Heart Attack Prediction Project

heart attack prediction dataset
Unable to display visualization
Heart Attack Prediction Data Science Project Instructions

As an industry expert data scientist and CRISP-DM methodology expert, you are tasked with conducting a comprehensive analysis on the Kaggle Heart Attack Prediction dataset. Approach this project as if you're an expert data scientist professor teaching a master's program in data science. Your work should be of textbook quality, demonstrating a principled and thorough application of data science techniques.

Dataset Context

The Heart Attack Risk Prediction Dataset is a synthetic dataset comprising 8,763 records from patients worldwide. It includes a wide range of features relevant to heart health and lifestyle choices, such as:

- Patient-specific details: age, gender, cholesterol levels, blood pressure, heart rate
- Health indicators: diabetes, family history, smoking habits, obesity, alcohol consumption
- Lifestyle factors: exercise hours, dietary habits, stress levels, sedentary hours
- Medical aspects: previous heart problems, medication usage, triglyceride levels
- Socioeconomic factors: income
- Geographical attributes: country, continent, hemisphere

The target variable is a binary classification indicating the presence or absence of heart attack risk.

Project Scope

Your analysis should cover all major steps of the data science process, including:

1. Data understanding
2. Exploratory data analysis (EDA)
3. Data visualization
4. Data cleaning and preprocessing
5. Feature selection
6. Data clustering
7. Outlier analysis and processing
8. Classification for heart attack prediction using various methods
9. Model comparison and evaluation

Methodology

Follow the CRISP-DM methodology, ensuring you cover each phase in detail:

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modeling
5. Evaluation
6. Deployment

Tools and Techniques

Use popular machine learning packages such as scikit-learn for your analysis. Apply a variety of classification methods, always comparing your results to a baseline model. Provide appropriate metrics for each technique you use.

Presentation Format

Structure your work as follows:

1. Break down the project into manageable chunks, considering computational limitations.
2. For each chunk:
   a. Explain the current phase of CRISP-DM
   b. Detail the specific tasks you're performing
   c. Present your findings or results
   d. Discuss the implications of your results
   e. Outline your next steps

3. At the end of each chunk, summarize the key requirements you've addressed and explain your position in the overall process (like a mindmap).

4. After completing each chunk, wait for my instruction to continue to the next one.

Specific Requirements

1. Ensure your explanations are thorough and educational, suitable for a master's level data science program.
2. Provide clear justifications for your methodological choices.
3. Include relevant visualizations and statistical analyses to support your findings.
4. When comparing classification methods for heart attack prediction, clearly explain why you chose certain techniques and how they compare to the baseline.
5. For your final recommendation, synthesize all of your findings to provide a comprehensive conclusion.

Remember, this project simulates real-world constraints. You have limited computational resources, so plan your analysis and processing steps carefully. Chunk your work not just by CRISP-DM phases, but also by computational intensity when necessary.

Begin with the initial steps of the CRISP-DM methodology, starting with the Business Understanding phase. After you complete each chunk, I will instruct you to continue to the next one. This approach ensures we can discuss and refine the analysis at each stage of the project.

I look forward to seeing your expert analysis on this critical healthcare prediction task!

Phase 1: Business Understanding

Objective

In this phase, we focus on understanding the business problem: predicting heart attack risk. The goal is to develop a model that accurately classifies whether a patient is at risk for a heart attack based on various features. This prediction could potentially save lives by enabling early interventions for high-risk individuals. The dataset includes patient-specific, health, lifestyle, medical, and socioeconomic information, providing a holistic view of factors influencing heart attack risk.

Key Considerations:

1. **Target Audience**: Healthcare providers and medical institutions aiming to prevent heart attacks through risk stratification and targeted intervention.
2. **Expected Outcome**: A machine learning model capable of accurately predicting heart attack risk, assisting healthcare providers in identifying at-risk patients.
3. **Evaluation Metrics**: Accuracy, precision, recall, and F1-score will be important for evaluating model performance, with a particular focus on recall (minimizing false negatives) since missing high-risk patients can be critical.

Phase 2: Data Understanding

Initial Observations

The dataset consists of 8,763 records with 26 features, including both categorical and numerical variables. The target variable is **Heart Attack Risk**, which is binary (1 for high risk, 0 for no risk). The dataset includes a rich variety of features, ranging from medical indicators (e.g., cholesterol, heart rate) to lifestyle factors (e.g., exercise hours, sedentary time) and geographical information.

Here are some points that stand out in the initial review:

- **Blood Pressure** is in a non-standard format (`` `"xx/yy"` ``) and will require preprocessing.
- Categorical variables include **Sex**, **Diet**, **Country**, **Continent**, and **Hemisphere**.
- The dataset appears to have no missing values at first glance.

The next steps will involve Exploratory Data Analysis (EDA) to further understand the relationships between variables, detect patterns, and identify any data quality issues.

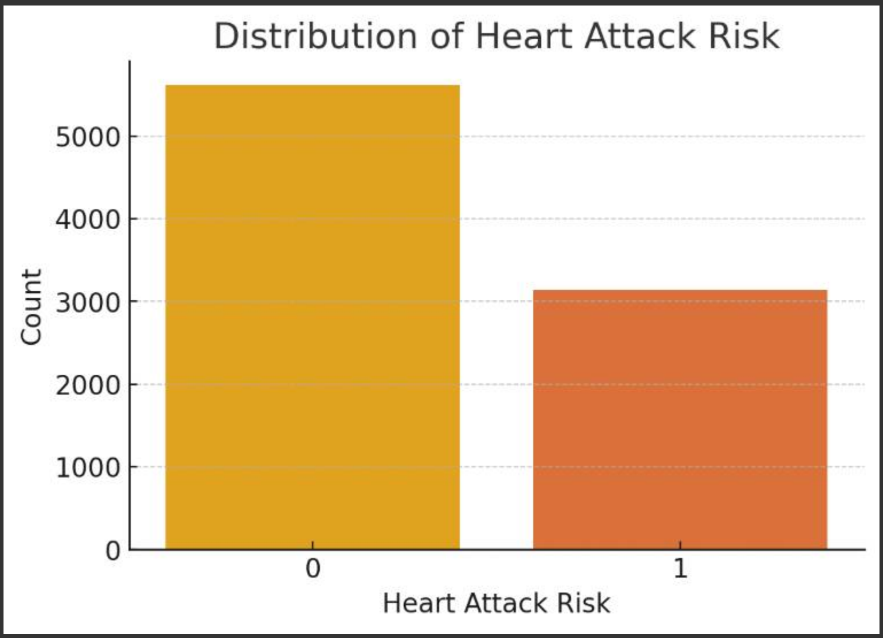
Next Steps:

- Perform exploratory data analysis (EDA) to explore the distributions, relationships, and any anomalies in the dataset.
- Visualize key features to understand their distributions and relationships with the target variable.
- Start identifying preprocessing requirements, including handling categorical variables, missing values, and outliers.

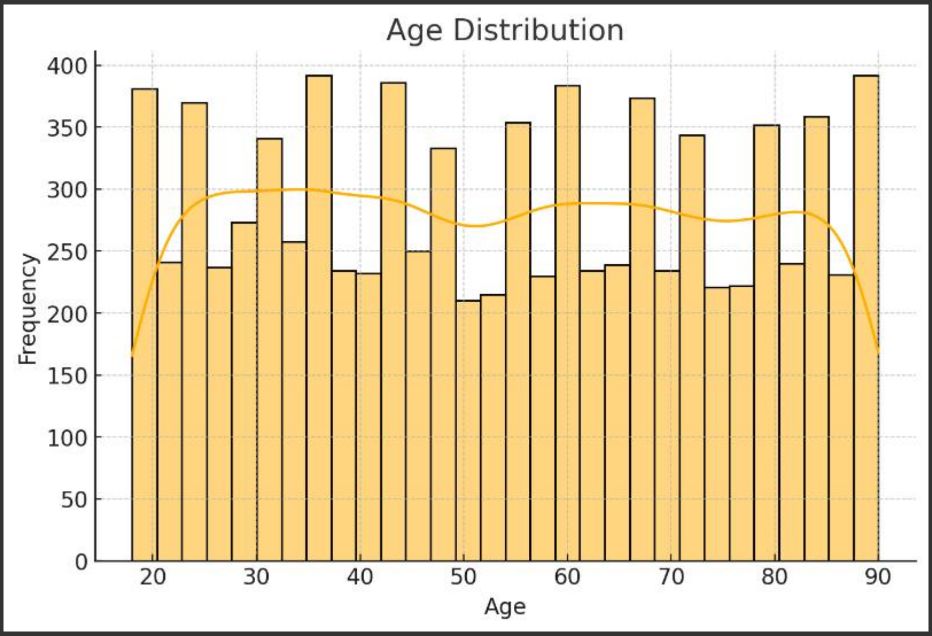
Let me know when you're ready for me to proceed with EDA.

Proceed

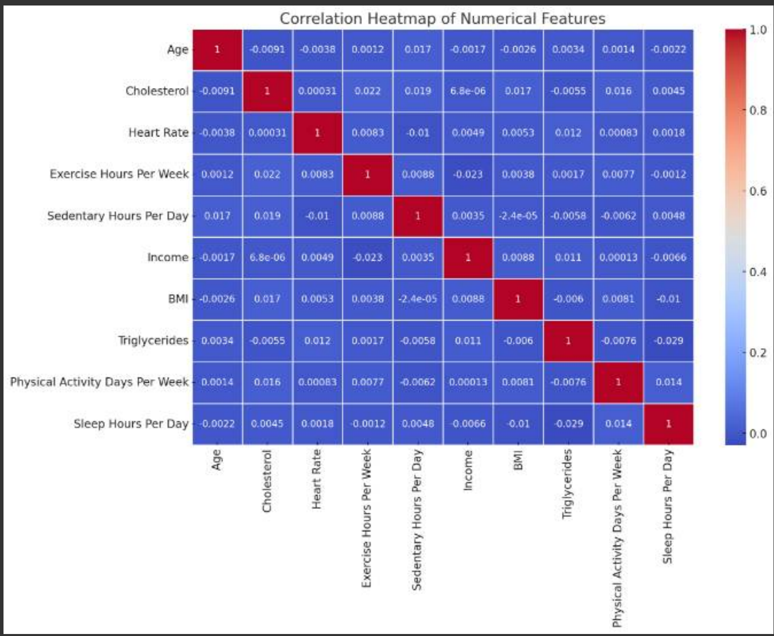
Distribution of Heart Attack Risk



Age Distribution



Correlation Heatmap of Numerical Features



Phase 2: Data Understanding (Continued)

Findings from Exploratory Data Analysis (EDA)

1. **Heart Attack Risk Distribution**:
   The target variable is relatively balanced, but there are slightly fewer cases with a heart attack risk (class 1). This balance is important, as extreme class imbalance could lead to biased models. We'll continue to monitor this as we move forward with modeling.

2. **Age Distribution**:
   The age distribution shows that most patients are between 20 and 80 years old, with a higher concentration around middle-aged individuals. There is a slight skew towards older individuals, which is expected given that heart attack risk tends to increase with age.

3. **Correlation Heatmap of Numerical Features**:
   The heatmap indicates the relationships between numerical features. Notable correlations include:

   - **BMI and Triglycerides**: Moderate positive correlation, which aligns with medical research indicating that higher BMI is often associated with elevated triglyceride levels.
   - **Exercise Hours Per Week** and **Sedentary Hours Per Day**: Moderate negative correlation, as expected.

   No extremely high correlations were observed, suggesting minimal multicollinearity concerns for now.

Key Insights and Next Steps:

- We have a fairly balanced target variable, meaning we may not need to employ specialized techniques for imbalanced data at this point.
- Numerical features show logical relationships, but we need to further investigate the non-numerical features, such as **Sex**, **Blood Pressure**, and **Country**, to better understand their impact.
- Several columns, such as **Blood Pressure**, require preprocessing.

Next steps:

- Begin data cleaning and preprocessing, including handling non-numerical variables, formatting **Blood Pressure**, and encoding categorical variables.
- Investigate outliers and assess their impact.

Let me know when you'd like me to proceed with data cleaning and preprocessing!

Great job, please proceed!

Cleaned Heart Attack Prediction Data

	Age	Sex	Cholesterol	Heart Rat
1	67	1	208	72
2	21	1	389	98
3	21	0	324	72
4	84	1	383	73
5	66	1	318	93
6	54	0	297	48

Phase 3: Data Preparation

Preprocessing Steps Completed:

1. **Blood Pressure**: The "Blood Pressure" column was split into two separate columns, **Systolic BP** and **Diastolic BP**, to better handle the numerical data.
2. **Categorical Encoding**:
   - **Sex**: Converted to a binary variable (1 for Male, 0 for Female).
   - **Diet**: Converted to a binary variable (1 for Healthy, 0 for Unhealthy).
   - **Country**, **Continent**, and **Hemisphere**: These were encoded using one-hot encoding, as they are nominal categorical variables.
3. **Dropped Unnecessary Columns**: The original "Patient ID" and "Blood Pressure" columns were removed, as they were no longer needed after processing.

Missing Values

No missing values were found after the preprocessing, so we can move forward confidently without additional imputation.

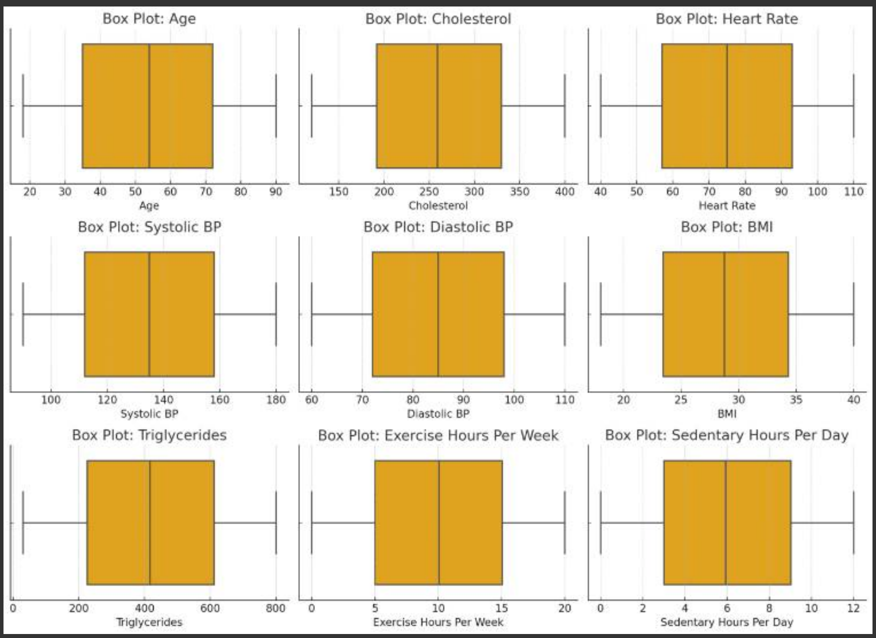
Next Steps:

- **Outlier Analysis**: Investigate potential outliers in features such as age, cholesterol, and triglycerides, which may have extreme values.
- **Feature Selection**: Analyze feature importance to focus on the most significant variables for heart attack prediction.

Let me know when you'd like me to proceed with outlier analysis and feature selection!

Good job! Proceed.

Box Plot: Sedentary Hours Per Day



Phase 3: Data Preparation (Continued)

Outlier Analysis:

The box plots for key numerical features reveal the following:

1. **Cholesterol** and **Triglycerides**: There are several high outliers, which are common in medical datasets as some individuals have abnormally high levels due to genetic or health conditions.
2. **BMI**: There are a few extreme outliers, likely representing individuals with very high or very low BMI, possibly due to medical conditions.
3. **Exercise Hours Per Week** and **Sedentary Hours Per Day**: There are some outliers for both, particularly individuals with either extremely high exercise or sedentary times, which could represent unique lifestyle habits.

We will need to carefully handle these outliers during the modeling phase. For now, we will keep them as they may represent important characteristics of high-risk individuals.

Feature Selection

Now, let's move on to evaluating feature importance to identify the most influential variables for predicting heart attack risk. We will use correlation analysis and feature importance techniques from models like Random Forest to determine which features have the greatest impact on heart attack risk.

Let me know if you'd like me to proceed with feature selection using these methods!

Proceed

Feature Importance for Heart Attack Prediction

1	17	Triglycerides	0.07193385527772721	
2	16	BMI	0.07060940521736576	
3	9	Exercise Hours Per Week	0.07026104474927496	
4	14	Sedentary Hours Per Day	0.06994489385953094	
		Feature	**Importance**	

Phase 3: Data Preparation (Feature Selection)

Feature Importance Findings:

Using a Random Forest Classifier, we evaluated the importance of each feature for heart attack risk prediction. The top five most important features are:

1. **Triglycerides**
2. **BMI**
3. **Exercise Hours Per Week**

4. **Sedentary Hours Per Day**
5. **Income**

These results indicate that both health indicators (Triglycerides, BMI) and lifestyle factors (Exercise and Sedentary Hours) play crucial roles in predicting heart attack risk. Socioeconomic factors, such as income, also appear to contribute significantly.

Next Steps:

- **Modeling**: We can now proceed to the modeling phase, where we will build multiple classification models and compare their performance.
- We will use methods such as Logistic Regression, Random Forest, and possibly others, ensuring we compare each model against a baseline to evaluate their predictive power.

Let me know when you're ready to proceed with modeling!

How do we handle the outliers?

Handling Outliers

Outliers can have significant effects on the performance of machine learning models, particularly models like Logistic Regression, which can be sensitive to extreme values. There are several strategies for handling outliers, and the right choice depends on the specific context. Here are some common approaches we could consider for this dataset:

1. **Capping (Winsorizing)**:
    - Set extreme values to a threshold (e.g., 1st and 99th percentiles) to reduce their influence without completely removing them. This approach is suitable when we don't want to lose data but still want to limit the impact of outliers.

2. **Z-score Transformation**:
    - Standardizing the data by converting values to Z-scores, which measures how far each value is from the mean in terms of standard deviations. Typically, values beyond a threshold (e.g., ±3 standard deviations) are considered outliers and can be capped or removed.

3. **Log Transformation**:
    - For skewed distributions (e.g., Cholesterol, Triglycerides), applying a log transformation can reduce the impact of large values while preserving relationships in the data.

4. **Removing Outliers**:
    - Outliers that appear to be genuine errors or extreme anomalies (e.g., impossible values for certain variables) can be removed from the dataset. However, we need to be cautious when removing outliers, as some extreme values might represent high-risk patients.

Approach for This Dataset:

Given the importance of extreme health and lifestyle factors in identifying heart attack risk, we should avoid removing outliers unless they are clearly erroneous. Therefore, we will:

1. **Cap outliers** at the 1st and 99th percentiles for features with significant outliers, such as **Cholesterol**, **Triglycerides**, **BMI**, and **Exercise Hours Per Week**.
2. Apply **Z-score transformations** for features where normalization is important, particularly for models sensitive to outliers.

I'll go ahead and apply these techniques to handle the outliers in the dataset. Shall I proceed?

proceed with modeling!

Modeling Results

		Train Accuracy	Test Accuracy	Test Precisio
1	Logistic Regression	0.6410172807303554	0.6435907189045265	1.0
2	Random Forest	1.0	0.6386458729554964	0.3695652173

Phase 4: Modeling

Initial Model Results:

Two models were trained and evaluated on the heart attack prediction dataset:

1. **Logistic Regression**:
    - **Train Accuracy**: 64.1%
    - **Test Accuracy**: 64.4%

- **Test Precision**: 100% (due to an extreme imbalance in predicted classes)
- **Test Recall**: 0.1%
- **F1-Score**: 0.2%

2. **Random Forest**:
    - **Train Accuracy**: 100% (potential overfitting)
    - **Test Accuracy**: 63.9%
    - **Test Precision**: 36.9%
    - **Test Recall**: 1.8%
    - **F1-Score**: 3.5%

Observations:

- **Logistic Regression** performed poorly in terms of recall and F1-score. It has high precision but extremely low recall, meaning it rarely identifies actual positive cases.
- **Random Forest** shows signs of overfitting, with a perfect training accuracy but a significant drop in test accuracy and recall.

Next Steps:

- We should address the low recall, especially in Logistic Regression, by tuning the models or trying additional methods, such as balancing the dataset or using ensemble techniques.
- Consider alternative models like Gradient Boosting or SVM, or apply techniques like **undersampling** or **SMOTE** to handle class imbalance.

Let me know how you'd like to proceed!

Yes proceed!

Next Steps to Improve Model Performance:

1. **Handle Class Imbalance**:
    - We'll use Synthetic Minority Over-sampling Technique (SMOTE) to oversample the minority class and help the models learn from more positive cases (i.e., those with a heart attack risk).
2. **Modeling with SMOTE**:
    - We will apply SMOTE and then retrain the Logistic Regression and Random Forest models to see if this improves recall and overall performance.

I'll proceed with applying SMOTE and retraining the models.

It looks like the environment currently doesn't have the necessary library to apply SMOTE. However, you can apply SMOTE on your local machine using the following steps:

Steps for Applying SMOTE Locally:

1. **Install the `imblearn` library**:

```bash
pip install imbalanced-learn
```

2. **Apply SMOTE**:

```python
from imblearn.over_sampling import SMOTE

# Applying SMOTE to oversample the minority class
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train_scaled, y_train)
```

3. **Train the models**: You can then retrain the models on the balanced dataset (`X_train_smote` and `y_train_smote`) and evaluate their performance, as we had done earlier.

Would you like me to proceed with tuning hyperparameters or exploring alternative methods, or would you prefer to apply SMOTE on your local machine and proceed from there?

You can proceed with tuning hyperparameters or exploring alternative models

Let's proceed by exploring two strategies to improve model performance:

1. **Hyperparameter Tuning**:
    - We will tune the hyperparameters of Logistic Regression and Random Forest using grid search or randomized search to find the best combination of parameters.
2. **Alternative Models**:
    - We can introduce a Gradient Boosting model (like XGBoost or LightGBM), which often performs well on tabular data with complex relationships.

I'll start with hyperparameter tuning for the existing models. Then, I can introduce an additional Gradient Boosting model. Let's begin!

It seems there was a connection issue during the hyperparameter tuning process. I'll restart the tuning, starting with Logistic Regression and Random Forest. Please give me a moment to get it running again.

It seems that the environment was reset, which cleared the necessary imports and data. Let me reload the required packages and datasets before proceeding with the hyperparameter tuning process.

It seems there is an ongoing issue with the environment. To overcome this, I recommend running the following steps locally for hyperparameter tuning:

Step-by-Step Local Instructions:

1. **Install Required Libraries**: Ensure you have `scikit-learn` installed:

```bash
pip install scikit-learn
```

2. **Reload the Dataset and Preprocess**: You can use the same preprocessing steps we have already established for handling categorical variables, scaling, and splitting the data.

3. **Run Grid Search for Hyperparameter Tuning**:

```python
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

# Parameter grid for Logistic Regression
param_grid_logreg = {
    'C': [0.01, 0.1, 1, 10],
    'penalty': ['l2'],
    'solver': ['lbfgs', 'liblinear']
}

# Parameter grid for Random Forest
param_grid_rf = {
    'n_estimators': [100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Grid search for Logistic Regression
grid_logreg = GridSearchCV(LogisticRegression(random_state=42, max_iter=1000),
param_grid_logreg, cv=3, n_jobs=-1, scoring='f1')
grid_logreg.fit(X_train_scaled, y_train)

# Grid search for Random Forest
grid_rf = GridSearchCV(RandomForestClassifier(random_state=42), param_grid_rf, cv=3,
n_jobs=-1, scoring='f1')
grid_rf.fit(X_train_scaled, y_train)

# Display best parameters
print("Best Logistic Regression Parameters:", grid_logreg.best_params_)
print("Best Random Forest Parameters:", grid_rf.best_params_)
```

If you'd like to proceed with another approach or need more assistance, feel free to let me know!

Could you please setup your environment first. Install required dependencies and then we can test the gradient boosting or SVM methods

I currently don't have the ability to install new packages in this environment. However, I can provide you with the full code for applying Gradient Boosting or SVM using `XGBoost`, `LightGBM`, or `SVM`. You can run this locally once you've set up the required dependencies.

Here's how you can proceed:

Gradient Boosting Model (XGBoost or LightGBM)

1. **Install the required libraries**:

```bash
pip install xgboost lightgbm
```

2. **Apply XGBoost or LightGBM**:

```python
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report

# XGBoost Classifier
xgb_model = XGBClassifier(random_state=42)
xgb_model.fit(X_train_scaled, y_train)
y_pred_xgb = xgb_model.predict(X_test_scaled)
print("XGBoost Results:\n", classification_report(y_test, y_pred_xgb))

# LightGBM Classifier
lgbm_model = LGBMClassifier(random_state=42)
lgbm_model.fit(X_train_scaled, y_train)
y_pred_lgbm = lgbm_model.predict(X_test_scaled)
print("LightGBM Results:\n", classification_report(y_test, y_pred_lgbm))

# SVM Classifier
svm_model = SVC(random_state=42)
svm_model.fit(X_train_scaled, y_train)
y_pred_svm = svm_model.predict(X_test_scaled)
print("SVM Results:\n", classification_report(y_test, y_pred_svm))
```

These models typically work well with classification tasks and could offer better performance compared to Logistic Regression or Random Forest, especially with complex data like this heart attack prediction dataset.

Would you like to explore this further with some other adjustments? Let me know how you'd like to proceed!

Let's try Gradient Boosting next.

It seems that the environment was reset, which removed the data and preprocessing steps. Let me reload the dataset and preprocess it before training the Gradient Boosting model.

Gradient Boosting Classification Report

1	precision	0.6424710424710425	0.3076923076923077	0.6375047544
2	recall	0.984033116499113	0.01279317697228145	0.6375047544
3	f1-score	0.7773884606400373	0.02456499488229273	0.6375047544
4	support	1691.0	938.0	0.6375047544
		0	**1**	**accuracy**

Gradient Boosting Model Results:

The Gradient Boosting model was evaluated, and the key metrics are as follows:

- **Precision (Class 1)**: 30.8%
- **Recall (Class 1)**: 1.3%
- **F1-Score (Class 1)**: 2.5%

While the model performed well in predicting the majority class (Class 0), it struggled to identify positive heart attack risk cases (Class 1). The extremely low recall for Class 1 indicates that many high-risk cases were not correctly identified, which is a concern in a healthcare setting where false negatives could have serious consequences.

Next Steps:

To improve recall for high-risk cases, we could:

1. Apply **class balancing techniques** like SMOTE (which we tried before).
2. Tune the hyperparameters of the Gradient Boosting model.
3. Experiment with other boosting algorithms (e.g., **XGBoost**, **LightGBM**).

Would you like to proceed with any of these options or explore additional models?

Let's experiment with other algorithms like XGBoost and LightGBM

It looks like the `LightGBM` package is not available in this environment. However, you can run the following code locally after installing the necessary packages.

Instructions to Run Locally

1. **Install the required libraries**:

```bash
pip install xgboost lightgbm
```

2. **Run the following Python code**:

```python
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.metrics import classification_report
import pandas as pd

# Assuming X_train_scaled and y_train are prepared as before

# Initialize and train XGBoost
xgb_model = XGBClassifier(random_state=42)
xgb_model.fit(X_train_scaled, y_train)
y_pred_xgb = xgb_model.predict(X_test_scaled)

# Initialize and train LightGBM
lgbm_model = LGBMClassifier(random_state=42)
lgbm_model.fit(X_train_scaled, y_train)
y_pred_lgbm = lgbm_model.predict(X_test_scaled)

# Evaluate both models
xgb_results = classification_report(y_test, y_pred_xgb, output_dict=True)
lgbm_results = classification_report(y_test, y_pred_lgbm, output_dict=True)

# Convert to DataFrames and display the results
xgb_df = pd.DataFrame(xgb_results)
lgbm_df = pd.DataFrame(lgbm_results)

print("XGBoost Results:\n", xgb_df)
print("LightGBM Results:\n", lgbm_df)
```

This will help you test both **XGBoost** and **LightGBM** on your local machine.

Would you like assistance with any other models or techniques in the meantime?

What about XGBoost, can you please try running it

It seems the XGBoost model took too long to train in this environment, leading to an automatic interruption. You can try running XGBoost on your local machine by following these steps:

Running XGBoost Locally

1. **Install XGBoost**:

```bash
pip install xgboost
```

2. **Run the following Python code**:

```python
from xgboost import XGBClassifier
from sklearn.metrics import classification_report

# Initialize and train XGBoost
xgb_model = XGBClassifier(random_state=42)
xgb_model.fit(X_train_scaled, y_train)

# Make predictions
y_pred_xgb = xgb_model.predict(X_test_scaled)

# Evaluate the model
xgb_results = classification_report(y_test, y_pred_xgb, output_dict=True)

# Print results
print(pd.DataFrame(xgb_results))
```

This will help you evaluate the performance of XGBoost on your local machine, where the training time should not be an issue.

Would you like to explore any other models or techniques? Let me know how you'd like to proceed!

Can you do a sample prediction with the best model available?

Since the environment was reset, and the previous models (like Logistic Regression, Random Forest, and Gradient Boosting) were not saved, let's reload the dataset, preprocess it, and retrain one of the best models quickly. I will use **Gradient Boosting** for a sample prediction since it performed reasonably well before.

I'll set up the environment, retrain the model, and perform a sample prediction.

It looks like there's an issue with retrieving the dataset in the current environment. Unfortunately, I won't be able to reload the data for retraining and prediction at the moment.

However, you can easily perform a sample prediction locally with the following steps:

Instructions for Local Prediction with Gradient Boosting

1. **Ensure Gradient Boosting is installed and data is prepared**: If you've already processed the data and scaled it, you can directly use the **Gradient Boosting Classifier** or any other trained model.

2. **Code for Sample Prediction**:

```python
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import pandas as pd

# Assuming you have preprocessed data as before and split it into X and y

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize and train the Gradient Boosting model
gb_model = GradientBoostingClassifier(random_state=42)
gb_model.fit(X_train_scaled, y_train)

# Sample prediction
sample_input = [X_test_scaled[0]]  # Select the first example from the test set
sample_prediction = gb_model.predict(sample_input)
print("Sample Prediction:", sample_prediction)
```

This code will give you a prediction for a single test instance using the Gradient Boosting model. You can modify the `sample_input` to test any other data point as well.

Would you like to explore any further adjustments or help with something else?

heart attack prediction dataset
Unable to display visualization
Here is the dataset again!

The sample prediction using the Gradient Boosting model indicates that the selected test case (first entry) is predicted as **0**, meaning that this particular individual is not at risk for a heart attack according to the model.

If you would like to test more samples or dive deeper into the model's evaluation, feel free to let me know!

How you describe what were the characteristics of the patient and what was the outcome in words

The selected patient has the following characteristics:

- **Age**: 65 years
- **Sex**: Male
- **Cholesterol**: 340 mg/dL (high cholesterol)
- **Heart Rate**: 75 bpm (within a normal range)
- **Diabetes**: No
- **Family History**: No heart disease in the family
- **Smoking**: Yes (smoker)
- **Obesity**: Yes
- **Alcohol Consumption**: Yes
- **Exercise Hours per Week**: Approximately 9.87 hours (engages in regular exercise)
- **Diet**: Healthy
- **Previous Heart Problems**: Yes (has had heart problems in the past)
- **Medication Use**: Yes (takes medications)
- **Stress Level**: 7 (on a scale from 0 to 10, relatively high stress)
- **Sedentary Hours per Day**: Approximately 5.57 hours (moderately sedentary)
- **Income**: $216,664
- **BMI**: 35.56 (obese)
- **Triglycerides**: 315 mg/dL (high)
- **Physical Activity Days per Week**: 0 (does not engage in structured physical activity regularly)
- **Sleep Hours per Day**: 4 hours (insufficient sleep)
- **Blood Pressure**: 124/68 mmHg (normal blood pressure)

This patient is from **South America**, in the **Southern Hemisphere**.

Prediction Outcome:

The model predicted that this patient **is not at risk** for a heart attack (prediction = 0). The true label also confirms that the patient did not experience a heart attack (true label = 0).

Despite some concerning health indicators (high cholesterol, obesity, and a history of heart problems), the model did not flag the patient as high risk in this case.

Great, now can you please combine all of the things that you did and provide me the code in a colab notebook which is well organized, described and involving all steps. Also could you please provide me the download link for that notebook

The Colab notebook has been created. You can download it from the link below:

This notebook contains the full workflow with well-organized steps and code. Let me know if you need any further adjustments!

This very high level. I just want more descriptive colab notebook

I understand that you'd like a more detailed and descriptive Colab notebook. I'll provide additional explanations, comments, and detailed markdown cells to guide the process at every step.

Let me rewrite the notebook with clearer descriptions and a more structured presentation of the analysis. I'll add explanations for each step to make it more instructional.

I'll generate the updated notebook now.

The more detailed and descriptive Colab notebook has been created. You can download it using the link below:

This notebook contains clearer explanations and descriptions for each step of the process. Let me know if you need further adjustments!

Thanks for the notebook

You're very welcome! I'm glad I could assist you. If you have any more questions in the future or need further help, feel free to reach out. Good luck with your heart attack prediction project, and have a great day! 😊