The goal of this analysis is to train a machine learning algorightms to accurately distinguish between a benign and malignant tumor to aid in clinical diagnosis.

Important features of dataset

- a) radius (mean of distances from center to points on the perimeter)
- b) texture (standard deviation of gray-scale values)
- c) perimeter
- d) area
- e) smoothness (local variation in radius lengths)
- f) compactness (perimeter^2 / area - 1.0)
- g) concavity (severity of concave portions of the contour)
- h) concave points (number of concave portions of the contour)
- i) symmetry
- j) fractal dimension ("coastline approximation" - 1)

In [ ]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
```

In [ ]:

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

In [ ]:

```
# delete unnecessary columns
df = df.drop(columns=['id', 'Unnamed: 32'], axis=1)
```

In [ ]:

```
# statistical info
df.describe()
```

In [ ]:

```
# datatype info
df.info()
```

In [ ]:

```
df.shape
```

In [ ]:

```
df.isna().sum()
```

In [ ]:

```
df2=df.drop(['diagnosis'],axis=1,inplace=False)
```

In [ ]:

```
df.info()
```

In [ ]:

```
from sklearn.model_selection import train_test_split
X = df.drop(columns=['diagnosis'])
df['diagnosis'] = np.where(df['diagnosis']=='M', 1, 0)
Y = df['diagnosis']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.30)
```

In [ ]:

```
# The normalize features to account for feature scaling
from sklearn.preprocessing import Normalizer
# Instantiate
norm = Normalizer()

# Fit
norm.fit(x_train)

# Transform both training and testing sets
X_train_norm = norm.transform(x_train)
X_test_norm = norm.transform(x_test)
```

In [ ]:

```
# logistic regression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [ ]:

```
# model training
model.fit(X_train_norm, y_train)
```

In [ ]:

```
# print metric to get performance
print("Accuracy: ",model.score(X_test_norm, y_test) * 100)
```

In [ ]:

```
# decision tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report,f1_score

model = DecisionTreeClassifier()
```

In [ ]:

```
model.fit(X_train_norm, y_train)
```

In [ ]:

```
# print metric to get performance
print("Accuracy: ",model.score(X_test_norm, y_test) * 100)
```

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,f1_score
model = RandomForestClassifier()
# training
model.fit(X_train_norm, y_train)
# testing
y_pred = model.predict(X_test_norm)
print(classification_report(y_test, y_pred))
#print(f1_score(y_test, y_pred))
```

In [ ]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.utils.vis_utils import plot_model
```

In [ ]:

```
Model = Sequential()
```

In [ ]:

```

```

In [ ]:

```
Model.add(Dense(units= 16, activation = 'relu', input_dim=30))
# units indicates here the ooutput dimention of Dense laye
Model.add(Dense(units=8, activation='relu'))
Model.add(Dense(units=6, activation='relu'))
Model.add(Dense(units=1, activation='sigmoid'))
```

In [ ]:

```
Model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
```

In [ ]:

```
Model.summary()
```

In [ ]:

```
plot_model(Model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

In [ ]:

```
history=Model.fit(X_train_norm, y_train,validation_split=0.2, batch_size=1, epochs=50)
```

In [ ]:

```
prediction = Model.predict(X_test_norm[1:])
```

In [ ]:

```
prediction
```

In [ ]:

```

```