In [ ]:

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

from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest, f_classif, chi2
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [ ]:

```
df = pd.read_csv("diabetes_dataset.csv")
```

Data Exploration

In [ ]:

```
df.shape
```

```
(100000, 31)
```

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 31 columns):
 #   Column                              Non-Null Count   Dtype  
---  ------                              --------------   -----  
 0   age                                 100000 non-null  int64  
 1   gender                              100000 non-null  object 
 2   ethnicity                           100000 non-null  object 
 3   education_level                     100000 non-null  object 
 4   income_level                        100000 non-null  object 
 5   employment_status                   100000 non-null  object 
 6   smoking_status                      100000 non-null  object 
 7   alcohol_consumption_per_week        100000 non-null  int64  
 8   physical_activity_minutes_per_week  100000 non-null  int64  
 9   diet_score                          100000 non-null  float64
 10  sleep_hours_per_day                 100000 non-null  float64
 11  screen_time_hours_per_day           100000 non-null  float64
 12  family_history_diabetes             100000 non-null  int64  
 13  hypertension_history                100000 non-null  int64  
 14  cardiovascular_history              100000 non-null  int64  
 15  bmi                                 100000 non-null  float64
 16  waist_to_hip_ratio                  100000 non-null  float64
 17  systolic_bp                         100000 non-null  int64  
 18  diastolic_bp                        100000 non-null  int64  
 19  heart_rate                          100000 non-null  int64  
 20  cholesterol_total                   100000 non-null  int64  
 21  hdl_cholesterol                     100000 non-null  int64  
 22  ldl_cholesterol                     100000 non-null  int64  
 23  triglycerides                       100000 non-null  int64  
 24  glucose_fasting                     100000 non-null  int64  
 25  glucose_postprandial                100000 non-null  int64  
 26  insulin_level                       100000 non-null  float64
 27  hba1c                               100000 non-null  float64
 28  diabetes_risk_score                 100000 non-null  float64
 29  diabetes_stage                      100000 non-null  object 
 30  diagnosed_diabetes                  100000 non-null  int64  
dtypes: float64(8), int64(16), object(7)
memory usage: 23.7+ MB
```

In [ ]:

```
df.describe()
```

	age	alcohol_consumption_per_week	physical_activity_minutes_per_week	diet_score	sleep_hours_per_day	screen_time_hours_per_day	family_history_diabetes	hypertension_history	cardiova
count	100000.00000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.0
mean	50.12041	2.003670	118.911640	5.994787	6.997818	5.996468	0.219410	0.250800	0.079200
std	15.60460	1.417779	84.409662	1.780954	1.094622	2.468406	0.413849	0.433476	0.270052
min	18.00000	0.000000	0.000000	0.000000	3.000000	0.500000	0.000000	0.000000	0.000000
25%	39.00000	1.000000	57.000000	4.800000	6.300000	4.300000	0.000000	0.000000	0.000000
50%	50.00000	2.000000	100.000000	6.000000	7.000000	6.000000	0.000000	0.000000	0.000000
75%	61.00000	3.000000	160.000000	7.200000	7.700000	7.700000	0.000000	1.000000	0.000000
max	90.00000	10.000000	833.000000	10.000000	10.000000	16.800000	1.000000	1.000000	1.000000

8 rows × 24 columns

In [ ]:

```
df.isnull().sum()
```

	0
age	0
gender	0
ethnicity	0
education_level	0
income_level	0
employment_status	0
smoking_status	0
alcohol_consumption_per_week	0
physical_activity_minutes_per_week	0
diet_score	0
sleep_hours_per_day	0
screen_time_hours_per_day	0
family_history_diabetes	0
hypertension_history	0
cardiovascular_history	0
bmi	0
waist_to_hip_ratio	0
systolic_bp	0
diastolic_bp	0
heart_rate	0
cholesterol_total	0
hdl_cholesterol	0
ldl_cholesterol	0

triglycerides	0
glucose_fasting	0
glucose_postprandial	0
insulin_level	0
hba1c	0
diabetes_risk_score	0
diabetes_stage	0
diagnosed_diabetes	0

dtype: int64

In [ ]:

```
df.duplicated().sum()
```

```
np.int64(0)
```

In [ ]:

```
df["diagnosed_diabetes"].value_counts()
```

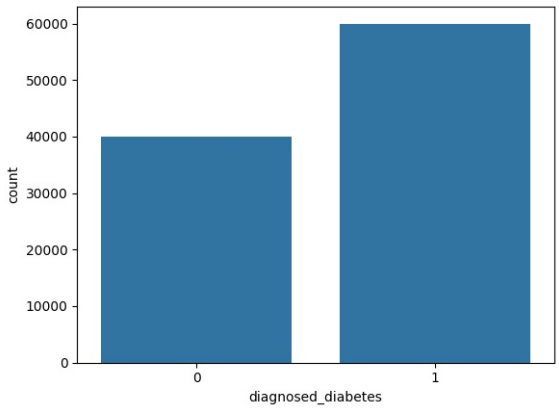
	count
diagnosed_diabetes	
1	59998
0	40002

dtype: int64

EDA

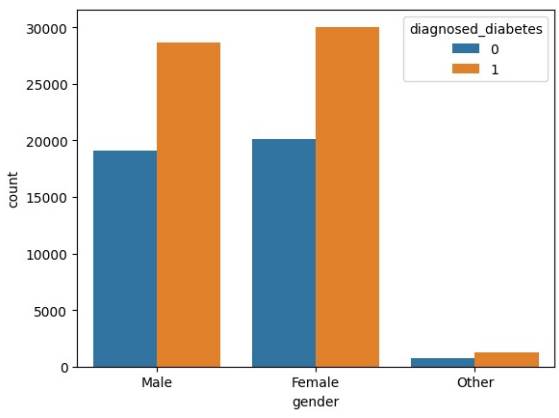
In [ ]:

```
sns.countplot(x="diagnosed_diabetes", data=df)
plt.show()
```



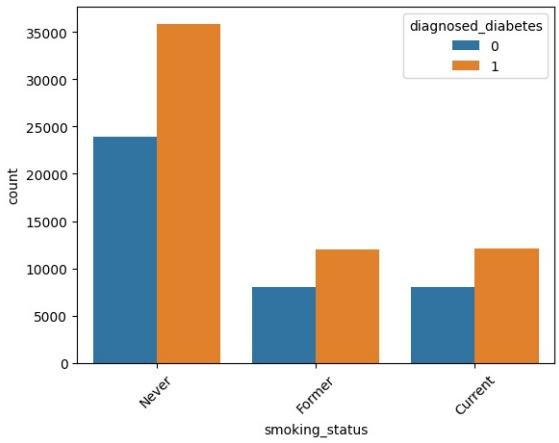
In [ ]:

```
sns.countplot(x="gender", hue="diagnosed_diabetes", data=df)
plt.show()
```



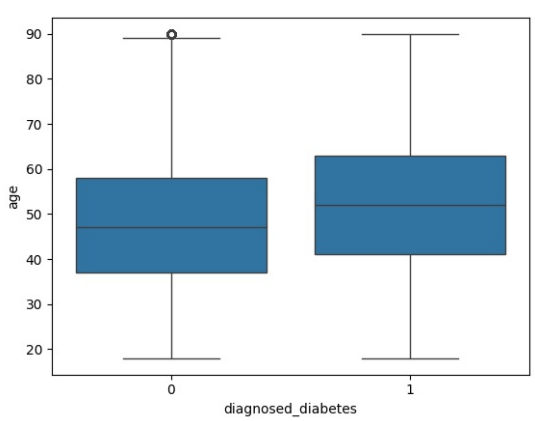
In [ ]:

```
sns.countplot(x="smoking_status", hue="diagnosed_diabetes", data=df)
plt.xticks(rotation=45)
plt.show()
```



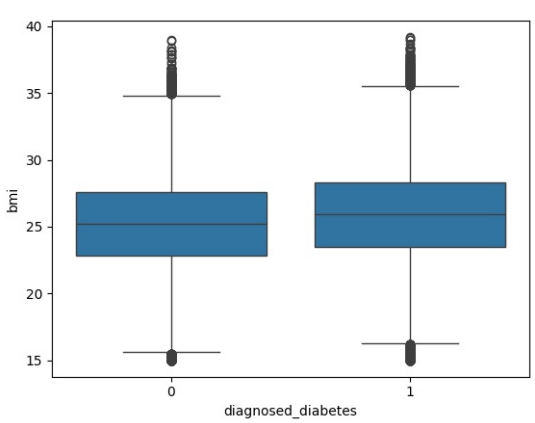
In [ ]:

```
sns.boxplot(x="diagnosed_diabetes", y="age", data=df)
plt.show()
```



In [ ]:

```
sns.boxplot(x="diagnosed_diabetes", y="bmi", data=df)
plt.show()
```



Data Preprocessing

In [ ]:

```
x = df.drop(
    columns=["diagnosed_diabetes", "diabetes_stage", "diabetes_risk_score"],errors="ignore",)
y = df["diagnosed_diabetes"]
```

In [ ]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

In [ ]:

```
categorical_cols = x.select_dtypes(include=["object", "category"]).columns
```

In [ ]:

```
numeric_cols = x.select_dtypes(include=[np.number]).columns
```

In [ ]:

```
preprocessor = ColumnTransformer(
    transformers=[
        ("num", StandardScaler(), numeric_cols),
        ("cat", OneHotEncoder(drop="first", handle_unknown="ignore"), categorical_cols),])
```

Outliers

In [ ]:

```
numeric_cols = x_train.select_dtypes(include=[np.number]).columns
categorical_cols = x_train.select_dtypes(
    include=["object", "category"]
).columns

for col in numeric_cols:
    Q1 = x_train[col].quantile(0.25)
    Q3 = x_train[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    x_train[col] = np.clip(x_train[col], lower, upper)
    x_test[col] = np.clip(x_test[col], lower, upper)
```

ANOVA Feature Selection

In [ ]:

```
x_train_prep = preprocessor.fit_transform(x_train)
x_test_prep = preprocessor.transform(x_test)
```

In [ ]:

```
anova = SelectKBest(score_func=f_classif, k=10)
x_train_anova = anova.fit_transform(x_train_prep, y_train)
x_test_anova = anova.transform(x_test_prep)
```

```
/usr/local/lib/python3.13/dist-packages/sklearn/feature_selection/_univariate_selection.py:111: UserWarning: Features [6 8] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/usr/local/lib/python3.13/dist-packages/sklearn/feature_selection/_univariate_selection.py:112: RuntimeWarning: invalid value encountered in divide
  f = msb / msw
```

In [ ]:

```
logistic_model = LogisticRegression(max_iter=1000, random_state=42)
logistic_model.fit(x_train_anova, y_train)
y_pred_logistic = logistic_model.predict(x_test_anova)

print("\n================ Logistic Regression ================")
print("Accuracy:", accuracy_score(y_test, y_pred_logistic))
print("\nClassification Report:\n", classification_report(y_test, y_pred_logistic))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_logistic))
```

```
================ Logistic Regression ================
Accuracy: 0.85675

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.81      0.82      8077
           1       0.87      0.89      0.88     11923

    accuracy                           0.86     20000
   macro avg       0.85      0.85      0.85     20000
weighted avg       0.86      0.86      0.86     20000


Confusion Matrix:
 [[ 6520  1557]
 [ 1308 10615]]
```

In [ ]:

```
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(x_train_anova, y_train)
y_pred_knn = knn_model.predict(x_test_anova)

print("\n================ K-Nearest Neighbors (KNN) ================")
print("Accuracy:", accuracy_score(y_test, y_pred_knn))
print("\nClassification Report:\n", classification_report(y_test, y_pred_knn))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_knn))
```

```
================ K-Nearest Neighbors (KNN) ================
Accuracy: 0.8557

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.86      0.83      8077
           1       0.90      0.85      0.88     11923

    accuracy                           0.86     20000
   macro avg       0.85      0.86      0.85     20000
weighted avg       0.86      0.86      0.86     20000


Confusion Matrix:
 [[ 6985  1092]
 [ 1794 10129]]
```

In [ ]:

```
svm_model = SVC(random_state=42)
svm_model.fit(x_train_anova, y_train)
y_pred_svm = svm_model.predict(x_test_anova)

print("\n================ Support Vector Machine (SVM) ================")
print("Accuracy:", accuracy_score(y_test, y_pred_svm))
print("\nClassification Report:\n", classification_report(y_test, y_pred_svm))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_svm))
```

```
================ Support Vector Machine (SVM) ================
Accuracy: 0.9022

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.96      0.89      8077
           1       0.97      0.87      0.91     11923

    accuracy                           0.90     20000
   macro avg       0.90      0.91      0.90     20000
weighted avg       0.91      0.90      0.90     20000


Confusion Matrix:
 [[ 7717   360]
 [ 1596 10327]]
```