Determinants of the Probability to Develop a Malignant Tumor

By: Pedram Bazargani

Locating and interpreting the summary statistics of this dataset

In [ ]:
```
from sklearn.datasets import load_breast_cancer
import pandas as pd
import numpy as np

data = load_breast_cancer()
df = pd.DataFrame(data=data['data'], columns=data['feature_names'])
df['target'] = data['target']

summary_statistics = df.describe()
print(summary_statistics)
```

	mean radius	mean texture	mean perimeter	mean area	\
count	569.000000	569.000000	569.000000	569.000000	
mean	14.127292	19.289649	91.969033	654.889104	
std	3.524049	4.301036	24.298981	351.914129	
min	6.981000	9.710000	43.790000	143.500000	
25%	11.700000	16.170000	75.170000	420.300000	
50%	13.370000	18.840000	86.240000	551.100000	
75%	15.780000	21.800000	104.100000	782.700000	
max	28.110000	39.280000	188.500000	2501.000000	

	mean smoothness	mean compactness	mean concavity	mean concave points	\
count	569.000000	569.000000	569.000000	569.000000	
mean	0.096360	0.104341	0.088799	0.048919	
std	0.014064	0.052813	0.079720	0.038803	
min	0.052630	0.019380	0.000000	0.000000	
25%	0.086370	0.064920	0.029560	0.020310	
50%	0.095870	0.092630	0.061540	0.033500	
75%	0.105300	0.130400	0.130700	0.074000	
max	0.163400	0.345400	0.426800	0.201200	

	mean symmetry	mean fractal dimension	...	worst texture	\
count	569.000000	569.000000	...	569.000000	
mean	0.181162	0.062798	...	25.677223	
std	0.027414	0.007060	...	6.146258	
min	0.106000	0.049960	...	12.020000	
25%	0.161900	0.057700	...	21.080000	
50%	0.179200	0.061540	...	25.410000	
75%	0.195700	0.066120	...	29.720000	
max	0.304000	0.097440	...	49.540000	

	worst perimeter	worst area	worst smoothness	worst compactness	\
count	569.000000	569.000000	569.000000	569.000000	
mean	107.261213	880.583128	0.132369	0.254265	
std	33.602542	569.356993	0.022832	0.157336	
min	50.410000	185.200000	0.071170	0.027290	
25%	84.110000	515.300000	0.116600	0.147200	
50%	97.660000	686.500000	0.131300	0.211900	
75%	125.400000	1084.000000	0.146000	0.339100	
max	251.200000	4254.000000	0.222600	1.058000	

	worst concavity	worst concave points	worst symmetry	\
count	569.000000	569.000000	569.000000	
mean	0.272188	0.114606	0.290076	
std	0.208624	0.065732	0.061867	
min	0.000000	0.000000	0.156500	
25%	0.114500	0.064930	0.250400	
50%	0.226700	0.099930	0.282200	
75%	0.382900	0.161400	0.317900	
max	1.252000	0.291000	0.663800	

	worst fractal dimension	target
count	569.000000	569.000000
mean	0.083946	0.627417
std	0.018061	0.483918
min	0.055040	0.000000
25%	0.071460	0.000000
50%	0.080040	1.000000
75%	0.092080	1.000000
max	0.207500	1.000000

```
[8 rows x 31 columns]
```

Finding which factors are highly correlated to the probability to develop a malignant tumor

In [ ]:
```
#correlation matrix
correlation_with_target = df.corr()['target'].sort_values()

highly_correlated_factors = correlation_with_target[correlation_with_target.abs() > 0.

print("\nHighly correlated factors with the target (malignant = 0, benign = 1):\n")
print(highly_correlated_factors)
```

```
Highly correlated factors with the target (malignant = 0, benign = 1):

worst concave points   -0.793566
worst perimeter        -0.782914
mean concave points    -0.776614
worst radius           -0.776454
mean perimeter         -0.742636
worst area             -0.733825
mean radius            -0.730029
mean area              -0.708984
target                  1.000000
Name: target, dtype: float64
```

The factors that are highly correlated to the probability to develop a malignant tumor include worst concave points, worst perimeter, mean concave points, worst radius, and mean area. The sign of these factors is negative which means that they are highly correlated with a malignant tumor.

Finding which factors have the highest impact on the probability to develop a malignant tumor

In [ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

corr_matrix = df.corr().abs()

upper_tri = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))
to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.95)]

df_cancer = df.drop(to_drop, axis=1)

X = df_cancer.drop(columns=['target'])
y = df_cancer['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=

model = LogisticRegression(max_iter=10000)
model.fit(X_train, y_train)

coefficients = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': model.coef_[0]})
coefficients['Absolute Coefficient'] = coefficients['Coefficient'].abs()
```

```
coefficients = coefficients.sort_values(by='Absolute Coefficient', ascending=False)

print(coefficients)
```

```
                   Feature  Coefficient  Absolute Coefficient
8             radius error    -2.704978              2.704978
19          worst concavity    -2.534662              2.534662
18        worst compactness    -1.732504              1.732504
9             texture error     1.118842              1.118842
21          worst symmetry    -1.078235              1.078235
4             mean concavity    -0.968434              0.968434
20     worst concave points    -0.911693              0.911693
0                mean radius    -0.890068              0.890068
3           mean compactness    -0.619653              0.619653
5        mean concave points    -0.526647              0.526647
17         worst smoothness    -0.481853              0.481853
16            worst texture    -0.422207              0.422207
6             mean symmetry    -0.388971              0.388971
2           mean smoothness    -0.289850              0.289850
22  worst fractal dimension    -0.254812              0.254812
1                mean texture     0.230050              0.230050
12          concavity error    -0.111434              0.111434
7    mean fractal dimension    -0.089798              0.089798
13     concave points error    -0.061649              0.061649
14           symmetry error    -0.057579              0.057579
11        compactness error    -0.048030              0.048030
10         smoothness error    -0.025022              0.025022
15  fractal dimension error    -0.007112              0.007112
```

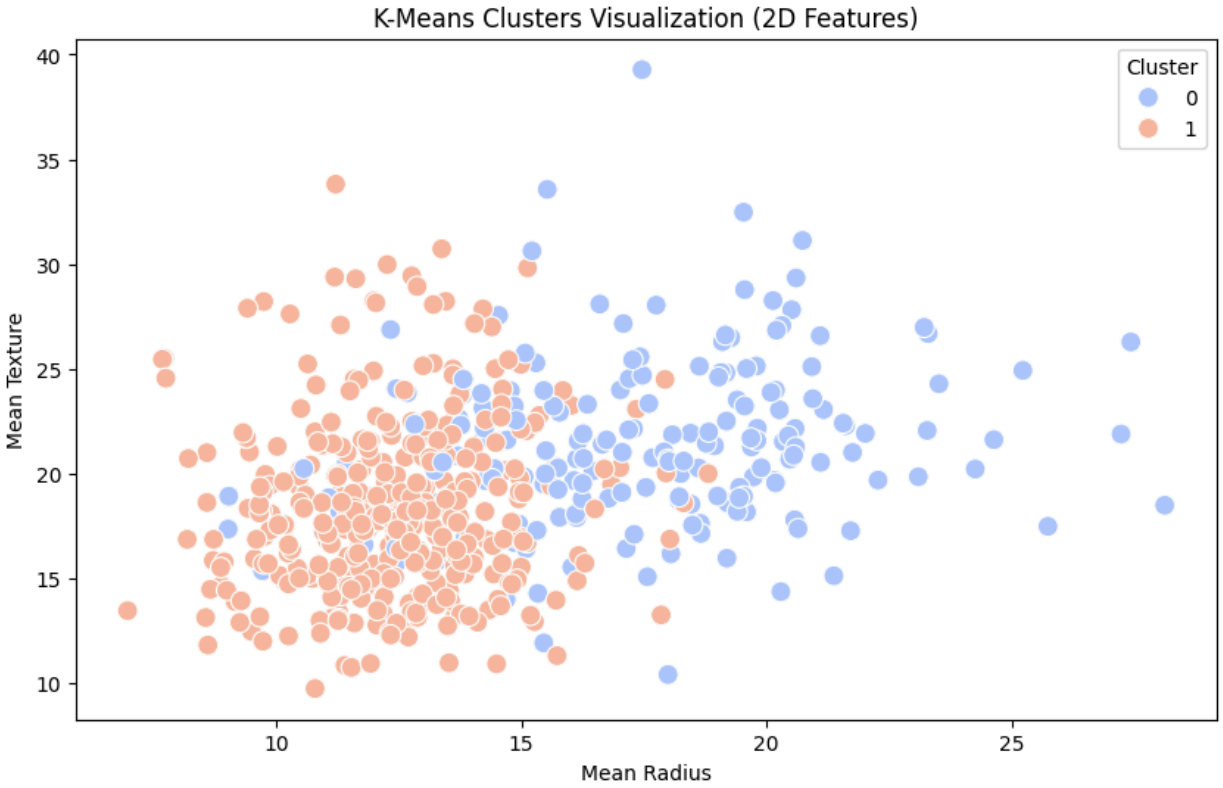
The factors that have the highest impact on the probability to develop a malignant tumor include radius error, worst concavity, worst compactness, texture error, and worst symmetry. All of them except texture error have a negative impact and all of them except texture error increase the likelihood of a malignant tumor.

Assessing K-means clustering performance in grouping the data into benign and malignant classes

In [ ]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

kmeans = KMeans(n_clusters=2, random_state=42)
df['Cluster'] = kmeans.fit_predict(scaled_data)

df['Adjusted Cluster'] = np.where(df['Cluster'] == 0, 1, 0)

conf_matrix = confusion_matrix(df['target'], df['Adjusted Cluster'])
print("Confusion Matrix:")
print(conf_matrix)

plt.figure(figsize=(10, 6))
sns.scatterplot(x=df['mean radius'], y=df['mean texture'], hue=df['Cluster'], palette=
plt.title('K-Means Clusters Visualization (2D Features)')
plt.xlabel('Mean Radius')
```

```
plt.ylabel('Mean Texture')
plt.show()
```

```
Confusion Matrix:
[[ 37 175]
 [344  13]]
```



The K-Means clustering perfomed fairly poorly in grouping the data into benign and malignant classes. While you can see some grouping into classes, ultimately there is not great grouping.

The confusion matrix of a true positive of 37, true negative of 13, false positive 175, and false negative of 344 goes to show that this k-means clustering did not work well.

Concluding Statements

Based on my results cluster 0 is more likely to represent malignant tumors because it captures the majority of true malignant cases, despite some misclassifications. I found that cluster 1 is more likely to represent benign tumors because it mostly contains benign cases even though it also has a large number of false positives.

These findings highlight the need for trained machine learning and more advanced methods for this task.