Name: Kaushik Kotian Roll no.:14 Div:D20B

EXPERIMENT NO : 6

Aim : Predicting Disease Risk from Patient Data: Leveraging Machine Learning in Healthcare

Theory :

In recent years, the healthcare industry has experienced a paradigm shift towards data-driven practices, leading to improvements in patient care, disease management, and overall healthcare outcomes. A key aspect of this transformation is the use of machine learning (ML) techniques to predict disease risks. These models are developed by analyzing vast amounts of patient data, including clinical records, demographics, and other health-related factors. The ultimate goal is to forecast the likelihood of diseases such as diabetes, heart disease, and cancer, enabling healthcare providers to intervene early and improve patient outcomes.

The Role of Machine Learning in Disease Prediction

Machine learning algorithms are highly effective in uncovering patterns in data that are not apparent to the human eye. These algorithms learn from historical patient data, extracting insights that can be applied to new, unseen data to predict disease risks. One of the primary benefits of using ML in healthcare is the ability to process and analyze large datasets. Healthcare organizations can now integrate data from various sources, such as electronic health records (EHRs), genetic data, medical imaging, and even wearable devices, to create more accurate and holistic predictions.

Among the most common algorithms used in healthcare for disease prediction are decision trees, random forests, support vector machines (SVM), neural networks, and logistic regression models. These models can be tailored to specific medical conditions by selecting relevant features, such as age, gender, genetic history, lab results, lifestyle factors, and pre-existing conditions.

Data Curation and Integration for Predictive Models

Data integration is critical for building robust predictive models. The Databricks Lakehouse for Healthcare and Life Sciences provides a platform that allows organizations to unlock all types of healthcare data, ranging from structured patient records to unstructured data such as clinical notes. By curating a longitudinal patient record—tracking a patient's health data over time—healthcare providers can gain insights into how various factors contribute to the development of chronic conditions.

Machine learning models can identify clinical and demographic covariates that influence the onset of diseases like diabetes, hypertension, or cardiovascular diseases. For example, in predicting heart disease, features such as blood pressure, cholesterol levels, age, smoking habits, and family history play a pivotal role. Once trained, these models can be applied to new patient data, aiding physicians in making real-time predictions and crafting personalized treatment plans.

Case Study: Predicting Heart Disease with Machine Learning

A real-world example of disease risk prediction using machine learning is a project conducted by a healthcare provider in the USA. This organization aimed to predict the likelihood of heart disease in its patient population, a critical need given the global impact of cardiovascular diseases. Heart disease is responsible for a significant number of deaths annually, and early detection is crucial for effective treatment.

The healthcare provider partnered with a Python development company to build a predictive model for heart disease detection. They utilized a dataset containing patient medical history, vital statistics, and lifestyle factors to train a machine learning model. The model was designed to estimate the probability of a patient having heart disease based on these features.

The team leveraged AutoML (automated machine learning) to automate the process of feature selection, model selection, and hyperparameter tuning. AutoML helped streamline the development process, enabling the data scientists to focus on refining the model's accuracy rather than manually testing various algorithms. The result was a predictive model that accurately forecasted heart disease risk, helping physicians intervene early and potentially saving lives.

Benefits and Challenges of Machine Learning in Healthcare

The use of machine learning in healthcare offers several benefits:

1. **Early Detection and Diagnosis**: Predictive models can detect the early stages of diseases, often before symptoms are apparent, allowing for timely interventions.
2. **Personalized Treatment Plans**: By analyzing individual patient data, ML models can suggest tailored treatment options, improving patient outcomes.
3. **Improved Operational Efficiency**: Predictive analytics can streamline operations by identifying high-risk patients, reducing hospital readmissions, and optimizing resource allocation.
4. **Data-Driven Insights**: Machine learning enables the discovery of new insights into disease progression and patient behavior, which can lead to the development of innovative therapies and care strategies.

However, integrating machine learning into healthcare systems is not without its challenges:

1. **Data Privacy and Security**: Healthcare data is sensitive, and protecting patient privacy is paramount. Organizations must ensure compliance with regulations such as the Health Insurance Portability and Accountability Act (HIPAA) when handling patient information.

2. **Data Quality and Standardization**: The success of predictive models depends on the quality of the data used. Inconsistent or incomplete data can lead to inaccurate predictions. Ensuring that data from various sources is standardized and cleaned is essential.
3. **Model Interpretability**: Healthcare professionals need to trust and understand the decisions made by machine learning models. In some cases, ML models, particularly deep learning models, can be seen as "black boxes," making it difficult for physicians to interpret the reasoning behind the predictions.
4. **Integration with Existing Systems**: Deploying machine learning models into clinical workflows can be complex. Healthcare systems need to be designed to support seamless integration with predictive analytics tools while ensuring ease of use for healthcare providers.

The Future of Predictive Analytics in Healthcare

As healthcare continues to embrace data-driven solutions, the role of machine learning in predicting disease risks is set to expand. With the rise of precision medicine, the ability to make accurate predictions based on genetic, environmental, and lifestyle factors will revolutionize the way diseases are diagnosed and treated. Technologies like cloud computing, big data analytics, and artificial intelligence are poised to further enhance the capabilities of machine learning models, enabling real-time predictions and personalized care.

In the near future, predictive models may be able to forecast disease outbreaks, predict treatment efficacy, and even recommend preventive measures tailored to individual patients. The integration of machine learning into healthcare systems will undoubtedly improve patient outcomes, reduce healthcare costs, and ultimately lead to better health for individuals and populations alike.

Dataset Glossary (Column-wise)

Patient ID - Unique identifier for each patient

Age - Age of the patient

Sex - Gender of the patient (Male/Female)

Cholesterol - Cholesterol levels of the patient

Blood Pressure - Blood pressure of the patient (systolic/diastolic)

Heart Rate - Heart rate of the patient

Diabetes - Whether the patient has diabetes (Yes/No)

Family History - Family history of heart-related problems (1: Yes, 0: No)

Smoking - Smoking status of the patient (1: Smoker, 0: Non-smoker)

Obesity - Obesity status of the patient (1: Obese, 0: Not obese)

Alcohol Consumption - Level of alcohol consumption by the patient (None/Light/Moderate/Heavy)

Exercise Hours Per Week - Number of exercise hours per week

Diet - Dietary habits of the patient (Healthy/Average/Unhealthy)

Previous Heart Problems - Previous heart problems of the patient (1: Yes, 0: No)

Medication Use - Medication usage by the patient (1: Yes, 0: No)

Stress Level - Stress level reported by the patient (1-10)

Sedentary Hours Per Day - Hours of sedentary activity per day

Income - Income level of the patient

BMI - Body Mass Index (BMI) of the patient

Triglycerides - Triglyceride levels of the patient

Physical Activity Days Per Week - Days of physical activity per week

Sleep Hours Per Day - Hours of sleep per day

Country - Country of the patient

Continent - Continent where the patient resides

Hemisphere - Hemisphere where the patient resides

Heart Attack Risk - Presence of heart attack risk (1: Yes, 0: No)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data = pd.read_csv("/content/heart_attack_prediction_dataset.csv")
data.head(5)
```

	Patient ID	Age	Sex	Cholesterol	Blood Pressure	Heart Rate	Diabetes	Family History	Smoking	Obesity	...	Sedentary Hours Per Day	Income	BMI	Trigl
0	BMW7812	67	Male	208	158/88	72	0	0	1	0	...	6.615001	261404	31.251233	
1	CZE1114	21	Male	389	165/93	98	1	1	1	1	...	4.963459	285768	27.194973	
2	BNI9906	21	Female	324	174/99	72	1	0	0	0	...	9.463426	235282	28.176571	
3	JLN3497	84	Male	383	163/100	73	1	1	1	0	...	7.648981	125640	36.464704	
4	GFO8847	66	Male	318	91/88	93	1	1	1	1	...	1.514821	160555	21.809144	

5 rows × 26 columns

```
data.shape
```

```
(8763, 26)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8763 entries, 0 to 8762
Data columns (total 26 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Patient ID                       8763 non-null   object 
 1   Age                              8763 non-null   int64  
 2   Sex                              8763 non-null   object 
 3   Cholesterol                      8763 non-null   int64  
 4   Blood Pressure                   8763 non-null   object 
 5   Heart Rate                       8763 non-null   int64  
 6   Diabetes                         8763 non-null   int64  
 7   Family History                   8763 non-null   int64  
 8   Smoking                          8763 non-null   int64  
 9   Obesity                          8763 non-null   int64  
 10  Alcohol Consumption              8763 non-null   int64  
 11  Exercise Hours Per Week          8763 non-null   float64
 12  Diet                             8763 non-null   object 
 13  Previous Heart Problems          8763 non-null   int64  
 14  Medication Use                   8763 non-null   int64  
 15  Stress Level                     8763 non-null   int64  
 16  Sedentary Hours Per Day          8763 non-null   float64
 17  Income                           8763 non-null   int64  
 18  BMI                              8763 non-null   float64
 19  Triglycerides                    8763 non-null   int64  
 20  Physical Activity Days Per Week  8763 non-null   int64  
 21  Sleep Hours Per Day              8763 non-null   int64  
 22  Country                          8763 non-null   object 
 23  Continent                        8763 non-null   object 
 24  Hemisphere                       8763 non-null   object 
 25  Heart Attack Risk                8763 non-null   int64  
dtypes: float64(3), int64(16), object(7)
memory usage: 1.7+ MB
```

```
for col in data.columns:
    unique_count = data[col].nunique()
    print(f"{col} unique value count: {unique_count}")
```

```
Patient ID unique value count: 8763
Age unique value count: 73
Sex unique value count: 2
Cholesterol unique value count: 281
Blood Pressure unique value count: 3915
Heart Rate unique value count: 71
Diabetes unique value count: 2
Family History unique value count: 2
Smoking unique value count: 2
Obesity unique value count: 2
Alcohol Consumption unique value count: 2
Exercise Hours Per Week unique value count: 8763
Diet unique value count: 3
Previous Heart Problems unique value count: 2
Medication Use unique value count: 2
Stress Level unique value count: 10
Sedentary Hours Per Day unique value count: 8763
Income unique value count: 8615
BMI unique value count: 8763
Triglycerides unique value count: 771
Physical Activity Days Per Week unique value count: 8
Sleep Hours Per Day unique value count: 7
Country unique value count: 20
Continent unique value count: 6
Hemisphere unique value count: 2
Heart Attack Risk unique value count: 2
```

Preprocessing

```
data.drop("Patient ID", axis=1, inplace=True)
```

```
nan_count = data.isnull().sum().sum()
# data.dropna(inplace=True)
nan_count
```

```
0
```

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

object_columns = ["Sex", "Diet", "Country", "Continent", "Hemisphere"]
```

```
for col in data[object_columns]:
    data[col] = label_encoder.fit_transform(data[col])
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8763 entries, 0 to 8762
Data columns (total 25 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Age                              8763 non-null   int64  
 1   Sex                              8763 non-null   int64  
 2   Cholesterol                      8763 non-null   int64  
 3   Blood Pressure                   8763 non-null   object 
 4   Heart Rate                       8763 non-null   int64  
 5   Diabetes                         8763 non-null   int64  
 6   Family History                   8763 non-null   int64  
 7   Smoking                          8763 non-null   int64  
 8   Obesity                          8763 non-null   int64  
 9   Alcohol Consumption              8763 non-null   int64  
 10  Exercise Hours Per Week          8763 non-null   float64
 11  Diet                             8763 non-null   int64  
 12  Previous Heart Problems          8763 non-null   int64  
 13  Medication Use                   8763 non-null   int64  
 14  Stress Level                     8763 non-null   int64  
 15  Sedentary Hours Per Day          8763 non-null   float64
 16  Income                           8763 non-null   int64  
 17  BMI                              8763 non-null   float64
 18  Triglycerides                    8763 non-null   int64  
 19  Physical Activity Days Per Week  8763 non-null   int64  
 20  Sleep Hours Per Day              8763 non-null   int64  
 21  Country                          8763 non-null   int64  
 22  Continent                        8763 non-null   int64  
 23  Hemisphere                       8763 non-null   int64  
 24  Heart Attack Risk                8763 non-null   int64  
dtypes: float64(3), int64(21), object(1)
memory usage: 1.7+ MB
```

```
data[['Systolic Pressure', 'Diastolic Pressure']] = data['Blood Pressure'].str.split('/', expand=True)

data.drop('Blood Pressure', axis=1, inplace=True)

data['Systolic Pressure'] = data['Systolic Pressure'].astype(int)
data['Diastolic Pressure'] = data['Diastolic Pressure'].astype(int)
```

```
data.head(5)
```

	Age	Sex	Cholesterol	Heart Rate	Diabetes	Family History	Smoking	Obesity	Alcohol Consumption	Exercise Hours Per Week	...	BMI	Triglycerides	Physical Activity Days Per Week
0	67	1	208	72	0	0	1	0	0	4.168189	...	31.251233	286	0
1	21	1	389	98	1	1	1	1	1	1.813242	...	27.194973	235	1
2	21	0	324	72	1	0	0	0	0	2.078353	...	28.176571	587	4
3	84	1	383	73	1	1	1	0	1	9.828130	...	36.464704	378	3
4	66	1	318	93	1	1	1	1	0	5.804299	...	21.809144	231	1

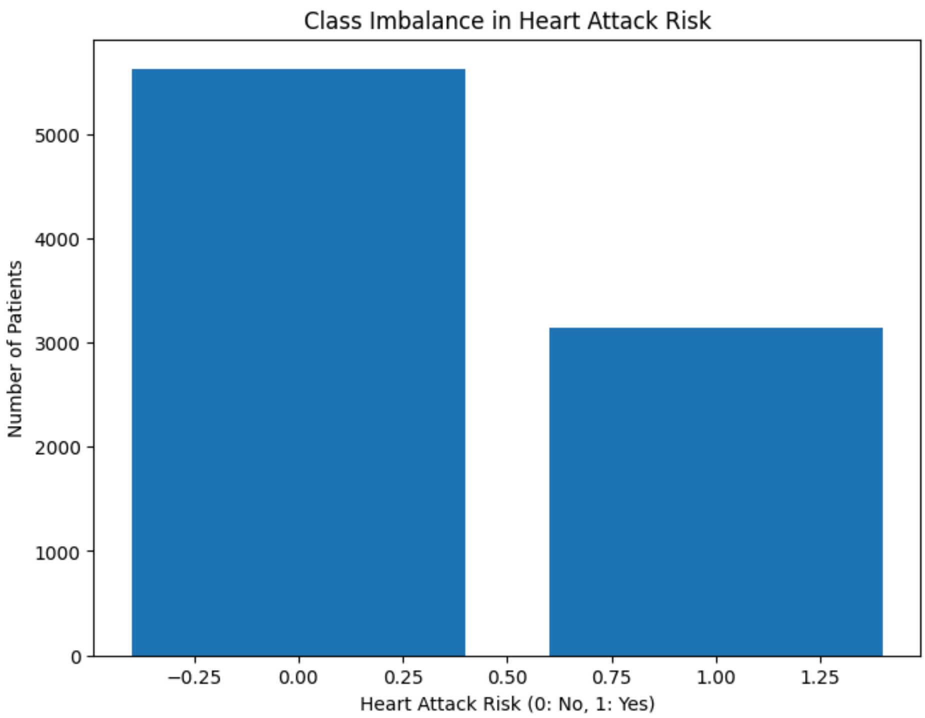
5 rows × 26 columns

```
column_to_move = 'Heart Attack Risk'

data = data[[col for col in data.columns if col != column_to_move] + [column_to_move]]

data #male 1
```

	Age	Sex	Cholesterol	Heart Rate	Diabetes	Family History	Smoking	Obesity	Alcohol Consumption	Exercise Hours Per Week	...	BMI	Triglycerides	Physic Activi Days P We
0	67	1	208	72	0	0	1	0	0	4.168189	...	31.251233	286	
1	21	1	389	98	1	1	1	1	1	1.813242	...	27.194973	235	
2	21	0	324	72	1	0	0	0	0	2.078353	...	28.176571	587	
3	84	1	383	73	1	1	1	0	1	9.828130	...	36.464704	378	
4	66	1	318	93	1	1	1	1	0	5.804299	...	21.809144	231	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8758	60	1	121	61	1	1	1	0	1	7.917342	...	19.655895	67	
8759	28	0	120	73	1	0	0	1	0	16.558426	...	23.993866	617	
8760	47	1	250	105	0	1	1	1	1	3.148438	...	35.406146	527	
8761	36	1	178	60	1	0	1	0	0	3.789950	...	27.294020	114	
8762	25	0	356	75	1	1	0	0	1	18.081748	...	32.914151	180	

8763 rows × 26 columns

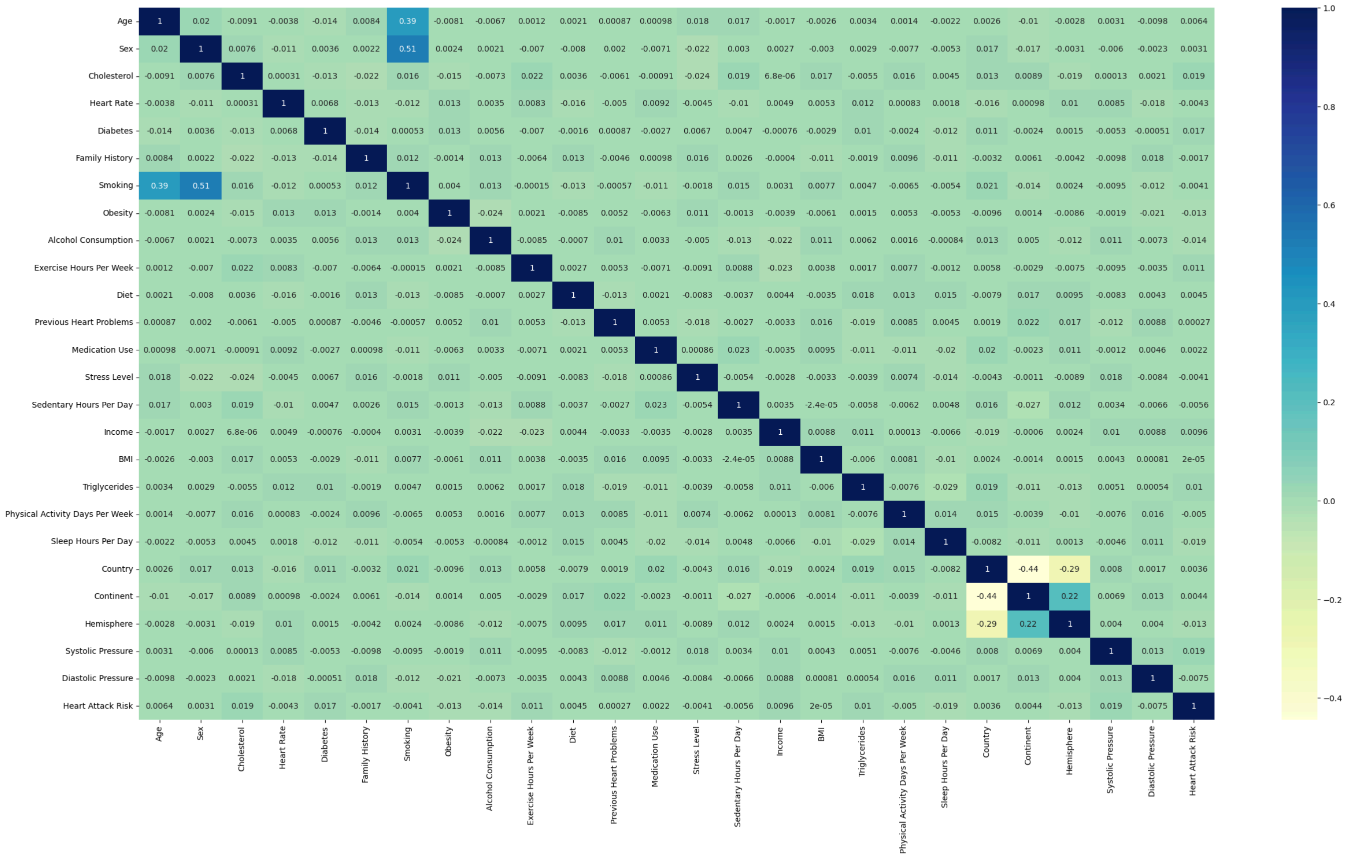
```
import matplotlib.pyplot as plt

target_column = 'Heart Attack Risk'
target_counts = data[target_column].value_counts()

plt.figure(figsize=(8, 6))
plt.bar(target_counts.index, target_counts.values)
plt.title('Class Imbalance in Heart Attack Risk')
plt.xlabel('Heart Attack Risk (0: No, 1: Yes)')
plt.ylabel('Number of Patients')
plt.show()

print(f"Class Distribution:\n{target_counts}")
```




```
Class Distribution:
Heart Attack Risk
0    5624
1    3139
```

```
plt.figure(figsize=(30,16))
sns.heatmap(data.corr(), annot=True, cmap="YlGnBu")
```

<Axes: >




```
print(data.corr()["Heart Attack Risk"].abs().sort_values(ascending=False))
```

```
Heart Attack Risk                  1.000000
Cholesterol                        0.019340
Systolic Pressure                  0.018585
Sleep Hours Per Day                0.018528
Diabetes                           0.017225
Alcohol Consumption                0.013778
Obesity                            0.013318
Hemisphere                         0.012704
Exercise Hours Per Week            0.011133
Triglycerides                      0.010471
Income                             0.009628
Diastolic Pressure                 0.007509
Age                                0.006403
Sedentary Hours Per Day            0.005613
Physical Activity Days Per Week    0.005014
Diet                               0.004540
Continent                          0.004446
Heart Rate                         0.004251
Stress Level                       0.004111
Smoking                            0.004051
Country                            0.003550
Sex                                0.003095
Medication Use                     0.002234
Family History                     0.001652
Previous Heart Problems            0.000274
BMI                                0.000020
Name: Heart Attack Risk, dtype: float64
```

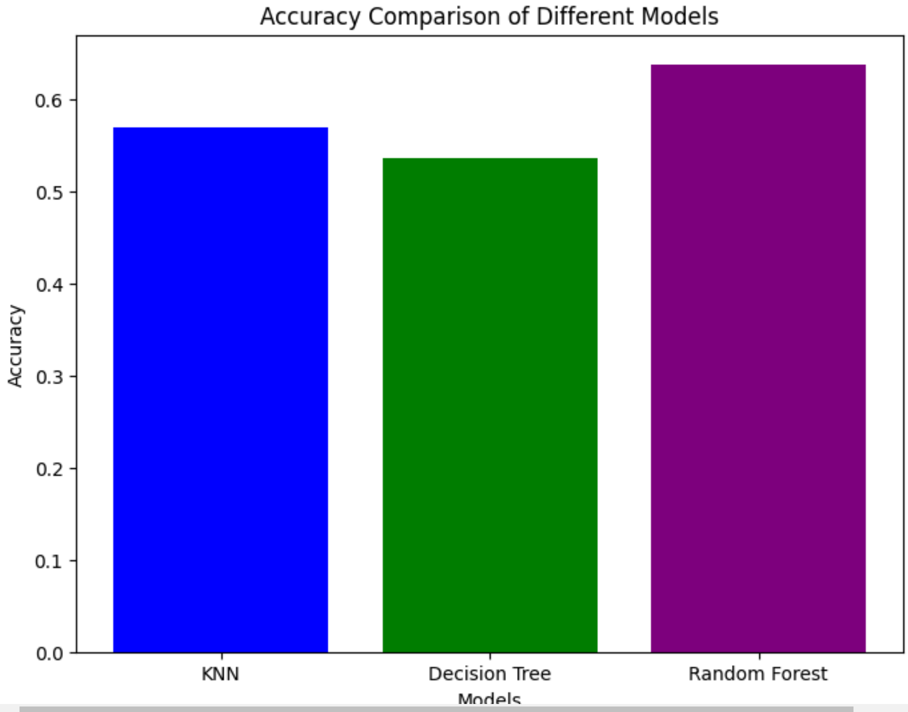
```
from sklearn.model_selection import train_test_split

X = data.drop(['Heart Attack Risk'], axis=1)
y = data['Heart Attack Risk']

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=42)
train_data = X_train.join(y_train)

X_train, y_train = train_data.drop(['Heart Attack Risk'], axis=1), train_data['Heart Attack Risk']
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
```

```python
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# KNN
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)
accuracy_knn = accuracy_score(y_test, y_pred_knn)
precision_knn = precision_score(y_test, y_pred_knn)
recall_knn = recall_score(y_test, y_pred_knn)
f1_knn = f1_score(y_test, y_pred_knn)
print("KNN - Accuracy:", accuracy_knn)
print("KNN - Precision:", precision_knn)
print("KNN - Recall:", recall_knn)
print("KNN - F1 Score:", f1_knn)

# Decision Tree
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)
accuracy_dt = accuracy_score(y_test, y_pred_dt)
precision_dt = precision_score(y_test, y_pred_dt)
recall_dt = recall_score(y_test, y_pred_dt)
f1_dt = f1_score(y_test, y_pred_dt)
print("\nDecision Tree - Accuracy:", accuracy_dt)
print("Decision Tree - Precision:", precision_dt)
print("Decision Tree - Recall:", recall_dt)
print("Decision Tree - F1 Score:", f1_dt)



# Random Forest
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
precision_rf = precision_score(y_test, y_pred_rf)
recall_rf = recall_score(y_test, y_pred_rf)
f1_rf = f1_score(y_test, y_pred_rf)
print("\nRandom Forest - Accuracy:", accuracy_rf)
print("Random Forest - Precision:", precision_rf)
print("Random Forest - Recall:", recall_rf)
print("Random Forest - F1 Score:", f1_rf)
```

```
KNN - Accuracy: 0.5705591479650057
KNN - Precision: 0.3537519142419602
KNN - Recall: 0.2462686567164179
KNN - F1 Score: 0.29038340666247636

Decision Tree - Accuracy: 0.5370863446177254
Decision Tree - Precision: 0.3569230769230769
Decision Tree - Recall: 0.37100213219616207
Decision Tree - F1 Score: 0.36382645060115004

Random Forest - Accuracy: 0.6386458729554964
Random Forest - Precision: 0.39655172413793105
Random Forest - Recall: 0.024520255863539446
Random Forest - F1 Score: 0.04618473895582329
```

```python
import matplotlib.pyplot as plt

# Model names and their corresponding accuracies
models = ['KNN', 'Decision Tree', 'Random Forest']
accuracies = [accuracy_knn, accuracy_dt, accuracy_rf]

# Create the bar plot
plt.figure(figsize=(8, 6))
plt.bar(models, accuracies, color=['blue', 'green', 'purple'])

# Add labels and title
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Accuracy Comparison of Different Models')

# Display the plot
plt.show()
```



""" Predicts the risk of heart attack based on input parameters.

Args: age: Age of the patient. sex: Gender of the patient (0 for Female, 1 for Male). cholesterol: Cholesterol levels of the patient. systolic_pressure: Systolic blood pressure of the patient. diastolic_pressure: Diastolic blood pressure of the patient. heart_rate: Heart rate of the patient. diabetes: Whether the patient has diabetes (1 for Yes, 0 for No). family_history: Family history of heart-related problems (1 for Yes, 0 for No). smoking: Smoking status of the patient (1 for Smoker, 0 for Non-smoker). obesity: Obesity status of the patient (1 for Obese, 0 for Not obese). alcohol_consumption: Level of alcohol consumption by the patient (0 for None, 1 for Light, 2 for Moderate, 3 for Heavy). exercise_hours_per_week: Number of exercise hours per week. diet: Dietary habits of the patient (0 for Average, 1 for Healthy, 2 for Unhealthy). previous_heart_problems: Previous heart problems of the patient (1 for Yes, 0 for No). medication_use: Medication usage by the patient (1 for Yes, 0 for No). stress_level: Stress level reported by the patient (1-10). sedentary_hours_per_day: Hours of sedentary activity per day. income: Income level of the patient. bmi: Body Mass Index (BMI) of the patient. triglycerides: Triglyceride levels of the patient. physical_activity_days_per_week: Days of physical activity per week. sleep_hours_per_day: Hours of sleep per day. country: Country of the patient (encoded value). continent: Continent where the patient resides (encoded value). hemisphere: Hemisphere where the patient resides (encoded value).

Returns: A value depicting the risk of heart attack (1 for Yes, 0 for No). """

```
def predict_heart_attack_risk_multiple_models(age, sex, cholesterol, systolic_pressure, diastc
                               heart_rate, diabetes, family_history, smoking, obesity,
                               alcohol_consumption, exercise_hours_per_week, diet,
                               previous_heart_problems, medication_use, stress_level,
                               sedentary_hours_per_day, income, bmi, triglycerides,
                               physical_activity_days_per_week, sleep_hours_per_day,
                               country, continent, hemisphere):

  # Create a DataFrame with the input values
  input_data = pd.DataFrame({
      'Age': [age],
      'Sex': [sex],
      'Cholesterol': [cholesterol],
      'Systolic Pressure': [systolic_pressure],
      'Diastolic Pressure': [diastolic_pressure],
      'Heart Rate': [heart_rate],
      'Diabetes': [diabetes],
      'Family History': [family_history],
      'Smoking': [smoking],
      'Obesity': [obesity],
      'Alcohol Consumption': [alcohol_consumption],
      'Exercise Hours Per Week': [exercise_hours_per_week],
      'Diet': [diet],
      'Previous Heart Problems': [previous_heart_problems],
      'Medication Use': [medication_use],
      'Stress Level': [stress_level],
      'Sedentary Hours Per Day': [sedentary_hours_per_day],
      'Income': [income],
      'BMI': [bmi],
      'Triglycerides': [triglycerides],
      'Physical Activity Days Per Week': [physical_activity_days_per_week],
      'Sleep Hours Per Day': [sleep_hours_per_day],
      'Country': [country],
      'Continent': [continent],
      'Hemisphere': [hemisphere]
  })

  # Use the trained models to predict the risk
```

age: 45

sex: 1

cholesterol: 200

systolic_pressure: 120

diastolic_pressure: 80

heart_rate: 70

diabetes: 0

family_history: 1

smoking: 0

obesity: 0

alcohol_consumption: 1

exercise_hours_per_week: 5

diet:

```
    prediction_knn = knn.predict(input_data[X_train.columns])[0]
    prediction_dt = dt.predict(input_data[X_train.columns])[0]
    prediction_rf = rf.predict(input_data[X_train.columns])[0]

    return prediction_knn, prediction_dt, prediction_rf


# Example usage:
age = 45  #@param {type:"number"}
sex = 1  #@param {type:"number"}
cholesterol = 200  #@param {type:"number"}
systolic_pressure = 120  #@param {type:"number"}
diastolic_pressure = 80  #@param {type:"number"}
heart_rate = 70  #@param {type:"number"}
diabetes = 0  #@param {type:"number"}
family_history = 1  #@param {type:"number"}
smoking = 0  #@param {type:"number"}
obesity = 0  #@param {type:"number"}
alcohol_consumption = 1  #@param {type:"number"}
exercise_hours_per_week = 5  #@param {type:"number"}
diet = 1  #@param {type:"number"}
previous_heart_problems = 0  #@param {type:"number"}
medication_use = 0  #@param {type:"number"}
stress_level = 6  #@param {type:"number"}
sedentary_hours_per_day = 8  #@param {type:"number"}
income = 50000  #@param {type:"number"}
bmi = 25  #@param {type:"number"}
triglycerides = 150  #@param {type:"number"}
physical_activity_days_per_week = 4  #@param {type:"number"}
sleep_hours_per_day = 7  #@param {type:"number"}
country = 10  #@param {type:"number"}
continent = 2  #@param {type:"number"}
hemisphere = 1  #@param {type:"number"}

risk_knn, risk_dt, risk_rf = predict_heart_attack_risk_multiple_models(age, sex, cholesterol,
                                 heart_rate, diabetes, family_history, smoking, obesity,
                                 alcohol_consumption, exercise_hours_per_week, diet,
                                 previous_heart_problems, medication_use, stress_level,
                                 sedentary_hours_per_day, income, bmi, triglycerides,
                                 physical_activity_days_per_week, sleep_hours_per_day,
                                 country, continent, hemisphere)

print("Heart Attack Risk (KNN):", risk_knn)
print("Heart Attack Risk (Decision Tree):", risk_dt)

print("Heart Attack Risk (Random Forest):", risk_rf)
```

1

previous_heart_problems: 0

medication_use: 0

stress_level: 6

sedentary_hours_per_day: 8

income: 50000

bmi: 25

triglycerides: 150

physical_activity_days_per_week: 4

sleep_hours_per_day: 7

country: 10

continent: 2

hemisphere: 1

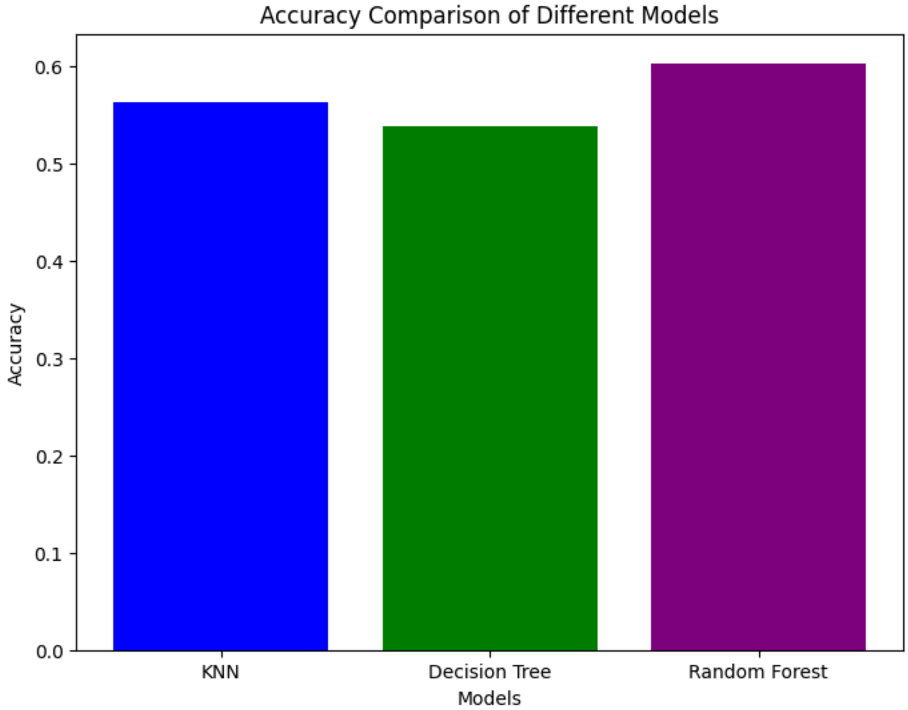
```
Heart Attack Risk (KNN): 1
Heart Attack Risk (Decision Tree): 0
Heart Attack Risk (Random Forest): 0
```

Top correlation

```
selected_features = ['Cholesterol', 'Sleep Hours Per Day', 'Diabetes', 'Alcohol Consumption', 'Obesity', 'Exercise Hours Per Week', 'Triglycerides', 'Heart Attack Risk']
reduced_data = data[selected_features]

X = reduced_data.drop(['Heart Attack Risk'], axis=1)
y = reduced_data['Heart Attack Risk']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# KNN
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)
accuracy_knn = accuracy_score(y_test, y_pred_knn)
precision_knn = precision_score(y_test, y_pred_knn)
recall_knn = recall_score(y_test, y_pred_knn)
f1_knn = f1_score(y_test, y_pred_knn)
print("KNN - Accuracy:", accuracy_knn)
print("KNN - Precision:", precision_knn)
print("KNN - Recall:", recall_knn)
print("KNN - F1 Score:", f1_knn)

# Decision Tree
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)
accuracy_dt = accuracy_score(y_test, y_pred_dt)
precision_dt = precision_score(y_test, y_pred_dt)
recall_dt = recall_score(y_test, y_pred_dt)
f1_dt = f1_score(y_test, y_pred_dt)
print("\nDecision Tree - Accuracy:", accuracy_dt)
print("Decision Tree - Precision:", precision_dt)
print("Decision Tree - Recall:", recall_dt)
print("Decision Tree - F1 Score:", f1_dt)

# Random Forest
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
precision_rf = precision_score(y_test, y_pred_rf)
recall_rf = recall_score(y_test, y_pred_rf)
f1_rf = f1_score(y_test, y_pred_rf)
print("\nRandom Forest - Accuracy:", accuracy_rf)
print("Random Forest - Precision:", precision_rf)
print("Random Forest - Recall:", recall_rf)
print("Random Forest - F1 Score:", f1_rf)
```

```
# Model names and their corresponding accuracies
models = ['KNN', 'Decision Tree',  'Random Forest']
accuracies = [accuracy_knn, accuracy_dt, accuracy_rf]

# Create the bar plot
plt.figure(figsize=(8, 6))
plt.bar(models, accuracies, color=['blue', 'green','purple'])

# Add labels and title
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Accuracy Comparison of Different Models')

# Display the plot
plt.show()
```

```
KNN - Accuracy: 0.5637124381894256
KNN - Precision: 0.33947772657450076
KNN - Recall: 0.23560767590618337
KNN - F1 Score: 0.2781623662680931

Decision Tree - Accuracy: 0.5386078356789654
Decision Tree - Precision: 0.3645320197044335
Decision Tree - Recall: 0.39445628997867804
Decision Tree - F1 Score: 0.3789042498719918

Random Forest - Accuracy: 0.603271205781666
Random Forest - Precision: 0.35924932975871315
Random Forest - Recall: 0.14285714285714285
Random Forest - F1 Score: 0.2044241037376049
```



Predicting using user input. Algorithms used KNN, DT, RF

```
def predict_heart_attack_risk_selected_features(cholesterol, sleep_hours_per_day, diabetes, alcohol_consu
  input_data = pd.DataFrame({
      'Cholesterol': [cholesterol],
      'Sleep Hours Per Day': [sleep_hours_per_day],
      'Diabetes': [diabetes],
      'Alcohol Consumption': [alcohol_consumption],
      'Obesity': [obesity],
      'Exercise Hours Per Week': [exercise_hours_per_week],
      'Triglycerides': [triglycerides]
  })

  prediction_knn = knn.predict(input_data)[0]
```

cholesterol: 200

sleep_hours_per_day: 7

diabetes: 0

alcohol_consumption: 1