```
#perform ETL
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer

# Step 1: Extract
# Assuming the dataset is saved as 'breast_cancer.csv' in the working
data = pd.read_csv('/content/drive/MyDrive/Cancer_Data.csv')

# Step 2: Transform
# 1. Handle missing values for numerical columns
# Exclude non-numeric columns
numerical_columns = data.select_dtypes(include=['number']).columns

# Perform imputation for numerical columns
imputer = SimpleImputer(strategy='mean')
data_imputed_numerical = pd.DataFrame(imputer.fit_transform(data[nume

# Combine imputed numerical columns with non-numerical columns
data_imputed = pd.concat([data['diagnosis'], data_imputed_numerical],

# 2. Convert data types if necessary
# Encode 'diagnosis' to numerical values
data_imputed['diagnosis'] = data_imputed['diagnosis'].map({'B': 0, 'M

# 3. Drop unnecessary columns
# Drop the 'id' column if it exists
if 'id' in data_imputed.columns:
    data_imputed.drop(columns=['id'], inplace=True)

# Drop the 'Unnamed: 32' column if it exists
if 'Unnamed: 32' in data_imputed.columns:
    data_imputed.drop(columns=['Unnamed: 32'], inplace=True)

# 4. Standardize features
# Separate features and target
X = data_imputed.drop(columns=['diagnosis'])
y = data_imputed['diagnosis']

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Create a new DataFrame with scaled features
X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)

# Combine scaled features with target
transformed_data = pd.concat([X_scaled_df, y.reset_index(drop=True)],

# Step 3: Load
# Save the transformed data to a new CSV file
transformed_data.to_csv('/content/drive/MyDrive/transformed_breast_ca

# Display the first few rows of the transformed dataset
transformed_data.head()
```

Cancer_Data.csv

1 to 25 of 569 entries Filter

diagnosis	radius_mean	texture_mean	perime
M	17.99	10.38	122.8
M	20.57	17.77	132.9
M	19.69	21.25	130
M	11.42	20.38	77.58
M	20.29	14.34	135.1
M	12.45	15.7	82.57
M	18.25	19.98	119.6
M	13.71	20.83	90.2
M	13	21.82	87.5
M	12.46	24.04	83.97
M	16.02	23.24	102.7
M	15.78	17.89	103.6
M	19.17	24.8	132.4
M	15.85	23.95	103.7
M	13.73	22.61	93.6
M	14.54	27.54	96.73
M	14.68	20.13	94.74
M	16.13	20.68	108.1
M	19.81	22.15	130
B	13.54	14.36	87.46
B	13.08	15.71	85.63
B	9.504	12.44	60.34
M	15.34	14.26	102.5
M	21.16	23.04	137.2
M	16.65	21.38	110

Show 25 per page

1 2 10 20 23

	radius_mean	texture_mean	perimeter_mean	area_mean	smooth
0	1.097064	-2.073335	1.269934	0.984375	
1	1.829821	-0.353632	1.685955	1.908708	
2	1.579888	0.456187	1.566503	1.558884	
3	-0.768909	0.253732	-0.592687	-0.764464	
4	1.750297	-1.151816	1.776573	1.826229	

5 rows × 31 columns

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report

# Load the transformed dataset
transformed_data = pd.read_csv('/content/drive/MyDrive/transformed_br

# Separate features (X) and target (y)
X = transformed_data.drop(columns=['diagnosis'])
y = transformed_data['diagnosis']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0

# Train the KNN classifier
knn = KNeighborsClassifier(n_neighbors=5)  # You can adjust the numbe
knn.fit(X_train, y_train)

# Predict on the test set
y_pred = knn.predict(X_test)

# Generate classification report
report = classification_report(y_test, y_pred, target_names=['Benign'

# Output the classification report
print(report)
```

```
              precision    recall  f1-score   support

      Benign       0.96      0.96      0.96        71
   Malignant       0.93      0.93      0.93        43

    accuracy                           0.95       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114
```

```
import numpy as np
from sklearn.preprocessing import StandardScaler

# Ask the user to input values for each feature
print("Please enter values for each feature:")
user_input = []
for feature in X.columns:
    value = input(f"{feature}: ")
    user_input.append(float(value))  # Convert input to float

# Convert the user input into a format suitable for prediction
user_input = np.array(user_input).reshape(1, -1)  # Reshape into a 2D

# Scale the user input using the same scaler used during training
```

```
# Scale the user input using the same scaler used during training
user_input_scaled = scaler.transform(user_input)

# Predict using the trained model
prediction = knn.predict(user_input_scaled)[0]

# Convert prediction to diagnosis
diagnosis = 'Malignant' if prediction == 1 else 'Benign'

# Output the prediction
print("Predicted Diagnosis:", diagnosis)
```

```
Please enter values for each feature:
radius_mean: 17.99
texture_mean: 10.38
perimeter_mean: 121.1
area_mean: 1200
smoothness_mean: 0.1184
compactness_mean: 0.21
concavity_mean: 0.31
concave points_mean: 0.13
symmetry_mean: 0.2419
fractal_dimension_mean: 0.07871
radius_se: 1.095
texture_se: 0.91
perimeter_se: 8.6
area_se: 154
smoothness_se: 0.006399
compactness_se: 0.049
concavity_se: 0.056
concave points_se: 0.01587
symmetry_se: 0.03004
fractal_dimension_se: 0.0061
radius_worst: 25.38
texture_worst: 17.33
perimeter_worst: 184.5
area_worst: 2019
smoothness_worst: 0.167
compactness_worst: 0.6667
concavity_worst: 0.76
concave points_worst: 0.2654
symmetry_worst: 0.46
fractal_dimension_worst: 0.119
Predicted Diagnosis: Malignant
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: Us
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: Us
  warnings.warn(
```