Name: Santiago, John Loyd C.
Course and Section: CPE 019 - CPE32S3
Date of Submission: March 26, 2024
Instructor: Engr. Roman Richard

link of the google colab: https://colab.research.google.com/drive/1WbZ54jv92WMjx0o4SqCdaQQTZexmnLul?usp=sharing

link of the dataset: https://www.kaggle.com/datasets/mlomuscio/beliefs-about-masks-among-young-adults

In this assignment, you are task to build a multilayer perceptron model. The following are the requirements:

- Choose any dataset
- Explain the problem you are trying to solve
- Create your own model
- Evaluate the accuracy of your model

Note: Submit a PDF, the dataset and the notebook you used for this assignment.

Choose any dataset

The dataset that I choose is about the habits and beliefs that teenagers have regarding face masks.

Explain the problem you are trying to solve

The problem I'm trying to solve is determining whether the beliefs and habits of teenagers affect whether they use face masks in public.

Preprocessing

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Activation
import matplotlib.pyplot as plt
from csv import reader
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from __future__ import print_function
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.optimizers import RMSprop
import matplotlib.pyplot as plt
%matplotlib inline

WP = pd.read_csv("./MaskBeliefs.csv")


WP.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 107 entries, 0 to 106
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Timestamp         107 non-null    object
 1   Boarding          107 non-null    object
 2   Age               107 non-null    int64
 3   Gender            106 non-null    object
 4   ResidentialElder  107 non-null    object
 5   InteractedElder   107 non-null    object
 6   Restaurant        105 non-null    float64
 7   PreventSpread     107 non-null    object
 8   Reason            107 non-null    object
 9   Public            107 non-null    object
dtypes: float64(1), int64(1), object(8)
memory usage: 8.5+ KB
```

```
WP.head(1000)
```

	Timestamp	Boarding	Age	Gender	ResidentialElder	InteractedElder	Restaurant	PreventSpread	Reason	Public
0	9/25/2020 15:04:43	Day	16	Female	No	Yes	1.0	Yes	To protect yourself AND others	Yes
1	9/25/2020 15:04:46	Boarding	17	Male	No	No	2.0	Yes	To protect yourself AND others	Yes
2	9/25/2020 15:04:58	Boarding	17	Male	No	Yes	0.0	Yes	To protect yourself AND others	Yes
3	9/25/2020 15:05:12	Day	17	Female	No	Yes	2.0	Yes	To protect yourself AND others	Yes
4	9/25/2020 15:05:12	Day	17	Female	No	Yes	2.0	Yes	To protect yourself AND others	Yes
...	...	...	...	...	...	...	...	...	...	...
102	9/28/2020 10:56:47	Boarding	17	Female	No	No	7.0	Yes	To protect yourself AND others	Yes
103	9/28/2020 12:08:13	Boarding	15	Male	No	No	1.0	Yes	To protect yourself AND others	Yes
104	9/28/2020 13:12:01	Boarding	15	Male	Yes	Yes	0.0	Yes	To protect yourself AND others	No
105	9/28/2020 23:27:53	Boarding	16	Male	No	No	0.0	Yes	To protect yourself AND others	Yes
106	9/29/2020 9:56:52	Day	16	Female	No	Yes	2.0	Yes	To protect yourself AND others	No

107 rows × 10 columns

Next steps: View recommended plots

```
WP= WP.fillna(0)

WP.head(1000)
```

	Timestamp	Boarding	Age	Gender	ResidentialElder	InteractedElder	Restaurant	PreventSpread	Reason	Public
0	9/25/2020 15:04:43	Day	16	Female	No	Yes	1.0	Yes	To protect yourself AND others	Yes
1	9/25/2020 15:04:46	Boarding	17	Male	No	No	2.0	Yes	To protect yourself AND others	Yes
2	9/25/2020 15:04:58	Boarding	17	Male	No	Yes	0.0	Yes	To protect yourself AND others	Yes
3	9/25/2020 15:05:12	Day	17	Female	No	Yes	2.0	Yes	To protect yourself AND others	Yes
4	9/25/2020 15:05:12	Day	17	Female	No	Yes	2.0	Yes	To protect yourself AND others	Yes
...	...	...	...	...	...	...	...	...	...	...
102	9/28/2020 10:56:47	Boarding	17	Female	No	No	7.0	Yes	To protect yourself AND others	Yes
103	9/28/2020 12:08:13	Boarding	15	Male	No	No	1.0	Yes	To protect yourself AND others	Yes
104	9/28/2020 13:12:01	Boarding	15	Male	Yes	Yes	0.0	Yes	To protect yourself AND others	No
105	9/28/2020 23:27:53	Boarding	16	Male	No	No	0.0	Yes	To protect yourself AND others	Yes
106	9/29/2020 9:56:52	Day	16	Female	No	Yes	2.0	Yes	To protect yourself AND others	No

107 rows × 10 columns

Next steps: View recommended plots

```
WP["PreventSpread"] = WP["PreventSpread"].apply(lambda toLabel: 1 if toLabel == 'Yes' else 0)
WP["Public"] = WP["Public"].apply(lambda toLabel: 1 if toLabel == 'Yes' else 0)
WP["Boarding"] = WP["Boarding"].apply(lambda toLabel: 1 if toLabel == 'Day' else 0)
WP["Gender"] = WP["Gender"].apply(lambda toLabel: 0 if toLabel == 'Male' else 1)
WP["ResidentialElder"] = WP["ResidentialElder"].apply(lambda toLabel: 1 if toLabel == 'Yes' else 0)
WP["InteractedElder"] = WP["InteractedElder"].apply(lambda toLabel: 1 if toLabel == 'Yes' else 0)


WP.head(1000)
```

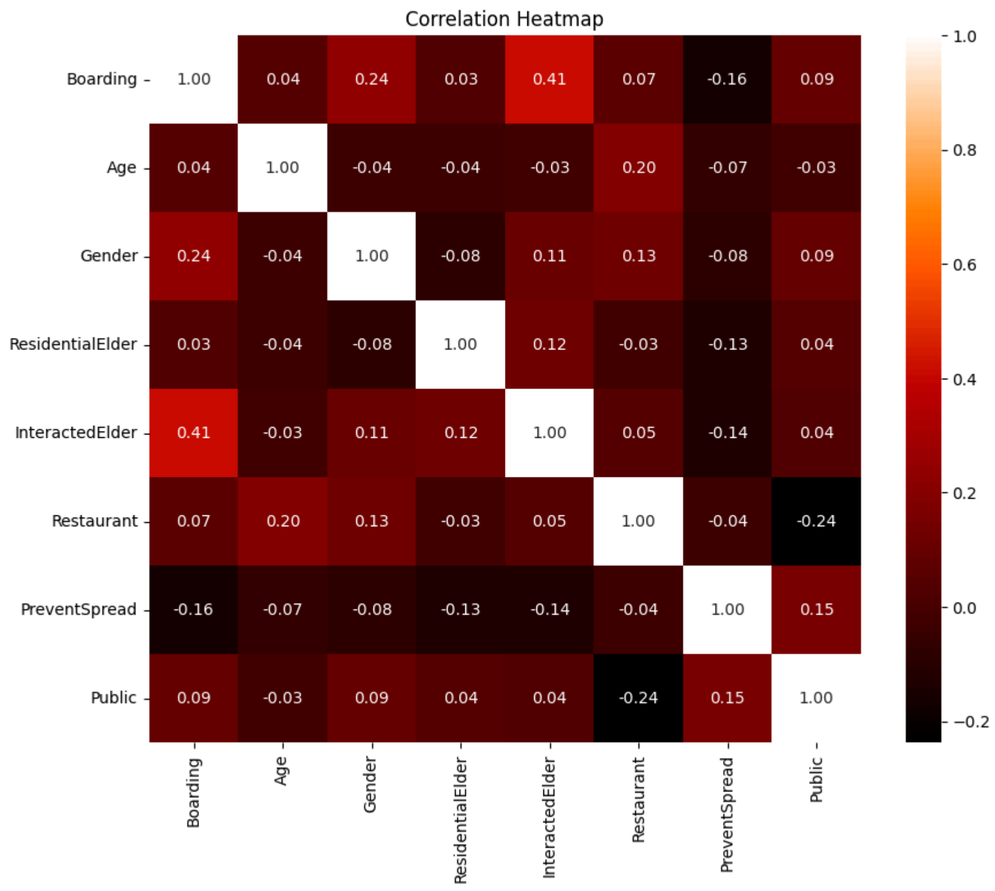
	Timestamp	Boarding	Age	Gender	ResidentialElder	InteractedElder	Restaurant	PreventSpread	Reason	Public
0	9/25/2020 15:04:43	1	16	1	0	1	1.0	1	To protect yourself AND others	1
1	9/25/2020 15:04:46	0	17	0	0	0	2.0	1	To protect yourself AND others	1
2	9/25/2020 15:04:58	0	17	0	0	1	0.0	1	To protect yourself AND others	1
3	9/25/2020 15:05:12	1	17	1	0	1	2.0	1	To protect yourself AND others	1
4	9/25/2020 15:05:12	1	17	1	0	1	2.0	1	To protect yourself AND others	1
...	...	...	...	...	...	...	...	...	...	...
102	9/28/2020 10:56:47	0	17	1	0	0	7.0	1	To protect yourself AND others	1
103	9/28/2020 12:08:13	0	15	0	0	0	1.0	1	To protect yourself AND others	1
104	9/28/2020 13:12:01	0	15	0	1	1	0.0	1	To protect yourself AND others	0
105	9/28/2020 23:27:53	0	16	0	0	0	0.0	1	To protect yourself AND others	1
106	9/29/2020 9:56:52	1	16	1	0	1	2.0	1	To protect yourself AND others	0

107 rows × 10 columns

Next steps: View recommended plots

```
correlation_matrix = WP.corr(method='pearson')
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='gist_heat', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

```
<ipython-input-145-aed63a9ab5e0>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = WP.corr(method='pearson')
```




```
NEW = WP.copy()


columns_to_delete = ['Reason','Timestamp' ]
existing_columns = [col for col in columns_to_delete if col in NEW.columns]

if existing_columns:
    NEW.drop(columns=existing_columns, inplace=True, axis=1)
    print("\nColumns {} deleted successfully.".format(existing_columns))


     Columns ['Reason', 'Timestamp'] deleted successfully.


WP.head(1000)
```

	Timestamp	Boarding	Age	Gender	ResidentialElder	InteractedElder	Restaurant	PreventSpread	Reason	Public
0	9/25/2020 15:04:43	1	16	1	0	1	1.0	1	To protect yourself AND others	1
1	9/25/2020 15:04:46	0	17	0	0	0	2.0	1	To protect yourself AND others	1
2	9/25/2020 15:04:58	0	17	0	0	1	0.0	1	To protect yourself AND others	1
3	9/25/2020 15:05:12	1	17	1	0	1	2.0	1	To protect yourself AND others	1
4	9/25/2020 15:05:12	1	17	1	0	1	2.0	1	To protect yourself AND others	1
...	...	...	...	...	...	...	...	...	...	...
102	9/28/2020 10:56:47	0	17	1	0	0	7.0	1	To protect yourself AND others	1
103	9/28/2020 12:08:13	0	15	0	0	0	1.0	1	To protect yourself AND others	1
104	9/28/2020 13:12:01	0	15	0	1	1	0.0	1	To protect yourself AND others	0
105	9/28/2020 23:27:53	0	16	0	0	0	0.0	1	To protect yourself AND others	1
106	9/29/2020 9:56:52	1	16	1	0	1	2.0	1	To protect yourself AND others	0

107 rows × 10 columns

Next steps: View recommended plots

```
NEW.head(1000)
```

	Boarding	Age	Gender	ResidentialElder	InteractedElder	Restaurant	PreventSpread	Public
0	1	16	1	0	1	1.0	1	1
1	0	17	0	0	0	2.0	1	1
2	0	17	0	0	1	0.0	1	1
3	1	17	1	0	1	2.0	1	1
4	1	17	1	0	1	2.0	1	1
...	...	...	...	...	...	...	...	...
102	0	17	1	0	0	7.0	1	1
103	0	15	0	0	0	1.0	1	1
104	0	15	0	1	1	0.0	1	0
105	0	16	0	0	0	0.0	1	1
106	1	16	1	0	1	2.0	1	0

107 rows × 8 columns

Next steps: View recommended plots

Creating model

```
normalizer = StandardScaler()

NEW.iloc[:, :-1] = normalizer.fit_transform(NEW.iloc[:, :-1])


X2 = NEW.iloc[:, :-1]
y2 = NEW.iloc[:, -1:]


X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.50, random_state=11111)



model = Sequential([
    Flatten(input_shape=(7,)),
    Dense(106, activation='relu'),
    Dense(25, activation='relu'),
    Dense(10, activation='softmax'),
])


model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_10 (Flatten)        (None, 7)                 0

 dense_30 (Dense)            (None, 106)               848

 dense_31 (Dense)            (None, 25)                2675

 dense_32 (Dense)            (None, 10)                260

=================================================================
Total params: 3783 (14.78 KB)
Trainable params: 3783 (14.78 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
model.compile(optimizer='adam',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])



epochs = 0
while epochs < 100:
    model.fit(X2_train, y2_train, epochs=1, batch_size=100, validation_split=0.2)
    print(f"Epoch {epochs+1} completed.")
    epochs += 1
```

```
 1/1 [==============================] - 0s 109ms/step - loss: 0.3366 - accuracy: 0.8810 - val_loss: 1.2158 - val_accuracy: 0.6364
 Epoch 72 completed.
 1/1 [==============================] - 0s 114ms/step - loss: 0.3319 - accuracy: 0.8810 - val_loss: 1.2132 - val_accuracy: 0.6364
 Epoch 73 completed.
 1/1 [==============================] - 0s 98ms/step - loss: 0.3275 - accuracy: 0.8810 - val_loss: 1.2107 - val_accuracy: 0.6364
 Epoch 74 completed.
 1/1 [==============================] - 0s 102ms/step - loss: 0.3233 - accuracy: 0.9048 - val_loss: 1.2082 - val_accuracy: 0.6364
 Epoch 75 completed.
 1/1 [==============================] - 0s 120ms/step - loss: 0.3192 - accuracy: 0.9048 - val_loss: 1.2057 - val_accuracy: 0.6364
 Epoch 76 completed.
 1/1 [==============================] - 0s 104ms/step - loss: 0.3154 - accuracy: 0.9286 - val_loss: 1.2031 - val_accuracy: 0.6364
 Epoch 77 completed.
 1/1 [==============================] - 0s 99ms/step - loss: 0.3118 - accuracy: 0.9286 - val_loss: 1.2007 - val_accuracy: 0.6364
 Epoch 78 completed.
 1/1 [==============================] - 0s 99ms/step - loss: 0.3084 - accuracy: 0.9286 - val_loss: 1.1985 - val_accuracy: 0.6364
 Epoch 79 completed.
 1/1 [==============================] - 0s 104ms/step - loss: 0.3052 - accuracy: 0.9286 - val_loss: 1.1965 - val_accuracy: 0.6364
 Epoch 80 completed.
 1/1 [==============================] - 0s 124ms/step - loss: 0.3021 - accuracy: 0.9286 - val_loss: 1.1948 - val_accuracy: 0.6364
 Epoch 81 completed.
 1/1 [==============================] - 0s 110ms/step - loss: 0.2990 - accuracy: 0.9286 - val_loss: 1.1932 - val_accuracy: 0.5455
 Epoch 82 completed.
 1/1 [==============================] - 0s 111ms/step - loss: 0.2961 - accuracy: 0.9286 - val_loss: 1.1918 - val_accuracy: 0.5455
 Epoch 83 completed.
 1/1 [==============================] - 0s 117ms/step - loss: 0.2932 - accuracy: 0.9286 - val_loss: 1.1905 - val_accuracy: 0.5455
 Epoch 84 completed.
 1/1 [==============================] - 0s 98ms/step - loss: 0.2904 - accuracy: 0.9286 - val_loss: 1.1892 - val_accuracy: 0.5455
 Epoch 85 completed.
 1/1 [==============================] - 0s 128ms/step - loss: 0.2878 - accuracy: 0.9286 - val_loss: 1.1878 - val_accuracy: 0.5455
 Epoch 86 completed.
 1/1 [==============================] - 0s 97ms/step - loss: 0.2854 - accuracy: 0.9286 - val_loss: 1.1866 - val_accuracy: 0.5455
 Epoch 87 completed.
 1/1 [==============================] - 0s 114ms/step - loss: 0.2830 - accuracy: 0.9286 - val_loss: 1.1855 - val_accuracy: 0.5455
 Epoch 88 completed.
 1/1 [==============================] - 0s 113ms/step - loss: 0.2806 - accuracy: 0.9286 - val_loss: 1.1844 - val_accuracy: 0.5455
 Epoch 89 completed.
 1/1 [==============================] - 0s 100ms/step - loss: 0.2783 - accuracy: 0.9286 - val_loss: 1.1833 - val_accuracy: 0.5455
 Epoch 90 completed.
 1/1 [==============================] - 0s 113ms/step - loss: 0.2761 - accuracy: 0.9286 - val_loss: 1.1824 - val_accuracy: 0.5455
 Epoch 91 completed.
 1/1 [==============================] - 0s 109ms/step - loss: 0.2739 - accuracy: 0.9286 - val_loss: 1.1817 - val_accuracy: 0.5455
 Epoch 92 completed.
 1/1 [==============================] - 0s 114ms/step - loss: 0.2717 - accuracy: 0.9286 - val_loss: 1.1812 - val_accuracy: 0.5455
 Epoch 93 completed.
 1/1 [==============================] - 0s 133ms/step - loss: 0.2697 - accuracy: 0.9286 - val_loss: 1.1807 - val_accuracy: 0.5455
 Epoch 94 completed.
 1/1 [==============================] - 0s 140ms/step - loss: 0.2677 - accuracy: 0.9286 - val_loss: 1.1804 - val_accuracy: 0.5455
 Epoch 95 completed.
 1/1 [==============================] - 0s 135ms/step - loss: 0.2658 - accuracy: 0.9286 - val_loss: 1.1803 - val_accuracy: 0.5455
 Epoch 96 completed.
 1/1 [==============================] - 0s 174ms/step - loss: 0.2639 - accuracy: 0.9286 - val_loss: 1.1803 - val_accuracy: 0.5455
 Epoch 97 completed.
 1/1 [==============================] - 0s 161ms/step - loss: 0.2621 - accuracy: 0.9286 - val_loss: 1.1805 - val_accuracy: 0.5455
 Epoch 98 completed.
 1/1 [==============================] - 0s 109ms/step - loss: 0.2603 - accuracy: 0.9286 - val_loss: 1.1807 - val_accuracy: 0.5455
 Epoch 99 completed.
 1/1 [==============================] - 0s 123ms/step - loss: 0.2586 - accuracy: 0.9286 - val_loss: 1.1808 - val_accuracy: 0.5455
 Epoch 100 completed.
```

Evaluate the accuracy of your model

```
results = model.evaluate(X2_test, y2_test, verbose=1)
print('Test loss, test accuracy:', results)
```

```
 2/2 [==============================] - 0s 9ms/step - loss: 0.6431 - accuracy: 0.7963
 Test loss, test accuracy: [0.6430695056915283, 0.7962962985038757]
```

The dataset I used in this activity aims to identify how the habits and beliefs of teenagers affects their decision to wear a face mask in public, even when it is not required during the COVID-19 pandemic base on multiple different criteria that serve as features for training a neural network model to predict whether teenagers wear face masks based on these parameters. After training with 100 epochs and 100 batches, the accuracy i achieved is 79%, which is quite low so it's not quite too accurate.