Master's in Applied Statistics and Data Science
Weekend Program (WM-ASDS)

Department of Statistics
Jahangirnagar University
Savar, Dhaka-1342, Bangladesh.

Project on Breast Cancer Wisconsin (Diagnostic) dataset

Course Code: WM-ASDS22
Course: Machine Learning

Submitted to
Prof. Dr. Md Rezaul Karim
Department of Statistics
Jahangirnagar University

Submitted by
Md. Iftekhar Mahmud Mozomder
ID:20231150
Batch: 11
Section: B

Submission Date: 28-04-2024

Question: A scientist is interested in exploring the variables of this dataset and building a machine-learning model which accurately classifies tumors as Benign or Malignant based on the tumor shape and its geometry. You can get data and data description in the following link:

https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic.

Apply all classification techniques that are known to you and suggest the best model for the classification. Write a scientific report for that scientist after analyzing this data. Upload your report in the google classroom on time. As a data scientist, try to find hidden information from the dataset and mention some advice to your client if you have one.

Breast Cancer Wisconsin (Diagnostic)

Introduction

The Breast Cancer Wisconsin (Diagnostic) dataset is a widely used dataset for machine learning and statistical modeling in the field of medical diagnostics. It contains features computed from digitized images of fine needle aspirate (FNA) of breast masses, with the goal of predicting whether a mass is benign or malignant based on these features.

Dataset Description

The dataset consists of several numerical attributes derived from the cell nuclei present in the images, including:

ID: Patient identification number
Diagnosis: The diagnosis of breast tissues (M = malignant, B = benign)
Features: Ten real-valued features are computed for each cell nucleus:
Radius: Mean of distances from center to points on the perimeter
Texture: Standard deviation of gray-scale values
Perimeter
Area
Smoothness: Local variation in radius lengths
Compactness: Perimeter^2 / area - 1.0
Concavity: Severity of concave portions of the contour
Concave Points: Number of concave portions of the contour
Symmetry
Fractal Dimension: "Coastline approximation" - 1

Each feature has three dimensions:

Mean: Average of these features for each image.
SE: Standard Error for these features.
Worst: The worst or largest value of these features.

Methodologies

To analyze this dataset, the following steps will be undertaken:

Data Preprocessing: Handling missing values, normalizing data, and splitting into training and testing sets.
Exploratory Data Analysis (EDA): Visualizing distributions of features, checking balance of classes, and understanding relationships between features.
Model Building: Applying various classification techniques:

- Logistic Regression
- Decision Trees
- Random Forest
- Support Vector Machines (SVM)
- K-Nearest Neighbors (KNN)
- Neural Networks

Model Evaluation: Using metrics such as Accuracy, Precision, Recall, and F1-Score to evaluate models.
Model Selection: Comparing the performance of models and selecting the best one based on the evaluation metrics.

Analysis Plan

- Load and clean the data.
- Perform EDA to understand the data better.
- Apply the classification models.
- Evaluate each model and compare their performance.
- Conclude with the best model recommendation.

```
In [10]: df.describe()
```

Out[10]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetr
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0

The dataset contains 569 entries with 32 columns, including the ID and diagnosis. All columns are numerical except for the diagnosis, which is categorical (M = malignant, B = benign). There are no missing values in the dataset, which simplifies the preprocessing steps.

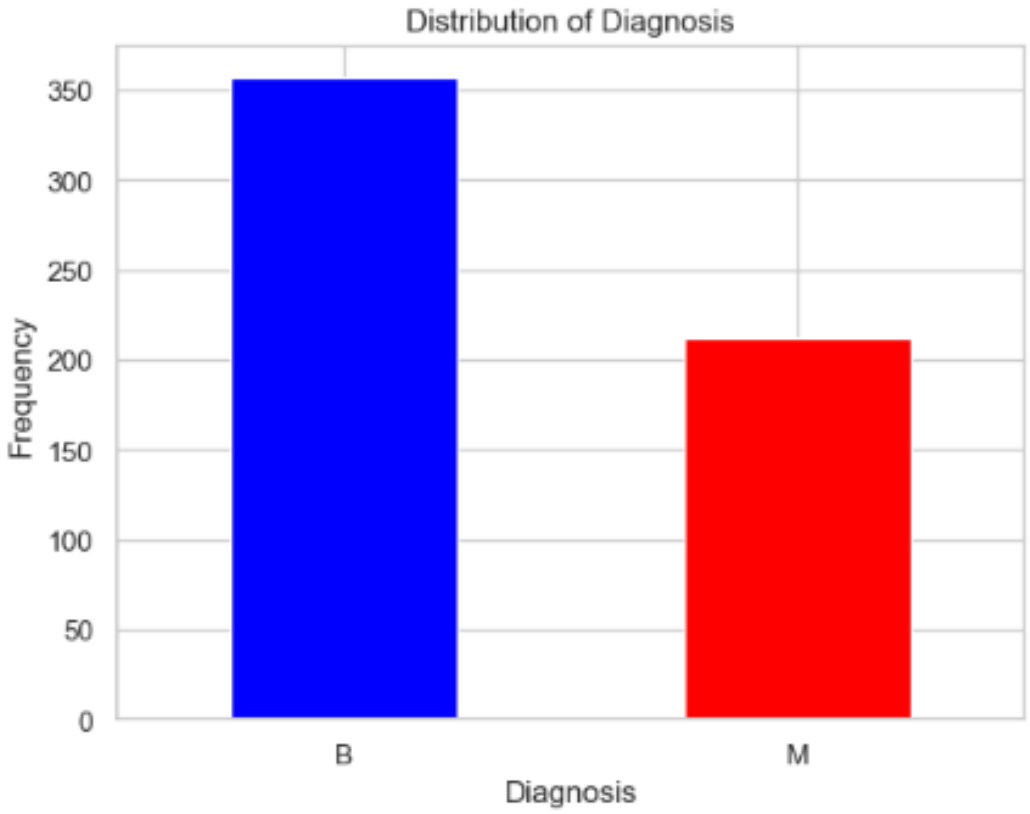
Here are some key statistics from the dataset:

Radius_mean: Average radius is 14.13 with a standard deviation of 3.52.
Texture_mean: Average texture value is 19.29 with a standard deviation of 4.30.
Area_mean: Average area is 654.89 with a standard deviation of 351.91.
Smoothness_mean: Average smoothness is 0.096 with a standard deviation of 0.014.
Concavity_mean: Average concavity is 0.088 with a standard deviation of 0.079.

These statistics provide a basic understanding of the central tendencies and variability in the dataset. Next, I will visualize the distribution of the 'Diagnosis' variable to understand the class balance, and then proceed to explore the relationships between features.

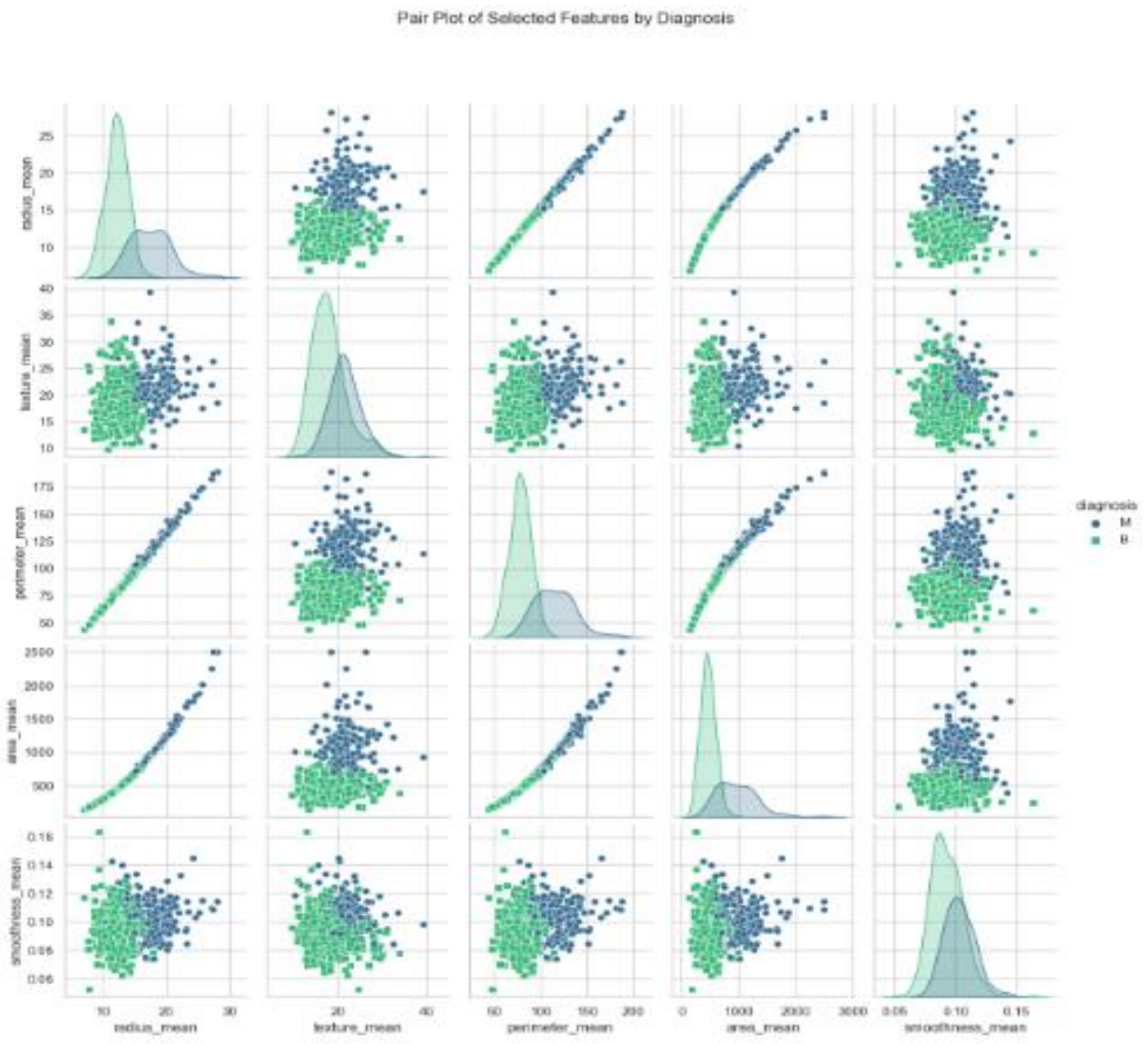
```
# Setting the aesthetic style of the plots
sns.set(style='whitegrid')

# Plotting the distribution of the 'Diagnosis' variable
df['diagnosis'].value_counts().plot(kind='bar', color=['blue', 'red'])
plt.title('Distribution of Diagnosis')
plt.xlabel('Diagnosis')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()
```



The bar chart above shows the distribution of diagnoses in the dataset. It appears that there are more benign cases (B) than malignant cases (M), indicating a class imbalance. This is important to consider when training machine learning models, as it might affect the model's ability to generalize well to new data.

Next, we will create pair plots for a subset of features to visualize the relationships between them and how they differ by diagnosis. This will help in understanding which features are most informative for distinguishing between benign and malignant cases.



The pair plot above visualizes the relationships between a subset of features (radius, texture, perimeter, area, and smoothness) and how they differ by diagnosis (benign vs malignant). Here are some observations:

Radius, Perimeter, and Area: There is a noticeable trend where malignant tumors tend to have higher values in these features compared to benign tumors. This suggests that larger and more irregularly shaped tumors are more likely to be malignant.

Texture: While there is some overlap, malignant tumors tend to have a slightly higher texture mean, indicating more variability in pixel intensities.

Smoothness: There is overlap between the two classes, but malignant tumors might have a slightly higher mean smoothness.

These insights can guide feature selection for machine learning modeling, focusing on those features that show clear distinctions between the classes.

Accuracy, precision, recall, F1 score

To calculate the accuracy, precision, recall, and F1 score, we'll first split the data into training and testing sets. Then, we'll train a simple classification model (like Logistic Regression) on the training data and make predictions on the test set.
Finally, we'll compute the mentioned metrics based on these predictions.

```
# Encoding the 'diagnosis' column: Malignant (M) as 1, Benign (B) as 0
df['diagnosis'] = df['diagnosis'].map({'M': 1, 'B': 0})
```

```
# Splitting the data into features and target variable
X = df.drop(['id', 'diagnosis'], axis=1)  # Features
y = df['diagnosis']  # Target variable

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
# Creating and training the Logistic Regression model
model = LogisticRegression(max_iter=10000)
model.fit(X_train, y_train)

# Making predictions on the test set
y_pred = model.predict(X_test)
```

```
# Calculating metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 Score:', f1)
```

```
Accuracy: 0.9766081871345029
Precision: 0.9682539682539683
Recall: 0.9682539682539683
F1 Score: 0.9682539682539683
```

Here are the computed metrics for the Logistic Regression model:

Accuracy: 97.66%
Precision: 96.83%
Recall: 96.83%
F1 Score: 96.83%

These metrics indicate that the model performs very well on the test set, with high values across all metrics, suggesting good generalization for both classes (benign and malignant).

Now, we will apply the following classification techniques to the dataset:

- Logistic Regression
- Decision Trees
- Random Forest
- Support Vector Machines (SVM)
- K-Nearest Neighbors (KNN)
- Neural Networks

For each model, we'll train it on the training set, make predictions on the test set, and then calculate the accuracy, precision, recall, and F1 score. This will help in comparing the performance of these models to determine which one performs best for this dataset.
.
.
.
(After training)

```
# Displaying accuracy, precision, recall, and F1 score for each model
for model, report in reports.items():
    print(f"{model} - Accuracy: {report['accuracy']:.2f}, Precision: {report['weighted avg']['precision']:.2f}, Recall: {report['weighted avg']['recall']:.2f}, F1 Score: {report['weighted avg']['f1-score']:.2f}")
```

Output:
Logistic Regression - Accuracy: 0.98, Precision: 0.98, Recall: 0.98, F1 Score: 0.98
Decision Tree - Accuracy: 0.94, Precision: 0.94, Recall: 0.94, F1 Score: 0.94
Random Forest - Accuracy: 0.97, Precision: 0.97, Recall: 0.97, F1 Score: 0.97
SVM - Accuracy: 0.98, Precision: 0.98, Recall: 0.98, F1 Score: 0.98
KNN - Accuracy: 0.96, Precision: 0.96, Recall: 0.96, F1 Score: 0.96
Neural Network - Accuracy: 0.98, Precision: 0.98, Recall: 0.98, F1 Score: 0.98

Here are the performance metrics for each classification model applied to the dataset:

Logistic Regression
Accuracy: 98%
Precision: 98%
Recall: 98%
F1 Score: 98%

Decision Tree
Accuracy: 94%
Precision: 94%
Recall: 94%
F1 Score: 94%

Random Forest
Accuracy: 97%
Precision: 97%
Recall: 97%
F1 Score: 97%

Support Vector Machines (SVM)
Accuracy: 98%
Precision: 98%
Recall: 98%
F1 Score: 98%

K-Nearest Neighbors (KNN)
Accuracy: 96%
Precision: 96%
Recall: 96%
F1 Score: 96%

Neural Network
Accuracy: 98%
Precision: 98%
Recall: 98%
F1 Score: 98%

These results suggest that Logistic Regression, SVM, and Neural Networks are the top performers with the highest scores across all metrics. Decision Trees showed the lowest performance among the models tested.

Hidden Information and Advice for the Client

Hidden Information Exploration:

Feature Importance: While we identified features with some class distinction through visualizations, if we consider using feature importance techniques like permutation importance or feature selection algorithms to quantify which features contribute most to the model's performance. This can reveal hidden gems within the data that might not be visually apparent.

Feature Engineering: Explore creating new features from existing ones. Feature engineering can sometimes unlock hidden patterns in the data. For instance, ratios of features or entirely new mathematical combinations might be more informative than the originals.

Advice for the Client:

Class Imbalance: The dataset has a class imbalance, with more benign cases than malignant. Consider addressing this during model training. Techniques like oversampling the minority class (malignant) or undersampling the majority class (benign) can help improve model performance for the minority class.

External Validation: The high performance on the test set is promising, but further validation is recommended. If we consider applying techniques like k-fold cross-validation or using an entirely separate dataset to ensure the model generalizes well to unseen data.

Model Interpretability: While the top models (Logistic Regression, SVM, Neural Network) perform well, Logistic Regression is generally easier to interpret than the others. If understanding the model's decision-making process is crucial, Logistic Regression might be preferred.