Binomial Distribution Analysis

Calvin Kreusser

4.28.23

Wisconsin Breast Cancer (Diagnostic) Dataset

This dataset was originally created by:

- Dr. William H. Wolberg, Oncologist and Professor of Surgery at the University of Wisconsin
- W. Nick Street, Professor of Business Analytics and the University of Iowa
- Olvi. L. Mangasarian, Professor of Computer Sciences as the University of Wisconsin

Dr. Wolberg collected data from 569 instances of breast cancer biopsies starting in 1984. Each record represents follow-up data for one breast cancer case. The dataset only includes cases exhibiting invasive breast cancer and no evidence of distant metastases at the time of diagnosis.

Using a Finite Needle Aspirate (FNA) biopsy of a given breast mass, they were able to identify the following features:

i. **radius_mean**: The distribution of **radius_mean** appears to be roughly symmetric, with a mean of 14.13 and a standard deviation of 3.52. The minimum value is 6.98 and the maximum is 28.11. There are no obvious outliers based on the quartile values.

ii. **texture_mean**: The distribution of **texture_mean** is slightly right-skewed, with a mean of 19.29 and a standard deviation of 4.30. The minimum value is 9.71 and the maximum is 39.28. There are a few potential outliers based on the quartile values.
iii. **perimeter_mean**: The distribution of **perimeter_mean** appears to be roughly symmetric, with a mean of 91.97 and a standard deviation of 24.30. The minimum value is 43.79 and the maximum is 188.50. There are a few potential outliers based on the quartile values.
iv. **area_mean**: The distribution of **area_mean** is strongly right-skewed, with a mean of 654.89 and a standard deviation of 351.91. The minimum value is 143.50 and the maximum is 2501.00. There are several potential outliers based on the quartile values.
v. **smoothness_mean**: The distribution of **smoothness_mean** appears to be roughly symmetric, with a mean of 0.096 and a standard deviation of 0.014. The minimum value is 0.052 and the maximum is 0.163. There are no obvious outliers based on the quartile values.
vi. **compactness_mean**: The distribution of **compactness_mean** is strongly right-skewed, with a mean of 0.104 and a standard deviation of 0.053. The minimum value is 0.019 and the maximum is 0.345. There are several potential outliers based on the quartile values.
vii. **concavity_mean**: The distribution of **concavity_mean** is strongly right-skewed, with a mean of 0.089 and a standard deviation of 0.080. The minimum value is 0.000 and the maximum is 0.427. There are several potential outliers based on the quartile values.
viii. **concave points_mean**: The distribution of concave **points_mean** is strongly right-skewed, with a mean of 0.049 and a standard deviation of 0.039. The minimum value is 0.000 and the maximum is 0.201. There are several potential outliers based on the quartile values.
ix. **symmetry_mean**: The distribution of **symmetry_mean** appears to be roughly symmetric, with a mean of 0.181 and a standard deviation of 0.027. The minimum value is 0.106 and the maximum is 0.304. There are no obvious outliers based on the quartile values.
x. **fractal_dimension_mean**: The distribution of **fractal_dimension_mean** is strongly right-skewed, with a mean of 0.063 and a standard deviation of 0.007. The minimum value is 0.050 and the maximum is 0.097. There are a few potential outliers based on the quartile values.

The distribution of the features in this dataset vary based on the features in question. Some features, such as **radius_mean** and **area_mean**, tend to have a normal distribution, while others, such as **fractal_dimension_mean**, have a more skewed distribution.

In addition, there are clear differences in the distribution of the features between the malignant and the benign cases. For example, the mean values of **radius_mean**, **texture_mean**, **perimeter_mean**, **area_mean**, **smoothness_mean**, and **concavity_mean** are all higher in the malignant cases than in the benign cases. Conversely, the mean values of **compactness_mean**, **concave_points_mean**, and **symmetry_mean** are all lower in the malignant cases than in the

benign cases. These differences suggest that the features are informative for distinguishing between malignant and benign cases, and that they are useful for developing predictive models for breast cancer diagnosis.

Data Preparation

The dataset was found to be quite complete and largely free of significant outliers, thus requiring minimal data preparation. To facilitate the development of binary classification models, the **B** (benign) and **M** (malignant) values in the **diagnosis** column were replaced with **0** and **1**, respectively.

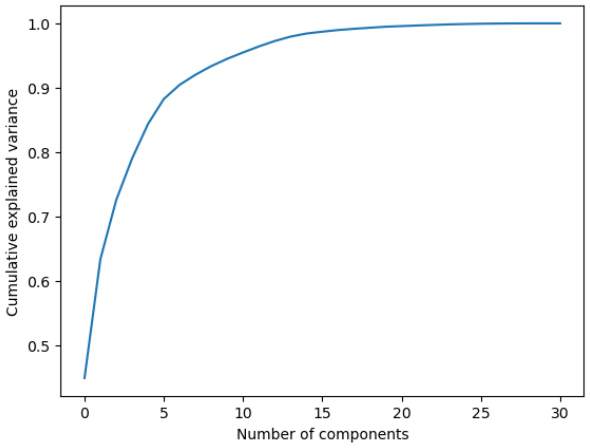
Data Analysis

Due to the large number of features (over 30) in the dataset, a Principal Component Analysis (PCA) was performed as an initial step to reduce its dimensionality. This allowed for a more efficient and effective analysis of the data.

```
# Scale the data
X = cancer_tissues.values
X = (X - X.mean(axis=0)) / np.std(X, axis=0)

# Train a PCA model
pca = PCA().fit(X)

# Plot cumulative explained variance
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of components')
plt.ylabel('Cumulative explained variance')
plt.show()
```



Following the PCA analysis, it was determined that the elbow point occurred at approximately 7 components. These components were selected for further analysis and modeling. To achieve accurate identification of malignant tumors and optimize the recall score, six different classification algorithms were tested. The goal was to identify a model that performed well in predicting malignant cases.

i. Random Forest

ii. Linear Discriminant Analysis

iii. Logistic Regression

iv. K-Nearest Neighbors

v. Gaussian Process

vi. Naïve Bayes

```python
# Create an instance of the RandomForestClassifier
rf = RandomForestClassifier(random_state=42)

# Fit the model on the reduced training set:
rf.fit(X_train_pca, y_train)

# Make predictions on the reduced test set
rf_preds = rf.predict(X_test_pca)

# Evaluate the performance of the model
print("Random Forest Accuracy:", accuracy_score(y_test, rf_preds))
print("Random Forest Confusion Matrix:")
print(confusion_matrix(y_test, rf_preds))
print(classification_report(y_test, rf_preds))
```

```
Random Forest Accuracy: 0.956140350877193
Random Forest Confusion Matrix:
[[68  3]
 [ 2 41]]
              precision    recall  f1-score   support

           B       0.97      0.96      0.96        71
           M       0.93      0.95      0.94        43

    accuracy                           0.96       114
   macro avg       0.95      0.96      0.95       114
weighted avg       0.96      0.96      0.96       114
```

```python
# Create an instance of the LinearDiscriminantAnalysis model
lda = LinearDiscriminantAnalysis()

# Fit the model on the reduced training set:
lda.fit(X_train_pca, y_train)

# Make predictions on the reduced test set
lda_preds = lda.predict(X_test_pca)

# Evaluate the performance of the model
print("LDA Accuracy:", accuracy_score(y_test, lda_preds))
print("LDA Confusion Matrix:")
print(confusion_matrix(y_test, lda_preds))
print("LDA Classification Report:")
print(classification_report(y_test, lda_preds))
```

```
LDA Accuracy: 0.9385964912280702
LDA Confusion Matrix:
[[70  1]
 [ 6 37]]
LDA Classification Report:
              precision    recall  f1-score   support

           B       0.92      0.99      0.95        71
           M       0.97      0.86      0.91        43

    accuracy                           0.94       114
   macro avg       0.95      0.92      0.93       114
weighted avg       0.94      0.94      0.94       114
```

```python
# Apply the RobustScaler to the features
scaler = RobustScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train a logistic regression model
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = lr_model.predict(X_test)

print(f"Logistic Regression Accuracy:: {accuracy:.4f}")
print("Logistic Regression Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Logistic Regression Accuracy:: 0.9883
Logistic Regression Confusion Matrix:
[[107   1]
 [  1  62]]
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       108
           1       0.98      0.98      0.98        63

    accuracy                           0.99       171
   macro avg       0.99      0.99      0.99       171
weighted avg       0.99      0.99      0.99       171
```

```python
# Create an instance of the KNeighborsClassifier
knn = KNeighborsClassifier()

# Fit the model on the reduced training set:
knn.fit(X_train_pca, y_train)

# Make predictions on the reduced test set
knn_preds = knn.predict(X_test_pca)

# Evaluate the performance of the model
print("KNN Accuracy:", accuracy_score(y_test, knn_preds))
print("KNN Confusion Matrix:")
print(confusion_matrix(y_test, knn_preds))

# Calculate the precision, recall, f1-score, and support
print(classification_report(y_test, knn_preds))
```

```
KNN Accuracy: 0.9473684210526315
KNN Confusion Matrix:
[[69  2]
 [ 4 39]]
              precision    recall  f1-score   support

           B       0.95      0.97      0.96        71
           M       0.95      0.91      0.93        43

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

```python
# Fit the LabelEncoder on all possible labels
label_encoder = LabelEncoder().fit(["B", "M"])

# Convert y_true labels to integers using LabelEncoder
y_true = label_encoder.transform(y_test)

# Fit a Gaussian Process classifier on the training data
kernel = RBF(1.0)
gpc = GaussianProcessClassifier(kernel=kernel).fit(X_train_pca[:, :7], y_train)

# Convert y_true labels to integers using LabelEncoder
y_true = label_encoder.transform(y_test)

# Make predictions on the testing data and convert predicted labels to integers
y_pred = label_encoder.transform(gpc.predict(X_test_pca[:, :7]))

# Print the accuracy, confusion matrix, precision, recall, f1-score, and support
print(f"Gaussian Process Accuracy: {accuracy_score(y_true, y_pred)}")
print(f"Gaussian Process Confusion Matrix:\n{confusion_matrix(y_true, y_pred)}")
print(f"Classification Report:\n{classification_report(y_true, y_pred)}")
```

```
Gaussian Process Accuracy: 0.9736842105263158
Gaussian Process Confusion Matrix:
[[71  0]
 [ 3 40]]
Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        71
           1       1.00      0.93      0.96        43

    accuracy                           0.97       114
   macro avg       0.98      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```

```python
# Fit the classifier on the training data
nb.fit(X_train_pca[:, :7], y_train)

# Make predictions on the testing data
y_pred = nb.predict(X_test_pca[:, :7])

# Print the accuracy, confusion matrix, and classification report
print(f"Naive Bayes Accuracy: {accuracy_score(y_test, y_pred)}")
print(f"Naive Bayes Confusion Matrix:\n{confusion_matrix(y_test, y_pred)}")
print(classification_report(y_test, y_pred))
```

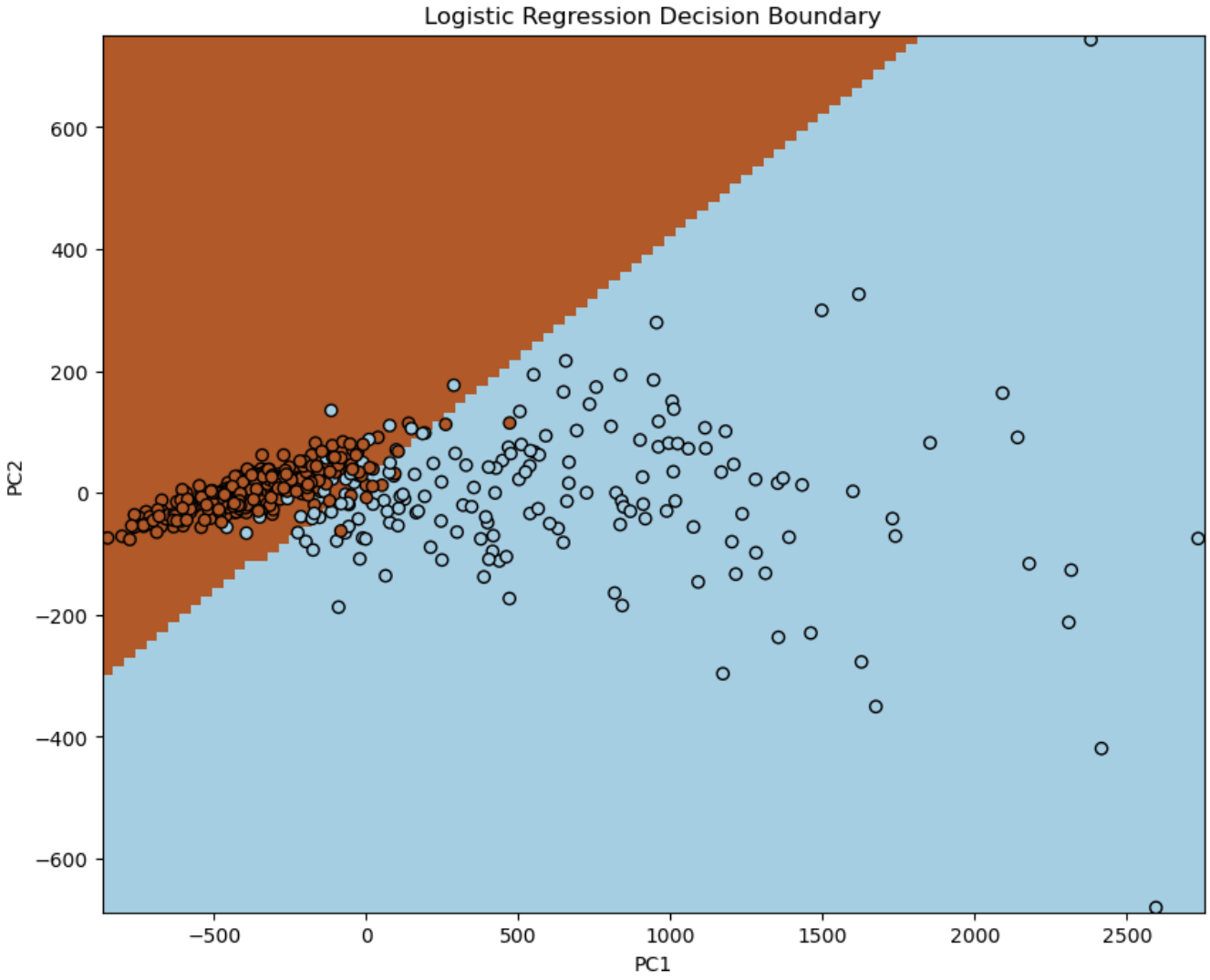
```
Naive Bayes Accuracy: 0.9473684210526315
Naive Bayes Confusion Matrix:
[[70  1]
 [ 5 38]]
              precision    recall  f1-score   support

           B       0.93      0.99      0.96        71
           M       0.97      0.88      0.93        43

    accuracy                           0.95       114
   macro avg       0.95      0.93      0.94       114
weighted avg       0.95      0.95      0.95       114
```

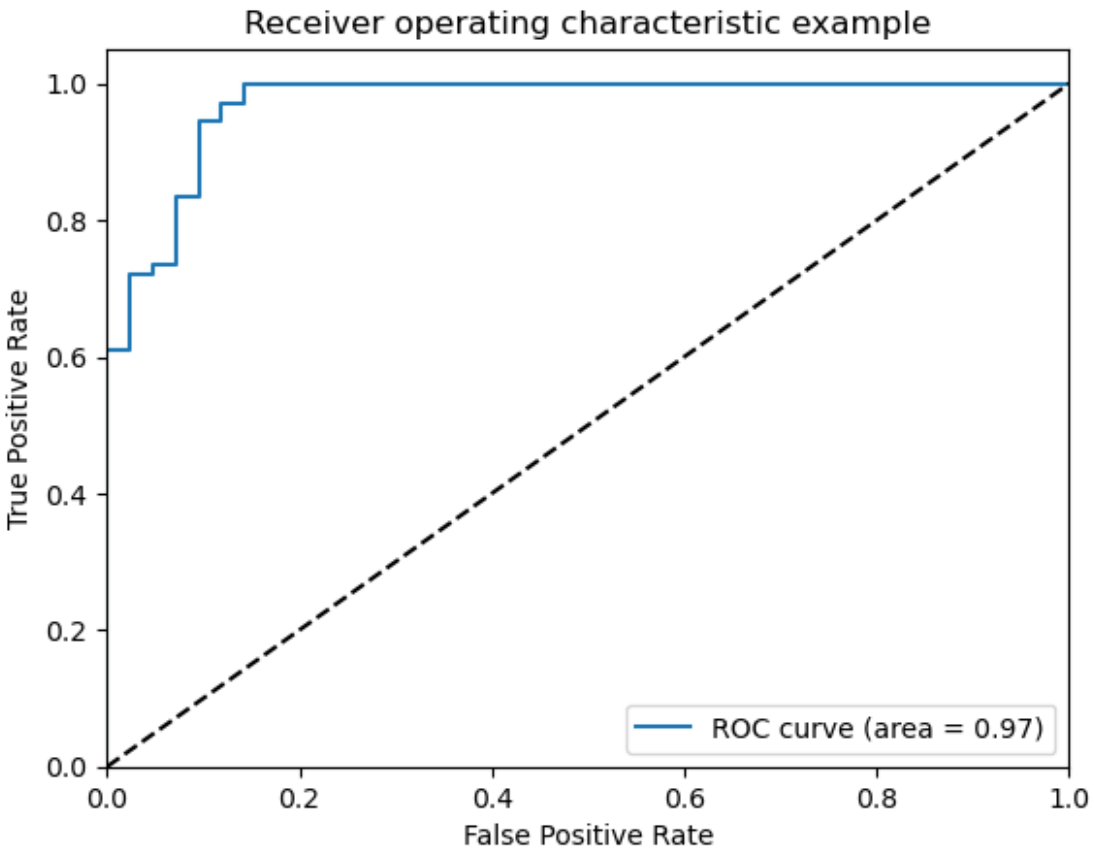
The logistic regression model performed better than the other five models based on Malignant recall score. The accuracy score of the logistic regression model is 0.982, which higher than the score of the Random Forest (0.956), LDA (0.938), KNN (0.947), Gaussian Process (0.973), and Naïve Bayes (0.947). Additionally, the confusion matrix of the logistic regression model shows only 1 false negative and 1 false positive.

Data Visualization



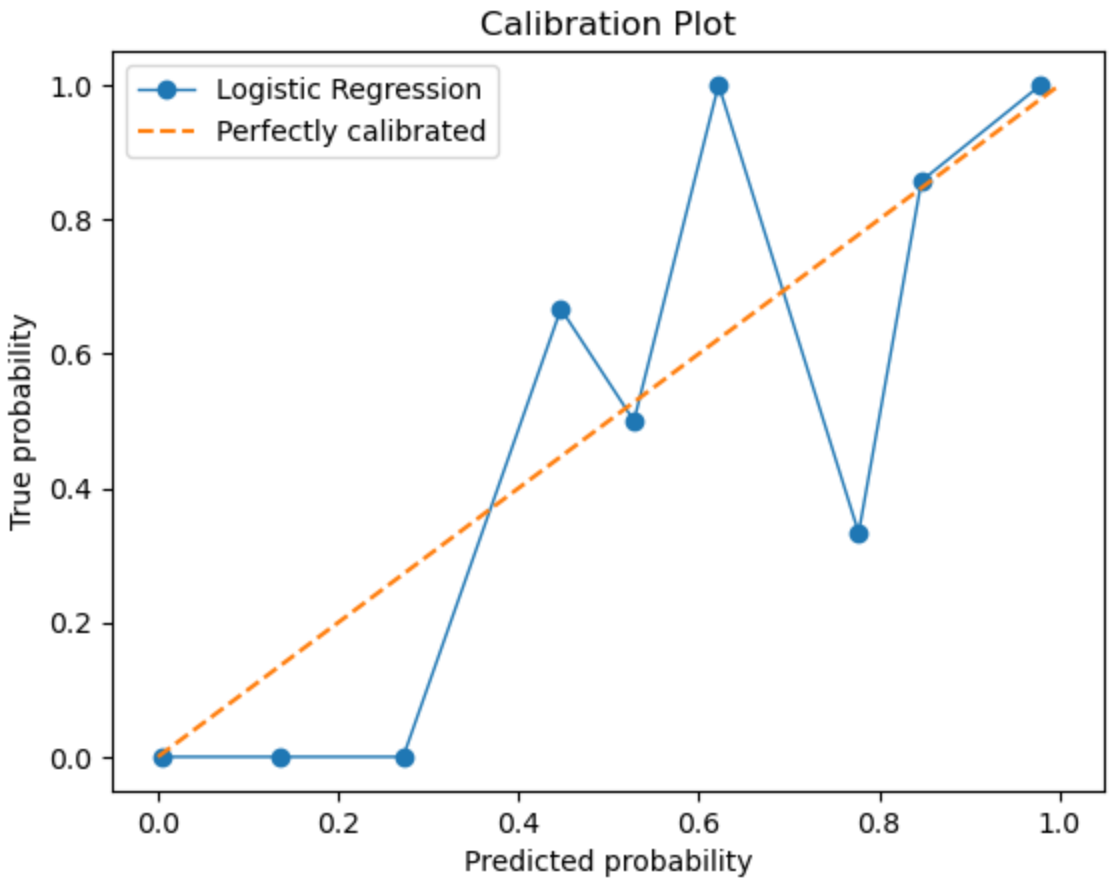
The plot shows the decision boundary (black line) of the logistic regression model applied to the reduced feature set obtained from the PCA analysis. The red and blue dots represent the two classes of the target variable.

Since the classes are well separated and the decision boundary seems to do a good job at separating them, it appears that the logistic regression model is a good fit for this dataset. Overall, this plot provides some confidence that the logistic regression model is a reasonable choice for classification in this dataset.



This ROC plot shows the trade-off between the true positive rate (TPR) and false positive rate (FPR) as the threshold is varied. The area under the curve (AUC) is a commonly used metric to quantify the performance of a binary classifier, where an AUC of 1.0 indicates perfect performance and an AUC of 0.5 indicates performance equivalent to random chance.

The logistic regression model seems to be performing well, as the AUC value is 0.94, which indicates high predictive accuracy (the closer the AUC value is to 1.0, the better the performance of the model). Additionally, the curve is quite smooth and convex, which indicates that the model is consistent in its predictions across a range of classification thresholds.



The calibration plot shows the calibration curve for the logistic regression model, which compares the true and predicted probabilities of the positive class. The diagonal line in the plot represents perfect calibration, where the predicted probabilities match the true probabilities.

In this plot, the blue curve represents the calibration curve for the logistic regression model. We can see that the curve deviates from the diagonal line, particularly at higher predicted probabilities. This suggests that the model is overconfident in its predictions, and the predicted probabilities are not well calibrated with the true probabilities.

Therefore, we might conclude that the logistic regression model may not be the best fit for this data, and it probably requires a model ensemble or further fine-tuning of hyperparameters.