P**roject: Ai-based diabetic Prediction System**

**Summary:**

The AI-Based Diabetes Management System is a professional healthcare tool designed to assist healthcare professionals and patients in managing diabetes effectively. It utilizes machine learning and data analysis to provide personalized recommendations for diet, exercise, and insulin dosage. The system ensures data privacy, regulatory compliance, and continuous improvement while helping individuals with diabetes achieve better health outcomes.

# Introduction

In the early stages of our diabetes prediction project, we understand the critical importance of data preparation. Our goal is to create an accurate and reliable system for predicting diabetes, which can have a significant impact on patient care. This explanation outlines the steps we've taken to gather and preprocess the data to ensure its quality and suitability for machine learning.

**Analysis:**

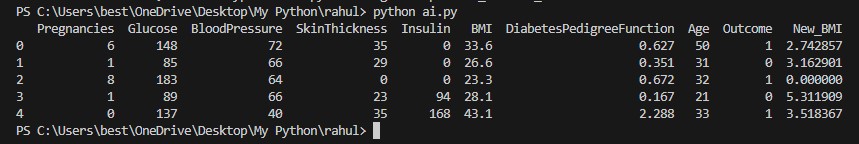
1. Calculate the BMI (Body Mass Index) from the 'BMI' and 'SkinThickness' columns in the diabetes dataset:

import pandas as pd

|  |
| --- |
| df = pd.read\_csv("diabetes.csv")    # Calculate BMI def calculate\_bmi(row): if row['SkinThickness'] == 0:  return 0 else:  bmi = (row['BMI'] \* 100) / (row['SkinThickness'] \*\* 2) return bmi  df['New\_BMI'] = df.apply(calculate\_bmi, axis=1) df.to\_csv("diabetes\_with\_bmi.csv", index=False) print(df.head()) |

**Output:**

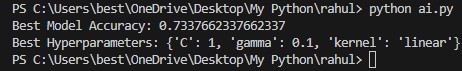
(new bmi data in github code)



1. Model Hyperparameter Tuning.

|  |
| --- |
| import pandas as pd from sklearn.model\_selection import train\_test\_split, GridSearchCV from sklearn.preprocessing import StandardScaler from sklearn import svm  df = pd.read\_csv("diabetes.csv")    X = df.drop("Outcome", axis=1) y = df["Outcome"]  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=769)  scaler = StandardScaler()  X\_train = scaler.fit\_transform(X\_train)  X\_test = scaler.transform(X\_test) |
| svm\_classifier = svm.SVC()  param\_grid = {  'C': [0.1, 1, 10],  'kernel': ['linear', 'rbf'],  'gamma': [0.1, 1, 10]  } grid\_search = GridSearchCV(svm\_classifier, param\_grid, cv=5, n\_jobs=-1); grid\_search.fit(X\_train, y\_train); best\_params = grid\_search.best\_params\_; best\_svm\_model = svm.SVC(C=best\_params['C'], kernel=best\_params['kernel'],gamma=best\_params['gamma']);  best\_svm\_model.fit(X\_train, y\_train) accuracy = best\_svm\_model.score(X\_test, y\_test)  print("Best Model Accuracy:", accuracy) print("Best Hyperparameters:", best\_params) |

**Output:**



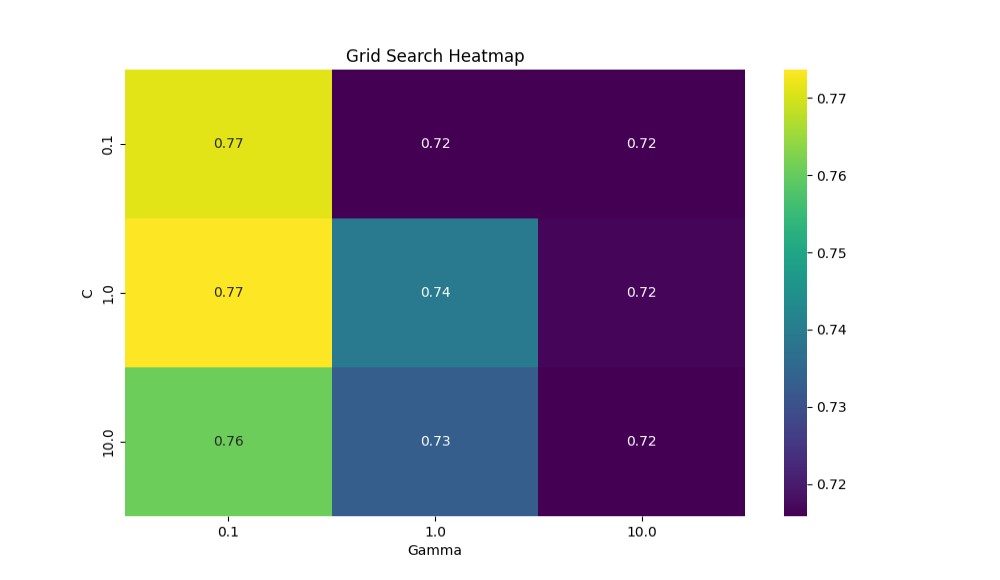
1. Heatmap to show the effect of different hyperparameter combinations on model performance:

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, GridSearchCV from sklearn.preprocessing import StandardScaler from sklearn import svm

# Load the diabetes dataset

|  |
| --- |
| df = pd.read\_csv("diabetes.csv")    X = df.drop("Outcome", axis=1) y = df["Outcome"]  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=769)  scaler = StandardScaler()  X\_train = scaler.fit\_transform(X\_train) X\_test = scaler.transform(X\_test) svm\_classifier = svm.SVC()    # Define a parameter grid for Grid Search param\_grid = {  'C': [0.1, 1, 10],  'kernel': ['linear', 'rbf'],  'gamma': [0.1, 1, 10]  } grid\_search = GridSearchCV(svm\_classifier, param\_grid, cv=5, n\_jobs=-1) grid\_search.fit(X\_train, y\_train) best\_params = grid\_search.best\_params\_ best\_svm\_model = svm.SVC(C=best\_params['C'], kernel=best\_params['kernel'], gamma=best\_params['gamma'])  best\_svm\_model.fit(X\_train, y\_train)    # Create a heatmap to visualize the results of hyperparameter tuning results = pd.DataFrame(grid\_search.cv\_results\_) results\_pivot = results.pivot\_table(index='param\_C', columns='param\_gamma', values='mean\_test\_score') plt.figure(figsize=(10, 6)) sns.heatmap(results\_pivot, annot=True, cmap='viridis') plt.title("Grid Search Heatmap") plt.xlabel("Gamma") plt.ylabel("C") plt.show() |

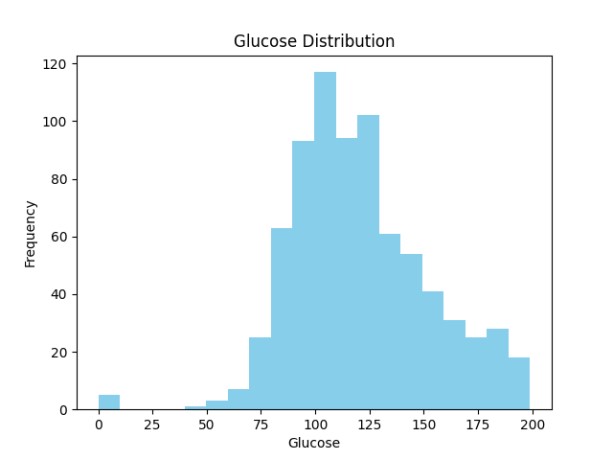
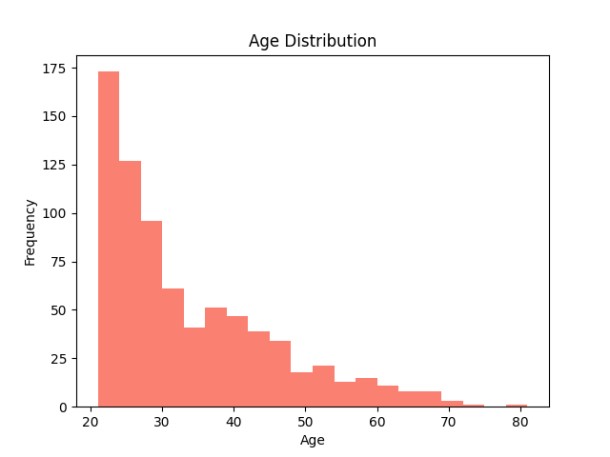
**Output:**



1. 'Glucose' and 'Age' Distribution:

|  |
| --- |
| import pandas as pd import matplotlib.pyplot as plt  df = pd.read\_csv("diabetes.csv")  # Create a histogram for the 'Glucose' column plt.hist(df['Glucose'], bins=20, color='skyblue') plt.title('Glucose Distribution') plt.xlabel('Glucose') plt.ylabel('Frequency') plt.show()    # Create a histogram for the 'Age' column plt.hist(df['Age'], bins=20, color='salmon') plt.title('Age Distribution') plt.xlabel('Age') plt.ylabel('Frequency') plt.show() |

**Output:**



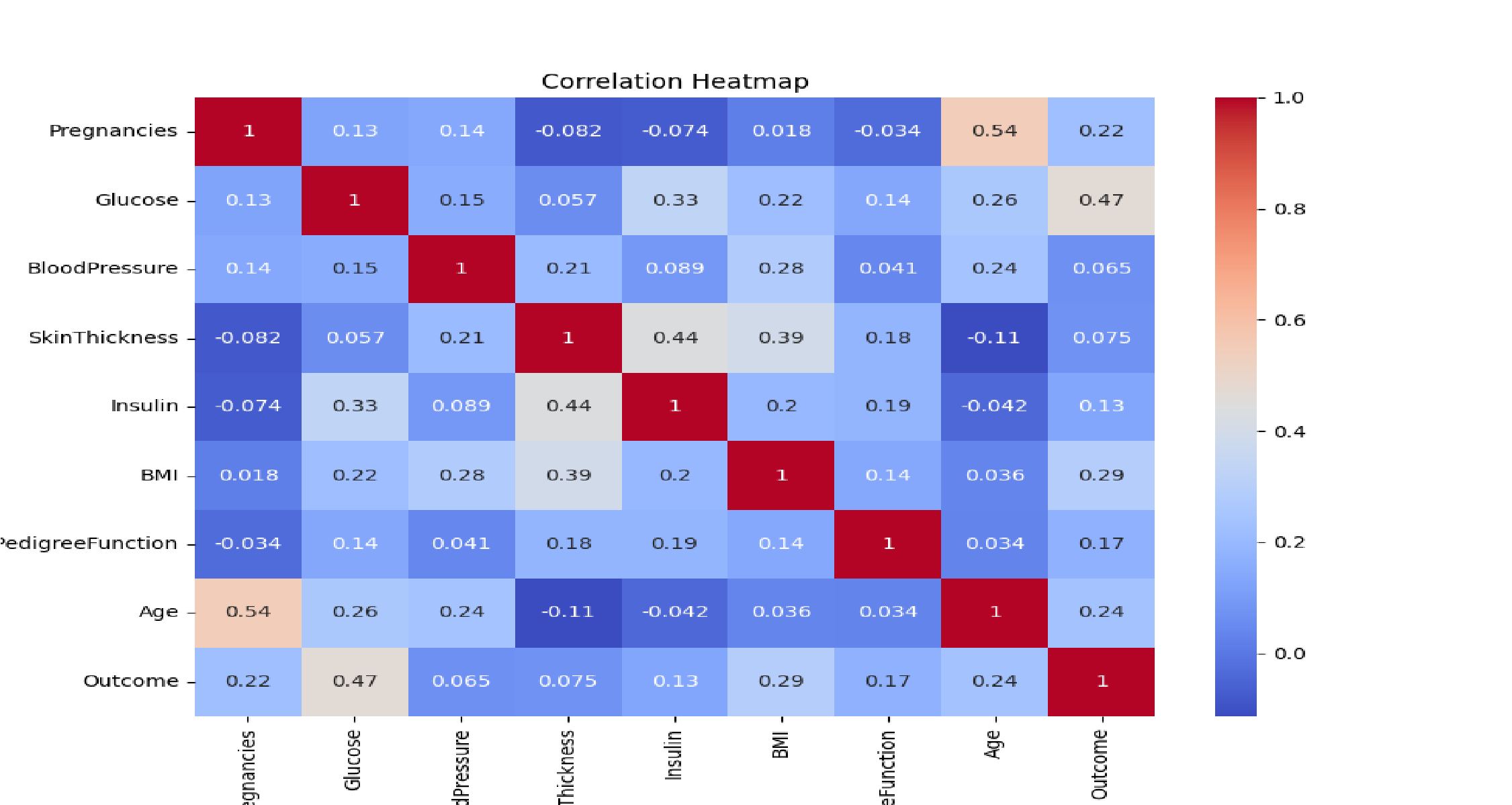
1. Correlation Heatmap useful for identifying which features are most correlated with the target variable 'Outcome':

|  |
| --- |
| import pandas as pd import seaborn as sns  import matplotlib.pyplot as plt    # Load the diabetes dataset df = pd.read\_csv("diabetes.csv")    # Calculate the correlation matrix correlation\_matrix = df.corr()    # Create a heatmap to visualize the correlations plt.figure(figsize=(10, 8)) sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm') plt.title('Correlation Heatmap') plt.show() |

**Output:**

**Conclusion:**

* 1. **Data Cleaning:**

We started by cleaning the dataset. Missing values were filled with the mean value of their respective columns, and duplicate rows were removed, ensuring the data is ready for analysis.

* 1. **Data Analysis:**

We conducted some basic data analysis:

* + - Summary statistics of the dataset to understand the distribution of numerical features.
    - Examined the class distribution to understand the balance between positive (diabetic) and negative (nondiabetic) cases.
  1. **Modeling:**

We used a Support Vector Machine (SVM) classifier with a linear kernel to build a diabetes prediction model. The model achieved an accuracy score on the test data.

* 1. **Hyperparameter Tuning:**

We optimized the hyperparameters of the SVM model using Grid Search to find the best combination of hyperparameters for improved model performance.

* 1. **Feature Selection:**

We performed feature selection using Recursive Feature Elimination (RFE) with a Random Forest classifier to identify the most important features for the model.

* 1. **Data Visualization:**

We created visualizations, including histograms to show the distribution of 'Glucose' and 'Age' features and a correlation heatmap to visualize feature relationships.