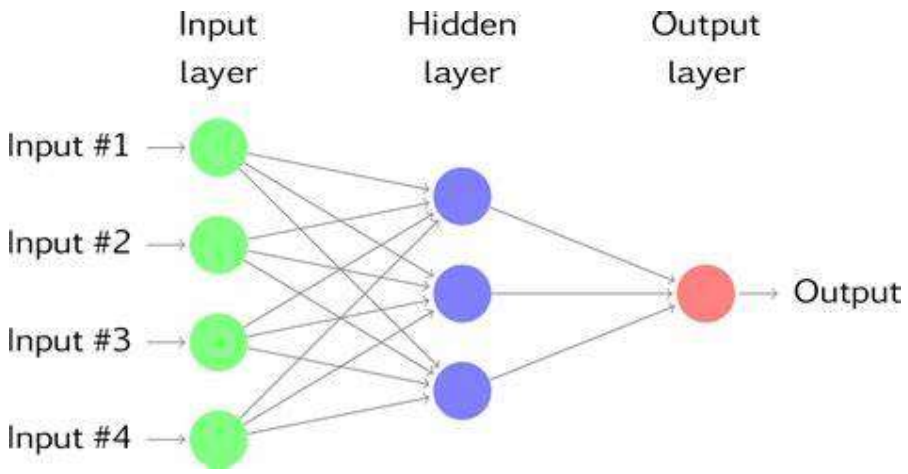

Figure 6 Example of Neural network

```
[ ]  # define the keras model
     model = Sequential()
     model.add(Dense(units=45, input_dim=8, activation='relu'))
     model.add(Dense(units=29, activation='relu'))
     model.add(Dense(units=1, activation='sigmoid'))
     model.summary()
```

Figure 7 Defining the model

The dense layer is used to construct the neural network layer. The dense layer receives input from all neurons in the previous layer.

Input layer - In a sequential model, the first layer must receive information about its input shape. In this case “input_dim=8” because this dataset as 8 input value and “relu” activation has been used.

Output layer - In a sequential model, the last layer is the output layer, and the output layer in this dataset is 1, hence "units = 1" is used. Because the Dataset output is just binary, activation “sigmoid" was utilised.

Model must configure the learning process, which is done with the compile () method. It is received with three arguments: 1. loss 2. Optimizer 3. Metrics

Adam optimizer is used with 0.01 learning rate.

```
# compile the keras model
import tensorflow as tf
# create an optimiser object and set the learning rate
opt = tf.keras.optimizers.Adam(learning_rate=0.01)
```

```
# Compile the model with the optimiser object
model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])
```

Figure 8 compiling the model

We got a model accuracy of 78.06 percent after compiling and training it.

```
# train model
history = model.fit(X_train1, y_train1, batch_size=10, epochs=75, validation_data=(X_val1, y_val1))
```

Figure 9 Training the model

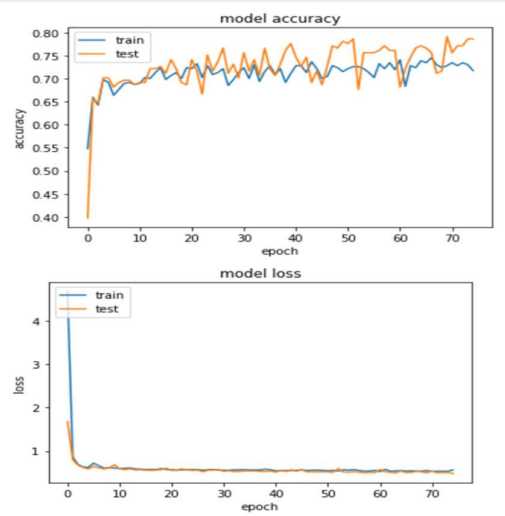

Figure 10 Model accuracy and lose

```
# get model predictions
y_pred1 = model.predict(X_val1)
print(y_pred1[:10])
```

```
[[0.30935037]
 [0.2947747 ]
 [0.2680152 ]
 [0.2508877 ]
 [0.58719355]
 [0.04067072]
 [0.6974376 ]
 [0.20083085]
 [0.65029895]
 [0.16025504]]
```

```
# convert to categorical predictions
y_pred_categorical1 = [1 if pred > 0.5 else 0 for pred in y_pred1]
print(y_pred_categorical1[:10])
```

```
[0, 0, 0, 0, 1, 0, 1, 0, 1, 0]
```

Figure 11 Prediction values.

Testing the model.

After training the model, it was put to the test with a test dataset.

```
df_test = pd.read_csv('/content/test.csv')
df_test
```

	Id	A1	A2	A3	A4	A5	A6	A7	A8
0	1	6	98	58	33	190	34.0	0.430	43
1	2	9	154	78	30	100	30.9	0.164	45
2	3	6	165	68	26	168	33.6	0.631	49
3	4	1	99	58	10	0	25.4	0.551	21
4	5	10	68	106	23	49	35.5	0.285	47
...	...	...	...	...	...	...	...	...	...
95	96	10	101	76	48	180	32.9	0.171	63
96	97	2	122	70	27	0	36.8	0.340	27
97	98	5	121	72	23	112	26.2	0.245	30
98	99	1	126	60	0	0	30.1	0.349	47
99	100	1	93	70	31	0	30.4	0.315	23

100 rows × 9 columns

Figure 12 Test dataset

```
predictions = model.predict(X_test)
print(predictions)

final_predictions = [1 if pred > 0.5 else 0 for pred in predictions]
print(final_predictions)
```

Figure 13 Model prediction

```
# Summarise class details
sns.countplot(x=Test_pred['Class'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f62f09b0210>
```

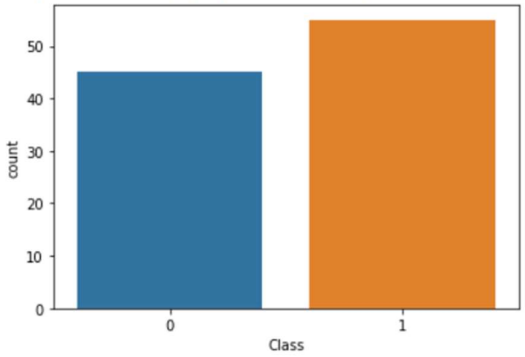

Figure 14 Final prediction

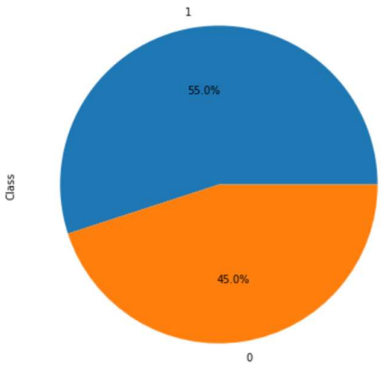

Figure 15 Final prediction in percentage

Conclusion:

I learned and developed the feed forward neural network by analysing medical records to predict whether patients will get diabetes in the next five years.

Colab link:
https://colab.research.google.com/drive/1no7hFLWGpH8qPG95QOAdLqlu2KrYSC7x?usp=sharing