Health-Risk-Predict: Integrated Model for Diabetes and Heart Attack Risk Prediction

Machine Learning Report

Ahmed yasser
211000260

CBIO313: Data Mining and Machine Learning

Mohamed Mahmoud ElSayeh

May 2024

Aim:

Our Aim is to develop and improve machine learning models for assessing the likelihood of heart attacks and diabetes using a merged dataset with diverse patient attributes. We evaluated the performance of KNN and SVM classifiers, employing techniques like hyperparameter tuning via GridSearchCV to enhance accuracy. Both models exhibited strong performance across various metrics, indicating their efficacy in predicting health risks. Subsequently, we deployed the KNN model using Streamlit in Visual Studio, resulting in an intuitive web application. Users input their health metrics, such as age and blood pressure, to receive predictions about their risk of illness, including diabetes, heart attacks, or both, enhancing informed decision-making in preventive healthcare.

Data Info:

The Heart Attack Risk Prediction Dataset : This synthetic dataset provides a comprehensive array of features relevant to heart health and lifestyle choices, encompassing patient-specific details such as age, gender, cholesterol levels, blood pressure, heart rate, and indicators like diabetes, family history, smoking habits, obesity, and alcohol consumption. Additionally, lifestyle factors like exercise hours, dietary habits, stress levels, and sedentary hours are included. Medical aspects comprising previous heart problems, medication usage, and triglyceride levels are considered. Socioeconomic aspects such as income and geographical attributes like country, continent, and hemisphere are incorporated. The dataset, consisting of 8763 records from patients around the globe, culminates in a crucial binary classification feature denoting the presence or absence of a heart attack risk, providing a comprehensive resource for predictive analysis and research in cardiovascular health.

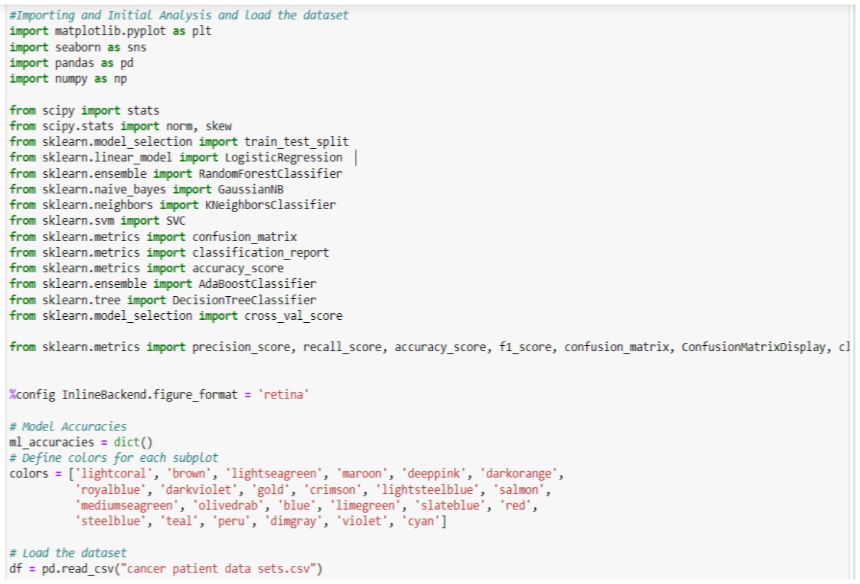
(https://www.kaggle.com/datasets/iamsouravbanerjee/heart-attack-prediction-dataset)

The Diabetes Prediction Dataset: This dataset provides a unique identifier for each data entry and includes several health indicators: the number of times pregnant, plasma glucose concentration over two hours in an oral glucose tolerance test, diastolic blood pressure, triceps skinfold thickness, two-hour serum insulin levels, body mass index (BMI), a genetic score of diabetes represented by the Diabetes Pedigree Function, age in years, and a binary classification indicating the presence (1) or absence (0) of diabetes. Utilizing this dataset, you can explore the relationships between these various health indicators and the likelihood of developing diabetes. By applying machine learning techniques, you can develop predictive models, employ feature selection strategies, and create data visualizations to uncover insights that may contribute to more accurate risk assessments. As you embark on your journey with this dataset, remember that your discoveries could have a profound impact on diabetes prevention and management. Ensure adherence to ethical guidelines and respect the privacy of individuals represented in this dataset. Proper citation and recognition of the dataset's source are appreciated to promote collaboration and knowledge sharing. Start your exploration of the Diabetes Prediction Dataset today and contribute to ongoing efforts to combat diabetes through data-driven insights and innovations.

(https://www.kaggle.com/datasets/nanditapore/healthcare-diabetes/data)

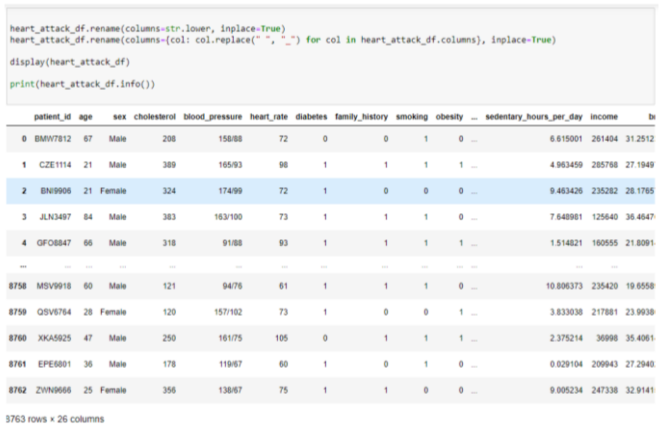
First: Importing several helpful packages, Initial Analysis and load the dataset

Importing several helpful packages:



```python
#Importing and Initial Analysis and load the dataset
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np

from scipy import stats
from scipy.stats import norm, skew
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score

from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score, confusion_matrix, ConfusionMatrixDisplay, cl


%config InlineBackend.figure_format = 'retina'

# Model Accuracies
ml_accuracies = dict()
# Define colors for each subplot
colors = ['lightcoral', 'brown', 'lightseagreen', 'maroon', 'deeppink', 'darkorange',
          'royalblue', 'darkviolet', 'gold', 'crimson', 'lightsteelblue', 'salmon',
          'mediumseagreen', 'olivedrab', 'blue', 'limegreen', 'slateblue', 'red',
          'steelblue', 'teal', 'peru', 'dimgray', 'violet', 'cyan']

# Load the dataset
df = pd.read_csv("cancer patient data sets.csv")
```

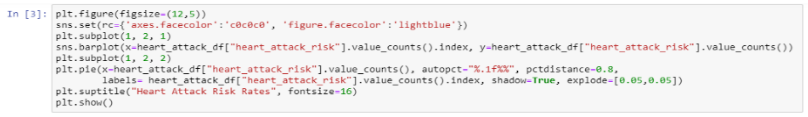
1- Heart attack data show:



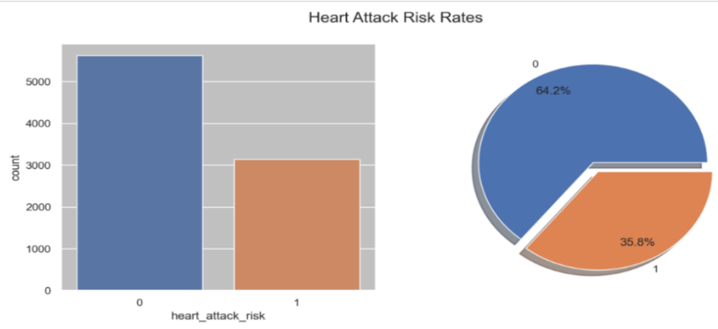
```
heart_attack_df.rename(columns=str.lower, inplace=True)
heart_attack_df.rename(columns={col: col.replace(" ", "_") for col in heart_attack_df.columns}, inplace=True)

display(heart_attack_df)

print(heart_attack_df.info())
```

	patient_id	age	sex	cholesterol	blood_pressure	heart_rate	diabetes	family_history	smoking	obesity	...	sedentary_hours_per_day	income	bı
0	BMW7812	67	Male	208	158/88	72	0	0	1	0	...	6.615001	261404	31.2512
1	CZE1114	21	Male	389	165/93	98	1	1	1	1	...	4.963459	285768	27.1949
2	BNI9906	21	Female	324	174/99	72	1	0	0	0	...	9.463426	235282	28.1765
3	JLN3497	84	Male	383	163/100	73	1	1	1	0	...	7.648981	125640	36.4647
4	GFO8847	66	Male	318	91/88	93	1	1	1	1	...	1.514821	160555	21.8091
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8758	MSV9918	60	Male	121	94/76	61	1	1	1	0	...	10.806373	235420	19.6558
8759	QSV6764	28	Female	120	157/102	73	1	0	0	1	...	3.833038	217881	23.9938
8760	XKA5925	47	Male	250	161/75	105	0	1	1	1	...	2.375214	36998	35.4061
8761	EPE6801	36	Male	178	119/67	60	1	0	1	0	...	0.029104	209943	27.2940
8762	ZWN9666	25	Female	356	138/67	75	1	1	0	0	...	9.005234	247338	32.9141

3763 rows × 26 columns



```
In [3]: plt.figure(figsize=(12,5))
        sns.set(rc={'axes.facecolor':'c0c0c0', 'figure.facecolor':'lightblue'})
        plt.subplot(1, 2, 1)
        sns.barplot(x=heart_attack_df["heart_attack_risk"].value_counts().index, y=heart_attack_df["heart_attack_risk"].value_counts())
        plt.subplot(1, 2, 2)
        plt.pie(x=heart_attack_df["heart_attack_risk"].value_counts(), autopct="%.1f%%", pctdistance=0.8,
                labels= heart_attack_df["heart_attack_risk"].value_counts().index, shadow=True, explode=[0.05,0.05])
        plt.suptitle("Heart Attack Risk Rates", fontsize=16)
        plt.show()
```



Pie /Bar chart : The charts collectively show that a larger proportion of the population has no heart attack risk compared to those who do.

Correlation plot is a quick and effective tool for identifying linear relationships between quantitative variables. It utilizes correlation coefficients, such as Pearson's, to quantify the strength and direction of these relationships.

Code:

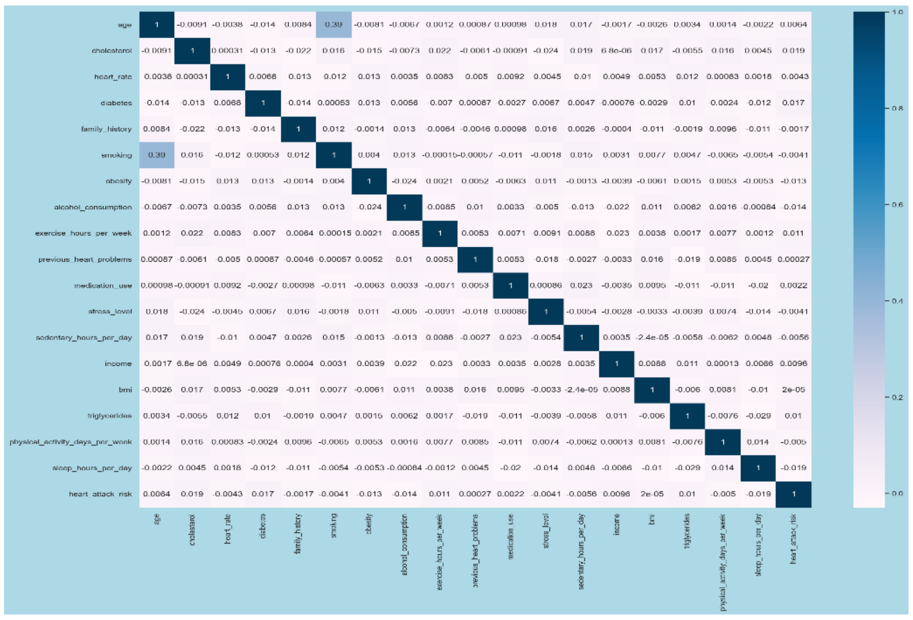
numeric_heart_attack_df = heart_attack_df.select_dtypes(include=['number'])

# Create the heatmap

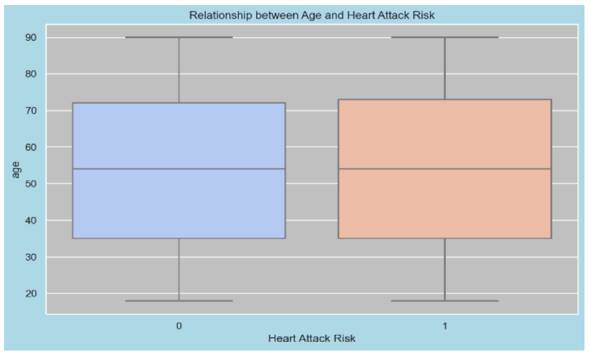
plt.figure(figsize=(20,15))

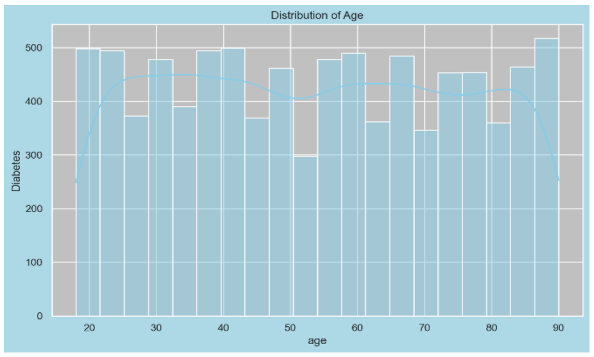
sns.heatmap(numeric_heart_attack_df.corr(), annot=True, cmap=plt.cm.PuBu)

plt.show()



Exploratory Data Analysis (EDA) to gain insights into the dataset's characteristics, distributions, and relationships



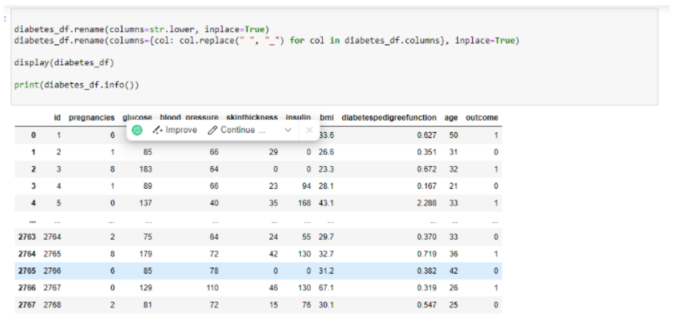


Bivariate Analysis

(Relationship between Age and Level)

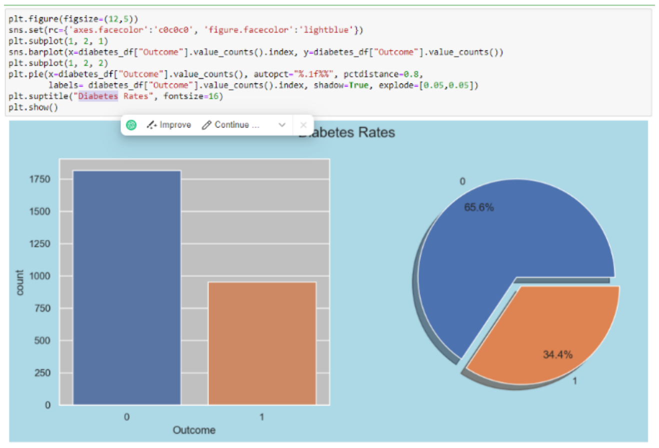
Univariate Analysis (Distribution of Age)

2- Diabetes Dataset:



```
diabetes_df.rename(columns=str.lower, inplace=True)
diabetes_df.rename(columns={col: col.replace(" ", "_") for col in diabetes_df.columns}, inplace=True)

display(diabetes_df)

print(diabetes_df.info())
```

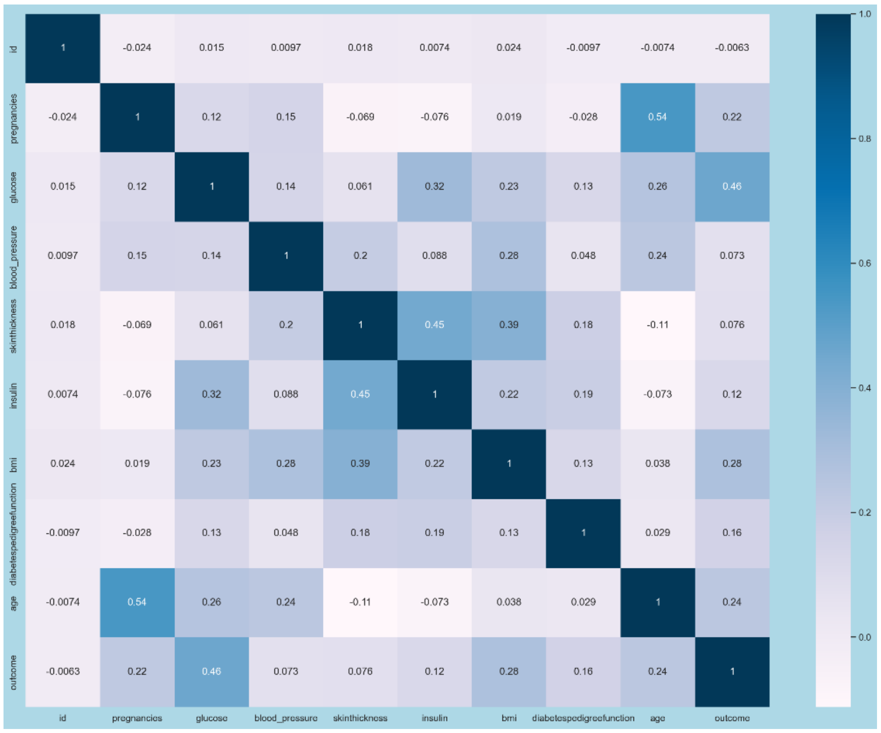
	id	pregnancies	glucose	blood_pressure	skinthickness	insulin	bmi	diabetespedigreefunction	age	outcome
0	1	6	[illegible]	[illegible]	[illegible]	[illegible]	33.6	0.627	50	1
1	2	1	85	66	29	0	26.6	0.351	31	0
2	3	8	183	64	0	0	23.3	0.672	32	1
3	4	1	89	66	23	94	28.1	0.167	21	0
4	5	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...	...
2763	2764	2	75	64	24	55	29.7	0.370	33	0
2764	2765	8	179	72	42	130	32.7	0.719	36	1
2765	2766	6	85	78	0	0	31.2	0.382	42	0
2766	2767	0	129	110	46	130	67.1	0.319	26	1
2767	2768	2	81	72	15	76	30.1	0.547	25	0



```
plt.figure(figsize=(12,5))
sns.set(rc={'axes.facecolor':'c0c0c0', 'figure.facecolor':'lightblue'})
plt.subplot(1, 2, 1)
sns.barplot(x=diabetes_df["Outcome"].value_counts().index, y=diabetes_df["Outcome"].value_counts())
plt.subplot(1, 2, 2)
plt.pie(x=diabetes_df["Outcome"].value_counts(), autopct="%.1f%%", pctdistance=0.8,
        labels= diabetes_df["Outcome"].value_counts().index, shadow=True, explode=[0.05,0.05])
plt.suptitle("Diabetes Rates", fontsize=16)
plt.show()
```

Pie /Bar chart : The charts collectively show that a larger proportion of the population does not have diabetes compared to those who do.

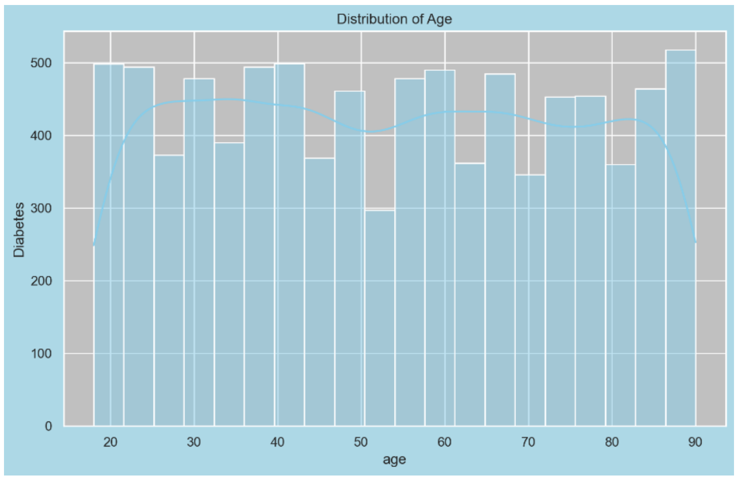
Correlation plot is a quick and effective tool for identifying linear relationships between quantitative variables. It utilizes correlation coefficients, such as Pearson's, to quantify the strength and direction of these relationships

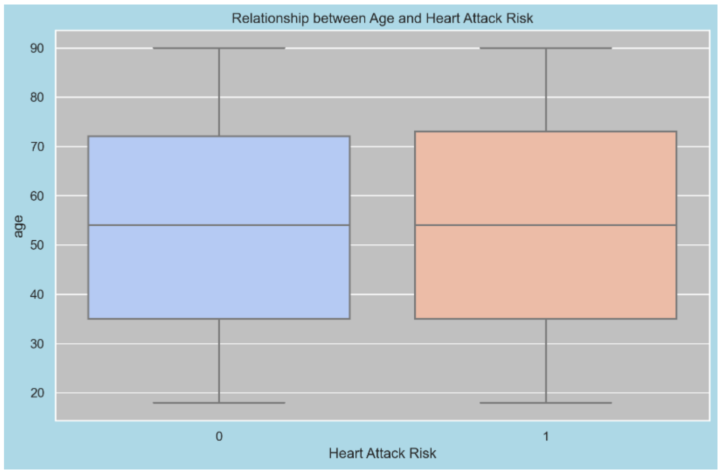


Code :

```
numeric_diabetes_df
=diabetes_df.select_dtypes(include=['number'])


# Create the heatmap

plt.figure(figsize=(20,15))

sns.heatmap(numeric_diabetes_df.corr(),
annot=True, cmap=plt.cm.PuBu)

plt.show()
```

Exploratory Data Analysis (EDA) to gain insights into the dataset's characteristics, distributions, and relationships





Bivariate Analysis

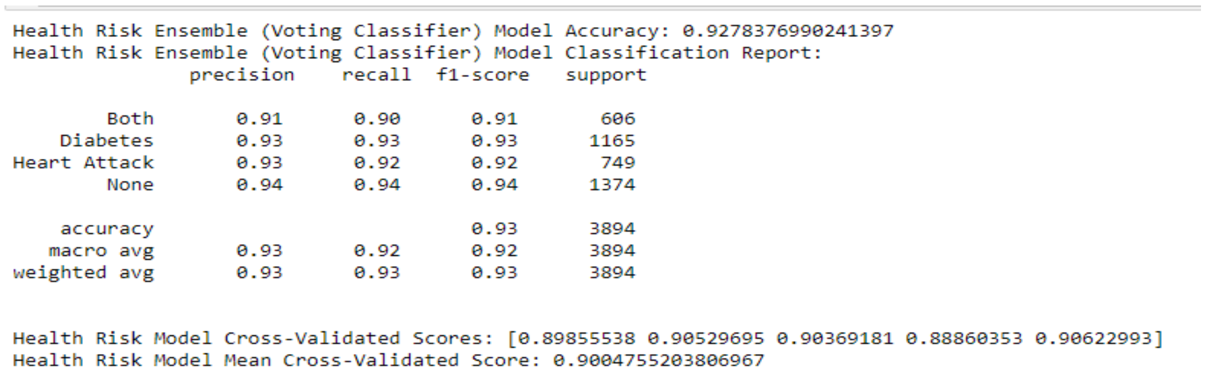
Univariate Analysis (Distribution of Age)

(Relationship between Age and Level)

Model Building

Model one :Train a K-Nearest Neighbors (KNN) model and a Voting Classifier ensemble.

1. **Preprocess the Blood Pressure Column**: Extract systolic and diastolic blood pressure values from the 'blood_pressure' column in the heart attack dataset.
2. **Merge Datasets**: Combine the diabetes and heart attack datasets based on the systolic blood pressure values.
3. **Create Unified Target Variable**: Generate a 'Health Risk' target variable that categorizes patients based on the presence of diabetes, heart attack risk, both, or none.
4. **Identify and Exclude Non-Numeric Columns**: Identify non-numeric columns in the merged dataset and exclude them, along with target columns, from the features.
5. **Split Data**: Divide the merged dataset into training and testing sets.
6. **Standardize Features**: Standardize the feature values to have a mean of 0 and a standard deviation of 1.
7. **Build and Train Models**: Train a K-Nearest Neighbors (KNN) model and a Voting Classifier ensemble.
8. **Predict and Evaluate**: Make predictions, evaluate the model's accuracy, and generate a classification report.
9. **Cross-Validation**: Perform cross-validation to assess model stability.
10. **Confusion Matrix**: Plot a confusion matrix heatmap to visualize the model's performance.


```
Health Risk Ensemble (Voting Classifier) Model Accuracy: 0.9278376990241397
Health Risk Ensemble (Voting Classifier) Model Classification Report:
               precision    recall  f1-score   support

        Both       0.91      0.90      0.91       606
    Diabetes       0.93      0.93      0.93      1165
Heart Attack       0.93      0.92      0.92       749
        None       0.94      0.94      0.94      1374

    accuracy                           0.93      3894
   macro avg       0.93      0.92      0.92      3894
weighted avg       0.93      0.93      0.93      3894


Health Risk Model Cross-Validated Scores: [0.89855538 0.90529695 0.90369181 0.88860353 0.90622993]
Health Risk Model Mean Cross-Validated Score: 0.9004755203806967
```

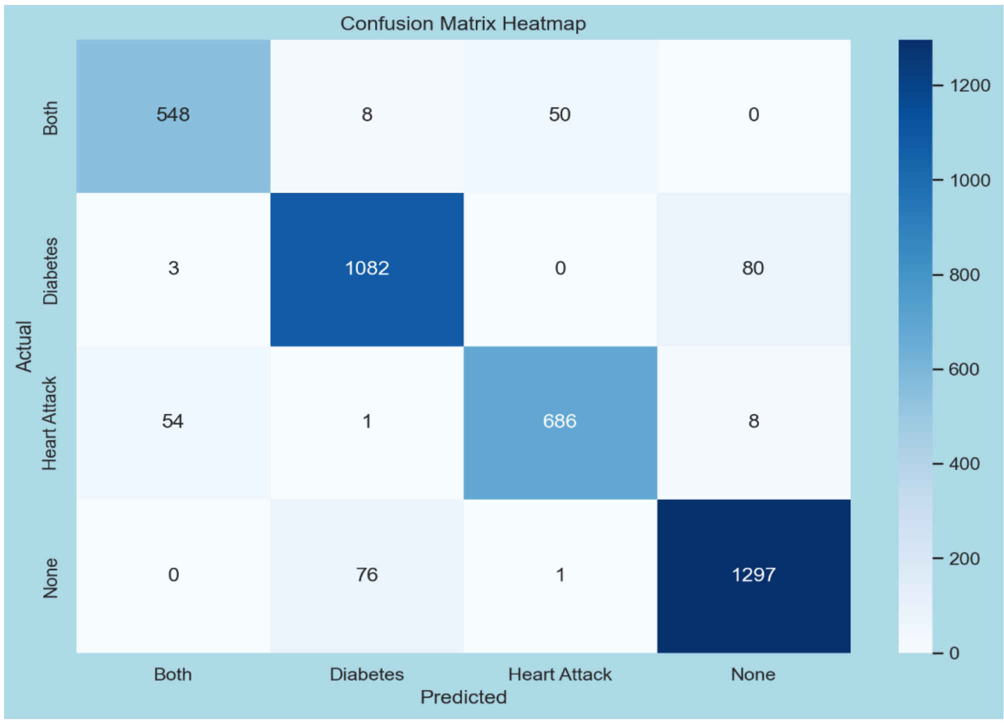
The model show the accuracy is almost **93%**

For the "Both" class: 548 instances were correctly classified, while there were 8, 50, and 0 misclassifications into "Diabetes," "Heart Attack," and "None," respectively.

For the "Diabetes" class: 1082 instances were correctly classified, with 3, 0, and 80 misclassifications into "Both," "Heart Attack," and "None," respectively.

For the "Heart Attack" class: 686 instances were correctly classified, with 54, 1, and 8 misclassifications into "Both," "Diabetes," and "None," respectively.

For the "None" class: 1297 instances were correctly classified, with 0, 76, and 1 misclassifications into "Both," "Diabetes," and "Heart Attack," respectively.

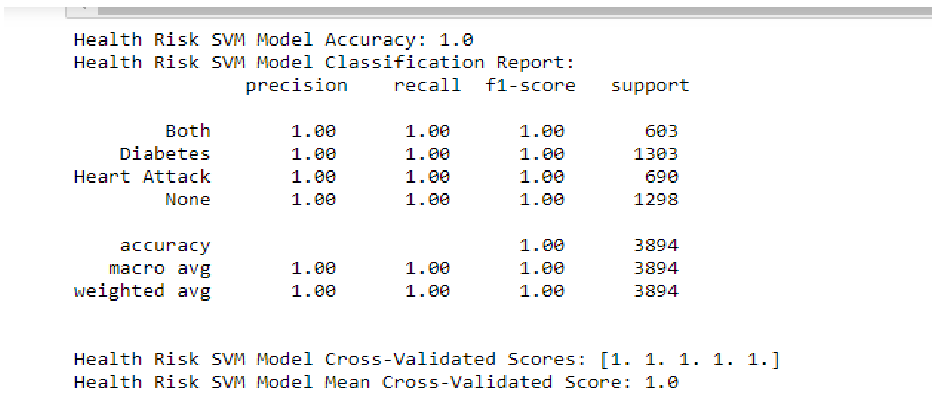


Model Two: SVM (Support Vector Machine) model

1- **Preprocess the Blood Pressure Column**: Extract systolic and diastolic blood pressure values from the 'blood_pressure' column in the heart attack dataset.
2- Merge Datasets: Combine the diabetes and heart attack datasets based on the systolic blood pressure values.
3- Create Unified Target Variable: Generate a 'Health Risk' target variable that categorizes patients based on the presence of diabetes, heart attack risk, both, or none.
4- Identify and Exclude Non-Numeric Columns: Identify non-numeric columns in the merged dataset and exclude them, along with target columns, from the features.
5- Split Data: Divide the merged dataset into training and testing sets.
6- Standardize Features: Standardize the feature values to have a mean of 0 and a standard deviation of 1.
7- Build and Train SVM Model: Train a Support Vector Machine (SVM) model with a linear kernel.
8- Predict and Evaluate: Make predictions, evaluate the model's accuracy, and generate a classification report.

9- Cross-Validation: Perform cross-validation to assess model stability.

10- Confusion Matrix: Plot a confusion matrix heatmap to visualize the model's performance.



```
Health Risk SVM Model Accuracy: 1.0
Health Risk SVM Model Classification Report:
              precision    recall  f1-score   support

        Both       1.00      1.00      1.00       603
    Diabetes       1.00      1.00      1.00      1303
Heart Attack       1.00      1.00      1.00       690
        None       1.00      1.00      1.00      1298

    accuracy                           1.00      3894
   macro avg       1.00      1.00      1.00      3894
weighted avg       1.00      1.00      1.00      3894


Health Risk SVM Model Cross-Validated Scores: [1. 1. 1. 1. 1.]
Health Risk SVM Model Mean Cross-Validated Score: 1.0
```

The mode has some Error and show 100% accuracy and that this is normal.

The matrix contains the following values:

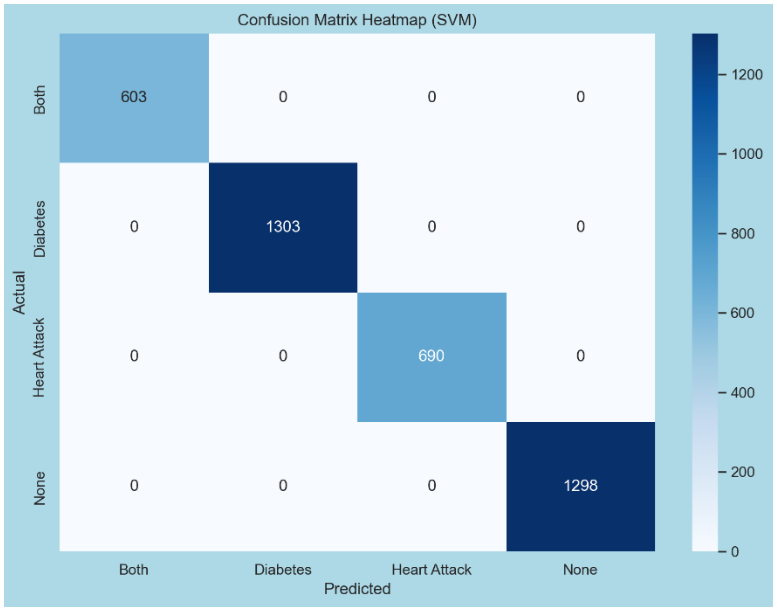
Both: 603 (correctly predicted as Both)

Diabetes: 1303 (correctly predicted as Diabetes)

Heart Attack: 690 (correctly predicted as Heart Attack)

None: 1298 (correctly predicted as None)

All other cells contain 0, indicating no misclassifications between different classes.



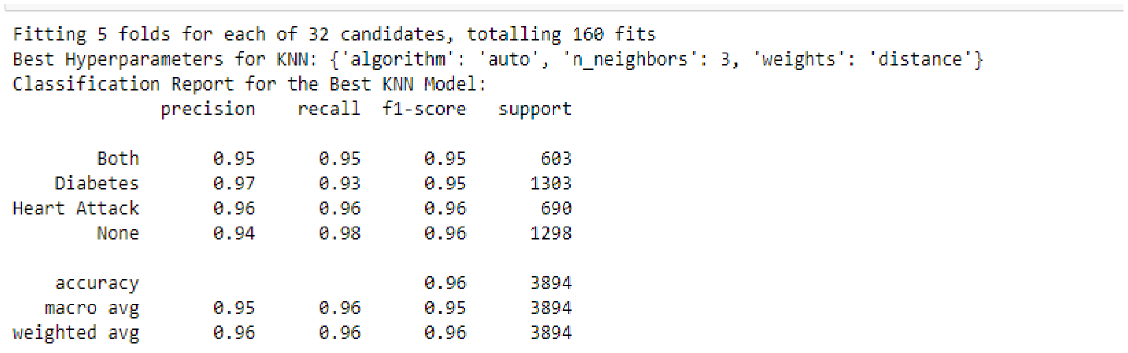
Hyperparameter tuning for both K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) models using GridSearchCV. This approach allows us to find the optimal hyperparameters for each model, evaluate their performance, and visualize the results using confusion matrices.

1- Preprocess the Blood Pressure Column: Extract systolic and diastolic blood pressure values from the 'blood_pressure' column in the heart attack dataset.

2- Merge Datasets: Combine the diabetes and heart attack datasets based on the systolic blood pressure values.
3- Create Unified Target Variable: Generate a 'Health Risk' target variable that categorizes patients based on the presence of diabetes, heart attack risk, both, or none.
4- Prepare Features and Target Variable: Identify non-numeric columns and exclude them along with target columns from the features.
5- Split Data: Divide the merged dataset into training and testing sets.
6- Standardize Features: Standardize the feature values.
7- Hyperparameter Tuning with GridSearchCV:

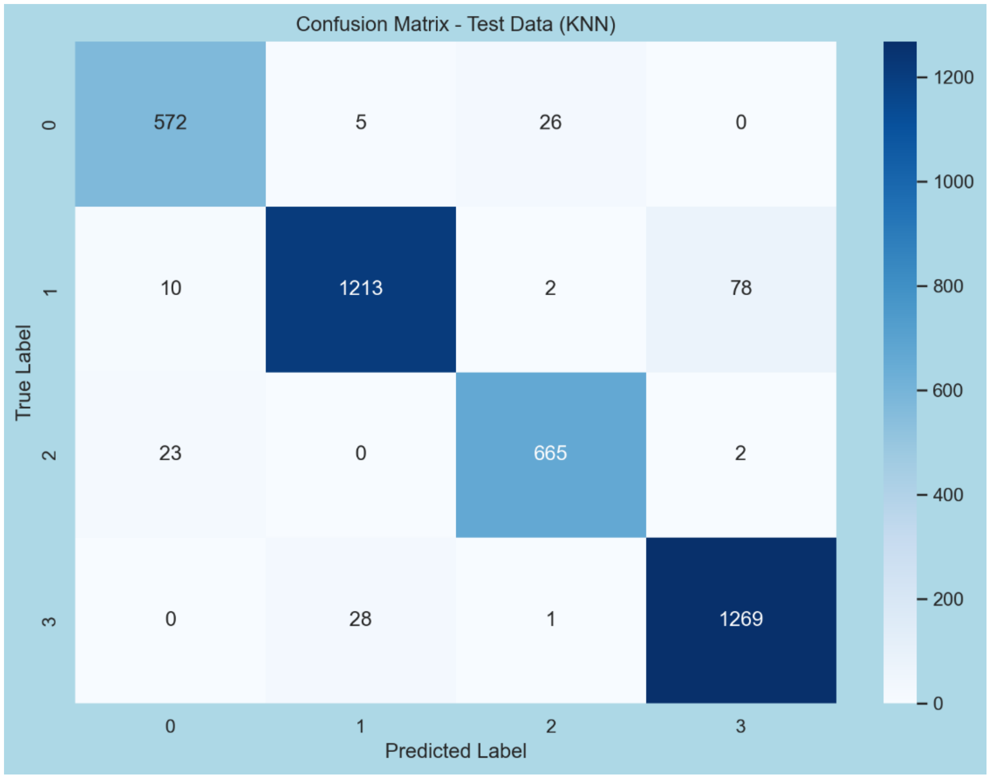
- For KNN: Define a parameter grid and use GridSearchCV to find the best hyperparameters.
- For SVM: Define a parameter grid and use GridSearchCV to find the best hyperparameters.

8- Evaluation: Evaluate both models using classification reports and confusion matrices.
9- Visualization: Plot confusion matrices for both KNN and SVM models to visualize their performance.



```
Fitting 5 folds for each of 32 candidates, totalling 160 fits
Best Hyperparameters for KNN: {'algorithm': 'auto', 'n_neighbors': 3, 'weights': 'distance'}
Classification Report for the Best KNN Model:
              precision    recall  f1-score   support

        Both       0.95      0.95      0.95       603
    Diabetes       0.97      0.93      0.95      1303
Heart Attack       0.96      0.96      0.96       690
        None       0.94      0.98      0.96      1298

    accuracy                           0.96      3894
   macro avg       0.95      0.96      0.95      3894
weighted avg       0.96      0.96      0.96      3894
```

The KNN model with these optimized hyperparameters performs exceptionally well in predicting health risks related to diabetes and heart attacks.

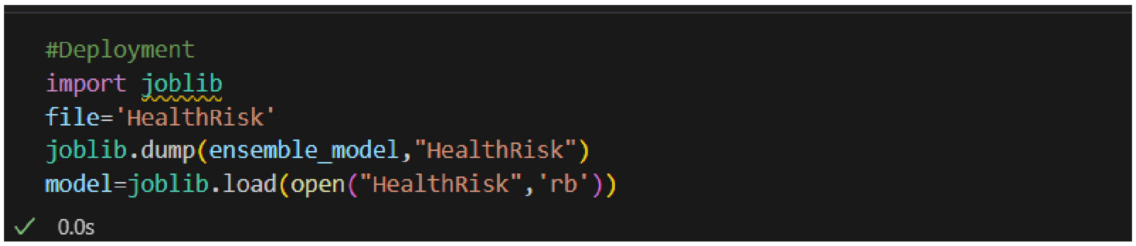


the top left cell, there are 572 instances where the true label was 0 and the model predicted it as 0. Similarly, in the bottom right cell, there are 1269 instances where the true label was 3 and the model predicted it as 3.

Deployment

After we build the models we choose KNN to make deployment.

We complete the deployment in VS



```
#Deployment
import joblib
file='HealthRisk'
joblib.dump(ensemble_model,"HealthRisk")
model=joblib.load(open("HealthRisk",'rb'))
```

We add this code to model notebook

Then we make new notebook called deployment.py

Streamlit Health Risk Prediction App

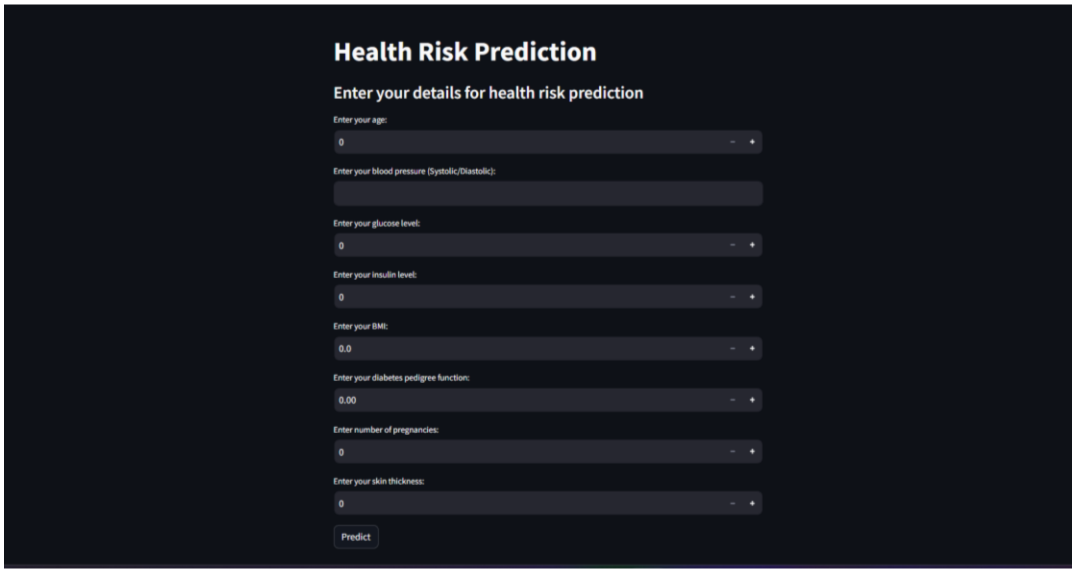
This code sets up a Streamlit web application that predicts health risks related to diabetes and heart attacks based on user inputs. Here's a detailed breakdown of the main components and functionalities of the app:

Main Components

1. **Imports and Model Loading**
   - Import necessary libraries for the app, including Streamlit, requests, pandas, joblib, and others.
   - Load a pre-trained machine learning model using joblib.load.
2. **Functions**
   - load_lottie(url): Loads a Lottie animation from a given URL.
   - predict(features): Uses the loaded model to make predictions based on the input features.
3. **Streamlit Page Configuration**
   - Configure the page with a title, icon, and initial sidebar state.
4. **Sidebar Menu**
   - Create a sidebar with navigation options: Home, Graphs, About, and Contact.
5. **Home Page**
   - Display input fields for user details necessary for health risk prediction: age, blood pressure, glucose level, insulin level, BMI, diabetes pedigree function, number of pregnancies, and skin thickness.
   - Handle and validate blood pressure input.
   - Process the inputs and prepare them for prediction.
   - Display the prediction result when the "Predict" button is clicked.

Then we run streamlit in terminal

Code (streamlit run deployment.py)



Health Risk Prediction

Enter your details for health risk prediction

Enter your age:
0

Enter your blood pressure (Systolic/Diastolic):

Enter your glucose level:
0

Enter your insulin level:
0

Enter your BMI:
0.0

Enter your diabetes pedigree function:
0.00

Enter number of pregnancies:
0

Enter your skin thickness:
0

Enter your age:

Enter your blood pressure (Systolic/Diastolic):

Enter your glucose level:

Enter your insulin level:

Enter your BMI:

Enter your diabetes pedigree function:

Enter number of pregnancies:

Enter your skin thickness:

After the patient enters this data, he or she clicks Predict the app give 4 responses

Frist : the patient has Both diabetes and heart attack

Second : the patient has Diabetes only

Third : the patient has Heart attack risk only

Fourth: the patient has None

Conclusion

In conclusion, our study aimed to develop and optimize machine learning models for predicting diabetes and heart attack risk using a merging dataset containing various patient attributes. We explored the performance of two models,KNN and SVM classifier, leveraging techniques like hyperparameter tuning with GridSearchCV to enhance predictive accuracy. Our findings revealed that both models achieved high accuracy, precision, recall, and F1-scores, indicating their effectiveness in predicting diabetes and heart attack risk.

The KNN, after hyperparameter tuning, demonstrated exceptional performance, achieving an accuracy of 96% on the test dataset. These results highlight the potential of machine learning algorithms in HealthRisk settings, offering valuable insights into factors influencing predicting diabetes and heart attack riskand paving the way for future research to refine and improve predictive models further.

Overall, our study underscores the importance of leveraging advanced analytics to augment medical decision-making and improve patient outcomes in preventive healthcare.