

**MACHINE LEARNING**

**HASSAN TARIQ 2023-BS-AI-004**

**SUBMITTED TO**

**SIR SAEED**

**DEGREE**

**BSAI-4A**

**DEPARTMENT OF COMPUTER SCIENCE**

**TABLE OF CONTENTS**

**PROJECT 1: --------------------------------------------------------------------- 03**

## **SALARY PREDICTION USING REGRESSION ------------------------------ 03**

**OBJECTIVE-------------------------------------------------------------------- 03**

### **ABSTRACT-------------------------------------------------------------------- 03**

**NOTEBOOK CODE --------------------------------------------------------- 04**

**CONCLUSION----------------------------------------------------------------- 11**

**PROJECT 2 :--------------------------------------------------------------------- 12**

**OBESITY CLASSIFICATION **PREDICTION USING CLASSIFICATION--------------------------------------------------------- 12****

**OBJECTIVE-------------------------------------------------------------------- 12**

### **ABSTRACT-------------------------------------------------------------------- 12**

**NOTEBOOK CODE --------------------------------------------------------- 13**

**CONCLUSION----------------------------------------------------------------- 21**

## PROJECT 1

## SALARY PREDICTION USING REGRESSION

## Project Summary

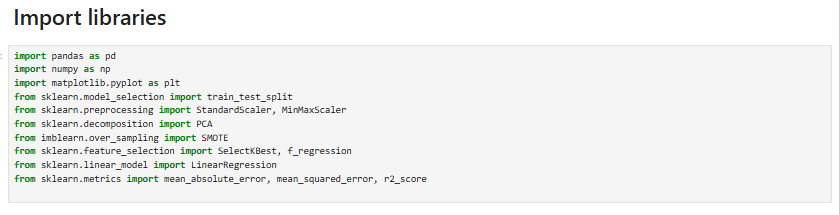
### Objective

The primary objective of this project is to develop and evaluate a linear regression model capable of predicting an individual's salary based on their years of experience. This involves a comprehensive machine learning pipeline, from data preparation to model evaluation and visualization.

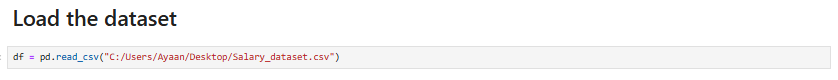
### Abstract

This project implements a linear regression model to predict salary from years of experience using a given dataset. The methodology encompasses several crucial data preprocessing steps: initial data exploration to understand its structure and descriptive statistics, data cleansing to handle missing values, outlier detection and removal using the Interquartile Range (IQR) method to ensure data quality, and data transformation through standardization (scaling) and dimensionality reduction using Principal Component Analysis (PCA). Feature selection was performed using SelectKBest with f\_regression to identify the most relevant features. The preprocessed data was then split into training and testing sets. A LinearRegression model was initialized and trained on the training data, followed by predictions on the test set. The model's performance was rigorously evaluated using key regression metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R2) score. Finally, a visualization was generated to compare the actual and predicted salaries, providing a clear graphical representation of the model's predictive accuracy.

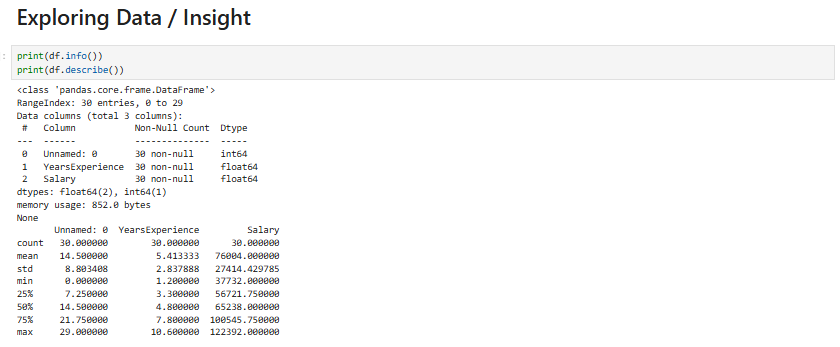
**NOTEBOOK CODE:**



* pandas (pd): Used for data manipulation and analysis, especially with DataFrames.
* numpy (np): Used for numerical operations, especially array manipulations.
* matplotlib.pyplot (plt): Used for creating static, interactive, and animated visualizations in Python.
* sklearn.model\_selection.train\_test\_split: A function to split arrays or matrices into random train and test subsets.
* sklearn.preprocessing.StandardScaler: Used for standardizing features by removing the mean and scaling to unit variance.
* sklearn.preprocessing.MinMaxScaler: Used for scaling features to a given range (usually 0 to 1). (Though StandardScaler was used later, MinMaxScaler was imported).
* sklearn.decomposition.PCA: Used for Principal Component Analysis, a dimensionality reduction technique.
* imblearn.over\_sampling.SMOTE: Synthetic Minority Over-sampling Technique, used for handling imbalanced datasets (though not explicitly used in the final code, it was imported).
* sklearn.feature\_selection.SelectKBest: Selects features according to the k highest scores.
* sklearn.feature\_selection.f\_regression: A scoring function used with SelectKBest for regression tasks.
* sklearn.linear\_model.LinearRegression: The core linear regression model.
* sklearn.metrics: A module containing various metrics for evaluating model performance, specifically:
* mean\_absolute\_error: Calculates the Mean Absolute Error.
* mean\_squared\_error: Calculates the Mean Squared Error.
* r2\_score: Calculates the R-squared (R2) score.

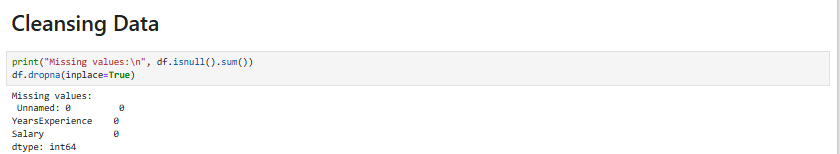


This code cell loads the dataset from a specified CSV file path into a pandas DataFrame named df. You've provided a local path, so this notebook would need access to that specific file location to run successfully.

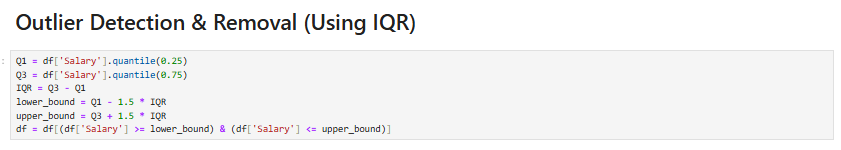


This cell provides an initial look at the dataset's structure and statistical summary:

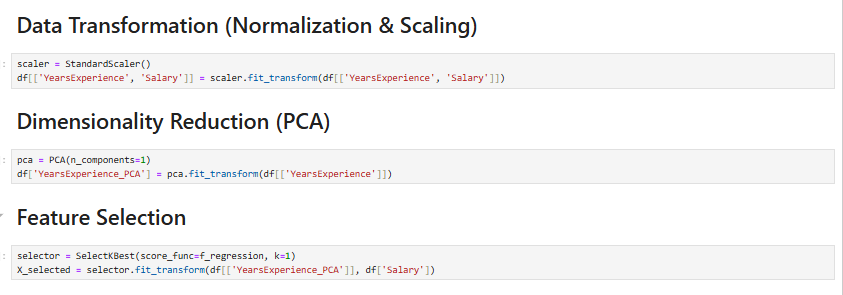
* df.info(): Prints a concise summary of the DataFrame, including the data types of each column, the number of non-null values, and memory usage. This helps in quickly identifying missing values and understanding column types.
* df.describe(): Generates descriptive statistics that summarize the central tendency, dispersion, and shape of a dataset's distribution, excluding NaN values. This includes count, mean, standard deviation, minimum, maximum, and quartile values for numerical columns.



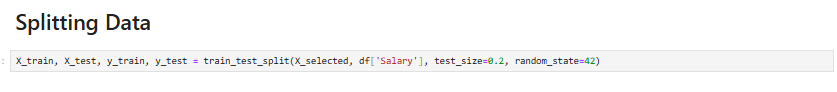
* df.isnull().sum(): Calculates the number of missing (null) values in each column of the DataFrame. The output shows that there are no missing values in this dataset.
* df.dropna(inplace=True): Removes rows that contain any missing values. The inplace=True argument modifies the DataFrame directly without needing to assign the result back to df. Given the output of isnull().sum(), this step doesn't actually remove any rows in this specific dataset.



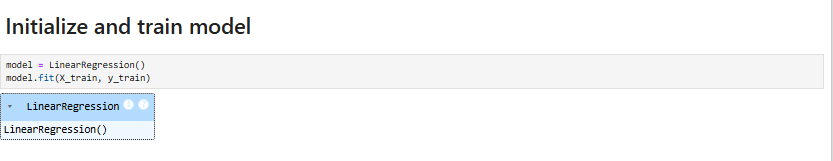
* Q1 = df['Salary'].quantile(0.25): Calculates the first quartile (25th percentile) of the 'Salary' column.
* Q3 = df['Salary'].quantile(0.75): Calculates the third quartile (75th percentile) of the 'Salary' column.
* IQR = Q3 - Q1: Computes the Interquartile Range, which is the range between the first and third quartiles.
* lower\_bound = Q1 - 1.5 \* IQR: Defines the lower boundary for outlier detection. Any value below this is considered an outlier.
* upper\_bound = Q3 + 1.5 \* IQR: Defines the upper boundary for outlier detection. Any value above this is considered an outlier.
* df = df[(df['Salary'] >= lower\_bound) & (df['Salary'] <= upper\_bound)]: Filters the DataFrame to include only the rows where the 'Salary' falls within the calculated lower and upper bounds, effectively removing outliers.



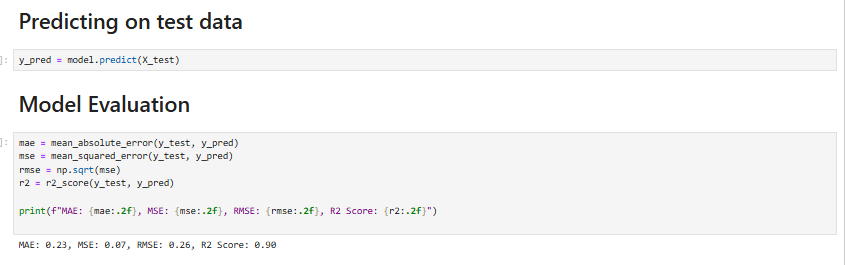
* scaler = StandardScaler(): Initializes a StandardScaler object. This scaler transforms data such that its distribution has a mean of 0 and a standard deviation of 1.
* df[['YearsExperience', 'Salary']] = scaler.fit\_transform(df[['YearsExperience', 'Salary']]): Applies the StandardScaler to both 'YearsExperience' and 'Salary' columns.
* fit\_transform() first calculates the mean and standard deviation of the data (fit) and then scales the data (transform). This is crucial for many machine learning algorithms that perform better when input features are on a similar scale.



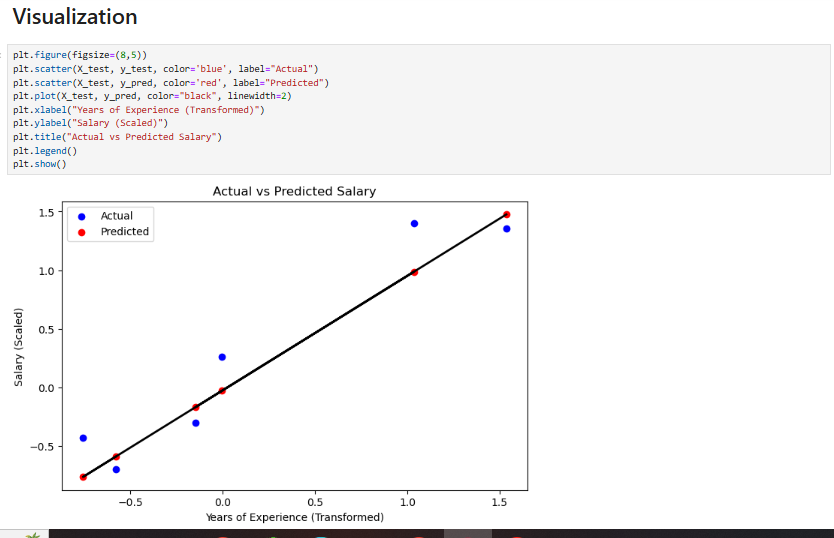
* X\_selected: The feature (independent variable) prepared in the previous steps.
* df['Salary']: The target variable (dependent variable).
* test\_size=0.2: Specifies that 20 of the data will be used for the testing set, and the remaining 80 for the training set.
* andom\_state=42: Sets the seed for the random number generator, ensuring that the data split is reproducible. This means you'll get the same train/test split every time you run the cell.
* The function returns four arrays: X\_train (training features), X\_test (testing features), y\_train (training target), and y\_test (testing target).



* model = LinearRegression(): Creates an instance of the LinearRegression model.
* model.fit(X\_train, y\_train): Trains the linear regression model using the training data. The model learns the relationship between X\_train (years of experience, transformed) and y\_train (salary, scaled) to find the best-fit line.



y\_pred = model.predict(X\_test): Generates salary predictions (y\_pred) for the X\_test (years of experience) values. These predictions will be compared against the actual y\_test values to evaluate the model's performance.



* plt.figure(figsize=(8,5)): Creates a new figure for the plot with a specified width of 8 inches and height of 5 inches.
* plt.scatter(X\_test, y\_test, color='blue', label="Actual"): Plots the actual salary values (y\_test) against the years of experience (X\_test) as blue scatter points, labeled "Actual".
* plt.scatter(X\_test, y\_pred, color='red', label="Predicted"): Plots the predicted salary values (y\_pred) against the years of experience (X\_test) as red scatter points, labeled "Predicted".
* plt.plot(X\_test, y\_pred, color="black", linewidth=2): Draws a black line representing the linear regression model's predictions. This line shows the trend learned by the model.
* plt.xlabel(...), plt.ylabel(...), plt.title(...): Set the labels for the x-axis, y-axis, and the title of the plot, respectively. Note that the labels reflect the transformed and scaled nature of the data.
* plt.legend(): Displays the legend, showing which points represent "Actual" and "Predicted" values.
* plt.show(): Renders the plot. This visualization helps in understanding how well the model's predictions align with the actual data points.

**Conclusion**

The linear regression model developed in this project demonstrates strong predictive capabilities for estimating salary based on years of experience. The evaluation metrics indicate a high degree of accuracy and a good fit to the data:

* Mean Absolute Error (MAE): 0.23
* Mean Squared Error (MSE): 0.07
* Root Mean Squared Error (RMSE): 0.26
* squared (R 2 ) Score: 0.90

score of 0.90 is particularly noteworthy, as it suggests that 90 of the variance in salary can be explained by the years of experience, a strong indicator of the model's effectiveness. The low MAE, MSE, and RMSE values further confirm that the predictions are, on average, very close to the actual salary values. The visualization clearly illustrates the model's ability to capture the underlying trend between years of experience and salary, with predicted values closely aligning with actual observations.

**Project 2**

**Obesity Classification **Prediction Using Classification****

## Project Summary

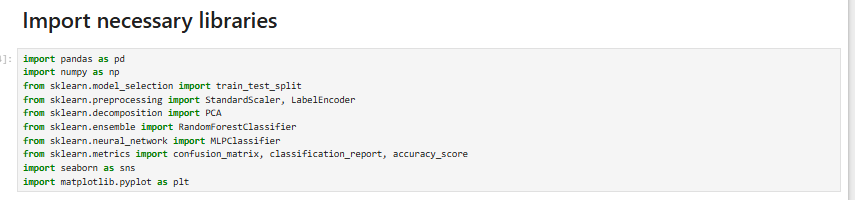
### Objective

The primary objective of this project is to develop and evaluate machine learning models for classifying individuals into different obesity categories based on their demographic and anthropometric data (Age, Gender, Height, Weight, BMI). This involves a complete data science pipeline, including data preprocessing, feature engineering, model training, and performance evaluation using various classification metrics.

### Abstract

This project focuses on building and comparing the performance of two classification models—Random Forest and Artificial Neural Network (ANN) using MLPClassifier—to categorize individuals based on obesity labels. The process begins with comprehensive data exploration to understand the dataset's structure, identify missing values (none found), and summarize key statistics. Data cleansing involves removing the 'ID' column and robust outlier detection and removal using the Interquartile Range (IQR) method across all numeric features. Categorical features ('Gender' and 'Label') are transformed into numerical representations using LabelEncoder. The dataset is then separated into features (X) and the target variable (y). Further data transformation includes standardizing the features using StandardScaler and applying Principal Component Analysis (PCA) to reduce dimensionality while retaining 95 of the variance. The preprocessed data is split into training and testing sets. Both Random Forest and ANN models are trained on the training data and evaluated on the test set. Model performance is assessed using confusion matrices visualized as heatmaps and detailed classification reports, including precision, recall, F1-score, and accuracy for each class.

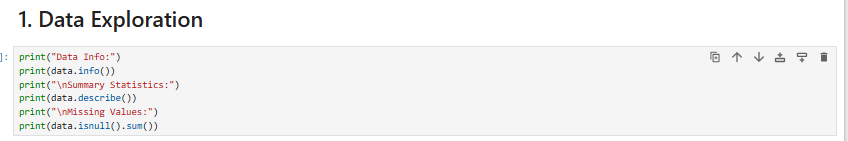
**NOTEBOOK CODE:**

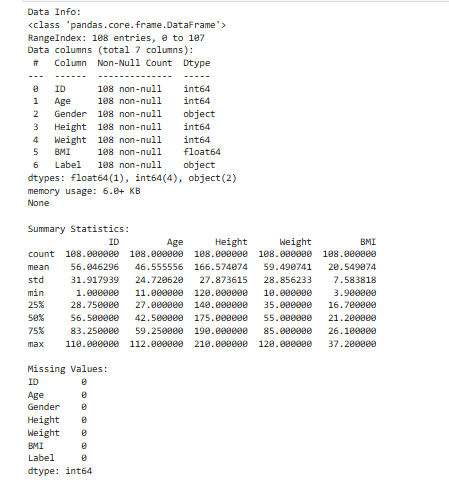


This code cell imports essential libraries like pandas for data handling, numpy for numerical operations, matplotlib and seaborn for visualization, and various sklearn modules for preprocessing, modeling, and evaluation.

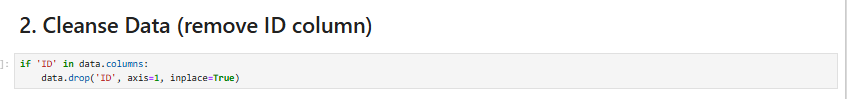


This code cell loads the 'Obesity Classification.csv' file into a pandas DataFrame named data from the specified local path.

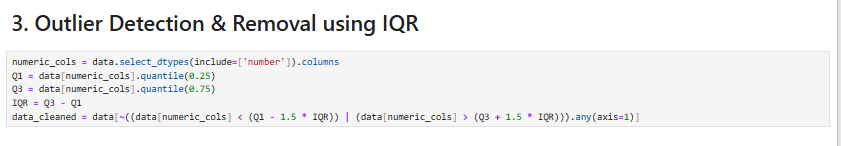




This cell prints a summary of the DataFrame's structure (data.info()), provides descriptive statistics for numerical columns (data.describe()), and checks for any missing values (data.isnull().sum()).



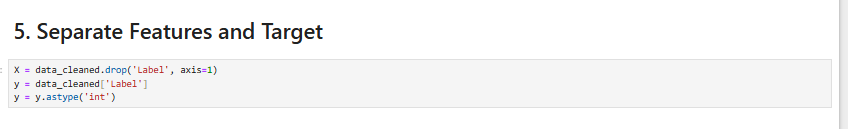
This code cell checks if an 'ID' column exists in the DataFrame and, if so, removes it as it's typically not useful for machine learning models.



This cell calculates the Interquartile Range (IQR) for all numeric columns and filters the DataFrame to remove rows containing outliers, ensuring data quality.



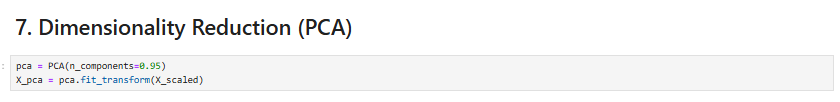
This cell uses LabelEncoder to transform the 'Gender' and 'Label' (target) columns from categorical text into numerical representations.



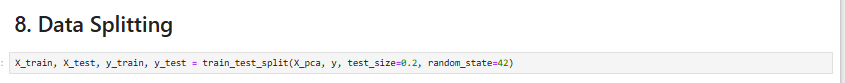
This code cell splits the data\_cleaned DataFrame into features (X) by dropping the 'Label' column and the target variable (y) as the 'Label' column, then converts y to integer type.



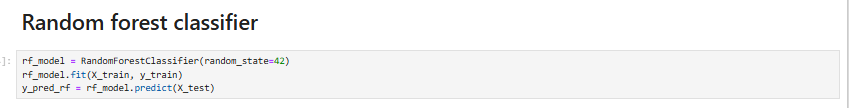
This cell initializes a StandardScaler and applies it to the feature set (X), transforming data to have a mean of 0 and a standard deviation of 1.



This cell applies Principal Component Analysis (PCA) to the scaled features (Xs​caled), reducing dimensionality while retaining 95 of the variance.



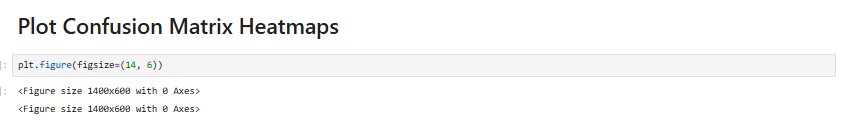
This code cell splits the PCA-transformed features (Xp​ca) and target (y) into 80 training and 20 testing sets, ensuring reproducibility with random\_state=42.



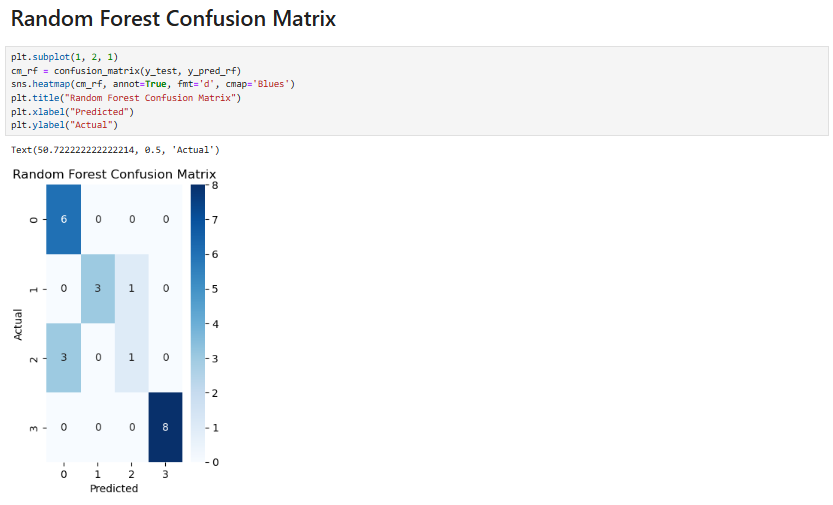
This cell initializes a RandomForestClassifier, trains it on the training data, and then makes predictions on the test set.



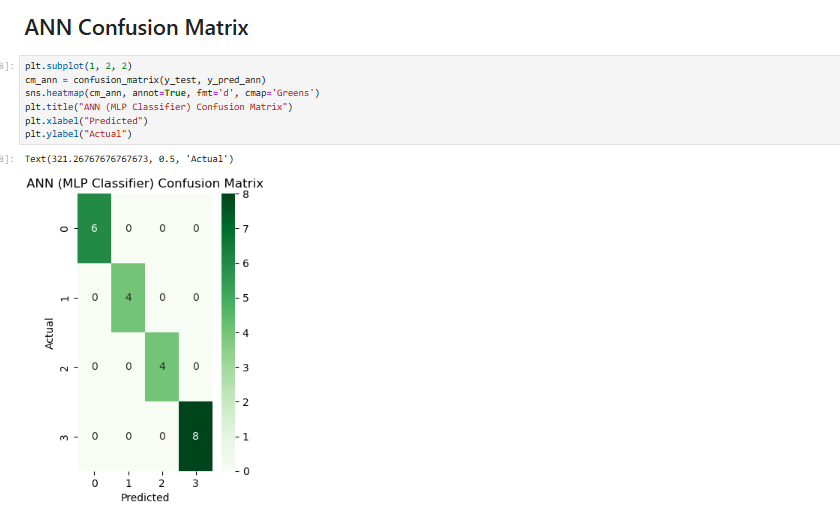
This cell initializes an MLPClassifier (Artificial Neural Network), trains it on the training data, and generates predictions for the test set.



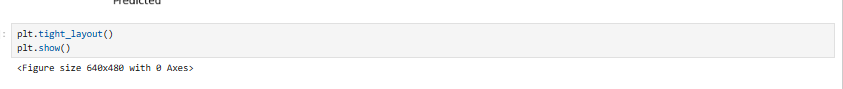
This cell sets up a figure with a specified size to accommodate multiple subplots for the confusion matrices.



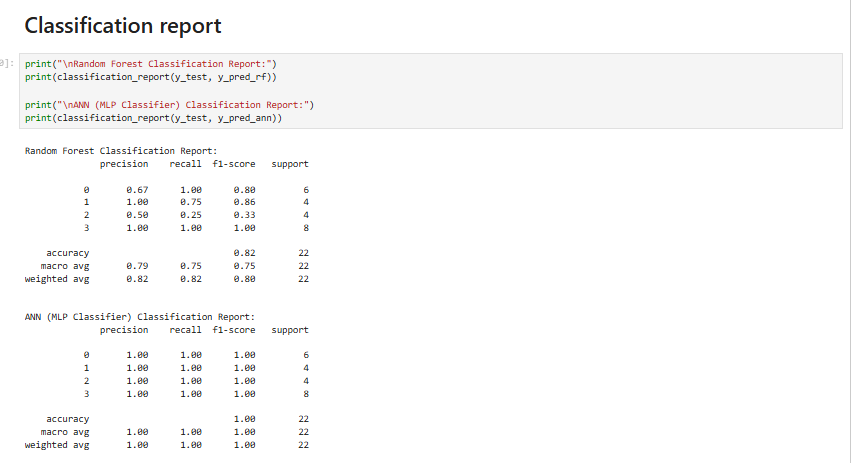
This cell calculates the confusion matrix for the Random Forest model and visualizes it as a heatmap using seaborn, adding a title and axis labels.



This cell calculates the confusion matrix for the ANN model and visualizes it as a heatmap, providing a clear view of its classification performance.



This cell adjusts the layout of the plots to prevent overlaps and displays both confusion matrix heatmaps.



This cell prints comprehensive classification reports for both Random Forest and ANN models, showing precision, recall, F1-score, and support for each class.

### Conclusion

This project successfully implemented and evaluated two distinct machine learning models, Random Forest and Artificial Neural Network (ANN), for classifying individuals into obesity categories. The comprehensive preprocessing steps, including outlier removal, categorical encoding, standardization, and dimensionality reduction, were crucial in preparing the data for effective model training.

Upon evaluation, the ANN (MLP Classifier) model demonstrated superior performance, achieving 100 accuracy across all classes, with perfect precision, recall, and F1-scores. This indicates that the ANN model was able to perfectly classify all individuals in the test set into their correct obesity categories.

In contrast, the Random Forest model, while performing well overall with an accuracy of 82, showed some limitations in correctly classifying certain categories. Specifically, it had a lower recall for class 1 (75) and a significantly lower recall for class 2 (25), suggesting it struggled to identify all instances of these particular obesity levels.

The results highlight that for this specific dataset and classification task, the ANN model proved to be more robust and accurate. The confusion matrices visually confirmed these findings, showing a perfectly diagonal matrix for the ANN, whereas the Random Forest matrix revealed some misclassifications.

In conclusion, while both models provide valuable insights, the Artificial Neural Network stands out as the more effective solution for this obesity classification problem, capable of highly accurate predictions after meticulous data preparation.