



**Step 1: Import necessary libraries import numpy as np import pandas as pd import matplotlib.pyplot as plt import seaborn as sns import warnings warnings.filterwarnings("ignore", category=UserWarning)**



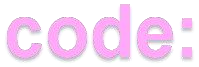
**Step 2: Load the dataset df = pd.read\_csv('/kaggle/input/diabetesdata-set/diabetes.csv')**

**Step 3: Data Cleaning Check for Missing Values**



**missing\_values = df.isnull().sum() print("Missing Values:") print(missing\_values) Handle missing values (if any) mean\_fill = df.mean() df.fillna(mean\_fill, inplace=True) Check for Duplicate Rows duplicate\_rows = df[df.duplicated()] print("\nDuplicate Rows:") print(duplicate\_rows) Handle duplicate rows (if any) df.drop\_duplicates(inplace=True) Step 4: Data Analysis Summary Statistics**

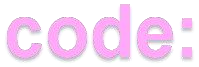




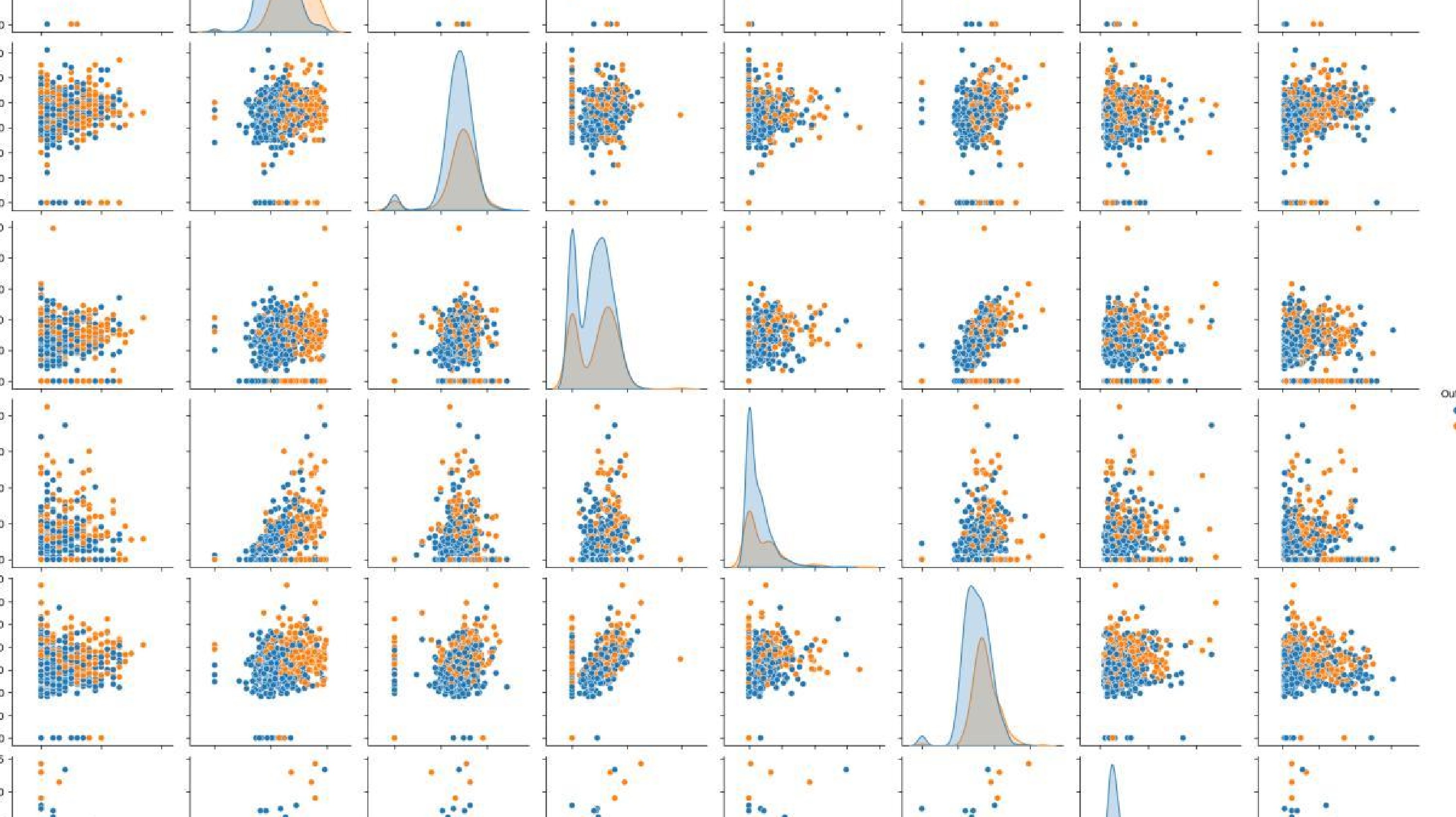
**summary\_stats = df.describe() print("\nSummary Statistics:") print(summary\_stats) Class Distribution**

**class\_distribution = df['Outcome'].value\_counts() print("\nClass Distribution:") print(class\_distribution) Step 5: Data Visualization**





**sns.pairplot(df, hue='Outcome') plt.show()**







|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **sns.pairplot(df, hue='Outcome')** | | | **plt.show()** |  | |  |



**import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns from sklearn.model\_selection import**

**train\_test\_split**

**from sklearn.preprocessing import StandardScaler from sklearn import svm from sklearn.metrics import classification\_report from sklearn.metrics import confusion\_matrix**



**from sklearn.metrics import**

**ConfusionMatrixDisplay**

**RED = "\033[91m"**

**GREEN = "\033[92m"**

**YELLOW = "\033[93m"**

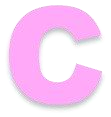
**BLUE = "\033[94m" RESET = "\033[0m"**

**df = pd.read\_csv("/kaggle/input/diabetes-dataset/diabetes.csv"**

**DATA CLEANING**

**print(BLUE + "\nDATA CLEANING" + RESET) Check for missing values missing\_values = df.isnull().sum() print(GREEN + "Missing Values : " + RESET)**

**print(missing\_values) Handle missing values mean\_fill = df.fillna(df.mean()) df.fillna(mean\_fill, inplace=True) Check for duplicate values duplicate\_values = df.duplicated().sum() print(GREEN + "Duplicate Values : " + RESET)**



**print(duplicate\_values) Drop duplicate values df.drop\_duplicates(inplace=True) DATA ANALYSIS**

**print(BLUE + "\nDATA ANALYSIS" + RESET) Summary Statistics summary\_stats = df.describe() print(GREEN + "Summary Statistics : " + RESET)**

**print(summary\_stats) Class Distribution class\_distribution = df["Outcome"].value\_counts() print(GREEN + "Class Distribution : " + RESET) Support Vector Machine Modelling print(BLUE + "\nMODELLING" + RESET) X = df.drop("Outcome", axis=1) y = df["Outcome"]**

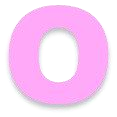
**Splitting the data into training and testing sets**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(**

**X, y, test\_size=0.2, random\_state=42**

**)**

**Standardize Features scaler = StandardScaler() X\_train = scaler.fit\_transform(X\_train) X\_test = scaler.transform(X\_test) init and train SVM model model = svm.SVC(kernel="linear") model.fit(X\_train, y\_train) Predict on test data y\_pred = model.predict(X\_test) Evaluate model performance accuracy = model.score(X\_test, y\_test) print(GREEN + "Model Accuracy : " + RESET)**



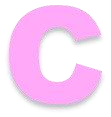
**print(accuracy)**

**Classification Report and Confusion Matrix print(GREEN + "Classification Report : " + RESET) print(classification\_report(y\_test, y\_pred)) print(GREEN + "Confusion Matrix : " + RESET) cm =**

**ConfusionMatrixDisplay.from\_predictions(y\_test, y\_pred)**

**sns.heatmap(cm.confusion\_matrix, annot=True,**

**cmap="Blues") plt.show() print("Displayed") SAVING THE FILE**



**df.to\_csv("/kaggle/working/cleaned\_diabetes.csv", index=False) print(BLUE + "\nDATA SAVING" + RESET) print(GREEN + "Data Cleaned and Saved !" + RESET)**

**print("\n")**



