**Student Name: Oyatokun Modupeola**

**Student ID: C0895705**

**Assignment: Breast Cancer Data Analysis and Streamlit App**

This report outlines the steps taken in analyzing a breast cancer dataset. The project encompasses loading the dataset, examining its structure, cleaning and preparing the data, visualizing relationships, and constructing a machine learning model to make predictions.

The first step is to create a new directory for the breast cancer assignment in VS Code. Initialized a Git repository.

Created and activated a virtual environment

**Loading Libraries and Dataset**

Importing necessary libraries and loading the dataset.

**tensorflow==2.15.0**

**pandas**

**numpy**

**scikit-learn**

**tensorboard**

**matplotlib**

**streamlit**

**scikeras**

**seaborn**

Load the dataset (the data set was gotten from Kaggle)

A black screen with white text

Description automatically generated

Exploring the Dataset

The first step in exploring the dataset involves viewing the first few rows and getting summary statistics to understand the data's structure and basic characteristics.

A screen shot of a black screen

Description automatically generated

**Data Cleaning and Preparation**

Data cleaning and preparation are essential steps to ensure the dataset is ready for analysis. This may involve dropping irrelevant columns or encoding categorical variables.

Dropping irrelevant columns. The 'Unnamed: 32' column is dropped.

A screen shot of a computer

Description automatically generated

The code data.diagnosis.replace (to\_replace=dict(M=1, B=0), inplace=True) replaces 'M' with 1 and 'B' with 0 in the 'diagnosis' column of the DataFrame data. This is typically done to convert categorical values (like 'M' and 'B') into numerical values (1 and 0) which are easier to work with in machine learning models. By performing this replacement, the 'diagnosis' column now contains numerical values (1 for 'Malignant' and 0 for 'Benign'), making it suitable for training machine learning models or performing statistical analysis where numerical inputs are expected.

**Replacing categorical variables to Numerical**

A computer screen with text

Description automatically generated

**Data Visualization**

Visualize the distribution of the target variable

**A screen shot of a computer program

Description automatically generated**

**A diagram of a distribution of diagnosis

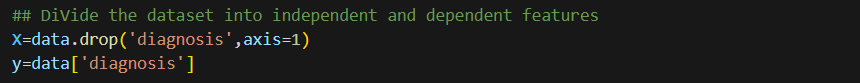
Description automatically generated**

A colorful grid with numbers

Description automatically generated with medium confidence

This visualization is particularly useful for understanding relationships between different features in dataset, which can guide feature selection, model building, and further analysis.The heatmap provides a visual representation of how strongly each variable is correlated with every other variable in the dataset.

Dividing the dataset to independent and dependent features. This is done by dropping the target variable which in the case is diagnosis.



Then split the dataset to train and test



Pickle is a Python module that implements binary protocols for serializing and deserializing Python objects.

Purpose Pickle

Reusability: By saving these objects to files, you can reuse the fitted label\_encoder\_diagnosis and scaler without needing to refit them each time you run your code. This is useful for consistent preprocessing in different sessions or environments.

Efficiency: Loading a serialized object is faster than refitting it, which can save time, especially when dealing with large datasets or complex preprocessing steps.

Sharing: Serialized objects can be easily shared with others, allowing them to use the same preprocessing steps without needing access to the original data.

A black background with text

Description automatically generated

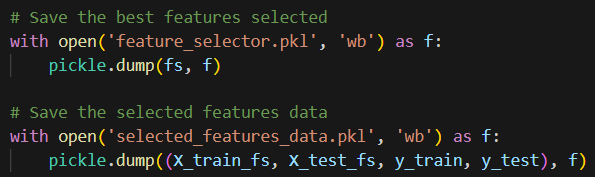
**Features Selection**

The select\_features function employs SelectKBest with the f\_classif scoring function to identify the top k features most pertinent to the target variable (y\_train). It fits the SelectKBest object to the training data, then transforms both the training and testing datasets to retain only the selected features. The function returns these transformed datasets and the fitted feature selector. This approach reduces the data's dimensionality, emphasizes the most significant features, and can enhance the performance of machine learning models.

A computer screen shot of text

Description automatically generated

Save the selected features



**Model Building and Grid Search CV for Model Tuning**

This process is crucial for improving model performance by finding the optimal hyperparameters for the neural network classifier.

This is done using scikit-learn's GridSearchCV and MLPClassifier

**GridSearchCV**: This is a module from scikit-learn that allows exhaustive search over specified parameter values for an estimator.

**MLPClassifier**: This is a Multi-Layer Perceptron classifier, which is a type of artificial neural network model.

**Parameter\_space:** This dictionary defines the hyperparameters to be tuned and their possible values.

Hidden\_layer\_sizes: Defines the architecture of the neural network. Different configurations of hidden layers and neurons are specified.

Activation: Activation functions for the hidden layers.

Solver: Optimization algorithms for weight adjustment.

Alpha: Regularization term to prevent overfitting.

Learning\_rate: Strategies for learning rate adaptation.

MLPClassifier(max\_iter=100): Initializes the MLPClassifier with a maximum of 100 iterations.

GridSearchCV(mlp, parameter\_space, n\_jobs=-1, cv=3):

mlp: The estimator (MLPClassifier) to be optimized.

parameter\_space: The hyperparameter space to search.

n\_jobs=-1: Uses all available CPUs to perform the computation.

cv=3: Uses 3-fold cross-validation.

Fits the grid search model on the training data. print theOutputs the best combination of hyperparameters found. (**return clf**) Returns the fitted grid search object.

A computer screen shot of text

Description automatically generated

**Saving the Model**:

* The trained model (clf) is saved to a file named clf\_model.pkl using joblib.dump().
* joblib.dump(clf, 'clf\_model.pkl'): Serializes and saves the trained model to the specified file.

A screen shot of a computer program

Description automatically generated

**Loading Selected Features Data**

* The selected feature data and corresponding labels, which were previously saved in a file named selected\_features\_data.pkl, are loaded using pickle.load().
* with open('selected\_features\_data.pkl', 'rb') as f: Opens the file in binary read mode ('rb').
* X\_train\_fs, X\_test\_fs, y\_train, y\_test: These variables store the loaded training and testing feature sets and labels.

**Performing Grid Search Cross-Validation**

* The grid\_search\_cv function is called with the training features (X\_train\_fs) and training labels (y\_train).
* This function performs hyperparameter tuning using GridSearchCV and returns the best estimator (clf), which is a trained MLPClassifier model.

**Implementing an Artificial Neural Network (ANN) Model**

Making Predictions and Evaluating the Model

**A screen shot of a computer code

Description automatically generated**

Defining the train\_evaluate\_ann Function

X\_train\_fs: The training feature set (with selected features).

y\_train: The training target set.

X\_test\_fs: The testing feature set (with selected features).

y\_test: The testing target set.

clf: The trained classifier model.

Prediction:

y\_pred = clf.predict(X\_test\_fs):

This line uses the predict method of the classifier clf to generate predictions for the test feature matrix X\_test\_fs.

y\_pred will contain the predicted labels for the test set.

Accuracy Calculation:

accuracy\_score(y\_test, y\_pred) calculates the accuracy of the predictions by comparing the true labels (y\_test) with the predicted labels (y\_pred).

classification\_report(y\_test, y\_pred) generates a detailed classification report that includes precision, recall, and F1-score for each class, as well as overall metrics such as macro and weighted averages.

A screenshot of a computer

Description automatically generated

Accuracy measures the proportion of correct predictions (both true positives and true negatives) out of all predictions

In this case, the accuracy is approximately 98.25%, meaning that the classifier correctly predicted the labels for about 98.25% of the test samples.

**Prediction.ipynb**

import libraries

A screen shot of a computer

Description automatically generated

Load the saved models

A screen shot of a computer program

Description automatically generated

This function, predict\_cancer, is designed to predict whether an input sample indicates breast cancer using a pre-trained machine learning model.This function takes raw input data, scales it, selects relevant features, and uses a pre-trained model to predict the probability that the input data indicates breast cancer.

By Inputting the input data,

The percentage of the prediction gives

A screen shot of a computer program

Description automatically generated

**Application for Deployment -** **Building a Streamlit App Locally**

Develop a simple Streamlit application where users can engage with the breast cancer dataset and observe model predictions. Integrate model predictions and user interaction seamlessly within the Streamlit app.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

**A screen shot of a computer

Description automatically generated**

**Deployment**

Local URL: <http://localhost:8501>

Two predictions were made by changing the input parameter.

Prediction - Benign

A screenshot of a computer screen

Description automatically generated

Prediction - Malignant

A screenshot of a screen

Description automatically generated

**GitHub Repository Setup**: Commit code and push changes to GitHub

GitHub link: [Dupsynusi/Breast-Cancer-Prediction-App (github.com)](https://github.com/Dupsynusi/Breast-Cancer-Prediction-App/tree/main)