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

EXPERIMENT NO : 4

Aim : AI for medical prognosis

Theory :

The implications of prognosis

Prognostic studies aimed to understand the course, determinants, or probability of outcomes in a cohort of patients. Prognostic information is useful for the following reasons:

- To provide information to patients
- To identify target groups for treatment
- To target specific prognostic factors for modification throughout treatment
- To provide the basis of personalized or risk-based medicine
- The design of randomized trials

1. Overview of Diagnosis and Prognosis in Healthcare:

- Diagnosis refers to the identification of a disease or condition based on signs, symptoms, and medical test results. A diagnosis is crucial for understanding the current state of a patient's health. For example, diagnosing diabetes involves interpreting test results like blood sugar levels (e.g., fasting glucose or HbA1c), which directly point to the presence or absence of the disease.
- Prognosis, on the other hand, goes beyond diagnosis to predict the future progression of the disease. It offers an estimation of how a patient's condition will evolve, such as whether it will improve, worsen, or remain stable. In some cases, prognosis also includes life expectancy predictions, quality of life assessments, or the likelihood of complications. For instance, for a diabetic patient, the prognosis may involve estimating the risk of future complications like cardiovascular disease, kidney failure, or neuropathy.

2. The Role of AI in Medical Prognosis:

With the advancement of machine learning (ML) and AI technologies, healthcare has seen a major transformation in predicting patient outcomes. AI models can analyze large datasets of patient records, lab results, and imaging to predict the future course of diseases. These predictions enable healthcare providers to:

- Make early interventions, preventing disease progression.
- Tailor treatment plans according to patient risk profiles.
- Improve long-term patient outcomes by providing targeted, preventive care.

3. Logistic Regression for Disease Prognosis:

3.1 Overview of Logistic Regression:

- Logistic regression is a commonly used statistical method for binary classification problems, where the outcome is categorical (e.g., yes/no, disease/no disease). It is particularly suitable for medical prognosis because it calculates the probability of an event occurring (such as the onset of a disease) based on the patient's medical history and physiological measurements.
- The model estimates relationships between a dependent variable (e.g., whether the patient will develop complications) and independent variables (e.g., age, blood sugar levels, body mass index, etc.). Logistic regression can handle large datasets efficiently and provides interpretable results, making it ideal for healthcare settings where explainability is crucial.

3.2 Mathematical Representation:

- The logistic regression model uses the sigmoid function to map any input to a probability between 0 and 1. The model calculates the odds of the dependent variable being 1 (e.g., the patient developing complications) as a linear combination of independent variables: P(Y=1∣X)=11+e−(β0+β1X1+β2X2+...+βnXn)P(Y = 1 | X) = \frac{1}{1 + e^{-(β_0 + β_1 X_1 + β_2 X_2 + ... + β_n X_n)}}P(Y=1∣X)=1+e−(β0+β1X1+β2X2+...+βnXn)1
  Where:
    - P(Y=1∣X)P(Y = 1 | X)P(Y=1∣X) is the probability that the patient will experience a certain outcome (e.g., high risk of complications).
    - X1,X2,...,XnX_1, X_2, ..., X_nX1,X2,...,Xn are the features (age, blood sugar levels, BMI, etc.).
    - β1,β2,...,βnβ_1, β_2, ..., β_nβ1,β2,...,βn are the coefficients learned from the data, representing the weight each feature carries in predicting the outcome.
- If the probability is greater than a set threshold (e.g., 0.5), the patient is classified as "high risk"; otherwise, they are classified as "low risk."

4. Application of Logistic Regression to Diabetic Patient Dataset:

4.1 Dataset Description:

- The dataset under consideration contains records of diabetic patients, including various medical features such as:
    - Age: Older patients are often at higher risk of complications.
    - Gender: Male and female patients may have different risk profiles for diabetes-related complications.
    - Blood Sugar Levels (HbA1c): High HbA1c levels indicate poorly controlled diabetes and are associated with a higher risk of complications.
    - BMI (Body Mass Index): Overweight or obese patients are more prone to cardiovascular diseases and other complications.
    - Duration of Diabetes: Longer duration increases the likelihood of developing complications.
    - Medical History: Additional features like history of heart disease or kidney issues.

4.2 Building the Model:

- Step 1: Data Preprocessing:
    - The dataset is cleaned, handling missing values and outliers. Features such as age, BMI, and blood sugar levels are standardized to ensure the model's coefficients are properly weighted.
- Step 2: Feature Selection:
    - Features that are most relevant for predicting diabetic complications are selected. For example, age, BMI, HbA1c levels, and duration of diabetes might be chosen based on their clinical importance.
- Step 3: Model Training:
    - The logistic regression model is trained on a portion of the dataset using labeled data where patient outcomes are known (e.g., complications present or absent). The model uses the training data to learn the coefficients β1,β2,...,βnβ_1, β_2, ..., β_nβ1,β2,...,βn for each feature.
- Step 4: Model Testing and Evaluation:
    - The trained model is then tested on unseen data (test set). The model's performance is evaluated using metrics such as accuracy, precision, recall, and AUC-ROC curve. These metrics help determine how well the model predicts whether a patient will experience complications.

4.3 Predicting the Prognosis of Diabetic Patients:

- Once trained, the model predicts the risk of complications for diabetic patients. For example, a patient with high blood sugar levels, high BMI, and long-standing diabetes may be predicted to have a higher probability of developing heart disease or neuropathy.
- These predictions help healthcare providers determine which patients require immediate intervention or more aggressive treatment plans. For instance, a patient at high risk of kidney complications may need to be monitored closely, undergo **additional tests, or receive medication to prevent further damage.**

5. AI's Impact on Medical Prognosis:

- Personalized Treatment Plans: AI models like logistic regression can provide personalized prognosis by factoring in patient-specific details, allowing clinicians to tailor treatment plans according to individual risk.
- Early Intervention: Predicting complications early leads to timely interventions, which can prevent the progression of diseases, reduce hospitalizations, and improve patient outcomes.
- Resource Optimization: Hospitals can prioritize resources like specialist appointments and advanced diagnostic tests for patients at higher risk based on the prognosis provided by AI.

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

X = pd.read_csv('/content/X_data.csv',index_col=0)
y_df = pd.read_csv('/content/y_data.csv',index_col=0) y = y_df['y']
df = pd.concat([X, y_df], axis=1)
df.head(20)
```

```
{"summary":"{\n \"name\": \"df\",\n \"rows\": 6000,\n \"fields\":  [\n {\n \"column\": \"Age\",\n
\"properties\": {\n  \"dtype\": \"number\",\n \"std\": 8.564392230753992,\n  \"min\":
35.16476092544677,\n \"max\": 103.27949724529527,\n  \"num_unique_values\": 6000,\n
\"samples\": [\n  47.560988544668255,\n 56.66079133908024,\n
57.10653704530141\n ],\n \"semantic_type\": \"\",\n  \"description\": \"\"\n }\n },\n {\n
\"column\": \"Systolic_BP\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\":
10.669267051431795,\n \"min\":  69.67542852731282,\n \"max\":
151.6996602188728,\n  \"num_unique_values\": 6000,\n \"samples\": [\n
101.6217705626216,\n 118.13456978269728,\n
102.7636120345311б\n ],\n \"semantic_type\": \"\",\n  \"description\": \"\"\n }\n },\n {\n
\"column\": \"Diastolic_BP\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\":
9.648199738886857,\n \"min\":  62.80710490744539,\n \"max\":
133.4563821720989,\n  \"num_unique_values\": 6000,\n \"samples\": [\n
86.38696731883141,\n 112.21625225363744,\n
83.166656524877З\n ],\n \"semantic_type\": \"\",\n  \"description\": \"\"\n }\n },\n {\n
\"column\": \"Cholesterol\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\":
```

10.433914618686725,\n \"min\": 69.96745265197228,\n \"max\":
148.2335443421762,\n \"num_unique_values\": 6000,\n \"samples\": [\n
106.60938579119816,\n 114.815663874924,\n
86.88775392690107\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n
\"column\": \"y\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\":
0.49983136012535345,\n \"min\": 0.0,\n \"max\": 1.0,\n \"num_unique_values\": 2,\n
\"samples\": [\n 0.0,\n 1.0\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n }\n]\
n}","type":"dataframe","variable_name":"df"}

```
X.head()
```

{"summary":"{\n \"name\": \"X\",\n \"rows\": 6000,\n \"fields\": [\ n {\n \"column\": \"Age\",\n
\"properties\": {\n \"dtype\": \"number\",\n \"std\": 8.564392230753992,\n \"min\":
35.16476092544677,\n \"max\": 103.27949724529527,\n \"num_unique_values\": 6000,\n
\"samples\": [\n 47.560988544668255,\n 56.66079133908024,\n
57.10653704530141\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n
\"column\": \"Systolic_BP\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\":
10.669267051431795,\n \"min\": 69.67542852731282,\n \"max\":
151.6996602188728,\n \"num_unique_values\": 6000,\n \"samples\": [\n
101.6217705626216,\n 118.13456978269728,\n
102.76361203453116\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n
\"column\": \"Diastolic_BP\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\":
9.648199738886857,\n \"min\": 62.80710490744539,\n \"max\":
133.4563821720989,\n \"num_unique_values\": 6000,\n \"samples\": [\n
86.38696731883141,\n 112.21625225363744,\n
83.1666565248773\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n
\"column\": \"Cholesterol\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\":
10.433914618686725,\n \"min\": 69.96745265197228,\n \"max\":
148.2335443421762,\n \"num_unique_values\": 6000,\n \"samples\": [\n
106.60938579119816,\n 114.815663874924,\n
86.88775392690107\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n }\n]\
n}","type":"dataframe","variable_name":"X"}

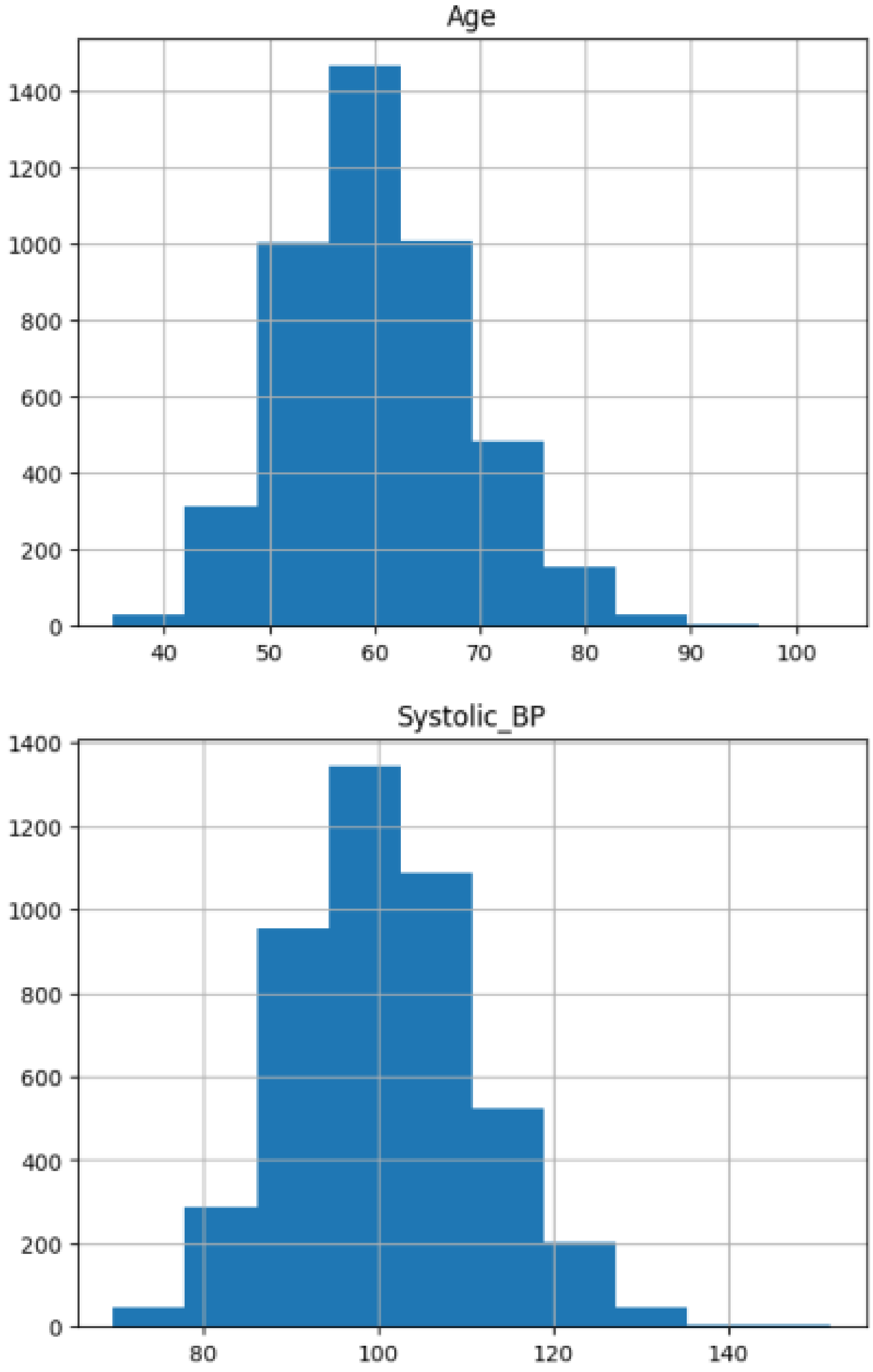
```
y.head()
```

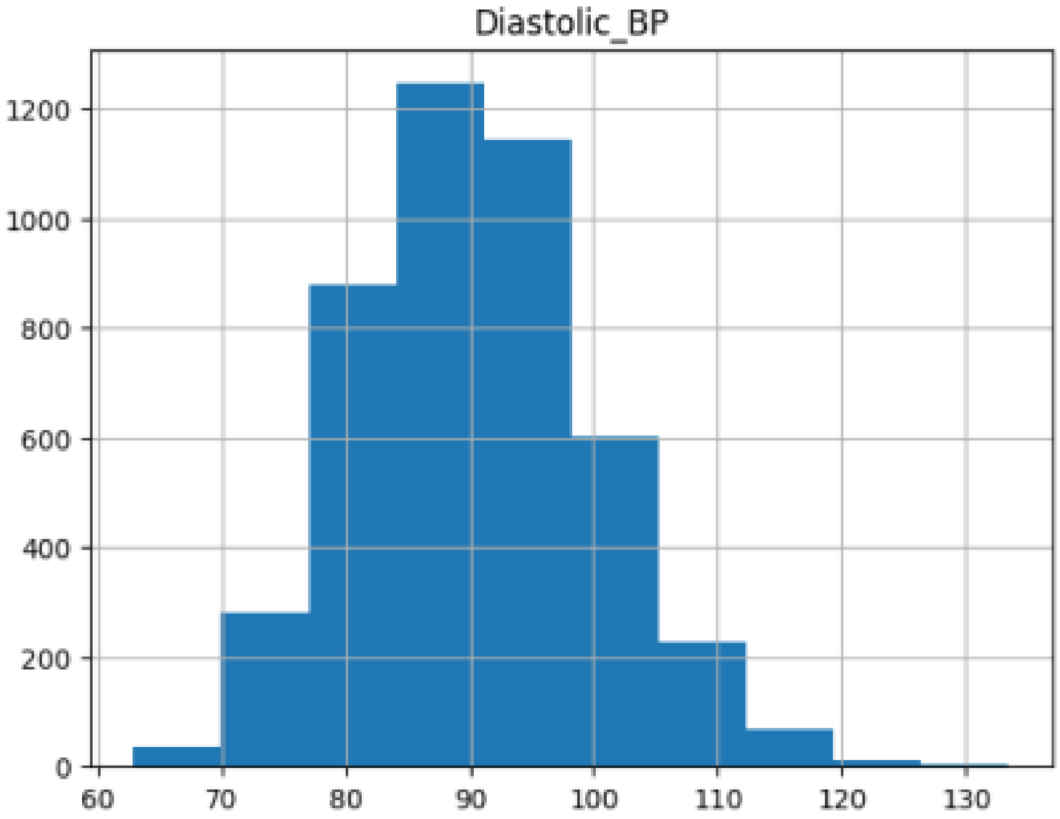
```
0 1.0
1 1.0
2 1.0
3 1.0
4 1.0
Name: y, dtype: float64
```

```
from sklearn.model_selection import train_test_split
```

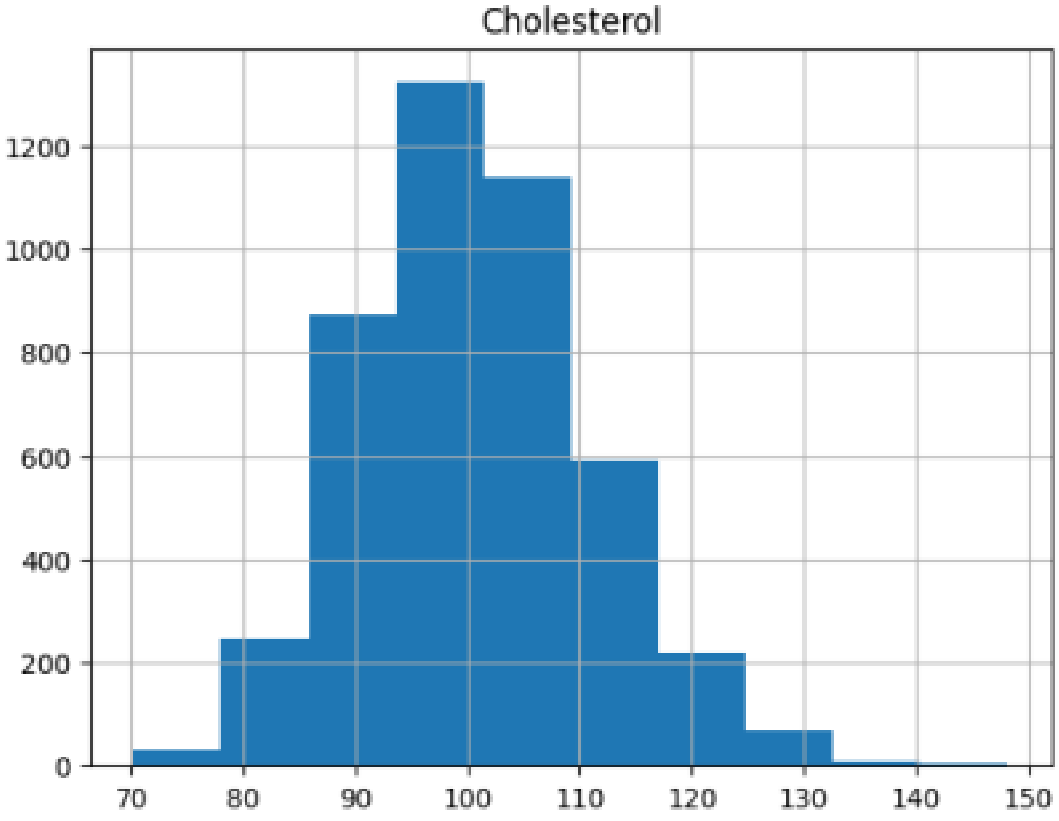
```
X_train_raw, X_test_raw, y_train, y_test = train_test_split(X, y,  train_size=0.75,
random_state=42)
```

```
for col in X.columns:
 X_train_raw.loc[:,col].hist()
 plt.title(col)
 plt.show()
```


Age
1400
1200
1000
800
600
400
200
0
40
50
60
70
80
90
100
Systolic_BP
1400
1200
1000
800
600
400
200
0
80
100
120
140

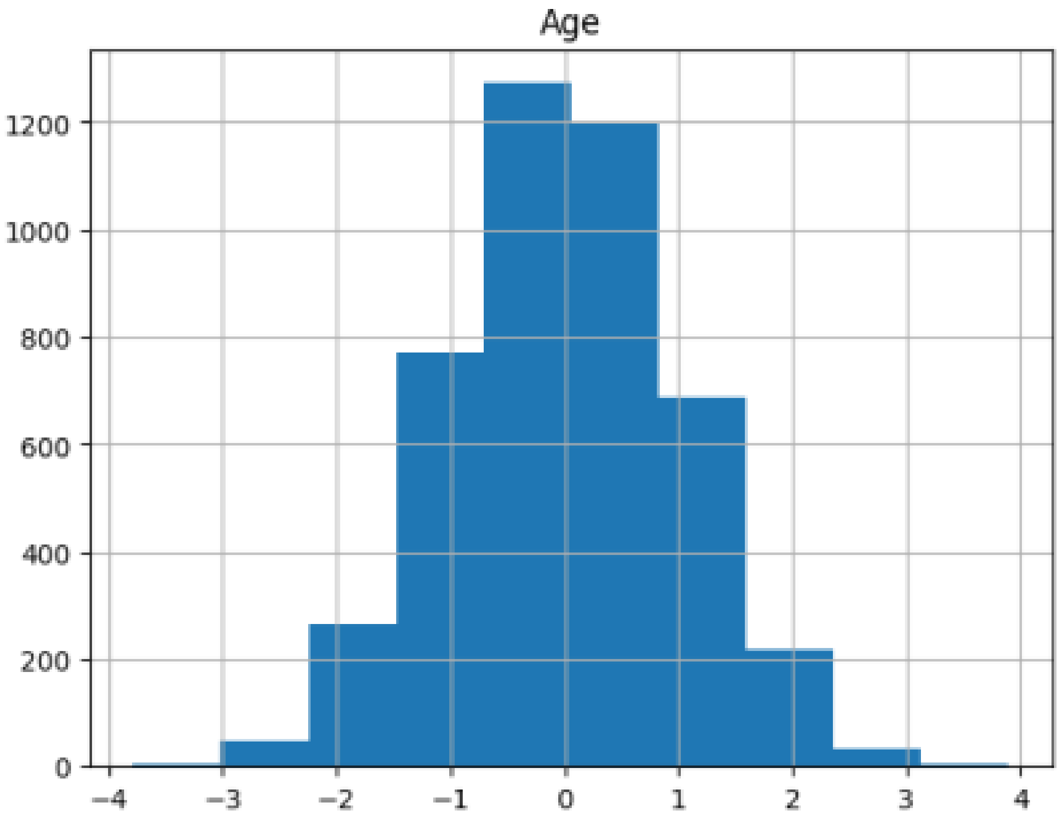


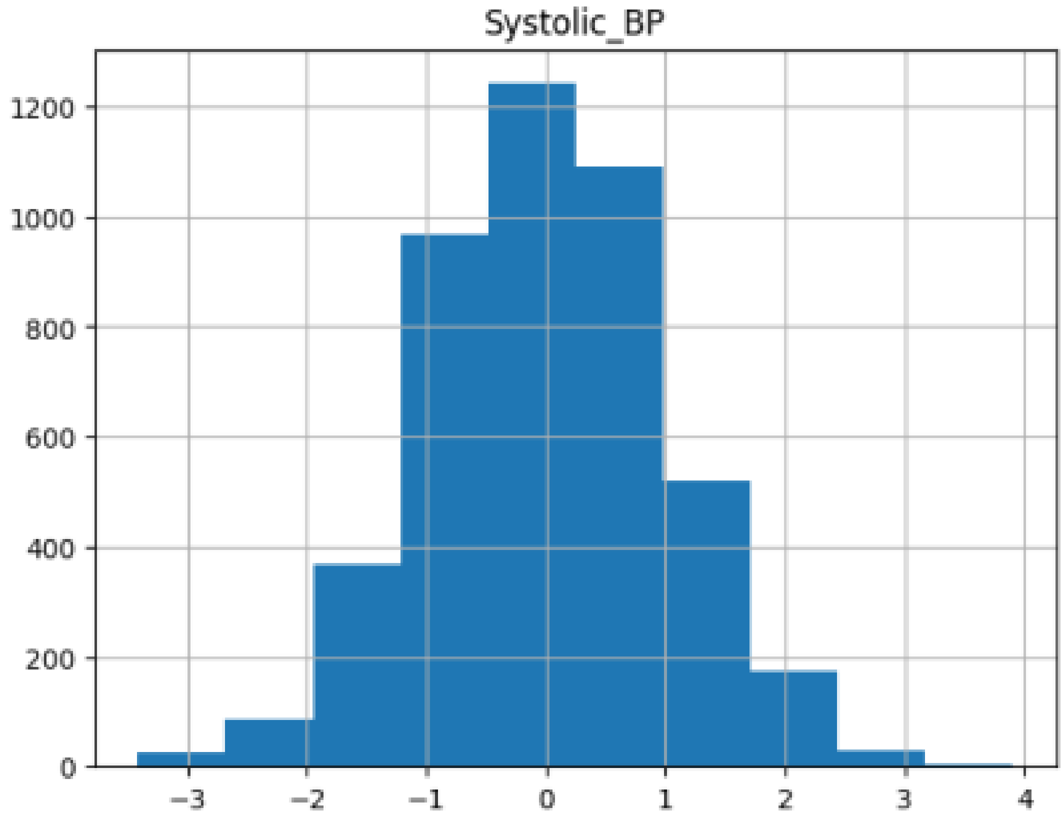
Diastolic_BP



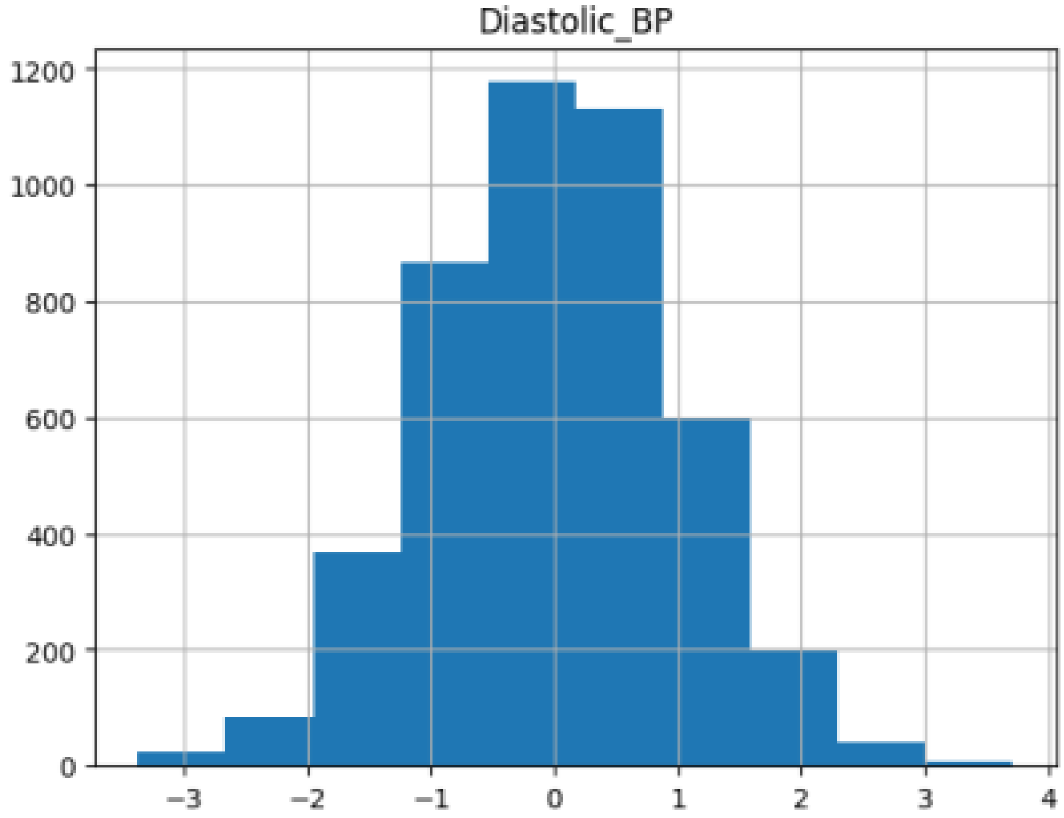
Cholesterol

```
def make_standard_normal(df_train, df_test):
 df_train_unskewed = np.log(df_train)
 df_test_unskewed = np.log(df_test)
 mean = df_train_unskewed.mean(axis = 0)
 stdev = df_train_unskewed.std(axis = 0)
 df_train_standardized = (df_train_unskewed - mean)/stdev  mean_ =
df_test_unskewed.mean()
 stdev_ = df_test_unskewed.std()
 df_test_standardized = (df_test_unskewed - mean)/stdev  return
df_train_standardized, df_test_standardized

X_train, X_test = make_standard_normal(X_train_raw, X_test_raw)

 for col in X_train.columns:
 X_train[col].hist()
 plt.title(col)
 plt.show()
```

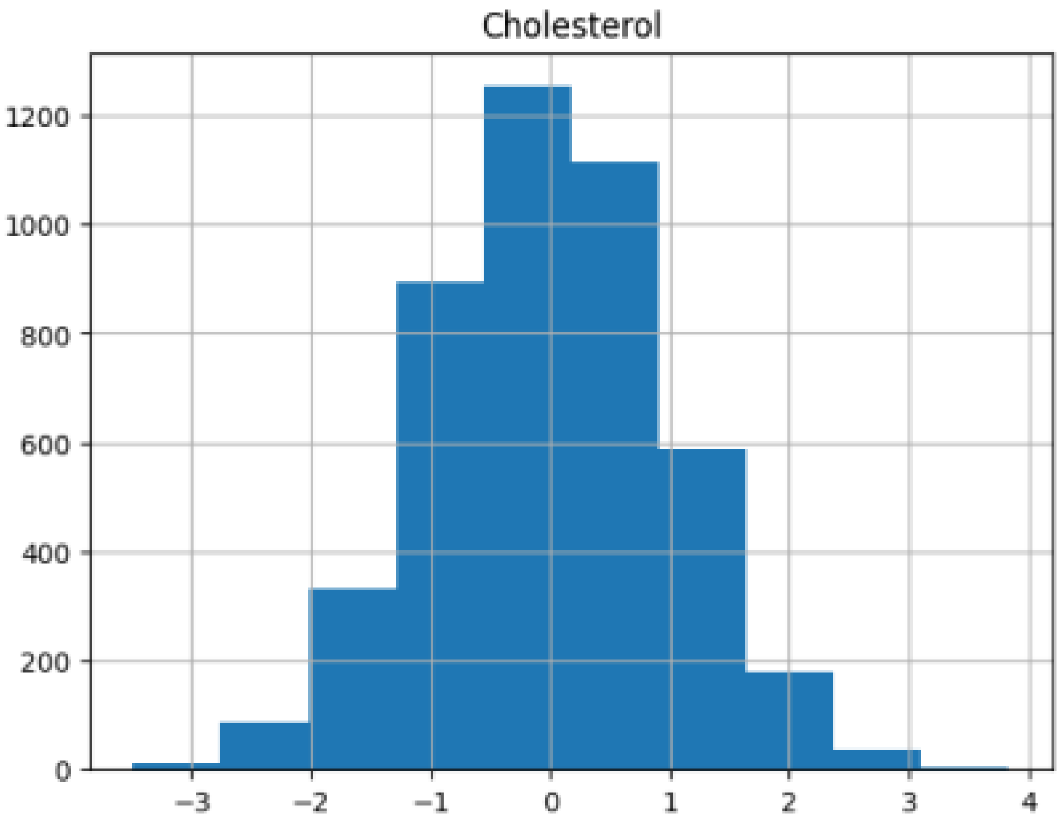




Systolic_BP



Diastolic_BP



```
def lr_model(X_train, y_train):
 from sklearn.linear_model import LogisticRegression  model =
LogisticRegression()
 model.fit(X_train,y_train)
 return model

model_X = lr_model(X_train, y_train)
model_X
from sklearn.metrics import accuracy_score
y_pred_proba = model_X.predict_proba(X_test)[:, 1]
y_pred = (y_pred_proba >= 0.5).astype(int)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy on test set: {accuracy:.4f}")

Accuracy on test set: 0.7447
```

Evaluate the Model Using the C-index

Now that we have a model, we need to evaluate it. We'll do this using the

c-index. · The c-index measures the discriminatory power of a risk score.

- Intuitively, a higher c-index indicates that the model's prediction is in agreement with the actual outcomes of a pair of patients.
- The formula for the c-index is

$$\mathrm{cindex} = \frac{\mathrm{concordant} + 0.5 \times \mathrm{ties}}{\mathrm{permissible}}$$

- A permissible pair is a pair of patients who have different outcomes.
- A concordant pair is a permissible pair in which the patient with the higher risk score also has the worse outcome.
- A tie is a permissible pair where the patients have the same risk score.
- We will implement the `cindex` function to compute c-index.
- `y_true` is the array of actual patient outcomes, 0 if the patient does not eventually get the disease, and 1 if the patient eventually gets the disease.
- `scores` is the risk score of each patient. These provide relative measures of risk, so they can be any real numbers. By convention, they are always non-negative.
- Here is an example of input data and how to interpret it:

```
y_true = [0,1]
scores = [0.45, 1.25]
```

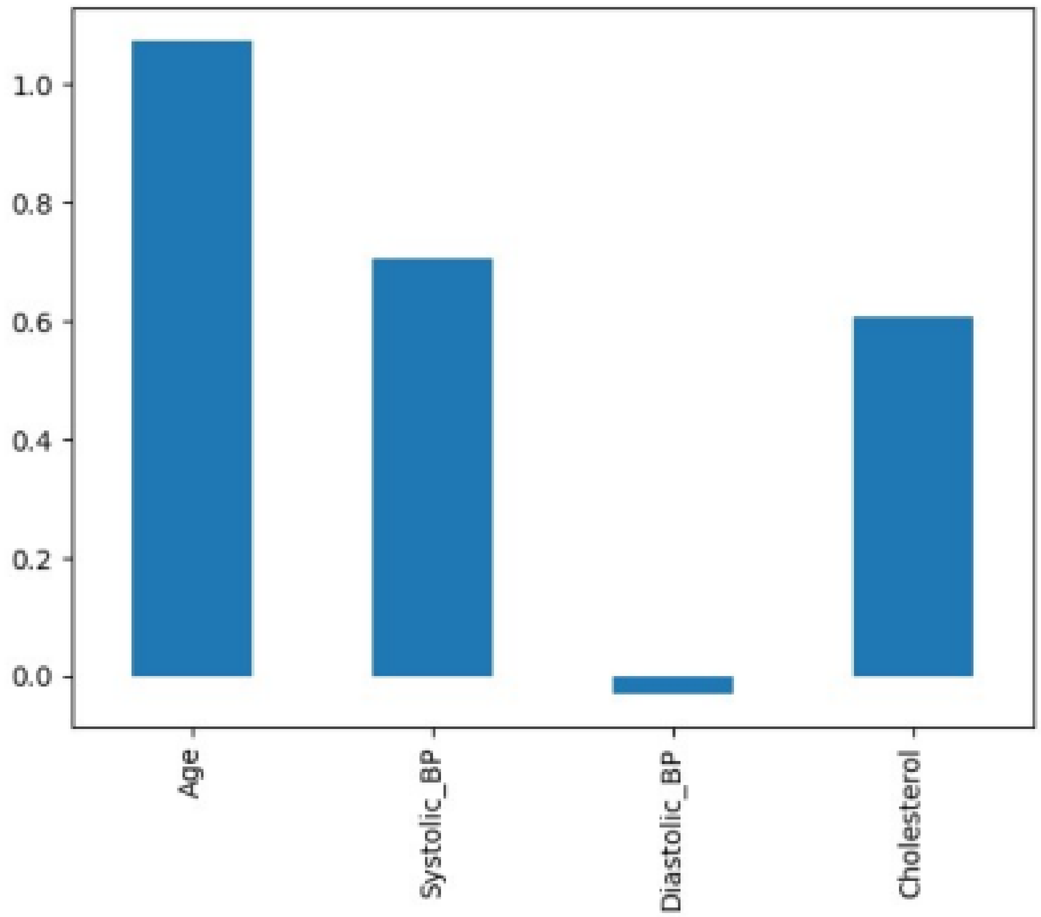
* There are two patients. Index 0 of each array is associated with patient 0. Index 1 is associated with patient 1.
* Patient 0 does not have the disease in the future (`y_true` is 0), and based on past information, has a risk score of 0.45. * Patient 1 has the disease at some point in the future (`y_true` is 1), and based on past information, has a risk score of 1.25.

```
def cindex(y_true, scores):
 n = len(y_true)
 assert len(scores) == n
 concordant = 0
 permissible = 0
 ties = 0
 for i in range(n):
 for j in range(i+1, n):
 if y_true[i]!= y_true[j]:
 permissible += 1
 if scores[i]== scores[j]:
 ties +=1
 continue
 if y_true[i] == 0 and y_true[j] == 1:
 if scores[i] < scores[j]:
 concordant +=1
 if y_true[i] ==1 and y_true[j] == 0:
 if scores[i] > scores[j]:
 concordant +=1
 c_index = (concordant + 0.5* ties)/permissible
 return c_index
```

```
scores = model_X.predict_proba(X_test)[:, 1]
print(scores)
c_index_X_test = cindex(y_test.values, scores)
print(f"c-index on test set is {c_index_X_test:.4f}")
```

```
 [0.23463557 0.82161904 0.28652138 ... 0.33877758 0.5439931
0.06001348]
c-index on test set is 0.8271
```

```
coeffs = pd.DataFrame(data = model_X.coef_, columns = X_train.columns)
coeffs.T.plot.bar(legend=None);
```



```
def add_interactions(X):
 features = X.columns
 m = len(features)
 X_int = X.copy(deep=True)
 for i in range(m):
 feature_i_name = features[i]
 feature_i_data = X_int[feature_i_name]
 for j in range(i+1, m):
 feature_j_name = features[j]
```

```
 feature_j_data = X_int[feature_j_name]
  feature_i_j_name  =  f"{feature_i_name}_x_{feature_j_name}"   X_int[feature_i_j_name]  =
feature_i_data * feature_j_data  return X_int

print("Original Data")
print(X_train.loc[:, ['Age', 'Systolic_BP']].head())
print("Data w/ Interactions")
                                        print(add_interactions(X_train.loc[:, ['Age', 'Systolic_BP']].head()))
```

```
Original Data
 Age Systolic_BP
```

```
2847 0.734743 0.242422
4336 0.252259 -1.197857
3386 0.073411 -1.270867
1721 -1.368268 -1.130665
2404 0.443414 -0.767159
Data w/ Interactions
 Age Systolic_BP Age_x_Systolic_BP
2847 0.734743 0.242422 0.178118
4336 0.252259 -1.197857 -0.302170
3386 0.073411 -1.270867 -0.093296
1721 -1.368268 -1.130665 1.547052
2404 0.443414 -0.767159 -0.340169
```

```
X_train_int = add_interactions(X_train)
X_test_int = add_interactions(X_test)
```

```
model_X_int = lr_model(X_train_int, y_train)
```

```
scores_X = model_X.predict_proba(X_test)[:, 1]
c_index_X_int_test = cindex(y_test.values, scores_X)
```

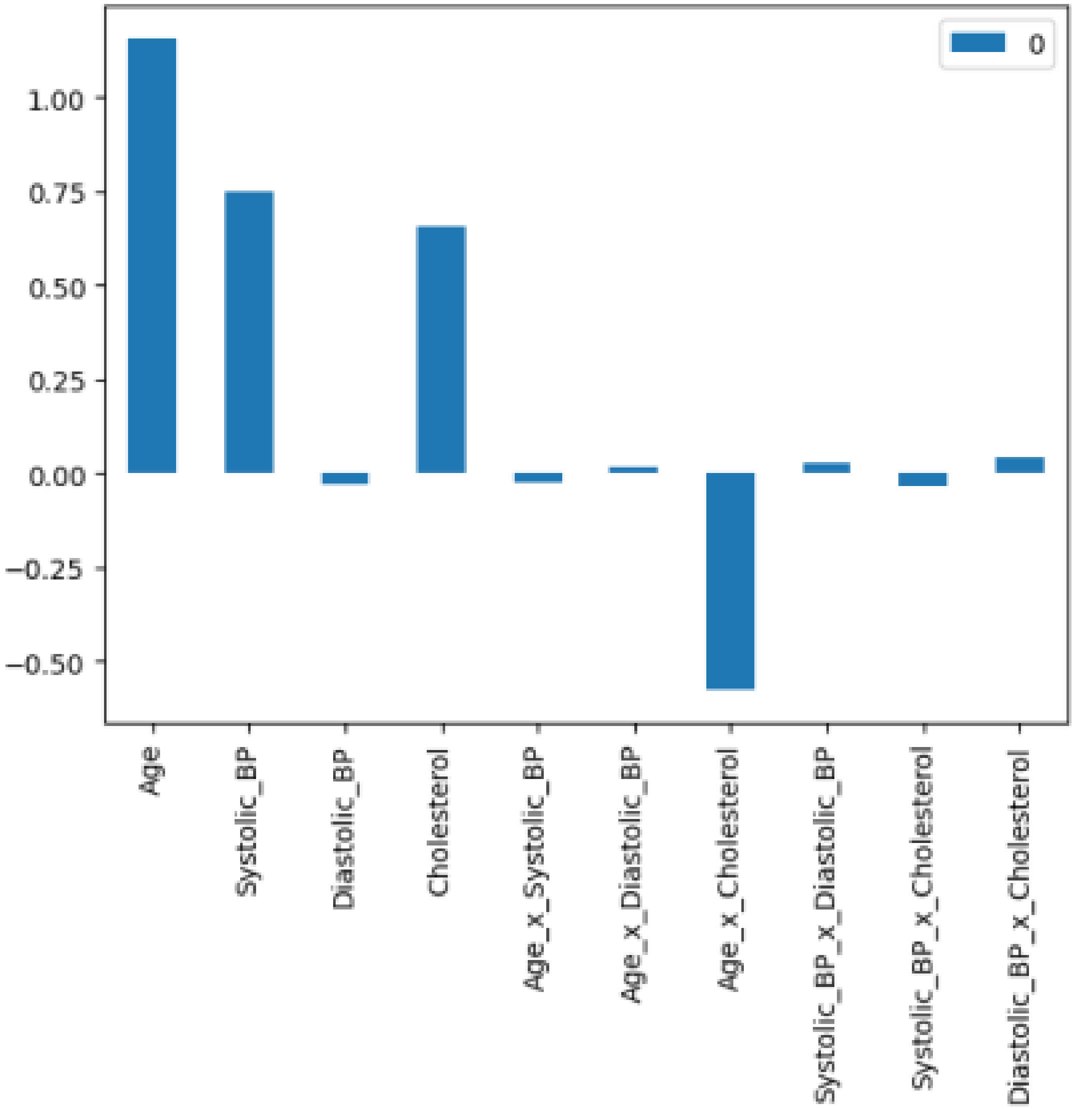
```
scores_X_int = model_X_int.predict_proba(X_test_int)[:, 1] c_index_X_int_test
= cindex(y_test.values, scores_X_int)
```

```
print(f"c-index on test set without interactions is
{c_index_X_test:.4f}")
print(f"c-index on test set with interactions is
{c_index_X_int_test:.4f}")
from sklearn.metrics import accuracy_score
```

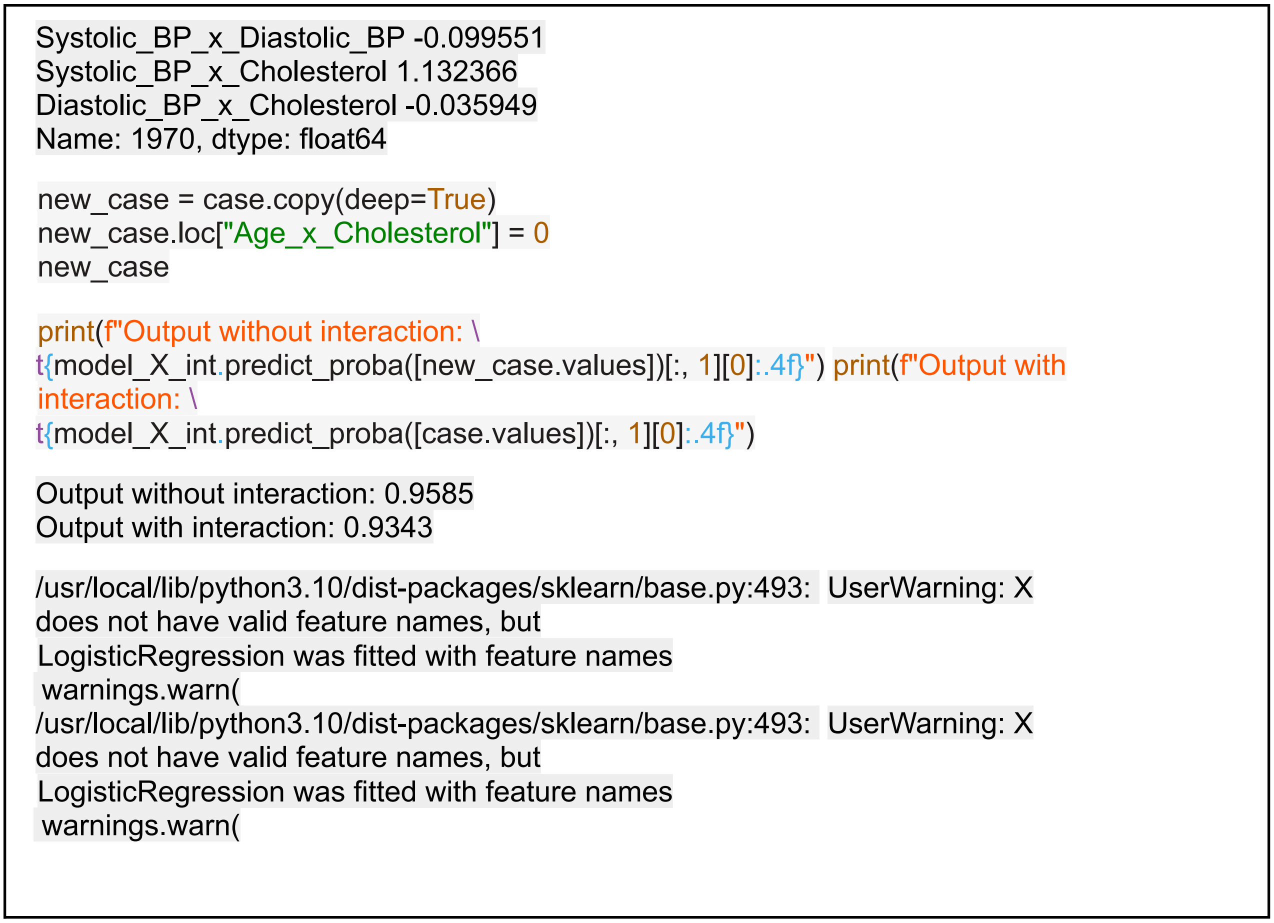
```
y_pred_proba = model_X_int.predict_proba(X_test_int)[:, 1] y_pred =
(y_pred_proba >= 0.5).astype(int)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy on test set: {accuracy:.4f}")
```

```
c-index on test set without interactions is 0.8271
c-index on test set with interactions is 0.8333
Accuracy on test set: 0.7540
```

```
int_coeffs = pd.DataFrame(data = model_X_int.coef_, columns =
X_train_int.columns)
int_coeffs.T.plot.bar();
```



```
index = index = 3488
case = X_train_int.iloc[index, :]
print(case)

Age 1.318437
Systolic_BP 1.770809
Diastolic_BP -0.056218
Cholesterol 0.639463
Age_x_Systolic_BP 2.334699
Age_x_Diastolic_BP -0.074119
Age_x_Cholesterol 0.843091
```



```
Systolic_BP_x_Diastolic_BP -0.099551
Systolic_BP_x_Cholesterol 1.132366
Diastolic_BP_x_Cholesterol -0.035949
Name: 1970, dtype: float64

new_case = case.copy(deep=True)
new_case.loc["Age_x_Cholesterol"] = 0
new_case

print(f"Output without interaction: \
t{model_X_int.predict_proba([new_case.values])[:, 1][0]:.4f}") print(f"Output with
interaction: \
t{model_X_int.predict_proba([case.values])[:, 1][0]:.4f}")

Output without interaction: 0.9585
Output with interaction: 0.9343

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493:  UserWarning: X
does not have valid feature names, but
LogisticRegression was fitted with feature names
 warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493:  UserWarning: X
does not have valid feature names, but
LogisticRegression was fitted with feature names
 warnings.warn(
```

Conclusion :

In this experiment, we utilized a Logistic Regression model to predict the prognosis of diabetic patients by analyzing their medical records and physiological measurements. This AI-driven prognosis system enables healthcare professionals to make data-driven decisions, allowing for early intervention, personalized treatment plans, and overall better patient management. The integration of AI in medical prognosis is an important step towards more efficient, personalized, and proactive healthcare, improving patient outcomes and reducing the burden of chronic diseases like diabetes.