Feature Selection and Data Classification Report

Junhao Fu
jf4519@nyu.edu
Code for this project: https://github.com/JunhaoFu/BDS_hw5/blob/bds_hw5/hw5.ipynb

1. Data Loading and Initial Checks

The dataset was loaded into a Python environment using the `pandas` library, which provides robust tools for reading and handling structured data. The `read_csv()` function was employed to import the data from a CSV file into a DataFrame, which allows for sophisticated slicing, indexing, and real-time data manipulation.

2. Missing Data Analysis

Upon successfully loading the dataset, preliminary data integrity checks were conducted, focusing primarily on the detection of missing values across different columns. This step is critical as the presence of missing values can affect the outcomes of subsequent analyses and machine learning models.

Using the combination of `.isnull()` and `.sum()` methods on the DataFrame, we were able to quantify missing data, thereby enabling targeted data cleaning strategies. The results indicated that [insert findings, e.g., several columns contained a significant number of missing entries, which will require imputation or other handling methods before further analysis].

These methods proved efficient for a rapid assessment of data quality, facilitating immediate decisions regarding data preprocessing needs. Further steps will involve detailed exploratory data analysis and deciding on the appropriate techniques for handling or imputing missing values to prepare the data for deeper analysis and modeling.

3. Transformation of Categorical Variables

To adapt our dataset for compatibility with machine learning algorithms, it was imperative to transform categorical variables into numerical formats. This transformation is crucial as it allows algorithms to perform operations on data that was initially non-numeric.

For this transformation, we employed the `LabelEncoder` from the `scikit-learn` library. Label encoding was specifically chosen for its efficiency and simplicity in converting

categories into a set of numbers. Each unique category within a variable was assigned a sequential integer.

This method was applied to several categorical columns within our dataset, such as “Race”, “Differentiate”, “Estrogen Status”, etc. Post-encoding, the categories that were originally represented as text (e.g., 'White', 'Black', 'Other'') were transformed into numerical labels (e.g., 0, 1, 2), maintaining their uniqueness but in a format amenable to algorithmic processing.

By transforming these categorical variables into numerical ones, we ensured that our dataset was fully prepared for the subsequent stages of machine learning modeling, including feature selection and model training.

4. Handling Outliers

As part of the data preprocessing phase, outlier detection and removal are critical steps that ensure the robustness and reliability of the machine learning models. Outliers can significantly skew the results of the data analysis and may lead to misleading conclusions, especially in models that are sensitive to extreme values.

For this project, we opted to use the z-score method to identify and handle outliers in the dataset. The z-score, or standard score, indicates how many standard deviations an element is from the mean of the population. A z-score provides a straightforward indication of how far from the norm a data point is.

By applying the z-score method, we enhanced the dataset's quality, which is expected to improve the subsequent analytics and predictive modeling tasks significantly. This approach has aligned our dataset more closely with the underlying assumptions required by many statistical learning methods, paving the way for more accurate and reliable outcomes.

5. Dimensionality Reduction

Dimensionality reduction is a crucial step in the preprocessing of data, especially in cases where the dataset includes a large number of variables. By reducing the number of dimensions without significant loss of information, we can decrease the computational cost and complexity of our models, and potentially enhance their performance by removing noise and redundant features.

In this project, Principal Component Analysis (PCA) was employed to reduce the dimensionality of our dataset. To determine the number of principal components to retain, we set a variance threshold of 0.95. This means that the selected components should explain at least 95% of the total variance in the dataset. This approach ensures

that we keep the most informative features, which encapsulate the majority of information present in the data.

By reducing the dimensionality from potentially hundreds of variables to just 8 principal components, we significantly simplified our model's complexity without sacrificing critical information. This reduction not only improves the efficiency of our machine learning algorithms but also helps in avoiding overfitting by minimizing the "curse of dimensionality."

6. Feature Ranking

Feature ranking with PCA results:

	Principal Component	Explained Variance Ratio	Cumulative Explained Variance
0	1	0.252178	0.252178
1	2	0.156021	0.408198
2	3	0.122975	0.531173
3	4	0.112070	0.643243
4	5	0.107593	0.750836
5	6	0.102562	0.853398
6	7	0.083838	0.937237
7	8	0.037756	0.974993

7. Models Training and Evaluation

1. KNN
2. Naive Bayes
3. Decision Tree
4. Random Forest
5. Gradient Boosting

The models were evaluated using accuracy, precision, recall, and F1 score.(KNN model was written from scratch, and other models was implemented by library)

	Model	Accuracy	Precision	Recall	F1_Score
0	KNN	0.929379	0.870466	1.000000	0.930748
1	Naive Bayes	0.666667	0.633690	0.705357	0.667606
2	Decision Tree	0.950565	0.905660	1.000000	0.950495
3	Random Forest	0.985876	0.971098	1.000000	0.985337
4	Gradient Boosting	0.899718	0.851459	0.955357	0.900421

8. Hyperparameter Tuning

We selected 2 models from part 7, Random Forest and Gradient Boosting.

For Random Forest, a grid search was implemented with following hyperparameters:

```
'n_estimators': [50, 100, 200],

'max_features': ['auto', 'sqrt', 'log2'],

'max_depth': [10, 20, 30],

'min_samples_split': [2, 5, 10]
```

The best parameters found for Random Forest were:

```
'max_depth': 20,

'max_features': 'sqrt',

'min_samples_split': 2,

'n_estimators': 50
```

For Gradient Boosting, a grid search was implemented with following hyperparameters:

```
'n_estimators': [50, 100, 200, 500],

'learning_rate': [0.01, 0.1, 0.15, 0.2],

'max_depth': [1,2, 3, 4, 5],
```

The best parameters found for Gradient Boosting were:

```
'learning_rate': 0.2,

'max_depth': 5,

'n_estimators': 500
```

The final models were:

```
'Random Forest': rf_grid.best_estimator_

'Gradient Boosting': gb_grid.best_estimator_
```

The evaluations of these 2 tuned models were:

```
Tuned Models:
               Model  Accuracy  Precision  Recall  F1 Score
0      Random Forest  0.988701   0.976744     1.0  0.988235
1  Gradient Boosting  0.984463   0.968300     1.0  0.983895
```

Both models were performing better than before.

8. Conclusion

This study utilized advanced data mining techniques to predict survival outcomes for breast cancer patients, demonstrating the power of machine learning in healthcare analytics. Initial data preprocessing—including encoding categorical variables, handling outliers, and reducing dimensionality via PCA—was critical to prepare the dataset for effective analysis.

We trained and evaluated five models(KNN, Naive Bayes, Decision Tree, Random Forest, Gradient Boosting), with notable improvements observed in the performance of Random Forest and Gradient Boosting models after hyperparameter tuning. The results indicate that our approaches were successful in accurately predicting patient survivability, the accuracy was more than 98%, validating the effectiveness of the applied methods.