Lung Cancer Prediction with Deep Learning

In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
sns.set_palette("Set3")
from matplotlib import style
style.use("ggplot")
import os
for dirname, _, filenames in os.walk('/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [2]:
```
df = pd.read_csv("../input/cancer patient data sets.csv")
```

Checking the data

In [3]:
```
df.head()
```

Out[3]:

	index	Patient Id	Age	Gender	Air Pollution	Alcohol use	Dust Allergy	OccuPational Hazards	Genetic Risk	chronic Lung Disease	...	Fatig
0	0	P1	33	1	2	4	5	4	3	2	...	
1	1	P10	17	1	3	1	5	3	4	2	...	
2	2	P100	35	1	4	5	6	5	5	4	...	
3	3	P1000	37	1	7	7	7	7	6	7	...	
4	4	P101	46	1	6	8	7	7	7	6	...	

5 rows × 26 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 26 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   index                     1000 non-null   int64
 1   Patient Id                1000 non-null   object
 2   Age                       1000 non-null   int64
 3   Gender                    1000 non-null   int64
 4   Air Pollution             1000 non-null   int64
 5   Alcohol use               1000 non-null   int64
 6   Dust Allergy              1000 non-null   int64
 7   OccuPational Hazards      1000 non-null   int64
 8   Genetic Risk              1000 non-null   int64
 9   chronic Lung Disease      1000 non-null   int64
 10  Balanced Diet             1000 non-null   int64
 11  Obesity                   1000 non-null   int64
 12  Smoking                   1000 non-null   int64
 13  Passive Smoker            1000 non-null   int64
 14  Chest Pain                1000 non-null   int64
 15  Coughing of Blood         1000 non-null   int64
 16  Fatigue                   1000 non-null   int64
 17  Weight Loss               1000 non-null   int64
 18  Shortness of Breath       1000 non-null   int64
 19  Wheezing                  1000 non-null   int64
 20  Swallowing Difficulty     1000 non-null   int64
 21  Clubbing of Finger Nails  1000 non-null   int64
 22  Frequent Cold             1000 non-null   int64
 23  Dry Cough                 1000 non-null   int64
 24  Snoring                   1000 non-null   int64
 25  Level                     1000 non-null   object
dtypes: int64(24), object(2)
memory usage: 203.2+ KB
```

In [5]:
```
df.drop(["index", "Patient Id"], axis = 1, inplace = True)
```

In [6]:
```
df.describe()
```

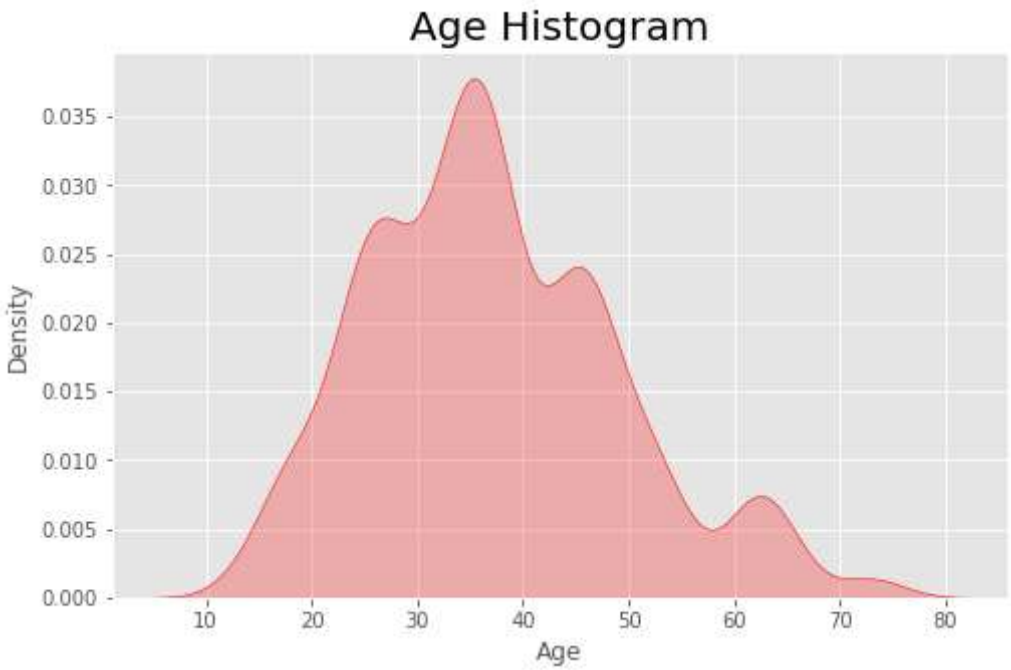
Out[6]:

	Age	Gender	Air Pollution	Alcohol use	Dust Allergy	OccuPational Hazards	Genetic Risk	
count	1000.000000	1000.000000	1000.0000	1000.000000	1000.000000	1000.000000	1000.000000	100
mean	37.174000	1.402000	3.8400	4.563000	5.165000	4.840000	4.580000	
std	12.005493	0.490547	2.0304	2.620477	1.980833	2.107805	2.126999	
min	14.000000	1.000000	1.0000	1.000000	1.000000	1.000000	1.000000	
25%	27.750000	1.000000	2.0000	2.000000	4.000000	3.000000	2.000000	
50%	36.000000	1.000000	3.0000	5.000000	6.000000	5.000000	5.000000	
75%	45.000000	2.000000	6.0000	7.000000	7.000000	7.000000	7.000000	
max	73.000000	2.000000	8.0000	8.000000	8.000000	8.000000	7.000000	

8 rows × 23 columns

While other variables range between 1 and 7, age spans from 14 to 73. This could potentially introduce bias. We will scale the data later.

In [7]:
```
plt.figure(figsize = (8,5))
sns.kdeplot(df.Age, shade = True, color = "r")
plt.title("Age Histogram", fontsize = 20)
plt.show()
print("Histogram's skewness is {} and kurtosis is {}".format(df.Age.skew(), df.Age.kur
```

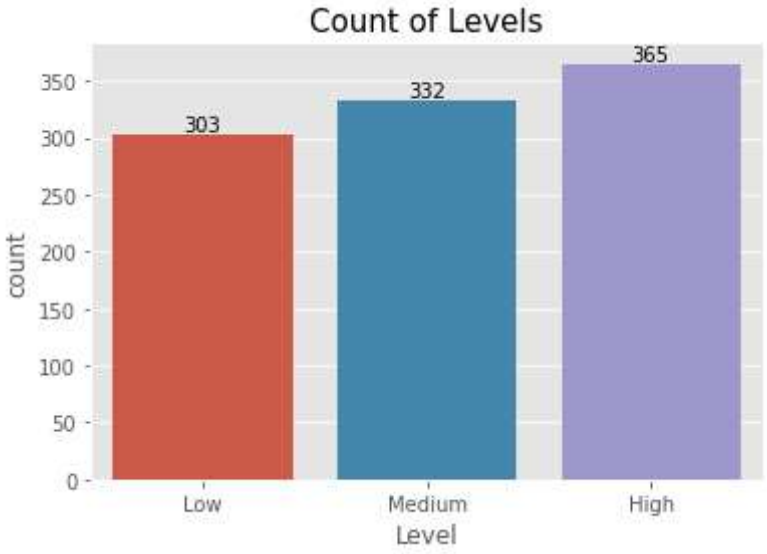



```
Histogram's skewness is 0.5510959291276972 and kurtosis is 0.059540224308944456
```

In [8]:
```
df.Level.unique()
```

Out[8]:
```
array(['Low', 'Medium', 'High'], dtype=object)
```

In [9]:
```
plt.figure(figsize = (6,4))
ax = sns.countplot(df.Level)
for bars in ax.containers:
    ax.bar_label(bars)
plt.title("Count of Levels", fontsize = 15);
```



In [10]:
```
df.Level = df.Level.replace("Low", 0)
df.Level = df.Level.replace("Medium", 1)
df.Level = df.Level.replace("High", 2)

df.Level = df.Level.astype("int64")
```

In [11]:
```
x = df.drop("Level", axis = 1)
y = pd.get_dummies(df["Level"])
```

In [12]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
```

Building Neural Network Model

A neural network is a simplified model of the way the human brain processes information. It operates by simulating a large number of interconnected processing units that resemble abstract versions of neurons, which are organized into layers.

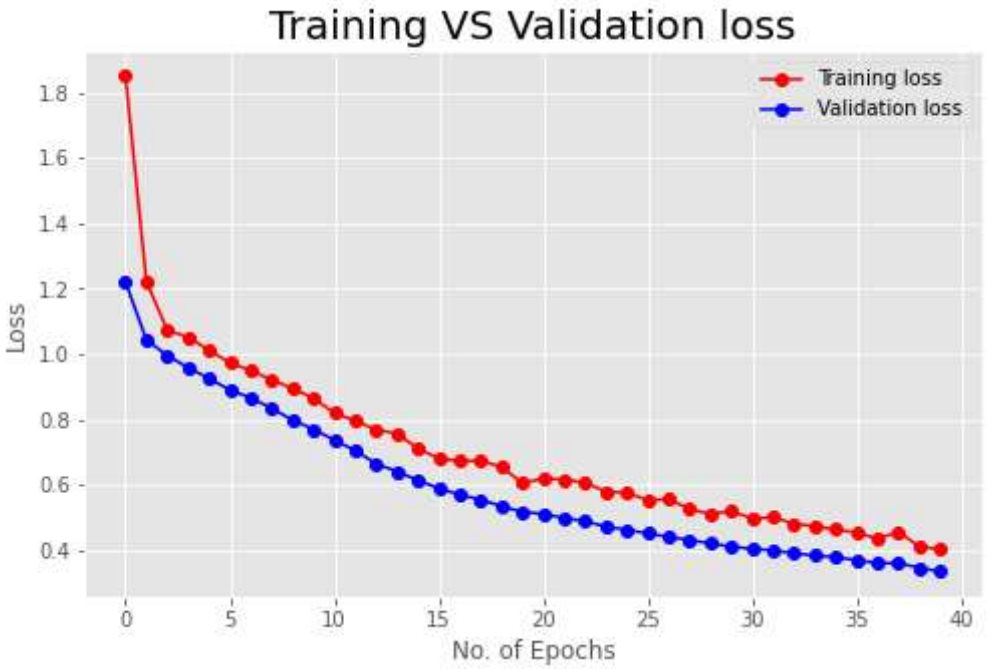
In [13]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation

model = Sequential()

model.add(Dense(8, activation = "relu", input_dim = x.shape[1]))
model.add(Dense(16, activation = "relu"))
model.add(Dropout(0.1))
model.add(Dense(8, activation = "relu"))
model.add(Dense(3, activation = "softmax"))

model.compile(optimizer = "adam", loss = "categorical_crossentropy", metrics = ["accur
history = model.fit(x, y, epochs = 40, validation_split = 0.3)
```

```
Epoch 1/40
22/22 [==============================] - 2s 11ms/step - loss: 1.8504 - accuracy: 0.32
86 - val_loss: 1.2207 - val_accuracy: 0.3200
Epoch 2/40
22/22 [==============================] - 0s 3ms/step - loss: 1.2215 - accuracy: 0.358
6 - val_loss: 1.0437 - val_accuracy: 0.3467
Epoch 3/40
22/22 [==============================] - 0s 3ms/step - loss: 1.0738 - accuracy: 0.414
3 - val_loss: 0.9959 - val_accuracy: 0.4467
Epoch 4/40
22/22 [==============================] - 0s 3ms/step - loss: 1.0514 - accuracy: 0.431
4 - val_loss: 0.9585 - val_accuracy: 0.5200
Epoch 5/40
22/22 [==============================] - 0s 3ms/step - loss: 1.0121 - accuracy: 0.464
3 - val_loss: 0.9256 - val_accuracy: 0.5567
Epoch 6/40
22/22 [==============================] - 0s 3ms/step - loss: 0.9731 - accuracy: 0.487
1 - val_loss: 0.8899 - val_accuracy: 0.5533
Epoch 7/40
22/22 [==============================] - 0s 3ms/step - loss: 0.9497 - accuracy: 0.522
9 - val_loss: 0.8661 - val_accuracy: 0.6500
Epoch 8/40
22/22 [==============================] - 0s 3ms/step - loss: 0.9209 - accuracy: 0.594
3 - val_loss: 0.8344 - val_accuracy: 0.6733
Epoch 9/40
22/22 [==============================] - 0s 3ms/step - loss: 0.8948 - accuracy: 0.575
7 - val_loss: 0.7981 - val_accuracy: 0.6800
Epoch 10/40
22/22 [==============================] - 0s 3ms/step - loss: 0.8651 - accuracy: 0.591
4 - val_loss: 0.7700 - val_accuracy: 0.6800
Epoch 11/40
22/22 [==============================] - 0s 3ms/step - loss: 0.8198 - accuracy: 0.640
0 - val_loss: 0.7369 - val_accuracy: 0.6967
Epoch 12/40
22/22 [==============================] - 0s 3ms/step - loss: 0.7958 - accuracy: 0.635
7 - val_loss: 0.7050 - val_accuracy: 0.6967
Epoch 13/40
22/22 [==============================] - 0s 4ms/step - loss: 0.7691 - accuracy: 0.634
3 - val_loss: 0.6642 - val_accuracy: 0.7033
Epoch 14/40
22/22 [==============================] - 0s 3ms/step - loss: 0.7565 - accuracy: 0.641
4 - val_loss: 0.6410 - val_accuracy: 0.7167
Epoch 15/40
22/22 [==============================] - 0s 3ms/step - loss: 0.7114 - accuracy: 0.662
9 - val_loss: 0.6139 - val_accuracy: 0.7167
Epoch 16/40
22/22 [==============================] - 0s 3ms/step - loss: 0.6808 - accuracy: 0.680
0 - val_loss: 0.5894 - val_accuracy: 0.7300
Epoch 17/40
22/22 [==============================] - 0s 3ms/step - loss: 0.6730 - accuracy: 0.681
4 - val_loss: 0.5699 - val_accuracy: 0.7333
Epoch 18/40
22/22 [==============================] - 0s 3ms/step - loss: 0.6739 - accuracy: 0.684
3 - val_loss: 0.5550 - val_accuracy: 0.7400
Epoch 19/40
22/22 [==============================] - 0s 3ms/step - loss: 0.6549 - accuracy: 0.680
0 - val_loss: 0.5348 - val_accuracy: 0.7533
Epoch 20/40
22/22 [==============================] - 0s 3ms/step - loss: 0.6050 - accuracy: 0.731
4 - val_loss: 0.5171 - val_accuracy: 0.7800
```

```
Epoch 21/40
22/22 [==============================] - 0s 3ms/step - loss: 0.6200 - accuracy: 0.724
3 - val_loss: 0.5116 - val_accuracy: 0.7533
Epoch 22/40
22/22 [==============================] - 0s 3ms/step - loss: 0.6166 - accuracy: 0.721
4 - val_loss: 0.4988 - val_accuracy: 0.8000
Epoch 23/40
22/22 [==============================] - 0s 3ms/step - loss: 0.6067 - accuracy: 0.711
4 - val_loss: 0.4887 - val_accuracy: 0.7933
Epoch 24/40
22/22 [==============================] - 0s 3ms/step - loss: 0.5800 - accuracy: 0.727
1 - val_loss: 0.4721 - val_accuracy: 0.7933
Epoch 25/40
22/22 [==============================] - 0s 3ms/step - loss: 0.5758 - accuracy: 0.725
7 - val_loss: 0.4628 - val_accuracy: 0.7933
Epoch 26/40
22/22 [==============================] - 0s 3ms/step - loss: 0.5536 - accuracy: 0.738
6 - val_loss: 0.4515 - val_accuracy: 0.8200
Epoch 27/40
22/22 [==============================] - 0s 3ms/step - loss: 0.5562 - accuracy: 0.735
7 - val_loss: 0.4410 - val_accuracy: 0.8267
Epoch 28/40
22/22 [==============================] - 0s 3ms/step - loss: 0.5270 - accuracy: 0.771
4 - val_loss: 0.4316 - val_accuracy: 0.8200
Epoch 29/40
22/22 [==============================] - 0s 3ms/step - loss: 0.5106 - accuracy: 0.782
9 - val_loss: 0.4208 - val_accuracy: 0.8333
Epoch 30/40
22/22 [==============================] - 0s 3ms/step - loss: 0.5196 - accuracy: 0.757
1 - val_loss: 0.4113 - val_accuracy: 0.8333
Epoch 31/40
22/22 [==============================] - 0s 3ms/step - loss: 0.4972 - accuracy: 0.780
0 - val_loss: 0.4044 - val_accuracy: 0.8400
Epoch 32/40
22/22 [==============================] - 0s 3ms/step - loss: 0.5005 - accuracy: 0.781
4 - val_loss: 0.3997 - val_accuracy: 0.8400
Epoch 33/40
22/22 [==============================] - 0s 3ms/step - loss: 0.4814 - accuracy: 0.795
7 - val_loss: 0.3909 - val_accuracy: 0.8500
Epoch 34/40
22/22 [==============================] - 0s 3ms/step - loss: 0.4734 - accuracy: 0.805
7 - val_loss: 0.3839 - val_accuracy: 0.8600
Epoch 35/40
22/22 [==============================] - 0s 3ms/step - loss: 0.4640 - accuracy: 0.802
9 - val_loss: 0.3792 - val_accuracy: 0.8600
Epoch 36/40
22/22 [==============================] - 0s 3ms/step - loss: 0.4536 - accuracy: 0.801
4 - val_loss: 0.3697 - val_accuracy: 0.8600
Epoch 37/40
22/22 [==============================] - 0s 3ms/step - loss: 0.4357 - accuracy: 0.815
7 - val_loss: 0.3612 - val_accuracy: 0.8700
Epoch 38/40
22/22 [==============================] - 0s 3ms/step - loss: 0.4565 - accuracy: 0.800
0 - val_loss: 0.3604 - val_accuracy: 0.8700
Epoch 39/40
22/22 [==============================] - 0s 3ms/step - loss: 0.4112 - accuracy: 0.830
0 - val_loss: 0.3469 - val_accuracy: 0.8700
Epoch 40/40
22/22 [==============================] - 0s 3ms/step - loss: 0.4041 - accuracy: 0.828
6 - val_loss: 0.3351 - val_accuracy: 0.8700
```

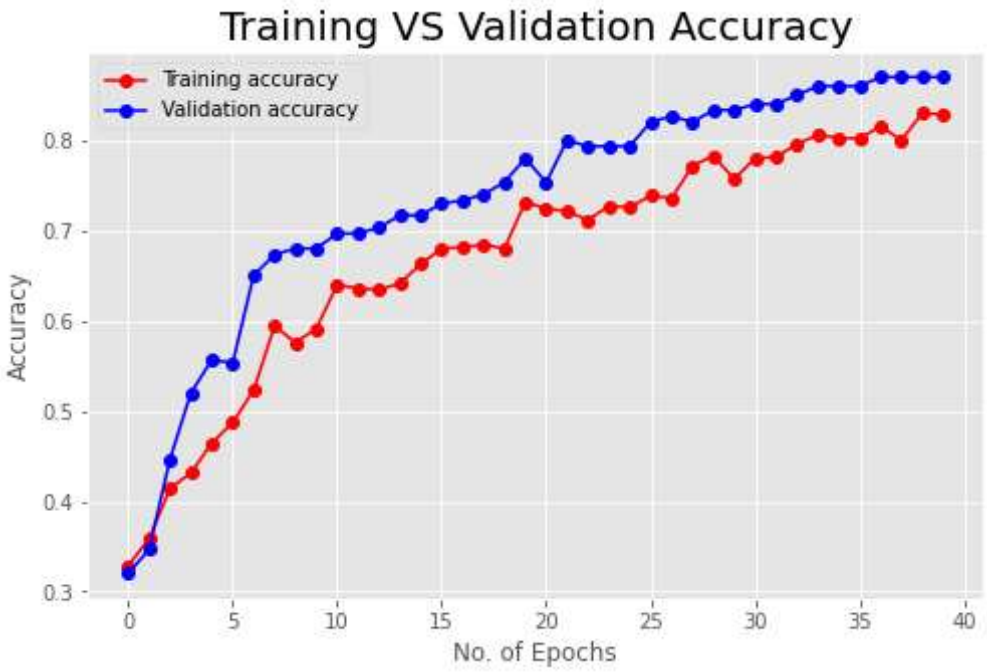
Checking losses in neural network model

In [14]:
```
plt.figure(figsize = (8,5))
plt.plot(history.history["loss"], color = "r", label= "Training loss", marker = "o")
plt.plot(history.history["val_loss"], color = "b", label= "Validation loss", marker =
plt.title("Training VS Validation loss", fontsize = 20)
plt.xlabel("No. of Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



Checking accuracies in neural network model.

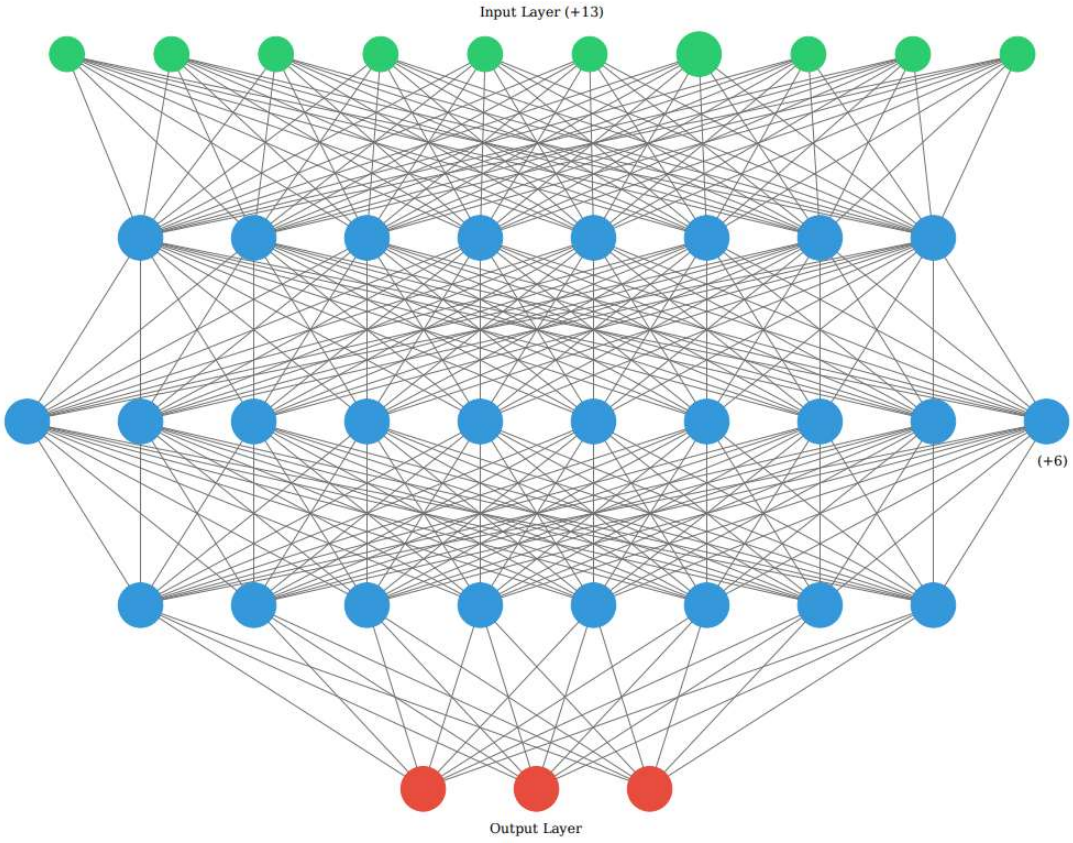
In [15]:
```
plt.figure(figsize = (8,5))
plt.plot(history.history["accuracy"], color = "r", label = "Training accuracy", marker
plt.plot(history.history["val_accuracy"], color = "b", label = "Validation accuracy",
plt.title("Training VS Validation Accuracy", fontsize = 20)
plt.xlabel("No. of Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



Visualizing Neural Network Model

Visualizing the Neural Network Model with ann_visualizer

In [16]:
```
#pip3 install graphviz

#from ann_visualizer.visualize import ann_viz

#ann_viz(model, view = True, filename = "ann.png")
```



This is the visualization of the neural network model.

As we can see, we have

- **23 input layers**
- **32 hidden layers**
- **3 output layers**