```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
```

```
df=pd.read_csv('diabetes_prediction_dataset.csv')
df.head()
```

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0

```
df.describe()
```

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabetes
count	100000.000000	100000.00000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000
mean	41.885856	0.07485	0.039420	27.320767	5.527507	138.058060	0.085000
std	22.516840	0.26315	0.194593	6.636783	1.070672	40.708136	0.278883
min	0.080000	0.00000	0.000000	10.010000	3.500000	80.000000	0.000000
25%	24.000000	0.00000	0.000000	23.630000	4.800000	100.000000	0.000000
50%	43.000000	0.00000	0.000000	27.320000	5.800000	140.000000	0.000000
75%	60.000000	0.00000	0.000000	29.580000	6.200000	159.000000	0.000000
max	80.000000	1.00000	1.000000	95.690000	9.000000	300.000000	1.000000

```
df.columns
```

```
Index(['gender', ' age  ', ' hypertension', ' heart_disease',
       ' smoking_history', ' bmi  ', ' HbA1c_level', ' blood_glucose_level',
       ' diabetes'],
      dtype='object')
```

```
df[" smoking_history"].unique()
```

```
array([' never          ', ' No Info        ', ' current        ',
       ' former         ', ' ever           ', ' not current    '],
      dtype=object)
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df[" smoking_history"] = le.fit_transform(df[" smoking_history"])
df['gender']= le.fit_transform(df['gender'])
df.head()
```

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	0	80.0	0	1	4	25.19	6.6	140	0
1	0	54.0	0	0	0	27.32	6.6	80	0
2	1	28.0	0	0	4	27.32	5.7	158	0
3	0	36.0	0	0	1	23.45	5.0	155	0
4	1	76.0	1	1	1	20.14	4.8	155	0

```
# Separate features (X) and target (y)
X = df.drop(' diabetes', axis=1)
y = df[' diabetes']
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
numerical_cols = X.select_dtypes(include=[np.number]).columns.tolist()
X[numerical_cols] = scaler.fit_transform(X[numerical_cols])
X.head()
```

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level
0	-0.841047	1.692704	-0.284439	4.936379	0.963327	-0.321056	1.001706	0.047704
1	-0.841047	0.538006	-0.284439	-0.202578	-1.153468	-0.000116	1.001706	-1.426210
2	1.187234	-0.616691	-0.284439	-0.202578	0.963327	-0.000116	0.161108	0.489878
3	-0.841047	-0.261399	-0.284439	-0.202578	-0.624269	-0.583232	-0.492690	0.416183
4	1.187234	1.515058	3.515687	4.936379	-0.624269	-1.081970	-0.679490	0.416183

```
# Perform train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(C=0.5, max_iter=1000)  # smaller C = stronger regularization
model.fit(X_train, y_train)


y_pred = model.predict(X_test)
```

```
from sklearn.model_selection import cross_val_score



print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

scores = cross_val_score(model, X, y, cv=5, scoring="accuracy")
print("Cross-validation scores:", scores)
print("Mean accuracy:", scores.mean())
```

```
Accuracy: 0.95865
Confusion Matrix:
 [[18127   165]
 [  662  1046]]
Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.99      0.98     18292
           1       0.86      0.61      0.72      1708

    accuracy                           0.96     20000
   macro avg       0.91      0.80      0.85     20000
weighted avg       0.96      0.96      0.96     20000

Cross-validation scores: [0.96105 0.96065 0.96005 0.9595  0.95965]
Mean accuracy: 0.9601799999999999
```

```
# Create sample data for prediction (use encoded values for categoricals and exact column names with spaces)
sample_data = {
    'gender': [0],  # 1 = male (encoded)
    ' age  ': [60],
    ' hypertension': [1],
    ' heart_disease': [0],
    ' smoking_history': [1],  # e.g., 2 = former (based on encoder)
    ' bmi  ': [36.5],
    ' HbA1c_level': [9.2],
    ' blood_glucose_level': [170]
}
sample_df = pd.DataFrame(sample_data)

# Scale numerical columns using the fitted scaler
# Scale numerical columns using the fitted scaler
sample_df[numerical_cols] = scaler.transform(sample_df[numerical_cols])

# Make prediction
prediction = model.predict(sample_df)
print("Predicted diabetes outcome (0=no, 1=yes):", prediction[0])
```

```
Predicted diabetes outcome (0=no, 1=yes): 1
```