Predicting heart disease using an artificial neural network.

In [ ]:
```
# importing libraries
from scipy.io.arff import loadarff
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statistics

from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import make_scorer, accuracy_score
from sklearn.model_selection import GridSearchCV
#from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split as tts
from sklearn.preprocessing import MinMaxScaler
#from sklearn.metrics import confusion_matrix

import tensorflow as tf
from tensorflow import keras
import keras
```

Data Preproccessing & Normalisation

In [ ]:
```
content = loadarff(r'data.arff') #reading/ loading the data
df = pd.DataFrame(content[0])

df.head()
```

Out[ ]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	Class
0	160.0	12.00	5.73	23.11	b'1'	49.0	25.30	97.20	52.0	b'2'
1	144.0	0.01	4.41	28.61	b'2'	55.0	28.87	2.06	63.0	b'2'
2	118.0	0.08	3.48	32.28	b'1'	52.0	29.14	3.81	46.0	b'1'
3	170.0	7.50	6.41	38.03	b'1'	51.0	31.99	24.26	58.0	b'2'
4	134.0	13.60	3.50	27.78	b'1'	60.0	25.99	57.34	49.0	b'2'

In [ ]:
```
attr = ['V1: sbp', 'V2: tobacco', 'V3: ldl', 'V4: adiposity', 'V5: famhist', 'V6: t
df.columns = ['V1: sbp', 'V2: tobacco', 'V3: ldl', 'V4: adiposity', 'V5: famhist',

attr #this is a list containing the column names which will be used later for norma
```

Out[ ]:
```
['V1: sbp',
 'V2: tobacco',
 'V3: ldl',
 'V4: adiposity',
 'V5: famhist',
 'V6: type',
 'V7: obesity',
 'V8: alcohol',
 'V9: age']
```

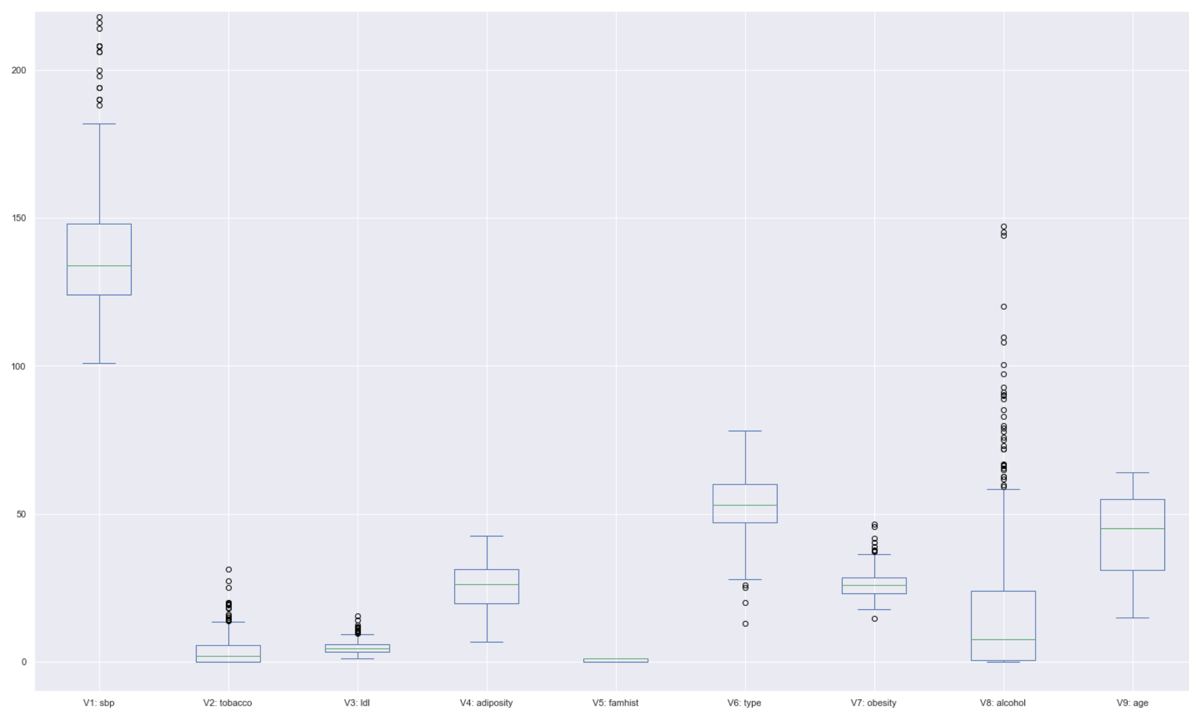
In [ ]:
```
encoder = LabelEncoder()
scaler = MinMaxScaler()
df['V5: famhist']=encoder.fit_transform(df['V5: famhist'])
df['Class: chd']=encoder.fit_transform(df['Class: chd'])
df.head()
```

Out[ ]:

	V1: sbp	V2: tobacco	V3: ldl	V4: adiposity	V5: famhist	V6: type	V7: obesity	V8: alcohol	V9: age	Class: chd
0	160.0	12.00	5.73	23.11	0	49.0	25.30	97.20	52.0	1
1	144.0	0.01	4.41	28.61	1	55.0	28.87	2.06	63.0	1
2	118.0	0.08	3.48	32.28	0	52.0	29.14	3.81	46.0	0
3	170.0	7.50	6.41	38.03	0	51.0	31.99	24.26	58.0	1
4	134.0	13.60	3.50	27.78	0	60.0	25.99	57.34	49.0	1

In [ ]:
```
df[attr].plot(kind='box',figsize=(25,15),ylim=[-10,220])
```

Out[ ]: <AxesSubplot:>



In [ ]:

```
# time to remove the dots (data outside of the boxplot) to make the prediction more
for col in attr:
    q3, q1 = np.percentile(df[col], [75,25])
    IQR = q3 - q1
    maxbP = q3 + 1.5 * IQR
    minbP = q1 - 1.5 * IQR
    df = df[(df[col] > minbP) & (df[col] < maxbP)]

df.head()
```

Out[ ]:

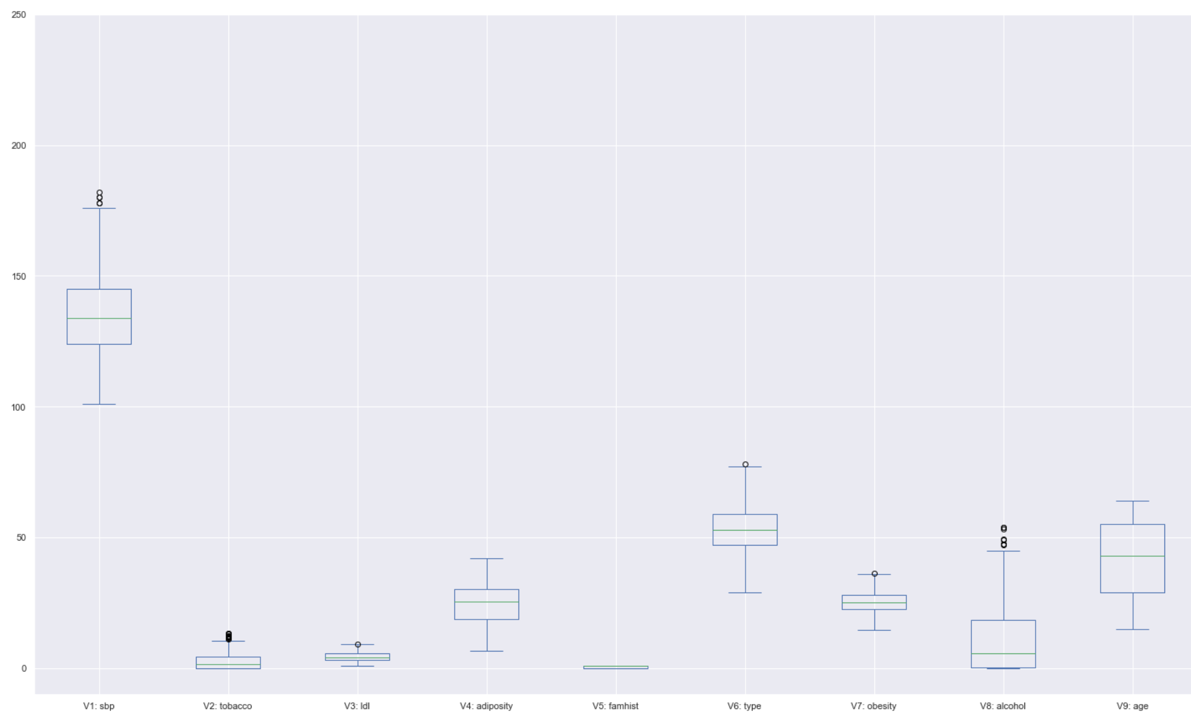
	V1: sbp	V2: tobacco	V3: ldl	V4: adiposity	V5: famhist	V6: type	V7: obesity	V8: alcohol	V9: age	Class: chd
1	144.0	0.01	4.41	28.61	1	55.0	28.87	2.06	63.0	1
2	118.0	0.08	3.48	32.28	0	52.0	29.14	3.81	46.0	0
3	170.0	7.50	6.41	38.03	0	51.0	31.99	24.26	58.0	1
5	132.0	6.20	6.47	36.21	0	62.0	30.77	14.14	45.0	0
6	142.0	4.05	3.38	16.20	1	59.0	20.81	2.62	38.0	0

In [ ]:

```
df[attr].plot(kind='box',figsize=(25,15),ylim=[-10,250])
```

Out[ ]:

```
<AxesSubplot:>
```



In [ ]:

```
df[attr] = scaler.fit_transform(df[attr])
df.head()
```

Out[ ]:

	V1: sbp	V2: tobacco	V3: ldl	V4: adiposity	V5: famhist	V6: type	V7: obesity	V8: alcohol	V9: age	Class: chd
1	0.530864	0.000741	0.417783	0.617273	1.0	0.530612	0.651195	0.038148	0.979592	1
2	0.209877	0.005926	0.304507	0.720858	0.0	0.469388	0.663603	0.070556	0.632653	0
3	0.851852	0.555556	0.661389	0.883150	0.0	0.448980	0.794577	0.449259	0.877551	1
5	0.382716	0.459259	0.668697	0.831781	0.0	0.673469	0.738511	0.261852	0.612245	0
6	0.506173	0.300000	0.292326	0.267005	1.0	0.612245	0.280790	0.048519	0.469388	0

In [ ]:

```
df.isnull().any()
```

Out[ ]:

```
V1: sbp          False
V2: tobacco      False
V3: ldl          False
V4: adiposity    False
V5: famhist      False
V6: type         False
V7: obesity      False
V8: alcohol      False
V9: age          False
Class: chd       False
dtype: bool
```

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 377 entries, 1 to 461
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   V1: sbp        377 non-null    float64
 1   V2: tobacco    377 non-null    float64
 2   V3: ldl        377 non-null    float64
 3   V4: adiposity  377 non-null    float64
 4   V5: famhist    377 non-null    float64
 5   V6: type       377 non-null    float64
 6   V7: obesity    377 non-null    float64
 7   V8: alcohol    377 non-null    float64
 8   V9: age        377 non-null    float64
 9   Class: chd     377 non-null    int32  
dtypes: float64(9), int32(1)
memory usage: 30.9 KB
```

In [ ]:

```
df.describe()
```

Out[ ]:

	V1: sbp	V2: tobacco	V3: ldl	V4: adiposity	V5: famhist	V6: type	V7: obesity	alc
count	377.000000	377.000000	377.000000	377.000000	377.000000	377.000000	377.000000	377.00
mean	0.427907	0.207350	0.428949	0.501601	0.588859	0.492448	0.496982	0.21
std	0.206805	0.241956	0.211649	0.218006	0.492695	0.188056	0.173281	0.24
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	0.283951	0.000000	0.272838	0.338132	0.000000	0.367347	0.362592	0.00
50%	0.407407	0.111111	0.389769	0.526390	1.000000	0.489796	0.485294	0.10
75%	0.543210	0.330370	0.560292	0.662997	1.000000	0.612245	0.614430	0.34
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00

In [ ]:
```
df['V5: famhist'].value_counts()
```

Out[ ]:
```
1.0    222
0.0    155
Name: V5: famhist, dtype: int64
```

In [ ]:
```
df['Class: chd'].value_counts()
```

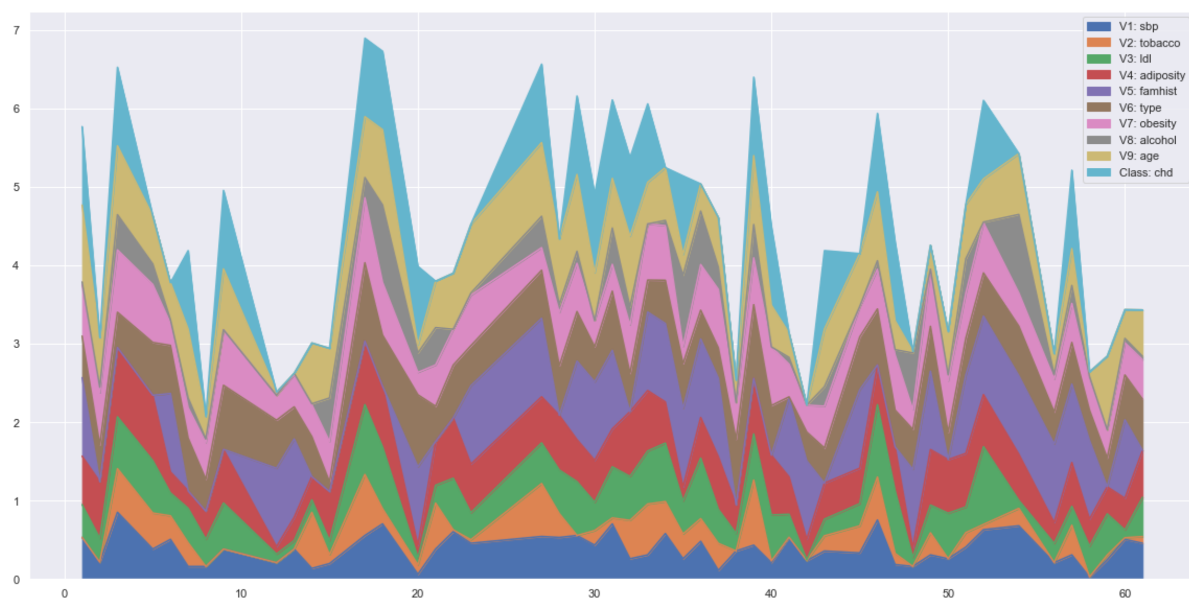
Out[ ]:
```
0    260
1    117
Name: Class: chd, dtype: int64
```

In [ ]:
```
scale = MinMaxScaler(feature_range =(0,100))
```

The variation of values across the DataFrame for first 50 values

In [ ]:
```
df.head(50).plot(kind='area',figsize=(20,10))
```

Out[ ]: <AxesSubplot:>



Distribution of Obesity according to the age

In [ ]:
```
df.plot(x='V9: age',y='V7: obesity',kind='scatter',figsize =(10,5))
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoi
ded as value-mapping will have precedence in case its length matches with *x* & *y
*.  Please use the *color* keyword-argument or provide a 2D array with a single row
if you intend to specify the same RGB or RGBA value for all points.
```

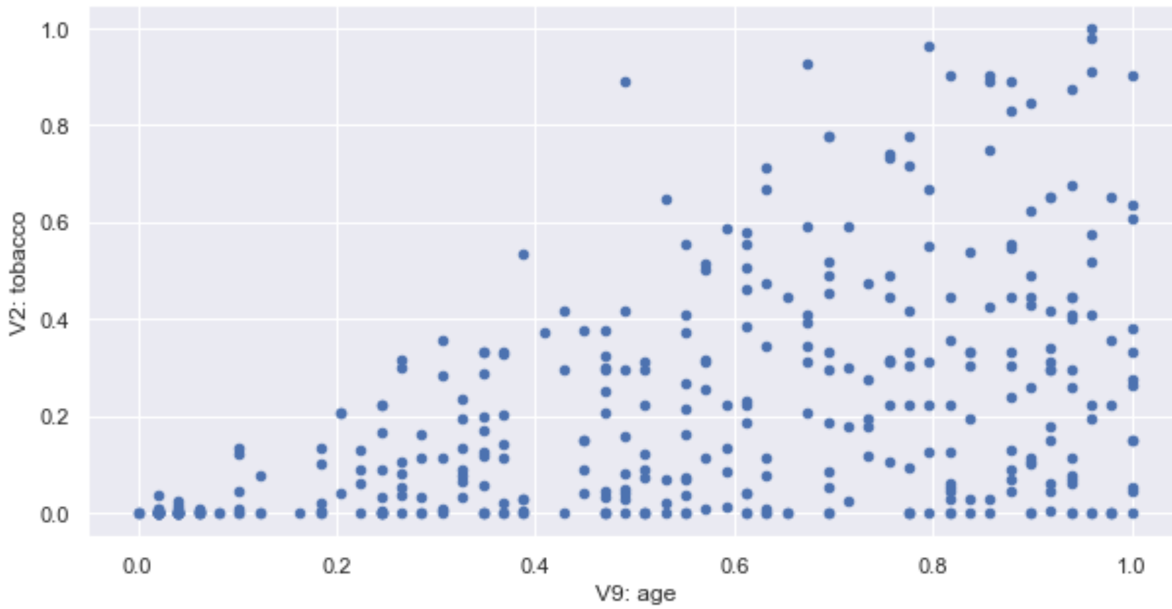
Out[ ]:
```
<AxesSubplot:xlabel='V9: age', ylabel='V7: obesity'>
```

Distribution of Tobacco consumption across age

In [ ]:
```
df.plot(x='V9: age',y='V2: tobacco',kind='scatter',figsize =(10,5))
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoi
ded as value-mapping will have precedence in case its length matches with *x* & *y
*.  Please use the *color* keyword-argument or provide a 2D array with a single row
if you intend to specify the same RGB or RGBA value for all points.
```

Out[ ]:
```
<AxesSubplot:xlabel='V9: age', ylabel='V2: tobacco'>
```

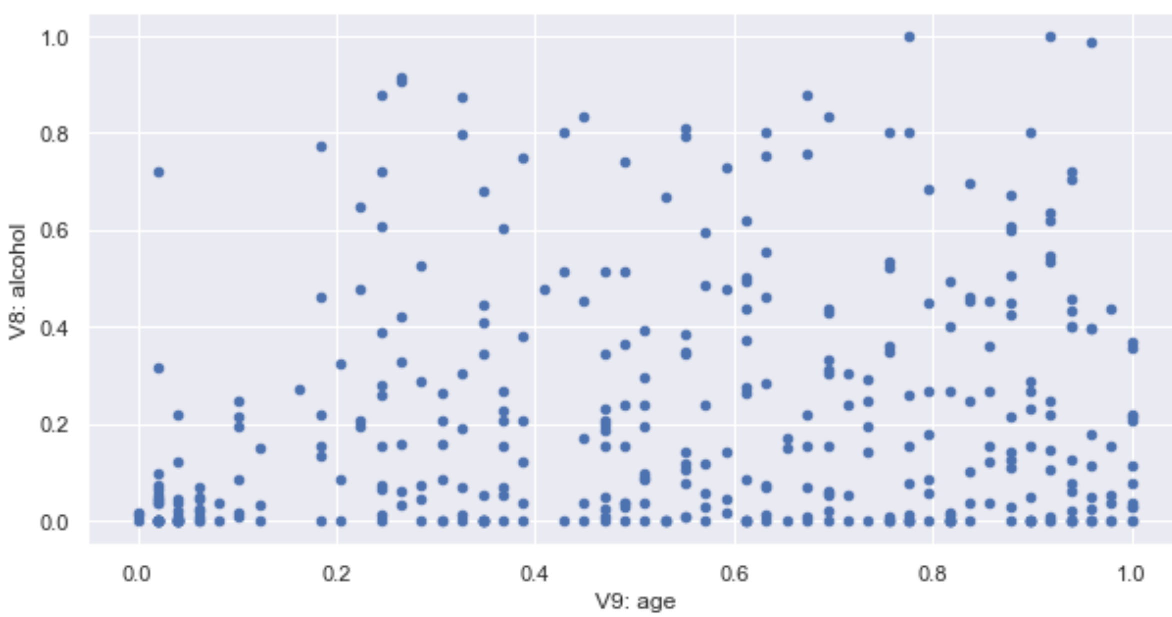


Distribution of Alcohol consumption across age

In [ ]:
```
df.plot(x='V9: age',y='V8: alcohol',kind='scatter',figsize =(10,5))
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoi
ded as value-mapping will have precedence in case its length matches with *x* & *y
*.  Please use the *color* keyword-argument or provide a 2D array with a single row
if you intend to specify the same RGB or RGBA value for all points.
```

Out[ ]:
```
<AxesSubplot:xlabel='V9: age', ylabel='V8: alcohol'>
```

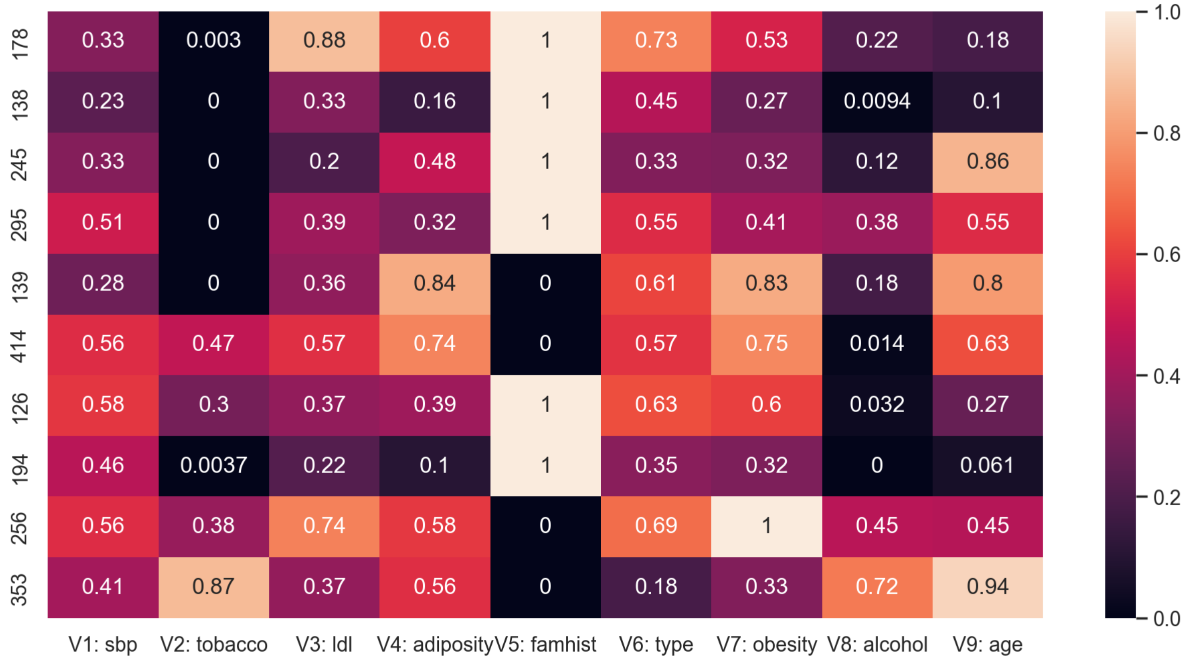


Testing and Training

In [ ]:
```
# splitting the data into test and train having a test size of 20% and 80% train si
Xtrain, Xtest, ytrain, ytest = tts(df[attr], df['Class: chd'], test_size=0.3, rando
```

In [ ]:
```
sns.set()
plt.figure(figsize=(12, 6), dpi=200)
sns.heatmap(data = Xtrain.head(10),annot = True)
```

Out[ ]: <AxesSubplot:>



In [ ]:
```
print('Training Features shape', Xtrain.shape)
print('Training Labels shape', ytrain.shape)
print('Testing Features shape', Xtest.shape)
print('Testing Labels shape', ytest.shape)
```

```
Training Features shape (263, 9)
Training Labels shape (263,)
Testing Features shape (114, 9)
Testing Labels shape (114,)
```

In [ ]:
```
rand = statistics.mode(ytrain)
pred = [rand] *len(ytest)

BaseAcc = accuracy_score(ytest , pred)
print(f'Base Line Accuracy: {round(BaseAcc*100, 0)}%' )
```

```
Base Line Accuracy: 65.0%
```

In [ ]:
```
mlp = MLPClassifier(max_iter=5000)
para = {'solver': ['lbfgs'],
             'alpha':[1e-4],
             'hidden_layer_sizes':(9,20,20,2),   # 9 input, 14-14 neuron in 2 layer
             'random_state': [1234],
             'max_iter':[10000],
             'early_stopping':[False]}

score = make_scorer(accuracy_score)

# Run grid search
grid = GridSearchCV(mlp, para, scoring=score,cv = 5)
grid = grid.fit(Xtrain, ytrain)

# Pick the best combination of parameters
ann_clf = grid.best_estimator_
mlp.fit(Xtrain, ytrain)
```

Out[ ]:
```
MLPClassifier(max_iter=5000)
```

In [ ]:
```
mlp_pred = mlp.predict(Xtest)
mlp_pred
```

Out[ ]:
```
array([0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0])
```

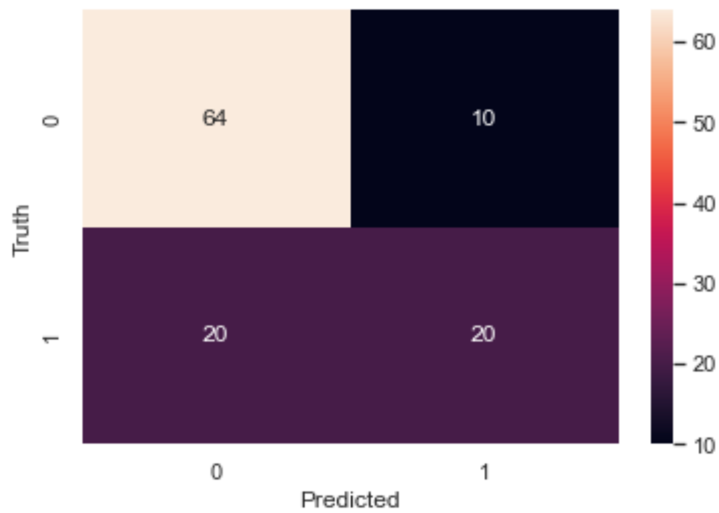
In [ ]:
```
mlp_acc = accuracy_score(ytest, mlp_pred)
print(f'the ANN accuracy is: {round(mlp_acc * 100, 0)}%' )
```

```
the ANN accuracy is: 74.0%
```

In [ ]:
```
cm = tf.math.confusion_matrix(labels=ytest, predictions=mlp_pred)

sns.heatmap(cm,annot =True,fmt = 'd')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[ ]:
```
Text(30.5, 0.5, 'Truth')
```



In [ ]:
```
trainMod = keras.Sequential([
    keras.layers.Dense(9, input_shape=(9,),activation='relu'),
    keras.layers.Dense(1,activation='sigmoid'),
     ])
trainMod.compile(optimizer='adam',
              loss = 'binary_crossentropy',
              metrics=['accuracy'])

history = trainMod.fit(Xtrain, ytrain, epochs=200)
history
```

```
Epoch 1/200
9/9 [==============================] - 0s 1ms/step - loss: 0.6477 - accuracy: 0.676
8
Epoch 2/200
9/9 [==============================] - 0s 323us/step - loss: 0.6368 - accuracy: 0.6
996
Epoch 3/200
9/9 [==============================] - 0s 998us/step - loss: 0.6278 - accuracy: 0.7
110
Epoch 4/200
9/9 [==============================] - 0s 2ms/step - loss: 0.6188 - accuracy: 0.699
6
Epoch 5/200
9/9 [==============================] - 0s 1ms/step - loss: 0.6112 - accuracy: 0.707
2
Epoch 6/200
9/9 [==============================] - 0s 1ms/step - loss: 0.6036 - accuracy: 0.722
4
Epoch 7/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5974 - accuracy: 0.726
2
Epoch 8/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5928 - accuracy: 0.726
2
Epoch 9/200
9/9 [==============================] - 0s 2ms/step - loss: 0.5868 - accuracy: 0.718
6
Epoch 10/200
9/9 [==============================] - 0s 818us/step - loss: 0.5824 - accuracy: 0.7
186
Epoch 11/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5784 - accuracy: 0.718
6
Epoch 12/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5750 - accuracy: 0.718
6
Epoch 13/200
9/9 [==============================] - 0s 2ms/step - loss: 0.5720 - accuracy: 0.722
4
Epoch 14/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5687 - accuracy: 0.718
6
Epoch 15/200
9/9 [==============================] - 0s 2ms/step - loss: 0.5664 - accuracy: 0.722
4
Epoch 16/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5642 - accuracy: 0.722
4
Epoch 17/200
9/9 [==============================] - 0s 2ms/step - loss: 0.5621 - accuracy: 0.718
6
Epoch 18/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5600 - accuracy: 0.722
4
Epoch 19/200
9/9 [==============================] - 0s 989us/step - loss: 0.5576 - accuracy: 0.7
262
Epoch 20/200
9/9 [==============================] - 0s 999us/step - loss: 0.5557 - accuracy: 0.7
```

```
300
Epoch 21/200
9/9 [==============================] - 0s 988us/step - loss: 0.5536 - accuracy: 0.7
262
Epoch 22/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5513 - accuracy: 0.733
8
Epoch 23/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5492 - accuracy: 0.733
8
Epoch 24/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5478 - accuracy: 0.726
2
Epoch 25/200
9/9 [==============================] - 0s 1000us/step - loss: 0.5455 - accuracy: 0.
7148
Epoch 26/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5440 - accuracy: 0.714
8
Epoch 27/200
9/9 [==============================] - 0s 814us/step - loss: 0.5419 - accuracy: 0.7
186
Epoch 28/200
9/9 [==============================] - 0s 991us/step - loss: 0.5402 - accuracy: 0.7
148
Epoch 29/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5388 - accuracy: 0.711
0
Epoch 30/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5375 - accuracy: 0.711
0
Epoch 31/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5358 - accuracy: 0.711
0
Epoch 32/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5342 - accuracy: 0.711
0
Epoch 33/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5330 - accuracy: 0.711
0
Epoch 34/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5324 - accuracy: 0.718
6
Epoch 35/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5303 - accuracy: 0.714
8
Epoch 36/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5295 - accuracy: 0.707
2
Epoch 37/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5282 - accuracy: 0.707
2
Epoch 38/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5272 - accuracy: 0.714
8
Epoch 39/200
9/9 [==============================] - 0s 2ms/step - loss: 0.5262 - accuracy: 0.714
8
Epoch 40/200
```

```
9/9 [==============================] - 0s 1ms/step - loss: 0.5258 - accuracy: 0.711
0
Epoch 41/200
9/9 [==============================] - 0s 189us/step - loss: 0.5249 - accuracy: 0.7
148
Epoch 42/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5244 - accuracy: 0.714
8
Epoch 43/200
9/9 [==============================] - 0s 2ms/step - loss: 0.5234 - accuracy: 0.714
8
Epoch 44/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5223 - accuracy: 0.718
6
Epoch 45/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5216 - accuracy: 0.718
6
Epoch 46/200
9/9 [==============================] - 0s 557us/step - loss: 0.5217 - accuracy: 0.7
186
Epoch 47/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5205 - accuracy: 0.722
4
Epoch 48/200
9/9 [==============================] - 0s 2ms/step - loss: 0.5198 - accuracy: 0.733
8
Epoch 49/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5191 - accuracy: 0.733
8
Epoch 50/200
9/9 [==============================] - 0s 998us/step - loss: 0.5183 - accuracy: 0.7
300
Epoch 51/200
9/9 [==============================] - 0s 1000us/step - loss: 0.5178 - accuracy: 0.
7300
Epoch 52/200
9/9 [==============================] - 0s 698us/step - loss: 0.5174 - accuracy: 0.7
300
Epoch 53/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5168 - accuracy: 0.733
8
Epoch 54/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5161 - accuracy: 0.726
2
Epoch 55/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5155 - accuracy: 0.737
6
Epoch 56/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5156 - accuracy: 0.737
6
Epoch 57/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5152 - accuracy: 0.737
6
Epoch 58/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5147 - accuracy: 0.733
8
Epoch 59/200
9/9 [==============================] - 0s 6ms/step - loss: 0.5142 - accuracy: 0.737
6
```

```
Epoch 60/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5135 - accuracy: 0.737
6
Epoch 61/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5130 - accuracy: 0.741
4
Epoch 62/200
9/9 [==============================] - 0s 997us/step - loss: 0.5124 - accuracy: 0.7
414
Epoch 63/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5120 - accuracy: 0.741
4
Epoch 64/200
9/9 [==============================] - 0s 2ms/step - loss: 0.5120 - accuracy: 0.737
6
Epoch 65/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5112 - accuracy: 0.741
4
Epoch 66/200
9/9 [==============================] - 0s 999us/step - loss: 0.5106 - accuracy: 0.7
452
Epoch 67/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5107 - accuracy: 0.741
4
Epoch 68/200
9/9 [==============================] - 0s 2ms/step - loss: 0.5102 - accuracy: 0.741
4
Epoch 69/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5099 - accuracy: 0.745
2
Epoch 70/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5092 - accuracy: 0.741
4
Epoch 71/200
9/9 [==============================] - 0s 2ms/step - loss: 0.5087 - accuracy: 0.752
9
Epoch 72/200
9/9 [==============================] - 0s 1000us/step - loss: 0.5086 - accuracy: 0.
7529
Epoch 73/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5079 - accuracy: 0.752
9
Epoch 74/200
9/9 [==============================] - 0s 445us/step - loss: 0.5087 - accuracy: 0.7
452
Epoch 75/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5077 - accuracy: 0.749
0
Epoch 76/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5072 - accuracy: 0.749
0
Epoch 77/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5071 - accuracy: 0.745
2
Epoch 78/200
9/9 [==============================] - 0s 999us/step - loss: 0.5069 - accuracy: 0.7
567
Epoch 79/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5062 - accuracy: 0.749
```

```
0
Epoch 80/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5058 - accuracy: 0.749
0
Epoch 81/200
9/9 [==============================] - 0s 1000us/step - loss: 0.5058 - accuracy: 0.
7490
Epoch 82/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5053 - accuracy: 0.752
9
Epoch 83/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5054 - accuracy: 0.745
2
Epoch 84/200
9/9 [==============================] - 0s 1000us/step - loss: 0.5052 - accuracy: 0.
7529
Epoch 85/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5047 - accuracy: 0.749
0
Epoch 86/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5039 - accuracy: 0.752
9
Epoch 87/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5035 - accuracy: 0.749
0
Epoch 88/200
9/9 [==============================] - 0s 999us/step - loss: 0.5035 - accuracy: 0.7
529
Epoch 89/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5030 - accuracy: 0.752
9
Epoch 90/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5029 - accuracy: 0.752
9
Epoch 91/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5028 - accuracy: 0.752
9
Epoch 92/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5029 - accuracy: 0.752
9
Epoch 93/200
9/9 [==============================] - 0s 1000us/step - loss: 0.5021 - accuracy: 0.
7490
Epoch 94/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5020 - accuracy: 0.749
0
Epoch 95/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5014 - accuracy: 0.752
9
Epoch 96/200
9/9 [==============================] - 0s 1000us/step - loss: 0.5012 - accuracy: 0.
7490
Epoch 97/200
9/9 [==============================] - 0s 2ms/step - loss: 0.5007 - accuracy: 0.752
9
Epoch 98/200
9/9 [==============================] - 0s 1ms/step - loss: 0.5007 - accuracy: 0.752
9
Epoch 99/200
```

```
9/9 [==============================] - 0s 1ms/step - loss: 0.5001 - accuracy: 0.752
9
Epoch 100/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4999 - accuracy: 0.752
9
Epoch 101/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4998 - accuracy: 0.752
9
Epoch 102/200
9/9 [==============================] - 0s 1000us/step - loss: 0.4994 - accuracy: 0.
7529
Epoch 103/200
9/9 [==============================] - 0s 341us/step - loss: 0.4990 - accuracy: 0.7
567
Epoch 104/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4992 - accuracy: 0.756
7
Epoch 105/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4984 - accuracy: 0.756
7
Epoch 106/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4984 - accuracy: 0.749
0
Epoch 107/200
9/9 [==============================] - 0s 996us/step - loss: 0.4981 - accuracy: 0.7
529
Epoch 108/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4979 - accuracy: 0.752
9
Epoch 109/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4976 - accuracy: 0.749
0
Epoch 110/200
9/9 [==============================] - 0s 2ms/step - loss: 0.4974 - accuracy: 0.756
7
Epoch 111/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4976 - accuracy: 0.756
7
Epoch 112/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4968 - accuracy: 0.756
7
Epoch 113/200
9/9 [==============================] - 0s 1000us/step - loss: 0.4965 - accuracy: 0.
7529
Epoch 114/200
9/9 [==============================] - 0s 1000us/step - loss: 0.4965 - accuracy: 0.
7490
Epoch 115/200
9/9 [==============================] - 0s 599us/step - loss: 0.4963 - accuracy: 0.7
529
Epoch 116/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4960 - accuracy: 0.752
9
Epoch 117/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4959 - accuracy: 0.752
9
Epoch 118/200
9/9 [==============================] - 0s 999us/step - loss: 0.4956 - accuracy: 0.7
529
```

```
Epoch 119/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4953 - accuracy: 0.756
7
Epoch 120/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4952 - accuracy: 0.756
7
Epoch 121/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4951 - accuracy: 0.756
7
Epoch 122/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4952 - accuracy: 0.756
7
Epoch 123/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4948 - accuracy: 0.756
7
Epoch 124/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4946 - accuracy: 0.756
7
Epoch 125/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4942 - accuracy: 0.756
7
Epoch 126/200
9/9 [==============================] - 0s 999us/step - loss: 0.4940 - accuracy: 0.7
567
Epoch 127/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4936 - accuracy: 0.756
7
Epoch 128/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4934 - accuracy: 0.756
7
Epoch 129/200
9/9 [==============================] - 0s 2ms/step - loss: 0.4931 - accuracy: 0.756
7
Epoch 130/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4934 - accuracy: 0.760
5
Epoch 131/200
9/9 [==============================] - 0s 2ms/step - loss: 0.4930 - accuracy: 0.756
7
Epoch 132/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4921 - accuracy: 0.756
7
Epoch 133/200
9/9 [==============================] - 0s 2ms/step - loss: 0.4925 - accuracy: 0.752
9
Epoch 134/200
9/9 [==============================] - 0s 1000us/step - loss: 0.4926 - accuracy: 0.
7490
Epoch 135/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4925 - accuracy: 0.752
9
Epoch 136/200
9/9 [==============================] - 0s 1000us/step - loss: 0.4918 - accuracy: 0.
7567
Epoch 137/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4916 - accuracy: 0.756
7
Epoch 138/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4914 - accuracy: 0.756
```

```
7
Epoch 139/200
9/9 [==============================] - 0s 2ms/step - loss: 0.4915 - accuracy: 0.752
9
Epoch 140/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4916 - accuracy: 0.760
5
Epoch 141/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4908 - accuracy: 0.760
5
Epoch 142/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4908 - accuracy: 0.760
5
Epoch 143/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4905 - accuracy: 0.760
5
Epoch 144/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4907 - accuracy: 0.760
5
Epoch 145/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4901 - accuracy: 0.760
5
Epoch 146/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4900 - accuracy: 0.760
5
Epoch 147/200
9/9 [==============================] - 0s 751us/step - loss: 0.4900 - accuracy: 0.7
605
Epoch 148/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4894 - accuracy: 0.760
5
Epoch 149/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4893 - accuracy: 0.760
5
Epoch 150/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4893 - accuracy: 0.760
5
Epoch 151/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4890 - accuracy: 0.760
5
Epoch 152/200
9/9 [==============================] - 0s 2ms/step - loss: 0.4889 - accuracy: 0.760
5
Epoch 153/200
9/9 [==============================] - 0s 2ms/step - loss: 0.4886 - accuracy: 0.760
5
Epoch 154/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4884 - accuracy: 0.756
7
Epoch 155/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4883 - accuracy: 0.752
9
Epoch 156/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4879 - accuracy: 0.756
7
Epoch 157/200
9/9 [==============================] - 0s 991us/step - loss: 0.4878 - accuracy: 0.7
567
Epoch 158/200
```

```
9/9 [==============================] - 0s 996us/step - loss: 0.4881 - accuracy: 0.7
605
Epoch 159/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4877 - accuracy: 0.760
5
Epoch 160/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4877 - accuracy: 0.760
5
Epoch 161/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4875 - accuracy: 0.760
5
Epoch 162/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4872 - accuracy: 0.760
5
Epoch 163/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4872 - accuracy: 0.760
5
Epoch 164/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4870 - accuracy: 0.760
5
Epoch 165/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4867 - accuracy: 0.756
7
Epoch 166/200
9/9 [==============================] - 0s 998us/step - loss: 0.4868 - accuracy: 0.7
529
Epoch 167/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4867 - accuracy: 0.756
7
Epoch 168/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4863 - accuracy: 0.760
5
Epoch 169/200
9/9 [==============================] - 0s 998us/step - loss: 0.4856 - accuracy: 0.7
567
Epoch 170/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4861 - accuracy: 0.752
9
Epoch 171/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4862 - accuracy: 0.749
0
Epoch 172/200
9/9 [==============================] - 0s 0s/step - loss: 0.4862 - accuracy: 0.7490
Epoch 173/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4861 - accuracy: 0.749
0
Epoch 174/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4875 - accuracy: 0.752
9
Epoch 175/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4861 - accuracy: 0.749
0
Epoch 176/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4857 - accuracy: 0.752
9
Epoch 177/200
9/9 [==============================] - 0s 0s/step - loss: 0.4856 - accuracy: 0.7605
Epoch 178/200
9/9 [==============================] - 0s 246us/step - loss: 0.4851 - accuracy: 0.7
```

```
605
Epoch 179/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4848 - accuracy: 0.760
5
Epoch 180/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4846 - accuracy: 0.760
5
Epoch 181/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4848 - accuracy: 0.760
5
Epoch 182/200
9/9 [==============================] - 0s 981us/step - loss: 0.4844 - accuracy: 0.7
605
Epoch 183/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4843 - accuracy: 0.752
9
Epoch 184/200
9/9 [==============================] - 0s 1000us/step - loss: 0.4840 - accuracy: 0.
7567
Epoch 185/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4838 - accuracy: 0.756
7
Epoch 186/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4837 - accuracy: 0.756
7
Epoch 187/200
9/9 [==============================] - 0s 1000us/step - loss: 0.4834 - accuracy: 0.
7605
Epoch 188/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4840 - accuracy: 0.756
7
Epoch 189/200
9/9 [==============================] - 0s 999us/step - loss: 0.4832 - accuracy: 0.7
567
Epoch 190/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4830 - accuracy: 0.756
7
Epoch 191/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4829 - accuracy: 0.756
7
Epoch 192/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4827 - accuracy: 0.756
7
Epoch 193/200
9/9 [==============================] - 0s 1000us/step - loss: 0.4827 - accuracy: 0.
7567
Epoch 194/200
9/9 [==============================] - 0s 999us/step - loss: 0.4829 - accuracy: 0.7
605
Epoch 195/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4822 - accuracy: 0.760
5
Epoch 196/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4821 - accuracy: 0.760
5
Epoch 197/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4822 - accuracy: 0.760
5
Epoch 198/200
```

```
9/9 [==============================] - 0s 1ms/step - loss: 0.4820 - accuracy: 0.756
7
Epoch 199/200
9/9 [==============================] - 0s 1ms/step - loss: 0.4818 - accuracy: 0.756
7
Epoch 200/200
9/9 [==============================] - 0s 1000us/step - loss: 0.4815 - accuracy: 0.
7567
```

Out[ ]:

```
<keras.callbacks.History at 0x26501169a80>
```

In [ ]:

```python
loss = history.history['loss']
accuracy = history.history['accuracy']
epochs = history.epoch
```

In [ ]:

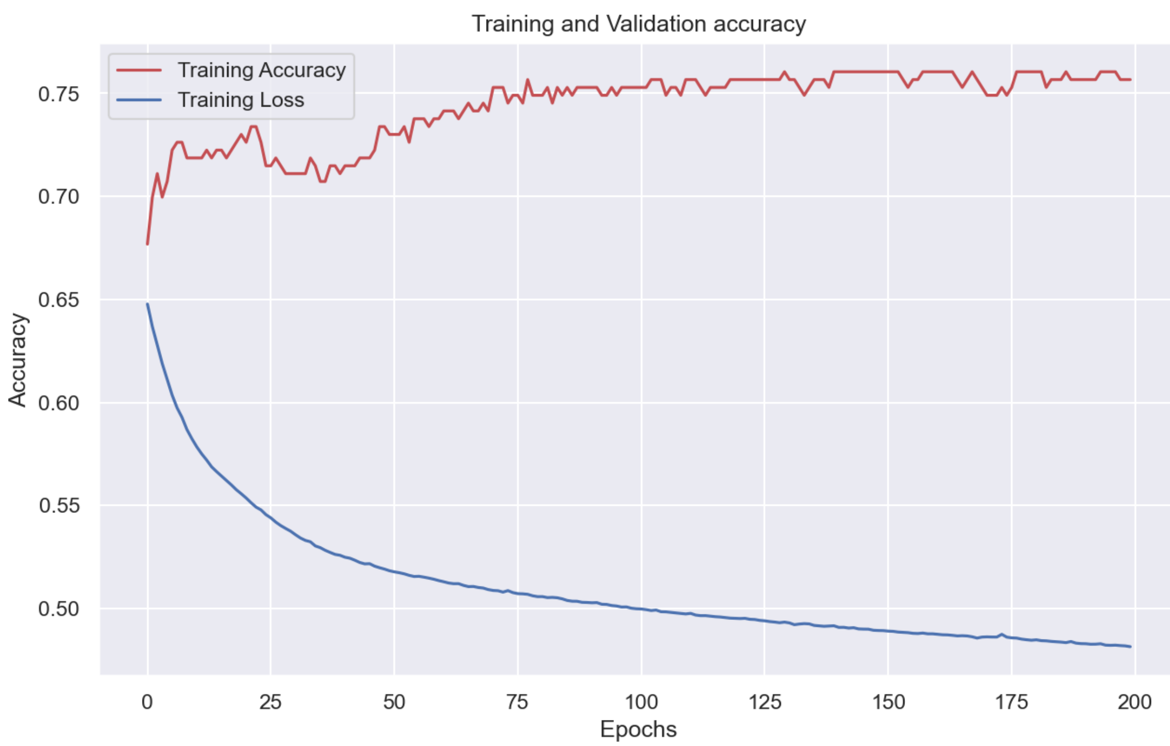
```python
mlp_acc = accuracy_score(ytest, mlp_pred)

print(f'MLP ANN accuracy is: {round(mlp_acc * 100, 0)}%' )
```

```
MLP ANN accuracy is: 74.0%
```

In [ ]:

```python
plt.figure(figsize=(10,6), dpi=150)
plt.plot(history.epoch, history.history['accuracy'], 'r', label='Training Accuracy'
plt.plot(history.epoch, history.history['loss'], 'b', label='Training Loss')
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



In [ ]:
```
cm = tf.math.confusion_matrix(labels=ytest, predictions=mlp_pred)

sns.heatmap(cm,annot =True,fmt = 'd')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[ ]:
```
Text(30.5, 0.5, 'Truth')
```