**Support Vector Machine (SVM) classifier**

1. Importing necessary libraries:
   * pandas: Used for data manipulation and analysis.
   * train\_test\_split: A function from sklearn.model\_selection module used to split the data into training and testing sets.
   * SVC: A class from sklearn.svm module used to create an SVM classifier.
   * accuracy\_score: A function from sklearn.metrics module used to calculate the accuracy of the model's predictions.
   * matplotlib.pyplot: A library used for data visualization.
2. Loading the dataset:
   * The code reads a CSV file named "framingham.csv" into a pandas DataFrame called **df**.
   * **df.head()** is used to display the first few rows of the DataFrame.
3. Data visualization using histograms:
   * The code creates a 3x3 grid of subplots using **plt.subplot()** and sets the figure size.
   * Each subplot displays a histogram of a specific column from the DataFrame using the **plot(kind="hist")** method.
   * The histograms visualize the distribution of different features in the dataset.
   * This process is repeated for a second 3x3 grid of subplots to visualize additional features.
   * **plt.tight\_layout()** is used to improve the spacing between subplots.
4. Data exploration and preprocessing:
   * **df[df['TenYearCHD']==0]['TenYearCHD'].value\_counts()** counts how many times the value 0 is present in the 'TenYearCHD' column.
   * **df["TenYearCHD"].value\_counts()** counts the values in the 'TenYearCHD' column.
   * **df.isnull().sum()** checks for missing values in the DataFrame.
   * **df=df.fillna(0)** fills the null cells in the DataFrame with zero.
5. Splitting the data into training and testing sets:
   * **X = df.iloc[:,:-1]** selects all columns except the last one as the feature variables.
   * **y = df.iloc[:,-1]** selects only the last column as the target variable.
   * **train\_test\_split(X, y, test\_size=20)** splits the data into training and testing sets. The test set size is set to 20% of the entire dataset.
   * The resulting split data is assigned to **x\_train**, **x\_test**, **y\_train**, and **y\_test** variables.
6. Creating and training the SVM model:
   * **svm = SVC()** creates an instance of the SVC class, representing the SVM classifier.
   * **model = svm.fit(x\_train, y\_train)** trains the SVM model using the training data.
7. Making predictions and evaluating the model:
   * **y\_pred = model.predict(x\_test)** uses the trained model to make predictions on the test data.
   * **accuracy = accuracy\_score(y\_pred, y\_test)** calculates the accuracy of the model's predictions by comparing them to the true labels from the test set.
8. Creating a DataFrame to display the actual and predicted values:
   * **score = pd.DataFrame({"Actual": y\_test, "Predicted": y\_pred})** creates a DataFrame called **score** with the "Actual" and "Predicted" values for comparison.
9. Displaying the first four rows of the score DataFrame using **score[:4]**.