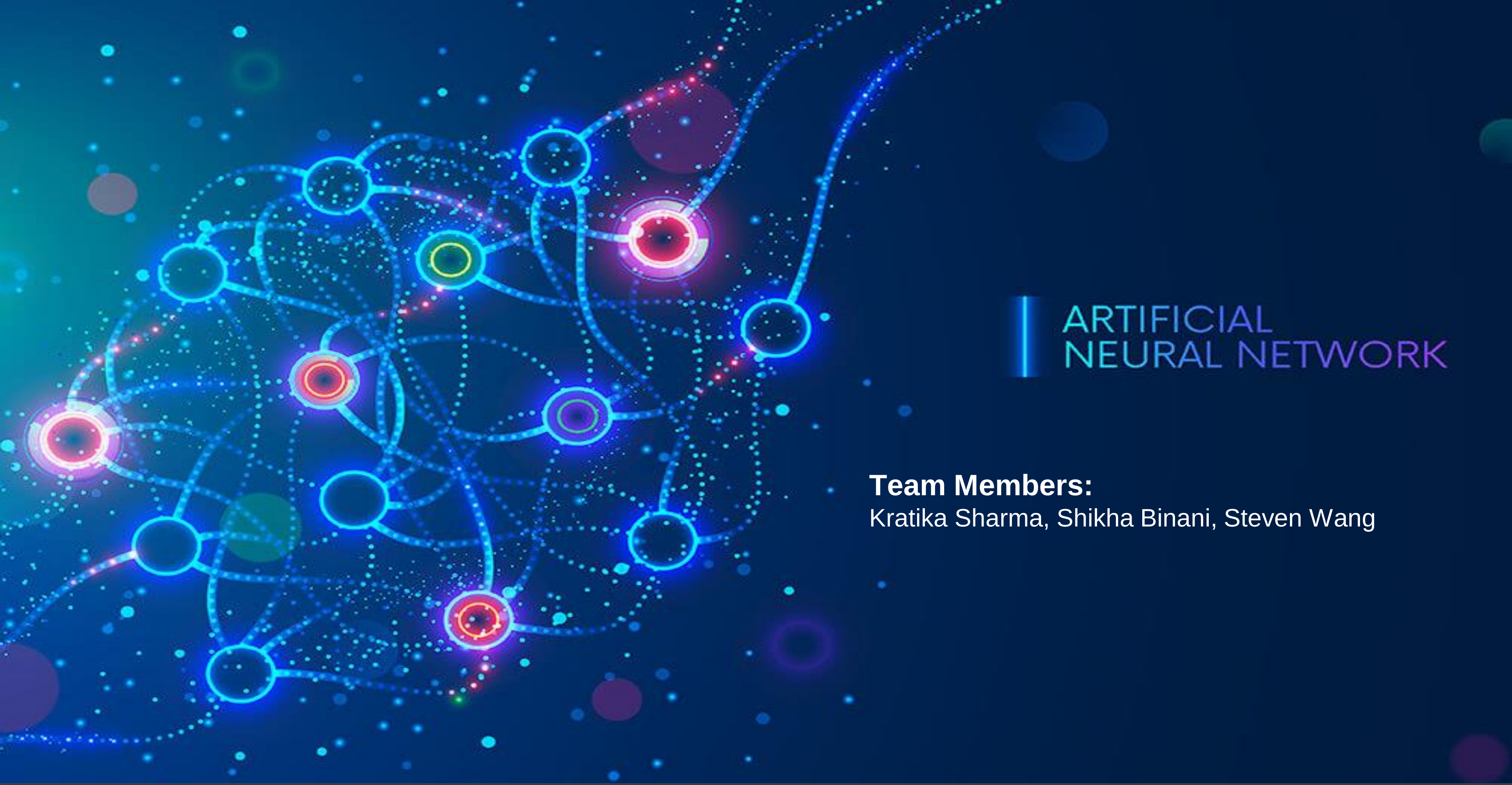
Mental Health Prediction using Artificial Neural Network



Team Members:
Kratika Sharma, Shikha Binani, Steven Wang

Why Artificial Neural Network?

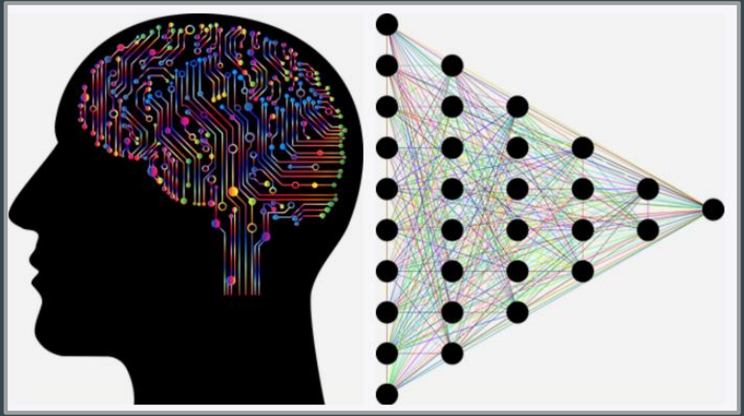
Ability to Analyze collection of connected units.

Self-Learning Capabilities - As more trials or iterations, the better the results

Most comfortable set up is a binary classification with only two classes: 0 and 1

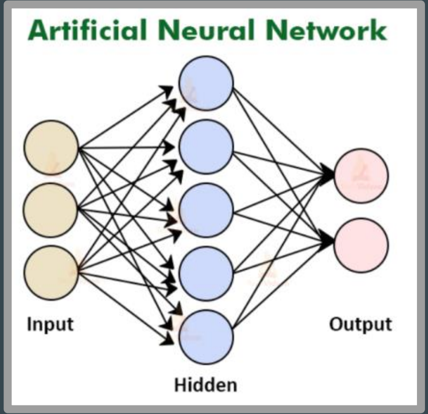
Produce better results as more data available

(Great for our dataset, since X variable responses might be connected)



Main Principal Objects

1. Layers: all the learning occurs in the layers. There are 3 layers:



- Input
- Hidden
- Output

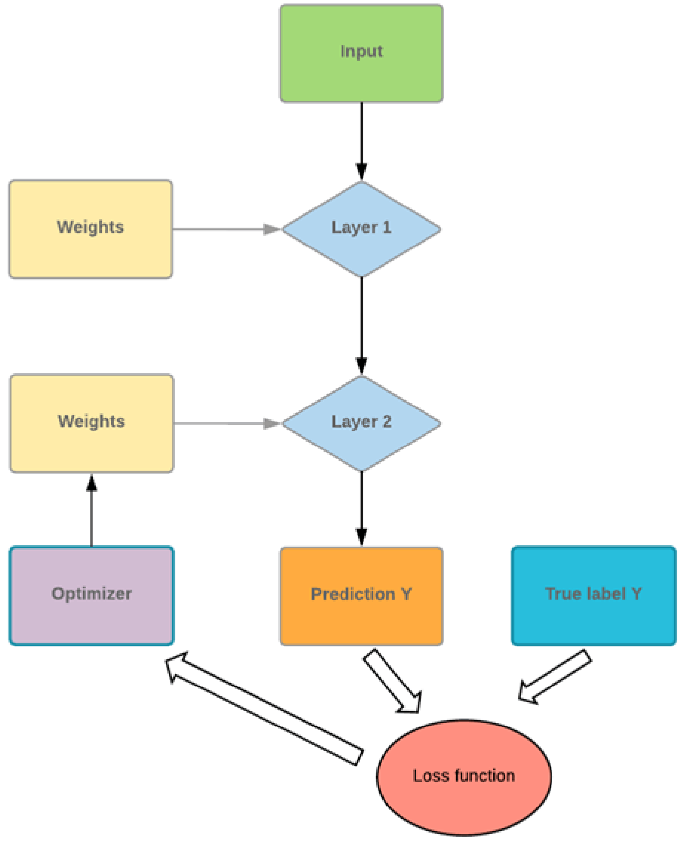
1. Feature and Label: Input data to the network (features) and output from the network (labels)

1. Loss function: Metric used to estimate the performance of the learning phase

1. Optimizer: Improve the learning by updating the knowledge in the network

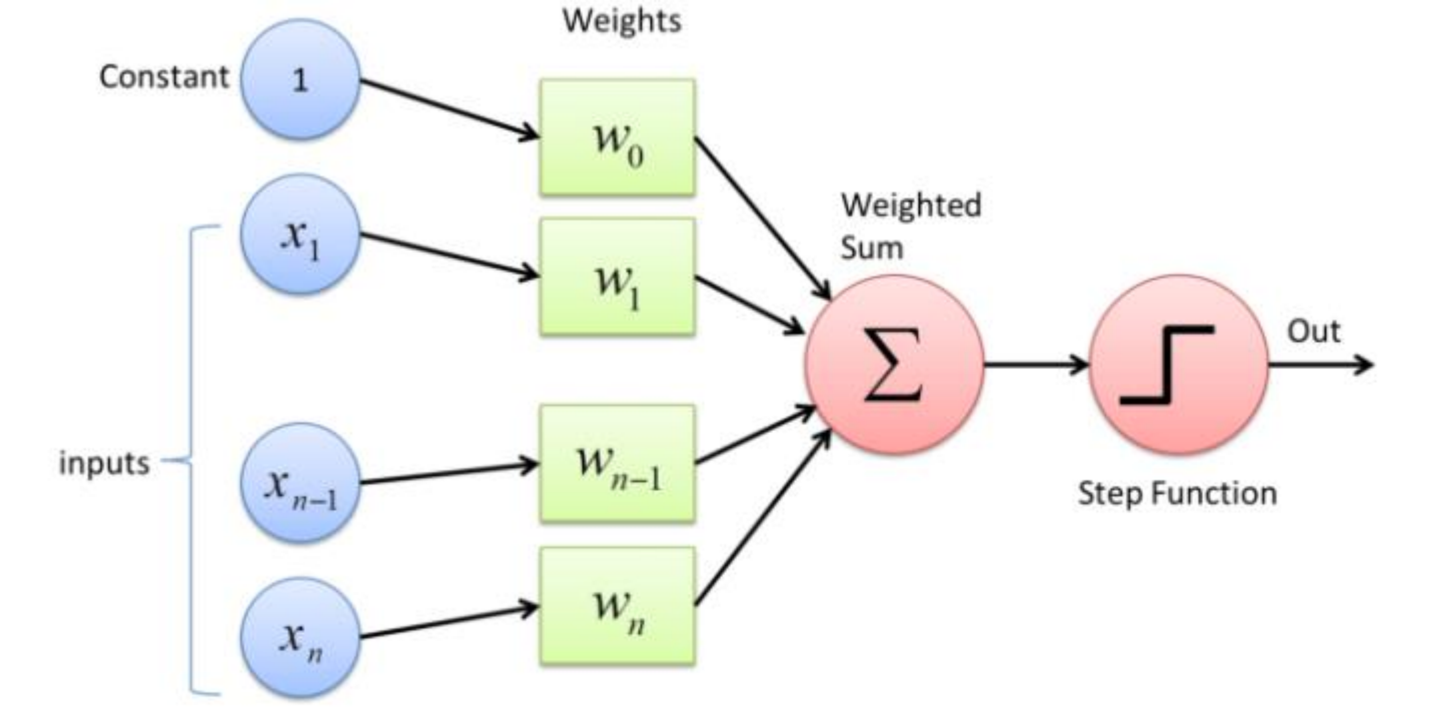
Framework of ANN

Basic Framework of an Artificial Neural Network, using 2 features.



Artificial Neural Network

Perceptron



Weights
Constant
1
w_0
x_1
w_1
Weighted
Sum
Σ
Out
inputs
x_{n-1}
w_{n-1}
Step Function
x_n
w_n

Activation Function

Activation function to allow the network to learn non-linear pattern.

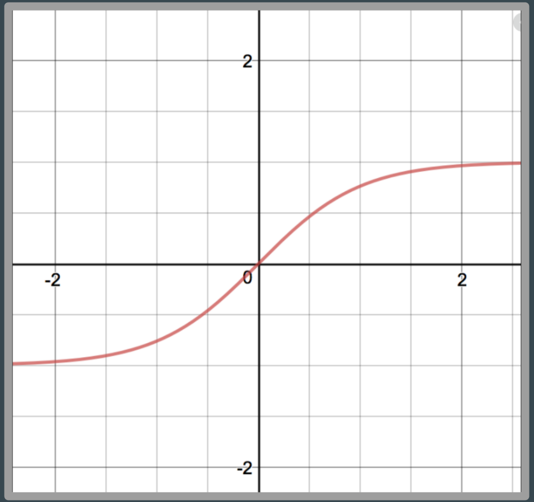
Common Activation Functions:

- Piecewise Linear
- Sigmoid
- Tanh
- Leaky Relu

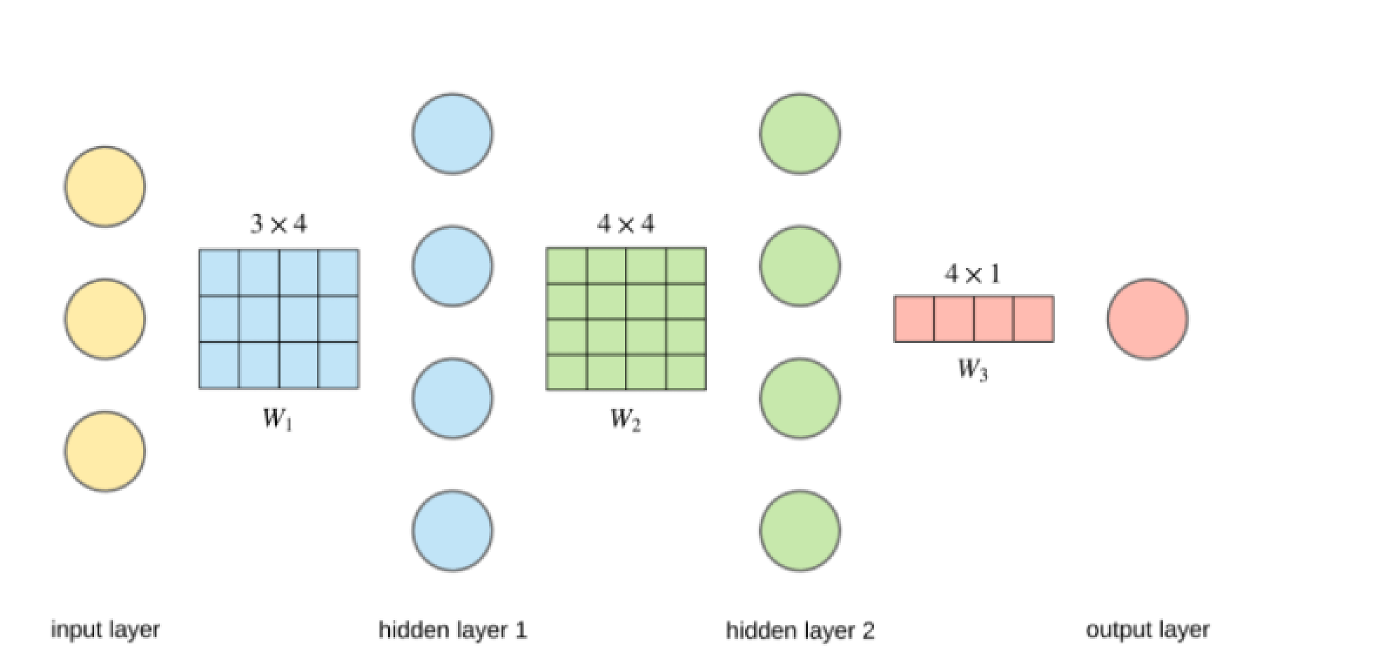
In short, the activation functions are used to map the input between the required values like (0, 1) or (-1, 1).

Model parameters (like W) can change shape of F(x)

Neurons will “learn” parts of these relationships, while others are “inactive’ in specific ranges.



Computation



Fitting an ANN model

- May not extrapolate well
  - Only “learn” from data given
  - Include many combinations of inputs
  - The more training data, the better (repetition mentioned in previous slide)
- Use an appropriate set of neurons. Using too many will result in overfitting.
- Make sure to verify ANN model using outside data.
- Can regularize to reduce overfitting
- Adjust weight to reduce error.

Example Neural Network in TensorFlow

Let's see an Artificial Neural Network example in action.

2 Inputs: x1 and x2 with a random value.

Output: Binary

The objective is to classify the label based on the two features.

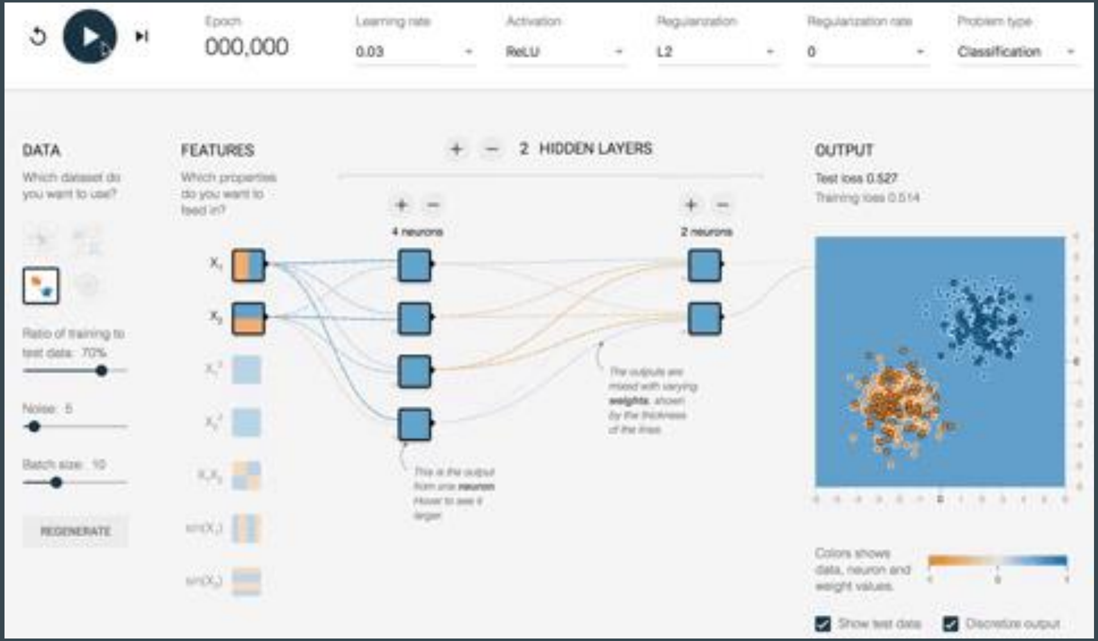
To carry out this task, the neural network architecture is defined as following:

Two hidden layers

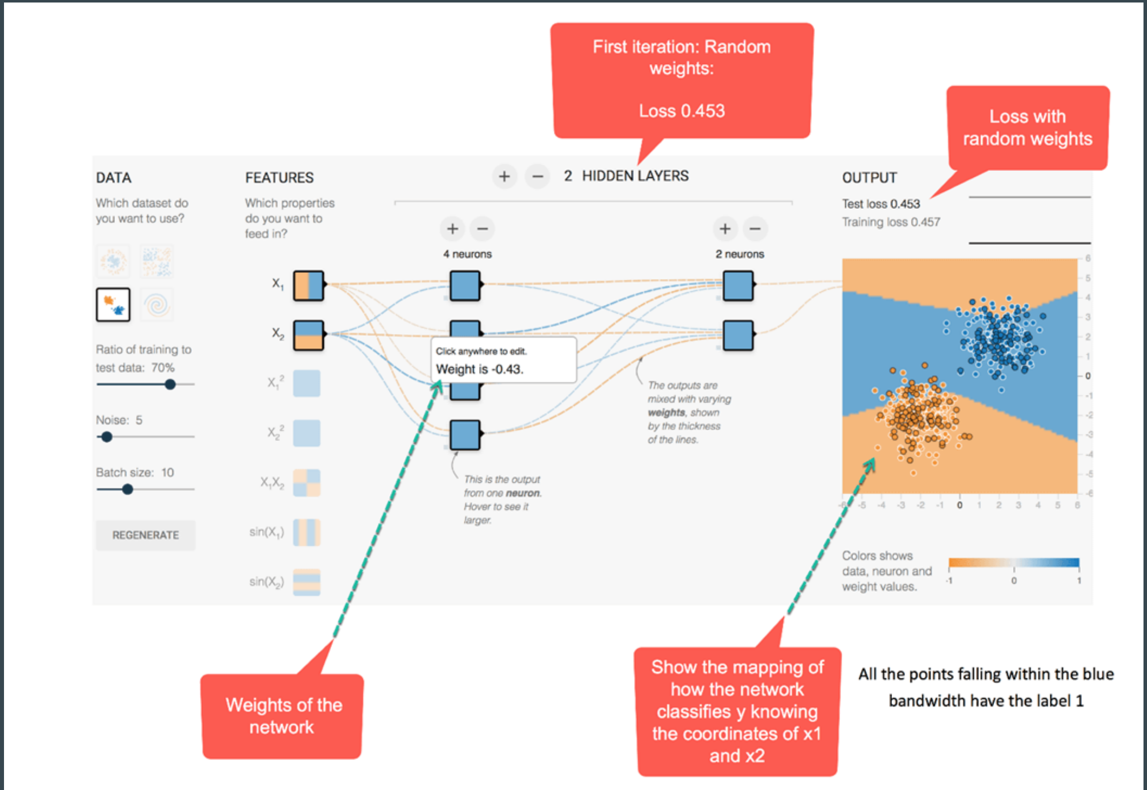
- First layer has four fully connected neurons
- Second layer has two fully connected neurons

The activation function is a **Relu**

Add an L2 Regularization with a **learning rate of 0.003**



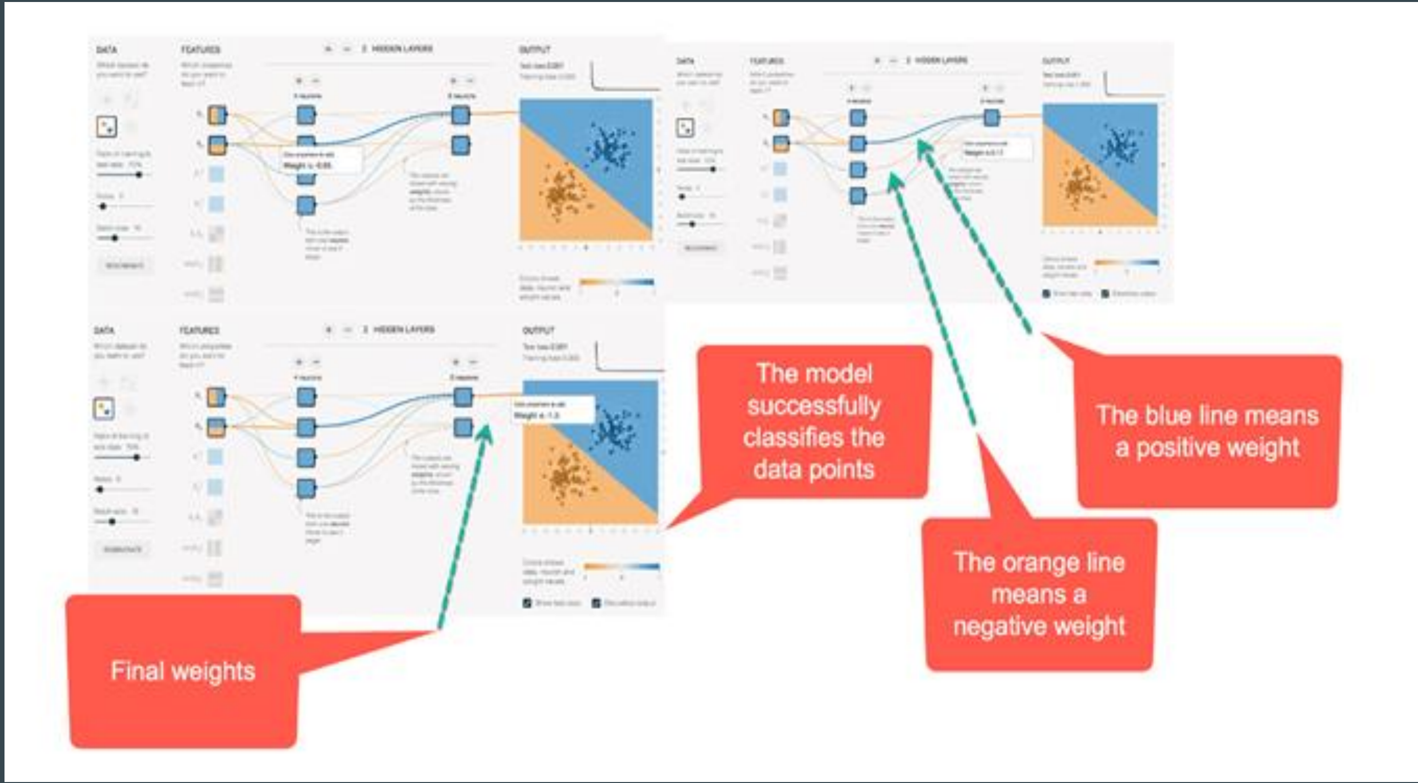
Weights without Optimization



Orange = Negative

Blue = Positive

Adjusting Weights with Optimization



Initial Weight:
-0.43

Weight after Optimization:
-0.95

Dataset - Mental Health Prediction

- 2014 data
- 1,259 Tech Industry Employees
- Western countries
- Measures :
  - Attitudes towards mental health
  - Frequency of Mental Health Disorders

Dataset Relevant Columns

- Age
- Gender
- Country
- State
- Self Employed
- Work_interfere
- Leave

- Family history
- Treatment
- No_employees
- Wellness_program
- Remote_work
- tech_company

Objectives

To predict:

1. Treatment for employees
2. Work Interference due to mental illness

Data Cleaning

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_employees	...	leave	mental_health_conse
0	2014-08-27 11:29:31	37	Female	United States	IL	NaN	No	Yes	Often	6-25	...	Somewhat easy	
1	2014-08-27 11:29:37	44	M	United States	IN	NaN	No	No	Rarely	More than 1000	...	Don't know	
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No	Rarely	6-25	...	Somewhat difficult	
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often	26-100	...	Somewhat difficult	
4	2014-08-27 11:30:22	31	Male	United States	TX	NaN	No	No	Never	100-500	...	Don't know	

5 rows × 27 columns

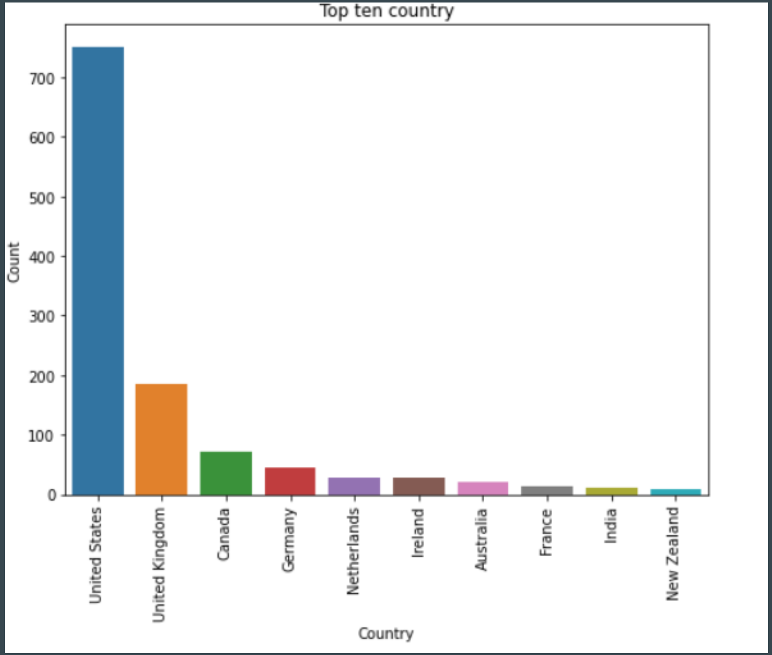
- Special characters (!'@:,-#)
- NaN cells
- Multiple items for cell.
- Classification for columns like gender
- Irrelevant/Meaningless columns like timestamp
- Non-Data related Columns like comments

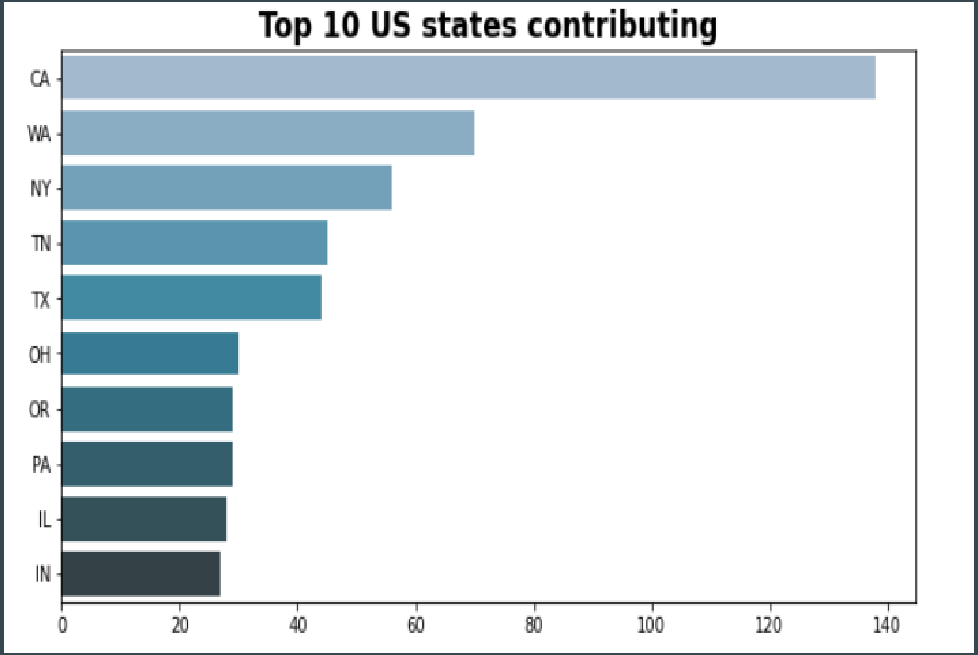
Cleaned File

	Age	Gender	self_employed	family_history	treatment	work_interfere	no_employees	remote_work	tech_company	obs_consequence	...	cw_Yes	su_M
0	37	0	0	0	1	3	1	0	1	0	...	1	
1	44	1	0	0	0	1	5	0	0	0	...	0	
2	32	1	0	0	0	1	1	0	1	0	...	1	
3	31	1	0	1	1	3	2	0	1	1	...	0	
4	31	1	0	0	0	0	3	1	1	0	...	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1254	26	1	0	0	1	2	2	0	1	0	...	0	
1255	32	1	0	1	1	3	2	1	1	0	...	1	
1256	34	1	0	1	1	2	5	0	1	0	...	0	
1257	46	0	0	0	0	2	3	1	1	0	...	0	
1258	25	1	0	1	1	2	2	0	0	0	...	0	

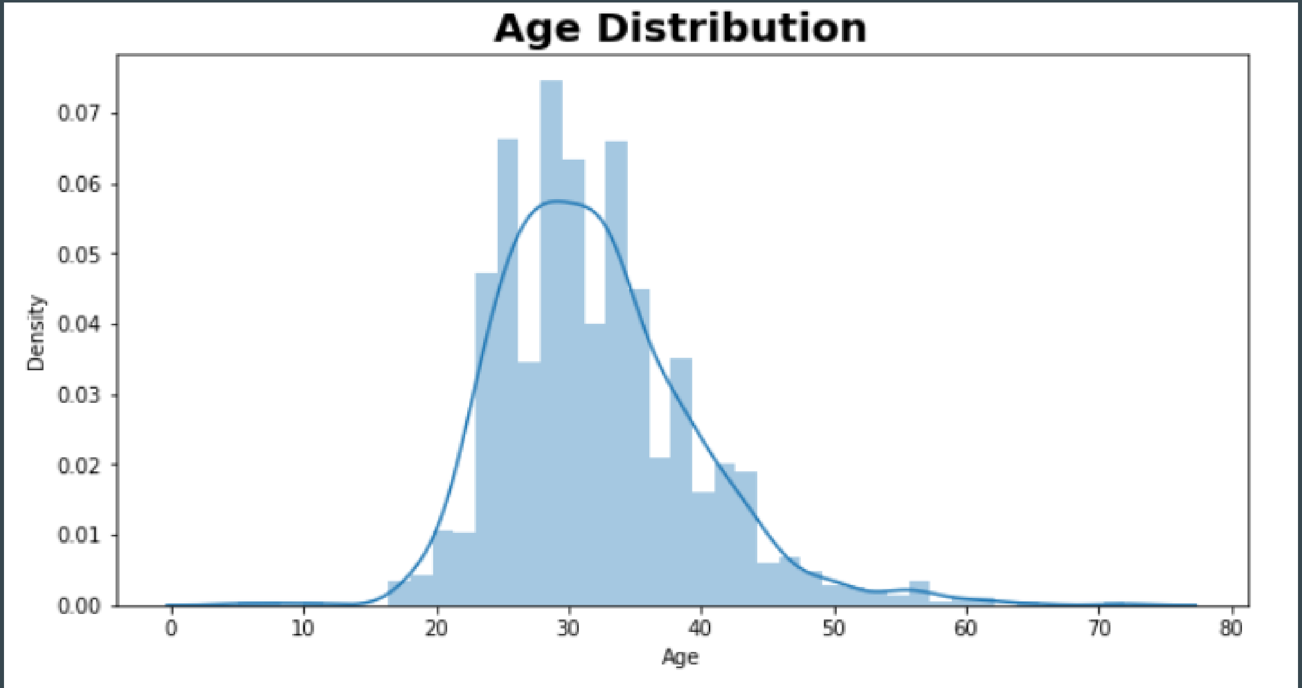
1259 rows × 144 columns

Data Visualization

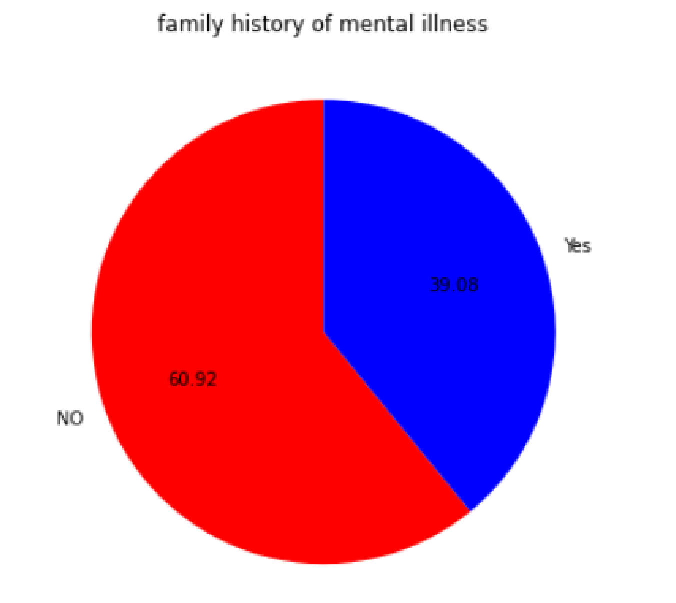


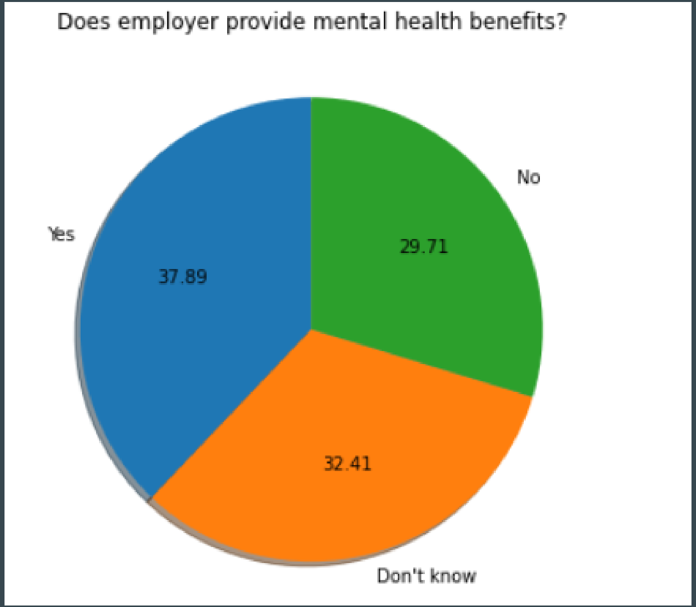


Data Visualization

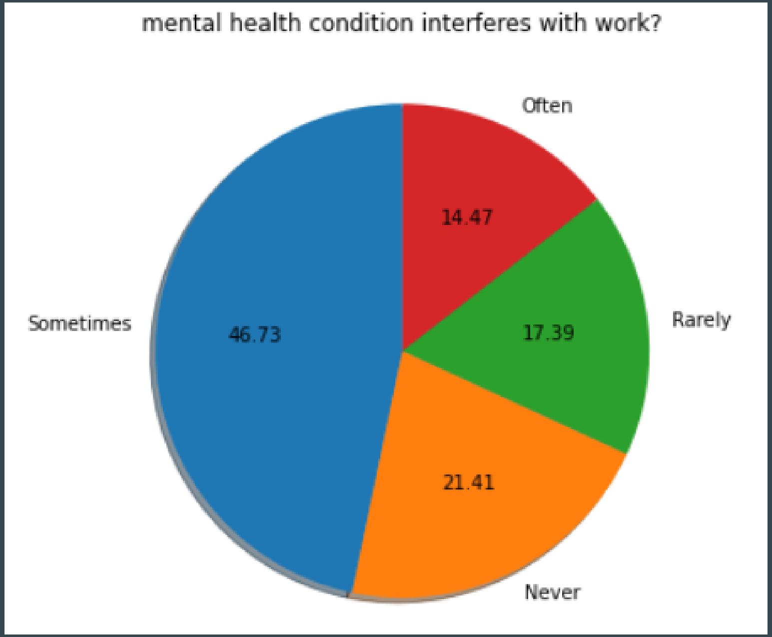


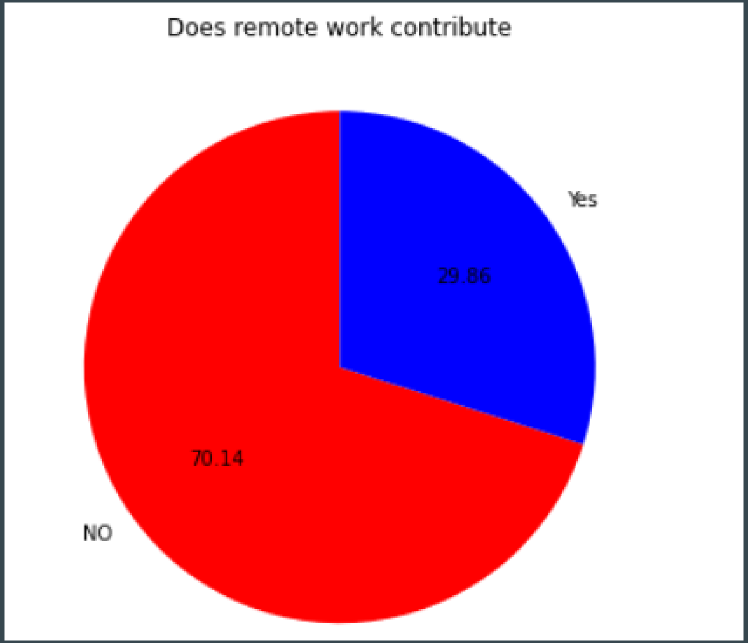
Data Visualization



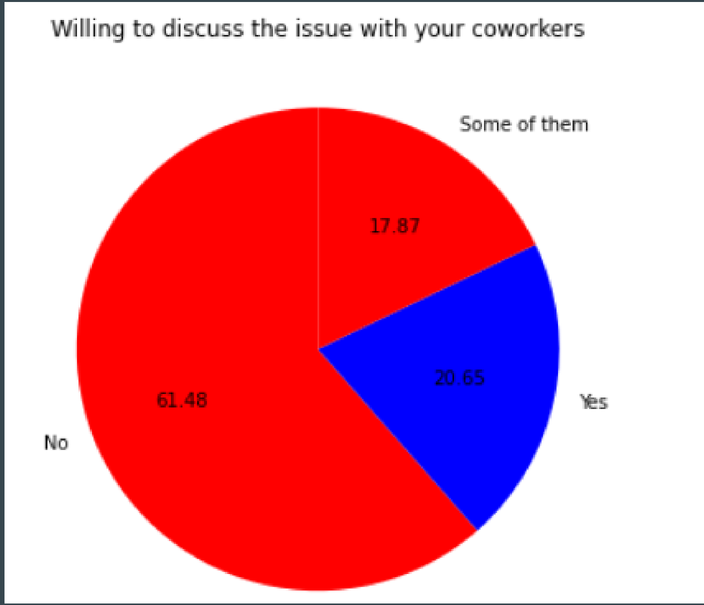


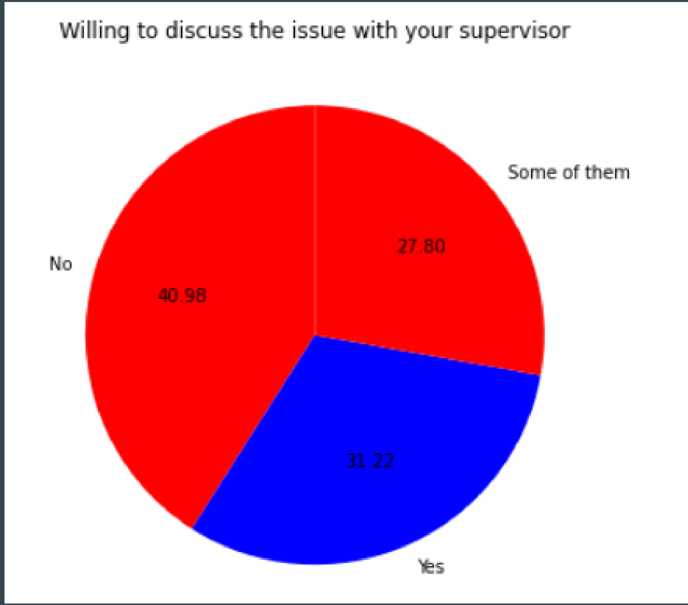
Data Visualization





Data Visualization





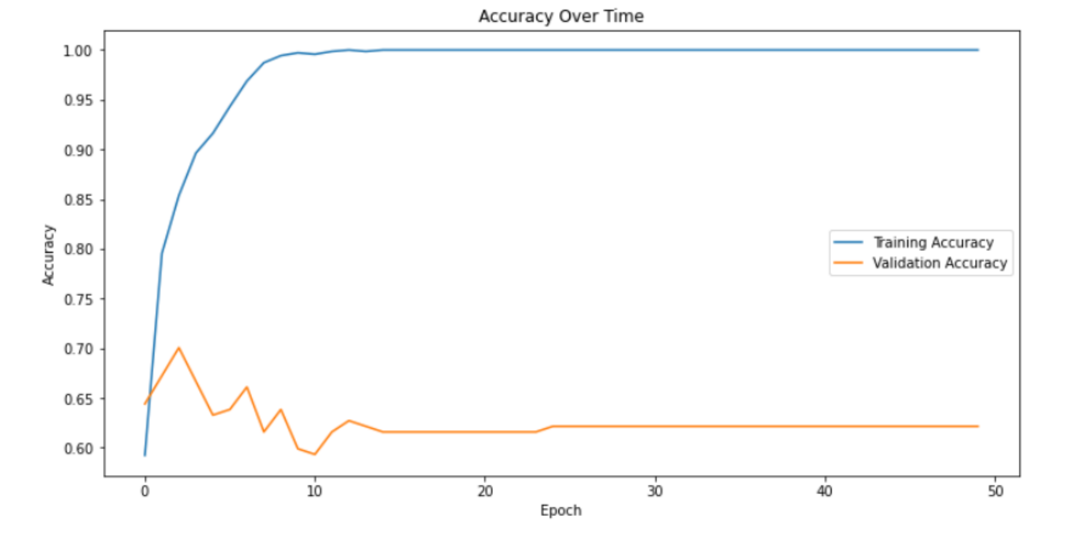
Model : ANN

Batch size = 64
Y variable = 'Treatment'

```
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=[
        'accuracy',
        tf.keras.metrics.AUC(name='auc')
    ]
)
```

```
Epoch 1/75
11/11 [==============================] - 1s 42ms/step - loss: 0.7691 - accuracy: 0.5413 - auc: 0.5858 - val_loss: 0.6576 - va
l_accuracy: 0.6328 - val_auc: 0.6976
Epoch 2/75
11/11 [==============================] - 0s 23ms/step - loss: 0.4692 - accuracy: 0.7674 - auc: 0.8719 - val_loss: 0.6762 - va
l_accuracy: 0.6667 - val_auc: 0.7253
Epoch 3/75
11/11 [==============================] - 0s 22ms/step - loss: 0.3723 - accuracy: 0.8373 - auc: 0.9210 - val_loss: 0.8133 - va
l_accuracy: 0.6158 - val_auc: 0.6914
Epoch 4/75
11/11 [==============================] - 0s 17ms/step - loss: 0.3094 - accuracy: 0.8734 - auc: 0.9554 - val_loss: 0.8101 - va
l_accuracy: 0.6780 - val_auc: 0.7005
Epoch 5/75
11/11 [==============================] - 0s 44ms/step - loss: 0.2445 - accuracy: 0.8972 - auc: 0.9718 - val_loss: 0.9194 - va
l_accuracy: 0.6610 - val_auc: 0.6967
Epoch 6/75
11/11 [==============================] - 0s 11ms/step - loss: 0.2034 - accuracy: 0.9116 - auc: 0.9848 - val_loss: 0.9955 - va
l_accuracy: 0.6441 - val_auc: 0.6847
Epoch 7/75
```

Model : ANN (Y variable as Treatment)




```
model.evaluate(X_test, y_test)
```

```
12/12 [==============================] - 0s 4ms/step - loss: 1.5133 - accuracy: 0.6825 - auc: 0.7240
```

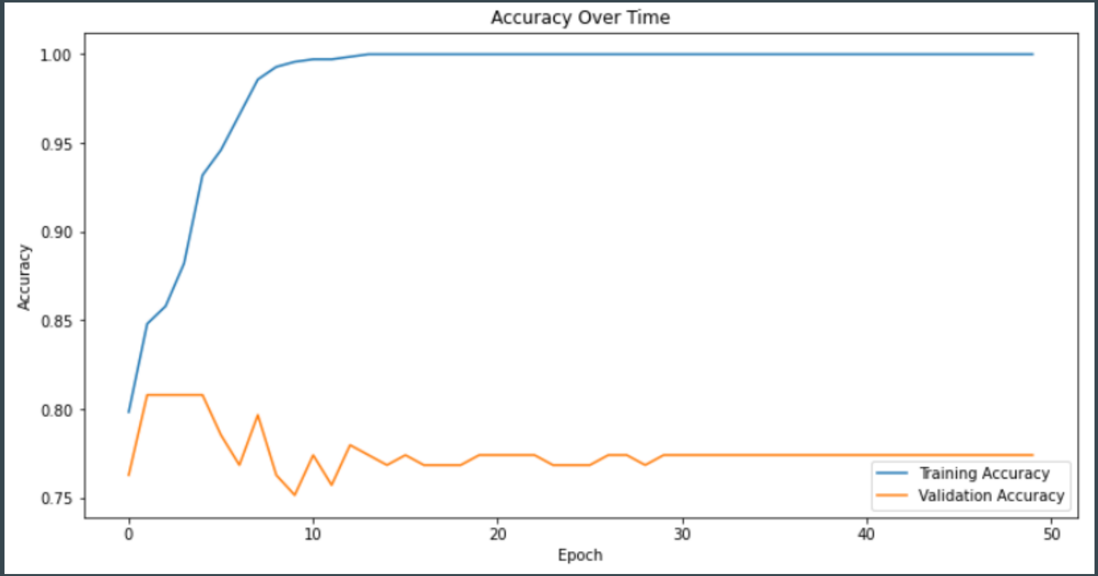
Model : ANN

Batch size = 64
Y variable = 'Work_interference'

```
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=[
        'accuracy',
        tf.keras.metrics.AUC(name='auc')
    ]
)
```

```
Epoch 1/50
11/11 [==============================] - 1s 46ms/step - loss: 0.6374 - accuracy: 0.7754 - auc: 0.5262 - val_loss: 0.5490 - va
l_accuracy: 0.7627 - val_auc: 0.7068
Epoch 2/50
11/11 [==============================] - 0s 19ms/step - loss: 0.4394 - accuracy: 0.8543 - auc: 0.7789 - val_loss: 0.5294 - va
l_accuracy: 0.8079 - val_auc: 0.6665
Epoch 3/50
11/11 [==============================] - 0s 17ms/step - loss: 0.3238 - accuracy: 0.8502 - auc: 0.8672 - val_loss: 0.4843 - va
l_accuracy: 0.8079 - val_auc: 0.6815
Epoch 4/50
11/11 [==============================] - 0s 17ms/step - loss: 0.2604 - accuracy: 0.8875 - auc: 0.9345 - val_loss: 0.5841 - va
l_accuracy: 0.8079 - val_auc: 0.6617
Epoch 5/50
11/11 [==============================] - 0s 14ms/step - loss: 0.2081 - accuracy: 0.9310 - auc: 0.9596 - val_loss: 0.6596 - va
l_accuracy: 0.8079 - val_auc: 0.6515
Epoch 6/50
11/11 [==============================] - 0s 14ms/step - loss: 0.1569 - accuracy: 0.9471 - auc: 0.9800 - val_loss: 0.7091 - va
l_accuracy: 0.7853 - val_auc: 0.6430
```

Model : ANN (Y variable as Work_Interference)




```
model.evaluate(X_test, y_test)
```

```
12/12 [==============================] - 0s 3ms/step - loss: 1.4652 - accuracy: 0.7937 - auc: 0.5936
```

Conclusion

Our ANN model predicts:

- Whether an employee should be treated with 68% accuracy
- Whether mental illness will interfere with 79% accuracy


THANK YOU!