Artificial Neural network

In [1]:
```
#!pip install tensorflow
```

In [2]:
```
# Importing necessary libraries
import pandas as pd

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split, cross_val_score

# Create your first MLP in Keras
from keras.models import Sequential
from keras.layers import Dense
import numpy as np
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
forest_data = pd.read_csv('forestfires.csv')
forest_data
```

Out[3]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	...	monthfeb	monthjan	m
0	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	...	0	0	
1	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	...	0	0	
2	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	...	0	0	
3	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	...	0	0	
4	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
512	aug	sun	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	...	0	0	
513	aug	sun	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	...	0	0	
514	aug	sun	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	...	0	0	
515	aug	sat	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	...	0	0	
516	nov	tue	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	...	0	0	

In [4]:
```
forest_data.isna().sum()
```

Out[4]:
```
month         0
day           0
FFMC          0
DMC           0
DC            0
ISI           0
temp          0
RH            0
wind          0
rain          0
area          0
dayfri        0
daymon        0
daysat        0
daysun        0
daythu        0
daytue        0
daywed        0
monthapr      0
```

In [5]:
```
forest_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517 entries, 0 to 516
Data columns (total 31 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   month          517 non-null    object 
 1   day            517 non-null    object 
 2   FFMC           517 non-null    float64
 3   DMC            517 non-null    float64
 4   DC             517 non-null    float64
 5   ISI            517 non-null    float64
 6   temp           517 non-null    float64
 7   RH             517 non-null    int64  
 8   wind           517 non-null    float64
 9   rain           517 non-null    float64
 10  area           517 non-null    float64
 11  dayfri         517 non-null    int64  
 12  daymon         517 non-null    int64  
 13  daysat         517 non-null    int64  
 14  daysun         517 non-null    int64  
 15  daythu         517 non-null    int64  
 16  daytue         517 non-null    int64  
 17  daywed         517 non-null    int64  
 18  monthapr       517 non-null    int64  
 19  monthaug       517 non-null    int64  
 20  monthdec       517 non-null    int64  
 21  monthfeb       517 non-null    int64  
 22  monthjan       517 non-null    int64  
 23  monthjul       517 non-null    int64  
 24  monthjun       517 non-null    int64  
 25  monthmar       517 non-null    int64  
 26  monthmay       517 non-null    int64  
 27  monthnov       517 non-null    int64  
 28  monthoct       517 non-null    int64  
 29  monthsep       517 non-null    int64  
 30  size_category  517 non-null    object 
dtypes: float64(8), int64(20), object(3)
memory usage: 125.3+ KB
```

In [6]:
```
forest_data.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
FFMC	517.0	90.644681	5.520111	18.7	90.2	91.60	92.90	96.20
DMC	517.0	110.872340	64.046482	1.1	68.6	108.30	142.40	291.30
DC	517.0	547.940039	248.066192	7.9	437.7	664.20	713.90	860.60
ISI	517.0	9.021663	4.559477	0.0	6.5	8.40	10.80	56.10
temp	517.0	18.889168	5.806625	2.2	15.5	19.30	22.80	33.30
RH	517.0	44.288201	16.317469	15.0	33.0	42.00	53.00	100.00
wind	517.0	4.017602	1.791653	0.4	2.7	4.00	4.90	9.40
rain	517.0	0.021663	0.295959	0.0	0.0	0.00	0.00	6.40
area	517.0	12.847292	63.655818	0.0	0.0	0.52	6.57	1090.84
dayfri	517.0	0.164410	0.371006	0.0	0.0	0.00	0.00	1.00
daymon	517.0	0.143133	0.350548	0.0	0.0	0.00	0.00	1.00
daysat	517.0	0.162476	0.369244	0.0	0.0	0.00	0.00	1.00
daysun	517.0	0.183752	0.387657	0.0	0.0	0.00	0.00	1.00
daythu	517.0	0.117988	0.322907	0.0	0.0	0.00	0.00	1.00
daytue	517.0	0.123791	0.329662	0.0	0.0	0.00	0.00	1.00
daywed	517.0	0.104449	0.306138	0.0	0.0	0.00	0.00	1.00
monthapr	517.0	0.017408	0.130913	0.0	0.0	0.00	0.00	1.00
monthaug	517.0	0.355899	0.479249	0.0	0.0	0.00	1.00	1.00
monthdec	517.0	0.017408	0.130913	0.0	0.0	0.00	0.00	1.00
monthfeb	517.0	0.038685	0.193029	0.0	0.0	0.00	0.00	1.00
monthjan	517.0	0.003868	0.062137	0.0	0.0	0.00	0.00	1.00
monthjul	517.0	0.061896	0.241199	0.0	0.0	0.00	0.00	1.00
monthjun	517.0	0.032882	0.178500	0.0	0.0	0.00	0.00	1.00
monthmar	517.0	0.104449	0.306138	0.0	0.0	0.00	0.00	1.00
monthmay	517.0	0.003868	0.062137	0.0	0.0	0.00	0.00	1.00
monthnov	517.0	0.001934	0.043980	0.0	0.0	0.00	0.00	1.00
monthoct	517.0	0.029014	0.168007	0.0	0.0	0.00	0.00	1.00
monthsep	517.0	0.332689	0.471632	0.0	0.0	0.00	1.00	1.00

In [7]:

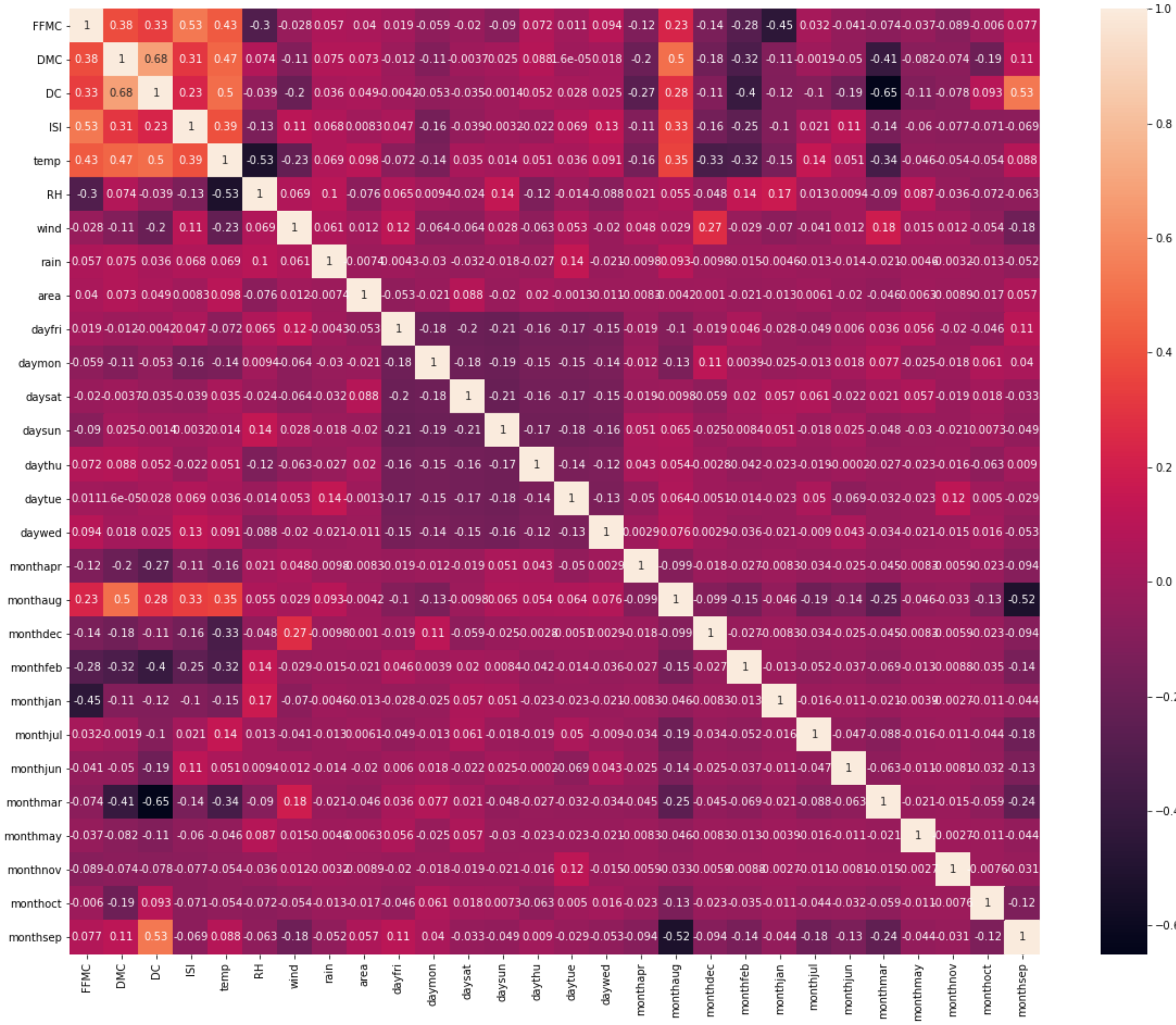
```
forest_data_corr = forest_data.corr()
forest_data_corr
```

Out[7]:

	FFMC	DMC	DC	ISI	temp	RH	wind	rain	
FFMC	1.000000	0.382619	0.330512	0.531805	0.431532	-0.300995	-0.028485	0.056702	
DMC	0.382619	1.000000	0.682192	0.305128	0.469594	0.073795	-0.105342	0.074790	
DC	0.330512	0.682192	1.000000	0.229154	0.496208	-0.039192	-0.203466	0.035861	
ISI	0.531805	0.305128	0.229154	1.000000	0.394287	-0.132517	0.106826	0.067668	
temp	0.431532	0.469594	0.496208	0.394287	1.000000	-0.527390	-0.227116	0.069491	
RH	-0.300995	0.073795	-0.039192	-0.132517	-0.527390	1.000000	0.069410	0.099751	-
wind	-0.028485	-0.105342	-0.203466	0.106826	-0.227116	0.069410	1.000000	0.061119	
rain	0.056702	0.074790	0.035861	0.067668	0.069491	0.099751	0.061119	1.000000	-
area	0.040122	0.072994	0.049383	0.008258	0.097844	-0.075519	0.012317	-0.007366	
dayfri	0.019306	-0.012010	-0.004220	0.046695	-0.071949	0.064506	0.118090	-0.004261	-
daymon	-0.059396	-0.107921	-0.052993	-0.158601	-0.136529	0.009376	-0.063881	-0.029945	-
daysat	-0.019637	-0.003653	-0.035189	-0.038585	0.034899	-0.023869	-0.063799	-0.032271	
daysun	-0.089517	0.025355	-0.001431	-0.003243	0.014403	0.136220	0.027981	-0.017872	-
daythu	0.071730	0.087672	0.051859	-0.022406	0.051432	-0.123061	-0.062553	-0.026798	
daytue	0.011225	0.000016	0.028368	0.068610	0.035630	-0.014211	0.053396	0.139311	-
daywed	0.093908	0.017939	0.024803	0.125415	0.090580	-0.087508	-0.019965	-0.020744	-
monthapr	-0.117199	-0.197543	-0.268211	-0.106478	-0.157051	0.021235	0.048266	-0.009752	-
monthaug	0.228103	0.497928	0.279361	0.334639	0.351404	0.054761	0.028577	0.093101	-
monthdec	-0.137044	-0.176301	-0.105642	-0.162322	-0.329648	-0.047714	0.269702	-0.009752	
monthfeb	-0.281535	-0.317899	-0.399277	-0.249777	-0.320015	0.140430	-0.029431	-0.014698	-
monthjan	-0.454771	-0.105647	-0.115064	-0.103588	-0.146520	0.170923	-0.070245	-0.004566	-
monthjul	0.031833	-0.001946	-0.100887	0.020982	0.142588	0.013185	-0.040645	-0.013390	
monthjun	-0.040634	-0.050403	-0.186183	0.111516	0.051015	0.009382	0.012124	-0.013510	-
monthmar	-0.074327	-0.407404	-0.650427	-0.143520	-0.341797	-0.089836	0.181433	-0.020744	-
monthmay	-0.037230	-0.081980	-0.114209	-0.060493	-0.045540	0.086822	0.015054	-0.004566	
monthnov	-0.088964	-0.074218	-0.078380	-0.076559	-0.053798	-0.035885	0.011864	-0.003225	-
monthoct	-0.005998	-0.187632	0.093279	-0.071154	-0.053513	-0.072334	-0.053850	-0.012665	-
monthsep	0.076609	0.110907	0.531857	-0.068877	0.088006	-0.062596	-0.181476	-0.051733	

28 rows × 28 columns

In [8]:
```
import seaborn as sns
plt.figure(figsize=(20,16))
sns.heatmap(forest_data_corr, annot = True)
None
```



In [9]:
```
from sklearn.metrics import classification_report
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
forest_data['size_category']= label_encoder.fit_transform(forest_data['size_categ
```

In [10]:
```
forest_data=forest_data.drop(columns=['month','day'],axis=1)
forest_data
```

Out[10]:

	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	dayfri	...	monthfeb	monthjan	mon
0	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.00	1	...	0	0	
1	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.00	0	...	0	0	
2	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.00	0	...	0	0	
3	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.00	1	...	0	0	
4	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.00	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
512	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	6.44	0	...	0	0	
513	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	54.29	0	...	0	0	
514	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	11.16	0	...	0	0	
515	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	0.00	0	...	0	0	
516	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	0.00	0	...	0	0	

517 rows × 29 columns

In [11]:
```
forest_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517 entries, 0 to 516
Data columns (total 29 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   FFMC           517 non-null    float64
 1   DMC            517 non-null    float64
 2   DC             517 non-null    float64
 3   ISI            517 non-null    float64
 4   temp           517 non-null    float64
 5   RH             517 non-null    int64  
 6   wind           517 non-null    float64
 7   rain           517 non-null    float64
 8   area           517 non-null    float64
 9   dayfri         517 non-null    int64  
 10  daymon         517 non-null    int64  
 11  daysat         517 non-null    int64  
 12  daysun         517 non-null    int64  
 13  daythu         517 non-null    int64  
 14  daytue         517 non-null    int64  
 15  daywed         517 non-null    int64  
 16  monthapr       517 non-null    int64  
 17  monthaug       517 non-null    int64  
 18  monthdec       517 non-null    int64  
 19  monthfeb       517 non-null    int64  
 20  monthjan       517 non-null    int64  
 21  monthjul       517 non-null    int64  
 22  monthjun       517 non-null    int64  
 23  monthmar       517 non-null    int64  
 24  monthmay       517 non-null    int64  
 25  monthnov       517 non-null    int64  
 26  monthoct       517 non-null    int64  
 27  monthsep       517 non-null    int64  
 28  size_category  517 non-null    int32  
dtypes: float64(8), int32(1), int64(20)
memory usage: 115.2 KB
```

In [12]:
```
x = forest_data.iloc[:,0:-1]
y = forest_data['size_category']
```

In [13]:
```
x
```

Out[13]:

	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	dayfri	...	monthdec	monthfeb	mo
0	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.00	1	...	0	0	
1	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.00	0	...	0	0	
2	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.00	0	...	0	0	
3	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.00	1	...	0	0	
4	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.00	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
512	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	6.44	0	...	0	0	
513	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	54.29	0	...	0	0	
514	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	11.16	0	...	0	0	
515	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	0.00	0	...	0	0	
516	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	0.00	0	...	0	0	

517 rows × 28 columns

In [14]:
```
y
```

Out[14]:
```
0      1
1      1
2      1
3      1
4      1
      ..
512    0
513    0
514    0
515    1
516    1
Name: size_category, Length: 517, dtype: int32
```

In [15]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size=0.20)
```

Artificial neural network model - backpropagation

In [16]:
```
#create model
model = Sequential()
model.add(Dense(42, input_dim =28, activation = 'relu'))
model.add(Dense(28,activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

In [17]:
```
#Complile model
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics=['accurac
```

In [18]:
```
#Fit the model
history   = model.fit(x_train,y_train, validation_split=0.33,epochs=180,batch_siz
```

```
Epoch 1/180
28/28 [==============================] - 4s 32ms/step - loss: 3.9574 - accura
cy: 0.6051 - val_loss: 2.5061 - val_accuracy: 0.5328
Epoch 2/180
28/28 [==============================] - 0s 7ms/step - loss: 1.9466 - accurac
y: 0.6341 - val_loss: 1.3441 - val_accuracy: 0.6788
Epoch 3/180
28/28 [==============================] - 0s 8ms/step - loss: 1.3318 - accurac
y: 0.6775 - val_loss: 2.0934 - val_accuracy: 0.7591
Epoch 4/180
28/28 [==============================] - 0s 7ms/step - loss: 2.1458 - accurac
y: 0.6920 - val_loss: 0.8428 - val_accuracy: 0.7445
Epoch 5/180
28/28 [==============================] - 0s 7ms/step - loss: 0.8388 - accurac
y: 0.7391 - val_loss: 0.6515 - val_accuracy: 0.7810
Epoch 6/180
28/28 [==============================] - 0s 7ms/step - loss: 0.7056 - accurac
y: 0.7681 - val_loss: 0.6501 - val_accuracy: 0.7518
Epoch 7/180
```

In [24]:
```
#evalute the model
score = model.evaluate(x,y)
print("%s; %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

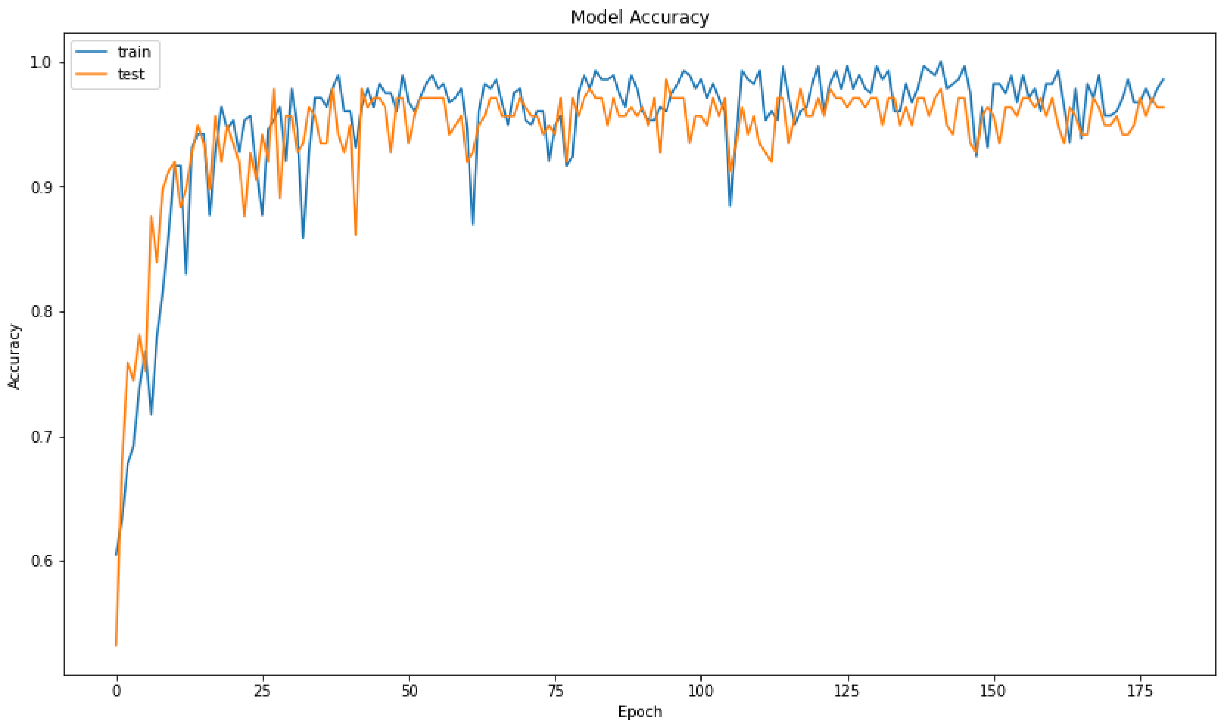
```
17/17 [==============================] - 0s 3ms/step - loss: 0.0913 - accuracy:
0.9691
accuracy; 96.91%
```

In [25]:
```
# Visualizing training history
# list all data in history
history.history.keys()
```

Out[25]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

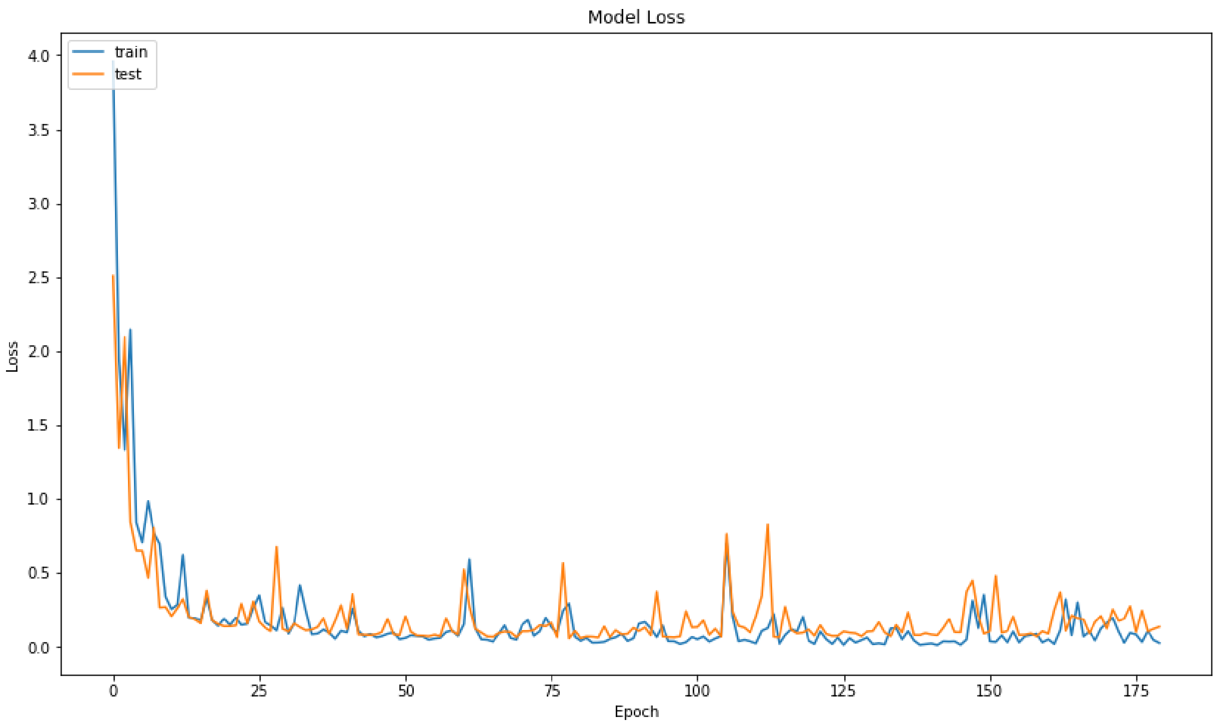
In [26]:
```
# summarize history for accuracy
from matplotlib import pyplot as plt

plt.figure(figsize=(14,8))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['train','test'], loc='upper left')
plt.show()
```



In [27]:
```
# summarize history for loss

plt.figure(figsize=(14,8))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



hyper parameter tuning

In [28]:
```
X =forest_data.iloc[:,0:-1]
Y =forest_data['size_category']
```

In [31]:
```
#standardization
from sklearn.preprocessing import StandardScaler
a = StandardScaler()
a.fit(X)
X_standardized = a.transform(X)
```

In [32]:
```
pd.DataFrame(X_standardized).describe().T
```

Out[32]:

	count	mean	std	min	25%	50%	75%	max
0	517.0	-1.754024e-15	1.000969	-13.045818	-0.080635	0.173229	0.408960	1.007353
1	517.0	3.070830e-16	1.000969	-1.715608	-0.660665	-0.040203	0.492739	2.819865
2	517.0	7.387171e-17	1.000969	-2.179108	-0.444828	0.469119	0.669663	1.261610
3	517.0	-3.865380e-17	1.000969	-1.980578	-0.553595	-0.136477	0.390409	10.335381
4	517.0	2.005703e-16	1.000969	-2.876943	-0.584238	0.070821	0.674164	2.484195
5	517.0	3.362881e-16	1.000969	-1.796637	-0.692456	-0.140366	0.534411	3.417549
6	517.0	-2.676776e-16	1.000969	-2.021098	-0.736124	-0.009834	0.492982	3.007063
7	517.0	-2.841054e-16	1.000969	-0.073268	-0.073268	-0.073268	-0.073268	21.572284
8	517.0	-1.274502e-16	1.000969	-0.202020	-0.202020	-0.193843	-0.098709	16.951110
9	517.0	4.874674e-17	1.000969	-0.443576	-0.443576	-0.443576	-0.443576	2.254407
10	517.0	-1.868267e-16	1.000969	-0.408709	-0.408709	-0.408709	-0.408709	2.446730
11	517.0	-2.238699e-16	1.000969	-0.440449	-0.440449	-0.440449	-0.440449	2.270410
12	517.0	-6.098711e-17	1.000969	-0.474467	-0.474467	-0.474467	-0.474467	2.107630
13	517.0	-1.004999e-16	1.000969	-0.365748	-0.365748	-0.365748	-0.365748	2.734120
14	517.0	2.405125e-17	1.000969	-0.375873	-0.375873	-0.375873	-0.375873	2.660475
15	517.0	-3.843906e-17	1.000969	-0.341512	-0.341512	-0.341512	-0.341512	2.928152
16	517.0	-1.344293e-16	1.000969	-0.133103	-0.133103	-0.133103	-0.133103	7.512952
17	517.0	2.473843e-16	1.000969	-0.743339	-0.743339	-0.743339	1.345282	1.345282
18	517.0	7.179943e-16	1.000969	-0.133103	-0.133103	-0.133103	-0.133103	7.512952
19	517.0	-1.933764e-16	1.000969	-0.200603	-0.200603	-0.200603	-0.200603	4.984977
20	517.0	-2.260174e-17	1.000969	-0.062318	-0.062318	-0.062318	-0.062318	16.046807
21	517.0	1.352883e-17	1.000969	-0.256865	-0.256865	-0.256865	-0.256865	3.893103
22	517.0	1.169277e-16	1.000969	-0.184391	-0.184391	-0.184391	-0.184391	5.423261
23	517.0	2.265542e-16	1.000969	-0.341512	-0.341512	-0.341512	-0.341512	2.928152
24	517.0	-2.596515e-16	1.000969	-0.062318	-0.062318	-0.062318	-0.062318	16.046807
25	517.0	1.443075e-16	1.000969	-0.044023	-0.044023	-0.044023	-0.044023	22.715633
26	517.0	6.253326e-16	1.000969	-0.172860	-0.172860	-0.172860	-0.172860	5.785038
27	517.0	4.024290e-16	1.000969	-0.706081	-0.706081	-0.706081	1.416268	1.416268

Tuning of hyperparameters - Batch size and Epochs

In [35]:
```
!pip install keras
```

```
Requirement already satisfied: keras in e:\anaconda\lib\site-packages (2.7.0)
```

In [36]:
```
from sklearn.model_selection import GridSearchCV
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from tensorflow.keras.optimizers import Adam
```

In [37]:
```
# create model
def create_model():
    model = Sequential()
    model.add(Dense(12, input_dim=28, kernel_initializer='uniform', activation='r
    model.add(Dense(8, kernel_initializer='uniform', activation='relu'))
    model.add(Dense(1, kernel_initializer='uniform', activation='sigmoid'))

    adam=Adam(lr=0.01)
    model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'
    return model
```

In [38]:
```
#create the model
model = KerasClassifier(build_fn= create_model,verbose = 0)

#Define the grid search parameters
batch_size = [10,20,40]
epochs = [10,50,100]
# Make a dictionary of thr grid of the search parameters
param_grid = dict(batch_size = batch_size,epochs = epochs)
# build and fit the GridSearchCV
grid = GridSearchCV(estimator = model,param_grid= param_grid, verbose = 10)
grid_result = grid.fit(X_standardized,Y)
```

```
Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 1/5; 1/9] START batch_size=10, epochs=1
0...................................
[CV 1/5; 1/9] END .................batch_size=10, epochs=10; total time=   5.9
s
[CV 2/5; 1/9] START batch_size=10, epochs=1
0...................................
[CV 2/5; 1/9] END .................batch_size=10, epochs=10; total time=   5.2
s
[CV 3/5; 1/9] START batch_size=10, epochs=1
0...................................
[CV 3/5; 1/9] END .................batch_size=10, epochs=10; total time=   5.0
s
[CV 4/5; 1/9] START batch_size=10, epochs=1
0...................................
[CV 4/5; 1/9] END .................batch_size=10, epochs=10; total time=   5.0
s
[CV 5/5; 1/9] START batch_size=10, epochs=1
0...................................
[CV 5/5; 1/9] END .................batch_size=10, epochs=10; total time=   5.0
s
[CV 1/5; 2/9] START batch_size=10, epochs=5
0...................................
[CV 1/5; 2/9] END .................batch_size=10, epochs=50; total time=  11.9
s
[CV 2/5; 2/9] START batch_size=10, epochs=5
0...................................
[CV 2/5; 2/9] END .................batch_size=10, epochs=50; total time=  11.9
s
[CV 3/5; 2/9] START batch_size=10, epochs=5
0...................................
[CV 3/5; 2/9] END .................batch_size=10, epochs=50; total time=  11.8
s
[CV 4/5; 2/9] START batch_size=10, epochs=5
0...................................
[CV 4/5; 2/9] END .................batch_size=10, epochs=50; total time=  11.8
s
[CV 5/5; 2/9] START batch_size=10, epochs=5
0...................................
[CV 5/5; 2/9] END .................batch_size=10, epochs=50; total time=  11.8
s
[CV 1/5; 3/9] START batch_size=10, epochs=10
0..................................
[CV 1/5; 3/9] END ................batch_size=10, epochs=100; total time=  21.6
s
```

```
[CV 2/5; 3/9] START batch_size=10, epochs=10
0.................................
[CV 2/5; 3/9] END ................batch_size=10, epochs=100; total time=  20.5
s
[CV 3/5; 3/9] START batch_size=10, epochs=10
0.................................
[CV 3/5; 3/9] END ................batch_size=10, epochs=100; total time=  20.6
s
[CV 4/5; 3/9] START batch_size=10, epochs=10
0.................................
[CV 4/5; 3/9] END ................batch_size=10, epochs=100; total time=  20.3
s
[CV 5/5; 3/9] START batch_size=10, epochs=10
0.................................
[CV 5/5; 3/9] END ................batch_size=10, epochs=100; total time=  20.3
s
[CV 1/5; 4/9] START batch_size=20, epochs=1
0..................................
[CV 1/5; 4/9] END .................batch_size=20, epochs=10; total time=   4.2
s
[CV 2/5; 4/9] START batch_size=20, epochs=1
0..................................
[CV 2/5; 4/9] END .................batch_size=20, epochs=10; total time=   4.3
s
[CV 3/5; 4/9] START batch_size=20, epochs=1
0..................................
[CV 3/5; 4/9] END .................batch_size=20, epochs=10; total time=   4.2
s
[CV 4/5; 4/9] START batch_size=20, epochs=1
0..................................
[CV 4/5; 4/9] END .................batch_size=20, epochs=10; total time=   4.2
s
[CV 5/5; 4/9] START batch_size=20, epochs=1
0..................................
[CV 5/5; 4/9] END .................batch_size=20, epochs=10; total time=   4.2
s
[CV 1/5; 5/9] START batch_size=20, epochs=5
0..................................
[CV 1/5; 5/9] END .................batch_size=20, epochs=50; total time=   7.7
s
[CV 2/5; 5/9] START batch_size=20, epochs=5
0..................................
[CV 2/5; 5/9] END .................batch_size=20, epochs=50; total time=   7.7
s
[CV 3/5; 5/9] START batch_size=20, epochs=5
0..................................
[CV 3/5; 5/9] END .................batch_size=20, epochs=50; total time=   8.7
s
[CV 4/5; 5/9] START batch_size=20, epochs=5
0..................................
[CV 4/5; 5/9] END .................batch_size=20, epochs=50; total time=   7.6
s
[CV 5/5; 5/9] START batch_size=20, epochs=5
0..................................
[CV 5/5; 5/9] END .................batch_size=20, epochs=50; total time=   7.6
s
[CV 1/5; 6/9] START batch_size=20, epochs=10
```

```
0..................................
[CV 1/5; 6/9] END ................batch_size=20, epochs=100; total time=  12.1
s
[CV 2/5; 6/9] START batch_size=20, epochs=10
0..................................
[CV 2/5; 6/9] END ................batch_size=20, epochs=100; total time=  12.0
s
[CV 3/5; 6/9] START batch_size=20, epochs=10
0..................................
[CV 3/5; 6/9] END ................batch_size=20, epochs=100; total time=  12.5
s
[CV 4/5; 6/9] START batch_size=20, epochs=10
0..................................
[CV 4/5; 6/9] END ................batch_size=20, epochs=100; total time=  12.1
s
[CV 5/5; 6/9] START batch_size=20, epochs=10
0..................................
[CV 5/5; 6/9] END ................batch_size=20, epochs=100; total time=  12.0
s
[CV 1/5; 7/9] START batch_size=40, epochs=1
0...................................
[CV 1/5; 7/9] END .................batch_size=40, epochs=10; total time=   3.9
s
[CV 2/5; 7/9] START batch_size=40, epochs=1
0...................................
[CV 2/5; 7/9] END .................batch_size=40, epochs=10; total time=   3.8
s
[CV 3/5; 7/9] START batch_size=40, epochs=1
0...................................
[CV 3/5; 7/9] END .................batch_size=40, epochs=10; total time=   3.8
s
[CV 4/5; 7/9] START batch_size=40, epochs=1
0...................................
WARNING:tensorflow:5 out of the last 16 calls to <function Model.make_test_func
tion.<locals>.test_function at 0x000001FA0DC29B80> triggered tf.function retrac
ing. Tracing is expensive and the excessive number of tracings could be due to
(1) creating @tf.function repeatedly in a loop, (2) passing tensors with differ
ent shapes, (3) passing Python objects instead of tensors. For (1), please defi
ne your @tf.function outside of the loop. For (2), @tf.function has experimenta
l_relax_shapes=True option that relaxes argument shapes that can avoid unnecess
ary retracing. For (3), please refer to https://www.tensorflow.org/guide/functi
on#controlling_retracing (https://www.tensorflow.org/guide/function#controlling
_retracing) and https://www.tensorflow.org/api_docs/python/tf/function (http
s://www.tensorflow.org/api_docs/python/tf/function) for  more details.
[CV 4/5; 7/9] END .................batch_size=40, epochs=10; total time=   3.8
s
[CV 5/5; 7/9] START batch_size=40, epochs=1
0...................................
WARNING:tensorflow:5 out of the last 13 calls to <function Model.make_test_func
tion.<locals>.test_function at 0x000001FA0DB23D30> triggered tf.function retrac
ing. Tracing is expensive and the excessive number of tracings could be due to
 (1) creating @tf.function repeatedly in a loop, (2) passing tensors with diffe
rent shapes, (3) passing Python objects instead of tensors. For (1), please def
ine your @tf.function outside of the loop. For (2), @tf.function has experiment
al_relax_shapes=True option that relaxes argument shapes that can avoid unneces
sary retracing. For (3), please refer to https://www.tensorflow.org/guide/funct
ion#controlling_retracing (https://www.tensorflow.org/guide/function#controllin
```

```
g_retracing) and https://www.tensorflow.org/api_docs/python/tf/function (http
s://www.tensorflow.org/api_docs/python/tf/function) for  more details.
[CV 5/5; 7/9] END ..................batch_size=40, epochs=10; total time=   3.8
s
[CV 1/5; 8/9] START batch_size=40, epochs=5
0...................................
[CV 1/5; 8/9] END ..................batch_size=40, epochs=50; total time=   6.7
s
[CV 2/5; 8/9] START batch_size=40, epochs=5
0...................................
[CV 2/5; 8/9] END ..................batch_size=40, epochs=50; total time=   5.7
s
[CV 3/5; 8/9] START batch_size=40, epochs=5
0...................................
[CV 3/5; 8/9] END ..................batch_size=40, epochs=50; total time=   5.7
s
[CV 4/5; 8/9] START batch_size=40, epochs=5
0...................................
[CV 4/5; 8/9] END ..................batch_size=40, epochs=50; total time=   5.8
s
[CV 5/5; 8/9] START batch_size=40, epochs=5
0...................................
[CV 5/5; 8/9] END ..................batch_size=40, epochs=50; total time=   6.1
s
[CV 1/5; 9/9] START batch_size=40, epochs=10
0..................................
[CV 1/5; 9/9] END .................batch_size=40, epochs=100; total time=   8.1
s
[CV 2/5; 9/9] START batch_size=40, epochs=10
0..................................

[CV 2/5; 9/9] END .................batch_size=40, epochs=100; total time=   8.2
s
[CV 3/5; 9/9] START batch_size=40, epochs=10
0..................................
[CV 3/5; 9/9] END .................batch_size=40, epochs=100; total time=   8.2
s
[CV 4/5; 9/9] START batch_size=40, epochs=10
0..................................
[CV 4/5; 9/9] END .................batch_size=40, epochs=100; total time=   8.2
s
[CV 5/5; 9/9] START batch_size=40, epochs=10
0..................................
[CV 5/5; 9/9] END .................batch_size=40, epochs=100; total time=   8.2
s
```

In [39]:
```
# summarize the results
print('Best : {}, using {}'.format(grid_result.best_score_,grid_result.best_paran
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
# Summarize the results
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means,stds, params ):
    print('{},{} with: {}'.format(mean,stdev,param))
```

```
Best : 0.9148058295249939, using {'batch_size': 10, 'epochs': 100}
0.8645071029663086,0.04543507484343193 with: {'batch_size': 10, 'epochs': 10}
0.9109783411026001,0.0500215105010093 with: {'batch_size': 10, 'epochs': 50}
0.9148058295249939,0.04465227790273617 with: {'batch_size': 10, 'epochs': 100}
0.8547796845436096,0.07583689529403034 with: {'batch_size': 20, 'epochs': 10}
0.9070575118064881,0.04912849167571362 with: {'batch_size': 20, 'epochs': 50}
0.8954816937446595,0.05699809270164309 with: {'batch_size': 20, 'epochs': 100}
0.7635922431945801,0.15143452693911913 with: {'batch_size': 40, 'epochs': 10}
0.9109783411026001,0.04412957120431605 with: {'batch_size': 40, 'epochs': 50}
0.9051157593727112,0.04906203215779241 with: {'batch_size': 40, 'epochs': 100}
```

Tuning of Hyperparameters:- Learning rate and Drop out rate

In [40]:
```
from keras.layers import Dropout

# Defining the model

def create_model(learning_rate,dropout_rate):
    model = Sequential()
    model.add(Dense(8,input_dim = 28,kernel_initializer = 'normal',activation = '
    model.add(Dropout(dropout_rate))
    model.add(Dense(4,input_dim = 28,kernel_initializer = 'normal',activation = '
    model.add(Dropout(dropout_rate))
    model.add(Dense(1,activation = 'sigmoid'))

    adam = Adam(lr = learning_rate)
    model.compile(loss = 'binary_crossentropy',optimizer = adam,metrics = ['accur
    return model

# Create the model

model = KerasClassifier(build_fn = create_model,verbose = 0,batch_size = 40,epoch

# Define the grid search parameters

learning_rate = [0.001,0.01,0.1]
dropout_rate = [0.0,0.1,0.2]

# Make a dictionary of the grid search parameters

param_grids = dict(learning_rate = learning_rate,dropout_rate = dropout_rate)

# Build and fit the GridSearchCV

grid = GridSearchCV(estimator = model,param_grid = param_grids,verbose = 10)
grid_result = grid.fit(X_standardized,Y)
```

```
Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 1/5; 1/9] START dropout_rate=0.0, learning_rate=0.00
1.......................
[CV 1/5; 1/9] END .....dropout_rate=0.0, learning_rate=0.001; total time=
5.4s
[CV 2/5; 1/9] START dropout_rate=0.0, learning_rate=0.00
1.......................
[CV 2/5; 1/9] END .....dropout_rate=0.0, learning_rate=0.001; total time=
3.9s
[CV 3/5; 1/9] START dropout_rate=0.0, learning_rate=0.00
1.......................
[CV 3/5; 1/9] END .....dropout_rate=0.0, learning_rate=0.001; total time=
3.8s
[CV 4/5; 1/9] START dropout_rate=0.0, learning_rate=0.00
1.......................
[CV 4/5; 1/9] END .....dropout_rate=0.0, learning_rate=0.001; total time=
3.8s
[CV 5/5; 1/9] START dropout_rate=0.0, learning_rate=0.00
1.......................
```

In [41]:
```
# Summarize the results
print('Best : {}, using {}'.format(grid_result.best_score_,grid_result.best_param
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
  print('{},{} with: {}'.format(mean, stdev, param))
```

```
Best : 0.7714152336120605, using {'dropout_rate': 0.0, 'learning_rate': 0.01}
0.728659451007843,0.1510055827692834 with: {'dropout_rate': 0.0, 'learning_rat
e': 0.001}
0.7714152336120605,0.14136657433344296 with: {'dropout_rate': 0.0, 'learning_ra
te': 0.01}
0.744772219657898,0.056601793865556374 with: {'dropout_rate': 0.0, 'learning_ra
te': 0.1}
0.7305825233459473,0.15435061319000673 with: {'dropout_rate': 0.1, 'learning_ra
te': 0.001}
0.7577669858932495,0.1485473087246425 with: {'dropout_rate': 0.1, 'learning_rat
e': 0.01}
0.7305825233459473,0.15435061319000673 with: {'dropout_rate': 0.1, 'learning_ra
te': 0.1}
0.7305825233459473,0.15435061319000673 with: {'dropout_rate': 0.2, 'learning_ra
te': 0.001}
0.75,0.15082954673425533 with: {'dropout_rate': 0.2, 'learning_rate': 0.01}
0.7383495092391967,0.15354239422184465 with: {'dropout_rate': 0.2, 'learning_ra
te': 0.1}
```

uning of Hyperparameters:- Activation Function and Kernel Initializer

In [42]:
```
from keras.layers import Dropout
# Defining the model

def create_model(activation_function,init):
    model = Sequential()
    model.add(Dense(8,input_dim = 28,kernel_initializer = init,activation = activ
    model.add(Dropout(0.1))
    model.add(Dense(4,input_dim = 28,kernel_initializer = init,activation = activ
    model.add(Dropout(0.1))
    model.add(Dense(1,activation = 'sigmoid'))

    adam = Adam(lr = 0.001)
    model.compile(loss = 'binary_crossentropy',optimizer = adam,metrics = ['accur
    return model

# Create the model

model = KerasClassifier(build_fn = create_model,verbose = 0,batch_size = 40,epoch

# Define the grid search parameters
activation_function = ['softmax','relu','tanh','linear']
init = ['uniform','normal','zero']

# Make a dictionary of the grid search parameters
param_grids = dict(activation_function = activation_function,init = init)

# Build and fit the GridSearchCV

grid = GridSearchCV(estimator = model,param_grid = param_grids,verbose = 10)
grid_result = grid.fit(X_standardized,Y)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
[CV 1/5; 1/12] START activation_function=softmax, init=unifor
m..................
[CV 1/5; 1/12] END activation_function=softmax, init=uniform; total time=
5.3s
[CV 2/5; 1/12] START activation_function=softmax, init=unifor
m..................
[CV 2/5; 1/12] END activation_function=softmax, init=uniform; total time=
4.2s
[CV 3/5; 1/12] START activation_function=softmax, init=unifor
m..................
[CV 3/5; 1/12] END activation_function=softmax, init=uniform; total time=
4.3s
[CV 4/5; 1/12] START activation_function=softmax, init=unifor
m..................
[CV 4/5; 1/12] END activation_function=softmax, init=uniform; total time=
4.5s
[CV 5/5; 1/12] START activation_function=softmax, init=unifor
m..................
```

In [43]:
```
# Summarize the results
print('Best : {}, using {}'.format(grid_result.best_score_,grid_result.best_param
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
  print('{},{} with: {}'.format(mean, stdev, param))
```

```
Best : 0.7442120909690857, using {'activation_function': 'linear', 'init': 'uni
form'}
0.7305825233459473,0.15435061319000673 with: {'activation_function': 'softmax',
'init': 'uniform'}
0.5305825233459472,0.2757845556417363 with: {'activation_function': 'softmax',
'init': 'normal'}
0.7305825233459473,0.15435061319000673 with: {'activation_function': 'softmax',
'init': 'zero'}
0.7325242757797241,0.15178268966445949 with: {'activation_function': 'relu', 'i
nit': 'uniform'}
0.7325242757797241,0.15178268966445949 with: {'activation_function': 'relu', 'i
nit': 'normal'}
0.7305825233459473,0.15435061319000673 with: {'activation_function': 'relu', 'i
nit': 'zero'}
0.7383868575096131,0.1343536254145011 with: {'activation_function': 'tanh', 'in
it': 'uniform'}
0.7267923831939698,0.13622830288028945 with: {'activation_function': 'tanh', 'i
nit': 'normal'}
0.7305825233459473,0.15435061319000673 with: {'activation_function': 'tanh', 'i
nit': 'zero'}
0.7442120909690857,0.1263171221477866 with: {'activation_function': 'linear',
'init': 'uniform'}
0.7383495211601258,0.14611891933450738 with: {'activation_function': 'linear',
'init': 'normal'}
0.7305825233459473,0.15435061319000673 with: {'activation_function': 'linear',
'init': 'zero'}
```

Tuning of Hyperparameter :-Number of Neurons in activation layer

In [44]:
```
# Defining the model

def create_model(neuron1,neuron2):
    model = Sequential()
    model.add(Dense(neuron1,input_dim = 28,kernel_initializer = 'uniform',activat
    model.add(Dropout(0.1))
    model.add(Dense(neuron2,input_dim = neuron1,kernel_initializer = 'uniform',ac
    model.add(Dropout(0.1))
    model.add(Dense(1,activation = 'sigmoid'))

    adam = Adam(lr = 0.001)
    model.compile(loss = 'binary_crossentropy',optimizer = adam,metrics = ['accur
    return model

# Create the model

model = KerasClassifier(build_fn = create_model,verbose = 0,batch_size = 40,epoch

# Define the grid search parameters

neuron1 = [4,8,16]
neuron2 = [2,4,8]

# Make a dictionary of the grid search parameters

param_grids = dict(neuron1 = neuron1,neuron2 = neuron2)

# Build and fit the GridSearchCV

grid = GridSearchCV(estimator = model,param_grid = param_grids,verbose = 10)
grid_result = grid.fit(X_standardized,Y)
```

```
Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 1/5; 1/9] START neuron1=4, neuron2=
2.........................................
[CV 1/5; 1/9] END ......................neuron1=4, neuron2=2; total time=   4.1
s
[CV 2/5; 1/9] START neuron1=4, neuron2=
2.........................................
[CV 2/5; 1/9] END ......................neuron1=4, neuron2=2; total time=   4.3
s
[CV 3/5; 1/9] START neuron1=4, neuron2=
2.........................................
[CV 3/5; 1/9] END ......................neuron1=4, neuron2=2; total time=   4.1
s
[CV 4/5; 1/9] START neuron1=4, neuron2=
2.........................................
[CV 4/5; 1/9] END ......................neuron1=4, neuron2=2; total time=   4.1
s
[CV 5/5; 1/9] START neuron1=4, neuron2=
2.........................................
[CV 5/5; 1/9] END ......................neuron1=4, neuron2=2; total time=   4.3
s
[CV 1/5; 2/9] START neuron1=4, neuron2=
4.........................................
[CV 1/5; 2/9] END ......................neuron1=4, neuron2=4; total time=   4.2
s
```

```
[CV 2/5; 2/9] START neuron1=4, neuron2=
4.......................................
[CV 2/5; 2/9] END .....................neuron1=4, neuron2=4; total time=   4.1
s
[CV 3/5; 2/9] START neuron1=4, neuron2=
4.......................................
[CV 3/5; 2/9] END .....................neuron1=4, neuron2=4; total time=   5.1
s
[CV 4/5; 2/9] START neuron1=4, neuron2=
4.......................................
[CV 4/5; 2/9] END .....................neuron1=4, neuron2=4; total time=   4.1
s
[CV 5/5; 2/9] START neuron1=4, neuron2=
4.......................................
[CV 5/5; 2/9] END .....................neuron1=4, neuron2=4; total time=   4.3
s
[CV 1/5; 3/9] START neuron1=4, neuron2=
8.......................................
[CV 1/5; 3/9] END .....................neuron1=4, neuron2=8; total time=   4.3
s
[CV 2/5; 3/9] START neuron1=4, neuron2=
8.......................................
[CV 2/5; 3/9] END .....................neuron1=4, neuron2=8; total time=   4.6
s
[CV 3/5; 3/9] START neuron1=4, neuron2=
8.......................................
[CV 3/5; 3/9] END .....................neuron1=4, neuron2=8; total time=   4.3
s
[CV 4/5; 3/9] START neuron1=4, neuron2=
8.......................................
[CV 4/5; 3/9] END .....................neuron1=4, neuron2=8; total time=   4.3
s
[CV 5/5; 3/9] START neuron1=4, neuron2=
8.......................................
[CV 5/5; 3/9] END .....................neuron1=4, neuron2=8; total time=   4.1
s
[CV 1/5; 4/9] START neuron1=8, neuron2=
2.......................................
[CV 1/5; 4/9] END .....................neuron1=8, neuron2=2; total time=   4.1
s
[CV 2/5; 4/9] START neuron1=8, neuron2=
2.......................................
[CV 2/5; 4/9] END .....................neuron1=8, neuron2=2; total time=   4.1
s
[CV 3/5; 4/9] START neuron1=8, neuron2=
2.......................................
[CV 3/5; 4/9] END .....................neuron1=8, neuron2=2; total time=   4.1
s
[CV 4/5; 4/9] START neuron1=8, neuron2=
2.......................................
[CV 4/5; 4/9] END .....................neuron1=8, neuron2=2; total time=   5.0
s
[CV 5/5; 4/9] START neuron1=8, neuron2=
2.......................................
[CV 5/5; 4/9] END .....................neuron1=8, neuron2=2; total time=   4.1
s
[CV 1/5; 5/9] START neuron1=8, neuron2=
```

```
4.......................................
[CV 1/5; 5/9] END .....................neuron1=8, neuron2=4; total time=   4.1
s
[CV 2/5; 5/9] START neuron1=8, neuron2=
4.......................................
[CV 2/5; 5/9] END .....................neuron1=8, neuron2=4; total time=   4.1
s
[CV 3/5; 5/9] START neuron1=8, neuron2=
4.......................................
[CV 3/5; 5/9] END .....................neuron1=8, neuron2=4; total time=   4.1
s
[CV 4/5; 5/9] START neuron1=8, neuron2=
4.......................................
[CV 4/5; 5/9] END .....................neuron1=8, neuron2=4; total time=   4.1
s
[CV 5/5; 5/9] START neuron1=8, neuron2=
4.......................................
[CV 5/5; 5/9] END .....................neuron1=8, neuron2=4; total time=   4.1
s
[CV 1/5; 6/9] START neuron1=8, neuron2=
8.......................................
[CV 1/5; 6/9] END .....................neuron1=8, neuron2=8; total time=   4.1
s
[CV 2/5; 6/9] START neuron1=8, neuron2=
8.......................................
[CV 2/5; 6/9] END .....................neuron1=8, neuron2=8; total time=   4.2
s
[CV 3/5; 6/9] START neuron1=8, neuron2=
8.......................................
[CV 3/5; 6/9] END .....................neuron1=8, neuron2=8; total time=   4.1
s
[CV 4/5; 6/9] START neuron1=8, neuron2=
8.......................................
[CV 4/5; 6/9] END .....................neuron1=8, neuron2=8; total time=   4.1
s
[CV 5/5; 6/9] START neuron1=8, neuron2=
8.......................................
[CV 5/5; 6/9] END .....................neuron1=8, neuron2=8; total time=   5.0
s
[CV 1/5; 7/9] START neuron1=16, neuron2=
2......................................
[CV 1/5; 7/9] END ....................neuron1=16, neuron2=2; total time=   4.1
s
[CV 2/5; 7/9] START neuron1=16, neuron2=
2......................................
[CV 2/5; 7/9] END ....................neuron1=16, neuron2=2; total time=   4.1
s
[CV 3/5; 7/9] START neuron1=16, neuron2=
2......................................
[CV 3/5; 7/9] END ....................neuron1=16, neuron2=2; total time=   4.1
s
[CV 4/5; 7/9] START neuron1=16, neuron2=
2......................................
[CV 4/5; 7/9] END ....................neuron1=16, neuron2=2; total time=   4.3
s
[CV 5/5; 7/9] START neuron1=16, neuron2=
2......................................
```

```
[CV 5/5; 7/9] END ....................neuron1=16, neuron2=2; total time=   4.3
s
[CV 1/5; 8/9] START neuron1=16, neuron2=
4.....................................
[CV 1/5; 8/9] END ....................neuron1=16, neuron2=4; total time=   4.3
s
[CV 2/5; 8/9] START neuron1=16, neuron2=
4.....................................
[CV 2/5; 8/9] END ....................neuron1=16, neuron2=4; total time=   4.3
s
[CV 3/5; 8/9] START neuron1=16, neuron2=
4.....................................
[CV 3/5; 8/9] END ....................neuron1=16, neuron2=4; total time=   4.2
s
[CV 4/5; 8/9] START neuron1=16, neuron2=
4.....................................
[CV 4/5; 8/9] END ....................neuron1=16, neuron2=4; total time=   4.2
s
[CV 5/5; 8/9] START neuron1=16, neuron2=
4.....................................
[CV 5/5; 8/9] END ....................neuron1=16, neuron2=4; total time=   4.1
s
[CV 1/5; 9/9] START neuron1=16, neuron2=
8.....................................
[CV 1/5; 9/9] END ....................neuron1=16, neuron2=8; total time=   5.0
s
[CV 2/5; 9/9] START neuron1=16, neuron2=
8.....................................
[CV 2/5; 9/9] END ....................neuron1=16, neuron2=8; total time=   4.1
s
[CV 3/5; 9/9] START neuron1=16, neuron2=
8.....................................
[CV 3/5; 9/9] END ....................neuron1=16, neuron2=8; total time=   4.1
s
[CV 4/5; 9/9] START neuron1=16, neuron2=
8.....................................
[CV 4/5; 9/9] END ....................neuron1=16, neuron2=8; total time=   4.1
s
[CV 5/5; 9/9] START neuron1=16, neuron2=
8.....................................
[CV 5/5; 9/9] END ....................neuron1=16, neuron2=8; total time=   4.3
s
```

In [45]:
```
# Summarize the results
print('Best : {}, using {}'.format(grid_result.best_score_,grid_result.best_param
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
  print('{},{} with: {}'.format(mean, stdev, param))
```

```
Best : 0.7597460746765137, using {'neuron1': 8, 'neuron2': 2}
0.7519417405128479,0.13090965185428644 with: {'neuron1': 4, 'neuron2': 2}
0.7442681074142456,0.11234521971869296 with: {'neuron1': 4, 'neuron2': 4}
0.7481329202651977,0.11386488121195477 with: {'neuron1': 4, 'neuron2': 8}
0.7597460746765137,0.1141448059440758 with: {'neuron1': 8, 'neuron2': 2}
0.7365011215209961,0.12116406906444306 with: {'neuron1': 8, 'neuron2': 4}
0.7422890305519104,0.12394596970524079 with: {'neuron1': 8, 'neuron2': 8}
0.7423450469970703,0.11163950404768495 with: {'neuron1': 16, 'neuron2': 2}
0.7443054556846619,0.10033954316259028 with: {'neuron1': 16, 'neuron2': 4}
0.7481702923774719,0.10204383026701777 with: {'neuron1': 16, 'neuron2': 8}
```

Training model with optimum values of Hyperparameters

In [46]:
```
from sklearn.metrics import classification_report, accuracy_score

# Defining the model

def create_model():
    model = Sequential()
    model.add(Dense(16,input_dim = 28,kernel_initializer = 'uniform',activation =
    model.add(Dropout(0.1))
    model.add(Dense(4,input_dim = 16,kernel_initializer = 'uniform',activation =
    model.add(Dropout(0.1))
    model.add(Dense(1,activation = 'sigmoid'))

    adam = Adam(lr = 0.001) #sgd = SGD(lr=learning_rate, momentum=momentum, decay
    model.compile(loss = 'binary_crossentropy',optimizer = adam,metrics = ['accur
    return model

# Create the model

model = KerasClassifier(build_fn = create_model,verbose = 0,batch_size = 40,epoch

# Fitting the model

model.fit(X_standardized,Y)

# Predicting using trained model

y_predict = model.predict(X_standardized)

# Printing the metrics
print(accuracy_score(Y,y_predict))
```

```
0.7794970986460348
```

Hyperparameters all at once-

- The hyperparameter optimization was carried out by taking 2 hyperparameters at once. We may have missed the best values. The performance can be further improved by finding the optimum values of hyperparameters all at once given by the code snippet below.*

In [47]:
```
def create_model(learning_rate,dropout_rate,activation_function,init,neuron1,neur
    model = Sequential()
    model.add(Dense(neuron1,input_dim = 28,kernel_initializer = init,activation =
    model.add(Dropout(dropout_rate))
    model.add(Dense(neuron2,input_dim = neuron1,kernel_initializer = init,activat
    model.add(Dropout(dropout_rate))
    model.add(Dense(1,activation = 'sigmoid'))

    adam = Adam(lr = learning_rate)
    model.compile(loss = 'binary_crossentropy',optimizer = adam,metrics = ['accur
    return model
```

In [48]:
```
# Create the model

model = KerasClassifier(build_fn = create_model,verbose = 0)
```

In [49]:
```
# Define the grid search parameters

batch_size = [10,20,40]
epochs = [10,50,100]
learning_rate = [0.001,0.01,0.1]
dropout_rate = [0.0,0.1,0.2]
activation_function = ['softmax','relu','tanh','linear']
init = ['uniform','normal','zero']
neuron1 = [4,8,16]
neuron2 = [2,4,8]
```

In [50]:
```
# Make a dictionary of the grid search parameters

param_grids = dict(batch_size = batch_size,epochs = epochs,learning_rate = learni
                   activation_function = activation_function,init = init,neuron1
```

In [51]:
```
# Build and fit the GridSearchCV

grid = GridSearchCV(estimator = model,param_grid = param_grids,verbose = 10)
grid_result = grid.fit(X_standardized,Y)

# Summarize the results
print('Best : {}, using {}'.format(grid_result.best_score_,grid_result.best_param
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
  print('{},{} with: {}'.format(mean, stdev, param))
```

```
Fitting 5 folds for each of 8748 candidates, totalling 43740 fits
[CV 1/5; 1/8748] START activation_function=softmax, batch_size=10, dropout_ra
te=0.0, epochs=10, init=uniform, learning_rate=0.001, neuron1=4, neuron2=2
[CV 1/5; 1/8748] END activation_function=softmax, batch_size=10, dropout_rate
=0.0, epochs=10, init=uniform, learning_rate=0.001, neuron1=4, neuron2=2; tot
al time=   5.2s
[CV 2/5; 1/8748] START activation_function=softmax, batch_size=10, dropout_ra
te=0.0, epochs=10, init=uniform, learning_rate=0.001, neuron1=4, neuron2=2
[CV 2/5; 1/8748] END activation_function=softmax, batch_size=10, dropout_rate
=0.0, epochs=10, init=uniform, learning_rate=0.001, neuron1=4, neuron2=2; tot
al time=   6.1s
[CV 3/5; 1/8748] START activation_function=softmax, batch_size=10, dropout_ra
te=0.0, epochs=10, init=uniform, learning_rate=0.001, neuron1=4, neuron2=2
[CV 3/5; 1/8748] END activation_function=softmax, batch_size=10, dropout_rate
=0.0, epochs=10, init=uniform, learning_rate=0.001, neuron1=4, neuron2=2; tot
al time=   5.6s
[CV 4/5; 1/8748] START activation_function=softmax, batch_size=10, dropout_ra
te=0.0, epochs=10, init=uniform, learning_rate=0.001, neuron1=4, neuron2=2
[CV 4/5; 1/8748] END activation_function=softmax, batch_size=10, dropout_rate
```

In [ ]: