NAME: Jasmine Njeri Mwangi

PHASE 3 MACHINE LEARNING PROJECT

1.0 BUSINESS UNDERSTANDING

Breast Cancer is one of the prevalent cancers affecting mostly women globally. Time is of the essence when it comes to its detection so that one can seek healthcare as early as possible. Medical institutions, research organizations and healthcare providers are constantly striving to enhance the accuracy and efficiency of diagnostic methods.

2.0 KEY OBJECTIVE

The key objective is to develop a predictive model that can assist in early detection and produce accurate diagnosis of breast cancer

In [ ]:
```
# Importing relevant libraries
import numpy as np
import pandas as pd
import sqlite3
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import classification_report, accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import GridSearchCV
from imblearn.over_sampling import SMOTE
```

In [ ]:
```
# Connecting to sqlite database
path =r"C:\Users\HP\AppData\Local\Temp\a16388ed-095c-4078-8f2d-971aaae987d0_BREAST
conn = sqlite3.connect(path)
```

In [ ]:
```
# Loading the breast cancer file into a dataframe
df = pd.read_csv(path)
df
```

Out[ ]:

	no-recurrence-events	30-39	premeno	30-34	0-2	no	3	left	left_low	no.1
0	no-recurrence-events	40-49	premeno	20-24	0-2	no	2	right	right_up	no
1	no-recurrence-events	40-49	premeno	20-24	0-2	no	2	left	left_low	no
2	no-recurrence-events	60-69	ge40	15-19	0-2	no	2	right	left_up	no
3	no-recurrence-events	40-49	premeno	0-4	0-2	no	2	right	right_low	no
4	no-recurrence-events	60-69	ge40	15-19	0-2	no	2	left	left_low	no
...	...	...	...	...	...	...	...	...	...	...
280	recurrence-events	30-39	premeno	30-34	0-2	no	2	left	left_up	no
281	recurrence-events	30-39	premeno	20-24	0-2	no	3	left	left_up	yes
282	recurrence-events	60-69	ge40	20-24	0-2	no	1	right	left_up	no
283	recurrence-events	40-49	ge40	30-34	3-5	no	3	left	left_low	no
284	recurrence-events	50-59	ge40	30-34	3-5	no	3	left	left_low	no

285 rows × 10 columns

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   no-recurrence-events  285 non-null    object
 1   30-39                 285 non-null    object
 2   premeno               285 non-null    object
 3   30-34                 285 non-null    object
 4   0-2                   285 non-null    object
 5   no                    285 non-null    object
 6   3                     285 non-null    int64 
 7   left                  285 non-null    object
 8   left_low              285 non-null    object
 9   no.1                  285 non-null    object
dtypes: int64(1), object(9)
memory usage: 22.4+ KB
```

In [ ]:
```
df.describe()
```

Out[ ]:

	3
count	285.000000
mean	2.045614
std	0.737351
min	1.000000
25%	2.000000
50%	2.000000
75%	3.000000
max	3.000000

3.0 LOADING DATA

In [ ]:
```
# The 'select_dtypes' method is used to filter columns with numerical data types ('
# Numerical Columns
print(f"Numerical Columns: {df.select_dtypes(include='number').columns}\n")
()

# The 'select__dtypes' method is used to filter columns with object data types (typ
# Categorical Columns
print(f"Categorical Columns: {df.select_dtypes(include='object').columns}")
```

```
Numerical Columns: Index(['3'], dtype='object')

Categorical Columns: Index(['no-recurrence-events', '30-39', 'premeno', '30-34', '0-
2', 'no',
       'left', 'left_low', 'no.1'],
      dtype='object')
```

4. DATA CLEANING

4.1 HANDLING MISSING VALUES AND DUPLICATES

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
no-recurrence-events    0
30-39                   0
premeno                 0
30-34                   0
0-2                     0
no                      0
3                       0
left                    0
left_low                0
no.1                    0
dtype: int64
```

In [ ]:
```
# calculate percentage of missing values
missing_percent = df.isnull().mean().round(4) * 100
missing_count = df.isnull().sum()
# calculate percentage of duplicate rows
duplicates_percent = df.duplicated().mean() * 100
# creating a result dataframe
result = pd.DataFrame({'Missing Values %': missing_percent,
                       'Missing Values Count': missing_count,
                       'Duplicate Values %': duplicates_percent})
# find column with most missing values
if missing_percent.any():
    column_most_missing = missing_percent.idxmax()
    print(f"{(column_most_missing).capitalize()} is the column with most null count
    print()
else:
    print("No column with missing values")
# Display if there are any duplicate rows
if duplicates_percent.max() > 0: # type: ignore
    column_most_duplicates = duplicates_percent.max() # type: ignore
    print("Column with most duplicates:",column_most_duplicates)
else:
    print("No duplicates")
print(result)
```

```
No column with missing values
Column with most duplicates: 4.912280701754386
                      Missing Values %  Missing Values Count  \
no-recurrence-events               0.0                     0
30-39                              0.0                     0
premeno                            0.0                     0
30-34                              0.0                     0
0-2                                0.0                     0
no                                 0.0                     0
3                                  0.0                     0
left                               0.0                     0
left_low                           0.0                     0
no.1                               0.0                     0

                      Duplicate Values %
no-recurrence-events            4.912281
30-39                           4.912281
premeno                         4.912281
30-34                           4.912281
0-2                             4.912281
no                              4.912281
3                               4.912281
left                            4.912281
left_low                        4.912281
no.1                            4.912281
```

In [ ]:
```
# Checking for number of duplicate rows
df.nunique()
```

Out[ ]:
```
no-recurrence-events     2
30-39                    6
premeno                  3
30-34                   11
0-2                      7
no                       3
3                        3
left                     2
left_low                 6
no.1                     2
dtype: int64
```

In [ ]:
```
df.drop_duplicates()
```

Out[ ]:

	no-recurrence-events	30-39	premeno	30-34	0-2	no	3	left	left_low	no.1
0	no-recurrence-events	40-49	premeno	20-24	0-2	no	2	right	right_up	no
1	no-recurrence-events	40-49	premeno	20-24	0-2	no	2	left	left_low	no
2	no-recurrence-events	60-69	ge40	15-19	0-2	no	2	right	left_up	no
3	no-recurrence-events	40-49	premeno	0-4	0-2	no	2	right	right_low	no
4	no-recurrence-events	60-69	ge40	15-19	0-2	no	2	left	left_low	no
...	...	...	...	...	...	...	...	...	...	...
280	recurrence-events	30-39	premeno	30-34	0-2	no	2	left	left_up	no
281	recurrence-events	30-39	premeno	20-24	0-2	no	3	left	left_up	yes
282	recurrence-events	60-69	ge40	20-24	0-2	no	1	right	left_up	no
283	recurrence-events	40-49	ge40	30-34	3-5	no	3	left	left_low	no
284	recurrence-events	50-59	ge40	30-34	3-5	no	3	left	left_low	no

271 rows × 10 columns

In [ ]:
```
df.nunique()
```

Out[ ]:
```
no-recurrence-events     2
30-39                    6
premeno                  3
30-34                   11
0-2                      7
no                       3
3                        3
left                     2
left_low                 6
no.1                     2
dtype: int64
```

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
no-recurrence-events    0
30-39                   0
premeno                 0
30-34                   0
0-2                     0
no                      0
3                       0
left                    0
left_low                0
no.1                    0
dtype: int64
```

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['no-recurrence-events', '30-39', 'premeno', '30-34', '0-2', 'no', '3',
       'left', 'left_low', 'no.1'],
      dtype='object')
```

MODELLING

In [ ]:
```
# Assign column names for better readability
# Assign column names for better readability
df.columns = [
    'Class', 'Age', 'Menopause', 'Tumor Size', 'Inv Nodes', 'Node Caps',
    'Deg Malig', 'Breast', 'Breast Quad', 'Irradiat'
]
```

In [ ]:
```
# Encode categorical variables

label_encoders = {}
for column in df.columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le
```

In [ ]:
```
# Split the data into features (X) and target (Y)
X = df.drop(columns=["Class"])
y = df["Class"]
```

In [ ]:
```
# Splitting the datset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =0.2, random_st
```

In [ ]:
```
# Training a random forest classifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train,y_train)
```

Out[ ]:
```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [ ]:
```
# Predict on the test set
y_pred = model.predict(X_test)
```

In [ ]:
```
# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=label_encoders["Class"]
```

In [ ]:
```
accuracy
```

Out[ ]:
```
0.7368421052631579
```

In [ ]:
```
report
```

Out[ ]:
```
'                    precision    recall  f1-score   support\n\nno-recurrence-ev
ents       0.75      0.89      0.81        37\n   recurrence-events       0.69
0.45      0.55        20\n\n            accuracy                           0.74
57\n           macro avg       0.72      0.67      0.68        57\n        weighte
d avg       0.73      0.74      0.72        57\n'
```

We can see that the model performs well in cases with no reccurence of breast cancer. The random forest classifier has achieved an accuracy of approximately 67.2%.

To improve this model we will need to balance the classes

In [ ]:
```
# Here I am applying SMOTE to balance the classes in the training set

smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)
```

In [ ]:
```
#Defining the parameter grid
param_grid = {
    "n_estimators" :[50,100,200],
    "max_depth" : [None, 10, 20, 30],
    "min_samples_split" : [2, 5, 10],
    "min_samples_leaf" : [1, 2, 4],
    "bootstrap" : [True, False]

}
```

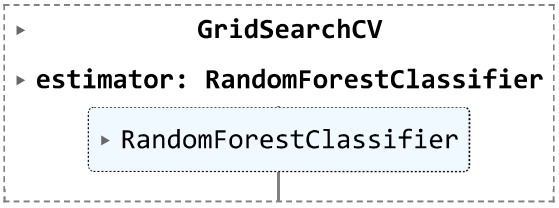
In [ ]:
```
# Initialize the Random Forest Model
rf_model = RandomForestClassifier(random_state = 42)
```

In [ ]:
```
# Initialize GridSerachCV for hyperparameter tuning
grid_search = GridSearchCV(estimator = rf_model, param_grid= param_grid, cv=5, n_jo
```

In [ ]:
```
# Fitting the model to the balanced training data
grid_search.fit(X_train_balanced, y_train_balanced)
```

```
Fitting 5 folds for each of 216 candidates, totalling 1080 fits
```

Out[ ]:



In [ ]:
```
# Getting the best estimator and evaluate on the test set
best_rf_model = grid_search.best_estimator_
y_pred_best = best_rf_model.predict(X_test)
```

In [ ]:
```
#Evaluating the optimized model
accuracy_best = accuracy_score(y_test, y_pred_best)
report_best = classification_report(y_test,y_pred_best, target_names = label_encode
```

In [ ]:
```
accuracy_best
```

Out[ ]:
```
0.7017543859649122
```

In [ ]:
```
report_best
```

Out[ ]:
```
'                    precision    recall  f1-score   support\n\nno-recurrence-ev
ents       0.72      0.89      0.80        37\n   recurrence-events       0.64
0.35      0.45        20\n\n            accuracy                           0.70
57\n           macro avg       0.68      0.62      0.62        57\n        weighte
d avg       0.69      0.70      0.67        57\n'
```

SUMMARY

Balancing the Dataset, improving the model, and planning for ongoing updates are key steps in creating a reliable tool for diagnosing breast cancer.

By continuing to refine this model and making sure it fits well into clinical practice, this tool can play an important role in detecting breast cancer early and helping patients get better healthcare.