DIABETES PREDICTION

```
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from imblearn.over_sampling import SMOTE
import warnings
warnings.filterwarnings("ignore")
```

```
data = pd.read_csv("/content/diabetes_prediction_dataset.csv")
data.head()
```

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

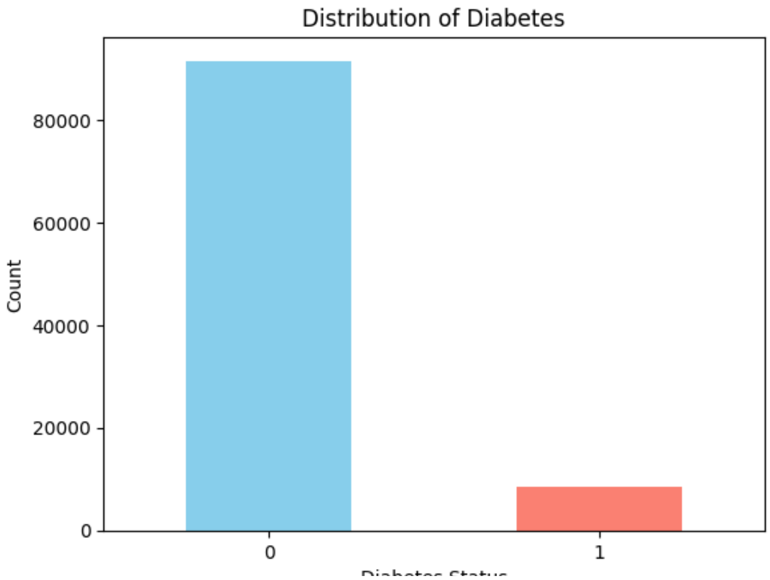
```
data.info()
```

```
<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
```

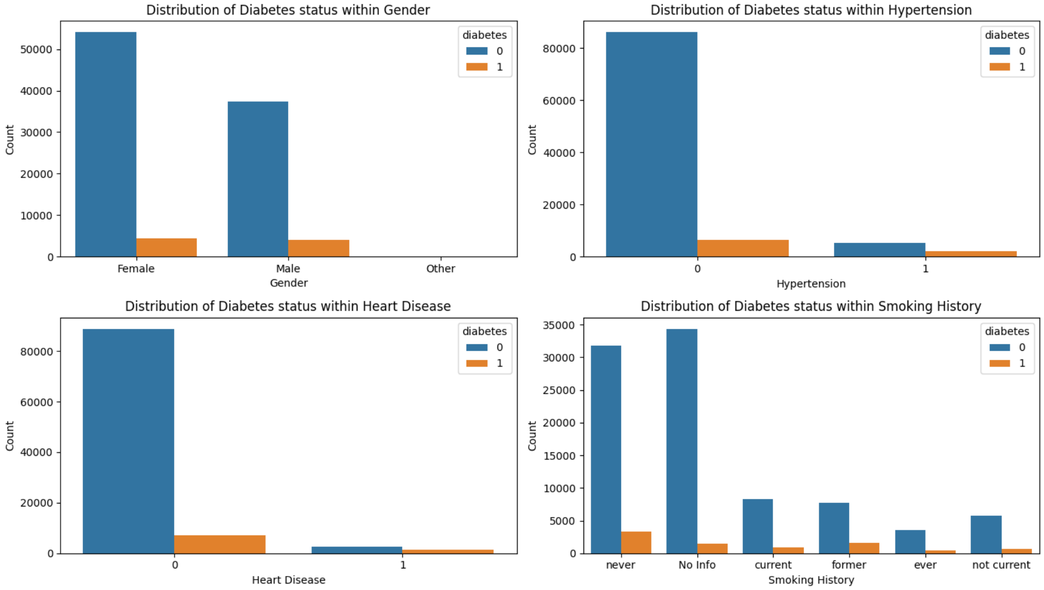
```
data.isna().sum()
```

```
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
```

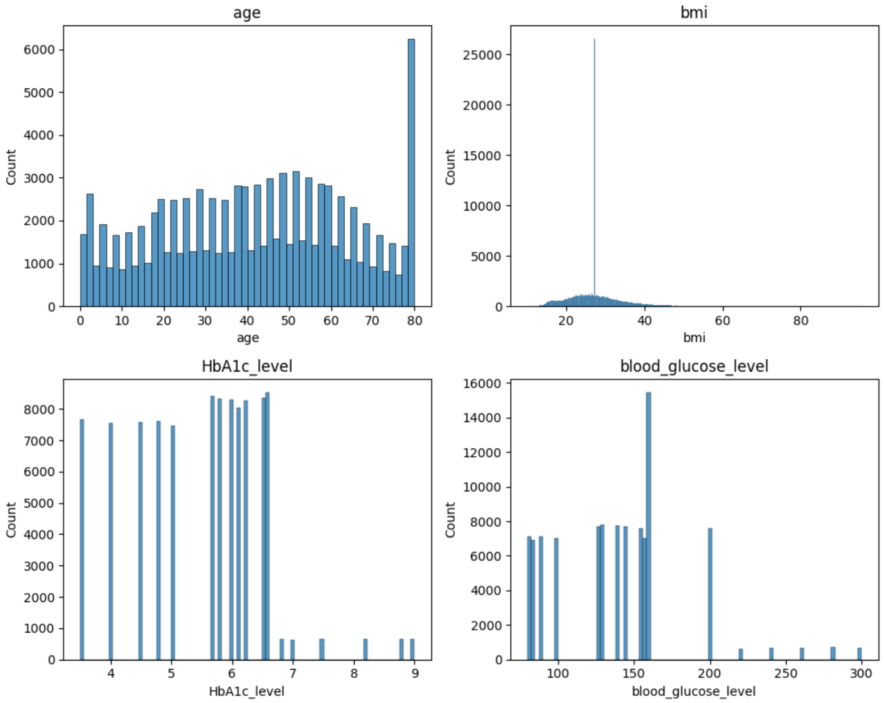
```
diabetes_counts = data['diabetes'].value_counts()
diabetes_counts.plot(kind='bar', color=['skyblue', 'salmon'])
plt.title('Distribution of Diabetes')
plt.xlabel('Diabetes Status')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.show()
```



```
# ploting categorical features alongiside target feature
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 8))

# Chart 1: Distribution of gender
sns.countplot(x="gender", data=data,hue = 'diabetes',ax=axes[0, 0])
axes[0, 0].set_title("Distribution of Diabetes status within Gender")
axes[0, 0].set_xlabel("Gender")
axes[0, 0].set_ylabel("Count")

# Chart 2: Distribution of a Hypertension
sns.countplot(x="hypertension", data=data, hue = 'diabetes', ax=axes[0, 1])
axes[0, 1].set_title("Distribution of Diabetes status within Hypertension")
axes[0, 1].set_xlabel("Hypertension")
axes[0, 1].set_ylabel("Count")

# Chart 3: Distribution of heart disease
sns.countplot(x="heart_disease", hue = 'diabetes',data=data, ax=axes[1, 0])
axes[1, 0].set_title("Distribution of Diabetes status within Heart Disease")
axes[1, 0].set_xlabel("Heart Disease")
axes[1, 0].set_ylabel("Count")

# Chart 4: Distribution of smoking history
sns.countplot(x="smoking_history", data=data, hue = 'diabetes', ax=axes[1, 1])
axes[1, 1].set_title("Distribution of Diabetes status within Smoking History")
axes[1, 1].set_xlabel("Smoking History")
axes[1, 1].set_ylabel("Count")

plt.tight_layout()
plt.show()
```



```
# Distribution of numeric features
# Create subplots
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 8))
numeric_features = ['age', 'bmi', 'HbA1c_level', 'blood_glucose_level']
for i, feature in enumerate(numeric_features):
    row = i // 2
    col = i % 2

    sns.histplot(data[feature], ax=axes[row, col])
    axes[row, col].set_title(f'{feature}')


plt.tight_layout()
plt.show()
```



```
#check for duplicates
duplicates = data[data.duplicated(keep=False)]
if not duplicates.empty:
  print("Duplicate rows found:")
  print(duplicates)
else:
  print("No duplicates found.")
```

```
Duplicate rows found:
       gender   age  hypertension  heart_disease smoking_history    bmi  \
1      Female  54.0             0              0         No Info  27.32
10     Female  53.0             0              0           never  27.32
14     Female  76.0             0              0         No Info  27.32
18     Female  42.0             0              0         No Info  27.32
41       Male   5.0             0              0         No Info  27.32
...       ...   ...           ...            ...             ...    ...
99980  Female  52.0             0              0           never  27.32
99985    Male  25.0             0              0         No Info  27.32
99989  Female  26.0             0              0         No Info  27.32
99990    Male  39.0             0              0         No Info  27.32
99995  Female  80.0             0              0         No Info  27.32

       HbA1c_level  blood_glucose_level  diabetes
1              6.6                   80         0
10             6.1                   85         0
14             5.0                  160         0
18             5.7                   80         0
41             6.6                  130         0
...            ...                  ...       ...
99980          6.1                  145         0
99985          5.8                  145         0
99989          5.0                  158         0
```

```
99990          6.1                  100        0
99995          6.2                   90        0

[6939 rows x 9 columns]
```

```
data = data.drop_duplicates()
data.duplicated().sum()
```

```
0
```

```
df_encoded = pd.get_dummies(data, columns=['gender', 'smoking_history'])
df_encoded.head()
```

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

```
columns = ['gender_Female', 'gender_Male',
       'smoking_history_No Info', 'smoking_history_current',
       'smoking_history_ever', 'smoking_history_former',
       'smoking_history_never', 'smoking_history_not current']
df_encoded[columns] = df_encoded[columns].astype(int)
```

```
#scaling numeric values
columns = ['age', 'bmi', 'HbA1c_level', 'blood_glucose_level']
scaler = StandardScaler()
df_encoded[columns] = scaler.fit_transform(df_encoded[columns])
df_encoded.head()
```

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabetes	gender_
0	1.700840	0	1	-0.314947	0.994563	0.043554	0	
1	0.543372	0	0	-0.000216	0.994563	-1.423096	0	
2	-0.614096	0	0	-0.000216	0.155970	0.483549	0	
3	-0.257952	0	0	-0.572051	-0.496269	0.410216	0	
4	1.522768	1	1	-1.061141	-0.682623	0.410216	0	

```
X = df_encoded.drop('diabetes', axis = 1)
y = df_encoded.diabetes
```

```
#split the X and y into train and test
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3, random_state = 42)
```

```
y_test.value_counts()
```

```
0    26267
1     2577
Name: diabetes, dtype: int64
```

```
from imblearn.over_sampling import SMOTE
smote = SMOTE(sampling_strategy = 'auto', random_state = 42)
X_train_resampled, y_train_resampled =smote.fit_resample(X_train, y_train)
```

```
classifiers = [
    ("Logistic Regression", LogisticRegression()),
    ("Decision Tree", DecisionTreeClassifier()),
    ("Random Forest", RandomForestClassifier())
]
```

```
# Iterate through classifiers, fit, and evaluate them
for name, classifier in classifiers:
    classifier.fit(X_train_resampled, y_train_resampled)
    y_pred = classifier.predict(X_test)

    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Model: {name}")
    print(f"Accuracy: {accuracy:.2f}")

    # Display classification report
    report = classification_report(y_test, y_pred)
    print(f"Classification Report:\n{report}")

    # Display confusion matrix
    confusion = confusion_matrix(y_test, y_pred)
    print(f"Confusion Matrix:\n{confusion}")
    print("=" * 50)
```

```
Model: Logistic Regression
Accuracy: 0.88
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.88      0.93     26267
           1       0.43      0.89      0.58      2577

    accuracy                           0.88     28844
   macro avg       0.71      0.89      0.76     28844
weighted avg       0.94      0.88      0.90     28844

Confusion Matrix:
[[23232  3035]
 [  284  2293]]
==================================================
Model: Decision Tree
Accuracy: 0.94
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97     26267
           1       0.67      0.75      0.71      2577

    accuracy                           0.94     28844
   macro avg       0.82      0.86      0.84     28844
weighted avg       0.95      0.94      0.95     28844

Confusion Matrix:
[[25314   953]
 [  644  1933]]
==================================================
Model: Random Forest
Accuracy: 0.96
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98     26267
           1       0.76      0.75      0.76      2577

    accuracy                           0.96     28844
   macro avg       0.87      0.86      0.87     28844
weighted avg       0.96      0.96      0.96     28844

Confusion Matrix:
[[25652   615]
 [  640  1937]]
==================================================
```

✓ 0s completed at 1:24 AM ● ×