NAME: Atharva Rajbanshi
NJIT UCID: ar2699
Email Address: ar2699@njit.edu
04/21/2024
Professor: Yasser Abduallah
CS 634 104 Data Mining

https://github.com/rajbanshiatharva/DS634_Final

Final Project Report

Implementation and Code Usage

Abstract:

Exploring the realm of binary classification opens up avenues for discerning patterns and characteristics in complex data sets, particularly when determining the potability of water. This project will delve into the comparative analysis of several machine learning approaches, including the K Neighbour's classifier, Gaussian NB, Random Forest, and Bidirectional LSTM. Let's dive into the core concepts and methodologies behind this endeavour.

Project Objective:

The primary objective is to classify water samples into two categories: potable and non-potable. This classification helps ensure the assessment of water safety and quality, which is essential for public health and environmental management.

Machine learning models

1. **K Neighbours Classifier:**
   - It uses the k-nearest neighbors algorithm to perform classification based on similarity measure.
   - Effective for small to medium-sized datasets and simple decision boundaries.
   - The k parameter controls the number of neighbors considered during classification.

2. **Gaussian Naïve Bayes:**
   - A probabilistic classifier based on Bayes theorem and Gaussian distribution assumption for input characters.
   - Especially useful for datasets with continuous features and limited training data.
   - It provides fast training and prediction times, making it suitable for real-time applications.

3. **Random forest:**
   - It uses a set of decision trees to perform the classification.

- Known for its robustness and efficiency in handling large datasets with high dimensionality.
- Ability to capture complex relationships between input functions and target variable.

4. **Bidirectional LSTM:**
   - A variant of the Long Short-Term Memory (LSTM) neural network architecture.
   - Designed to efficiently model sequential data using both past and future information.
   - Particularly adept at capturing temporal dependencies and patterns in time series data.

Project workflow:

1. **Data preprocessing:**
   - Load and pre-process water quality data, including features such as pH, hardness and chloramines.
   - Treat missing values, outliers, and data normalization to ensure model compatibility.

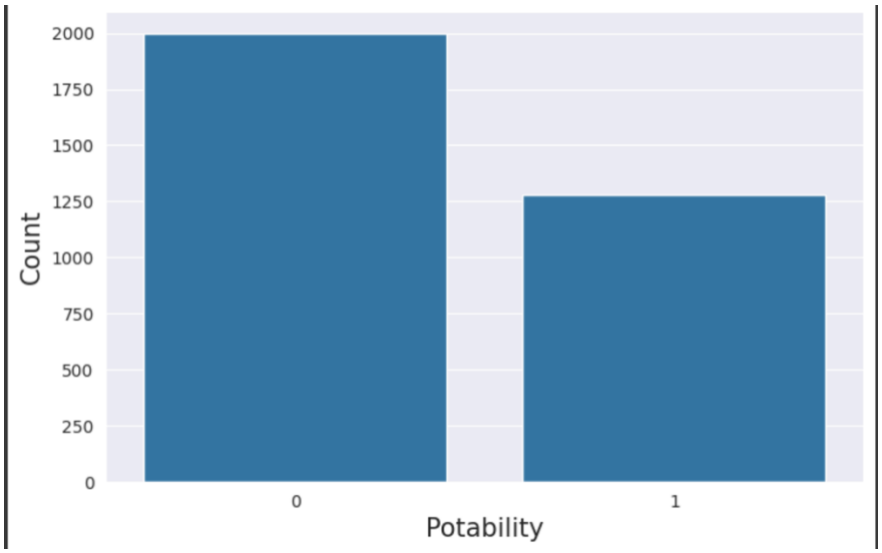

Fig.1 Result Classes

Classes 0 (Not potable) and 1 (Potable) are almost balanced. Thus, no upsampling or downsampling.

```
] ''' checking null values '''
  df.isnull().sum()

ph                491
Hardness            0
Solids              0
Chloramines         0
Sulfate           781
Conductivity        0
Organic_carbon      0
Trihalomethanes   162
Turbidity           0
Potability          0
dtype: int64
```

Null values present.

Therefore, all null values are replaced with Median Values.

```
''' preparing data for model '''

ph_median = df[df['Potability'] == 0]['ph'].median(skipna=True)
df.loc[(df['Potability'] == 0) & (df['ph'].isna()), 'ph'] = ph_median

ph_median_1 = df[df['Potability'] == 1]['ph'].median(skipna=True)
df.loc[(df['Potability'] == 1) & (df['ph'].isna()), 'ph'] = ph_median_1

sulf_median = df[df['Potability'] == 0]['Sulfate'].median(skipna=True)
df.loc[(df['Potability'] == 0) & (df['Sulfate'].isna()), 'Sulfate'] = sulf_median

sulf_median_1 = df[df['Potability'] == 1]['Sulfate'].median(skipna=True)
df.loc[(df['Potability'] == 1) & (df['Sulfate'].isna()), 'Sulfate'] = sulf_median_1

traih_median = df[df['Potability'] == 0]['Trihalomethanes'].median(skipna=True)
df.loc[(df['Potability'] == 0) & (df['Trihalomethanes'].isna()), 'Trihalomethanes'] = traih_median

trah_median_1 = df[df['Potability'] == 1]['Trihalomethanes'].median(skipna=True)
df.loc[(df['Potability'] == 1) & (df['Trihalomethanes'].isna()), 'Trihalomethanes'] = trah_median_1
```

```
] df.isnull().sum()

ph                 0
Hardness           0
Solids             0
Chloramines        0
Sulfate            0
Conductivity       0
Organic_carbon     0
Trihalomethanes    0
Turbidity          0
Potability         0
dtype: int64
```

2. **Model training:**
   - Implement each model using popular machine learning frameworks such as scikit-learn and TensorFlow/Keras.
   - Split the data set into training and test sets for model evaluation.
   - Train each model with the training data and evaluate its performance on the test set.

```
print("X_train shape: ", X_train.shape)
print("X_test shape: ", X_test.shape)
```

```
X_train shape:  (2620, 9)
X_test shape:  (656, 9)
```

Feature Scaling done using Standard Scaler.

```
''' standard scaler '''
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

3. Performance Rating:

Assess model performance using metrics such as True Positive Rate (TPR), Specificity (SPC), Precision (PPV) and others specified.

```
#Defining Metrics
def calculate_performance_metrics(y_true, y_pred):
    conf_matrix = confusion_matrix(y_true, y_pred)
    TP = conf_matrix[1, 1]
    TN = conf_matrix[0, 0]
    FP = conf_matrix[0, 1]
    FN = conf_matrix[1, 0]

    TPR = TP / (TP + FN)  # True Positive Rate or Sensitivity
    SPC = TN / (TN + FP)  # Specificity
    PPV = TP / (TP + FP)  # Positive Predictive Value or Precision
    NPV = TN / (TN + FN)  # Negative Predictive Value
    FPR = FP / (FP + TN)  # False Positive Rate
    FDR = FP / (FP + TP)  # False Discovery Rate
    FNR = FN / (FN + TP)  # False Negative Rate
    ACC = (TP + TN) / (TP + TN + FP + FN)  # Accuracy
    F1 = 2 * (TP) / (2*(TP) + FP + FN)  # F1 Score
    BACC = (TPR + SPC) / 2  # Balanced Accuracy
    TSS = TPR - FPR  # True Skill Statistics
    HSS = (2 * (TP * TN - FP * FN)) / ((TP + FN) * (FN + TN) + (TP + FP) * (FP + TN))  # Heidke Skill Score
```

1. **True Positive Rate (TPR):**

- Also known as sensitivity or recall, TPR measures the proportion of actual positive cases that were correctly identified by the model.

  TPR = TP / (TP + FN)

2. **Specificity (SPC):**

   - Specificity measures the proportion of actual negative cases that were correctly identified by the model.

     TN / (TN + FP)

3. **Precision (PPV):**

   - Precision measures the proportion of true positive predictions among all positive predictions made by the model.

     TP / (TP + FP)

4. **Negative Predictive Value (NPV):**

   - NPV measures the proportion of true negative predictions among all negative predictions made by the model.

     NPV = TN / (TN + FN)

5. **False Positive Rate (FPR):**

   - FPR measures the proportion of actual negative cases that were incorrectly classified as positive by the model.

     FPR = FP / (FP + TN)

6. **False Discovery Rate (FDR):**

   - FDR measures the proportion of false positive predictions among all positive predictions made by the model.

     FDR = FP / (FP + TP)

7. **False Negative Rate (FNR):**

   - FNR measures the proportion of actual positive cases that were incorrectly classified as negative by the model.

     FNR = FN / (FN + TP)

8. **Accuracy (ACC):**

   - Accuracy measures the overall correctness of the model's predictions, calculated as the ratio of correctly classified cases to the total number of cases.

     ACC = (TP + TN) / (TP + TN + FP + FN)

9. **F1 Score:**

    - The F1 score is the harmonic mean of precision and recall, providing a balance between the two metrics. It is particularly useful when the classes are imbalanced.

      F1 = 2 * TP / (2*TP + FP + FN)

10. **Balanced Accuracy (BACC):**

    - BACC is the arithmetic mean of sensitivity (TPR) and specificity (SPC), providing a balanced evaluation of the model's performance across both classes.

      BACC = (TPR + SPC) / 2

11. **True Skill Statistics (TSS):**

    - TSS measures the discriminatory power of the model by subtracting the false positive rate from the true positive rate.

      TSS = TPR - FPR

12. **Heidke Skill Score (HSS):**

    - HSS measures the skill of the model compared to random chance, taking into account correct and incorrect classifications.

      HSS = (2 * (TP * TN - FP * FN)) / ((TP + FN) * (FN + TN) + (TP + FP) * (FP + TN))

Conclusion:

In conclusion, this project showcases the effectiveness of various machine learning models in the binary classification of water potability. Each model, including K Neighbours Classifier, Gaussian NB, Random Forest, and Bidirectional LSTM, offers unique advantages and performance characteristics. The comparative analysis provides valuable insights into their suitability for water quality assessment and management.

The source code (.py file) and data sets (.csv files) will be attached to the zip file. *Link to Git Repository*

https://github.com/rajbanshiatharva/DS634_Final