In [1]:
```
#Importing Required Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split, cross_val_score, Shuffl
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, Grad
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.decomposition import PCA
```

executed in 7.53s, finished 12:25:05 2023-06-30

In [2]:
```
!pip install imbalanced-learn
```

executed in 8.89s, finished 12:25:14 2023-06-30

```
Requirement already satisfied: imbalanced-learn in c:\users\dell\anaconda3\li
b\site-packages (0.10.1)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\dell\anaconda3
\lib\site-packages (from imbalanced-learn) (1.0.2)
Requirement already satisfied: joblib>=1.1.1 in c:\users\dell\anaconda3\lib\s
ite-packages (from imbalanced-learn) (1.2.0)
Requirement already satisfied: scipy>=1.3.2 in c:\users\dell\anaconda3\lib\si
te-packages (from imbalanced-learn) (1.7.3)
Requirement already satisfied: numpy>=1.17.3 in c:\users\dell\anaconda3\lib\s
ite-packages (from imbalanced-learn) (1.21.5)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\dell\anaconda
3\lib\site-packages (from imbalanced-learn) (2.2.0)
```

Importing CSV File

In [3]:
```
data=pd.read_csv('diabetes_prediction_dataset.csv')
```

executed in 275ms, finished 12:25:15 2023-06-30

In [4]:
```
data.head()
```

executed in 59ms, finished 12:25:17 2023-06-30

Out[4]:

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

In [5]:
```
data.tail()
```
executed in 31ms, finished 12:25:20 2023-06-30

Out[5]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glu
99995	Female	80.0	0	0	No Info	27.32	6.2	
99996	Female	2.0	0	0	No Info	17.37	6.5	
99997	Male	66.0	0	0	former	27.83	5.7	
99998	Female	24.0	0	0	never	35.42	4.0	
99999	Female	57.0	0	0	current	22.43	6.6	

In [6]:
```
data.dtypes
```
executed in 18ms, finished 12:25:22 2023-06-30

Out[6]:
```
gender                object
age                  float64
hypertension           int64
heart_disease          int64
smoking_history       object
bmi                  float64
HbA1c_level          float64
blood_glucose_level    int64
diabetes               int64
dtype: object
```

Summary of DataFrame

In [7]:
```
data.describe()
```
executed in 85ms, finished 12:25:25 2023-06-30

Out[7]:

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose
count	100000.000000	100000.00000	100000.000000	100000.000000	100000.000000	100000.0
mean	41.885856	0.07485	0.039420	27.320767	5.527507	138.0
std	22.516840	0.26315	0.194593	6.636783	1.070672	40.7
min	0.080000	0.00000	0.000000	10.010000	3.500000	80.0
25%	24.000000	0.00000	0.000000	23.630000	4.800000	100.0
50%	43.000000	0.00000	0.000000	27.320000	5.800000	140.0
75%	60.000000	0.00000	0.000000	29.580000	6.200000	159.0
max	80.000000	1.00000	1.000000	95.690000	9.000000	300.0

In [8]:
```
data.isnull()
```
executed in 107ms, finished 12:25:28 2023-06-30

Out[8]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_gl
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
99995	False	False	False	False	False	False	False	
99996	False	False	False	False	False	False	False	
99997	False	False	False	False	False	False	False	
99998	False	False	False	False	False	False	False	
99999	False	False	False	False	False	False	False	

100000 rows × 9 columns

Checking Null Counts

In [9]:
```
data.isnull().sum()
```
executed in 88ms, finished 12:25:30 2023-06-30

Out[9]:
```
gender                0
age                   0
hypertension          0
heart_disease         0
smoking_history       0
bmi                   0
HbA1c_level           0
blood_glucose_level   0
diabetes              0
dtype: int64
```

In [10]:
```
data.info()
```
executed in 89ms, finished 12:25:33 2023-06-30

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   gender               100000 non-null  object 
 1   age                  100000 non-null  float64
 2   hypertension         100000 non-null  int64  
 3   heart_disease        100000 non-null  int64  
 4   smoking_history      100000 non-null  object 
 5   bmi                  100000 non-null  float64
 6   HbA1c_level          100000 non-null  float64
 7   blood_glucose_level  100000 non-null  int64  
 8   diabetes             100000 non-null  int64  
dtypes: float64(3), int64(4), object(2)
memory usage: 6.9+ MB
```

In [11]:
```
data.duplicated().sum()
```
executed in 100ms, finished 12:25:35 2023-06-30

Out[11]:
```
3854
```

Shape of Dataset

In [12]:
```
data.shape
```
executed in 15ms, finished 12:25:37 2023-06-30

Out[12]:
```
(100000, 9)
```

Remove Duplicate Values

In [13]:
```
data.drop_duplicates(inplace=True)
```
executed in 107ms, finished 12:25:39 2023-06-30

In [14]:
```
data.shape
```
executed in 17ms, finished 12:25:43 2023-06-30

Out[14]:
```
(96146, 9)
```

Columns in DataFrame

In [15]:
```
data.columns
```
executed in 18ms, finished 12:25:45 2023-06-30

Out[15]:
```
Index(['gender', 'age', 'hypertension', 'heart_disease', 'smoking_history',
       'bmi', 'HbA1c_level', 'blood_glucose_level', 'diabetes'],
      dtype='object')
```

Unique Values in Given Columns

In [16]:
```
columns=['gender', 'age', 'hypertension', 'heart_disease', 'smoking_history',
       'bmi', 'HbA1c_level', 'blood_glucose_level', 'diabetes']
for col in columns:
    print(col,data[col].unique())
```

executed in 62ms, finished 12:25:48 2023-06-30

```
gender ['Female' 'Male' 'Other']
age [80.   54.   28.   36.   76.   20.   44.   79.   42.   32.   53.   78.
 67.   15.   37.   40.    5.   69.   72.    4.   30.   45.   43.   50.
 41.   26.   34.   73.   77.   66.   29.   60.   38.    3.   57.   74.
 19.   46.   21.   59.   27.   13.   56.    2.    7.   11.    6.   55.
  9.   62.   47.   12.   68.   75.   22.   58.   18.   24.   17.   25.
  0.08 33.   16.   61.   31.    8.   49.   39.   65.   14.   70.    0.56
 48.   51.   71.    0.88 64.   63.   52.    0.16 10.   35.   23.    0.64
  1.16  1.64  0.72  1.88  1.32  0.8   1.24  1.    1.8   0.48  1.56  1.08
  0.24  1.4   0.4   0.32  1.72  1.48]
hypertension [0 1]
heart_disease [1 0]
smoking_history ['never' 'No Info' 'current' 'former' 'ever' 'not current']
bmi [25.19 27.32 23.45 ... 59.42 44.39 60.52]
HbA1c_level [6.6 5.7 5.  4.8 6.5 6.1 6.  5.8 3.5 6.2 4.  4.5 9.  7.  8.8 8.2
7.5 6.8]
blood_glucose_level [140  80 158 155  85 200 145 100 130 160 126 159  90 260
220 300 280 240]
diabetes [0 1]
```

In [17]:
```
data[data['gender'] == "Other"].shape
```

executed in 39ms, finished 12:25:51 2023-06-30

Out[17]: (18, 9)

In [18]:
```
data[data['smoking_history']=='No Info'].shape
```

executed in 42ms, finished 12:25:53 2023-06-30

Out[18]: (32887, 9)

In [19]:
```
data[data['smoking_history'] == "not current"].shape
```

executed in 34ms, finished 12:25:55 2023-06-30

Out[19]: (6367, 9)

In [20]:
```
data[data['smoking_history'] == "former"].shape
```

executed in 42ms, finished 12:25:58 2023-06-30

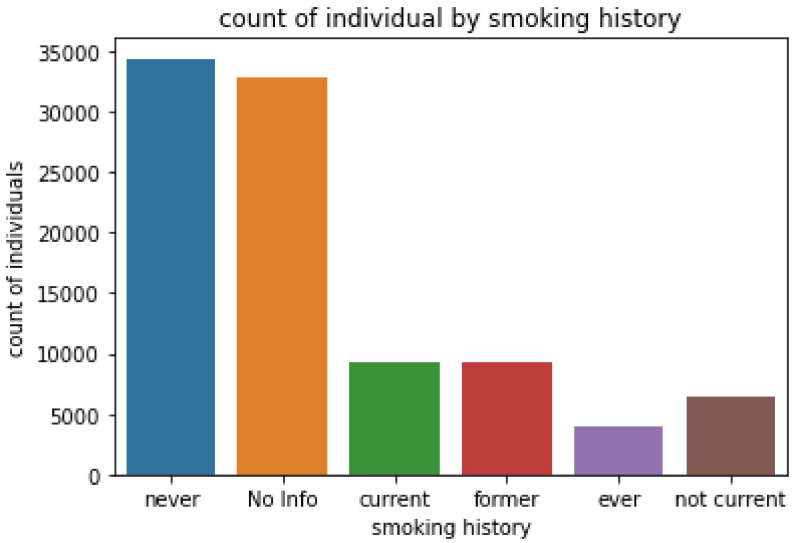
Out[20]: (9299, 9)

Smoking History Distribution

In [21]:
```
sns.countplot('smoking_history',data=data)
plt.title('count of individual by smoking history')
plt.xlabel('smoking history')
plt.ylabel('count of individuals')
plt.show()
```

executed in 513ms, finished 12:26:01 2023-06-30

```
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [22]:
```
# Filter data based on Smoking history and diabetes status
never1 = data[(data['smoking_history'] == 'never') & (data['diabetes'] == 1)]
never0 = data[(data['smoking_history'] == 'never') & (data['diabetes'] == 0)]
noinfo1 = data[(data['smoking_history'] == 'No Info') & (data['diabetes'] ==
noinfo0 = data[(data['smoking_history'] == 'No Info') & (data['diabetes'] ==
current1 = data[(data['smoking_history'] == 'current') & (data['diabetes'] ==
current0 = data[(data['smoking_history'] == 'current') & (data['diabetes'] ==
former1 = data[(data['smoking_history'] == 'former') & (data['diabetes'] == 1
former0 = data[(data['smoking_history'] == 'former') & (data['diabetes'] == 0
ever1 = data[(data['smoking_history'] == 'ever') & (data['diabetes'] == 1)]
ever0 = data[(data['smoking_history'] == 'ever') & (data['diabetes'] == 0)]
notcurrent1 = data[(data['smoking_history'] == 'not current') & (data['diabet
notcurrent0 = data[(data['smoking_history'] == 'not current') & (data['diabet
```

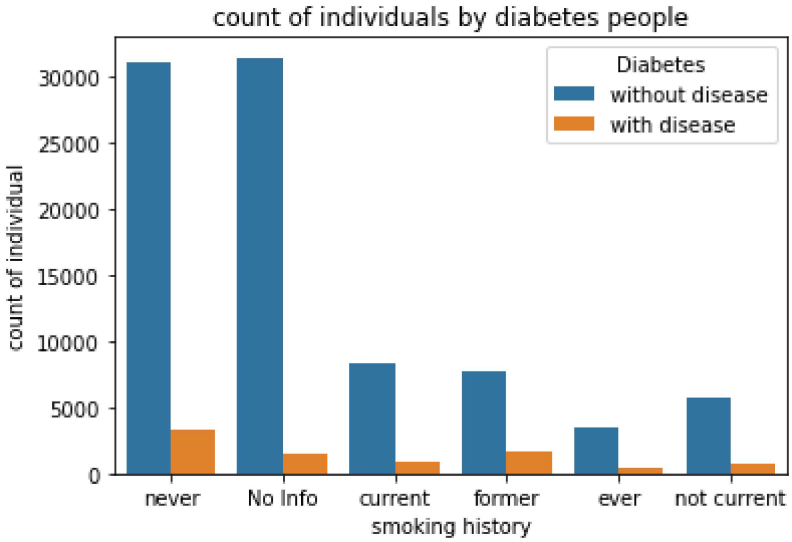
executed in 238ms, finished 12:26:03 2023-06-30

In [23]:
```
# Combine the filtered data into a single DataFrame
smoking_history_data = pd.concat([never1, never0, noinfo1, noinfo0, current1,
```

executed in 37ms, finished 12:26:06 2023-06-30

In [24]:
```
sns.countplot(x='smoking_history',hue='diabetes',data=smoking_history_data)
plt.title('count of individuals by diabetes people')
plt.xlabel('smoking history')
plt.ylabel('count of individual')
plt.legend(title='Diabetes',labels=['without disease','with disease'])
plt.show()
```

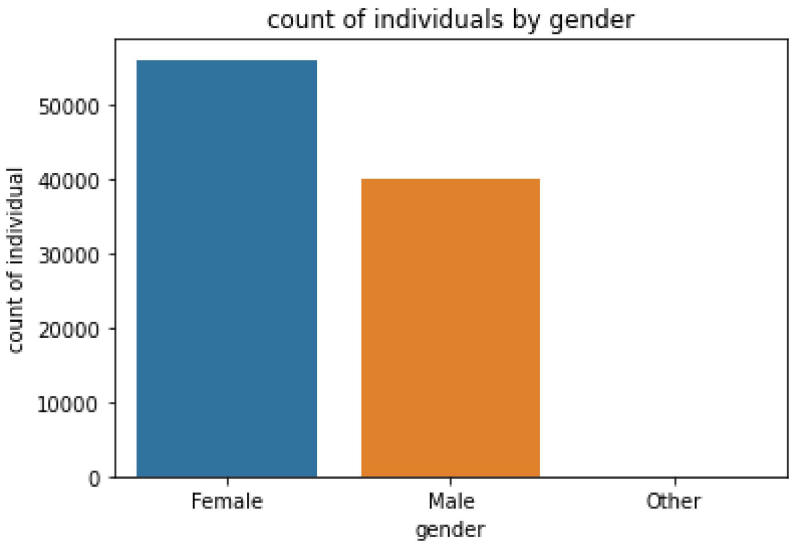
executed in 435ms, finished 12:26:08 2023-06-30



Gender Distribution

In [25]:
```
sns.countplot(x='gender',data=data)
plt.title('count of individuals by gender')
plt.xlabel('gender')
plt.ylabel('count of individual')
plt.show()
```

executed in 296ms, finished 12:26:11 2023-06-30



In [26]:
```
# Filter data based on gender and diabetes status
male1 = data[( data['gender'] == 'Male') & ( data['diabetes'] == 1)]
male0 =  data[( data['gender'] == 'Male') & ( data['diabetes'] == 0)]
female1 =  data[( data['gender'] == 'Female') & ( data['diabetes'] == 1)]
female0 =  data[( data['gender'] == 'Female') & ( data['diabetes'] == 0)]
other1 =  data[( data['gender'] == 'Other') & ( data['diabetes'] == 1)]
other0 =  data[( data['gender'] == 'Other') & ( data['diabetes'] == 0)]
```

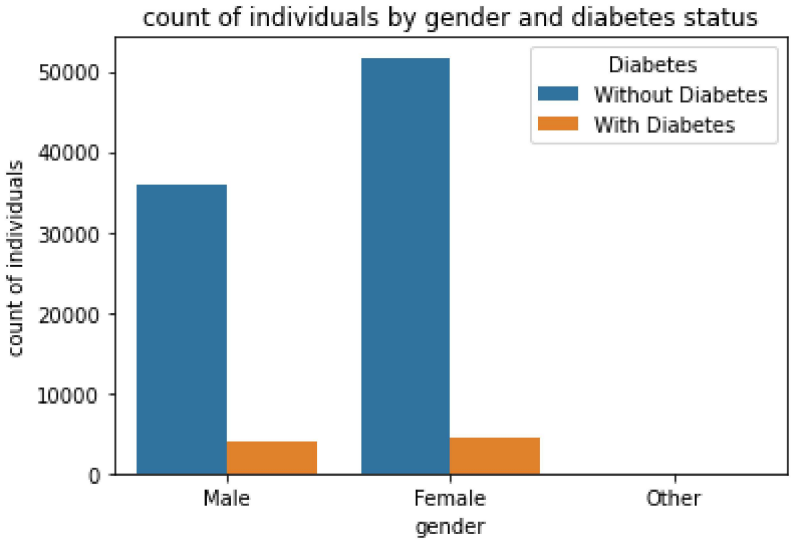
executed in 147ms, finished 12:26:14 2023-06-30

In [27]:
```
# Combine the filtered data into a single DataFrame
gender_data = pd.concat([male1, male0, female1, female0, other1, other0])
```

executed in 38ms, finished 12:26:18 2023-06-30

In [28]:
```
# Create the histogram using Seaborn
sns.countplot(x='gender',hue='diabetes',data=gender_data)
plt.title('count of individuals by gender and diabetes status')
plt.xlabel('gender')
plt.ylabel('count of individuals')
plt.legend(title='Diabetes', labels=['Without Diabetes', 'With Diabetes'])
plt.show()
```

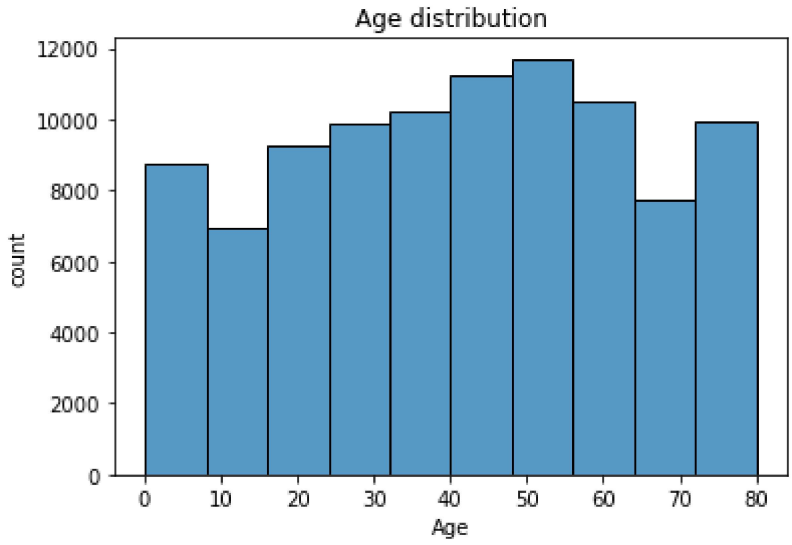
executed in 375ms, finished 12:26:20 2023-06-30



Age distribution

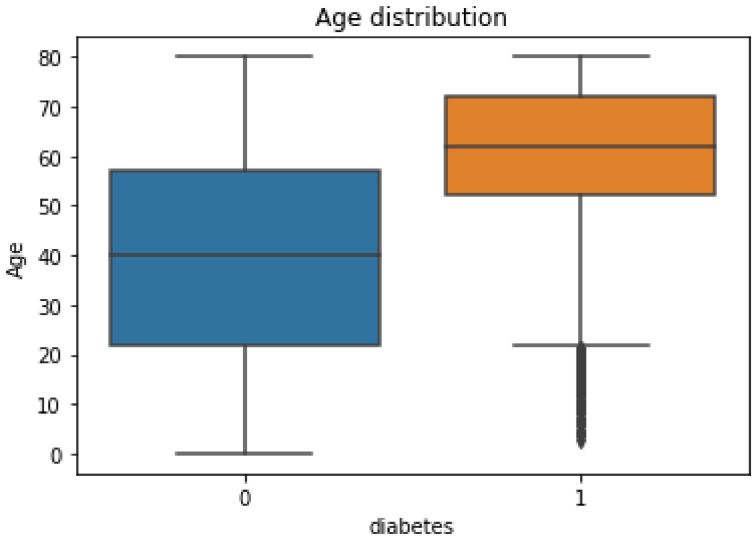
In [29]:
```
sns.histplot(data['age'],bins=10)
plt.title('Age distribution')
plt.xlabel('Age')
plt.ylabel('count')
plt.show()
```

executed in 423ms, finished 12:26:24 2023-06-30



In [30]:
```
sns.boxplot(x=data['diabetes'],y=data['age'])
plt.title('Age distribution')
plt.xlabel('diabetes')
plt.ylabel('Age')
plt.show()
```

executed in 261ms, finished 12:26:26 2023-06-30



Age-related changes, sedentary lifestyles, higher body weight, genetic factors, concurrent medical illnesses, certain medications, poor food, chronic inflammation, and decreased awareness and screening make older people more vulnerable to diabetes. Age-related changes in lifestyle, consistent exercise, a balanced diet, and the right medical treatment can help lower the risk of and manage diabetes.

Hypertension Distribution

In [31]:
```
# Filter data based on hypertension and diabetes status
hypertension1 = data[(data['hypertension'] == 1) & (data['diabetes'] == 1)]
hypertension0 = data[(data['hypertension'] == 1) & (data['diabetes'] == 0)]
nohypertension1 = data[(data['hypertension'] == 0)  & (data['diabetes'] == 1)
nohypertension0 = data[(data['hypertension'] == 0)  & (data['diabetes'] == 0)
```

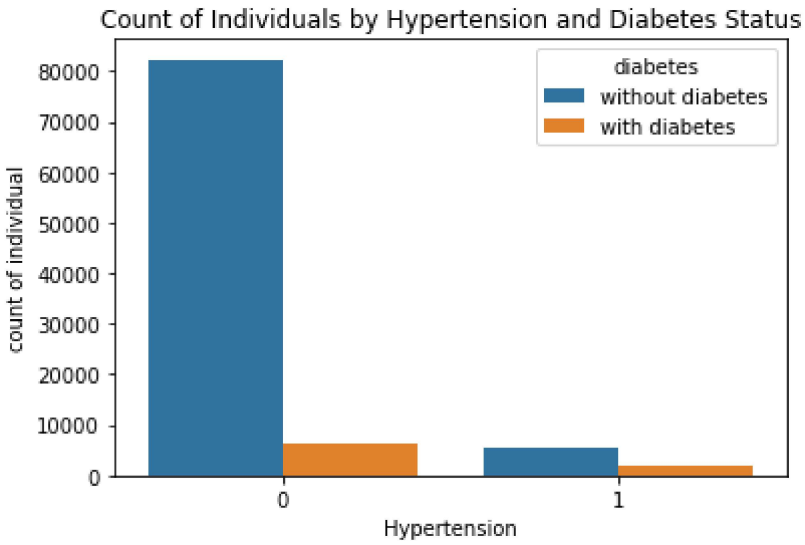
executed in 38ms, finished 12:26:30 2023-06-30

In [32]:
```
# Combine the filtered data into a single DataFrame
hypertension_data= pd.concat([hypertension1, hypertension0, nohypertension1,
```

executed in 30ms, finished 12:26:33 2023-06-30

In [33]:
```
# Create the histogram using Seaborn
sns.countplot(x='hypertension',hue='diabetes',data=hypertension_data)
plt.title('Count of Individuals by Hypertension and Diabetes Status')
plt.xlabel('Hypertension')
plt.ylabel('count of individual')
plt.legend(title='diabetes',labels=['without diabetes','with diabetes'])
plt.show()
```

executed in 375ms, finished 12:26:35 2023-06-30



People with hypertension are more susceptible to diabetes.

Heart Disease Distribution

In [34]:
```
# Filter data based on heart_disease and diabetes status
heart_disease1 = data[(data['heart_disease'] == 1) & (data['diabetes'] == 1)]
heart_disease0 = data[(data['heart_disease'] == 1) & (data['diabetes'] == 0)]
no_heart_disease1 = data[(data['heart_disease'] == 0) & (data['diabetes'] ==
no_heart_disease0 = data[(data['heart_disease'] == 0) & (data['diabetes'] ==
```

executed in 44ms, finished 12:26:37 2023-06-30

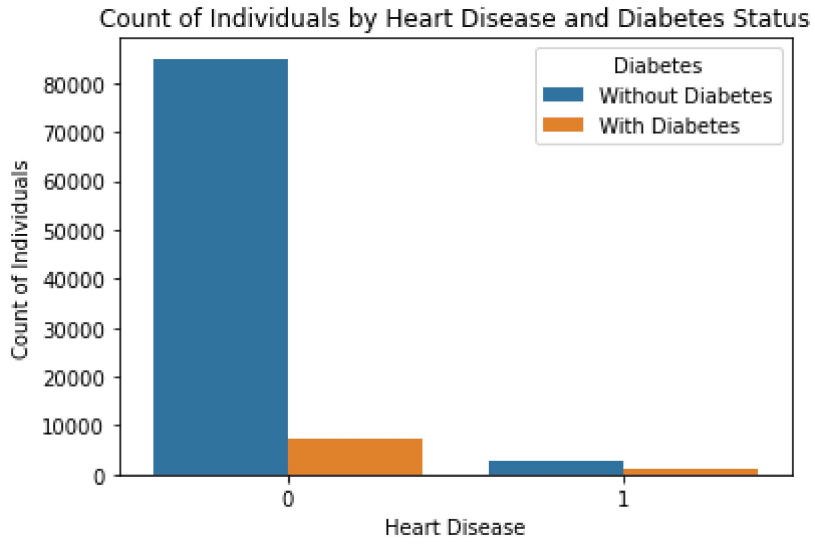
In [ ]:
```
```

In [35]:
```
# Combine the filtered data into a single DataFrame
heart_disease_data = pd.concat([heart_disease1, heart_disease0, no_heart_dise
```

executed in 22ms, finished 12:26:41 2023-06-30

In [36]:
```
# Create the histogram using Seaborn
sns.countplot(x='heart_disease', hue='diabetes', data=heart_disease_data)
plt.xlabel('Heart Disease')
plt.ylabel('Count of Individuals')
plt.title('Count of Individuals by Heart Disease and Diabetes Status')
plt.legend(title='Diabetes', labels=['Without Diabetes', 'With Diabetes'])
plt.show()
```

executed in 313ms, finished 12:26:43 2023-06-30

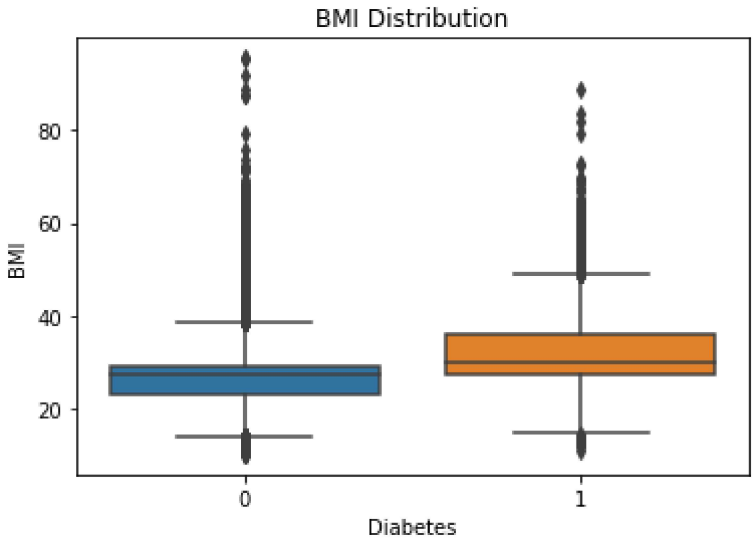


People with heart disease are more susceptible to diabetes.

BMI Distribution

In [37]:
```
sns.boxplot(x=data['diabetes'], y=data['bmi'])
plt.title('BMI Distribution')
plt.xlabel('Diabetes')
plt.ylabel('BMI')
plt.show()
```

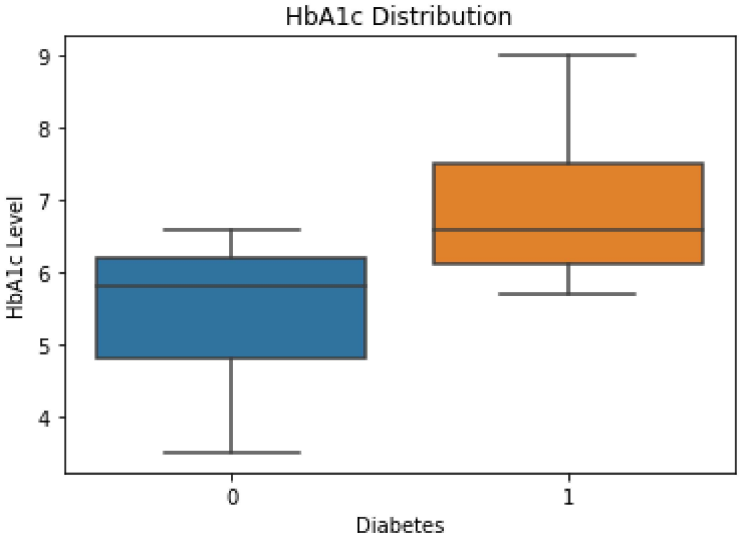
executed in 254ms, finished 12:26:46 2023-06-30



HbA1c Level Distribution

In [38]:
```
sns.boxplot(x=data['diabetes'], y=data['HbA1c_level'])
plt.title('HbA1c Distribution')
plt.xlabel('Diabetes')
plt.ylabel('HbA1c Level')
plt.show()
```

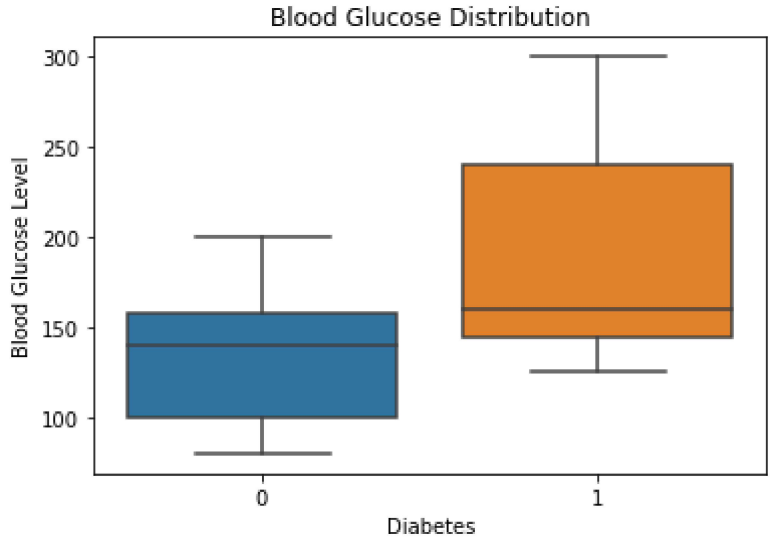
executed in 227ms, finished 12:26:49 2023-06-30



Blood Glucose Level Distribution

In [39]:
```
sns.boxplot(x=data['diabetes'], y=data['blood_glucose_level'])
plt.title('Blood Glucose Distribution')
plt.xlabel('Diabetes')
plt.ylabel('Blood Glucose Level')
plt.show()
```

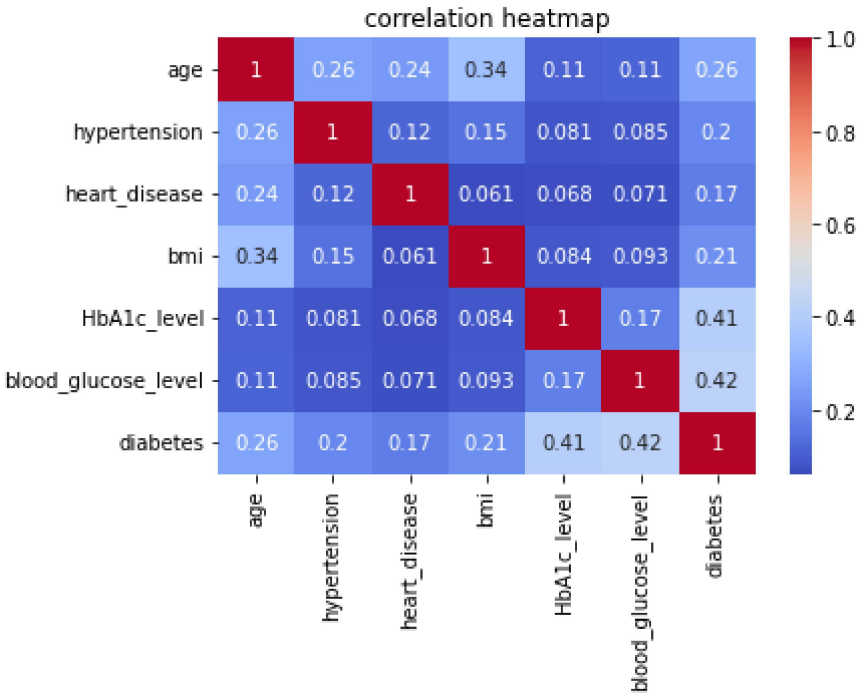
executed in 242ms, finished 12:26:52 2023-06-30



Correlation Heatmap

In [40]:
```
corr = data[['age', 'hypertension', 'heart_disease',
       'bmi', 'HbA1c_level', 'blood_glucose_level', 'diabetes']].corr()
sns.heatmap(corr,annot=True, cmap='coolwarm')
plt.title('correlation heatmap')
plt.show()
```

executed in 762ms, finished 12:26:55 2023-06-30



Doing One-Hot Encoding of 'gender' and 'smoking_history' columns

In [41]:
```
dummy_data=pd.get_dummies(data[['gender','smoking_history']],drop_first=True)
dummy_data.head()
```

executed in 78ms, finished 12:27:07 2023-06-30

Out[41]:

	gender_Male	gender_Other	smoking_history_current	smoking_history_ever	smoking_history_f
0	0	0	0	0	
1	0	0	0	0	
2	1	0	0	0	
3	0	0	1	0	
4	1	0	1	0	

In [ ]:

In [42]:
```
data.head()
```
executed in 32ms, finished 12:27:09 2023-06-30

Out[42]:

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

In [43]:
```
dummy_data.head()
```
executed in 26ms, finished 12:27:12 2023-06-30

Out[43]:

	gender_Male	gender_Other	smoking_history_current	smoking_history_ever	smoking_history_f
0	0	0	0	0	
1	0	0	0	0	
2	1	0	0	0	
3	0	0	1	0	
4	1	0	1	0	

Concatinating One-Hot Encoding columns with other columns

In [44]:
```
X = pd.concat([data[['age','hypertension','heart_disease','bmi','HbA1c_level'
X.head()
```
executed in 49ms, finished 12:27:14 2023-06-30

Out[44]:

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

Scaling X

In [45]:
```
scaler=MinMaxScaler()
X=scaler.fit_transform(X)
X[0]
```

executed in 63ms, finished 12:27:17 2023-06-30

Out[45]:
```
array([1.        , 0.        , 1.        , 0.17717087, 0.56363636,
       0.27272727, 0.        , 0.        , 0.        , 0.        ,
       0.        , 1.        , 0.        ])
```

Defining y

In [46]:
```
y=data['diabetes']
```

executed in 8ms, finished 12:27:20 2023-06-30

Value Counts of y; is person diabetic or not

In [47]:
```
y.value_counts()
```

executed in 20ms, finished 12:27:22 2023-06-30

Out[47]:
```
0    87664
1     8482
Name: diabetes, dtype: int64
```

We can see our y is imbalanced. So for solving this problem, we oversample our dataset with creating imaginary '1' values in dataset.

SMOTE

In [48]:
```
smote=SMOTE(sampling_strategy='minority')
X,y=smote.fit_resample(X,y)
y.value_counts()
```

executed in 1.30s, finished 12:27:26 2023-06-30

Out[48]:
```
0    87664
1    87664
Name: diabetes, dtype: int64
```

Defining with different algorithms with there parameters

In [49]:

```
algos = {
    'logistic_regression': {
        'model': LogisticRegression(),
        'params': {
        }
    },
    'SVM': {
        'model': SVC(),
        'params': {
        }
    },
    'decision_tree': {
        'model': DecisionTreeClassifier(),
        'params': {
            'criterion': ['gini', 'entropy', 'log_loss'],
            'splitter': ['best','random']
        }
    },
    'random_forest': {
        'model': RandomForestClassifier(),
        'params': {
            'n_estimators': [1,5,10,20,50],
            'criterion': ['gini', 'entropy', 'log_loss']
        }
    },
    'ada_boost_classifier': {
        'model': AdaBoostClassifier(),
        'params': {
            'n_estimators': [1,5,10,20,50,100]
        }
    },
    'gradient_boosting_classifier': {
        'model': GradientBoostingClassifier(),
        'params': {
            'n_estimators': [1,5,10,20,50,100],
            'loss': ['log_loss', 'exponential']
        }
    },
    'bagging_classifier': {
         'model': BaggingClassifier(),
        'params': {
            'n_estimators': [1,5,10,20,50]
        }
    },
    'gaussian_naive_bayes': {
        'model': GaussianNB(),
        'params': {
        }
    },
    'multinomial_naive_bayes': {
        'model': MultinomialNB(),
        'params': {
        }
    }
```

```
}
```

executed in 30ms, finished 12:27:30 2023-06-30

Training all algorithms and finding best parameter and saving model score in 'scores'

In [50]:

```
scores= []
cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=10)
for algo_name, mp in algos.items():
    reg = GridSearchCV(mp['model'], mp['params'], cv=cv, return_train_score=Fa
    reg.fit(X, y)
    scores.append(
        {
            'model': algo_name,
            'best_score': reg.best_score_,
            'best_params': reg.best_params_
        }
    )
```

executed in 2h 56m 46s, finished 15:24:20 2023-06-30

```
C:\Users\DELL\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:372: FitFailedWarning:
10 fits failed out of a total of 30.
The score on these train-test partitions for these parameters will be set to
nan.
If these failures are not expected, you can try to debug them by setting erro
r_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\DELL\anaconda3\lib\site-packages\sklearn\model_selection\_va
lidation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\DELL\anaconda3\lib\site-packages\sklearn\tree\_classes.py",
line 937, in fit
    super().fit(
  File "C:\Users\DELL\anaconda3\lib\site-packages\sklearn\tree\_classes.py",
line 352, in fit
    criterion = CRITERIA_CLF[self.criterion](
KeyError: 'log_loss'

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\DELL\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:
969: UserWarning: One or more of the test scores are non-finite: [0.96374836
0.959773   0.96423316 0.96184908        nan        nan]
  warnings.warn(
C:\Users\DELL\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:372: FitFailedWarning:
25 fits failed out of a total of 75.
The score on these train-test partitions for these parameters will be set to
nan.
If these failures are not expected, you can try to debug them by setting erro
r_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\DELL\anaconda3\lib\site-packages\sklearn\model_selection\_va
lidation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\DELL\anaconda3\lib\site-packages\sklearn\ensemble\_forest.p
y", line 450, in fit
    trees = Parallel(
  File "C:\Users\DELL\anaconda3\lib\site-packages\joblib\parallel.py", line 1
085, in __call__
    if self.dispatch_one_batch(iterator):
  File "C:\Users\DELL\anaconda3\lib\site-packages\joblib\parallel.py", line 9
01, in dispatch_one_batch
    self._dispatch(tasks)
  File "C:\Users\DELL\anaconda3\lib\site-packages\joblib\parallel.py", line 8
19, in _dispatch
    job = self._backend.apply_async(batch, callback=cb)
```

```
  File "C:\Users\DELL\anaconda3\lib\site-packages\joblib\_parallel_backends.p
y", line 208, in apply_async
    result = ImmediateResult(func)
  File "C:\Users\DELL\anaconda3\lib\site-packages\joblib\_parallel_backends.p
y", line 597, in __init__
    self.results = batch()
  File "C:\Users\DELL\anaconda3\lib\site-packages\joblib\parallel.py", line 2
88, in __call__
    return [func(*args, **kwargs)
  File "C:\Users\DELL\anaconda3\lib\site-packages\joblib\parallel.py", line 2
88, in <listcomp>
    return [func(*args, **kwargs)
  File "C:\Users\DELL\anaconda3\lib\site-packages\sklearn\utils\fixes.py", li
ne 216, in __call__
    return self.function(*args, **kwargs)
  File "C:\Users\DELL\anaconda3\lib\site-packages\sklearn\ensemble\_forest.p
y", line 185, in _parallel_build_trees
    tree.fit(X, y, sample_weight=curr_sample_weight, check_input=False)
  File "C:\Users\DELL\anaconda3\lib\site-packages\sklearn\tree\_classes.py",
line 937, in fit
    super().fit(
  File "C:\Users\DELL\anaconda3\lib\site-packages\sklearn\tree\_classes.py",
line 352, in fit
    criterion = CRITERIA_CLF[self.criterion](
KeyError: 'log_loss'

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\DELL\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:
969: UserWarning: One or more of the test scores are non-finite: [0.94567958
0.96589289 0.97132265 0.97375235 0.97585125 0.94431073
 0.96770091 0.97183026 0.97419152 0.97557748        nan        nan
        nan        nan        nan]
  warnings.warn(
C:\Users\DELL\anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:372: FitFailedWarning:
30 fits failed out of a total of 60.
The score on these train-test partitions for these parameters will be set to
nan.
If these failures are not expected, you can try to debug them by setting erro
r_score='raise'.

Below are more details about the failures:
----------------------------------------------------------------------------
---
30 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\DELL\anaconda3\lib\site-packages\sklearn\model_selection\_va
lidation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\DELL\anaconda3\lib\site-packages\sklearn\ensemble\_gb.py", l
ine 525, in fit
    self._check_params()
  File "C:\Users\DELL\anaconda3\lib\site-packages\sklearn\ensemble\_gb.py", l
ine 282, in _check_params
    raise ValueError("Loss '{0:s}' not supported. ".format(self.loss))
ValueError: Loss 'log_loss' not supported.
```

```
  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\DELL\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:
969: UserWarning: One or more of the test scores are non-finite: [       nan
nan        nan        nan        nan        nan
 0.84823476 0.84823476 0.9000057  0.90816175 0.92866024 0.9567102 ]
  warnings.warn(
```

In [51]:
```
score = pd.DataFrame(scores, columns=['model','best_score','best_params'])
score
```
executed in 100ms, finished 15:29:10 2023-06-30

Out[51]:

	model	best_score	best_params
0	logistic_regression	0.887042	{}
1	SVM	0.897542	{}
2	decision_tree	0.964233	{'criterion': 'entropy', 'splitter': 'best'}
3	random_forest	0.975851	{'criterion': 'gini', 'n_estimators': 50}
4	ada_boost_classifier	0.956322	{'n_estimators': 100}
5	gradient_boosting_classifier	0.956710	{'loss': 'exponential', 'n_estimators': 100}
6	bagging_classifier	0.976884	{'n_estimators': 50}
7	gaussian_naive_bayes	0.644294	{}
8	multinomial_naive_bayes	0.617784	{}

In [54]:
```
X_train,X_rem,y_train,y_rem=train_test_split(X,y,test_size=0.3,random_state=1
```
executed in 152ms, finished 15:32:11 2023-06-30

In [55]:
```
X_test, X_valid, y_test, y_valid = train_test_split(X_rem, y_rem, test_size=0
```
executed in 67ms, finished 15:32:12 2023-06-30

In [56]:
```
model = RandomForestClassifier(n_estimators=100, criterion='entropy')

model.fit(X_train, y_train)
```
executed in 25.4s, finished 15:32:53 2023-06-30

Out[56]:
```
RandomForestClassifier(criterion='entropy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [57]:
```
#Accuracy Score
model.score(X_test, y_test)
```
executed in 1.06s, finished 15:34:36 2023-06-30

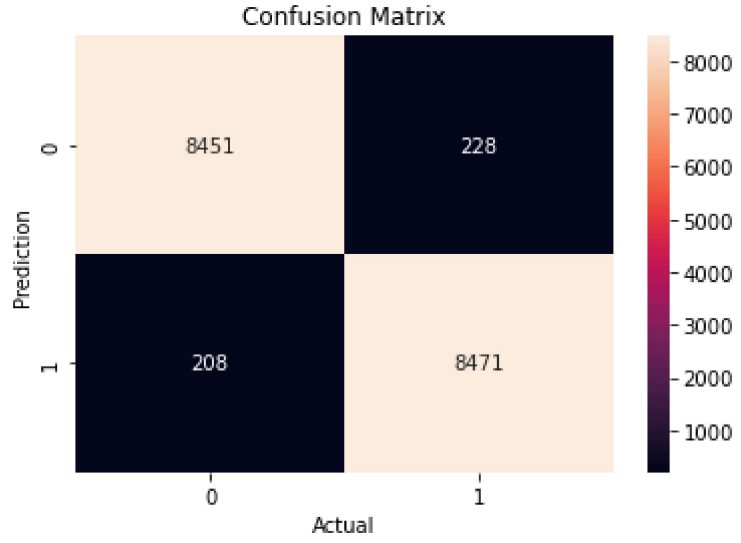
Out[57]:
```
0.9728157543770041
```

In [58]:
```
#Predicting for X_valid
y_pred = model.predict(X_valid)
```
executed in 536ms, finished 15:34:56 2023-06-30

In [59]:
```
#Confusion Matrix
cm = confusion_matrix(y_valid, y_pred)
cm
```
executed in 22ms, finished 15:35:16 2023-06-30

Out[59]:
```
array([[8451,  228],
       [ 208, 8471]], dtype=int64)
```

In [60]:
```
#Heatmap of Confusion Matrix
sns.heatmap(cm, annot=True, fmt=".0f")
plt.xlabel("Actual")
plt.ylabel("Prediction")
plt.title("Confusion Matrix")
plt.show()
```
executed in 329ms, finished 15:35:32 2023-06-30



In [61]:
```
#Classification Report
print(classification_report(y_valid, y_pred))
```
executed in 77ms, finished 15:35:49 2023-06-30

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.97      8679
           1       0.97      0.98      0.97      8679

    accuracy                           0.97     17358
   macro avg       0.97      0.97      0.97     17358
weighted avg       0.97      0.97      0.97     17358
```

In [62]:
```
#Principal Component Analysis(PCA)
#For reducing dimensions of dataset

pca = PCA(0.95)

X_pca = pca.fit_transform(X)
```

executed in 315ms, finished 15:36:11 2023-06-30

In [63]:
```
#Training our model
X_train, X_rem, y_train, y_rem = train_test_split(X_pca, y, test_size=0.3, ra
X_test, X_valid, y_test, y_valid = train_test_split(X_rem, y_rem, test_size=0
model = RandomForestClassifier(n_estimators=100, criterion='entropy')
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

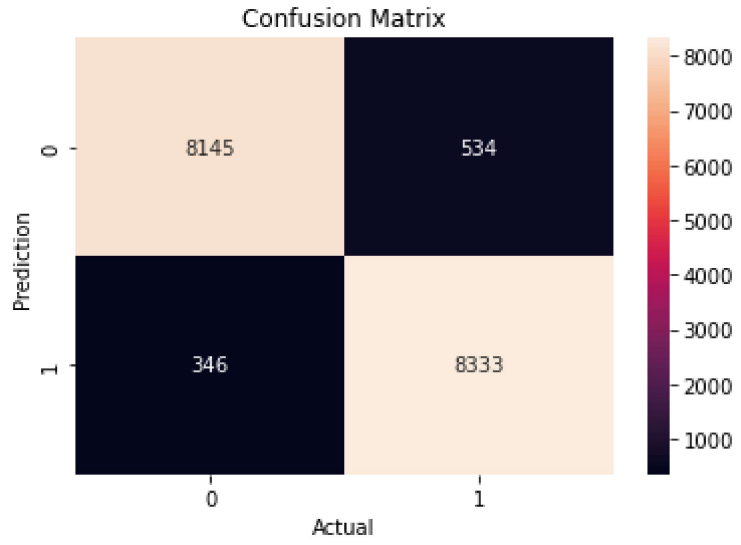
executed in 2m 26s, finished 15:39:02 2023-06-30

Out[63]: 0.9474475752674442

In [64]:
```
#Confusion Matrix
y_pred = model.predict(X_valid)
cm = confusion_matrix(y_valid, y_pred)

sns.heatmap(cm, annot=True, fmt=".0f")
plt.xlabel("Actual")
plt.ylabel("Prediction")
plt.title("Confusion Matrix")
plt.show()
```

executed in 1.29s, finished 15:41:06 2023-06-30



In [65]:
```
#Classification Report
print(classification_report(y_valid, y_pred))
```

executed in 90ms, finished 15:41:26 2023-06-30

```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      8679
           1       0.94      0.96      0.95      8679

    accuracy                           0.95     17358
   macro avg       0.95      0.95      0.95     17358
weighted avg       0.95      0.95      0.95     17358
```