4-25-2024

Evaluating Dimension Reduction and Machine Learning Classifiers for Breast Cancer Prognosis

Tristan Douglas Kennedy
University of Alabama in Huntsville

Evaluating Dimension Reduction and Machine Learning Classifiers for Breast Cancer Prognosis

by

Tristan Douglas Kennedy

An Honors Capstone
submitted in partial fulfillment of the requirements
for the Honors Diploma
to

The Honors College

of

The University of Alabama in Huntsville

April 25th, 2024

Capstone Project Director: Dr. Chaity Banerjee Mukherjee

[illegible] 04-16-2024

Student Date

Dr. Chaity Banerjee Mukherjee

Digitally signed by Dr. Chaity Banerjee Mukherjee
DN: OU=Computer Science, O=UAH, CN=Dr. Chaity Banerjee Mukherjee, E=chaity.banerjee.mukherjee@uah.edu
Reason: I have reviewed this document
Location: Huntsville AL
Date: 2024.04.16 15:30:51-05'00'
Foxit PDF Editor Version: 12.1.1

Project Director Date

Letha Etzkorn

Digitally signed by Letha Etzkorn
Date: 2024.04.16 16:16:21 -05'00'

Department Chair Date

Sean Lane

Digitally signed by Sean M Lane
Date: 2024.04.17 13:50:33 -05'00'

Honors College Dean Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)

Honors Thesis Copyright Permission

This form must be signed by the student and submitted with the final manuscript.

_______Tristan Kennedy_______
Student Name (printed)

[illegible]
Student Signature

_________04/16/2024________
Date

Table of Contents

Abstract

In the realm of machine learning, the performance and suitability of different algorithms can vary significantly depending on the problem domain and the characteristics of the data. Evaluating and comparing the capabilities of various algorithms using real-world datasets is crucial for identifying the most appropriate techniques for specific tasks. This project aims to conduct an analysis of three widely used machine learning algorithms: logistic regression, K-nearest neighbors (KNN), and support vector machines (SVM), by applying them to a real-world dataset. The dataset used in this study consists of physical parameters collected from consecutive breast cancer patients by Dr. Wolberg since 1984. Although the dataset originated from a medical context involving breast cancer patients, the analysis primarily focuses on evaluating the performance of the selected machine learning algorithms in accurately classifying instances based on the provided features, rather than exploring the specific medical implications. The performance of the three models was evaluated using confusion matrices, which provide a comprehensive view of the models' accuracy, including true positives, true negatives, false positives, and false negatives. The written document accompanying this project provides a detailed description of the dataset, feature selection process, model development, and evaluation methodologies. It also presents and discusses the results obtained from the three classification algorithms, highlighting their strengths and limitations in predicting metastatic breast cancer recurrence.

Introduction

The ability to accurately classify instances or predict outcomes is of paramount importance across various domains. From medical diagnostics to financial forecasting, and from natural language processing to image recognition, the performance of classification algorithms can have far-reaching implications. However, the efficacy of these algorithms is often dependent on the characteristics of the data at hand, as well as the careful selection and tuning of the appropriate techniques. By employing a systematic approach to feature selection, model training, and evaluation, this project seeks to provide insights into the strengths and limitations of these algorithms, contributing to the broader understanding of their applicability in diverse classification tasks.

The dataset used in this study, while originating from the medical domain, serves as a representative example of the challenges faced in real-world data analysis. The high dimensionality of the data, potential presence of missing values, and the need for effective feature engineering are common hurdles that must be addressed to ensure the reliability and accuracy of the resulting models. The goal of this project is an exercise in not only showcasing the breadth of techniques used in machine learning but also contributing to the analysis of commonly used machine learning algorithms, and their effectiveness on a specific given dataset.

While this project is constrained by the limited dataset, accurately predicting metastatic breast cancer recurrence could significantly impact treatment planning and patient outcomes. If a machine learning algorithm could reliably determine the likelihood of cancer recurrence, understanding the strengths and limitations of different classifiers for this task would be invaluable. Although training a large-scale predictive model would likely require more extensive data processing and advanced machine learning techniques, this project serves as a proof of concept and comparative analysis of dimension reduction and classification methods applied to breast cancer prognosis. By evaluating the performance of logistic regression, K-nearest neighbors, and support vector machines on this dataset, this study provides insights into the suitability of these algorithms for predicting breast cancer recurrence, laying the groundwork for further research and model development in this critical domain.

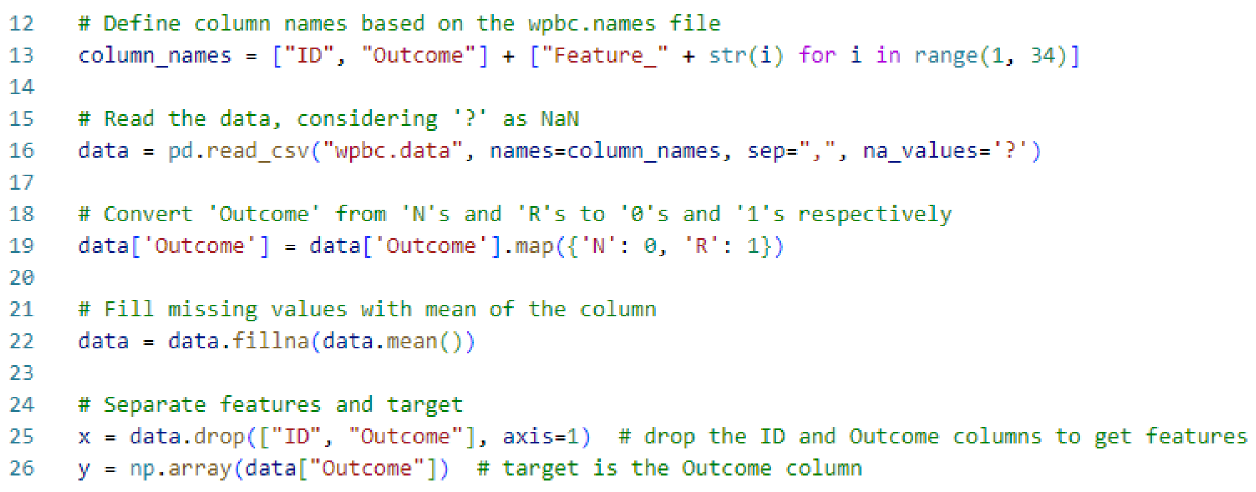
Methodology

Dataset

In more specifics, the dataset used is a dataset titled “Breast Cancer Wisconsin (Prognostic)” which was donated for educational and scientific use in 1995. Each of 198 records represents a follow-up for one breast cancer case where the outcome is listed as either recur (R) or non-recur (N), meaning the cancer returned or it did not after a period of time being undetectable. The first 30 features include very specific physical features of the cell nuclei such as perimeter, area, circumference, and concavity, while the last 3 features are: the amount of time between check-ups, lymph node status (number of positive axillary lymph nodes) , and tumor size. The dataset includes 4 missing instances of lymph node status which are handled during the preprocessing step of the data management. The complete list of 33 features can be found on the original dataset website linked on the reference page.

Preprocessing

Before applying any machine learning algorithms, the dataset underwent several preprocessing steps to ensure data quality and consistency. The following code snippet showcases how the data is read into lists and separated via column name using the pandas library. The pandas library allows for missing values “na_values” to be easily identified as ? 's which is how they are defined in the initial dataset. This then allows for the ‘fillna’ command of the pandas library to be leveraged when the values are later filled in. Line 19 sees the positive classifier defined as R, in this case, while the negative or zero classifier is N. FInally, on line 22, missing values were identified and imputed using the column average method, as shown in the following code snippet:



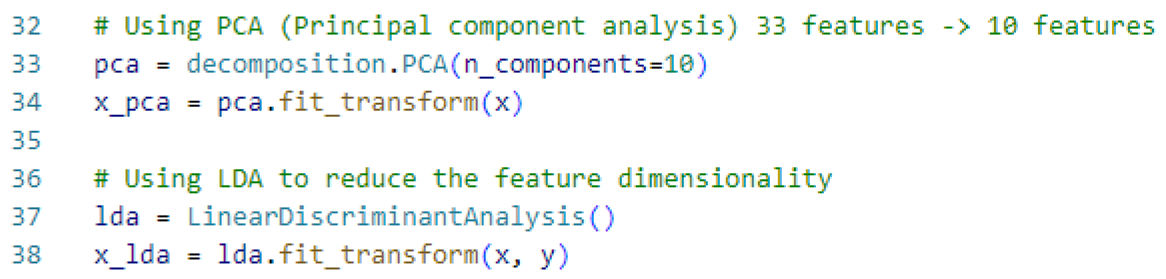
```
12   # Define column names based on the wpbc.names file
13   column_names = ["ID", "Outcome"] + ["Feature_" + str(i) for i in range(1, 34)]
14
15   # Read the data, considering '?' as NaN
16   data = pd.read_csv("wpbc.data", names=column_names, sep=",", na_values='?')
17
18   # Convert 'Outcome' from 'N's and 'R's to '0's and '1's respectively
19   data['Outcome'] = data['Outcome'].map({'N': 0, 'R': 1})
20
21   # Fill missing values with mean of the column
22   data = data.fillna(data.mean())
23
24   # Separate features and target
25   x = data.drop(["ID", "Outcome"], axis=1)  # drop the ID and Outcome columns to get features
26   y = np.array(data["Outcome"])  # target is the Outcome column
```

Figure 1: Lines 12 through 26 of Program Code

Additionally, any necessary data transformations, in this case standardization, were performed automatically within library functions to ensure compatibility with the chosen algorithms and improve model performance.

Feature Selection

Given the dimensionality of the original dataset, comprising 30 physical parameters and an additional 3 miscellaneous parameters, some feature selection techniques were employed to identify the most relevant features for the classification task. Principal Component Analysis (PCA) as well as Linear Discriminant Analysis (LDA) were used to reduce the dimensionality of the data while retaining the most significant information.



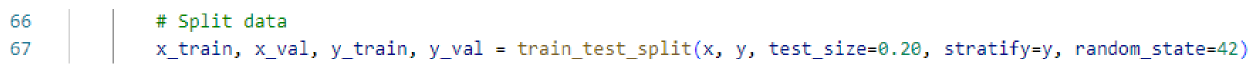
```
32   # Using PCA (Principal component analysis) 33 features -> 10 features
33   pca = decomposition.PCA(n_components=10)
34   x_pca = pca.fit_transform(x)
35
36   # Using LDA to reduce the feature dimensionality
37   lda = LinearDiscriminantAnalysis()
38   x_lda = lda.fit_transform(x, y)
```

Figure 2: Lines 32 through 38 of Program Code

PCA is a technique used in data science and machine learning to reduce the dimensionality of a dataset while still retaining the high-level patterns or trends that are found in the data. While it does not necessarily retain any of the original data, it distills the data down into a more usable, simpler form. This is done using linear combinations and in this case resulted in 10 new, related features (principal components) which were subsequently used as input for the classification algorithms. LDA is another technique aiming to do a similar thing, but specifically in regards to a classification problem. It results in the data being transformed into n-1 classes and focuses on maximizing the separation between classes as opposed to maximizing variance like PCA does.

Model Training and Evaluation

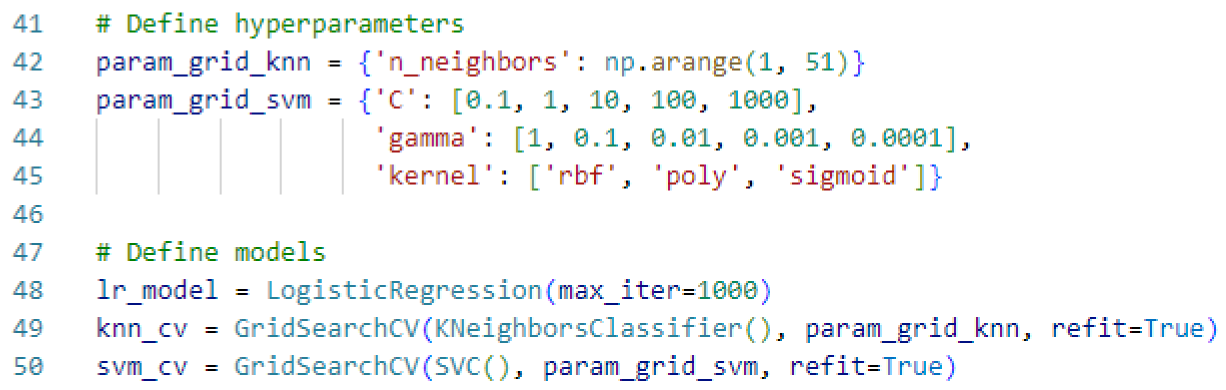
Three widely used machine learning algorithms were selected for this study: logistic regression, K-nearest neighbors (KNN), and support vector machines (SVM). Each algorithm was trained and evaluated using the selected features and the corresponding target variable (recurrence or non-recurrence). The models were each trained on a random 80% of the data and then validated against the remaining 20% of the data after. This split is stratified to the y values (the classification) so that the data is trained on a realistic split of % R and % N as to not bias in any direction as shown below:



```
66        # Split data
67        x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=0.20, stratify=y, random_state=42)
```

Figure 3: Lines 66 through 67 of Program Code

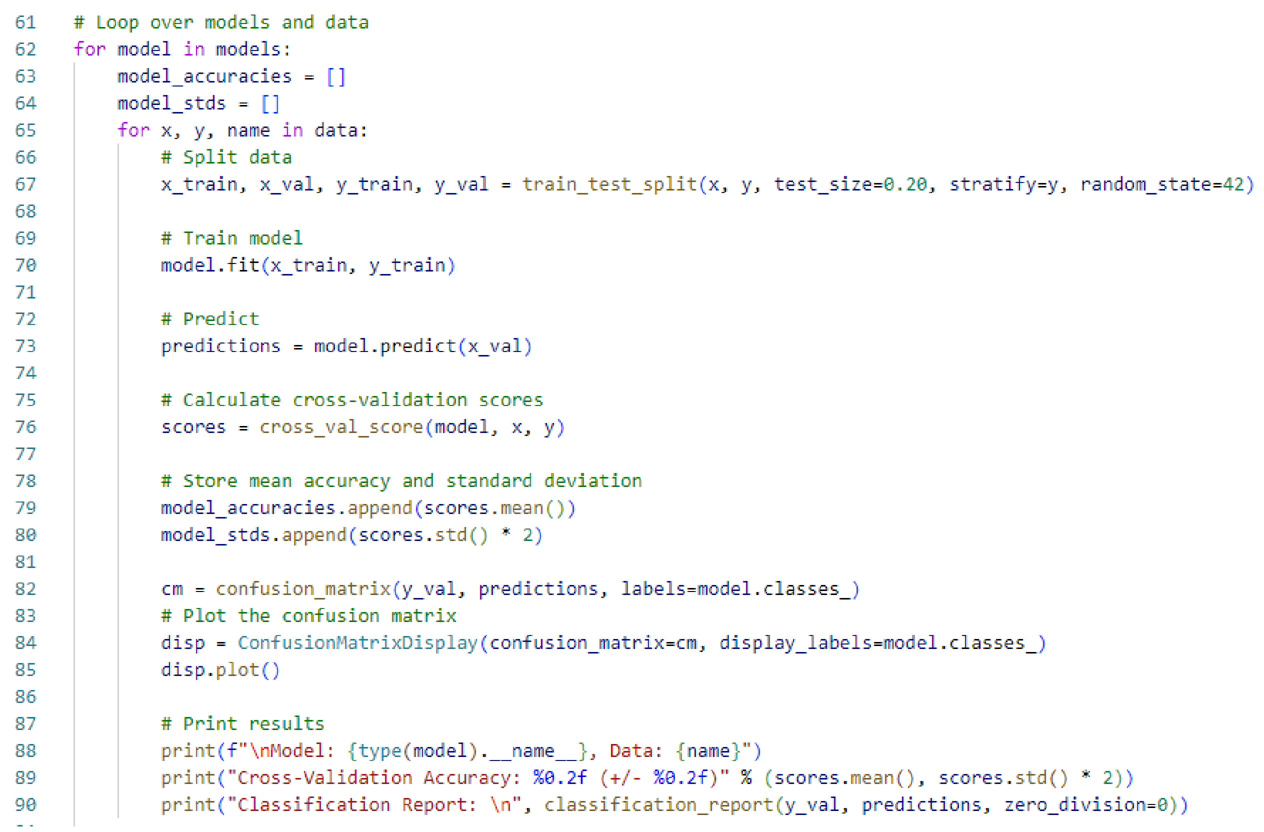
Once the data was separated into validation and training sets, the actual training of the machine learning models could begin. While the training process is consistent across the models, the implementation and underlying algorithms differ for each classifier. One important step in setting up the KNN model and the SVM model was hyperparameter tuning. KNN has one important hyperparameter, n_neighbors, while SVM has three potentially important hyperparameters, C, gamma, and kernel. I chose to simply tune the hyperparameters using an exhaustive GridSearchCV. This loops through the options defined in the grid dictionaries shown below on lines 42 and 43 and finds the hyperparameters that perform best with cross-validation, ultimately training on the given dataset using those hyperparameters.



```
41   # Define hyperparameters
42   param_grid_knn = {'n_neighbors': np.arange(1, 51)}
43   param_grid_svm = {'C': [0.1, 1, 10, 100, 1000],
44                     'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
45                     'kernel': ['rbf', 'poly', 'sigmoid']}
46
47   # Define models
48   lr_model = LogisticRegression(max_iter=1000)
49   knn_cv = GridSearchCV(KNeighborsClassifier(), param_grid_knn, refit=True)
50   svm_cv = GridSearchCV(SVC(), param_grid_svm, refit=True)
```

Figure 4: Lines 41 through 50 of Program Code

With the models set up, they can be looped through and trained as shown below. Once trained the models are predicted on using the validation set, and a cross validation score is calculated for each model and dimensionality reduction combination. This includes each of: no dimensionality reduction, PCA, and LDA alongside the three aforementioned algorithms. For each combination of feature selection method and classification algorithm, the cross-validation accuracies and standard deviations are computed and stored for later visualization. Additionally, the confusion matrices are calculated and plotted to provide a comprehensive assessment of the models' performance. Finally, the classification_report for each combination is calculated and printed to the screen so that it can be analyzed later.



```
61  # Loop over models and data
62  for model in models:
63      model_accuracies = []
64      model_stds = []
65      for x, y, name in data:
66          # Split data
67          x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=0.20, stratify=y, random_state=42)
68  
69          # Train model
70          model.fit(x_train, y_train)
71  
72          # Predict
73          predictions = model.predict(x_val)
74  
75          # Calculate cross-validation scores
76          scores = cross_val_score(model, x, y)
77  
78          # Store mean accuracy and standard deviation
79          model_accuracies.append(scores.mean())
80          model_stds.append(scores.std() * 2)
81  
82          cm = confusion_matrix(y_val, predictions, labels=model.classes_)
83          # Plot the confusion matrix
84          disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model.classes_)
85          disp.plot()
86  
87          # Print results
88          print(f"\nModel: {type(model).__name__}, Data: {name}")
89          print("Cross-Validation Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
90          print("Classification Report: \n", classification_report(y_val, predictions, zero_division=0))
```

Figure 5: Lines 61 through 90 of Program Code

The cross-validation approach used allowed for a more reliable estimation of the models' generalized performance by evaluating them on multiple train-validation splits of the data. This methodology helped mitigate the risk of overfitting and provided a more comprehensive assessment of the models' capabilities. This approach provided not only a view of the accuracy and confusion matrix for the current model but also the average and standard deviation of the accuracies over multiple iterations. It facilitated a definitive showcase of which algorithms were most well-suited for classifying the available dataset.

Results and Discussion

Following the methodology exactly as described above, and an execution of the code yields the results formatted below. Each of the confusion matrices and classification reports were analyzed with the cross validation accuracies for the model and dimension reduction technique combinations.

Cross Validation Accuracies

As discussed, cross validation accuracies lend themselves to an overview of the models capabilities, and are less prone to overfitting. This is due to the fact that the score is calculated by “folding” the dataset into 5 different sections and retraining on the dataset for each fold. This yields an average accuracy as well as a standard deviation which was multiplied by 2 to represent where 95% of the accuracy scores would fit into, given multiple retrainings of the model on different training/validation splits.

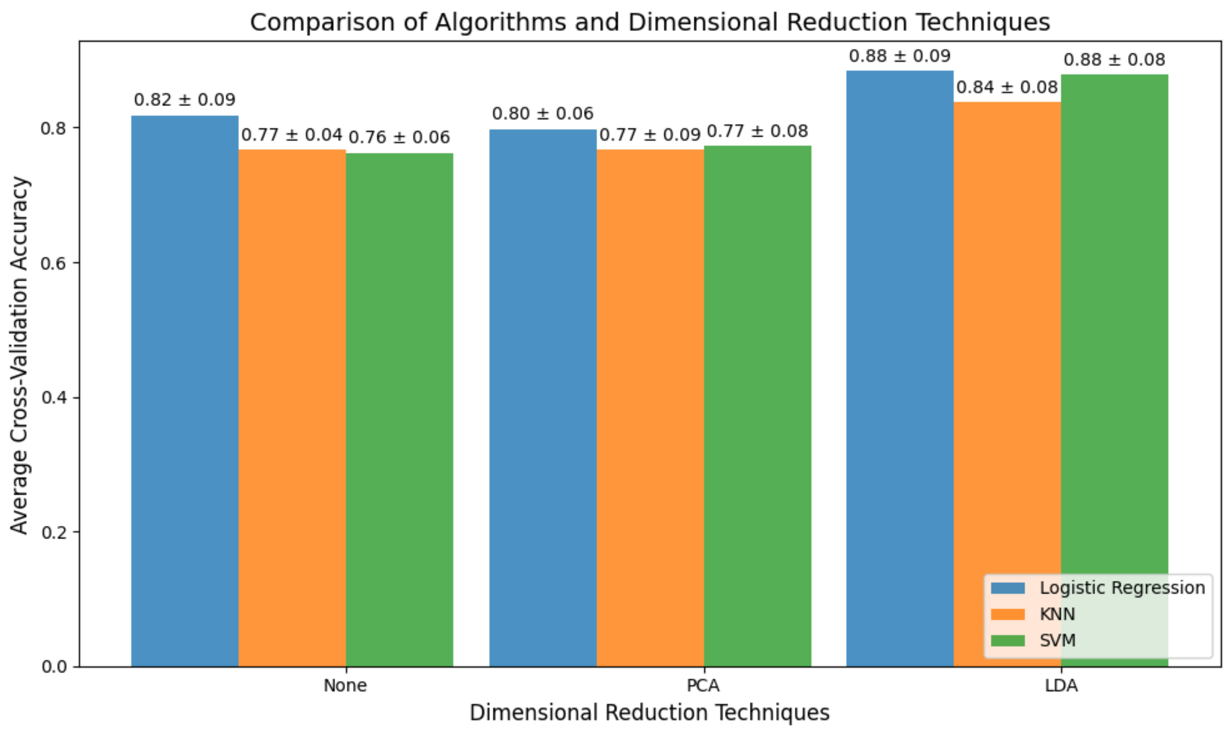


Figure 6: Average Cross Validation Accuracy by Dimensional Reduction Technique and Machine Learning Algorithm

Looking at the above results it is evident that the PCA dimensionality reduction technique did not lend itself to the dataset very well, and in some cases hindered the performance of the machine learning model. This was not the case at all for LDA, which outperformed both non-reduced and PCA in all machine learning algorithms. Additionally, an initial analysis of the data only analyzing the accuracy would conclude that Logistic Regression outperforms the other models on this given dataset.

Confusion Matrix and Classification Report Analysis

The results for each combination of dimensionality reduction technique and algorithm continue in the following section, including the classification report, which can be calculated from the resulting confusion matrices. All of these results were yielded on an 80% training, 20% validation split, with the random_state=42 being used to ensure consistent, and reproducible results. The classification report includes data calculated from the confusion matrix, including the following:

1. Precision - The quality of the predictions by the model; when this was predicted, how often was the prediction correct.
2. Recall - The ability for the model to correctly predict the class; when this is the class, how often was the answer predicted correctly.
3. F1 Score - A mean measure between the precision and the recall.
4. Support - The number of data points that exist within the validation set of the class

The raw confusion matrix data can be found in the appendix.

	Precision	Recall	F1-Score	Support
0	0.88	0.94	0.91	31
1	0.71	0.56	0.62	9
Weighted Avg.	0.84	0.85	0.84	40

Figure 7: Table for Logistic Regression, No Dimensionality Reduction

	Precision	Recall	F1-Score	Support
0	0.82	0.87	0.84	31
1	0.43	0.33	0.38	9
Weighted Avg.	0.73	0.75	0.74	40

Figure 8: Table for Logistic Regression, PCA

	Precision	Recall	F1-Score	Support
0	0.86	1.00	0.93	31
1	1.00	0.44	0.62	9
Weighted Avg.	0.89	0.88	0.86	40

Figure 9: Table for Logistic Regression, LDA

	Precision	Recall	F1-Score	Support
0	0.78	1.00	0.87	31
1	0.00	0.00	0.00	9
Weighted Avg.	0.60	0.78	0.68	40

Figure 10: Table for KNN, No Dimensionality Reduction

	Precision	Recall	F1-Score	Support
0	0.79	1.00	0.89	31
1	1.00	0.11	0.20	9
Weighted Avg.	0.84	0.80	0.73	40

Figure 11: Table for KNN, PCA

	Precision	Recall	F1-Score	Support
0	0.83	0.97	0.90	31
1	0.75	0.33	0.46	9
Weighted Avg.	0.81	0.82	0.80	40

Figure 12: Table for KNN, LDA

	Precision	Recall	F1-Score	Support
0	0.88	0.94	0.91	31
1	0.71	0.56	0.62	9
Weighted Avg.	0.84	0.85	0.84	40

Figure 13: Table for SVM, No Dimensionality Reduction

	Precision	Recall	F1-Score	Support
0	0.80	0.90	0.85	31
1	0.40	0.22	0.29	9
Weighted Avg.	0.71	0.75	0.72	40

Figure 14: Table for SVM, PCA

	Precision	Recall	F1-Score	Support
0	0.89	1.00	0.94	31
1	1.00	0.56	0.71	9
Weighted Avg.	0.91	0.90	0.89	40

Figure 15: Table for SVM, LDA

The data here continues to indicate that the initial analysis of LDA outperforming PCA and the non-reduced dataset is consistent. Additionally, logistic regression continues to have the edge in most of the reduction techniques indicating that the data itself might be more linearly separable. The LDA reduced set displays a similar performance between the SVM and Logistic Regression, with SVM potentially coming out slightly on top with higher Precision and Recall.

Conclusions

The goal of this project was to compare the performance of different machine learning classifiers alongside dimensionality reduction techniques to identify their strengths and weaknesses when it comes to identifying breast cancer recurrence given a specific dataset. This yielded the following conclusions: For the given dataset, LDA far outperformed PCA as well as the non-reduced dataset. Additionally logistic regression performed the most consistently on the data, indicating a linear relationship between the features and the classifiers. The overall best performing algorithm though was a support vector machine trained specifically with the LDA reduced data. This could indicate potential overfitting to the entire dataset, caused by the reduced dimensions influence on the hyperparameter tuning of the SVM.

Overall, the performance of this project was a success, with multiple models being trained that performed more accurately than just selecting recur or non-recur for each entry. Additionally, the accuracies observed for all of the models trained on the LDA reduced dataset were satisfactory, and indicated the potential for training a more complex model on a larger breast cancer dataset to improve the accuracy of predicting metastatic recurrence in the future

Throughout the course of this project, several lessons were learned that could inform future research and applications. The importance of thorough data preprocessing and effective feature selection techniques was highlighted through the vast performance increase from dimensionality reduction techniques and standardization. Additionally, The use of cross-validation strategies and appropriate model tuning techniques, such as hyperparameter optimization, played a crucial role in obtaining reliable performance estimates and mitigating the risk of overfitting.

Finally, several recommendations for future improvements and project directions could be made. While LDA proved effective in this study, other dimensionality reduction techniques could be explored to assess their impact on model performance and feature interpretability. Collaborating with domain experts and incorporating prior knowledge or constraints specific to the problem domain could enhance the interpretability and applicability of the models in real-world scenarios. Acquiring additional data points or exploring alternative datasets from similar domains could further validate the generalizability of the findings and potentially uncover new insights. While accuracy and confusion matrices were the primary evaluation metrics used in this study, other metrics such as area under the receiver operating characteristic curve (AUC-ROC) or precision-recall curves could provide additional insights into the models' performance, particularly in imbalanced datasets.

References

Wolberg,William, Street,W., and Mangasarian,Olvi. (1995). Breast Cancer Wisconsin (Prognostic). UCI Machine Learning Repository. https://doi.org/10.24432/C5GK50.

Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011

Data structures for statistical computing in python, McKinney, Proceedings of the 9th Python in Science Conference, Volume 445, 2010.

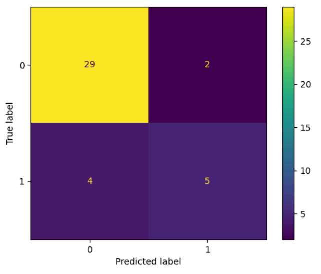
Harris, C.R., Millman, K.J., van der Walt, S.J. et al. *Array programming with NumPy*. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.

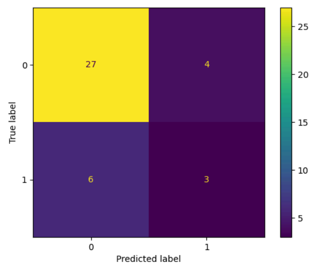
J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

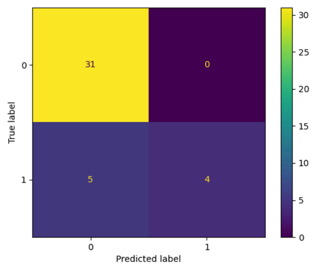
Appendix

1. Confusion Matrices

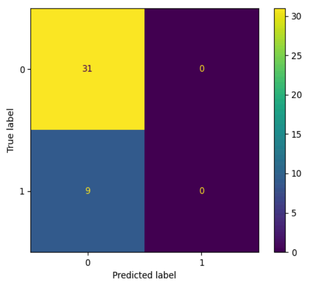
Logistic Regression: Default Dimensions, PCA, LDA

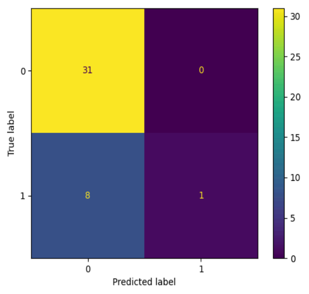


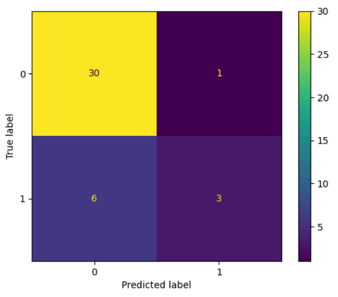




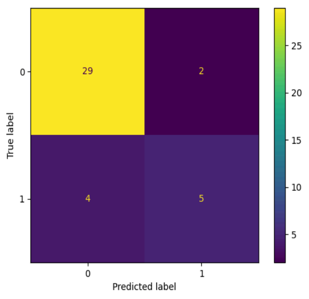
KNN: Default Dimensions, PCA, LDA

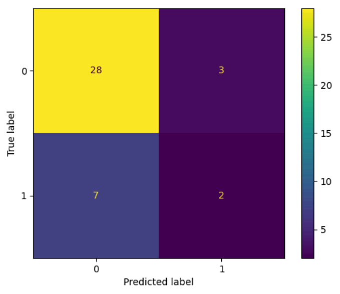


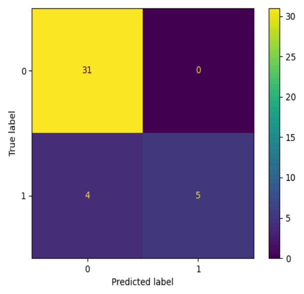




SVM: Default Dimensions, PCA, LDA







2. Full Program Code used for Calculations

```
from sklearn.model_selection import train_test_split, cross_val_score,
      GridSearchCV

from sklearn import preprocessing, decomposition

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

from sklearn.linear_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
      classification_report

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt


# Define column names based on the wpbc.names file

column_names = ["ID", "Outcome"] + ["Feature_" + str(i) for i in range(1,
      34)]


# Read the data, considering '?' as NaN

data = pd.read_csv("wpbc.data", names=column_names, sep=",",
      na_values='?')


# Convert 'Outcome' from 'N's and 'R's to '0's and '1's respectively

data['Outcome'] = data['Outcome'].map({'N': 0, 'R': 1})


# Fill missing values with mean of the column

data = data.fillna(data.mean())


# Separate features and target
```

```
x = data.drop(["ID", "Outcome"], axis=1)  # drop the ID and Outcome
      columns to get features

y = np.array(data["Outcome"])  # target is the Outcome column




# Standardize the data to ensure optimal performance in algorithms

x = preprocessing.StandardScaler().fit_transform(x)


# Using PCA (Principal component analysis) 33 features -> 10 features

pca = decomposition.PCA(n_components=10)

x_pca = pca.fit_transform(x)


# Using LDA to reduce the feature dimensionality

lda = LinearDiscriminantAnalysis()

x_lda = lda.fit_transform(x, y)




# Define hyperparameters

param_grid_knn = {'n_neighbors': np.arange(1, 51)}

param_grid_svm = {'C': [0.1, 1, 10, 100, 1000],

                  'gamma': [1, 0.1, 0.01, 0.001, 0.0001],

                  'kernel': ['rbf', 'poly', 'sigmoid']}


# Define models

lr_model = LogisticRegression(max_iter=1000)

knn_cv = GridSearchCV(KNeighborsClassifier(), param_grid_knn, refit=True)

svm_cv = GridSearchCV(SVC(), param_grid_svm, refit=True)
```

```
# Lists to store models and data
models = [lr_model, knn_cv, svm_cv]
data = [(x, y, 'Default'), (x_pca, y, 'PCA'), (x_lda, y, 'LDA')]

# Lists to store mean accuracies and standard deviations
mean_accuracies = []
std_deviations = []

# Loop over models and data
for model in models:
    model_accuracies = []
    model_stds = []
    for x, y, name in data:
        # Split data
        x_train, x_val, y_train, y_val = train_test_split(x, y,
      test_size=0.20, stratify=y, random_state=42)

        # Train model
        model.fit(x_train, y_train)

        # Predict
        predictions = model.predict(x_val)

        # Calculate cross-validation scores
        scores = cross_val_score(model, x, y)

        # Store mean accuracy and standard deviation
```

```
            model_accuracies.append(scores.mean())

            model_stds.append(scores.std() * 2)


            cm = confusion_matrix(y_val, predictions, labels=model.classes_)

            # Plot the confusion matrix

            disp = ConfusionMatrixDisplay(confusion_matrix=cm,
          display_labels=model.classes_)

            disp.plot()


            # Print results

            print(f"\nModel: {type(model).__name__}, Data: {name}")

            print("Cross-Validation Accuracy: %0.2f (+/- %0.2f)" %
          (scores.mean(), scores.std() * 2))

            print("Classification Report: \n", classification_report(y_val,
          predictions, zero_division=0))


        mean_accuracies.append(model_accuracies)

        std_deviations.append(model_stds)


# Example data

algorithms = ['Logistic Regression', 'KNN', 'SVM']

dim_reduction_techniques = ['None', 'PCA', 'LDA']


# Set up the bar plot

bar_width = 0.3

index = np.arange(len(dim_reduction_techniques))

fig, ax = plt.subplots(figsize=(10, 6))


for i, algorithm in enumerate(algorithms):
        offset = -bar_width + i * bar_width

        bars = ax.bar(index + offset, mean_accuracies[i], bar_width,
          label=algorithm, alpha=0.8)

        for j, bar in enumerate(bars):

            height = bar.get_height()

            ax.annotate(f'{mean_accuracies[i][j]:.2f} ±
          {std_deviations[i][j]:.2f}',

                        xy=(bar.get_x() + bar.get_width() / 2, height),

                        xytext=(0, 3),

                        textcoords="offset points",

                        ha='center', va='bottom')


ax.set_xlabel('Dimensional Reduction Techniques', fontsize=12)

ax.set_ylabel('Average Cross-Validation Accuracy', fontsize=12)

ax.set_title('Comparison of Algorithms and Dimensional Reduction
          Techniques', fontsize=14)

ax.set_xticks(index + bar_width / 2)

ax.set_xticklabels(dim_reduction_techniques)

ax.legend(loc=4)


plt.tight_layout()

plt.show()
```