```
import pandas as pd
import numpy as np
from sklearn.datasets import make_classification

# Generating synthetic data
n_samples = 1000

# Generating features
age = np.random.randint(18, 80, size=n_samples)
gender = np.random.choice(['Male', 'Female'], size=n_samples)
bmi = np.random.uniform(18, 40, size=n_samples)
blood_pressure = np.random.randint(90, 180, size=n_samples)
cholesterol = np.random.randint(120, 300, size=n_samples)
family_history = np.random.choice([0, 1], size=n_samples)
exercise_hours = np.random.randint(0, 24, size=n_samples)
smoking_status = np.random.choice(['Never Smoked', 'Former Smoker', 'Current Smo
alcohol_consumption = np.random.choice(['None', 'Moderate', 'Heavy'], size=n_sam
stress_level = np.random.randint(0, 11, size=n_samples)  # Assume stress level r
sleep_duration = np.random.randint(4, 12, size=n_samples)  # Assume sleep durati
fast_food_intake = np.random.randint(0, 4, size=n_samples)  # Assume frequency o

# Generating target variable (disease presence)
# For simplicity, let's generate a binary target where 1 indicates presence and
# You can replace this with your own logic for generating target variable based
X, y = make_classification(n_samples=n_samples, n_features=11, n_classes=2, rand

# Creating a DataFrame
data = pd.DataFrame({
    'age': age,
    'gender': gender,
    'bmi': bmi,
    'blood_pressure': blood_pressure,
    'cholesterol': cholesterol,
    'family_history': family_history,
    'exercise_hours': exercise_hours,
    'smoking_status': smoking_status,
    'alcohol_consumption': alcohol_consumption,
    'stress_level': stress_level,
    'sleep_duration': sleep_duration,
    'fast_food_intake': fast_food_intake,
    'disease': y
})

# Encoding categorical variables
data = pd.get_dummies(data, columns=['gender', 'smoking_status', 'alcohol_consum

# Saving the dataset to a CSV file
```

```
data.to_csv('health_data_extended.csv', index=False)
print(data.head())
```

```
   age        bmi  blood_pressure  cholesterol  family_history  \
0   50  35.545929             167          251               0
1   21  18.052021             139          191               1
2   27  25.326220             111          266               0
3   20  24.180481             165          189               1
4   29  26.727182              90          248               0

   exercise_hours  stress_level  sleep_duration  fast_food_intake  disease
0              23             4               6                 3        0
1              11             6               9                 3        0
2              15             5              11                 3        1
3              21             6              11                 0        0
4               8             0               5                 3        1

   gender_Female  gender_Male  smoking_status_Current Smoker  \
0              0            1                              0
1              0            1                              0
2              1            0                              0
3              1            0                              0
4              1            0                              1

   smoking_status_Former Smoker  smoking_status_Never Smoked  \
0                             0                            1
1                             1                            0
2                             0                            1
3                             1                            0
4                             0                            0

   alcohol_consumption_Heavy  alcohol_consumption_Moderate  \
0                          0                             0
1                          0                             0
2                          1                             0
3                          0                             0
4                          1                             0

   alcohol_consumption_None
0                         1
1                         1
2                         0
3                         1
4                         0
```

```
import pandas as pd

# Load the dataset
data = pd.read_csv('health_data_extended.csv')
```

```
# Perform data cleaning and preprocessing
# Handle missing values
data.dropna(inplace=True)
print(data)

# Handle outliers (if necessary)

# Normalize or standardize features
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaled_features = scaler.fit_transform(data.drop(columns=['disease']))
data[data.columns[:-1]] = scaled_features
```

```
      age        bmi  blood_pressure  cholesterol  family_history  \
0      48  38.489044             145          262               1
1      22  29.519678              91          151               1
2      78  22.508102             145          165               1
3      21  27.198484             149          289               1
4      61  33.442012             156          220               1
..    ...        ...             ...          ...             ...
995    42  19.387862             128          286               1
996    24  37.187901             162          297               1
997    54  28.479496             101          166               0
998    37  26.498280             125          211               0
999    68  23.527290             167          207               1

      exercise_hours  stress_level  sleep_duration  fast_food_intake  diseas
0                 10             1               8                 0
1                  5             3               9                 3
2                 10             9               8                 2
3                  7            10               5                 1
4                  0             6               7                 0
..               ...           ...             ...               ...      ..
995                7             7               7                 2
996               10             6               6                 3
997                7             1               6                 2
998               18             6               5                 1
999                6             3               7                 2

      gender_Female  gender_Male  smoking_status_Current Smoker  \
0                 1            0                              0
1                 1            0                              0
2                 1            0                              0
3                 1            0                              0
4                 1            0                              0
..              ...          ...                            ...
995               1            0                              0
996               0            1                              0
997               1            0                              0
```

```
998              0            1                                1
999              1            0                                1

     smoking_status_Former Smoker  smoking_status_Never Smoked  \
0                               0                             1
1                               1                             0
2                               1                             0
3                               1                             0
4                               0                             1
..                            ...                           ...
995                             0                             1
996                             0                             1
997                             1                             0
998                             0                             0
999                             0                             0

     alcohol_consumption_Heavy  alcohol_consumption_Moderate  \
0                            0                             0
1                            0                             1
2                            0                             0
3                            0                             0
4                            1                             0
..                         ...                           ...
```

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.model_selection import train_test_split, cross_val_score, GridSear
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_s
```

```
data = pd.read_csv('health_data_extended.csv')

# Perform data cleaning and preprocessing
# Handle missing values
data.dropna(inplace=True)

# Normalize or standardize features
scaler = MinMaxScaler()  # Using MinMaxScaler to ensure all features are non-ne
scaled_features = scaler.fit_transform(data.drop(columns=['disease']))
data[data.columns[:-1]] = scaled_features

# Feature selection using chi-squared test
X = data.drop(columns=['disease'])
y = data['disease']
selector = SelectKBest(score_func=chi2, k=5)
selected_features = selector.fit(X, y)
selected_features_indices = selected_features.get_support(indices=True)
selected_features_names = X.columns[selected_features_indices]
X_selected = data[selected_features_names]
```

```
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.
```

```
logistic_regression = LogisticRegression()
decision_tree = DecisionTreeClassifier()
random_forest = RandomForestClassifier()
svm = SVC()
```

```
# Fit the models
logistic_regression.fit(X_train, y_train)
decision_tree.fit(X_train, y_train)
random_forest.fit(X_train, y_train)
svm.fit(X_train, y_train)
```

▾ SVC

SVC()

```
# Predictions
lr_predictions = logistic_regression.predict(X_test)
dt_predictions = decision_tree.predict(X_test)
rf_predictions = random_forest.predict(X_test)
svm_predictions = svm.predict(X_test)

# Evaluate model performance
print("Logistic Regression:")
print("Accuracy:", accuracy_score(y_test, lr_predictions))
print("Precision:", precision_score(y_test, lr_predictions))
print("Recall:", recall_score(y_test, lr_predictions))
print("F1 Score:", f1_score(y_test, lr_predictions))
# Evaluate Decision Tree model
print("\nDecision Tree:")
print("Accuracy:", accuracy_score(y_test, dt_predictions))
print("Precision:", precision_score(y_test, dt_predictions))
print("Recall:", recall_score(y_test, dt_predictions))
print("F1 Score:", f1_score(y_test, dt_predictions))

# Evaluate Random Forest model
print("\nRandom Forest:")
print("Accuracy:", accuracy_score(y_test, rf_predictions))
print("Precision:", precision_score(y_test, rf_predictions))
print("Recall:", recall_score(y_test, rf_predictions))
print("F1 Score:", f1_score(y_test, rf_predictions))

# Evaluate SVM model
print("\nSVM:")
print("Accuracy:", accuracy_score(y_test, svm_predictions))
print("Precision:", precision_score(y_test, svm_predictions))
print("Recall:", recall_score(y_test, svm_predictions))
print("F1 Score:", f1_score(y_test, svm_predictions))
```

```
Logistic Regression:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0

Decision Tree:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0

Random Forest:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0

SVM:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

```
cv_scores = cross_val_score(logistic_regression, X_selected, y, cv=5)
print("Cross-Validation Scores for Logistic Regression:", cv_scores)
print("Mean CV Score for Logistic Regression:", cv_scores.mean())
```

```
Cross-Validation Scores for Logistic Regression: [1. 1. 1. 1. 1.]
Mean CV Score for Logistic Regression: 1.0
```

```
from sklearn.model_selection import RandomizedSearchCV

param_distributions = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

random_search = RandomizedSearchCV(random_forest, param_distributions, n_iter=1
random_search.fit(X_train, y_train)

# Best parameters
print("Best Parameters:", random_search.best_params_)
```

```
Best Parameters: {'n_estimators': 100, 'min_samples_split': 5, 'min_samples
```

```
# Predictions using the best model
best_model = random_search.best_estimator_
best_model_predictions = best_model.predict(X_test)

# Evaluate the best model
print("Best Model Performance:")
print("Accuracy:", accuracy_score(y_test, best_model_predictions))
print("Precision:", precision_score(y_test, best_model_predictions))
print("Recall:", recall_score(y_test, best_model_predictions))
print("F1 Score:", f1_score(y_test, best_model_predictions))
```

```
Best Model Performance:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

User Interface:(optional) A user-friendly interface that allows users to input their health-related data and receive predictions about the likelihood of having a particular disease.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.model_selection import train_test_split, cross_val_score, GridSearc
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc
```

```
data = pd.read_csv('health_data_extended.csv')

# Perform data cleaning and preprocessing
# Handle missing values
data.dropna(inplace=True)

# Normalize or standardize features
scaler = MinMaxScaler()  # Using MinMaxScaler to ensure all features are non-neg
scaled_features = scaler.fit_transform(data.drop(columns=['disease']))
data[data.columns[:-1]] = scaled_features

# Feature selection using chi-squared test
X = data.drop(columns=['disease'])
y = data['disease']
selector = SelectKBest(score_func=chi2, k=5)
```

```
selector = SelectKBest(score_func=chi2, k=5)
selected_features = selector.fit(X, y)
selected_features_indices = selected_features.get_support(indices=True)
selected_features_names = X.columns[selected_features_indices]
X_selected = data[selected_features_names]



# Train a Random Forest model
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.2
random_forest = RandomForestClassifier()
random_forest.fit(X_train, y_train)

# Define a function to make predictions
def predict_disease(features):
    scaled_features = scaler.transform([features])
    selected_features = scaled_features[:, selected_features_indices]
    prediction = random_forest.predict(selected_features)[0]
    return prediction

# User Interface
print("Welcome to the Disease Prediction System!")
print("Please enter your health-related data:")

age = int(input("Enter your age: "))
gender = input("Enter your gender (Male/Female): ")
bmi = float(input("Enter your BMI: "))
blood_pressure = int(input("Enter your blood pressure: "))
cholesterol = int(input("Enter your cholesterol level: "))
family_history = int(input("Do you have a family history of the disease? (0 for
exercise_hours = int(input("Enter your weekly exercise hours: "))
smoking_status = input("Enter your smoking status (Never Smoked/Former Smoker/Cu
alcohol_consumption = input("Enter your alcohol consumption level (None/Moderate
stress_level = int(input("Enter your stress level (0-10): "))
sleep_duration = int(input("Enter your average sleep duration (hours): "))
fast_food_intake = int(input("Enter your weekly frequency of fast food intake: "

features = [age, bmi, blood_pressure, cholesterol, family_history, exercise_hour

# Convert categorical inputs to one-hot encoded format
gender_male = 1 if gender.lower() == 'male' else 0
gender_female = 1 if gender.lower() == 'female' else 0

smoking_status_never = 1 if smoking_status.lower() == 'never smoked' else 0
smoking_status_former = 1 if smoking_status.lower() == 'former smoker' else 0
smoking_status_current = 1 if smoking_status.lower() == 'current smoker' else 0

alcohol_none = 1 if alcohol_consumption.lower() == 'none' else 0
alcohol_moderate = 1 if alcohol_consumption.lower() == 'moderate' else 0
```

```
alcohol_moderate = 1 if alcohol_consumption.lower() == 'moderate' else 0
alcohol_heavy = 1 if alcohol_consumption.lower() == 'heavy' else 0

features += [gender_male, gender_female, smoking_status_never, smoking_status_fo

# Make prediction
prediction = predict_disease(features)

# Output prediction
if prediction == 1:
    print("\nBased on the provided data, you are predicted to have the disease."
else:
    print("\nBased on the provided data, you are predicted to not have the disea
```

```
Welcome to the Disease Prediction System!
Please enter your health-related data:
Enter your age: 21
Enter your gender (Male/Female): Male
Enter your BMI: 24.2
Enter your blood pressure: 250
Enter your cholesterol level: 250
Do you have a family history of the disease? (0 for No, 1 for Yes): 1
Enter your weekly exercise hours: 2
Enter your smoking status (Never Smoked/Former Smoker/Current Smoker): Curr
Enter your alcohol consumption level (None/Moderate/Heavy): Heavy
Enter your stress level (0-10): 10
Enter your average sleep duration (hours): 4
Enter your weekly frequency of fast food intake: 6

Based on the provided data, you are predicted to not have the disease.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X
  warnings.warn(
```