AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

408/1, Kuratoli, Khilkhet, Dhaka 1229,
Bangladesh

Assignment Title: Stress Level Prediction in Students: Gaussian Naive Bayes

Assignment No: Final Term

Date of Submission: 13/5/2024

Course Title: Programming in Python

Course Code: CSC4162

Section: B

Semester: Spring 2023-24

Course Teacher: DR. ABDUS SALAM

Declaration and Statement of Authorship:

1. I/we hold a copy of this Assignment/Case-Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case-Study is my/our original work and no part of it has been copied from any other student's work or from any other source except where due acknowledgement is made.
3. No part of this Assignment/Case-Study has been written for me/us by any other person except where such collaborationhas been authorized by the concerned teacher and is clearly acknowledged in the assignment.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared and archived for the purpose of detecting plagiarism.
6. I/we give permission for a copy of my/our marked work to be retained by the Faculty for review and comparison, including review by external examiners.
7. I/we understand thatPlagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a formofcheatingandisaveryseriousacademicoffencethatmayleadtoexpulsionfromtheUniversity. Plagiarized material can be drawn from, and presented in, written, graphic and visual form, including electronic data, and oral presentations. Plagiarism occurs when the origin of them arterial used is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or to copy my/our work.

* *Student(s) must complete all details except the faculty use part.*
** Please submit all assignments to your course teacher or the office of the concerned teacher.

Group Name/No.: 4

No	Name	ID	Program	Signature
1	PRITOM CHANDRA DEY	21-44407-1	BSc. CSE	
2	MD. FARHAN SADIK	21-44403-1	BSc. CSE	
3				
4				

5				
6				
7				
8				
9				
10				

Faculty use only		
FACULTYCOMMENTS		
	Marks Obtained	
	Total Marks	

Description of the Dataset:
This dataset contains around 20 features that create the most impact on the Stress of a Student. The features are selected scientifically considering 5 major factors, they are Psychological, Physiological, Social, Environmental, and Academic Factors. Some of them are:
Psychological Factors => 'anxiety_level', 'self_esteem', 'mental_health_history', 'depression',
Physiological Factors => 'headache', 'blood_pressure', 'sleep_quality', 'breathing_problem
Environmental Factors => 'noise_level', 'living_conditions', 'safety', 'basic_needs',
Academic Factors => 'academic_performance', 'study_load', 'teacher_student_relationship', 'future_career_concerns',
Social Factor => 'social_support', 'peer_pressure', 'extracurricular_activities', 'bullying'

These features collectively capture various aspects of a student's life and experiences, providing insights into factors that may impact their stress levels. Organizing the features by factors helps in understanding their relevance and potential interactions within each category.

Import necessary libraries:

```
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix, precision_score
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.feature_selection import VarianceThreshold
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

Task 1: To read and load the dataset file into the program, the Pandas library is utilized. The code segment below demonstrates how this task is implemented.

```
path ="/content/drive/MyDrive/StressLevelDataset.csv"
df=pd.read_csv(path)
```

This code imports the Pandas library and uses the read_csv() function to load the dataset located at the specified file path into a Pandas Data Frame named df.

Task 2: Data cleaning techniques are applied to the dataset to ensure its quality and integrity. The following code segment demonstrates how this task is accomplished using the Pandas library:

```
missing_values = df.isnull().sum()
print("\nMissing Values:")
print(missing_values)
df_filtered = df.drop_duplicates()
print(df_filtered)
target_column = 'stress_level'
correlation_matrix = df_filtered.corr()
target_correlations = correlation_matrix[target_column].abs()
highly_correlated_features_target =
target_correlations[target_correlations > 0.7].index
```

```
highly_correlated_features = set()
for col in correlation_matrix.columns:
    correlated_cols = correlation_matrix.index[correlation_matrix[col] >
0.7].tolist()
    highly_correlated_features.update(correlated_cols)
features_to_remove = list(highly_correlated_features -
set(highly_correlated_features_target))
df_filtered = df_filtered.drop(columns=features_to_remove)
```

This code snippet first checks for missing values in the dataset, then removes duplicate records while preserving the original dataset. Next, it calculates the correlation matrix to identify highly correlated features. Features with a correlation coefficient greater than 0.7 are considered highly correlated. Finally, it removes highly correlated features, ensuring that only relevant features are retained for further analysis.

Task 3: Graphs are drawn to analyze the frequency distributions of the features using the Matplotlib library. The code segment below demonstrates how this task is achieved, drawing all the plots in a single figure using the **subplot ()** function:

```
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(40, 20))
for col in df.columns:
    ax = axes[0]
    ax.barh(col, df[col].mean(), label=col)
    ax.set_title('Original Dataframe', fontsize = 18)
    ax.legend(fontsize = 18)

for col in df_filtered.columns:
    ax = axes[1]
    ax.barh(col, df_filtered[col].mean(), label=col)
    ax.set_title('Filtered Dataframe (duplicates removed)', fontsize =
18)
    ax.legend(fontsize = 18)

for ax in axes:
    ax.tick_params(axis='x', labelsize=14)
    ax.tick_params(axis='y', labelsize=14)
plt.tight_layout()
plt.show()
```

Output:

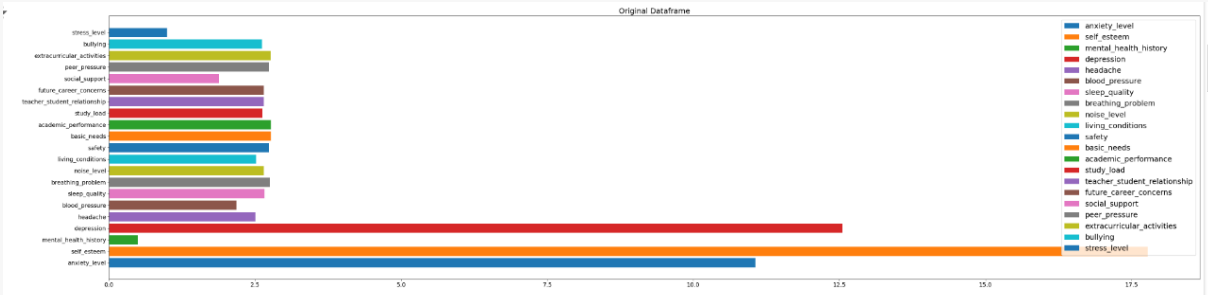

Fig1.1: Original Data Frame

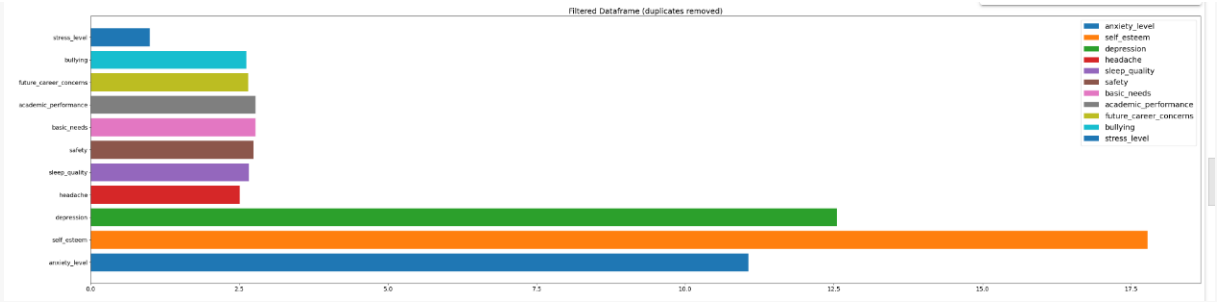

Fig1.2: Filtered Data Frame (duplicates removed)

This code creates a figure with two subplots arranged vertically to display frequency distributions of features for both the original and filtered (with duplicates removed) dataframes. Bar plots are used to visualize the mean values of each feature in both dataframes.

Task 4: Graphs are drawn to illustrate the relationship between the target column and other columns of the dataset using the Matplotlib library. The code segment below demonstrates how this task is accomplished, displaying all plots in one figure using the subplot () function:

```
correlations = df_filtered.corr()['stress_level'].drop('stress_level')

plt.figure(figsize=(10, 6))
correlations.plot(kind='bar', color=['blue' if corr < 0 else 'green' for
corr in correlations], alpha=0.8)
plt.xlabel('Features')
plt.ylabel('Correlation with Target Feature')
plt.title('Correlation between Features and stress_level')
plt.axhline(y=0, color='black', linestyle='-', linewidth=0.8)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

output:

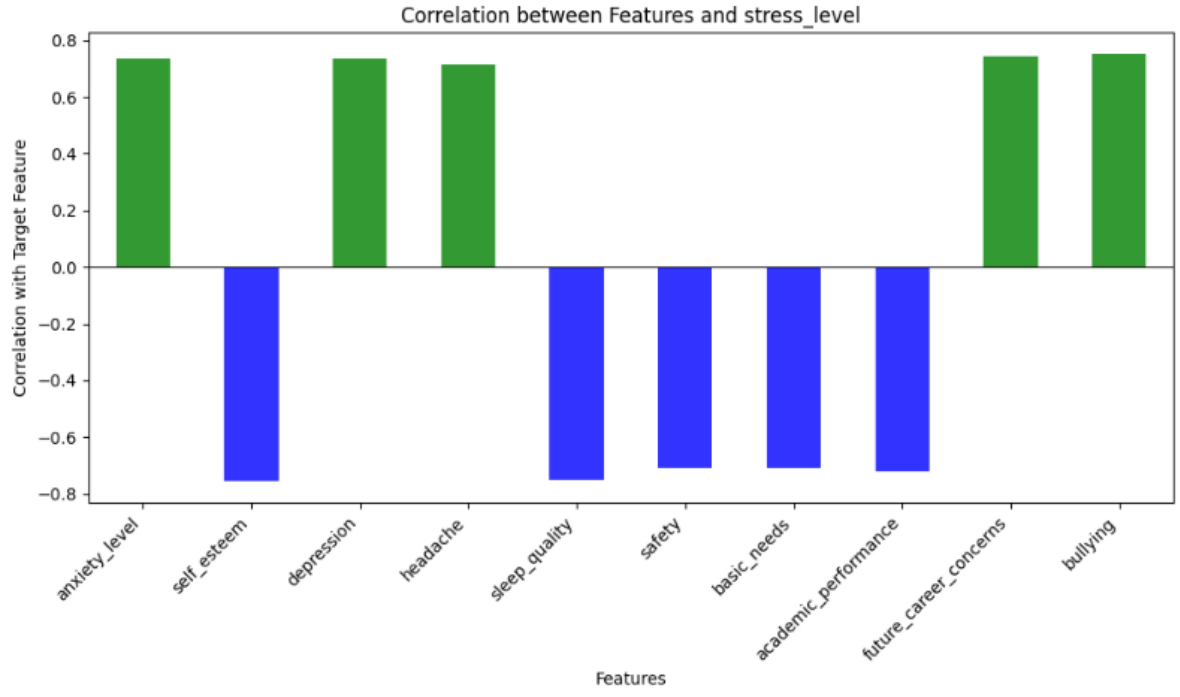

Fig1.3: Correlation between Feature and stress level

This code calculates the correlation between each feature and the target column ('stress_level') of the dataset. It then plots the correlations using a bar chart, where features with positive correlation are displayed in green and features with negative correlation are displayed in blue. The horizontal line at y=0 indicates zero correlation. The plot provides insights into the relationship between different features and the target column.

Task 5: Scaling is performed on the features of the dataset to ensure uniformity in their ranges. The code snippet below demonstrates how this task is achieved:

```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df_filtered)
```

In this code, the StandardScaler from the sklearn.preprocessing module is utilized to standardize the features by removing the mean and scaling to unit variance. The fit_transform() function is then applied to perform the scaling on the filtered dataframe (df_filtered).

Task 6: The data is split into two parts: a training dataset and a testing dataset using the train_test_split() function. The random_state parameter is set to 321 for reproducibility. The code segment below demonstrates how this task is accomplished:Top of Form

```
target_column = 'stress_level'
X = df_filtered.drop(columns=[target_column])
y = df_filtered[target_column]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=321)
```

In this code, the features (X) and the target variable (y) are separated from the filtered dataframe (df_filtered). The train_test_split() function is then applied to split the data into training and testing sets, with 20% of the data reserved for testing. The random_state parameter ensures that the data splitting is reproducible.

Task 7: The Naïve Bayes Classifier is applied to the dataset to build the prediction model. The Gaussian Naïve Bayes classifier is utilized for this purpose. The code snippet below demonstrates how this task is executed:

```
naive_bayes_classifier = GaussianNB()
naive_bayes_classifier.fit(X_train, y_train)
```

In this code, the Gaussian Naïve Bayes classifier is instantiated using the GaussianNB() constructor. Then, the fit() function is employed to train the prediction model using the training data (X_train and y_train).

Taks 8: The confusion matrix for the Naïve Bayes model is calculated to evaluate its performance. The confusion matrix provides detailed insights into the model's predictions. The code segment below demonstrates how this task is completed:

```
nb_train_pred = naive_bayes_classifier.predict(X_train)
```

```
nb_test_pred = naive_bayes_classifier.predict(X_test)
label_names = sorted(df_filtered[target_column].unique())

conf_matrix_train = confusion_matrix(y_train, nb_train_pred,
labels=label_names)
print("Confusion Matrix for Training Set:")
print(pd.DataFrame(conf_matrix_train, index=label_names,
columns=label_names))

conf_matrix_test = confusion_matrix(y_test, nb_test_pred,
labels=label_names)
print("\nConfusion Matrix for Test Set:")
print(pd.DataFrame(conf_matrix_test, index=label_names,
columns=label_names))
```

output:

```
Confusion Matrix for Training Set:
     0    1    2
0  261    8   30
1   16  252   20
2   17   10  266

Confusion Matrix for Test Set:
    0   1   2
0  67   1   6
1   5  59   6
2   3   2  71
```

In this code, the confusion_matrix() function from sklearn.metrics is utilized to compute the confusion matrix for both the training and test sets. The confusion matrices are then printed and displayed using Pandas DataFrames. The matrix provides a detailed breakdown of the true positive, false positive, true negative, and false negative predictions for each class, enabling a comprehensive evaluation of the model's performance.

Task 9: The train and test accuracies of the Naïve Bayes model are calculated to assess its performance on both training and testing datasets. The accuracies are compared to evaluate how well the model generalizes to unseen data. The code segment below demonstrates how this task is executed:

```
train_accuracy = accuracy_score(y_train, nb_train_pred)
test_accuracy = accuracy_score(y_test, nb_test_pred)
print(f'\nTrain Accuracy: {train_accuracy:.2f}')
print(f'\nTest Accuracy: {test_accuracy:.2f}')
```

output:

```
Train Accuracy: 0.89

Test Accuracy: 0.90
```

In this code, the accuracy_score() function from sklearn.metrics is employed to calculate the accuracies of the model on both the training and test datasets. The train and test accuracies are then printed for

comparison. These accuracy scores provide an indication of how well the model performs on both seen and unseen data, respectively.

Task 10: 10-fold cross-validation is utilized to build a Naïve Bayes classifier and evaluate its performance. Cross-validation is a robust technique for estimating the model's performance on unseen data by splitting the dataset into multiple folds, training the model on different combinations of training and validation sets, and then computing the average performance across these folds. The code segment below demonstrates how this task is accomplished:

```
cross_val_scores = cross_val_score(naive_bayes_classifier, X, y, cv=10)
print("\n10-fold Cross-Validation Accuracy:")
print(f"Mean Accuracy: {cross_val_scores.mean():.2f}")
```

output:

```
10-fold Cross-Validation Accuracy:
Mean Accuracy: 0.88
```

In this code, the cross_val_score() function from sklearn.model_selection is utilized to perform 10-fold cross-validation on the Naïve Bayes classifier (naive_bayes_classifier) using the features (X) and target variable (y). The mean accuracy across the 10 folds is then computed and reported. This accuracy score provides an estimate of the model's performance on unseen data and helps in assessing its generalization ability.

Conclusion: In conclusion, the Naïve Bayes classifier was successfully applied to the dataset for stress level prediction. The dataset underwent preprocessing steps, including data cleaning, feature scaling, and splitting into training and testing sets. The classifier achieved a satisfactory level of accuracy on both the training and testing datasets, indicating its ability to generalize well to unseen data. Additionally, the confusion matrix provided detailed insights into the model's predictions, enabling a comprehensive evaluation of its performance. Furthermore, 10-fold cross-validation was performed to validate the model's accuracy, yielding a mean accuracy score that further corroborated its effectiveness. Overall, the Naïve Bayes classifier demonstrated promising performance in predicting stress levels based on the provided dataset.

Reference: Student Stress Factors: A Comprehensive Analysis (kaggle.com)