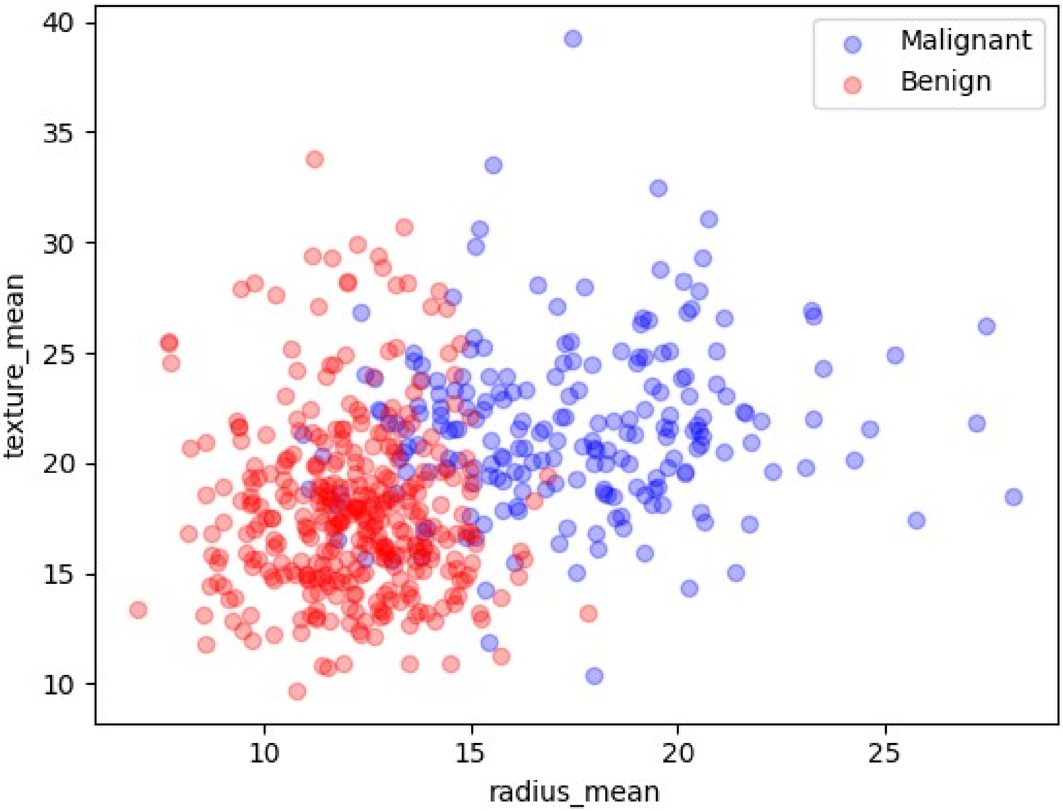
```
import pandas as pd  #Import necessary libraries and modules
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

df = pd.read_csv('/content/Cancer_DS.csv')

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['diagnosis'].unique()
```

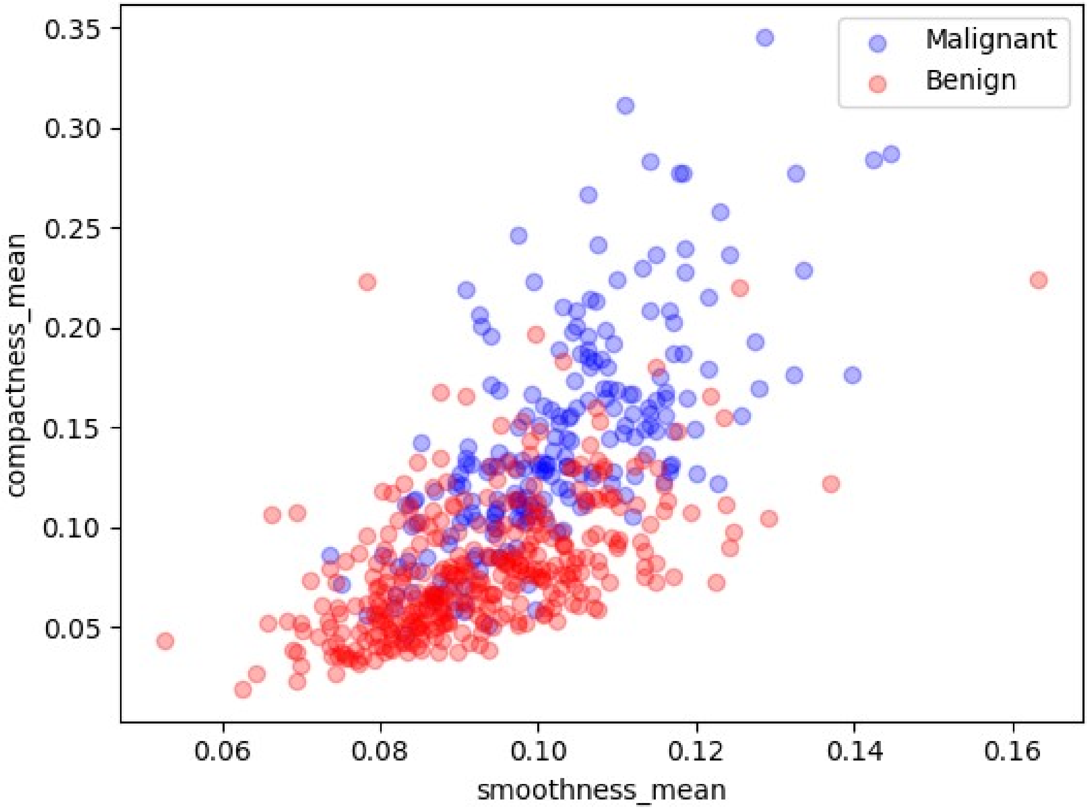
```
array(['M', 'B'], dtype=object)
```

```
import matplotlib.pyplot as plt
M = df[df.diagnosis == "M"]
B = df[df.diagnosis == "B"]

plt.scatter(M.radius_mean,M.texture_mean, color = "blue", label =
"Malignant", alpha = 0.3)
plt.scatter(B.radius_mean,B.texture_mean, color = "red", label =
"Benign", alpha = 0.3)

plt.xlabel("radius_mean")
plt.ylabel("texture_mean")

plt.legend()
plt.show()
```



```
plt.scatter(M.smoothness_mean,M.compactness_mean, color = "blue",
label = "Malignant", alpha = 0.3)
plt.scatter(B.smoothness_mean,B.compactness_mean, color = "red", label
= "Benign", alpha = 0.3)
```

```
plt.xlabel("smoothness_mean")
plt.ylabel("compactness_mean")

plt.legend()
plt.show()
```



```
df['diagnosis'].unique() #This code extracts and displays the unique
values found in the 'diagnosis' column of the DataFrame 'df'.

array(['M', 'B'], dtype=object)
```

The output shows that the 'diagnosis' column contains two unique values, 'M' and 'B'.

```
from sklearn import preprocessing #The code uses label encoding to
transform the 'diagnosis' column, replacing 'B' with 0 and 'M' with 1.
label_encoder = preprocessing.LabelEncoder()

df["diagnosis"] = label_encoder.fit_transform(df['diagnosis']) # 0 -
B, 1 - M
df['diagnosis']

0       1
1       1
2       1
```

```
3      1
4      1
      ..
564    1
565    1
566    1
567    1
568    0
Name: diagnosis, Length: 569, dtype: int64
```

The 'diagnosis' column has been successfully transformed using label encoding, where 'B' is replaced with 0 and 'M' with 1, resulting in a new column with values of 0 or 1 for each entry.

```
train, test = train_test_split(df, test_size = 0.3) #The code splits the dataset into training and testing sets, extracting feature sets and labels for both.

trainX = train[df.columns[2:-1]]
trainY=train[df.columns[1]]

testX= test[df.columns[2:-1]]
testY =test[df.columns[1]]

from sklearn import metrics
#This code uses a k-Nearest Neighbors classifier with k=3 to predict labels for the test set and calculates the accuracy of the predictions.
c_knn = KNeighborsClassifier(n_neighbors=3)
c_knn.fit(trainX.values,trainY.values)

y_pred = c_knn.predict(testX.values)
print("Accuracy : ",metrics.accuracy_score(testY.values,y_pred))

Accuracy :  0.9005847953216374
```

The accuracy of the k-Nearest Neighbors classifier with k=3 on the test set is approximately 92.40%.

```
sample = [testX.values[170]] #The code classifies a single test sample as "Benign" or "Malignant" based on the prediction made by a k-Nearest Neighbors classifier.
pred = c_knn.predict(sample)
res=pred[0]
if res==0:
  print('Benign')
if res==1:
  print('Malignant')

Benign
```

The code predicts that the test sample is "Malignant."