```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OrdinalEncoder
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
df=pd.read_csv('data (1).csv')
```

```
df.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness
0	842302	M	17.99	10.38	122.80	1001.0	0
1	842517	M	20.57	17.77	132.90	1326.0	0
2	84300903	M	19.69	21.25	130.00	1203.0	0
3	84348301	M	11.42	20.38	77.58	386.1	0
4	84358402	M	20.29	14.34	135.10	1297.0	0

5 rows × 33 columns

```
df.info()
```

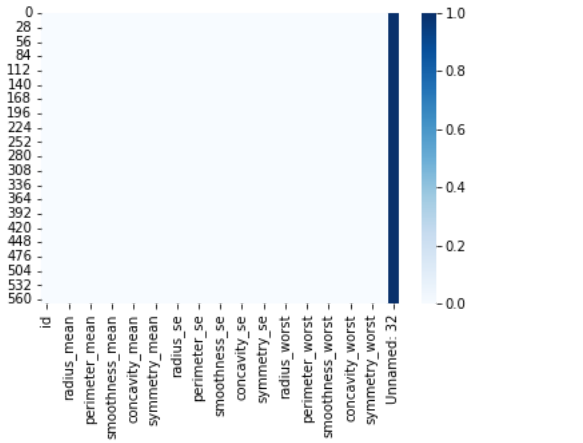
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

```
df.isnull().sum()
```

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
Unnamed: 32              569
dtype: int64
```

```
sns.heatmap(df.isnull(),cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f424b352ad0>
```




```
df.drop('Unnamed: 32', axis=1, inplace=True)
```

```
df
```

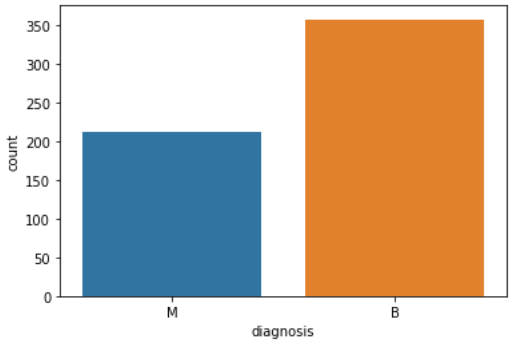
	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	sm
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	

```
df["diagnosis"].value_counts()
```

```
B    357
M    212
Name: diagnosis, dtype: int64
```

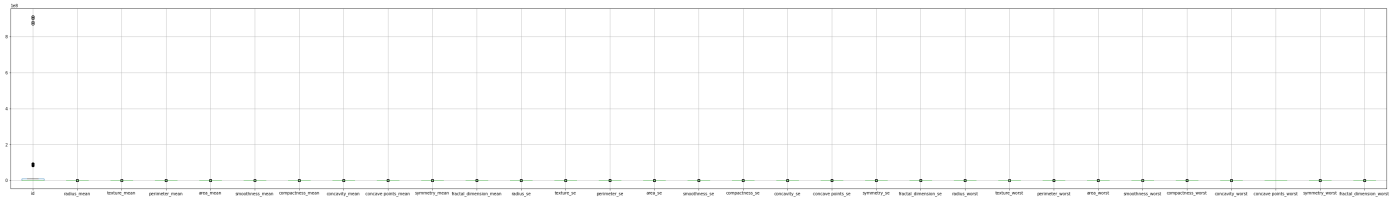
```
cnt_plot= sns.countplot(df["diagnosis"],label="Count")
B, M = df["diagnosis"].value_counts()
print('Number of Benign: ',B)
print('Number of Malignant : ',M)
```

```
Number of Benign:  357
Number of Malignant :  212
```



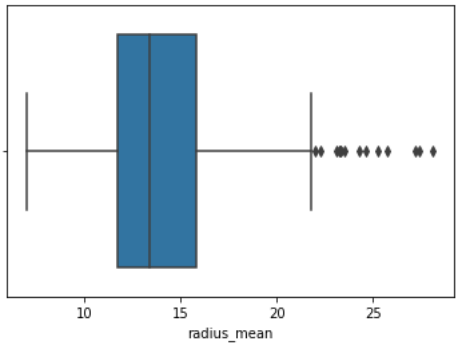

```
plt.figure(figsize=(60,8))
df.boxplot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f424b13ba90>
```



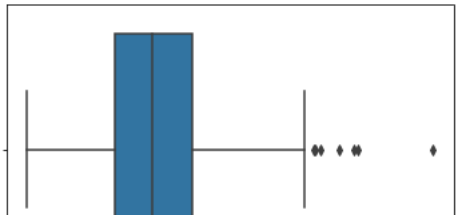
```
sns.boxplot(df['radius_mean'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f424ad8e950>
```



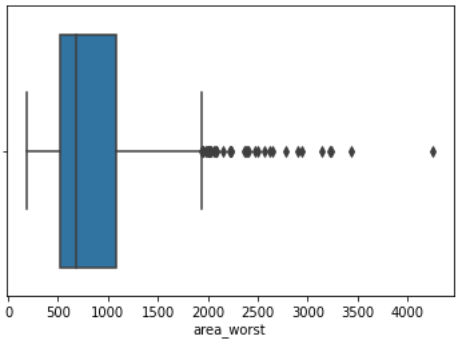

```
sns.boxplot(df['texture_mean'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f424ad69c10>
```



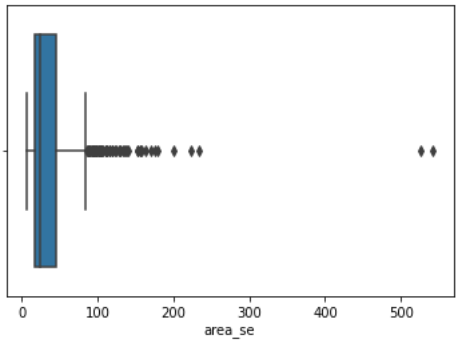
```
sns.boxplot(df['area_worst'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f424accfc90>
```




```
sns.boxplot(df['area_se'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f424ac49810>
```




```
df = df[(df['radius_mean'] < 23) & (df['texture_mean'] < 35) & (df['area_worst'] < 2300) & (df['area_se'] < 150)]
```

```
df
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	c point
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0
5	843786	M	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780	0
...	...	...	...	...	...	...	...	...	...	
563	926125	M	20.92	25.09	143.00	1347.0	0.10990	0.22360	0.31740	0
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0

546 rows × 32 columns

```
df.drop("id",axis=1,inplace=True)
```

```
df["diagnosis"]=df.diagnosis.replace({'B':0,'M':1})
```

```
corr = df.corr()
```

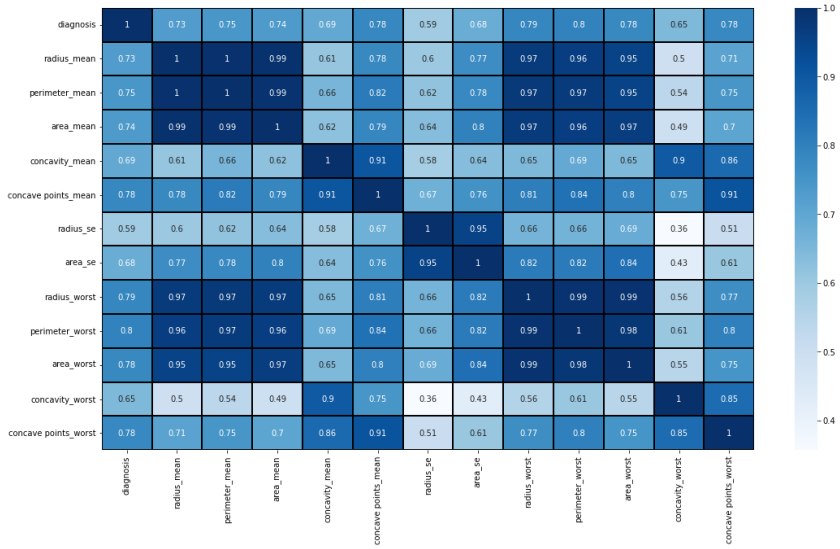
```
corr[abs(corr['diagnosis']) > 0.59].index
```

```
Index(['diagnosis', 'radius_mean', 'perimeter_mean', 'area_mean',
       'concavity_mean', 'concave points_mean', 'radius_se', 'area_se',
       'radius_worst', 'perimeter_worst', 'area_worst', 'concavity_worst',
       'concave points_worst'],
      dtype='object')
```

```
df=df[['diagnosis', 'radius_mean', 'perimeter_mean', 'area_mean',
       'concavity_mean', 'concave points_mean', 'radius_se', 'area_se',
       'radius_worst', 'perimeter_worst', 'area_worst', 'concavity_worst',
       'concave points_worst']]
```

```
plt.figure(figsize = (18, 10))
sns.heatmap(df.corr(),cmap='Blues',linewidths=1,linecolor='black',annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f424b25cd10>
```




```
x=df.drop(['diagnosis'],axis=1)
x
```

	radius_mean	perimeter_mean	area_mean	concavity_mean	concave points_mean	radius_s
1	20.57	132.90	1326.0	0.08690	0.07017	0.543
2	19.69	130.00	1203.0	0.19740	0.12790	0.745
3	11.42	77.58	386.1	0.24140	0.10520	0.495
4	20.29	135.10	1297.0	0.19800	0.10430	0.757
5	12.45	82.57	477.1	0.15780	0.08089	0.334
...	...	...	...	...	...	.

```
y=df.diagnosis
y
```

```
1      1
2      1
3      1
4      1
5      1
      ..
563    1
565    1
566    1
567    1
568    0
Name: diagnosis, Length: 546, dtype: int64
```

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=1)


from sklearn.preprocessing import StandardScaler
sc=StandardScaler()

xtrain=sc.fit_transform(xtrain)
xtest=sc.transform(xtest)
# xtrain = (xtrain-xtrain.mean())/(xtrain.max()-xtrain.min())
# xtest = (xtest-xtest.mean())/(xtest.max()-xtest.min())


from tensorflow.keras.callbacks import EarlyStopping
early_stop = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=25)


model = Sequential()

model.add(Dense(20,activation='relu'))
model.add(Dense(20,activation='relu'))
model.add(Dense(1,activation='sigmoid')) #Output layer (Since it's a binary classification problem)

#Using accuracy as loss function
model.compile(optimizer='sgd',loss='binary_crossentropy',metrics=['accuracy'])

model.fit(xtrain,ytrain,epochs=200,validation_data=(xtest, ytest),verbose=1,batch_size=128,callbacks=[early_stop])
```

```
Epoch 179/200
3/3 [==============================] - 0s 17ms/step - loss: 0.1542 - accuracy: 0.9476 - val_loss: 0.1262 - val_accuracy: 0.9451
Epoch 180/200
3/3 [==============================] - 0s 16ms/step - loss: 0.1540 - accuracy: 0.9476 - val_loss: 0.1260 - val_accuracy: 0.9451
Epoch 181/200
3/3 [==============================] - 0s 14ms/step - loss: 0.1537 - accuracy: 0.9476 - val_loss: 0.1258 - val_accuracy: 0.9451
Epoch 182/200
3/3 [==============================] - 0s 16ms/step - loss: 0.1535 - accuracy: 0.9476 - val_loss: 0.1255 - val_accuracy: 0.9451
Epoch 183/200
3/3 [==============================] - 0s 14ms/step - loss: 0.1533 - accuracy: 0.9476 - val_loss: 0.1253 - val_accuracy: 0.9451
Epoch 184/200
3/3 [==============================] - 0s 17ms/step - loss: 0.1530 - accuracy: 0.9476 - val_loss: 0.1251 - val_accuracy: 0.9451
Epoch 185/200
3/3 [==============================] - 0s 15ms/step - loss: 0.1528 - accuracy: 0.9476 - val_loss: 0.1249 - val_accuracy: 0.9451
Epoch 186/200
3/3 [==============================] - 0s 16ms/step - loss: 0.1525 - accuracy: 0.9476 - val_loss: 0.1247 - val_accuracy: 0.9451
Epoch 187/200
3/3 [==============================] - 0s 18ms/step - loss: 0.1523 - accuracy: 0.9476 - val_loss: 0.1245 - val_accuracy: 0.9451
Epoch 188/200
3/3 [==============================] - 0s 15ms/step - loss: 0.1521 - accuracy: 0.9476 - val_loss: 0.1243 - val_accuracy: 0.9451
Epoch 189/200
3/3 [==============================] - 0s 24ms/step - loss: 0.1519 - accuracy: 0.9476 - val_loss: 0.1241 - val_accuracy: 0.9451
Epoch 190/200
3/3 [==============================] - 0s 15ms/step - loss: 0.1517 - accuracy: 0.9476 - val_loss: 0.1239 - val_accuracy: 0.9451
Epoch 191/200
3/3 [==============================] - 0s 20ms/step - loss: 0.1515 - accuracy: 0.9476 - val_loss: 0.1238 - val_accuracy: 0.9451
Epoch 192/200
3/3 [==============================] - 0s 17ms/step - loss: 0.1513 - accuracy: 0.9476 - val_loss: 0.1236 - val_accuracy: 0.9451
Epoch 193/200
3/3 [==============================] - 0s 16ms/step - loss: 0.1511 - accuracy: 0.9476 - val_loss: 0.1234 - val_accuracy: 0.9451
Epoch 194/200
3/3 [==============================] - 0s 16ms/step - loss: 0.1508 - accuracy: 0.9476 - val_loss: 0.1232 - val_accuracy: 0.9451
Epoch 195/200
3/3 [==============================] - 0s 15ms/step - loss: 0.1507 - accuracy: 0.9476 - val_loss: 0.1231 - val_accuracy: 0.9451
Epoch 196/200
3/3 [==============================] - 0s 16ms/step - loss: 0.1505 - accuracy: 0.9476 - val_loss: 0.1229 - val_accuracy: 0.9451
Epoch 197/200
3/3 [==============================] - 0s 22ms/step - loss: 0.1503 - accuracy: 0.9476 - val_loss: 0.1227 - val_accuracy: 0.9451
Epoch 198/200
3/3 [==============================] - 0s 16ms/step - loss: 0.1501 - accuracy: 0.9476 - val_loss: 0.1226 - val_accuracy: 0.9451
```

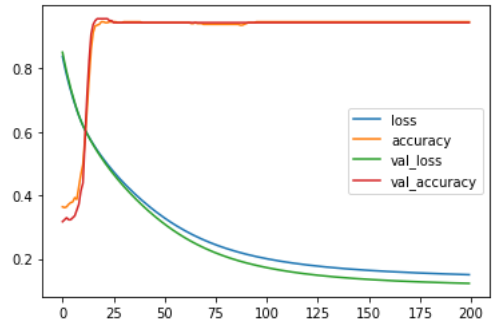
```
model.history.history
```

```
{'accuracy': [0.3638743460178375,
  0.36125653982162476,
  0.3638743460178375,
  0.37172773480415344,
  0.376963347196579,
  0.37958115339279175,
  0.3926701545715332,
  0.38743454217910767,
  0.4267015755176544,
  0.4659685790538788,
  0.49738219380378723,
  0.5785340070724487,
  0.6701570749282837,
  0.7696335315704346,
  0.8612565398216248,
  0.9136125445365906,
  0.9345549941062927,
  0.9371727705001831,
  0.9397905468940735,
  0.9476439952850342,
  0.9476439952850342,
  0.945026159286499,
  0.945026159286499,
  0.9476439952850342,
  0.9476439952850342,
  0.9476439952850342,
  0.9476439952850342,
  0.945026159286499,
  0.945026159286499,
  0.945026159286499,
  0.9476439952850342,
  0.9476439952850342,
  0.9476439952850342,
  0.9476439952850342,
  0.9476439952850342,
  0.9476439952850342,
  0.9476439952850342,
  0.9476439952850342,
  0.9476439952850342,
  0.945026159286499,
```

```
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
        0.945026159286499,
```

```
lossdf=pd.DataFrame(model.history.history)
lossdf.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f42499fd110>
```




```
ypred=model.predict(xtest)
ypred=ypred>0.5
```

```
from sklearn.metrics import classification_report
print(classification_report(ytest,ypred))
```

```
                  precision    recall  f1-score   support

               0       0.97      0.95      0.96       112
               1       0.89      0.94      0.92        52

        accuracy                           0.95       164
       macro avg       0.93      0.94      0.94       164
    weighted avg       0.95      0.95      0.95       164
```

```
# from tensorflow.math import confusion_matrix
# ypred = model.predict(xtest)
# conf_matrix = confusion_matrix(ytest,ypred)
# cm = sns.heatmap(conf_matrix, annot=True, cmap='gray', annot_kws={'size':30})
# cm_labels = ['Negative','Positive']
# cm.set_xlabel('True')
# cm.set_xticklabels(cm_labels)
# cm.set_ylabel('Predicted')
# cm.set_yticklabels(cm_labels);
```

```
# from sklearn.metrics import confusion_matrix
# confusion_matrix(ytest,ypred)
```