

# BREAST CANCER PRIDITION

BY GROUP 5:  
VŨ ĐỨC ANH  
VŨ VĨNH THÁI  
NGUYỄN QUANG HIỆP

Ung thư vú hiện đang là một trong những căn bệnh nguy hiểm hàng đầu đối với phụ nữ trên toàn thế giới. Việc phát hiện sớm và điều trị kịp thời có thể cứu sống hàng triệu người, nhưng không phải lúc nào quá trình chẩn đoán cũng dễ dàng. Trong những năm gần đây, sự phát triển của công nghệ học máy (machine learning) đã mang lại một bước tiến mới trong việc hỗ trợ chẩn đoán ung thư vú. Hôm nay, chúng ta sẽ cùng tìm hiểu về cách học máy đang thay đổi cuộc chiến chống lại căn bệnh này, từ việc tăng độ chính xác trong chẩn đoán đến hỗ trợ ra quyết định lâm sàng một cách hiệu quả hơn.



# Các bước thực hiện

- 1 Importing Libraries:** Nạp các thư viện cần thiết để xử lý và phân tích dữ liệu.
- 2 DATA PREPROCESSING:** Xử lý dữ liệu, bao gồm làm sạch và chuẩn hóa.
- 3 missing value:** Xử lý các giá trị bị thiếu trong dữ liệu.
- 4 EDA (Exploratory Data Analysis):** Phân tích dữ liệu khám phá để hiểu rõ hơn về đặc điểm của dữ liệu.
- 5 CORR (Correlation Analysis):** Phân tích tương quan để xác định mối quan hệ giữa các biến.
- 6 Building Model:** Xây dựng các mô hình học máy dựa trên dữ liệu đã xử lý.
- 7 Apply Machine Learning Algorithms:** Thử nghiệm nhiều thuật toán học máy khác nhau.
- 8 Model Comparison:** So sánh các mô hình để xác định mô hình tối ưu.

# Importing Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno
```

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

```
plt.style.use('ggplot')
```

0.0s

```
df = pd.read_csv("breast_cancer.csv")
```

0.2s



	id	diagnosis	radius_mean	texture_mean	peri
0	842302	M	17.99	10.38	
1	842517	M	20.57	17.77	
2	84300903	M	19.69	21.25	
3	84348301	M	11.42	20.38	
4	84358402	M	20.29	14.34	
			smoothness_mean	compactness_mean	concavity_mean
0			0.11840	0.27760	0.3001
1			0.08474	0.07864	0.0869
2			0.10960	0.15990	0.1974
3			0.14250	0.28390	0.2414
4			0.10030	0.13280	0.1980

# DATA PREPROCESSING

```
df.diagnosis.unique()
```

✓ 0.0s

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

```
# Supervised-> target
```

```
# Unsupervised
```

```
df.describe()
```

✓ 0.0s

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.016199
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.004434
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.002608
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.005353
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.006654
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.008463

# missing value

```
df.info()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 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
...
30  symmetry_worst         569 non-null    float64
31  fractal_dimension_worst 569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

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

✓ 0.0s

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

# EDA (Exploratory Data Analysis)

```
# each 5 row its having 6 columns
```

```
# density graph
```

```
plt.figure(figsize=(20,15))
```

```
plotnumber=1
```

```
for column in df:
```

```
    if plotnumber<=30:
```

```
        ax = plt.subplot(5,6, plotnumber)
```

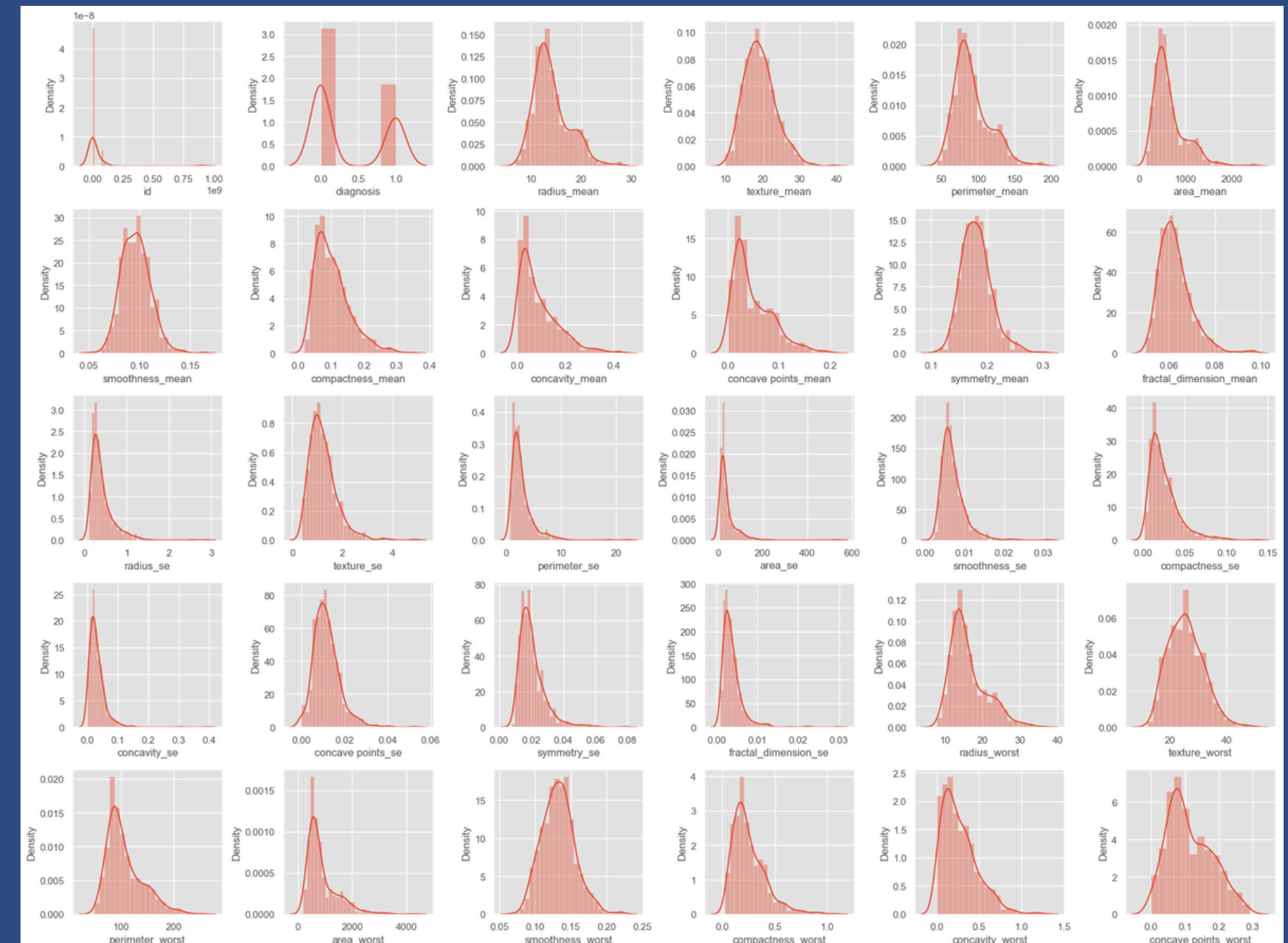
```
        sns.distplot(df[column])
```

```
        plt.xlabel(column)
```

```
        plotnumber+=1
```

```
plt.tight_layout()
```

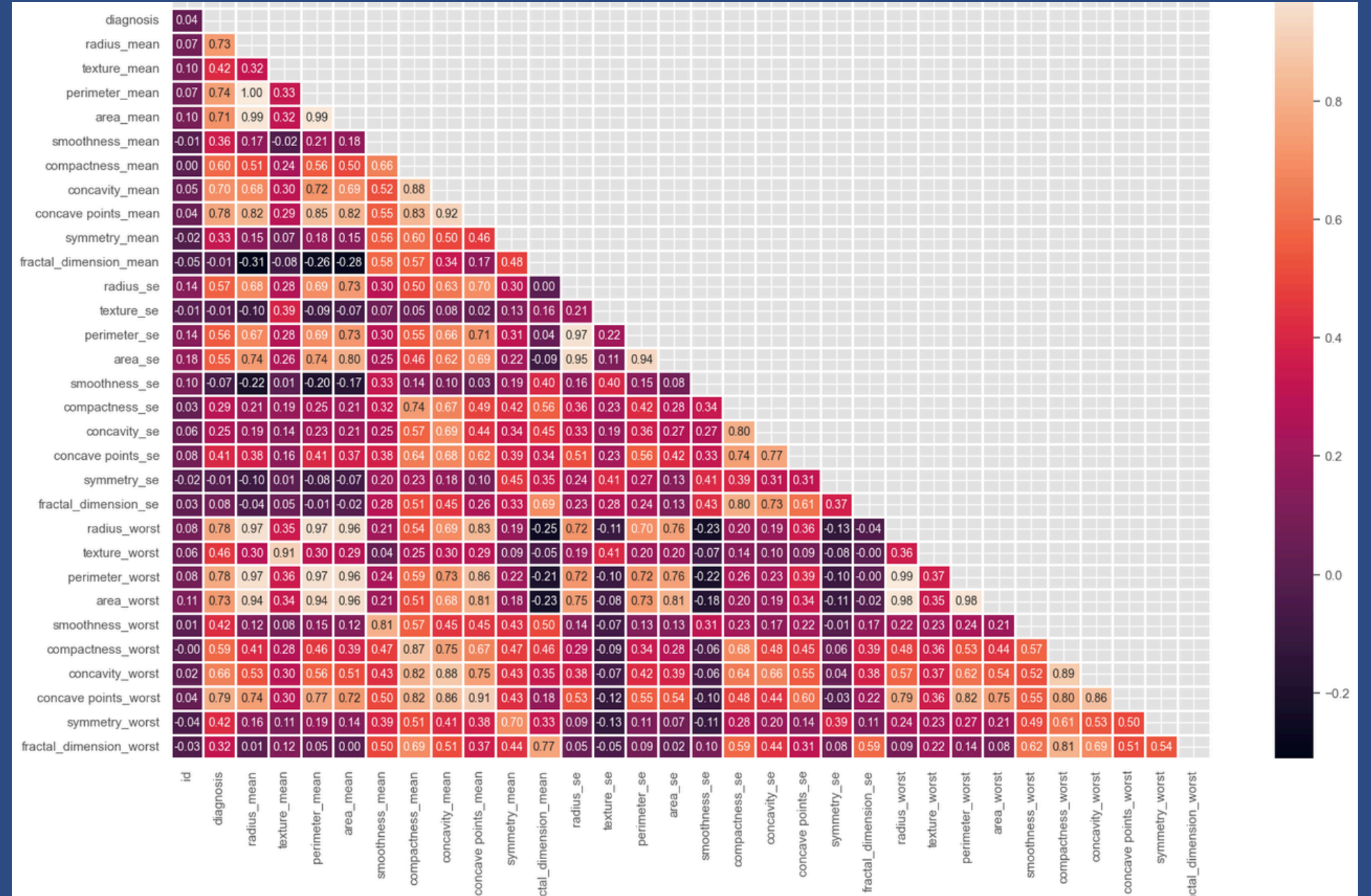
```
plt.show()
```





# CORR (Correlation Analysis)

```
# heatmap
plt.figure(figsize=(20,12))
corr=df.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
sns.heatmap(corr, mask=mask, linewidths=1, annot=True, fmt = ".2f")
plt.show()
```





# Building Model

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

✓ 0.0s

```
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

✓ 0.0s

```
# scaling data  
from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler()  
X_train = scaler.fit_transform(X_train)  
X_test = scaler.transform(X_test)
```

✓ 0.0s

```
X_train.shape
```

✓ 0.0s

# Apply Machine Learning Algorithms

## Thuật toán KNN

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(X_train, y_train)

✓ 0.0s
```

KNeighborsClassifier ⓘ ?

KNeighborsClassifier()

```
y_pred = knn.predict(X_test)

✓ 0.0s
```

2

```
y_pred
✓ 0.0s

array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
       1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,
       0, 1, 1, 0])
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
print(accuracy_score(y_train, knn.predict(X_train)))
knn_acc = accuracy_score(y_test, knn.predict(X_test))
print(knn_acc)
y_pred = knn.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

✓ 0.0s
```

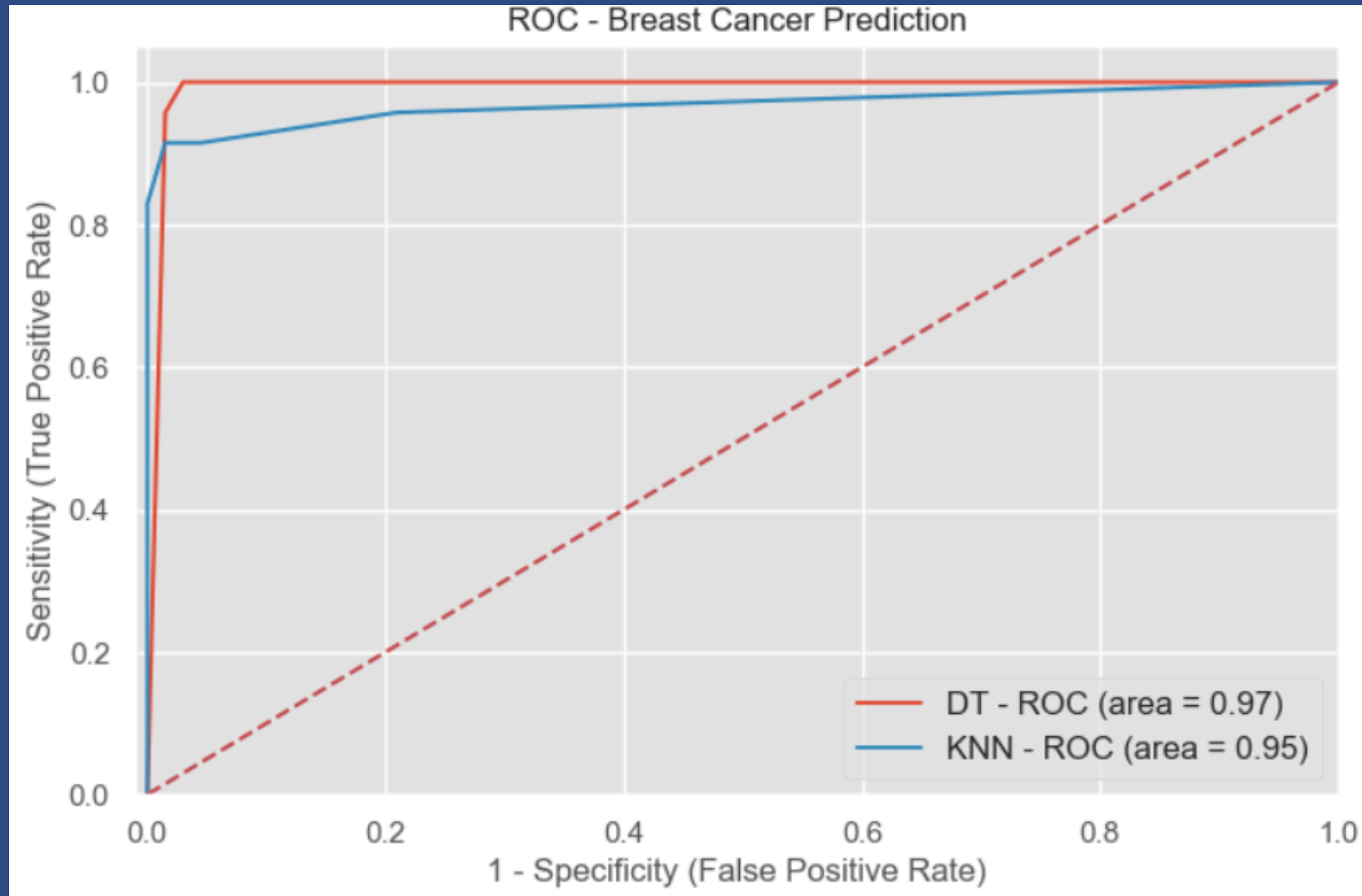
0.967032967032967  
0.956140350877193

```
[[66  1]
 [ 4 43]]
```

	precision	recall	f1-score	support
0	0.94	0.99	0.96	67
1	0.98	0.91	0.95	47
accuracy			0.96	114
macro avg	0.96	0.95	0.95	114
weighted avg	0.96	0.96	0.96	114

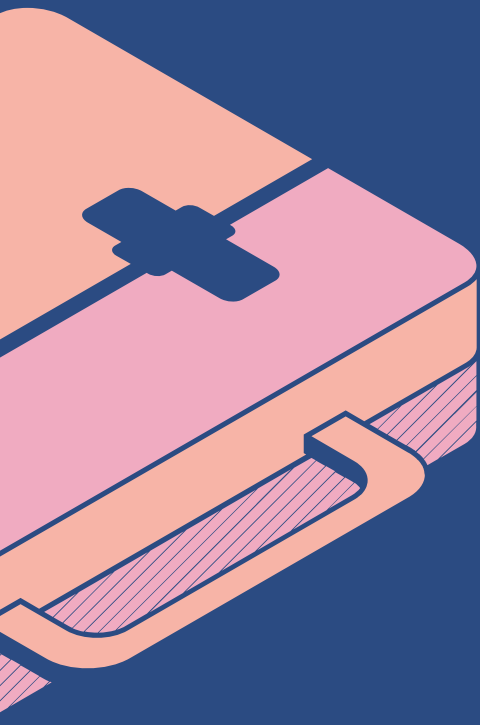
3

# Model Comparison





## Kết luận



Qua quá trình thực hiện các bước phân tích và đánh giá mô hình, chúng ta nhận thấy rằng cả hai thuật toán K-Nearest Neighbors (KNN) và Decision Tree (DT) đều cho kết quả chẩn đoán tốt. Tuy nhiên, thuật toán Decision Tree thể hiện độ chính xác cao hơn so với KNN. Sự khác biệt này có thể đến từ khả năng của DT trong việc nắm bắt cấu trúc dữ liệu và xử lý các đặc trưng một cách hiệu quả hơn. Điều này cho thấy rằng việc lựa chọn thuật toán phù hợp rất quan trọng trong việc phát triển các hệ thống chẩn đoán, nhằm đạt được độ chính xác tối ưu trong quá trình phân loại.

**THANK YOU FOR WATCHING**