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

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: 963 entries, 0 to 962
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Timestamp         107 non-null    object
 1   Boarding          963 non-null    object
 2   Age               963 non-null    int64
 3   Gender            954 non-null    object
 4   ResidentialElder  963 non-null    object
 5   InteractedElder   963 non-null    object
 6   Restaurant        945 non-null    float64
 7   PreventSpread     963 non-null    object
 8   Reason            963 non-null    object
 9   Public            963 non-null    object
dtypes: float64(1), int64(1), object(8)
memory usage: 75.4+ 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
...	...	...	...	...	...	...	...	...	...	...
958	NaN	Boarding	17	Female	No	No	7.0	Yes	To protect yourself AND others	Yes
959	NaN	Boarding	15	Male	No	No	1.0	Yes	To protect yourself AND others	Yes
960	NaN	Boarding	15	Male	Yes	Yes	0.0	Yes	To protect yourself AND others	No
961	NaN	Boarding	16	Male	No	No	0.0	Yes	To protect yourself AND others	Yes
962	NaN	Day	16	Female	No	Yes	2.0	Yes	To protect yourself AND others	No

963 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
...	...	...	...	...	...	...	...	...	...	...
958	0	Boarding	17	Female	No	No	7.0	Yes	To protect yourself AND others	Yes
959	0	Boarding	15	Male	No	No	1.0	Yes	To protect yourself AND others	Yes
960	0	Boarding	15	Male	Yes	Yes	0.0	Yes	To protect yourself AND others	No
961	0	Boarding	16	Male	No	No	0.0	Yes	To protect yourself AND others	Yes
962	0	Day	16	Female	No	Yes	2.0	Yes	To protect yourself AND others	No

963 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)
```

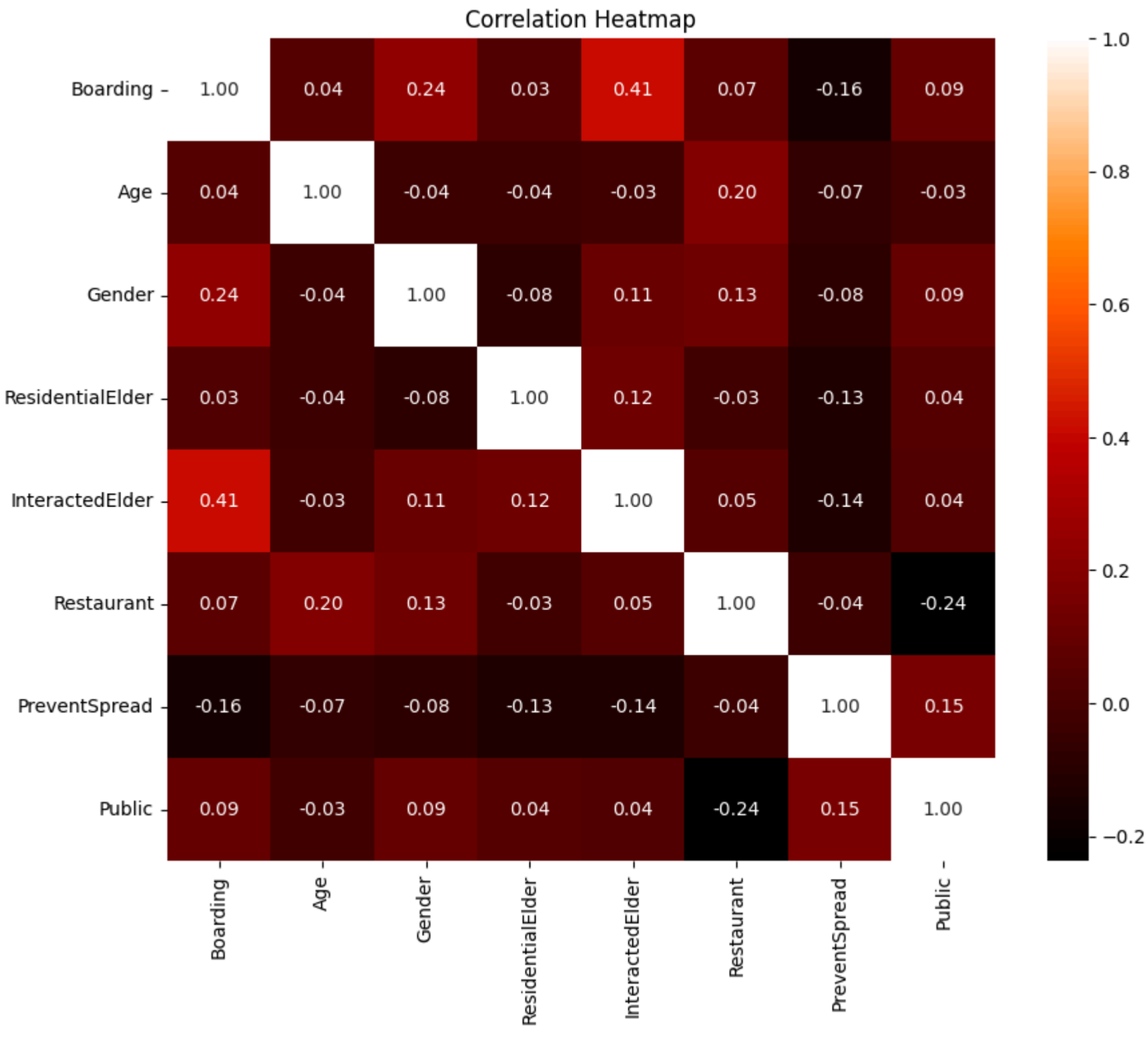
	Timestamp	Boarding	Age	Gender	ResidentialElder	InteractedElder	Restaurant	PreventSpread	Reason	Public
0	9/25/2020 15:04:43	1	16	1	0	1	1.0	1	To protect yourself AND others	1
1	9/25/2020 15:04:46	0	17	0	0	0	2.0	1	To protect yourself AND others	1
2	9/25/2020 15:04:58	0	17	0	0	1	0.0	1	To protect yourself 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
...	...	...	...	...	...	...	...	...	...	...
958	0	0	17	1	0	0	7.0	1	To protect yourself AND others	1
959	0	0	15	0	0	0	1.0	1	To protect yourself AND others	1
960	0	0	15	0	1	1	0.0	1	To protect yourself AND others	0
961	0	0	16	0	0	0	0.0	1	To protect yourself AND others	1
962	0	1	16	1	0	1	2.0	1	To protect yourself AND others	0

963 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-47-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 speci
  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
...	...	...	...	...	...	...	...	...	...	...
958	0	0	17	1	0	0	7.0	1	To protect yourself AND others	1
959	0	0	15	0	0	0	1.0	1	To protect yourself AND others	1
960	0	0	15	0	1	1	0.0	1	To protect yourself AND others	0
961	0	0	16	0	0	0	0.0	1	To protect yourself AND others	1
962	0	1	16	1	0	1	2.0	1	To protect yourself AND others	0

963 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
...	...	...	...	...	...	...	...	...
958	0	17	1	0	0	7.0	1	1
959	0	15	0	0	0	1.0	1	1
960	0	15	0	1	1	0.0	1	0
961	0	16	0	0	0	0.0	1	1
962	1	16	1	0	1	2.0	1	0

963 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(962, activation='relu'),
    Dense(250, activation='relu'),
    Dense(10, activation='softmax'),
])
```

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_2 (Flatten)         (None, 7)                 0

 dense_6 (Dense)             (None, 962)               7696

 dense_7 (Dense)             (None, 250)               240750

 dense_8 (Dense)             (None, 10)                2510

=================================================================
Total params: 250956 (980.30 KB)
Trainable params: 250956 (980.30 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

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

```
epochs = 0
while epochs < 900:
    model.fit(X2_train, y2_train, epochs=1, batch_size=900, validation_split=0.2)
    print(f"Epoch {epochs+1} completed.")
    epochs += 1
```

```
 1/1 [==============================] - 0s 136ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4104 - val_accuracy: 0.8969
 Epoch 872 completed.
 1/1 [==============================] - 0s 165ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4139 - val_accuracy: 0.8866
 Epoch 873 completed.
 1/1 [==============================] - 0s 147ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4106 - val_accuracy: 0.8969
 Epoch 874 completed.
 1/1 [==============================] - 0s 151ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4135 - val_accuracy: 0.8866
 Epoch 875 completed.
 1/1 [==============================] - 0s 130ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4111 - val_accuracy: 0.8969
 Epoch 876 completed.
 1/1 [==============================] - 0s 115ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4128 - val_accuracy: 0.8866
 Epoch 877 completed.
 1/1 [==============================] - 0s 145ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4118 - val_accuracy: 0.8969
 Epoch 878 completed.
 1/1 [==============================] - 0s 132ms/step - loss: 0.1254 - accuracy: 0.9271 - val_loss: 0.4123 - val_accuracy: 0.8866
 Epoch 879 completed.
 1/1 [==============================] - 0s 132ms/step - loss: 0.1254 - accuracy: 0.9271 - val_loss: 0.4125 - val_accuracy: 0.8866
 Epoch 880 completed.
 1/1 [==============================] - 0s 131ms/step - loss: 0.1254 - accuracy: 0.9271 - val_loss: 0.4118 - val_accuracy: 0.8969
 Epoch 881 completed.
 1/1 [==============================] - 0s 236ms/step - loss: 0.1254 - accuracy: 0.9271 - val_loss: 0.4130 - val_accuracy: 0.8866
 Epoch 882 completed.
 1/1 [==============================] - 0s 178ms/step - loss: 0.1254 - accuracy: 0.9271 - val_loss: 0.4115 - val_accuracy: 0.8969
 Epoch 883 completed.
 1/1 [==============================] - 0s 221ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4134 - val_accuracy: 0.8866
Epoch 884 completed.
1/1 [==============================] - 0s 204ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4113 - val_accuracy: 0.8969
Epoch 885 completed.
1/1 [==============================] - 0s 225ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4139 - val_accuracy: 0.8866
Epoch 886 completed.
1/1 [==============================] - 0s 162ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4111 - val_accuracy: 0.8969
Epoch 887 completed.
1/1 [==============================] - 0s 134ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4142 - val_accuracy: 0.8866
Epoch 888 completed.
1/1 [==============================] - 0s 133ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4111 - val_accuracy: 0.8969
Epoch 889 completed.
1/1 [==============================] - 0s 164ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4143 - val_accuracy: 0.8866
Epoch 890 completed.
1/1 [==============================] - 0s 148ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4112 - val_accuracy: 0.8969
Epoch 891 completed.
1/1 [==============================] - 0s 127ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4143 - val_accuracy: 0.8866
Epoch 892 completed.
1/1 [==============================] - 0s 137ms/step - loss: 0.1255 - accuracy: 0.9271 - val_loss: 0.4114 - val_accuracy: 0.8969
Epoch 893 completed.
1/1 [==============================] - 0s 126ms/step - loss: 0.1256 - accuracy: 0.9271 - val_loss: 0.4143 - val_accuracy: 0.8866
Epoch 894 completed.
1/1 [==============================] - 0s 117ms/step - loss: 0.1256 - accuracy: 0.9271 - val_loss: 0.4116 - val_accuracy: 0.8969
Epoch 895 completed.
1/1 [==============================] - 0s 120ms/step - loss: 0.1257 - accuracy: 0.9271 - val_loss: 0.4145 - val_accuracy: 0.8866
Epoch 896 completed.
1/1 [==============================] - 0s 132ms/step - loss: 0.1258 - accuracy: 0.9271 - val_loss: 0.4117 - val_accuracy: 0.8969
Epoch 897 completed.
1/1 [==============================] - 0s 149ms/step - loss: 0.1259 - accuracy: 0.9271 - val_loss: 0.4150 - val_accuracy: 0.8866
Epoch 898 completed.
1/1 [==============================] - 0s 119ms/step - loss: 0.1260 - accuracy: 0.9271 - val_loss: 0.4114 - val_accuracy: 0.8969
Epoch 899 completed.
1/1 [==============================] - 0s 116ms/step - loss: 0.1260 - accuracy: 0.9271 - val_loss: 0.4156 - val_accuracy: 0.8866
Epoch 900 completed.
```

Evaluate the accuracy of your model

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

```
16/16 [==============================] - 0s 3ms/step - loss: 0.3423 - accuracy: 0.8942
Test loss, test accuracy: [0.3423386514186859, 0.8941908478736877]
```

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 900 epochs and 900 batches, the accuracy i achieved is 89%, which is quite acceptable but not too accurate.